

PEGASUS: Physically Enhanced Gaussian Splatting Simulation System for 6DoF Object Pose Dataset Generation

Lukas Meyer^{1,†}, Floris Erich², Yusuke Yoshiyasu², Marc Stamminger¹, Noriaki Ando² and Yukiyasu Domaie²



Fig. 1: Representative scenes generated by *PEGASUS*. By separately reconstructing objects and environment with Gaussian Splatting and connecting them to a physics engine a vast variety of scenes can be generated utilizing novel view synthesis. At each snapshot, multiple data points such as RGB images, segmentation masks, depth maps, 2D/3D bounding boxes, and object poses can be extracted.

Abstract—We introduce **Physically Enhanced Gaussian Splatting Simulation System (*PEGASUS*) for 6DoF object pose dataset generation**, a versatile dataset generator based on 3D Gaussian Splatting. Environment and object representations can be easily obtained using commodity cameras to reconstruct with Gaussian Splatting. *PEGASUS* allows the composition of new scenes by merging the respective underlying Gaussian Splatting point cloud of an environment with one or multiple objects. Leveraging a physics engine enables the simulation of natural object placement within a scene through interaction between meshes extracted for the objects and the environment. Consequently, an extensive amount of new scenes - static or dynamic - can be created by combining different environments and objects. By rendering scenes from various perspectives, diverse data points such as RGB images, depth maps, semantic masks, and 6DoF object poses can be extracted. Our study demonstrates that training on data generated by *PEGASUS* enables pose estimation networks to successfully transfer from synthetic data to real-world data. Moreover, we introduce the *Ramen* dataset, comprising 30 Japanese cup noodle items. This dataset includes spherical scans that capture images from both the object hemisphere and the Gaussian Splatting reconstruction, making them compatible with *PEGASUS*.

I. INTRODUCTION

Robotic manipulation in changing environments with various novel objects pose significant challenges. Tasks such as object detection, segmentation, and pose estimation require substantial training data to adapt to new scenarios. The creation of suitable datasets can be achieved either

by annotating real-world data (e.g., YCB-V [1], HOPE [2], LINEMOD [3]) or by utilizing synthetic datasets (e.g., Fallen Things (FAT) [4]). NDDS [5] and BlenderProc [6] synthetically generate domain-specific datasets, facilitating the straightforward insertion of modeled object assets into synthetic environments. However, training on synthetically generated datasets may be compromised by a lack of realism and the time-consuming process of creating detailed models for specific environments and objects.

To address these limitations and reduce the reality gap, we make use of advanced novel view synthesis techniques. The rendering quality of novel view synthesis methods, such as Neural Radiance Fields (NeRF) [7] have steadily increased over the past years [8]. With 3D Gaussian Splatting (3DGS) [9], a real-time method using explicit representation, it becomes possible to create more realistic scenes. By leveraging this explicit representation, we can build a modular pipeline by combining multiple Gaussian splatting point clouds for dataset generation. This approach allows for the rapid and flexible generation of high-quality, realistic synthetic data, which can significantly enhance training and adaptation for robotic manipulation tasks in diverse environments.

In this work, we introduce *PEGASUS*, a physically enhanced Gaussian Splatting simulation environment, designed to create innovative datasets for 6DoF object pose estimation. By utilizing 3DGS, it allows us to close or at least narrow the gap between synthetic training data and real-world applications. Furthermore, it is very simple to generate custom assets: new models are obtained by scanning real-world objects and environments and reconstructing them in an automated pipeline. By combining 3DGS assets a comprehensive dataset can be created to fine-tune object pose estimation networks for desired operating environments.

In *PEGASUS*, we separately create the environment and objects with Gaussian Splatting and simulate the interaction of both elements using a physics engine. In this matter, we

¹Visual Computing Erlangen, Friedrich-Alexander-Universität Erlangen-Nürnberg-Fürth, Germany. E-Mail: lukas.meyer@fau.de

²National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

[†]This work was conducted during an internship at the National Institute of Advanced Industrial Science and Technology.

*This paper is one of the achievements of joint research with and is owned copyrighted material of ROBOT Industrial Basic Technology Collaborative Innovation Partnership. This research has been supported by the New Energy and Industrial Technology Development Organization (NEDO), under the project ID JPNP20016.

can render novel views for RGB images, semantic masks, depth maps, and metadata such as the object pose and 2D/3D bounding boxes. By extracting the data in the *Benchmark for 6D Object Pose Estimation* (BOP) data format [10] it can be easily used to train pose estimation networks and other network types.

