

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

Lukas Meyer^{1,†}  , Floris Erich²  , Yusuke Yoshiyasu², Marc Stamminger¹, Noriaki Ando², Yukiyasu Domae²

¹Visual Computing, Friedrich-Alexander-Universität Erlangen-Nürnberg-Fürth, Germany

²Industrial CPS Research Center, National Institute of Advanced Industrial Science and Technology, Japan

lukas.meyer@fau.de



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 captures images from both object hemisphere and the Gaussian Splatting reconstruction, making them compatible with *PEGASUS*.

Index Terms—dataset generation, robotics, radiance fields, sim2real

I. INTRODUCTION

Numerous Western nations are contending with the challenges posed by shifting demographics [1], particularly the reduction in their working-age population [2]. Japan is grappling with a similar issue, experiencing a decrease in its overall population, prompting substantial investments in robotics across multiple industries like healthcare, manufacturing, and agriculture to maintain workforce stability [4].

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* [3]. Introducing robots into these compact retail spaces can prove to be a valuable asset for tasks like restocking products and efficiently managing inventory.

To apply deep learning approaches for object pose estimation most datasets are focused towards western style product. Examples of such datasets are YCB-V [11], HOPE [8], Fallen Things (FAT) [9] and LINEMOD [10]. For generating domain specific datasets the default approach would be to synthetically generate data with NDDS [6] or BlenderProc [5]. This approach allows for the straightforward insertion of modeled object assets into synthetic environments. However, a drawback is the presence of a domain gap when trained on synthetic generated dataset due to the lack of realism. We overcome this problem by using photorealistic rendering

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

by using novel view synthesis. An additional drawback is the time-consuming necessity of creating detailed models for specific environments and objects. On the other hand assets for *PEGASUS* can be simply obtained by scanning real world objects and environments by reconstructing them with novel view synthesis methods such as 3D Gaussian Splatting (3DGS) [12]. By combining 3DGS assets a comprehensive dataset can be created to fine tune object pose estimation networks for desired operating 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 leveraging advanced novel view synthesis techniques. In this framework 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 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 BOP data format [30] 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) [42], when trained on *PEGASUS*-generated dataset, can operate a grasping task with an UR5 based on our dataset and successfully shows synthetic to real transfer.

The contribution of this paper are:

- *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 objects (YCB-V).

II. RELATED WORK

We first briefly review the different types of representation used for radiance fields and discuss methods to generate synthetic datasets.

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) [20] 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, modifying NeRF involves adjusting these weights. CLIP-NeRF [24] integrates CLIP [25] to manipulate the shape and appearance of NeRF by training a deformation network, which limits its suitability for rapid editing.

InstantNGP [21] employs a multi-resolution hash grid for spatial information storage. Scene modifications necessitate altering the grid structure, as demonstrated by NeRFShop [22]. However, NeRFShop’s volumetric manipulation, which involves interactive region selection and manual object deformation, is not conducive to automation.

ADOP [26] and VET [27] utilize point cloud-based radiance fields, enhancing points with neural features. Despite their high-quality rendering and fast inference, these methods still rely on Multi View Stereo (MVS) for internal geometry, which is time consuming.

3DGS [12] introduces a novel approach combining point cloud rendering with Gaussian splats. Their 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*.

Dataset Generation: Existing datasets are typically categorized as either synthetic or real-world. Synthetic datasets, like BlenderProc [5] or NVIDIA Deep learning Dataset Synthesizer (NDDS) [6], 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 [31].

Conversely, creating real-world datasets, such as YCB-V [11], 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 [29] offers to directly annotate Neural Radiance Field’s 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. By 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

In this section we cover the prerequisites on 3D Gaussian Splatting (3DGS) and 6DoF manipulation on 3DGS point clouds.

A. Gaussian Splatting

3D Gaussian Splatting [12] is an efficient method for performing novel view synthesis from captured scenes. It

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

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

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

The 3D Gaussians have to be projected onto the 2D image plane in order to optimize the parameters of the Gaussian Splatting point cloud. Therefore a differential tile-based gaussian rasterization pipeline proposed by [12] 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 mean μ . By back projecting the 3D Gaussians onto the 2D image the covariance matrix in image space [16] is computed through:

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

\mathbf{W} is defined as the transformation matrix from world to camera space and \mathbf{J} being 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 [12]. 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 the 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}_f of the spherical harmonics. For manipulating a Gaussian point cloud only \mathbf{P}_μ , \mathbf{P}_Σ and \mathbf{P}_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 $\mathbf{t}_\Delta = (x_\Delta, y_\Delta, z_\Delta)^\top$ to the mean values $(x, y, z)^\top$ of the Gaussian's. The remaining

parts of the Gaussian point cloud (such as \mathbf{P}_Σ and \mathbf{P}_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}_f of the spherical harmonics. For \mathbf{P}_μ and \mathbf{P}_Σ the rotation is directly applied on their corresponding point clouds. If we want to obtain the same view dependent effects for the scene as the unrotated one, a rotation has to be applied to the spherical harmonics coefficients.

