Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Master's Thesis

Simulation-Based Dataset Generation for Benchmarking Uncertainty Quantification of Object Pose Estimation

Sathwik Panchangam

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Sebastian Houben
Prof. Dr. Nico Hochgeschwender
M.Sc. Deebul Nair

April 2025

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work. The report was, in part, written with the help of the AI assistant Grammarly, as described in the appendix. I am aware that content generated by AI systems is no substitute for careful scientific work, which is why all AI-generated content has been critically reviewed by me, and I take full responsibility for it. Parts of this work belongs to the Research and Development Project submitted to Hochschule Bonn-Rhein-Sieg:

- S. Panchangam, D. Nair, and N. Hochgeschwender, "Benchmarking uncertainty estimation of deep learning models using synthetic dataset," Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany, Tech. Rep., Jan. 2023, Research and Development Project Report [1]
- D. Nair, S. Panchangam, M. A. Olivares-Mendez, and N. Hochgeschwender, "Embodied runtime monitoring of learning-enabled robot perception components," in 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE). IEEE, 2024, pp. 2983–2989 [2].

Date	Sathwik Panchangam

Abstract

The application of deep learning in high-risk domains such as robotic manipulation, vision, autonomous navigation, and healthcare has increased the need for the DNN models to provide well-calibrated and trustworthy predictions. Despite their success across various perception tasks including 6D object pose estimation, deep learning models struggle to provide reliable predictions under real-world complexities like varying illumination, occlusions, and novel background textures. The lack of comprehensive benchmarks that evaluate model robustness against such unseen data variations hinders the effective assessment and comparison of deep neural network (DNN) performance, particularly concerning the crucial aspect of prediction uncertainty. While data augmentation partially addresses this gap, generating diverse variations for factors like object deformation, distance, and background textures remains challenging.

To overcome these limitations, this thesis introduces a user-friendly dataset generation tool, using the python API (Bpy) of Blender3D software. The tool is capable of generating diverse synthetic datasets tailored for benchmarking DNNs in 6D object pose estimation task. The proposed approach includes a detailed data generation process which allows for the controlled manipulation of important environmental factors, such as lighting, object distance, multiple object instances, and background textures,blur and object's deformation. The tool also accommodates, parameters for generating synthetic datasets for various computer vision applications like,classification, object-detection and 6D object pose estimation in different dataset formats, including BOP and YOLO.

Furthermore, this research evaluated the performance of various uncertainty estimation methods like, Gaussian NLL, Laplace NLL, Ensembles and Bingham Loss in the context of regression based 6D object pose estimation task. The experiments are conducted for various environmental factors like, lighting, distance of objects from camera and deformation of objects, utilizing the capabilities of the developed synthetic dataset generation tool. This thesis seeks to illustrate the effectiveness of the synthetic dataset generation tool in offering a controlled and varied environment for systematically evaluating the uncertainty estimates generated by various DNN architectures and training methodologies. This research contributes to the advancement of robust and reliable deep learning perception, by providing a dataset generation tool and benchmark for evaluating uncertainty in the tasks like 6D object pose estimation.

Acknowledgements

First and foremost, I extend my heartfelt thanks to my supervisors Prof. Dr. Sebastian Houben, Prof. Dr. Nico Hochgeshwender for providing me with this opportunity and their unwavering support. Next I would like to thank MSc. Deebul Nair for his continuous encouragement and support. He guided me always with my best interest in mind.

I would like to especially thank my friends Kaushik Manjunatha, Krishna Teja Nallanukala , Suhasini Venkatesh, Prabhudev Bengaluru Kumar for their enduring presence, continuous support and encouragement during the tough times. I would also like to thank Sai Anudeep Sajja for helping me with setting up the robot for real world experiments.

Lastly, I am eternally thankful to my parents, Mr. Krishna Sai & Mrs. Uma Rajani, my brother Mr. Goutham Panchangam, and my sister in law Mrs. Krishan Swetha for their love and support. To all of you, I owe a debt of gratitude that I can only hope to repay by living up to the ideals you have all instilled in me. Thank you for being part of my journey.

Contents

List of Figures

List of Tables

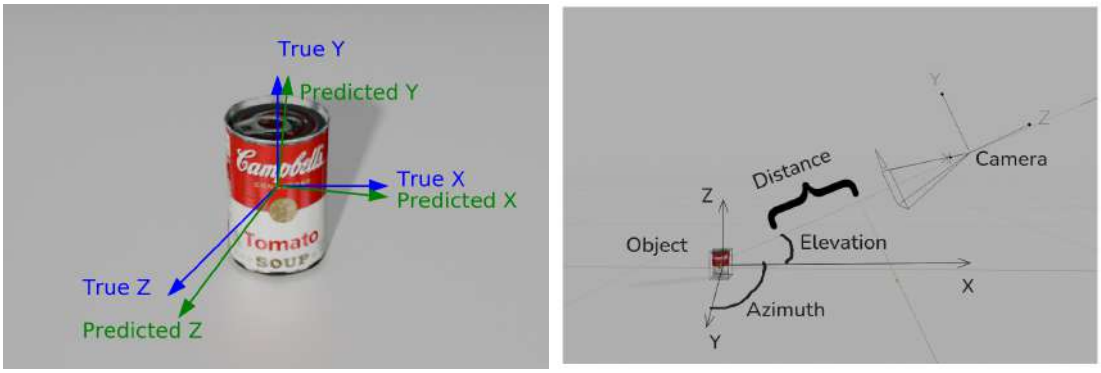
1

Introduction

This chapter outlines the motivation for this thesis, challenges encountered in the field and problem statement along with the research questions that this study seeks to answer.

1.1 Motivation

6D pose estimation task is well known in the fields of robotics, autonomous driving,and medical care. It provides a means for the user to recover the pose of an object from 2D images as shown in Figure 1a. The placement of any object in the real world can be represented using its Pose (3D Location and 3D Rotation). Determining this pose for the object can help the robot, in preforming various tasks like picking, placing and insertion of objects in the desired locations [3]. Estimating the accurate 6D pose of the objects, also helps in avoiding the collisions in autonomous driving situations [4] and can also lead to better performance in SLAM approaches for localization and mapping [5]. In general the 6D pose of an object in real world can be represented using two methods, one is using the 3D location and 3D Rotation parameters as shown in the figure 1.1a. The other approach represents the 6D pose using Elevation, Azimuth and Distance values as shown in the figure 1.1b.



(a) 6D Pose Format (3D Location, 3D Rotation) (b) Azimuth-Elevation-Distance Format

Figure 1.1: Different Representations of 6D Pose in 3D Space

To address the issue of 6D object pose estimation, research has identified a number of techniques for estimating the 6D pose. One such approach is that of learning-based methods in which, deep learning

models utilize a variety of inputs, including RGB images, depth images, 2D-3D bounding box information, segmentation masks, 3D models of the objects and camera intrinsic parameters in order to estimate the pose in 3d space. All these tasks require, diverse datasets with high quality annotations for training the deep learning models. Annotating the 6D poses of objects in images is fundamentally more complex than annotating for tasks like as object identification or image classification, as the annotations must be provided in a 3D metric space. Additionally, depending on the task, large datasets with thousands of images are required for training the deep learning models. Due to the substantial demand for extensive labeled training data in deep learning, a recent research trend has concentrated on generating synthetic datasets for training purposes. Many of these datasets are realistic and they demand substantial modeling effort from experienced 3D artists. These synthetic datasets provides more variations and high quality annotated datasets for training the deep learning models and achieving good accuracy in the pose estimates.

Nevertheless, the emergence of synthetic dataset generating tools has enabled researchers to produce a variety of test datasets with diverse shapes and distribution shifts which may subsequently be employed to assess the effectiveness of pose estimators. These tools demand the prior knowledge of the simulation tools and research has provided many synthetic dataset generation tools. But they do not incorporate all the information needed for evaluating the efficacy of different deep learning models. In this regard, a prior work has been conducted in earlier research and development ($R\&D$) project [1] in which we have developed a synthetic dataset generation tool using the API of Blender3D software which is capable to produce datasets for classification tasks in diverse experimental scenarios, including those involving varying lighting levels, distances, blurring effects, and background textures. But this tool is not enough for evaluating the performance of deep learning models in 6D object pose estimation task. Hence in this thesis, we extended the capabilities of the foundational dataset generation tool developed during our prior ($R\&D$) project to generate datasets for object detection and 6D object pose estimation tasks.

The accuracy of deep learning models increases with dataset variations; nevertheless, the incorporation of appearance in-variance may lead to significantly questionable pose estimates. Moreover, uncertainty may escalate when objects are distorted or when the camera's distance is either excessively far or excessively close, resulting in information inside the image that is not recognized by the trained model. These uncertainties may result in task failure or system failure, and may also cause severe damage to both humans and equipment. To assess the efficacy and comprehend the behavior of any deep learning-based pose estimation methods, it is essential to model the uncertainty along with the predictions.

1.2 Problem Statement

With the use of deep learning approaches in high risk applications like, robotic manipulation, perception, autonomous navigation, healthcare applications and beyond, the need for models with well calibrated and reliable estimates has increased. The applications cover various tasks like, regression, classification, object detection, 6D object pose estimation and beyond. Irrespective of the task, the deep learning models are expected to produce accurate, reliable and well calibrated estimates. But in real world, deep learning models fail to provide reliable estimates due to various reasons including, occlusions, lighting, background textures etc. Particularly, the models mostly fail to produce reliable estimates on unseen data, which

cannot be avoided in real world applications thus, resulting in the failure of the task. Hence in any task involving deep learning, it is important to have a good DNN model which can provide reliable estimates. However, with the increasing diversity of DNN architectures and training methodologies, it has become challenging for the evaluation and comparison of the performance of different deep learning models.

As the field evolves rapidly, it is essential to establish methodologies and benchmarks for assessing the trustworthiness of the expectations made by the deep learning models. To this extent research has provided standard real world datasets like CIFAR-10, ImageNet, BOP (Benchmark for object pose estimation challenge) [6], as benchmarks for evaluating the performance of deep learning models. But all these datasets do not employ different variations in the data for testing the performance of DNNs against new unseen data like, different lighting or background textures or noise in the data. For this, research uses augmentation techniques as a means to solve the problem to some extent. While, test time augmentation techniques are helpful for creating variations in lighting, blur, masked occlusions etc, it is challenging to create augmented data for object's deformation, distance and background textures. To solve this researchers use Synthetic datasets, which are artificial images generated using simulation software and rendering engines. The existing dataset generating options are either complicated to use or require prior familiarity with the simulation software. This thesis aims to create a method for generating synthetic datasets to address some shortcomings in the current literature. This encompasses a more detailed data creation procedure, control of illumination, distance, numerous instances of objects and the background textures, as well as several dataset formats including BOP and YOLO.

Additionally, research indicates that regression-based methods demonstrate superior performance in terms of accuracy, real time speed and complexity of the pipeline [7].As important as the choice and variations in the datasets, testing methodology also plays an important role in evaluating the performance of deep learning models. A complete evaluation includes determining the model's robustness, generalization, and calibration under a range of circumstances in addition to assessing accuracy on a held-out test set. The dependability of a model can be greatly influenced by the testing procedures used, particularly in real-world situations where uncertainty needs to be measured or change in data distributions. The incorporation of a measure of uncertainty into the predictions of a deep learning model will facilitate the formulation of well-informed decisions, thereby reducing the risks associated with potential accidents. In order to address this issue, we are continuing our research into the evaluation of different uncertainty estimation methods for the task of regression based 6D object pose estimation. This is being done in accordance with the research questions that have been formulated in section 1.2.1. It is our intention to demonstrate that the synthetic dataset generation tool developed in this study will prove beneficial for the benchmarking of uncertainty estimation of object pose estimation task.

1.2.1 Research Questions

This thesis primarily deals with the estimation of uncertainty in regression based 6D object pose using rgb images. This thesis addresses the following Research Questions (RQ):

- **RQ1:** What are the state of the art literature in Synthetic data generation tools.
- **RQ2:** What is the impact of lighting conditions on Uncertainty Estimation Methods in 6D Object Pose Estimation.
- **RQ3:** What is the impact of object's distance on Uncertainty Estimation Methods in 6D Object Pose Estimation.
- **RQ4:** What is the impact of object's deformation on Uncertainty Estimation Methods in 6D Object Pose Estimation.

2

State of the Art

2.1 Synthetic Dataset Generation Tools

Inorder to achieve great accuracy in 6D pose estimation, deep learning models require huge amounts of representative datasets which contain wide range of poses,illumination,background textures, depth maps, point clouds and semantic maps. Having such datasets will help deep learning models to generalize well in unseen environments. However obtaining real world data for 6D pose estimation is challenging as it requires expensive setup and markers for obtaining the ground truth pose. In continuation, one should also deal with the noises from sensors like RGB-D cameras, LIDAR sensors and marker placements, while collecting the data. To solve this problem computer generated datasets came into existence and most of the current research uses synthetic data in many applications like, object detection, optical flow, semantic segmentation and 6D pose estimation.

To this extent, synthetic datasets have become a prominent feature of various deep learning tasks, including object detection, depth estimation, semantic segmentation, and 6D pose estimation. Furthermore, they are being employed in conjunction with real-world datasets to enhance the efficacy of deep learning models in practical applications. For example, Yongzhi Su et al., proposed a domain adaptation scheme which transforms synthetic and real images into an intermediate domain so that the correspondences in this domain will be a superior fit for both domains [8]. In the works of Denninger et al., authors have used blender software's API functionalities to produce realistic-looking pictures of situations that offer flawless segmentation masks, depth, or normal images.

The current state of the art provides synthetic dataset generation tools such as Kubric, which is built using Blender and the PyBullet physics engine [9], UNREALCV, which is built using Unreal Engine [10], BlenderProc, which is built using Blender and the Bullet physics engine [11], and numerous other tools that utilize platforms such as Nvidia, Unity, and Gazebo. While these tools are capable of generating datasets for 6D object pose estimation, they lack the capacity to express the level of detail or granularity required for the generation process. Furthermore, the user lacks control over the majority of parameters, including lighting, camera placement, object placement, and other aspects of 6D pose estimation tasks, such as the presence of multiple instances of the same or different objects.

Lofgren et al., proposed a framework that learns to generate realistic synthetic images for training 6D object pose estimation networks. The method employs a Generative Adversarial Network (GAN) to refine

the rendering process of the synthetic objects, making them visually more close to the real images. The approach shows improved pose estimation accuracy on real-world datasets when models are trained on these GAN-enhanced synthetic images [12].

A close work to our Idea has been implemented in the Master thesis of Proneeth Sharma et.al, [13] where author has generated datasets for 6D pose in different lighting conditions, background-textures and different sizes of objects, for the task of Few shot learning. In the work of Proneeth Sharma et, al. [13], author has scaled the size of objects to make them appear big or small by keeping the camera's distance constant. But it is not the case in real world. Instead we can directly move the camera towards the object in a defined path to get the near and far images of the object. This allows us to store the information of the distance between camera and object as a metadata which will be helpful for the performance of different uncertainty estimation methods.

In regard to the proposed benchmark protocol by the authors Proneeth Sharma et.al, [13] , a parallel effort has been undertaken in our prior research and development ($R\&D$) project [1]. In this earlier work, we have developed a synthetic dataset generation tool which is capable of producing datasets for classification tasks in diverse experimental scenarios, including those involving varying lighting levels, distances, blurring effects, and background textures.

2.2 Uncertainty Estimation for 6D Object Pose

Based on the approach, object pose estimation methods are mainly categorized into three categories: Feature based, Template based and Learning based methods [7]. Among these categories, learning-based methods have gained greater popularity due to the capabilities of convolutional neural network (CNN) architectures. These learning-based methods employ either bounding box information or classification methods in conjunction with the PnP algorithm (Point in Perspective) for the purpose of recovering the 6D pose of the object. One method for estimating the 6D pose of an object is to utilize features derived from traditional computer vision techniques, such as edge detection, gray value analysis, and keypoint detection. This approach involves establishing a correspondence between the two-dimensional image and the three-dimensional object model. Additionally, research indicates that learning-based methods demonstrate superior performance in terms of accuracy and precision when trained on huge amounts of data [7].

Deep learning based 6D pose estimation methods involves one-stage or two-stage approaches based on the pose refining step. Some of the approaches uses PnP algorithm and RANSAC for refining the pose before the final 6D pose output.