Our experiments demonstrate that the pose estimation network, Deep Object Pose (DOPE) [11] when trained on *PEGASUS*-generated dataset, can operate a grasping task with Universal Robots *UR5* based on our dataset and successfully shows synthetic to real transfer. In summary, we make the following contributions:

- *PEGASUS*: A dataset generation tool for photo-realistic 6DoF object pose estimation, utilizing 3D Gaussian Splatting. The tool’s code has been made open-source¹.
- *Ramen Dataset*²: A comprehensive collection of over 30 products, featuring images, COLMAP reconstructions, and 3D Gaussian Splatting reconstructions.
- *PEGASET* Dataset: A collection of scanned environments and 21 re-scanned objects from YCB-V.

II. RELATED WORK

Neural Radiance Fields: Novel view synthesis has recently become a popular research topic that studies techniques for generating novel views of captured scenes. Various approaches, including implicit representations, voxels, and point clouds, are used for scene representation.

Neural Radiance Fields (NeRF) [7] build a continuous implicit representation by optimizing a Multi-Layer Perceptron (MLP) through volumetric rendering. This process encodes information within the MLP weights, requiring network queries for every spatial point to extract color and density data. Consequently, editing NeRFs involves retraining, which is computationally expensive.

InstantNGP [19] employs a multi-resolution hash grid for spatial information storage, resulting in much faster access and thus training and rendering time. However, scene modifications necessitate altering the grid structure, as demonstrated by NeRFShop [20]. NeRFShop enables volumetric manipulation, yet this involves interactive region selection and manual object deformation, and it is not conducive to automation. CLIP-NeRF [22] integrates CLIP [23] to manipulate the shape and appearance of NeRF by training a deformation network, which limits its suitability for rapid editing.

Alternatives are non-volumetric representations, based on surfaces or points. Such representations are not trained from scratch, but start with a point cloud reconstructed using standard structure-from-motion methods [18], SLAM or LIDAR. ADOP [24] and TRIPS [25] fall into this class, they utilize point cloud-based radiance fields, enhancing points with neural features. These approaches achieve high rendering quality and fast rendering time, but the optimization of the neural features and the rendering network is still time-consuming.

¹*PEGASUS* Code: <https://github.com/meyerls/PEGASUS>

²*Ramen-Dataset*: https://meyerls.github.io/pegasus_web

3D Gaussian Splatting (3DGS) [9] also starts with a reconstructed point cloud, but renders these as semi-transparent Gaussian splats. Such splats are a powerful rendering primitive, which makes it possible to optimize them directly, without requiring neural parameters and a final neural rendering network. 3DGS show high rendering quality at a very high rendering speed. The specialized differential Gaussian rasterization pipeline facilitates straightforward manipulation (transformation, insertion, deletion) of the underlying point cloud. This enables an easy and rapid combination of different reconstructions, a crucial aspect of *PEGASUS*, which is why we 3DGS as the basis for our approach.

Dataset Generation: Existing datasets are typically categorized as either synthetic or real-world. Synthetic datasets, like BlenderProc [6] or NVIDIA Deep Learning Dataset Synthesizer (NDDS) [5], offer the advantage of generating numerous unique scenes. However, they face challenges in asset modeling and achieving photorealistic rendering, often resulting in a domain gap when applied to real-world scenarios. To mitigate this, physically-based rendering techniques are employed, incorporating complex lighting effects such as scattering, refraction, and reflection, to enhance realism [28].

Conversely, creating real-world datasets, such as YCB-V [1], is a labor-intensive process that requires meticulous annotation, often prone to human error. Capturing a wide variety of scenes to ensure dataset variance further adds to the complexity. Despite these challenges, real-world datasets generally offer better generalization for deep learning applications than their synthetic counterparts.

NeuralLabeling [27] offers to directly annotate Neural Radiance fields and precisely extract the underlying object structures. To create a large dataset it is still time-consuming to generate a vast variety of scenes.

PEGASUS adopts a hybrid approach, combining the strengths of both synthetic and real-world datasets. Leveraging the modularity of synthetic data, it allows for the generation of new scenes through the combination of scanned objects and diverse environments, leading to a multitude of data points. Additionally, *PEGASUS* employs novel view synthesis techniques to render photorealistic scenes that are practically indistinguishable from real-world data.