Spherical harmonics play a pivotal role in the domain of computer graphics, especially for rendering view-dependent color representation relative to a viewing angles denoted by θ (azimuth) and ϕ (elevation). Broadly speaking, they are akin to a 2D Fourier series expansion on the two-dimensional surface but defined on the surface of a sphere. The spherical harmonic base functions, represented by $Y_l^m(\theta, \phi)$, act as the orthonormal basis functions.

The real-valued spherical harmonic function can be expressed as:

$$c(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^l f_{lm} Y_{lm}(\theta, \phi). \quad (4)$$

Where c denotes the un-normalized color value for one color channel, and f_{lm} is the expansion coefficient for degree l and mode m .

To rotate the spherical harmonics, one does not directly apply the rotation to the basis function . Instead, the transformation is carried out on the coefficients [38].

For this purpose the Wigner D-matrix can be utilized [37], [38]. For a given rotation matrix \mathbf{R} , the rotated SH coefficients f'_{lm} can be expressed as:

$$f'_{lm} = \sum_{m'=-l}^l D_{mm'}^{(l)}(\mathbf{R}) f_{lm'} \quad (5)$$

Where $D_{mm'}^{(l)}(\mathbf{R})$ is the Wigner D-function for the desired rotation of degree l .

IV. METHODOLOGY

In this chapter, we introduce *PEGASUS* (Physical Enhanced Gaussian Splatting Simulation System), a system designed to generate novel, multi-modal datasets using Gaussian Splatting. This method allows for the extraction of multiple data points, as illustrated in Fig. 1.

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.

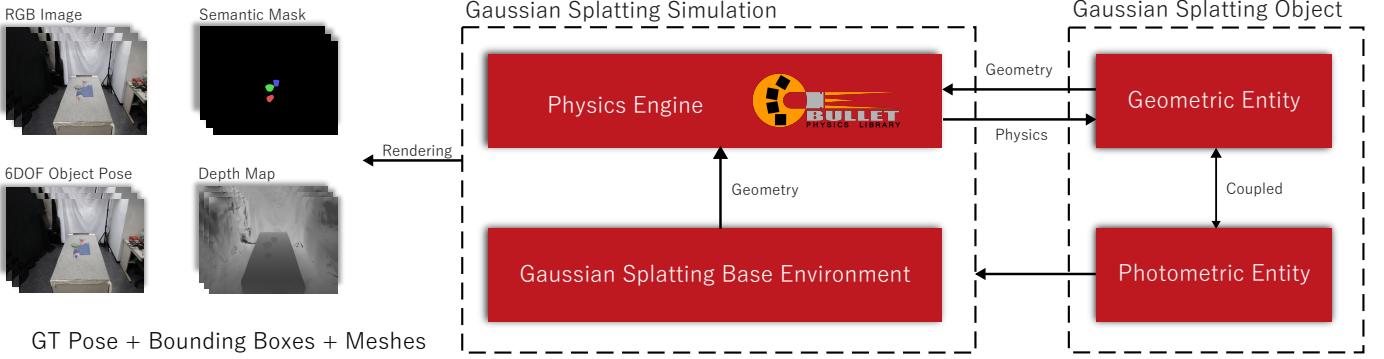


Fig. 2: Pipeline of the PEGASUS dataset generator. The 3DGS base environment (see Section IV-A1) comprises both the 3DGS reconstruction and a mesh reconstructed from its point cloud. The ‘Object’ includes the 3DGS representation of the object (discussed as the photometric entity in Section IV-A2) and a low-poly mesh of the same object (covered as the geometric entity in Section IV-A2). 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-A3), 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 [30].

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 10 different scenes using a DSLR camera, capturing 100 to 150 images for each scene. We then used Structure from Motion (SfM)[19] to recover the set poses \mathcal{E} and a sparse point cloud $\mathbf{P}_{\text{sparse}} \in \mathbb{R}^{N_{\text{sparse}} \times 3}$.

For true-to-scale metric reconstructions of indoor scenes, we followed the CherryPicker [33] approach, placing an aruco marker in the scene and scaling both the poses and the reconstruction according to the recovered scale factor. For outdoor scenes, we manually computed the scale factor.