In general DNN models by themselves do not provide any measure of uncertainty and good uncertainty quantification (UQ) approaches are required to offer evidence for the credibility assessment of the deep learning systems. One way to measure uncertainty in deep learning is through the use of probabilistic models which can output not only a prediction, but also a probability distribution over the possible predictions. This allows us to quantify the uncertainty of the model by looking at the spread of the probability distribution. Especially, estimating the uncertainty in the 6D object pose estimation task is very crucial as the predicted poses define the execution of subsequent tasks like, picking up objects and

inserting them into their respective places.

The existing state-of-the-art in uncertainty estimation in deep learning depends on two or more deep neural network models to assess predicted uncertainty through modifying the output layer, loss function, or network architecture. To assess the efficacy of various uncertainty estimation techniques, research has established benchmarks such as out-of-domain distribution, adversarial attacks, data shift, and calibration metrics, wherein predicted uncertainty is compared with the true distribution using either a holdout set or cross-validation.

In our prior research and development project [1], various uncertainty estimation methods like, Ensembles, Monte Carlo Dropout, Evidential Deep learning were evaluated in the context of object classification task.

2.3 Challenges and Limitations of previous work

- Despite the popularity of synthetic dataset generation tools presented in the literature, the tools lack to provide level of detail for different parameters in dataset generation process.
- Generating synthetic data for different applications requires knowledge of 3D modeling software. It can be time-consuming and resource-intensive, particularly for tasks that require a large amounts of high resolution data.
- The user must have some knowledge of the graphics software to create scenes and generate the required datasets. Some tools require the complete installation of the graphics software, such as Unreal Engine and Unity.
- The dataset generation tool developed in our prior Research and Development project is not capable of generating datasets for 6D object pose estimation task and deformation constraint [1].
- The popular synthetic dataset generation tools like Kubric [9],BlenderProc [11] and UnrealCV [10] does not provide direct specifications for generating datasets for single and multiple instances of same objects in the scene, which are very much important for the research field.
- The quality of the uncertainty estimates mainly depends on the method used for estimating the uncertainty and these methods should be simple such that they do not introduce major changes to the deep learning architecture thereby causing an issue of retraining which results in high computational costs and time.
- Assessing the quality of uncertainty estimates is not straightforward, as there are no direct ground truth for available for evaluation **??**.
- A well-calibrated Uncertainty Estimation Method yields estimates that closely correspond to actual outcomes, and research has introduced metrics such as Interval Score [14], UCS-(Uncertainty Calibration Score) [15], and Reliability Curves [15] and Entropy [16] to assess the model's dependability. However, these measurements possess certain drawbacks. Metrics like, interval score, uncertainty calibration error and reliability diagrams cannot be used for comparing the predictions from different distributions.
- In the case of Bingham distribution, as it is modelled using a zero mean Gaussian and it is considered as "the maximum entropy distribution on the hypersphere" [17], Cumulative Distribution Function (CDF) cannot be computed which is required for computing metrics like, interval score, uncertainty calibration error and reliability diagrams.

3

Methodology

This chapter describes the proposed approaches and the state-of-the-art uncertainty estimation methods in the context of 6D object pose estimation.

3.1 Meta Model for Constraint Based Testing

As important as the choice and variations in the datasets, testing methodology also plays an important role in evaluating the performance of deep learning models. A complete evaluation includes determining the model's robustness, generalization, and calibration under a range of circumstances in addition to assessing accuracy on a held-out test set. The dependability of a model can be greatly influenced by the testing procedures used, particularly in real-world situations where change in data distributions appear or uncertainty needs to be measured.

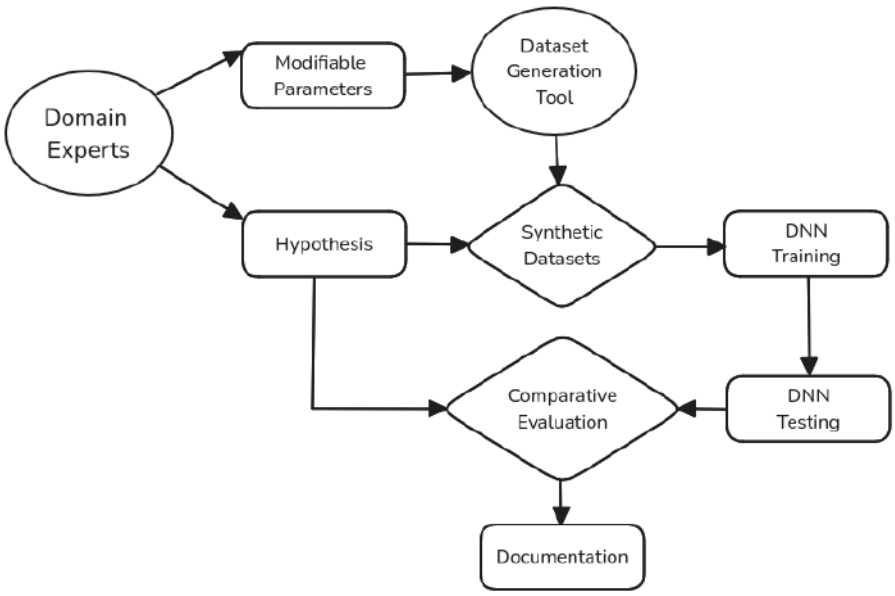


Figure 3.1: Meta model for constraint based testing

In this thesis work, we propose a hypothesis driven constraint based testing approach for evaluating

the performance of deep learning models under various real world abnormalities like, lighting,deformation of objects etc. We achieve this by leveraging the capabilities of simulation software for generating the constraint specific datasets. The approach begins by creating the hypothesis for the expected behavior of the deep learning models under specific test constraint. The constraints are basically developed based on the abnormalities encountered in the real world such as environmental changes or distributional shifts. In this work, we intend to create constraints based on the conditions like, lighting, distance of objects, noise in the image, deformation of object, occlusion, background textures etc.

1. The first step in the process is to create a hypothesis. Based on the task or requirement the domain expert creates a hypothesis for the behavior of the deep learning models. For example based on lighting the hypothesis (H) could be that the DNN models trained on normal lighting conditions are expected to have high uncertainty when tested in bright lighting or dark lighting conditions. This hypothesis is not restricted to any metric it can be created based on the task and the nature of the deep learning models.

2. Next depending on the hypothesis, the domain expert in simulation software identifies the modifiable parameters which can help generating the datasets for testing the hypothesis (H). For example, based on the hypothesis for the testing the lighting conditions, the position, orientation or the intensity of the light object can be modified for generating the datasets for normal lighting, bright lighting and dark lighting conditions. These datasets can be generated using some synthetic dataset generation tools which are capable of providing images along with the ground truth scene parameters, for example, here we need the intensity of light .

3. Based on the requirement different DNN architectures and training methodologies are chosen by the domain experts. Then using the generated normal lighting dataset, the experts train the DNN models such that they generalize well on the validation set.

4. Next, the models trained on normal lighting conditions is then tested on bright and dark lighting datasets using performance measure desired by the domain expert. For example here in this hypothesis as we are interested in checking the uncertainty, entropy metric can be chosen as the performance measure.

5. Now using the test results and the ground truth scene parameter values (intensity values for each rendered image) we perform a comparative evaluation for evaluating the performance of deep learning models against the test constraint. In this hypothesis entropy vs light intensity.

6. Based on the Hypothesis and the test results, we find out the best performing DNN model which satisfies the created hypothesis. If the hypothesis is not met by any of the selected deep learning models, then the domain experts decide to change the hypothesis or find the models that satisfy the hypothesis.

7. Finally, the results are documented such that they can be reproduced by other researchers. This includes the scene configuration files, training and testing parameters.

3.2 Blender Based Dataset Generation Tool

In our earlier R&D (Research and Development project) [1], we have developed a synthetic dataset generation tool using the API of Blender3D software. The tool was developed primarily for generating synthetic datasets for object classification tasks. While the tool was effective in classification context, it must be extended for generating datasets for broader range of computer vision applications like, object detection, semantic segmentation, 6D object pose estimation etc, to serve as a benchmarking framework.

Expanding the tool's capabilities, allows researches for evaluating the performance of deep learning models across diverse perception tasks. In this thesis, we aim to extend the capabilities and generalize the functionality of the dataset generation tool developed during our previous Research and Development (R&D) project [1]. While the goal of this work is to broaden the tool's overall capabilities, the main focus of this thesis is on 6D object pose estimation. As such, the methods and discussions throughout are centered around this particular task.

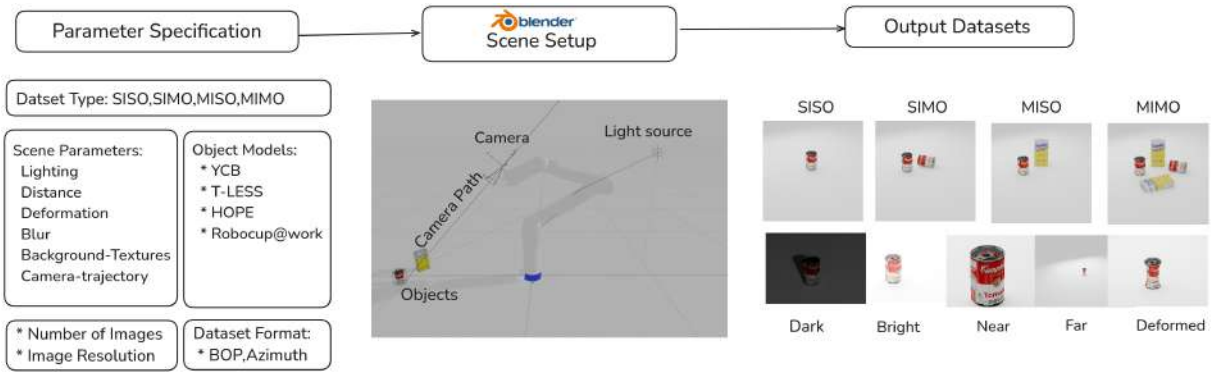


Figure 3.2: Dataset generation methodology for 6D object pose estimation task

During our earlier work, we have identified and documented various modifiable parameters in blender software for which are helpful for synthetic datasets for specific test constraints like, lighting, distance, blur, background textures etc [1]. The same modifiable parameters can be used in this work for generating the datasets for different test constraints. The approach starts by defining a configuration file that specifies the most important settings for controlling the scene and dataset generation. This configuration file is defined in a standard format using a json file structure as it helps to provide parameters separately for individual constraints.

The configuration file describes the features in the dataset generation tool and the users can set the parameters based on the scene and the task. These parameters depend on the scene created in the blender software. A scene in blender is basically a workspace which holds everything for a specific project like, camera setup, lighting setup, world settings, render settings and objects. Blender considers everything as objects which contain meshes (3D representation of real world objects), cameras, light sources, curves, text etc. Since this thesis work focuses on 6D object pose estimation task, it might be little confusing with the terminology of blender software with the object pose estimation context. Hence it is important

to clarify the terminology used in this thesis report. For clarity and consistency, we will use the term object specifically to refer to mesh objects—that is, the 3D representations of real-world items such as a mustard bottle or a tomato soup can. Other elements in the scene, such as cameras, light sources, and curves, will be referred to explicitly by their respective names to avoid ambiguity.

As shown in figure 3.2, here have created a scene with different items like robotic arm, camera, light source, camera path, white background plane and the main objects tomato soup can and sugar box from YCB objects **??**. Using this scene setup and modifying the settings of the items present in the scene we generate datasets for various test constraints. For example, in case of generating datasets for 6D object pose estimation task as shown in figure 3.2, the users initially provide the dataset type (SISO, SIMO, MISO, MIMO) in the configuration file. Then the users can select the main objects by specifying the directory path for the cad models from which the tool will import the objects into the scene.

For each test constraint, users are required to provide a set of scene parameters which basically define the values for the modifiable parameters like, light intensity, camera focal length, depth of field, camera position etc. For bright and dark constraints, we modify the strength parameter of the light source in the blender's scene to introduce the variation in lighting and create datasets for bright lighting and dark lighting conditions.

Similarly for Far and Near constraints, we move the camera along the camera path as shown in the figure 3.2, to introduce variation in distance. This is achieved by setting the camera to move along the predefined camera path using Follow Path constraint in blender setting. By adjusting the offset parameter of the follow path constraint, we generate datasets such that the main objects in Far constraint appears far away from the camera and in Near constraint the objects appear very close to the camera.

For blur constraint, we can introduce noise into camera using the depth of field value to simulate and render blur images. Specifically, we modify the aperture (f-stop) value in the depth of field settings, to introduce a shallow depth of field, causing parts of the image to appear out of focus and blurry. For the textures constraint, random background textures can be applied to the white background plane to create variations in textures. In blender the textures can be applied in two different ways, one way is to create a procedural texture using blender shader nodes editor for which users require knowledge in the blender software. In the second approach the textures which are saved as images can be applied directly even without knowledge in the software. These image textures are generally known as PBR (Physics Based Rendered) textures and they are freely available on various online platforms like Polygon materials [18]. By incorporating a range of realistic textures, we create dataset for textures constraint with different background textures.

For deformation constraint, we can deform the objects using blender's modifiers like simple_deform modifier or directly transforming the mesh vertices such that they depict deformed object. While simple_deform modifier allows the users to deform the object using, twist, bend, taper and stretch properties, it might not be the ideal solution. This is because in real world all objects do not bend or twist as the deformation depends on the type of object's material properties. For example, cardboard can be crushed but a metal rod cannot be crushed but it can be bent. For deforming the objects dynamically we can specify its material properties as a parameter like rigid_plastic, metal or cardboard and based on

the material property we can either use the simple_deform modifier or directly transform the vertices of the mesh object in 3D space.

3.3 Loss Functions

3.3.1 Gaussian NLL Loss

Gaussian Negative Log Likelihood loss function is widely used in regression task as it can model predictive uncertainty. When trained using Gaussian NLL loss, the deep learning models consider the outputs to be the parameters of Gaussian distribution. This means that the models trained using Gaussian NLL loss provides predictions in the form of mean and the confidence in the prediction in the form of the Gaussian variance [19]. The loss function can be formulated by taking the negative log-likelihood of the Gaussian distribution 3.1, [20].

$$p(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(prediction - ground\_truth)^2}{2\sigma^2}) \tag{3.1}$$

3.3.2 Laplace NLL Loss

Laplace Negative Log Likelihood function is probabilistic loss function which is capable of modeling the uncertainty along with predictions in regression tasks. The Laplace distribution can be described using the probability density function provide in the equation 3.2. Here the parameter μ represents the predictions and the scale (s) parameter represents the uncertainty. Laplace NLL loss can be formulated by taking the negative log-likelihood of the Laplace distribution 3.3 [21].

$$p(y|\mu, s) = \frac{1}{2s} exp(-\frac{|y - \mu|}{s}) \tag{3.2}$$

$$L_{NLL}(x, y, \theta) = -log(p(y|f_\theta^\mu, f_\theta^s)) = log(2f_\theta^s) + \frac{|y - f_\theta^\mu|}{f_\theta^s} \tag{3.3}$$

3.3.3 Ensembles Testing

Deep Ensembles models are often used to improve the predictive performance [22]. In Ensembles method multiple deep learning models (M) are trained individually using the same loss function for example, Gaussian NLL loss [22]. During test time the outputs from all the deep learning models is averaged to get the predictions and uncertainty. Ensemble models can estimate both aleatoric and epistemic uncertainty [23]. As the output is obtained by combining the predictions of different deep learning models, the ensemble prediction is considered as a mixture of Gaussian models [23]. The final ensemble mean predictions can be computed by directly averaging the output expectations of all the models 3.4 [23].

$$\mu_{ens} = \frac{1}{M}\sum_{m}^{M}\mu_m \tag{3.4}$$

In ensembles model, as we are using Gaussian NLL loss, the variance cannot be directly averaged to get the final ensemble variance. We compute the variance in ensemble model using the equation 3.5.

$$\sigma^2_{ens} = \frac{1}{M}\sum_{m=1}^{M}\sigma^2_m + \frac{1}{M}\sum_{m=1}^{M}(\mu_m - \mu_{ens})^2 \tag{3.5}$$

Equation 3.5 represents both aleatoric and epistemic uncertainties where the aleatoric uncertainty is expressed by the equation 3.6 [23], and the epistemic uncertainty is represented by the equation3.7 [23].

$$\sigma^2_{ale} = \frac{1}{M}\sum_{m=1}^{M}\sigma^2_m \tag{3.6}$$

$$\sigma^2_{epi} = \frac{1}{M}\sum_{m=1}^{M}(\mu_m - \mu_{ens})^2 \tag{3.7}$$

3.3.4 Bingham Loss

Bingham distribution is generally used for modeling uncertainty in object rotations. For estimating the translation parameters, Gaussian distribution can be used [24]. Also the Bingham distribution is modelled using a zero mean Gaussian and it is considered as "the maximum entropy distribution on the hypersphere" [17]. The main advantage of using Bingham distribution for modeling uncertainty in rotations is because of its capability of capturing antipodal symmetry of unit quaternions [24], [25]. The probability density function of bingham distribution is represented by the equation 3.8 [26] [17].

$$p(x; M, Z) = \frac{1}{N(MZM^T)}exp(x^T MZM^T x), x \in \mathbb{R}^{4*4} \tag{3.8}$$

In the equation 3.8, x is represented as unit vector on the sphere $\mathbb{S}^d \subset \mathbb{R}^{d+1}$. M is a 4x4 orthogonal matrix which represents the rotation parameters. Especially, the fourth column of the M matrix represent the predicted quaternion [26]. Z is a diagonal matrix of eigen values "Z = diag(z1, z2, z3, 0)" in which the Z1, Z2,Z3 values are in ascending order. The values of Z are known as the concentration parameters and they describe the uncertainty. F is considered as the normalization constant.