III. PREREQUISITES

A. Gaussian Splatting

3D Gaussian Splatting [9] is an efficient method for performing novel view synthesis from captured scenes. It utilizes an unstructured, discrete representation in the form of a point cloud, which offers significant flexibility for modifying and manipulating inherent geometry.

The input for Gaussian Splatting comprises a set of images capturing a static scene or object, corresponding poses, camera intrinsics, and a sparse point cloud, which are typically generated using Structure from Motion (SfM) [18]. This sparse point cloud is then transformed into a more complex 3D Gaussian Splatting point cloud, denoted as $\mathbf{P}_{GS} = \{\mathbf{P}_\mu, \mathbf{P}_\Sigma, \mathbf{P}_\alpha, \mathbf{P}_f\}$. Each point in this cloud, represented by \mathbf{x} (also interpretable as the mean position μ of the

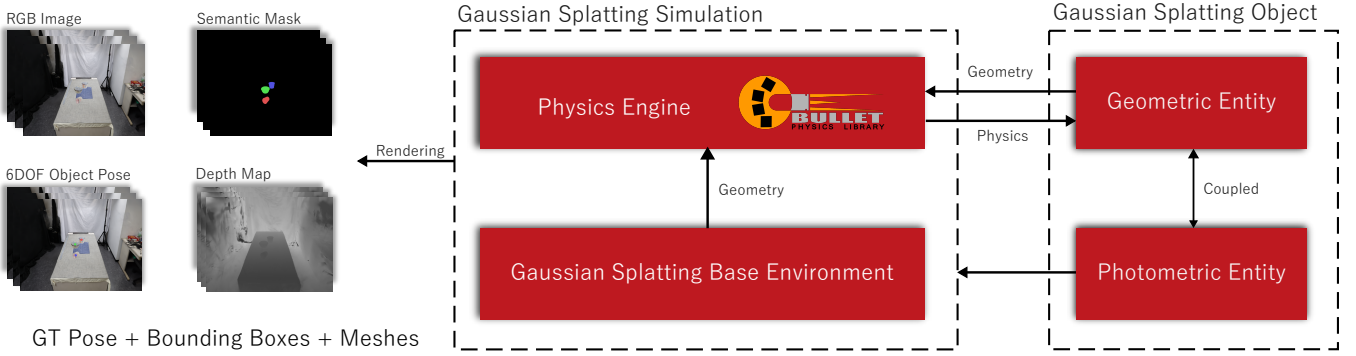


Fig. 2: Pipeline of the *PEGASUS* dataset generator. The 3DGS base environment (see Section IV-A.1) comprises both the 3DGS reconstruction and a mesh reconstructed from its point cloud. The 'Object' includes the 3DGS representation of the object (discussed as the photometric entity in Section IV-A.2) and a low-poly mesh of the same object (covered as the geometric entity in Section IV-A.2). By utilizing the mesh of the base environment and the object entity, an arbitrary number of objects can be simulated in the physics engine (refer to Section IV-A.3), facilitating realistic and random placement of the objects within the scene. When the trajectories of the objects are applied to the photometric instances of the environment and the object, we are capable of rendering dynamic and static scenes from various viewpoints and time steps. These data are then saved in the BOP data format [10].

Gaussian), is associated with a covariance matrix Σ , an opacity value α , and a set of spherical harmonic coefficients \mathbf{f} , which are used for directional appearance coloring.

The 3D Gaussians have to be projected onto the 2D image plane to optimize the parameters of the Gaussian Splatting point cloud. Therefore a differential tile-based Gaussian rasterization pipeline proposed by [9] is utilized. Each Gaussian is characterized by

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x})^T \Sigma^{-1}(\mathbf{x})} \quad (1)$$

where Σ is a full 3D covariance matrix defined in world space and centered at the point means μ . By projecting the 3D Gaussians back onto the 2D image the covariance matrix in image space [15] is computed through:

$$\Sigma' = \mathbf{J}\Sigma\mathbf{W}^T\mathbf{J}^T. \quad (2)$$

\mathbf{W} is defined as the transformation matrix from world to camera space and \mathbf{J} is the Jacobian of the affine approximation of the projective transformation.

After mapping the 3D Gaussians onto the 2D image plane the alpha-blended rendering is performed for each pixel in front-to-back depth order to evaluate the final color and alpha values [9]. The blending of the N ordered sample points in a pixel is computed by:

$$\mathbf{C} = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \quad (3)$$

\mathbf{c}_i and α_i are defined as color and opacity of the i th Gaussian.