To correctly align and center the scene in space (assuming planar scenes), we employed RANSAC [36] to fit a plane into the sparse point cloud. This process oriented the scene so that the \mathbf{z} vector pointed upwards, and the center of mass of the inlier points was positioned at the center of the model coordinate system.

Subsequently, we performed Gaussian Splatting of the scene using the default parameters suggested by 3DGS [12] over 30.000 iterations. 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 needed to convert this point cloud into a mesh. For this purpose, we employed the alpha shape algorithm [35] 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 [40] 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 take a spherical, sparse, and metrically reconstructed SfM model of the object as input. The Gaussian Splatting process, using the same settings as in the base environment, creates this entity. This photometric entity can then be integrated into the simulation, allowing 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 first cleaned to remove outliers, and then we use the alpha shape algorithm [35] for mesh reconstruction. To achieve a smoother surface, we apply Laplacian smoothing [34]. The geometric entity is stored as a low-polygon triangle mesh, optimizing the computational efficiency when simulating this mesh with the physics engine.

3) Physics Engine: We selected PyBullet [39] as our physics engine, renowned for its lightweight and versatile capabilities. 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



Fig. 3: Gallery of data generated by *PEGASUS*. It shows scenes generated with 10 different base environments and an arbitrary combination of the 30 elements from the *Ramen* dataset and from the 21 objects from the YCB-V dataset.

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 integrate the Gaussian Splatting base environment 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 [12].

In the context of 6DoF object pose estimation using this dataset generation tool, we extract the transformations of objects from the model coordinate system to the camera coordinate system. This process determines the pose of the objects within the scene. To render the scene from various viewpoints, we select a set of random ground truth poses and create a trajectory by interpolating between these poses.

Ultimately, we save the raw data, including rendered RGB images, depth maps, segmentation maps of the silhouette and visible masks, 2D/3D bounding boxes as well as the transformation matrices from object to world and world to camera coordinate system. This data is formatted according to the popular BOP-dataset format [30].

We want to emphasise 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 Gaussian Splatting 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

In this section, we discuss the *Ramen* dataset and outline the pipeline for processing each individual object. In an earlier work [28] a similar scanning system was used to create NeRF representations of household objects.

The *Ramen* dataset comprises over 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* [40], a commercial 360° turntable system, alongside the *3D MultiArm 2000* [41] camera system. This setup enabled synchronous image acquisition from five different angles capturing 150 images for each hemisphere and automatic background removal.

The initial step involved scanning a planar calibration board with a feature-rich surface and an ArUco marker of known



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

size. This process, aided by COLMAP [19], allowed for precise pose computation. Subsequently, the scene (including poses and point cloud) was scaled to achieve a metric reconstruction [33]. Scene alignment was conducted by fitting a plane into the point cloud using RANSAC [36], ensuring the normal vector of the visible plane faced the positive z -direction. The poses obtained from the calibration board were repurposed as a calibration reconstruction.

For each cup noodle 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 in a similar fashion. 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. The bottom row remained unregistered due to its texture-less appearance, however this did not seem to affect the quality of the reconstruction.

Each object was reconstructed using 3DGS and a low-poly mesh extracted from the point cloud (as detailed in section IV-A2). To integrate these meshes into the physics engine, we generated a URDF file for every object (including environment objects), documenting both visual and collision model information.

V. EXPERIMENTS

This section presents experiments demonstrating the successful application of data generated by *PEGASUS* in training a neural network. Additionally, it showcases the use of a UR5 robot to execute real-world pick-and-place operations.

We selected the Deep Object Pose (DOPE) [42] 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 robot 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.

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, 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 towards 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* [18] or *Rosie* [17], 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.

ACKNOWLEDGEMENT

We extend our sincere gratitude to **Abdullah Mustafa** for his valuable feedback and to **Toshio Uesiba** for his extensive expertise in UR5. This paper is one of the achievements of joint research with and is jointly 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.