Computing the normalization constant is the primary difficulty in implementing the bingham loss function. But it can be computed using numerical approximation methods. In the work of (Gilitschenski et al., 2020) authors used RBF interpolation method for computing the normalization constant utilizing a pre computed lookup table [26]. The lookup table plays a major role on the performance of the Bingham loss and it is generated based on the upper and lower bounds of the coordinates in the dataset.

4

Solution

The initial Research and Development project presented in [1], resulted in the creation of a dataset generation tool, primarily designed for object classification task. This tool leveraged the capabilities of Blender3D software [27] and its python API (Bpy) for creating datasets for various constraints like, lighting,distance,background_textures,and blur_images in the context of object classification problem. Building upon this foundational dataset generation tool, this thesis presents a significantly enhanced version capable of producing synthetic datasets for wide range of computer vision applications like object detection, 6D object pose estimation, semantic segmentation, surface normal estimation, depth estimation etc.

4.1 Blender Based Dataset Generation Tool

As discussed in the previous chapter 3.2, synthetic datasets can be generated by varying different parameters like, lighting, camera position, object position, background textures, camera depth of field or the shape of the object itself. In general, these parameters are provided as a configuration file for generating the synthetic datasets based on the user's requirement. For this the user is expected to have knowledge of the simulation software and the configuration file is written manually. To overcome this challenge, in this work a GUI application is developed for creating the configuration file using Streamlit package.

With the help of GUI based configuration file generation approach, user will have more control and clear understanding of the dataset generation process with very minimum knowledge of the simulation software. In this section we will explore the configuration file generation and dataset generation process simultaneously, illustrating how Streamlit GUI enables users to easily modify parameters and then see the effect of these parameters on the final generated datasets.

4.1.1 Dataset Settings

Figure 4.1 depicts the dataset settings, where users can choose options based on their requirement. In the Dataset Type option users can select the task for which they want to create datasets. Available options are Classification, Object Detection and 6D object Pose Estimation. Since our primary focus in this thesis is on 6D object pose estimation task we explore the configuration file and dataset generation

process specifically in this context. In the next option (Dataset Label format) scene_folder represent the folder structure of the generated datasets. Here the folder structure is similar to BOP dataset format similar to YCB-V,TLESS datasets. In a dataset will have separate folder for rgb images, depth images, segmentation masks and the annotations are saved as JSON files.

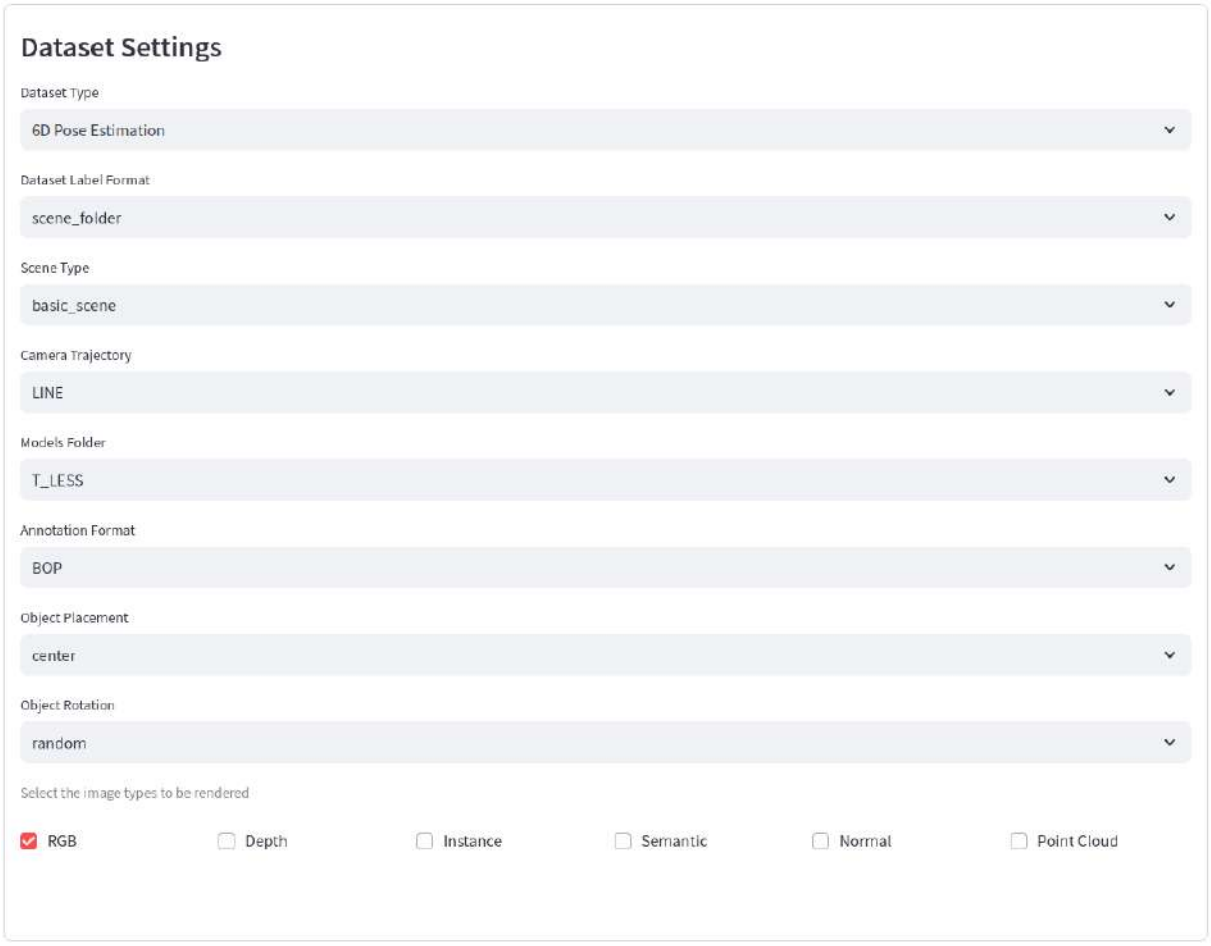


Figure 4.1: Configuration File Generator: Dataset Settings

Scene Settings

Next comes the Scene Type, for generating any kind of synthetic datasets the first most important criteria is to create an environment which is known as scene. Depending on the task and complexity, this scene can be created such that it depicts the real world or it can contain a simple background for the dataset generation. Here a basic scene is provided for the users with no prior knowledge. Along with the scene type we can also choose the Trajectory for the camera's movement in the scene (LINE or CURVE or Random). Based on this selection the dataset generation tool will create scene with a normal plane

background, a light source, a camera.

For camera trajectory a nurbs path is added as a camera track for simulating the camera movement in a line as shown in the figure 4.2a and a bezier circle is added as camera track for moving the camera in a curved trajectory as shown in figure 4.2b. In blender we can set constraints to the objects control them. Here we use the Follow Path constraint and make the camera follow the camera_track object. By controlling the offset parameter of follow path constraint we get variation in camera's position. A detailed explanation of the constraints present in blender are explained in our earlier Research and Development project [1].

Similarly for variation in lighting another bezier circle object is added as Light track and the light source is constrained to the light track using Follow Path constraint. Additionally, a Track To constraint is added to the light source so that the light source will always be pointed towards the main object (Eg: Mustard Bottle) as shown in the figures [4.2a, 4.2b].

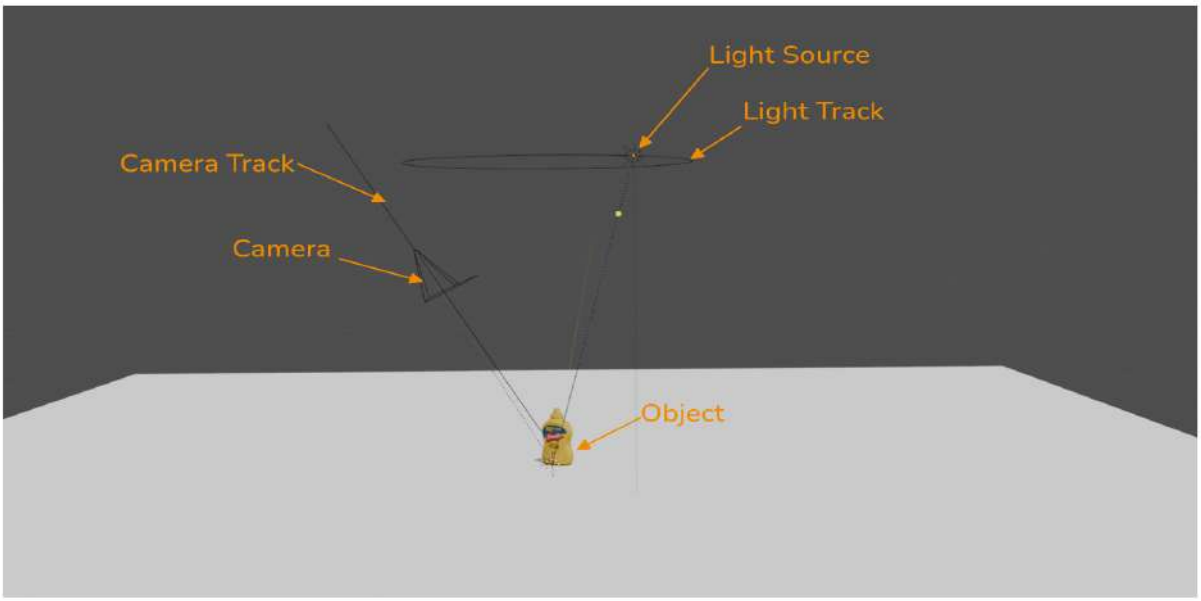
For the users with knowledge in blender software a custom environment or scene can be created based on the requirements and integrated into the developed dataset generation tool. For this the custom scene can be added as blender file in .blend format to the scenes_dir of the dataset generation tool package as shown in 4.1.

```
Dataset_Generation_Tool/
+-- data/
|   +-- config_dir/
|   +-- scenes_dir/
|   +-- models_dir/
|   |   +-- YCB/
|   |   |   +-- models/
|   |   |   +-- models_info.json
|   |   +-- TLESS/
|   |   |   +-- models/
|   |   |   +-- models_info.json
|   +-- textures_dir/
|   +-- constraint_textures_dir/
|   +-- urdf_dir/
+-- utils/
+-- src/
```

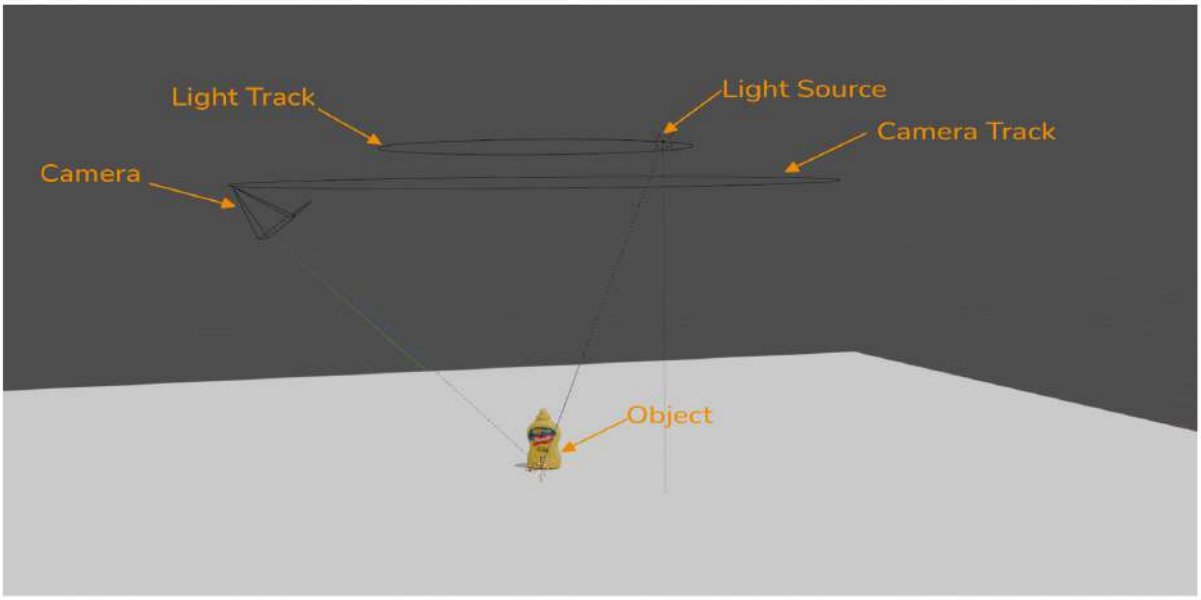
Listing 4.1: Dataset Generation Tool Directory Structure

While creating a custom environment the user need not to import the main objects (Eg: Mustard bottle) into the scene. This step will be done by the dataset generation tool in the later step. For the tool to work properly the user has to follow three standard requirements.

- First the user has to create light_track and camera_track objects in the scene.
- Second step is to assign Follow path constraint for both camera and light source objects and make the objects follow their respective paths i.e, light source should follow light_track and camera should follow camera_track.



(a) Basic scene setup with line trajectory



(b) Basic scene setup with curve trajectory

Figure 4.2: Basic scene setup in blender3D for dataset generation

- The Final requirement is that the user should follow a standard naming convention for the objects present in the basic scene setup (Background_plane, Sun,Camera,light_track,camera_track).

In this work an indoor environment is created for the C025 Laboratory in Hochschule Bonn Rhein Sieg University of Applied Sciences. An Example image of the scene rendered using Blender 3D is depicted in

the figure 4.3. It is taken care that the dimensions of the objects and the laboratory match closely to the real world. To the scene a Kinova Gen 3 6DOF Robotic Arm is added and the robotic arm is rigged using standard rigging procedures and the Inverse Kinematics is setup for animating. The camera object is parented on top of the end effector and a camera track is added to the scene. Here instead of making the camera following the camera_track, the tip of end effector is made to follow the camera track. This is specifically done to simulate the movement of the robotic arm. Since the camera is parented to the end effector we automatically get the variation in camera movement when the robot is moved. In this scene, all the ceiling lights are grouped as a single object and named as Sun and the light intensity across all the ceiling light objects are controlled at a time for creating variation in lighting. For the background plane, the Top Flat surface of the Table object is named as Background_plane. This naming convention is important for applying background textures later in the dataset generation process. Similarly any custom scene can be created by the users by following the standard requirements of the dataset generation tool. The core idea of creating custom environments is that the scene will help the user to adjust various parameters in the simulation software for generating datasets based on the requirement.