B. 6-DOF Manipulation of Gaussian Splatting

The manipulation of Gaussian Splatting benefits from its underlying explicit representation. The appearance of the Gaussian point cloud \mathbf{P}_{GS} is defined by the Gaussian's \mathbf{P}_{Σ} , 3D mean value \mathbf{P}_{μ} , opacity values \mathbf{P}_{α} and coefficients $\mathbf{P}_{\mathbf{f}}$ of the spherical harmonics. For manipulating a Gaussian point cloud only \mathbf{P}_{μ} , \mathbf{P}_{Σ} and $\mathbf{P}_{\mathbf{f}}$ are relevant as the scalar opacity values in \mathbf{P}_{α} do not change if a transformation is applied. For

applying a transformation matrix $\mathbf{T} = [\mathbf{R}|\mathbf{t}]$ the translational and rotational part has to be considered separately.

The translational part \mathbf{t} is a straightforward operation. This involves applying a translation vector $\mathbf{t}_{\Delta} = (x_{\Delta}, y_{\Delta}, z_{\Delta})^T$ to the mean values $\mathbf{x} = (x, y, z)^T$ of the Gaussians. The remaining parts of the Gaussian point cloud (such as \mathbf{P}_{Σ} and $\mathbf{P}_{\mathbf{f}}$) are not affected.

Rotating a Gaussian Splatting point cloud is more complex. Here, a rotation matrix \mathbf{R} has to be applied on the points \mathbf{P}_{μ} , the covariance matrices \mathbf{P}_{Σ} and the coefficients $\mathbf{P}_{\mathbf{f}}$ of the spherical harmonics. For \mathbf{P}_{μ} and \mathbf{P}_{Σ} the rotation is directly applied to their corresponding point clouds.

To also obtain the identical view-dependent effects as for the original scene one has to apply a rotation also to the spherical harmonics [35]. This is commonly done by not rotating the base functions of the spherical harmonics but rather their coefficients [35]. In [37] a method is proposed to decompose the rotation matrix \mathbf{R} into its Euler angles and build a block diagonal sparse matrix [35] to rotate every band individually. Due to the rotational in-variance of spherical harmonics, this rotation is lossless. For a detailed explanation please refer to [37], [35].

IV. METHODOLOGY

The core principle of *PEGASUS* is the separate consideration of the environment and individual objects. By situating a set of objects within multiple environments, a vast set of scenes can be created. The integration of the physics engine *PyBullet* [39] into *PEGASUS* enables the simulation of natural object placement in scenes and the creation of dynamic scenes within the Gaussian Splatting Simulation Environment.

A. Gaussian Splatting Simulation Environment

The core pipeline of *PEGASUS*, as illustrated in Fig. 2, is composed of several distinct blocks. Below, we detail the essential components of this pipeline: the base environment, the creation of Gaussian Splatting objects, and the integration of a physics engine into our simulation setup. Together, these blocks form the backbone of the *PEGASUS* pipeline, enabling

the creation of complex, multi-modal datasets that closely mimic real-world conditions.

1) *Base Environment*: The base environment serves as the foundational construct for building the actual scene in *PEGASUS*. To create this, we recorded 9 different planar scenes using a DSLR camera, capturing between 100 and 150 images per scene. We then used SfM to recover the set of poses and a sparse point cloud $\mathbf{P}_{\text{sparse}}$.

The scenes are then automatically scaled to obtain a true-to-scale metric reconstruction by placing an ArUco marker in the scene [30] and afterward aligning the planar area to the center so that the \mathbf{z} vector points upwards.

Subsequently, we performed Gaussian Splatting on the scene using the default parameters suggested by 3DGS [9]. Towards the end of this process, we extracted the plain point cloud by obtaining the mean value of each Gaussian splat. To integrate with the physics engine, we converted this point cloud into a mesh. For this purpose, we employed the alpha shape algorithm [33] as a mesh recovery technique, transforming the 3DGS-extracted points into a geometric mesh.

2) *Gaussian Splatting Object*: The reconstruction of objects in *PEGASUS* follows a procedure similar to that of the base environment. We utilize an Ortery scanning system [41] for image acquisition. Detailed information about this process and the various types of objects are discussed in Section IV-C. Below, we introduce the concept of the Gaussian Splatting object, which comprises two main components: the photometric and geometric entities.