REFERENCES

- [1] United Nations, "Shifting Demographics," 2023, Accessed on: Nov. 5, 2023.
- [2] T. Schneider and G. H. Hong, "Shrinkanomics: Policy Lessons from Japan on Aging," *Finance & Development*, 2020. Accessed on: Nov. 5, 2023.
- [3] Statista, "Number of convenience stores in Japan from 2008 to 2020," <https://www.statista.com/statistics/810901/japan-convenience-store-numbers/>, Accessed on: Nov. 1, 2023.
- [4] R. S. Jones, "The Japanese Economy: Strategies to cope with a shrinking and aging population", RIETI <https://www.rieti.go.jp/en/events/bbl/23020101.html>, Accessed on: Dec. 18, 2023.
- [5] M. Denninger *et al.*, "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering," *Journal of Open Source Software*, 2023.
- [6] T. To *et al.* *et al.*, "NDDS: NVIDIA Deep Learning Dataset Synthesizer," 2018.
- [7] B. Calli *et al.*, "Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols," *IEEE Robotics and Automation Magazine*, 2015.
- [8] S. Tyree *et al.*, "6-DoF Pose Estimation of Household Objects for Robotic Manipulation: An Accessible Dataset and Benchmark," *IROS*, 2022.
- [9] J. Tremblay, T. To and S. Birchfield. "Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation," *CoRR*, 2018.
- [10] S. Hinterstoisser *et al.*, "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes," *ACCV*, 2012.
- [11] Y. Xiang *et al.*, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *CoRR*, 2017.
- [12] B. Kerbl *et al.*, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *SIGGRAPH*, 2023.
- [13] G. Kopanas *et al.*, "Point-Based Neural Rendering with Per-View Optimization," *Computer Graphics Forum*, 2021.
- [14] G. Kopanas *et al.*, "Neural Point Catacaustics for Novel-View Synthesis of Reflections," *ACM Transactions on Graphics*, 2022.
- [15] Z. Yang *et al.*, "Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction," *ArXiv*, 2023.
- [16] M. Zwicker *et al.*, "EWA volume splatting," *IEEE Visualization*, EWA volume splatting, 2001.
- [17] Tianhe Yu *et al.*, "Scaling Robot Learning with Semantically Imagined Experience," *ArXiv*, 2023.
- [18] Taoran Yi *et al.*, "GaussianDreamer: Fast Generation from Text to 3D Gaussian Splatting with Point Cloud Priors," *ArXiv*, 2023.
- [19] J. L. Schönberger and J. M. Frahm, "Structure-from-Motion Revisited," *CVPR*, 2016.
- [20] B. Mildenhall *et al.*, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *ECCV*, 2020.
- [21] T. Müller *et al.*, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *SIGGRAPH*, 2022.
- [22] C. Jambon *et al.*, "NeRFshop: Interactive Editing of Neural Radiance Fields". *I3D*, 2023.
- [23] A. Yu *et al.*, "PlenOctrees for Real-time Rendering of Neural Radiance Fields," *CoRR*, 2021.
- [24] C. Wang *et al.*, "CLIP-NeRF: Text-and-Image Driven Manipulation of Neural Radiance Fields," *CVPR*, 2022.
- [25] A. Radford *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," *ICML*, 2021.
- [26] D. Rückert, L. Franke and M. Stamminger. *ADOP: Approximate Differentiable One-Pixel Point Rendering*. CoRR. 2021.
- [27] L. Franke, D. Rückert, L. Fink, M. Innmann and M. Stamminger. "VET: Visual Error Tomography for Point Cloud Completion and High-Quality Neural Rendering," *SIGGRAPH Asia*, 2023.
- [28] F. Erich *et al.*, "Neural Scanning: Rendering and Determining Geometry of Household Objects Using Neural Radiance Fields". *SII*, 2023.
- [29] F. Erich *et al.*, "NeuralLabeling: A versatile toolset for labeling vision datasets using Neural Radiance Fields," *ArXiv*, 2023.
- [30] M. Hodan *et al.*, "BOP: Benchmark for 6D Object Pose Estimation," *ECCV*, 2018.
- [31] T. Hodan *et al.*, "Photorealistic Image Synthesis for Object Instance Detection". *ICIP*, 2019.
- [32] G. Pitteri *et al.*, "On Object Symmetries and 6D Pose Estimation from Images". *3DV*, 2019.
- [33] L. Meyer, A. Gilson, O. Scholz, M. Stamminger. "CherryPicker: Semantic Skeletonization and Topological Reconstruction of Cherry Trees," *CVPRW*, 2023.
- [34] A. Nealen *et al.*, "Laplacian Mesh Optimization," *GRAPHITE*, 2006.
- [35] H. Edelsbrunner and E. Mücke, "Three-dimensional alpha shapes". *ACM Transactions on Graphics*, 1994.
- [36] L. Mariga, *pyRANSAC-3D*, October 2022, DOI: 10.5281/zenodo.7212567..
- [37] R. Green, "Spherical Harmonic Lighting: The Gritty Details," 2003.
- [38] W. Jarosz, "Efficient Monte Carlo Methods for Light Transport in Scattering Media," *Dissertation*, 2008.
- [39] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016-2021.
- [40] Ortery 3D PhotoBench 280, Ortery, 2023.
- [41] 3D MultiArm 2000, Ortery, 2023.
- [42] J. Tremblay *et al.*, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," *CoRL*, 2018.