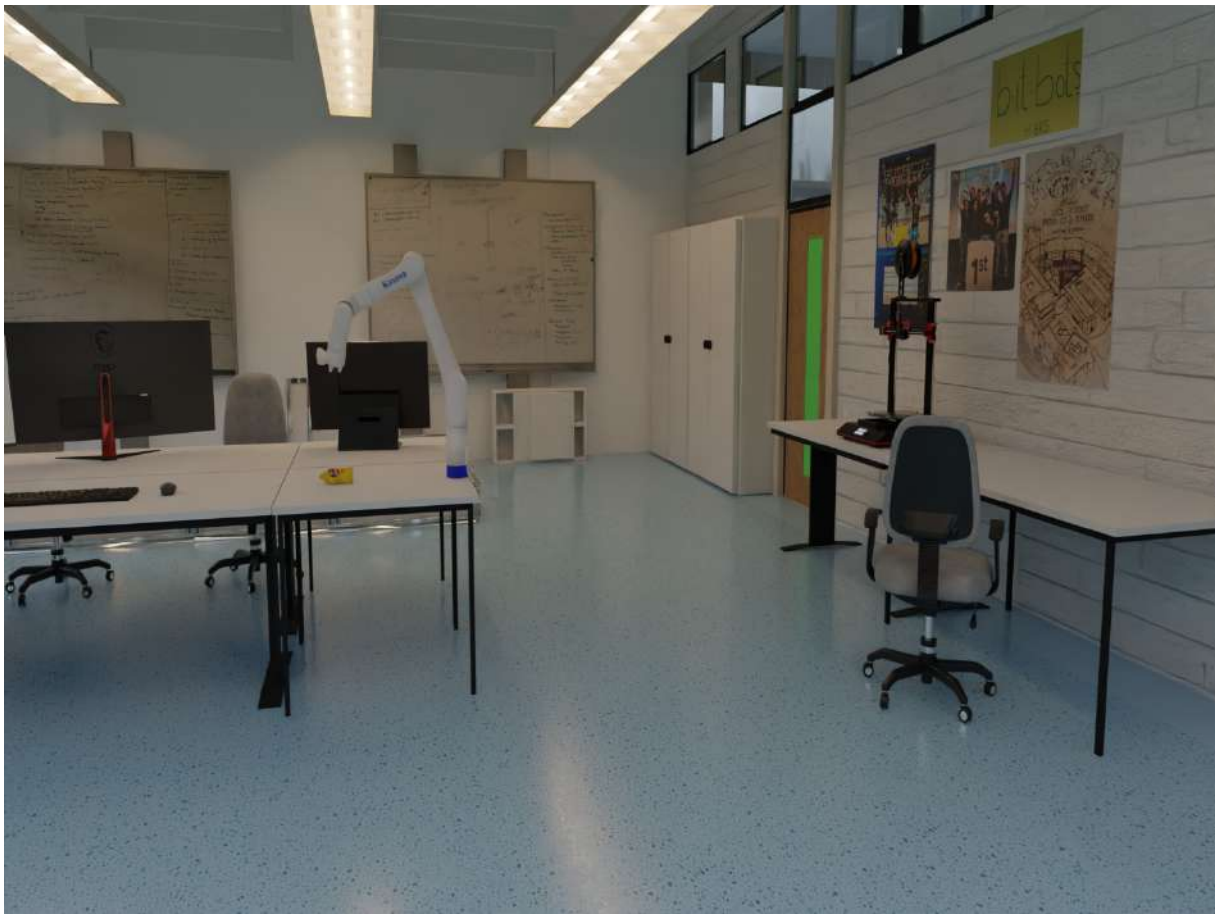


Figure 4.3: HBRS C_025 lab environment setup in Blender3D

Objects settings for the dataset generation

As shown in the figure Dataset Settings 4.1, the next option in the tool is Models Folder. The tool provides cad models from for different datasets, YCB [28],Texture Less objects (TLESS) [29] , HOPE [30] and RoboCup@Work. Based on the requirement of the user new objects can be integrated into the dataset generation tool by placing the cad models of the objects in the models_dir of dataset generation tool 4.1.

It is very important to create the models_info.json file in the models_dir. The models_info file represents the object meta data properties such as dimensions of the object and object material as shown in the figure 4.2. The open source cad models present in (Benchmark for Object Pose) BOP challenge [6] already provide the models_info.json file along with the 3d cad models in .ply or .obj or .stl formats. For proper working of the developed dataset generation tool the user need to add an additional information object_material (soft_plastic,rigid_plastic,cardboard,steel,aluminium) as depicted in the listing 4.2. This step is crucial for the deformation of objects.

```
{
  "1": {
    "size_x": 34.9916,
    "size_y": 34.9916,
    "size_z": 61.2,
    "object_material": "soft_plastic"
  },
  "2": {}
}
```

Listing 4.2: Example metadata for a single object

Furthermore in Dataset settings Figure 4.1, we have Annotation Format, Object Placement and Object Rotation. For 6d pose estimation task the annotations can be saved in BOP (Benchmark for object pose estimation) format [6] or Azimuth-Elevation-Distance Format and the bounding box annotations are saved in YOLO format. For object detection task the user can select the option of saving the annotations in YOLO or in COCO format. This thesis work aims to generate datasets in BOP Format for 6D object pose estimation task.

Blender uses the right-hand coordinate system for both the world frame and camera frame. The world coordinate system in Blender provides Y as the forward axis,z as the up axis and x as the right axis. By default, the 6D pose of the objects is obtained in the world frame. But for the 6D object pose estimation task, we need the poses of each object in the camera frame. So while generating the datasets, the pose of each object is transformed into the camera's frame before saving the annotations. The quaternions obtained from blender are unit quaternions and they do not require any additional normalization. The quaternion_annotations are obtained by directly decomposing the pose transformation matrix of the object using bpy module.

$$\mathbf{T}_{\text{obj}}^{\text{camera}} = \left(\mathbf{T}_{\text{camera}}^{\text{world}}\right)^{-1} \cdot \mathbf{T}_{\text{obj}}^{\text{world}} \tag{4.1}$$

For a good 6d object pose dataset, a wide range of object poses are essential. By default when the cad models of the objects are imported into the blender's scene, they are placed at the world origin and when we import multiple models, all the models will be cluttered or merged with each other. Hence it is important to place the objects properly in the scene. The dataset generation tool allows the user to select the placement of the objects with Object Placement parameter. Using this parameter, objects can be placed at center of the background_plane or in a circle arrangement or in random positions with in the camera frame while generating the datasets.

Similar to the object placement, the dataset should also cover diverse variations in object's orientation. For this the dataset generation tool provides Object rotation parameter. With the help of this parameter the user can get random orientations of the objects while generating the datasets. Also fixed orientation option can be used in the tasks requiring consistent object poses.

Finally in the Dataset Settings, the user has the options to select different types of data based on the requirement or task. The developed dataset generation tool allows the user to select different image types like; RGB images, Depth images, Segmentation Masks, Surface Normal Maps and as well as point clouds of the objects. Based on desired task these options can be checked and the corresponding images will be saved in the respective folders of the generated datasets.

4.1.2 6D Pose Settings

Based on the type and number of objects present in the scene, 6D object pose estimation task can be divided into 4 categories as depicted in the figure 4.4. Single Instance of Single Object (SISO)4.4a, Single Instance of Multiple Objects (SIMO)4.4b, Multiple Instances of Single Object (MISO) 4.4c and Multiple Instances of Multiple Objects (MIMO) 4.4d.

(a) SISO (b) SIMO (c) MISO (d) MIMO

Figure 4.4: 6D pose dataset based on object categories

The dataset generation tool developed in this work accounts for creating datasets for the 4 object categories shown in figure 4.4. As shown in figure 4.5 the user can select any one of the option from the 4 object categories using the Dataset Type option and based on the requirement the user can select all of the objects present in the models folder or only specific objects. The number of instances for each object can be adjusted using the number of instances parameter as shown in the figure 4.5. The parameter

selection changes dynamically based on the object categories.

The figure 4.5 depicts an example usage of the configuration file generator. This thesis aims to conduct experiments for a Single Instance of Single object and thus all the datasets were generated using SISO object category. In this work, two classes from YCB objects namely, YCB_Mustard_bottle and YCB_Tomato_Soup_Can were selected as desired objects and the configuration files for both objects are saved separately.

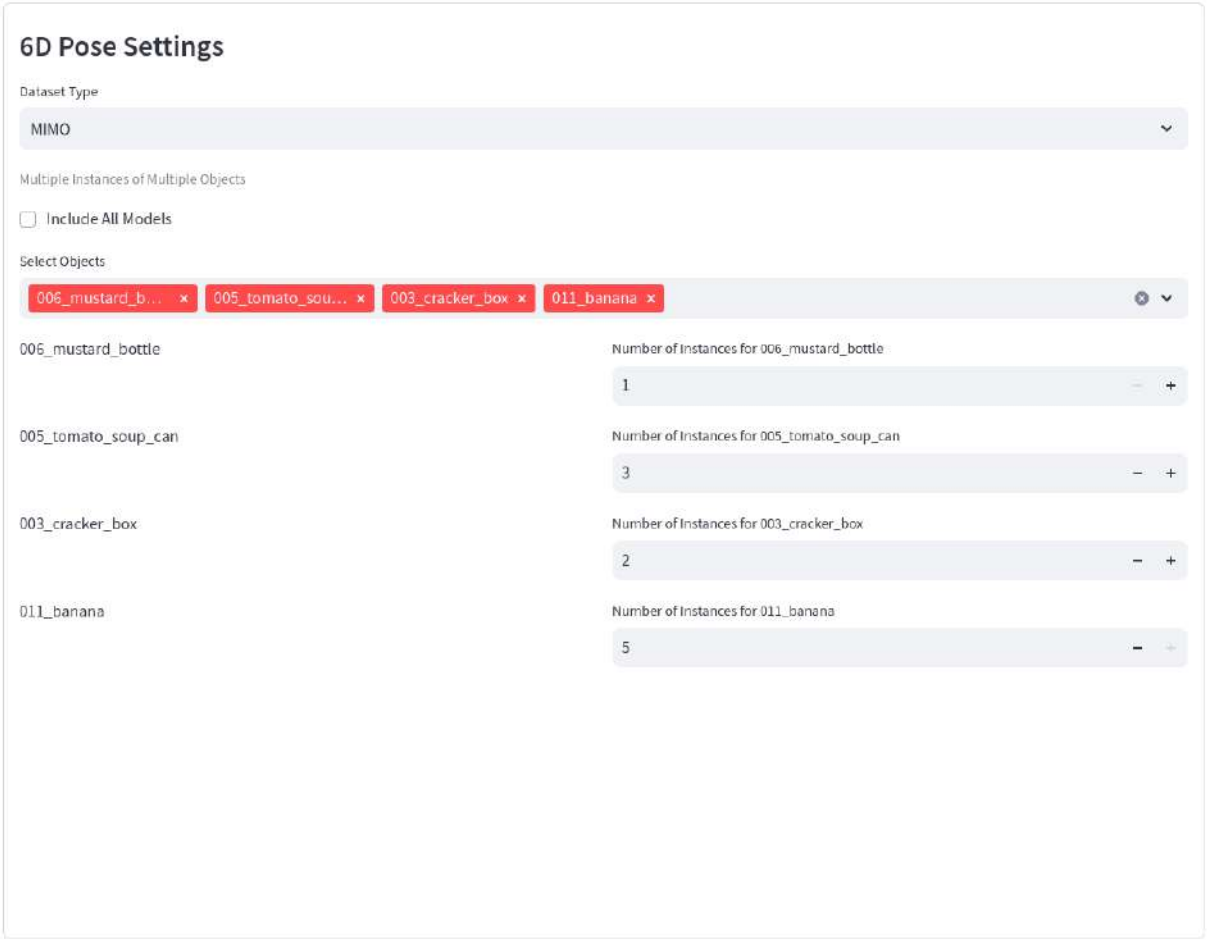


Figure 4.5: Configuration File Generator: 6D Pose Settings

4.1.3 Render Settings

Render Settings allows the users to select the parameters for the render engine, image resolution, image format and focal length of the blender's camera object. During rendering process, the 3D scene visible in blender's camera frame is converted into a 2D image with the help of render engine. Blender provides two rendering engines CYCLES and EEVEE for generating the images.

While CYCLES is a path-traced render engine that is used to generate photo realistic images, EEVEE is used as a real time rendering engine aiming to prioritize speed over the realism. In this work, dataset generation tool is mainly developed for CYCLES render engine but depending on the requirement the user can also use EEVEE as the render engine as shown in the figure 4.6. In this work we use CYCLES as the main render engine for generating the datasets.

Figure 4.6: Configuration File Generator: Render Settings

The dataset generation tool also allows the users to specify the resolution of the images as shown in the figure 4.6. Higher the resolution the more time it takes for generating the datasets. In this work, all the images are generated in 224 x 224 resolution. For the file format the tool provides only two formats, PNG and JPEG. In blender the default focal length is set to 50mm, this can be changed according to the requirement and the scene setup of the user. In this work a focal length of 26mm is used for generating the datasets as it is similar to the focal length of most real world cameras.

4.2 Constraint Settings for Dataset Generation

As discussed in the previous chapter Methodology 3, the main aim of the developed dataset generation tool is to create datasets based on constraints like lighting, distance, deformation, blur, background textures etc. This can be achieved by utilizing the capabilities of Blender's API and modifying the parameters of various settings in the blender scene. Some of the modifiable parameters which are helpful for generating the datasets are provide in the table 4.1.

Leveraging the modifiable parameters presented in the table4.1, the dataset generation tool is developed to generate datasets for specific constraints like; Normal, Bright, Dark, Far, Near, Blur, Background-Textures and Deformation. The tool allows the users to directly control and adjust the parameters for each constraint individually. The following sections will provide an in-depth explanation of the dataset-generation process for each constraint.

Name	Modifiable Parameters	Blender constraints
Light Source	Position, Orientation, Type (Point,Sun,Area,Spot), Light Intensity	Follow Path, Track To
Camera	Position, Orientation, Focal Length, Depth of Field, Resolution	Follow Path, Track To, Parent
3D Object	Position, Orientation, 3D Mesh Data, Materials and Textures	Sub-division Surface Modifier

Table 4.1: Modifiable parameters in Blender 3D software for dataset generation

4.2.1 Normal Constraint

For any good training dataset, diversity of scenes, variation in data and good quality of annotations are important, such that they help the generalization of deep learning models. Hence in this work, we have generated datasets using normal constraint parameters and considered them as the nominal training datasets. The dataset generated under normal constraint constitutes for a good range of poses, lighting and distance of objects.

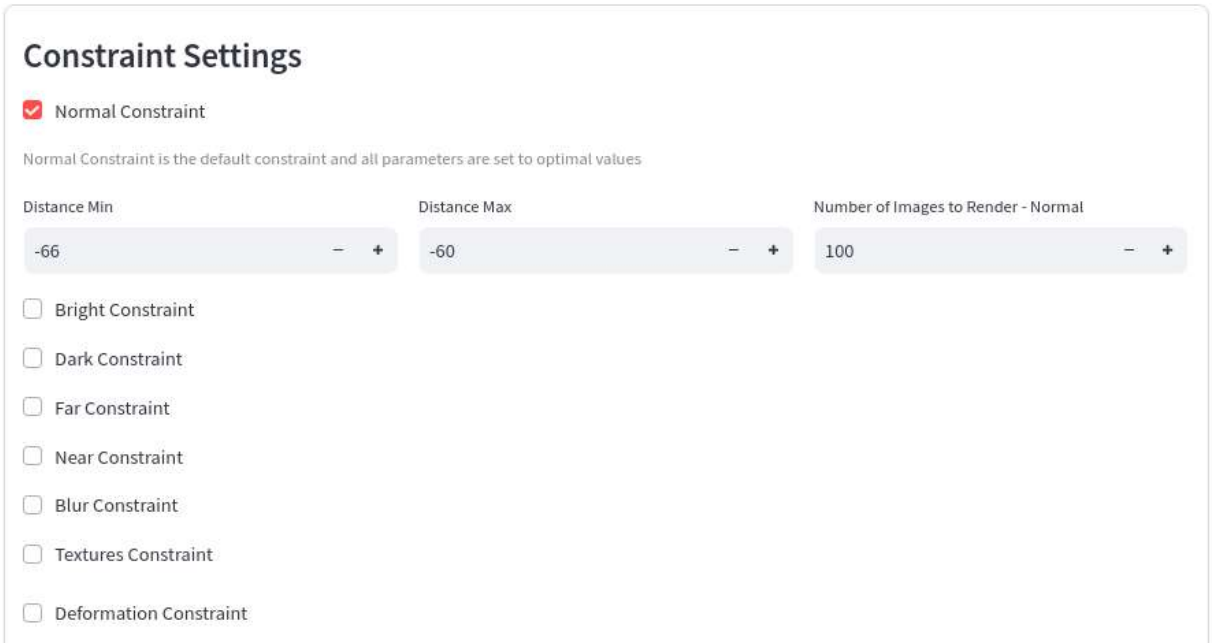


Figure 4.7: Configuration File Generator: Normal constraint

In this constraint, based on the scene setup, first a normal range (l_x,l_y) is selected for the light intensity. This range is selected such that the objects in the rendered images appear clearly in good

lighting conditions. During rendering for each image a light intensity value is sampled from uniform distribution using the normal light range ($l_x = 3.0$, $l_y = 5.0$) such that it creates variation for lighting conditions. This sampled light intensity value is then used as the strength parameter for the light source (Sun) in blender.

Next, to introduce variation in the camera's pose, as discussed in the scene setup section 4.1.1, a camera track is created using blender's nurbs path object as a camera track as shown in the figure 4.2a. Next, blender's Follow Path constraint is applied to the camera so that the camera moves along the camera track. The movement of the camera along the camera track can be controlled using the offset parameter of the Follow Path constraint.