Photometric Entity: The photometric entity is crucial for rendering objects using Gaussian Splatting. We start with a spherical, sparse, and metrically reconstructed Structure from Motion (SfM) model of the object as input. Using Gaussian Splatting with the same settings as in the base environment setup, we generate this photometric entity. It can then be integrated into the simulation, enabling simultaneous rendering of both the base environment and the object.

Geometric Entity: For the geometric entity, we start with the colored point cloud extracted from the Gaussian Splatting reconstruction. The point cloud is initially cleaned by removing outliers, and then the alpha shape algorithm [33] is applied for mesh reconstruction. To achieve a smoother surface, we use Laplacian smoothing [31]. The geometric entity is stored as a low-polygon triangle mesh, optimizing computational efficiency when simulating this mesh with the physics engine.

3) *Physics Engine*: We selected PyBullet [39] as our lightweight physics engine. Within the simulation, the environment mesh is integrated as a static component. For the objects, we determine an appropriate height to drop varying quantities (user-selected) into the scene, simulating natural object placement. Throughout the simulation, we track and extract the orientation and translation of each object, represented as a quaternion and a translation vector, respectively. *PEGASUS* is thus equipped to simulate both static and dynamic scenes, offering a comprehensive range of possibilities for scene creation and analysis.

B. PEGASUS Dataset Generation

To create a dataset using Gaussian Splatting with *PEGASUS*, we include Gaussian Splatting base environments and Gaussian Splatting objects. This approach, enhanced by physical placement techniques, allows us to generate an unlimited number of unique scenes. To generate a new scene, we simulate the trajectory using our physics engine, apply transformations to each Gaussian Splatting object, and render the scene with the Gaussian Splatting rasterizer [9].

To render the scene from various viewpoints, we randomly select a set of ground truth poses and create a trajectory by interpolating between these poses. Lastly, we save the raw data, including rendered RGB images, depth maps, segmentation maps of the silhouette and visibility masks, 2D/3D bounding boxes as well as the transformation matrices from object to world and world to camera coordinate systems. This data is formatted according to the popular BOP-dataset format [10]. We want to emphasize that *PEGASUS* is also capable of rendering dynamic scenes, which broadens the potential applications of the generated data. It is worth noting that this approach can be easily extended with custom data. To incorporate a new environment, one only needs to record a set of images and convert them into a 3DGS instance. The same process applies to any object; a simple scan is enough to compute the photometric and geometric properties required for integration into *PEGASUS*. A set of images extracted from the dataset generator is shown in Fig. 3.

C. Data Set

The focus of our research is on the development of robotic systems in the service sector to support personnel in retail. With Japan having one of the highest population densities, it boasts a considerable number of 24-hour convenience stores known as *konbini* [40]. Introducing robots into these compact retail spaces can prove to be a valuable asset for tasks like restocking products and efficiently managing inventory. Therefore we focused on the specific product group of ramen noodles gathered in the *Ramen* dataset. The dataset comprises of 30 varieties of cup noodles, readily available in most mini markets. A comprehensive summary of all products is illustrated in Fig. 4.

For recording the objects, we employed the *3D PhotoBench 280* from Ortery [41], a commercial 360° turntable system, alongside the *3D MultiArm 2000* [42] camera system. This setup enabled synchronous image acquisition from five different angles capturing 150 images for each hemisphere and applying automatic background removal. The process is heavily inspired by *Neural Scanning* [26].

The initial step involved scanning a planar calibration board with a feature-rich surface and an ArUco marker of known size. This process, aided by COLMAP [18], allowed for precise pose computation. Subsequently, the scene (including poses and point cloud) was scaled to achieve a metric reconstruction [30] and aligned to the xy plane ensuring the normal vector of the visible plane faced the positive \mathbf{z} -direction. The poses obtained from the board were repurposed as a calibration reconstruction. For each cup noodle



Fig. 3: Gallery of data generated by *PEGASUS*. It shows scenes generated with 9 different base environments and an arbitrary combination of the 30 elements from the *Ramen* dataset and from the 21 YCB objects [32] from the YCB-V dataset.

product, the poses from the upper hemisphere’s calibration reconstruction were utilized. The sparse point cloud was then recomputed by triangulating the matched feature points from the product images. To capture photometric information about the bottom part, we scanned the bottom hemisphere of the flipped product symmetrically. However, it was not possible to reuse the calibration target for the bottom hemisphere, as the flipping of the object altered the pose locations. Our methodology involved registering the bottom images into the existing top reconstruction, resulting in approximately 270 registered images per product. Each object was reconstructed

using 3DGS and a low-poly mesh was extracted from the point cloud (as detailed in section IV-A.2). To integrate these meshes into the physics engine, we store the information for every object, including environment objects, in a Unified Robotics Description Format (URDF) file, documenting both visual and collision model information.

In addition to the *Ramen* dataset, we provide *PEGASET* as a second dataset. It comprises a selection of 21 objects from the well-known YCB-V Dataset [1]. By including these fundamental objects, we aim to encourage broader adoption by the robotics community.

V. EXPERIMENTS

This section presents experiments demonstrating the successful use of Universal Robots *UR5* to execute real-world pick-and-place operations on data generated by *PEGASUS*.

We selected the Deep Object Pose (DOPE) [11] network structure for our experiments. For dataset generation, three distinct data sets were created, each comprising 60,000 images featuring a single cup noodle, set in three different environments. In total, we generated 2,000 unique scenes with 30 images per scene, captured from various perspectives. The generation process took 6 hours on a laptop with an Intel i9 12th Gen CPU and NVIDIA RTX 3080 Ti (Mobile) GPU. Regarding training, we utilized the default hyper-parameters of DOPE and trained the network for 15 epochs. The *UR5* was configured to accurately pick the center of the cup noodle and place it into a basket. Our experiments successfully demonstrated the capability of the robot to sequentially pick up 10 out of 10 cup noodles in a row, as well as to grasp various types of cup noodles.



Fig. 4: 30 Objects recorded for our *Ramen* dataset of common Japanese cup noodles available at most supermarkets.

VI. LIMITATIONS

Our method, while effective, is not without its limitations. One significant shortfall is the absence of realistic shadow rendering in our system. Consequently, incorporating shadow maps or screen space ambient occlusion represents a natural and necessary next step in our development process. Additionally, when placing objects within an environment, our current approach does not account for re-lighting, scattering, refraction, or reflection. This omission can result in scenes that appear somewhat unnatural. Another challenge we faced involves scanning texture-less environments, which often leads to a noisy Gaussian splatting reconstruction. This noise manifests as large Gaussian splats that may overlap or interfere with objects, potentially causing visual artifacts. Addressing these issues is crucial for enhancing the realism and visual fidelity of our rendered scenes.

VII. CONCLUSION

We have introduced *PEGASUS*, a versatile dataset generator designed to enhance accuracy and quality in object pose estimation. Alongside *PEGASUS*, we present the *Ramen* dataset, which includes over 30 diverse products. The dataset generator adeptly creates photorealistic renderings, semantic masks, and depth maps, and captures the object pose. *PEGASUS* is specifically engineered to generate domain-specific data sets, aiding in the fine-tuning of neural networks that extend beyond mere pose estimation tasks.

To further empower *PEGASUS*, it is crucial to accumulate a more extensive collection of environments and objects. This expansion is key to evolving toward a more generalizable dataset generator. Another intriguing avenue for enhancement involves applying augmentations directly to the objects or their environments. Techniques like diffusion models, such as *GaussianDreamer* [17] or *Rosie* [16], can be employed to alter the shape and appearance of objects and environments, offering a new dimension of flexibility in dataset generation.

Exploring the scanning of more complex scenes using LIDAR-based 3DGS presents an exciting opportunity. This approach could significantly enhance the realism of the environments integrated into our system. By leveraging LIDAR's detailed spatial data, we can capture intricate scene details and textures, paving the way for even more lifelike and accurate representations in our dataset generation process.

ACKNOWLEDGMENT

We extend our sincere gratitude to **Abdullah Mustafa** for his valuable feedback and to **Toshio Ueshiba** for his extensive expertise in UR5.