The normal range for the distance (d_x,d_y) is selected such that the objects in the image appear clearly without any occlusion or loss of information as shown in the figure 4.7. Based on the scene setup used in this work, ($d_x = -68$, $d_y = -70$) values were selected for generating datasets under normal conditions. Based on the scene setup, the distance offset values (d_x,d_y) needs to be adjusted. For the users it is suggested to generate few sample images first to check if the object's appearance is in desired conditions and then generate the actual datasets.

The variability for the location and rotation of the objects is set to random using the settings depicted in the figure 4.1. Based on the selected parameters, the rendered rgb images for the normal constraint are depicted in the figure 4.8. The images for other types of data like, depth maps, segmentation maps, surface normal maps, point clouds are depicted in the Appendix **??**

Figure 4.8: Example images rendered under normal constraint

The annotations for all rendered images are saved in BOP format in the form of json files. The annotations for the complete dataset are divided and saved into 4 different json files (scene_gt.json, scene_camera.json, scene_gt_info.json, blender_gt.json). Each file represents different information of the scene and objects present in the rendered images.

The first json file, scene_gt.json provides the annotations of the object's pose in camera frame for

all rendered images. The listing 4.3 depicts an example annotation for a single rendered image. The annotations depicted in the listing describes the pose of the object in camera frame. In this work we have used the annotations from scene_gt.json 4.3 for training the deep learning models in the context of 6D object pose estimation task. Here we specifically used the quaternion rotations (cam_Q_m2c) and translation vector (cam_t_m2c) as ground truth pose annotations during training and testing of deep learning models.

```
"0":{
      "obj_id": 1,
      "obj_name": "mustardbottle",
      "cam_R_m2c": [
        [-0.4097, -0.7472, -0.5232],
        [0.2070, -0.6348, 0.7444],
        [-0.8884, 0.1967, 0.4148]
      ],
      "cam_t_m2c": [-0.0787, 0.0913, -0.5039],
      "cam_Q_m2c": [0.3042, -0.4501, 0.3001, 0.7841]
    }
```

Listing 4.3: Example annotation from scene_gt.json file

Second json file, scene_camera.json provides the information of the camera with respect to the blender's world frame. It provides camera's intrinsic and extrinsic parameters as K matrix and the rotation matrix and translation vector of the camera with respect the blender's world frame. An example annotation from scene_camera.json is depicted in the listing 4.4.

```
"0":{
      "cam_K": [161.77777099609375, 0.0, 112.0, 0.0, 242.6666717529297, 112.0, 0.0,
          0.0, 1.0],
      "cam_R_w2c": [
        0.44148467329682023, 8.411994318853858e-09, 0.8972688310597726,
        -0.8281727160197699, -0.3848168972319831, 0.4074871839475885,
        0.34528418805542793, -0.9229929152994023, -0.1698907470998805
      ],
      "cam_t_w2c": [-2.094540989903556e-09, -2.3207770399609215e-09,
          -0.3820344664082122],
      "depth_scale": 0.1
    }
```

Listing 4.4: Example annotation from scene_camera.json file

Third json file, scene_gt_info.json provides bounding box information objects present in the rendered images in YOLO annotation format as depicted in the listing 4.5. This file also provides the pose of object with respect to the blender's world frame along with the scale and dimensions of the object. For each object a class name and class id are assigned while generating the datasets.

As depicted in the listing 4.5, "bbox_obj" represents bounding box coordinates in pixel space and "bbox_visib" represents the bounding box annotations in camera frame. The for the pose of each object

present in the scene, the translation vector is saved as position and the orientation is described by rotation_quaternion and rotation_euler parameters.


```
    "0":{
          "bbox_obj": [13.2979, 0.0, 89.6290, 106.1058],
          "bbox_visib": [0.0594, 0.0, 0.4001, 0.4737],
          "px_count_all": 50176,
          "visib_fract": 1.0,
          "class_label": 1,
          "class_name": "mustardbottle",
          "3d_pose_data": {
            "location": [-0.0720, -0.0651, 0.0967],
            "rotation_quaternion": [0.2180, 0.3747, 0.1361, -0.8908],
            "rotation_euler": [
              [-0.6242, 0.4903, -0.6083],
              [-0.2863, -0.8680, -0.4058],
              [-0.7269, -0.0791, 0.6822]
            ],
            "scale": [1.0, 1.0, 1.0],
            "dimensions": [0.0972, 0.0666, 0.1913]
          }
        }
```


Listing 4.5: Example annotation from scene_gt_info.json file

Finally, the fourth json file, blender_gt.json provides the information about the ground parameters for the blender's scene as depicted in the listing 4.6. Here "Distance_offset_follow_path_constraint" describes the offset value for the camera movement during the time of the rendered image. "Light_intensity" parameter represents the strength of light source while rendering the image. "Blur_value" and "Focal_length" parameters represents the depth of field value and the focal length of the blender's camera respectively.


```
    "0":{
          "Distance_offset_follow_path_constraint": -63.868607,
          "Light_intensity": 3.504551,
          "Blur_value": 10.0,
          "Focal_length": 20.0,
          "Camera_sensor_width": 36.0,
          "Camera_sensor_height": 24.0,
          "Deformation_value": 0.0,
          "Distance": {
            "mustardbottle": 0.518115
          },
          "Scene_bounds": {
            "min_bound": [-6.0, -6.0, -1.0785],
            "max_bound": [6.0, 6.0, 3.0785]
          }
        }
```


Listing 4.6: Example annotation from blender_gt.json file

In the listing 4.6, "Deformation_value" represents the amount of object deformation. A value of 0.0 represents no deformation. The "Distance" parameter provides the euclidean distance between the camera and each object present in the rendered image. Finally, "Scene_bounds" parameter represents the minimum and maximum values of the 3D coordinates of the complete scene.

4.2.2 Bright Constraint

This constraint is intended to generate datasets for bright lighting conditions. As we discussed before in normal constraint, we have created a intensity range (l_x,l_y) for normal lighting. In this constraint, we have created another range for the light intensity (l_{x1}, l_{y1}) such that $l_{x1} > l_x$ and $l_{y1} > l_y$. Figure 4.9 depicts the parameter selection in the configuration file where the (l_{x1}, l_{y1}) values can be provided and using the number of images parameter users can render the required dataset for images under bright lighting conditions as shown in the Figure 4.10. In this work we have used ($l_{x1} = 10$, $l_{y1} = 15$) as the range for bright lighting.

Figure 4.9: Configuration File Generator: Bright constraint

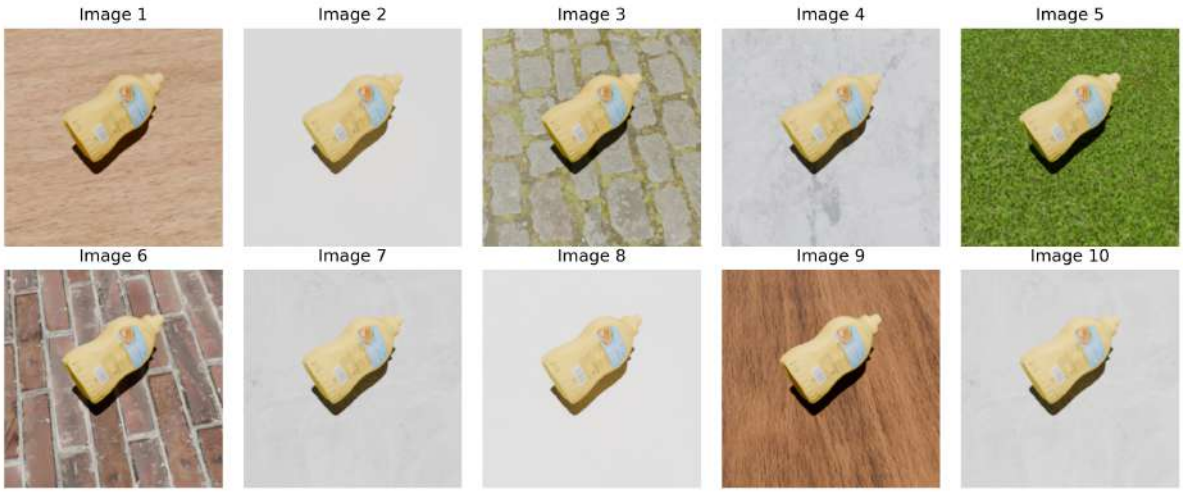


Figure 4.10: Example images for bright lighting constraint dataset

In this work, we have used the datasets generated under bright constraint settings, for testing the performance of deep learning models under high lighting conditions. Hence we kept the position and

orientation of the objects fixed while generating the datasets. Also the distance between the camera and object is fixed under bright lighting conditions. Depending on the requirement, users can also generate the datasets with random position and orientation of objects.

4.2.3 Dark Constraint

This constraint is intended to generate datasets for dark or low lighting conditions. Similar to the light intensity range (l_x,l_y) created for normal constraint, in dark constraint, we have created another range for the light intensity (l_{x2}, l_{y2}) such that $l_{x2} < l_x$ and $l_{y2} < l_y$. Figure 4.11 depicts the parameter selection in the configuration file where the (l_{x2}, l_{y2}) values can be provided and using the number of images parameter users can render the required dataset for images under dark or low lighting conditions as shown in the Figure 4.12. In this work we have used ($l_{x2} = 0.1$, $l_{y2} = 1.5$) as the range for dark lighting.

Figure 4.11: Configuration File Generator: Bright constraint

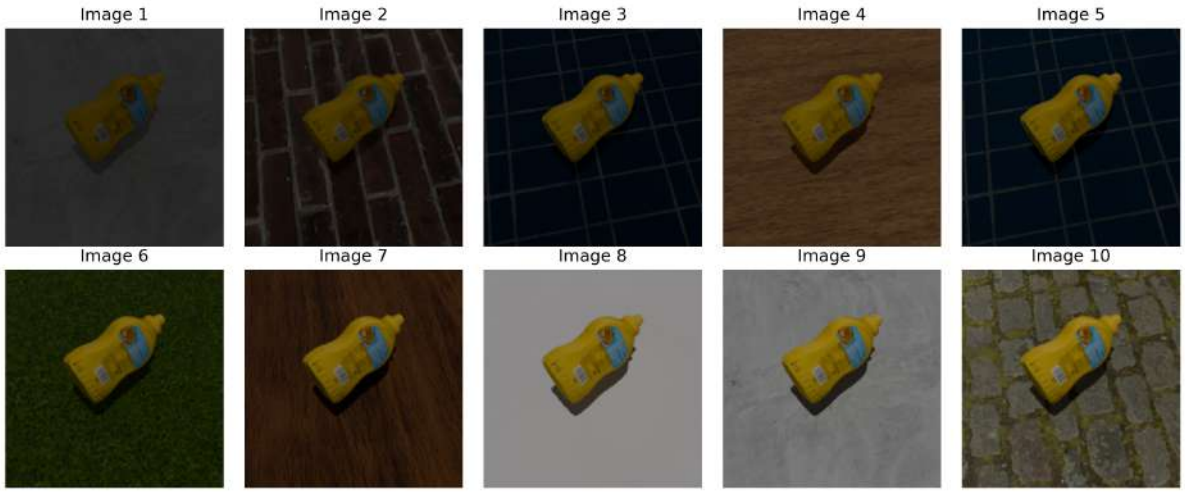


Figure 4.12: Example images for dark lighting constraint dataset

Similar to bright constraint, we have used the datasets generated under dark constraint settings, for testing the performance of deep learning models. Hence we kept the position, orientation and distance of the objects fixed while generating the datasets.

4.2.4 Far Constraint

This constraint is intended to generate datasets where objects in the rendered images appear far away from the camera. Similar to the distance range (d_x,d_y) created in normal constraint, in far constraint we have created another range for the distance as (d_{x1}, d_{y1}) such that $d_{x1} > d_x$ and $d_{y1} > d_y$. Figure 4.13 depicts the parameter selection in the configuration file where the (d_{x1}, d_{y1}) values can be provided and using the number of images parameter users can render the required dataset for images under far distance conditions as shown in the Figure 4.14. In this work we have used ($d_{x1} = -65$, $d_{y1} = -67$) as the range for far distance conditions.

Figure 4.13: Configuration File Generator: Far constraint

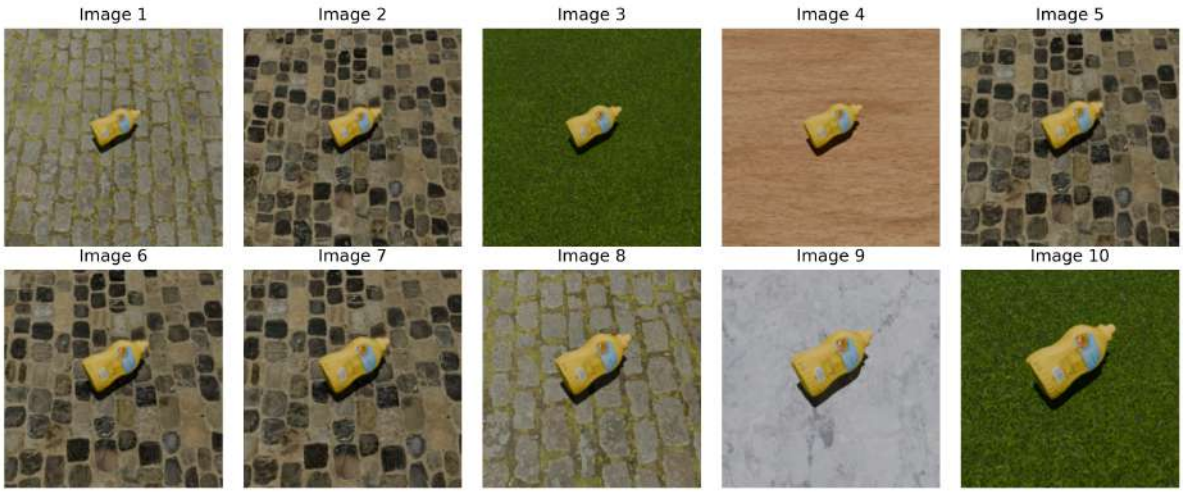


Figure 4.14: Example images for far distance constraint dataset

In this work, the datasets generated under far constraint settings are used for testing the performance of deep learning models when the objects are far away from the camera. Hence we kept the position and orientation of the objects fixed while generating the datasets and varied the distance alone. Also the lighting conditions are kept in normal light intensity range (l_x,l_y).

4.2.5 Near Constraint

This constraint is intended to generate datasets where objects in the rendered images appear very close to the camera. Similar to the distance range (d_x,d_y) created in normal constraint, in near constraint we have created another range for the distance as (d_{x2}, d_{y2}) such that $d_{x2} < d_x$ and $d_{y2} < d_y$. Figure 4.15 depicts the parameter selection in the configuration file where the (d_{x2}, d_{y2}) values can be provided and using the number of images parameter users can render the required dataset for images under far distance conditions as shown in the Figure 4.16. In this work we have used ($d_{x2} = -71$, $d_{y2} = -73$) as the range for near distance conditions.

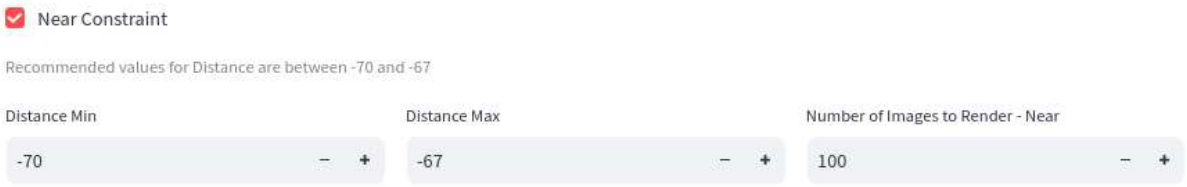


Figure 4.15: Configuration File Generator: Near constraint

Figure 4.16: Example images for near distance constraint dataset

Similar to far constraint, the datasets generated under near constraint settings are used for testing the performance of deep learning models when the objects are too close to the camera. Hence we kept the position and orientation of the objects fixed while generating the datasets and varied the distance alone. Also the lighting conditions are kept in normal light intensity range (l_x,l_y).

4.2.6 Blur Constraint

This constraint is intended to generate datasets where the objects in the rendered images appear blurry. To achieve this we have utilized the depth of field parameter of blender's camera for creating blurry images. Since the objective of this constraint is to test the performance of deep learning models in varying blur conditions, we have created a range for the depth of field value as (b_x,b_y) representing the blur values.

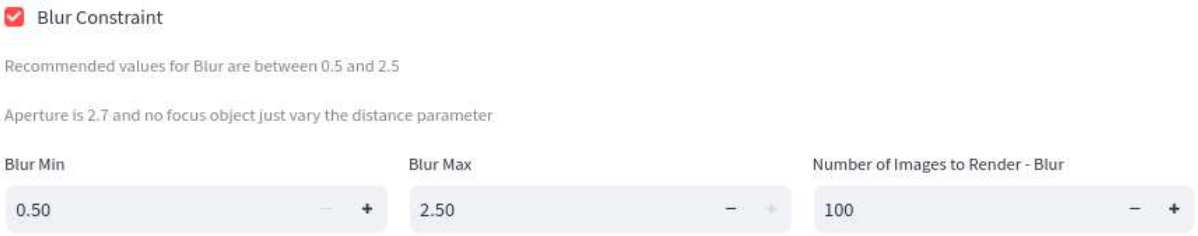


Figure 4.17: Configuration File Generator: Blur constraint

Figure 4.17 depicts the parameter selection in the configuration file where the (b_x,b_y) values can be provided and using the number of images parameter users can render the required dataset for images under blur constraint. Figure 4.18 depicts the example images for the dataset generated under blur constraint settings where ($b_x = 0.5$,$b_y = 2.5$) as the range for the depth of field value.

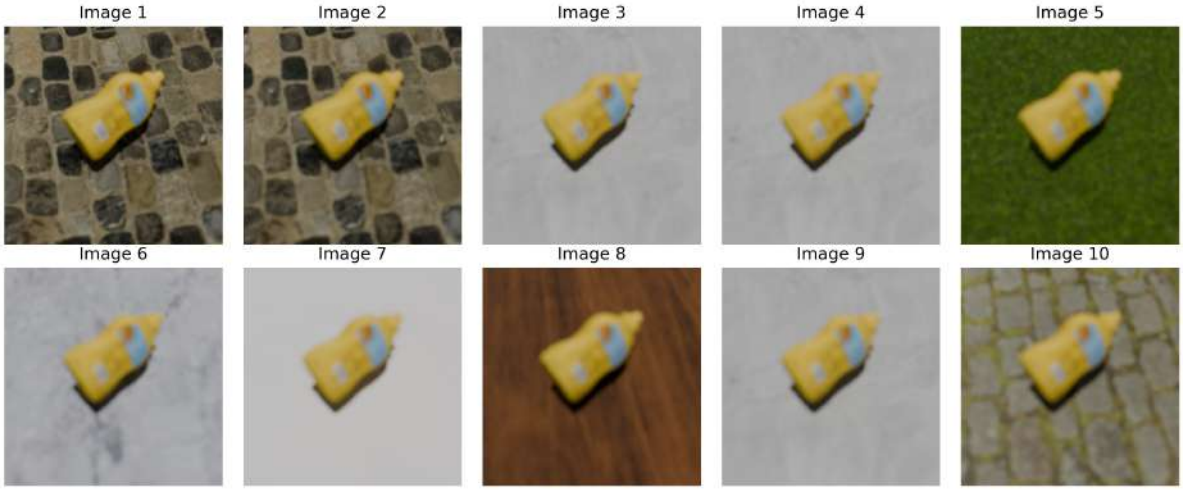


Figure 4.18: Example images for blur constraint dataset

By default, the depth of field value in blender is 10 and it produces clear images without any noise. Lowering the depth of field value increases the blurriness in the rendered images. A depth of field value of 0 produces complete blur image.

4.2.7 Textures Constraint

This constraint is intended to check the impact of unseen background textures on the performance of deep learning models. In the normal constraint 4.2.1, we have generated a dataset in nominal conditions and we consider the background textures present in this constraint belonging to class A. Here in textures constraint, we use different textures belonging to class B for generating the dataset. The textures in class B are not present in class A. (Textures_Constraint textures $\notin$ Normal_Constraint textures).

Figure 4.19: Configuration File Generator: Textures constraint

Figure 4.19 depicts the parameter selection in the configuration file for the textures constraint where the users can provide the path for the constraint textures folder and using the number of images parameter, the required dataset for images under textures constraint can be generated. Figure 4.20 depicts the example images for the dataset generated under textures constraint settings.

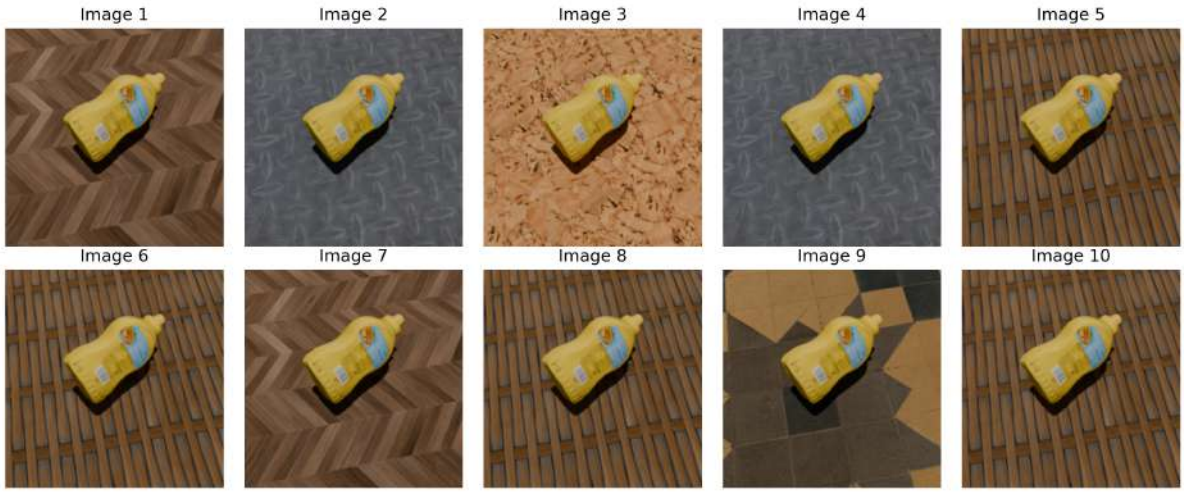


Figure 4.20: Example images for Textures constraint dataset

Instead of providing the path for constraint textures directory, the users can place the new textures directly into the constraint_textures_dir of the dataset generation tool package 4.1.

4.2.8 Deformation Constraint

This constraint is intended to check the impact of object's deformation on the performance of deep learning models. Figure 4.21, depicts the parameters for deformation constraint where users can select the parameters based on the requirement and generate the datasets. The dataset generation tool allows the user to deform the objects in 4 different ways (Bend, Twist, Crush, Dent) which can be selected using the Deformation type parameter.

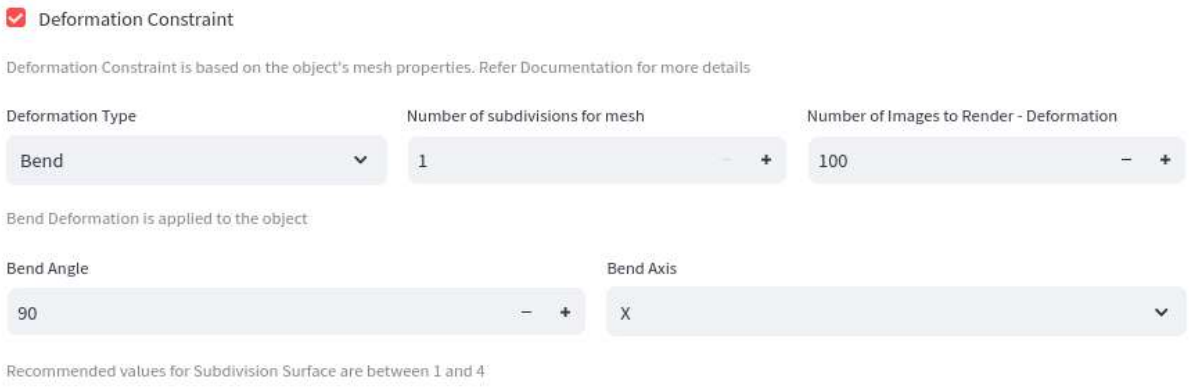


Figure 4.21: Configuration File Generator: Deformation constraint

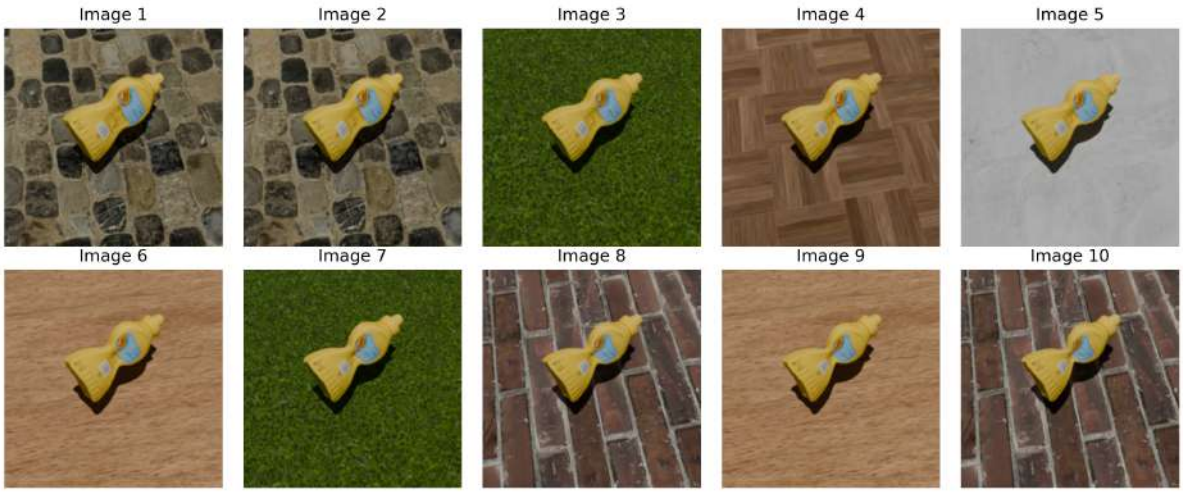


Figure 4.22: Example images for deformation constraint dataset

Sometimes the deformation of the objects will not be as expected. One of the reason for such scenario is due to the lack of enough mesh vertices in the 3D object. Number of subdivisions parameter is useful to increase the amount of vertices in the 3D object model. Using a higher value for subdivision increases the rendering time. Based on the deformation type, the other helpful parameters like, Bend angle and

Bend axis change. For example in the case of Bend deformation type, the users have the options to select the axis in which the object has to be bent and provide the angle to which the object should be bent. Figure 4.22 depicts the example images for the dataset generated under deformation constraint using dent deformation type.

4.3 Dataset Generation Tool Features

In this work as blender based synthetic dataset generation tool was developed to streamline the process of generating synthetic datasets ensuring the flexibility and ease of use. The developed tool is intended to generate synthetic datasets for diverse range of computer vision applications. This section describes the core functionalities of the developed dataset generation tool.

- **GUI based configuration file generator**: Allows for task specific parameter selection.
- **Computer vision tasks**: The tool is capable of generating datasets for classification, object detection and 6D object pose estimation tasks.
- **Pose estimation categories**: The tool can generate datasets for different object pose estimation categories like, SISO, SIMO, MISO and MIMO.
- **Diverse data representations**: The tool is capable of generating RGB images, Depth maps, Segmentation Maps, Surface Normal Maps and Point Cloud data.
- **Diverse object models**: The tool provides option for selecting the different 3D object models like, YCB, T-LESS, HOPE and Robocup@Work. Enables the users to integrate custom 3D object models in .obj or .ply format.
- **Annotation Format**: Supports different annotation formats. For 6D pose estimation task, user can select annotations in BOP or Elevation-Azimuth format. For object detection task, users can select YOLO or COCO annotation format.
- **Environment configuration**: Allows to control lighting , camera paths, background textures, camera intrinsic parameters.
- **Constraints**: The tool is capable of generating datasets for various environmental conditions like, bright lighting, dark lighting, far distance objects, near distance of objects, blurry images, various background textures, deformation of objects and normal conditions.
- **Custom Environment**: The tool also provides a custom designed environment depicting the C_025 laboratory of Hoshschule Bonn-Rhein-Sieg university of applied sciences.

4.4 Uncertainty Estimation in 6D Object Pose

In real world applications, models with high accuracy in 6D object pose estimation task alone are not sufficient as they do not provide a measure of confidence in the predictions. Estimating uncertainty in the predictions of deep learning models in 6D object pose estimation task, aids the researches in informed decision making and avoiding failures in safety critical tasks.

To this extent, literature provides different approaches for estimating the 6D pose and regression based 6D object pose estimation methods have proven to have high accuracy. The current state-of-the-art shows PoseCNN as the top performers for 6D pose estimation using RGB images and Shape-Match Loss function [7]. But this does not provide any direct measure of uncertainty estimates. PoseCNN uses VGG16 as the backbone deep learning architecture for estimating the 6D pose.

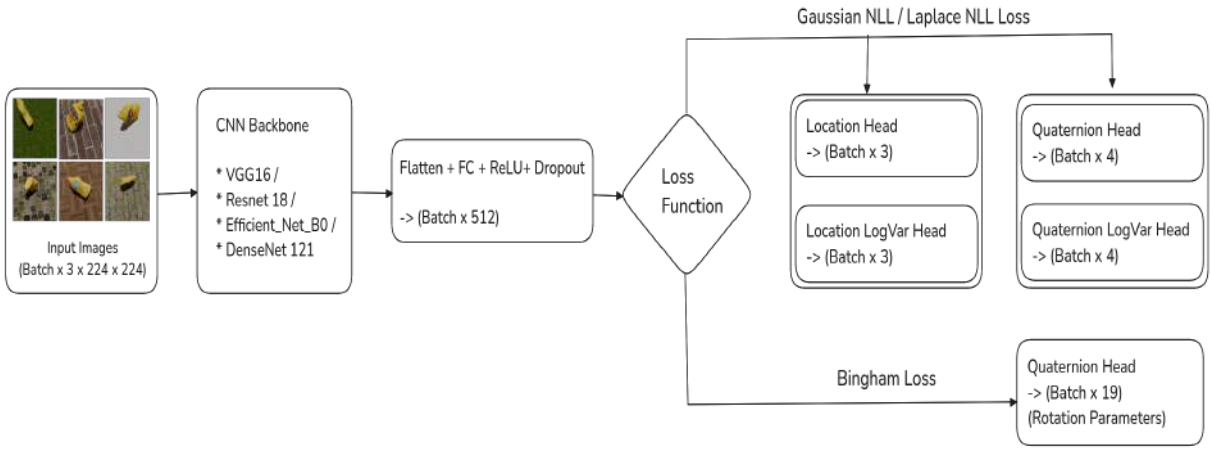


Figure 4.23: DNN Pipeline

In this work, we use VGG16 as the baseline model and using the loss functions described in previous chapter 3.3, we intend to train the models for estimating the predictive uncertainly. Figure 4.23, shows the training pipeline of different models. In this thesis work, Gaussian NLL, Laplace NLL, Bingham_loss are considered as the main loss functions and depending on the loss function the final layer of deep learning architecture is changed. For example, for Gaussian NLL the model predicts both mean and variance while Laplace NLL outputs mean and scale parameters. For incorporating this, we split the final layer of VGG16 architecture into four heads, Location Head, Quaternion Head, Location Log Variance Head and Quaternion Log Variance head. For ensuring the positivity of the variance we pass the variance values to F.softplus method in pytorch functional class. As we know, for a valid quaternion the prediction should be of unit norm. Hence we normalize the predicted quaternions before getting the final outputs from the model. The values obtained from the Location Log Variance Head and Quaternion Log Variance head represents the uncertainty parameters.

In the case of Bingham Loss, the model is expected to output 19 values as rotation parameters. and 3 parameters for location. Here in Bingham model, we use Gaussian NLL for the location parameters and therefore Location Head and Location Log Variance Head are used. For the 19 rotation parameters

as separate head is created where the final layer outputs 19 values. Inside the loss function these 19 rotation parameters are split into 4x4 M matrix where 16 values are used. The remaining 3 rotation parameters represent the concentration parameters (Z). The M matrix and Z vector are decomposed using Gram-Schmidt orthonormalization method. Here the fouth column of M matrix provides the predicted quaternion and the Z vector represents the uncertainty in rotations. For the location the uncertainty is represented by the values obtained from Location Log Variance Head.

5

Experimental Setup and Evaluation

This chapter presents an in depth explanation of the conducted experiments, datasets used for training and testing of deep learning models, metrics used for evaluation of 6D object pose and the predictive uncertainty.