REFERENCES

- [1] Y. Xiang *et al.*, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *CoRR*, 2017.
- [2] S. Tyree *et al.*, "6-DoF Pose Estimation of Household Objects for Robotic Manipulation: An Accessible Dataset and Benchmark," *IROS*, 2022.
- [3] S. Hinterstoisser *et al.*, "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes," *ACCV*, 2012.
- [4] J. Tremblay *et al.*, "Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation," *CoRR*, 2018.
- [5] T. To *et al.*, "NDDS: NVIDIA Deep Learning Dataset Synthesizer," 2018.
- [6] M. Denninger *et al.*, "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering," *Journal of Open Source Software*, 2023.
- [7] B. Mildenhall *et al.*, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *ECCV*, 2020.
- [8] K. Gao, *et al.*, "NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review," *ArXiv*, 2023
- [9] B. Kerbl *et al.*, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *SIGGRAPH*, 2023.
- [10] M. Hodan *et al.*, "BOP: Benchmark for 6D Object Pose Estimation," *ECCV*, 2018.
- [11] J. Tremblay *et al.*, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," *CoRL*, 2018.
- [12] G. Kopanas *et al.*, "Point-Based Neural Rendering with Per-View Optimization," *Computer Graphics Forum*, 2021.
- [13] G. Kopanas *et al.*, "Neural Point Catacaustics for Novel-View Synthesis of Reflections," *ACM Transactions on Graphics*, 2022.
- [14] Z. Yang *et al.*, "Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction," *ArXiv*, 2023.
- [15] M. Zwicker *et al.*, "EWA volume splatting," *IEEE Visualization*, EWA volume splatting, 2001.
- [16] Tianhe Yu *et al.*, "Scaling Robot Learning with Semantically Imagined Experience," *ArXiv*, 2023.
- [17] Taoran Yi *et al.*, "GaussianDreamer: Fast Generation from Text to 3D Gaussian Splatting with Point Cloud Priors," *ArXiv*, 2023.
- [18] J. L. Schönberger and J. M. Frahm, "Structure-from-Motion Revisited," *CVPR*, 2016.
- [19] T. Müller *et al.*, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *SIGGRAPH*, 2022.
- [20] C. Jambon *et al.*, "NeRFshop: Interactive Editing of Neural Radiance Fields". *13D*, 2023.
- [21] A. Yu *et al.*, "PlenOctrees for Real-time Rendering of Neural Radiance Fields," *CoRR*, 2021.
- [22] C. Wang *et al.*, "CLIP-NeRF: Text-and-Image Driven Manipulation of Neural Radiance Fields," *CVPR*, 2022.
- [23] A. Radford *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," *ICML*, 2021.
- [24] D. Rückert, L. Franke and M. Stamminger. *ADOP: Approximate Differentiable One-Pixel Point Rendering*. *CoRR*. 2021.
- [25] L. Franke *et al.*, "TRIPS: Trilinear Point Splatting for Real-Time Radiance Field Rendering," *Eurographics*, 2024.
- [26] F. Erich *et al.*, "Neural Scanning: Rendering and Determining Geometry of Household Objects Using Neural Radiance Fields". *SII*, 2023.
- [27] F. Erich *et al.*, "NeuralLabeling: A versatile toolset for labeling vision datasets using Neural Radiance Fields," *ArXiv*, 2023.
- [28] T. Hodan *et al.*, "Photorealistic Image Synthesis for Object Instance Detection". *ICIP*, 2019.
- [29] G. Pitteri *et al.*, "On Object Symmetries and 6D Pose Estimation from Images". *3DV*, 2019.
- [30] L. Meyer, *et al.*, "CherryPicker: Semantic Skeletonization and Topological Reconstruction of Cherry Trees," *CVPRW*, 2023.
- [31] A. Nealen *et al.*, "Laplacian Mesh Optimization," *GRAPHITE*, 2006.
- [32] B. Calli *et al.*, "Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols," *IEEE Robotics and Automation Magazine*, 2015.
- [33] H. Edelsbrunner and E. Mücke, "Three-dimensional alpha shapes". *ACM Transactions on Graphics*, 1994.
- [34] L. Mariga, *pyRANSAC-3D*, 2022, DOI: 10.5281/zenodo.7212567..
- [35] R. Green, "Spherical Harmonic Lighting: The Gritty Details," 2003.
- [36] W. Jarosz, "Efficient Monte Carlo Methods for Light Transport in Scattering Media," *Dissertation*, 2008.
- [37] Jan Kautz, Peter-Pike Sloan, and John Snyder. 2002. "Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics," *Eurographics*, 2002.
- [38] M. Geiger *et al.*, "Euclidean neural networks: e3nn," *GitHub*, 2022.
- [39] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016-2021.
- [40] Statista, *Number of convenience stores in Japan from 2013 to 2022*, 2023.
- [41] *Ortery 3D PhotoBench 280*, Ortery, 2023.
- [42] *3D MultiArm 2000*, Ortery, 2023.