5.1 Datasets

To study the impact of Lighting, Distance and Deformation of objects on Uncertainty Estimation Methods, the following synthetic datasets were generated using the developed blender based synthetic dataset generation tool. All images were rendered using CYCLES rendering engine with a resolution of 224 x 224 and 200 samples per pixel. The denoising parameter is also enabled while rendering the images to produce less noisy images and also for improving the rendering speed. For generating the datasets, an asymmetric object YCB_Mustard_Bottle is chosen from YCBV objects and this model is used for generating datasets for testing the hypothesis. All experiments are preformed on Single Instance of Single Object condition (SISO). The Mustard bottle object is placed at random position and random orientation while rendering images for training dataset and while generating test datasets the object is kept at fixed position and orientation across all the test samples. For additional details of the datasets refer to the section 5.1

S.No	**Dataset Name**	**Purpose**	**Samples**
1.	Normal Dataset	Training, Validation	Training - 2000, Validation - 500
2.	Normal Dataset	Testing	200
3.	Bright Lighting	Testing	200
4.	Dark Lighting	Testing	200
5.	Far Distance	Testing	200
6.	Near Distance	Testing	200
7.	Deformation	Testing	200

Table 5.1: Datasets used for conducting experiments and testing the hypothesis

5.2 Experimental Setup

For training the deep learning models using the loss functions discussed in section 3.3, the normal training dataset containing 2500 image samples with random location and orientation of object is split into 80% training data and 20% Validation data. VGG16 is chosen as a baseline model for all the experiments with default pre-trained weights. Depending on the loss function different hyper-parameters are selected and tuned based on the model's performance. Table 5.2 depicts the final optimal hyper-parameters.

Model Name	Architecture	Dataset Condition	Location Loss	Rotation Loss	Epochs	Batch Size	Learning Rate	Learning Rate Patience	Weight Decay	Dropout Probability
Gaussian Model	VGG16	Normal	Gaussian NLL	Gaussian NLL	100	64	0.0001	8	0.0001	0.25
Laplace Model	VGG16	Normal	Laplace NLL	Laplace NLL	100	64	0.0001	8	0.0001	0.2
Bingham Model	VGG16	Normal	Gaussian NLL	Bingham NLL	50	32	0.0001	4	0.0001	0.3
Ensembles Model	Ensembles	Normal	Gaussian NLL	Gaussian NLL	75 - 100	32 - 64	0.0001	6 - 8	0.0001	0.25 - 0.3
Conformal Prediction	VGG16	Normal	MSE Loss	Geodesic Loss	75	32	0.0001	5	0.00001	0.25

Table 5.2: Hyperparameters

The optimization is performed using AdamW optimizer and ReduceLROnPlateau learning rate scheduler is used with a lr_patience as mentioned in the hyper-parameters table 5.2. For Bingham loss function, Gaussian interpolation kernel and Gram-Schmidt orthogonalization techniques are used for converting the predictions of DNN model(19 parameters) to 4x4 rotation matrix and 1x3 (Z) matrix as discussed in the loss functions section 3.3.

For testing ensembles model 4 different DNN architectures (VGG16, Resnet18, Efficient_Net_B0 and DenseNet121) are trained separately using normal training dataset. The hyper-parameters are adjusted according to the performance of the DNN architecture and all these 4 DNN architectures are trained using Gaussian NLL loss. The hyper-parameter details are provided in the table 5.2. During test time, the test image passed through these 4 DNN architectures separately and the output predictions and variance are averaged to obtain as a single prediction.

Write something about conformal prediction if required. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla.

Since our experiments focus on evaluating the impact of Lighting,Distance and Deformation of the object, the location and orientation of YCB_Mustad_Bottle object is kept fixed during testing and 200 image samples are generated for each test constraint as mentioned in the table 5.1.

5.3 Evaluation Metrics

For evaluating the performance of deep learning models in the context of regression based 6D object pose estimation task, we have used standard regression metrics like, RMSE , Angular error, Interval Score and Validation Error. For evaluating the object's pose, we have used ADD and ADD-S metrics. For the uncertainty evaluation, we have used Entropy metric.

5.3.1 RMSE (Root Mean Squared Error)

To evaluate the accuracy of the predicted location coordinates of the deep learning models, we have used the standard Root Mean Squared Error between the ground truth location and the predicted location.

$$\mathrm{RMSE}_{\mathrm{Location}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{gt\_location}_i - \mathrm{pred\_location}_i\right)^2} \tag{5.1}$$

5.3.2 Angular error

For evaluating the accuracy of the predicted quaternion rotations, we used the angular error metric. Quaternions have double cover problem meaning q and -q represent the same orientation. So to measure the angular error of quaternions , we have computed the inverse cosine distance [31] between the ground truth quaternions and the predicted quaternions as mentioned in the equation 5.2

$$angular\_error = 2\arccos\left(\left|\mathbf{q}_{\mathrm{gt}}^{\top}\mathbf{q}_{\mathrm{pred}}\right|\right) \tag{5.2}$$

5.3.3 ADD (Average Distance of Model Points)

Average distance of model points provides the error in pose between the predicted pose and ground truth pose [32]. It requires the 3d model of the object for calculating the error. ADD metric is used for the objects with no indistinguishable views and is calculated using the equation 5.3 [32].

$$\mathrm{eADD}(\hat{P}, \bar{P}; M) = \mathrm{avg}_{x\in M}\left\|\bar{P}x - \hat{P}x\right\|_2 \tag{5.3}$$

5.3.4 ADD-S (Average Distance of Model Points for Symmetric objects)

Similar to ADD metric, ADD-S metric is used to compute the pose error for the objects with indistinguishable views [32]. It also requires the 3d model of the object for computing the error between predicted pose and the ground truth pose. It is calculated using the equation 5.4 [32].

$$\mathrm{eADD\text{-}S}(\hat{P}, \bar{P}; M) = \mathrm{avg}_{x1\in M}\mathrm{min}_{x2\in M}\left\|\bar{P}x1 - \hat{P}x2\right\|_2 \tag{5.4}$$

5.4 Uncertainty Metrics

Since we are evaluating the performance of the models under different distributions, we need proper metrics for comparing the uncertainties provided by different distributions. Various methods have been proposed in the literature and in this work we use Interval score, Entropy and Expected Calibration Error metrics for evaluating the uncertainty.

5.4.1 Interval Score

Interval score is used as a proper scoring rule for assessing the quality of probabilistic forecasts [14]. Interval score is negatively oriented meaning lower interval score represents better performance of the deep learning model. Interval score is computed using the equation 5.5 [14], [33].

$$S_{\alpha}^{int}(l,u;x) = (u-l) + \frac{2}{\alpha}(l-x)\mathbb{1}\{x<l\} + \frac{2}{\alpha}(x-u)\mathbb{1}\{x>u\} \tag{5.5}$$

5.4.2 Entropy

Entropy can be used to asses the degree of uncertainty in the DNN model's expectations. It is a metric for distributional randomness or uncertainty [16], [1]. The calculation of entropy depends on the type of output distributions of DNN models. In this work we compute distributional entropy for three different distributions namely Gaussian , Laplace and Bingham distributions.

For Gaussian distribution we mean and variance parameters for computing the entropy. For Laplace distribution, we use mean and scale parameters for computing the entropy. Both Gaussian and Laplace distributions are available in scipy package and the entropy values can be computed directly using this package.

For Bingham distribution, the uncertainty is represented by the concentration parameters Z as discussed in the section 3.3.4. Also computing entropy for Bingham distribution is not straight forward and it requires a normalization constant and its gradients with respect to the concentration parameters. In general the normalization constants and gradients are obtained from a pre-computed look up table and the entropy can be approximated without any numerical integration [17]. The entropy of Bingham distribution can be computed using the equation 5.6 [17]. Here F is the normalization constant and ∇F is the gradients with respect to the concentration parameters Λ.

$$h(f) = -\int_{x \in S^d} f(x) \log f(x) = \log F - \Lambda \cdot \nabla F / F \tag{5.6}$$

6

Results and Discussion

This chapter provides the comparative evaluation of the uncertainty in the predictions of trained deep learning models under various test constraints. All the models are trained using normal constraint dataset such that the models generalize properly and do not over-fit to the training data. Since we cannot compute interval score for Bingham loss, in this work, we use entropy metric for comparing the performance of the deep learning models.

6.1 Comparative evaluation of different uncertainty estimation methods on various lighting conditions

RQ2: What is the impact of lighting conditions on Uncertainty Estimation Methods in 6D Object Pose Estimation Task

6.1.1 Datasets Used

(a) Normal constraint (b) Bright constraint (c) Dark constraint

Figure 6.1: Datasets generated for lighting constraint

To evaluate the performance of different deep learning models in various lighting constraints like, normal,bright and dark, we have generated 3 different datasets for each constraint by modifying the

intensity of the light source (Sun) in the scene setup 4.2a. For generating the normal constraint dataset, the we have used the normal lighting range as, ($l_x = 3.0$, $l_y = 5.0$) and generated 200 samples as the first test dataset. For bright constraint, we have used ($l_{x1} = 10$, $l_{y1} = 15$) as the light range and generated the second test dataset with 200 samples. Finally, for the dark constraint, we have used ($l_{x2} = 0.1$, $l_{y2} = 1.5$) as the light range and generated the third test dataset with 200 samples.

6.1.2 Hypothesis

The uncertainty in bright and dark constraints is expected to be higher than the uncertainty in normal constraint.

Assumptions

1. All the DNN models are trained on normal constraint dataset in which the light intensity is in the range of (l_x, l_y).
2. The range of light intensity in bright constraint (l_{x1}, l_{y1}) is greater than the range of light intensity in normal constraint (l_x, l_y).
3. The range of light intensity in dark constraint (l_{x2}, l_{y2}) is less than the range of light intensity in normal constraint (l_x, l_y).

$$l_{x2} < l_x < l_{x1} \quad \text{and} \quad l_{y2} < l_y < l_{y1}$$

Test Constraints

1. Test scenario1 :- The datasets generated under bright constraint are tested with models trained on normal constraint dataset.
2. Test scenario2 :- The datasets generated under dark constraint are tested with models trained on normal constraint dataset.

6.1.3 Observation on lighting constraints

In lighting constraint, two experiments were performed in which two objects, YCB_Mustard_Bottle and YCB_Tomato_Soup_Can objects were trained and tested separately. Table 6.1 shows the results of test metrics for YCB_Mustard_Bottle under normal,bright and dark constraints. For evaluating the performance of deep learning models in estimating the 6D object pose, we use RMSE for location, Angular error metric for quaternions and ADD/ADD-S metrics for the overall pose. From the location RMSE results, it is observed that, the performance of Laplace model is good in estimating the location parameters compared to Gaussian Bingham and Ensembles models. From the Angular metric results, it is observed that, The model, trained on Laplace loss showed good performance of estimating the quaternions with less than 5.6 deg across all the three constraints. The ADD metric also shows the same.

Lighting Constraint					
Metric	Constraint	Gaussian Model	Laplace Model	Ensembles Model	Bingham Model
Location RMSE	Normal	0.016	**0.004**	0.01	0.009
	Bright	0.017	**0.006**	0.013	0.024
	Dark	0.035	**0.01**	0.015	0.013
Angular Error degrees	Normal	5.487	**2.664**	3.528	3.499
	Bright	6.511	**2.133**	9.983	2.786
	Dark	10.543	**5.602**	6.661	11.756
Location Interval Score	Normal	0.711	0.123	0.638	-
	Bright	0.723	0.127	0.701	-
	Dark	0.792	0.141	0.696	-
Quaternion Interval Score	Normal	0.968	1.1	1.059	-
	Bright	0.991	1.124	1.197	-
	Dark	1.082	1.197	1.11	-
ADD	Normal	0.028	**0.008**	0.018	0.016
	Bright	0.03	**0.01**	0.025	0.043
	Dark	0.062	**0.017**	0.027	0.022

Table 6.1: Metrics for YCB_Mustard_Bottle under lighting constraints

For evaluating the uncertainty we use entropy metric and we have computed entropy for location and quaternion predictions separately. In this work we evaluate the performance of the uncertainty estimation methods and the impact of test constraints on uncertainty estimation qualitatively using box-plots.

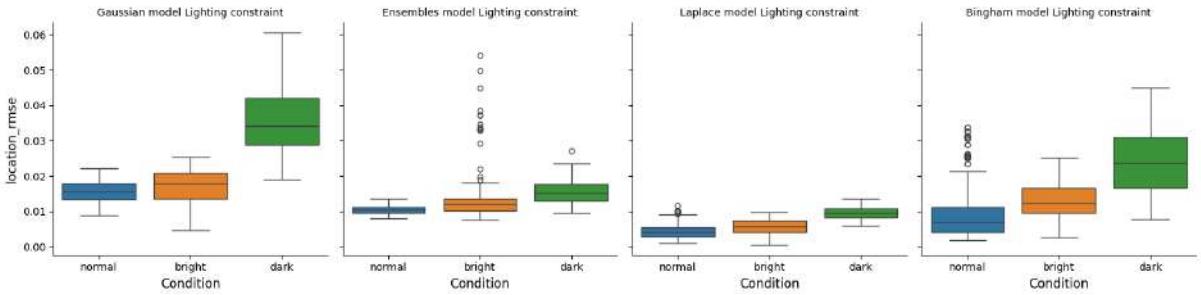


Figure 6.2: Box plot for location RMSE results from normal,bright and dark constraints

For evaluating the overall performance of the deep learning model, we need to consider both accuracy and the predictive uncertainty. Figure 6.2 depicts the distribution of the RMSE values for normal,bright and dark constraint. Here we can observe that test samples in Laplace model have lower RMSE values compared to other models. Now from figure 6.3, we investigate the behavior of entropy. Here we expect the models to have low uncertainty in normal constraint and high uncertainty in bright and dark constraint. This is because the models are trained on normal constraint dataset and both bright and dark constraint datasets were never seen by the models before.

From the figure 6.3, we observe that all four models have clear separation in the box-plots for normal, bright and dark constraints. This means that all the four models are good at estimating the uncertainty. But since the Laplace Loss has the best location RMSE value, we consider Laplace model as the best performing model in predicting location parameters.Also we can say that there is impact of lighting

conditions on the four DNN models and as lighting varies the uncertainty increases.

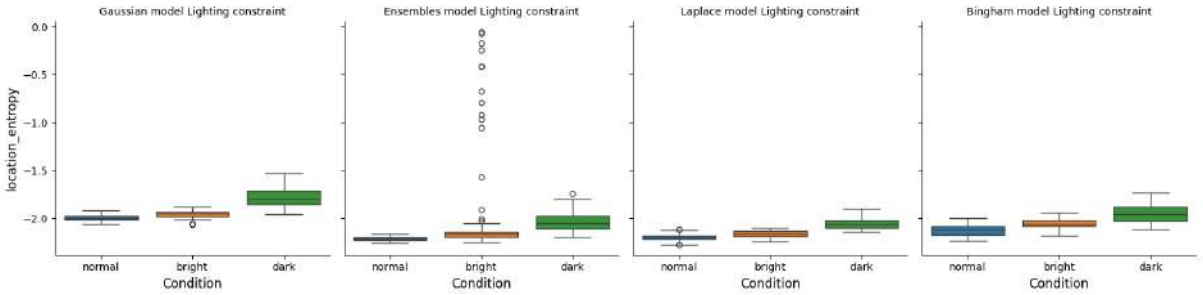


Figure 6.3: Box plot for location entropy results from normal,bright and dark constraints

In the case of quaternions we consider both angular error and quaternion entropy for evaluating the best performing model. Figure 6.4 depicts the distribution of the angular error values for normal,bright and dark constraint. Here we can observe that both Laplace and Bingham models have more test samples under 5 deg angular error across all test constraints. But to select the best performing model we need to investing the uncertainty from the figure 6.5.

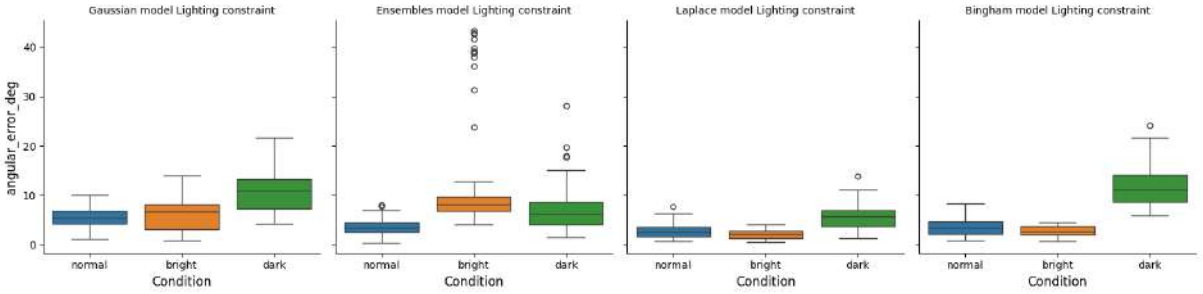


Figure 6.4: Box plot for angular error results from normal,bright and dark constraints

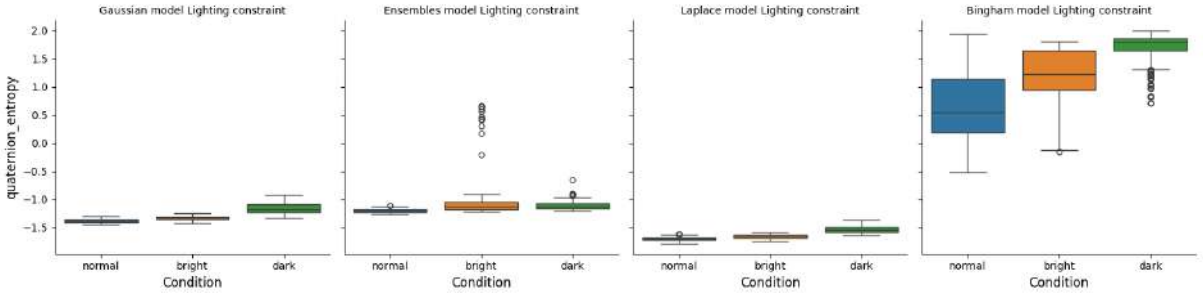


Figure 6.5: Box plot for quatenion entropy results from normal,bright and dark constraints

From the figure 6.5, we can observe that, in Laplace model, the separation of distribution between normal and bright constraint is less. But in Bingham model, we can see a clear separation between

the uncertainties of normal, bright and dark constraints. Hence in lighting constraint, we consider the Bingham model as the best model. Based on the insights, the performance of overall pose estimation in lighting constrained can be improved using Laplace NLL loss for location parameters and Bingham loss for rotation parameters.

In the second experiment, we have trained the DNN models, using YCB_Tomato_Soup_Can object under normal constraint settings and tested the performance on YCB_Tomato_Soup_Can object under bright and dark constraint settings. The datasets used and the results for the metrics in lighting constraint is provided in the Appendix-B A

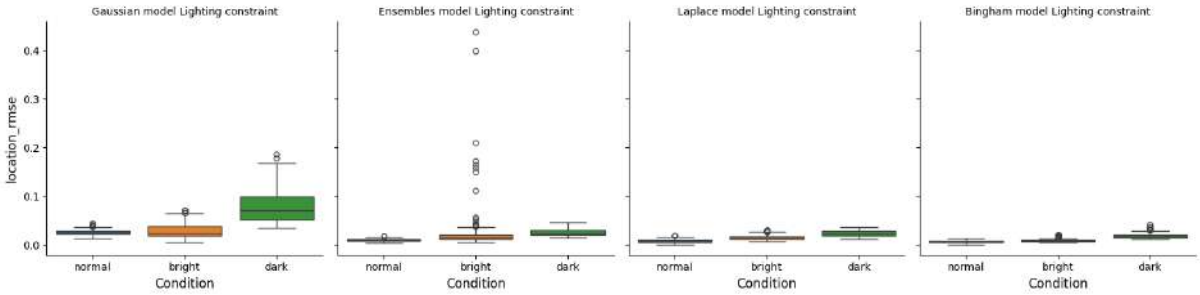


Figure 6.6: Box plot for location RMSE results from normal,bright and dark constraints for tomato soup can

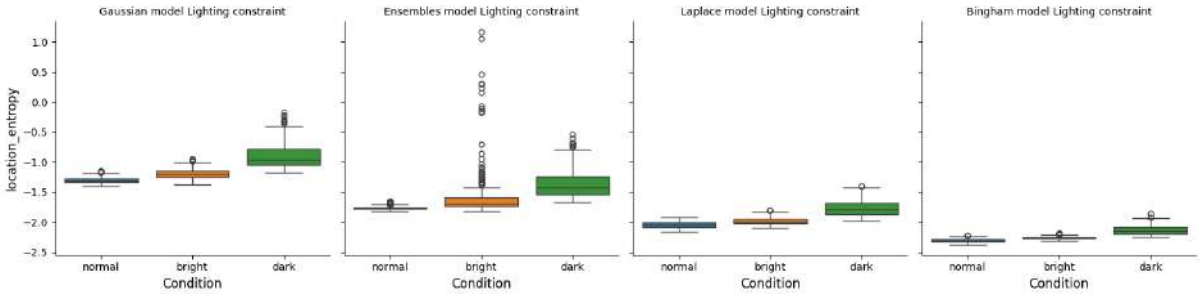


Figure 6.7: Box plot for location entropy results from normal,bright and dark constraints for tomato soup can

From the figure 6.6, we can see that both Laplace model and Bingham model have low RMSE values compared to Gaussian and Ensemble models. From figure 6.7, we can see the separation in the distributions of normal,bright and dark constraints in all four models. Since Bingham loss has more test samples with low RMSE value, we consider Bingham model as the best performing model for predicting location parameters in lighting constraint.

From the figure 6.8, we can see that Bingham model has low angular error values compared to Gaussian, Laplace and Ensemble models. From figure 6.9, we can see the separation in the distributions of normal and dark constraints in Bingham model. For bright the uncertainty is not as expected. But it is possible that the model is still performing good in bright lighting conditions. So on overall we consider Bingham

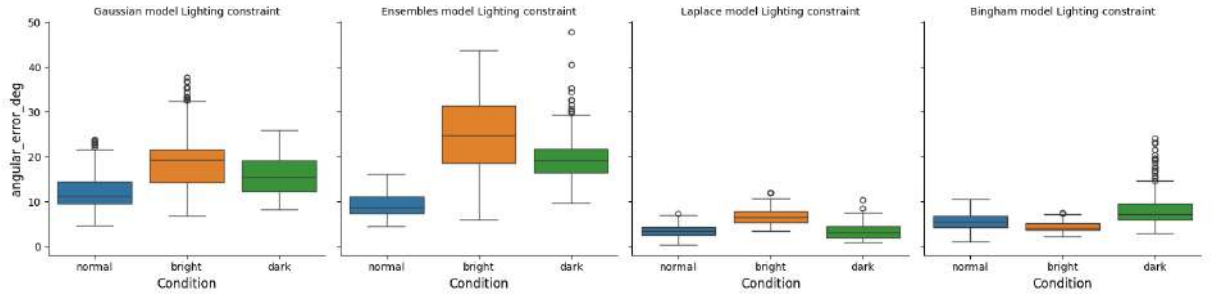


Figure 6.8: Box plot for angular error results from normal,bright and dark constraints for tomato soup can

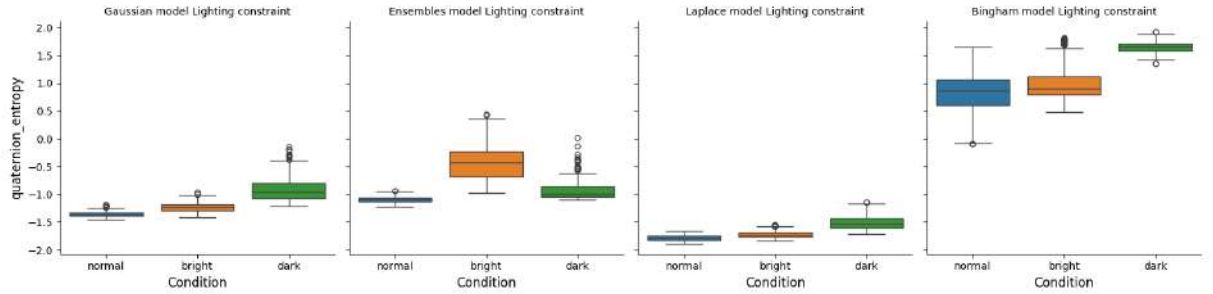


Figure 6.9: Box plot for quatenion entropy results from normal,bright and dark constraints for tomato soup can

model as the best performing model for predicting location and rotation parameters in lighting constraint. Gaussian, Laplace and Ensemble model, failed to perform well with symmetric objects

6.2 Comparative evaluation of different uncertainty estimation methods on varying object's distance conditions

RQ3: What is the impact of object's distance on Uncertainty Estimation Methods in 6D Object Pose Estimation Task

6.2.1 Datasets Used

(a) Normal constraint (b) Far constraint (c) Near constraint

Figure 6.10: Datasets generated for distance constraint

6.2.2 Hypothesis

The uncertainty in far and near constraints is expected to be higher than the uncertainty in normal constraint.

Assumptions

1. All the DNN models are trained on normal constraint dataset in which the distance of objects to the camera is in the normal range of (d_x, d_y).

2. The range of object's distance in far constraint (d_{x1}, d_{y1}) is greater than the range of object's distance in normal constraint (d_x, d_y).

3. The range of object's distance in near constraint (d_{x2}, d_{y2}) is less than the range of object's distance in normal constraint (d_x, d_y).

$$d_{x2} < d_x < d_{x1} \quad \text{and} \quad d_{y2} < d_y < d_{y1}$$

Test Constraints

1. Test scenario1 :- The datasets generated under far constraint are tested with models trained on normal constraint dataset.
2. Test scenario2 :- The datasets generated under near constraint are tested with models trained on normal constraint dataset.

6.2.3 Observations on distance constraints

Distance Constraint					
Metric	Constraint	Gaussian Model	Laplace Model	Ensembles Model	Bingham Model
Location RMSE	Normal	0.016	**0.004**	0.01	0.009
	Far	0.025	**0.02**	0.021	0.053
	Near	0.04	**0.026**	0.022	0.027
Angular Error degrees	Normal	5.487	2.664	3.528	**3.499**
	Far	18.282	21.94	16.511	**11.13**
	Near	7.618	12.595	9.691	**6.409**
Location Interval Score	Normal	0.711	0.123	0.638	-
	Far	0.798	0.153	0.695	-
	Near	0.757	0.135	0.689	-
Quaternion Interval Score	Normal	0.968	1.1	1.059	-
	Far	1.106	1.249	1.188	-
	Near	1.005	1.136	1.244	-
ADD	Normal	0.028	**0.008**	0.018	0.016
	Far	0.047	**0.036**	0.039	0.047
	Near	0.069	0.046	**0.04**	0.093

Table 6.2: Metrics for YCB_Mustard_Bottle under distance constraints

Similar to the lighting constraint, in distance constraint, two experiments were performed in which YCB_Mustard_Bottle and YCB_Tomato_Soup_Can objects were trained and tested separately. Table **??** shows the results of test metrics for YCB_Mustard_Bottle under normal,far and near constraints. In the table, from the location RMSE results, it is observed that, the performance of Laplace model is good in estimating the location parameters compared to Gaussian Bingham and Ensembles models. From the Angular metric results, it is observed that, optimal performance is achieved by Bingham model in predicting the quaternions. Even though Laplace model has less angular error in normal constraint, it failed to learn the distance of objects and showed high angular error in both far and near constraints.

If we check the results of ADD metric, the results show that the performance of Laplace model is good. But this cannot be considered as the final result, because of the high angular error. In general, ADD metric provides low values even if the object is rotated 180 deg on its principal axis. This could be one of the possible reasons for observing low ADD values in Laplace model even with high angular error. In the case of distance constraint, we consider the Bingam Model to have the best performance in estimating 6D pose because of its low angular error when compared to Gaussian, Laplace and ensemble models.

For evaluating the overall performance of the deep learning model, we need to consider both accuracy and the predictive uncertainty. Figure 6.11 depicts the distribution of the RMSE values for normal,far

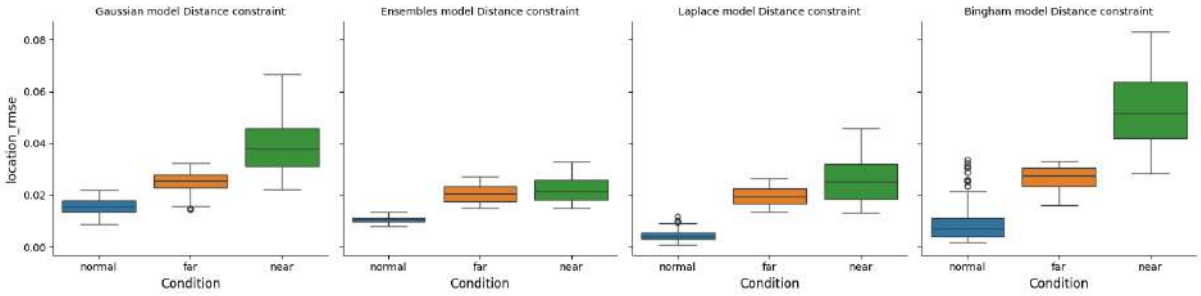


Figure 6.11: Box plot for location RMSE results from normal,far and near constraints

and near constraints. Here we can observe that test samples in Ensemble model have lower RMSE values compared to other models. Now from figure 6.12, we need to investigate the behavior of entropy. Here we expect the models to have low uncertainty in normal constraint and high uncertainty in far and near constraint. This is because the models are trained on normal constraint dataset and both far and near constraint datasets were never seen by the models before.

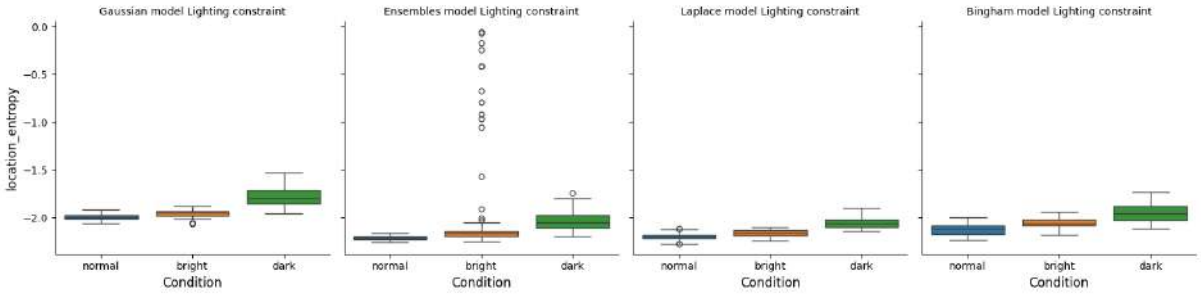


Figure 6.12: Box plot for location entropy results from normal,bright and dark constraints

From the figure 6.12, we observe that both Gaussian and Bingham models show clear separation for the uncertainty. But due to relatively high RMSE values we do not consider them. In the figure 6.12, we can observe clear separation of uncertainties in Ensembles model compared to Laplace models. So we consider Ensembles model to be the best performing model for prediciton location parameters.

In the case of quaternions we consider both angular error and quaternion entropy for evaluating the best performing model. Figure 6.13 depicts the distribution of the angular error values for normal,far and near constraints. Here we can observe that both Laplace and Bingham models have more test samples under 5 deg angular error across all test constraints. But to select the best performing model we need to investing the uncertainty from the figure 6.14.

From the figure 6.14, we can observe that, in both Laplace and Bingham model, the separation of distribution between normal and far and near constraints is almost same. To understand the behavior better we need a scatter plot.

From the figure 6.15, we can see that in Laplace loss, even though there is a separation in the

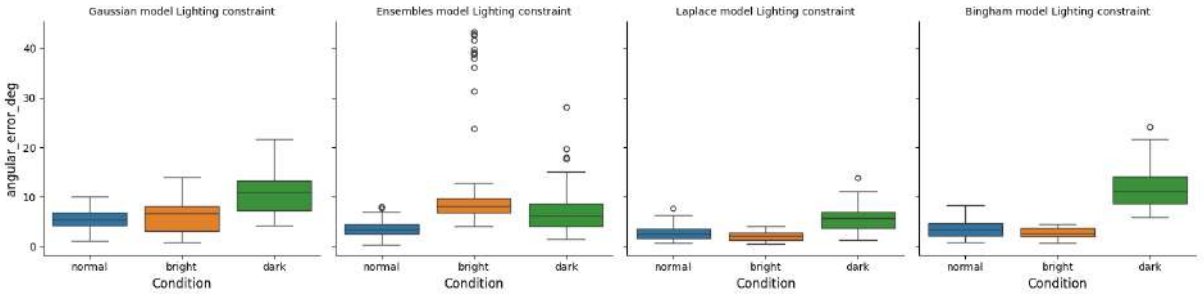


Figure 6.13: Box plot for angular error results from normal,far and near constraints

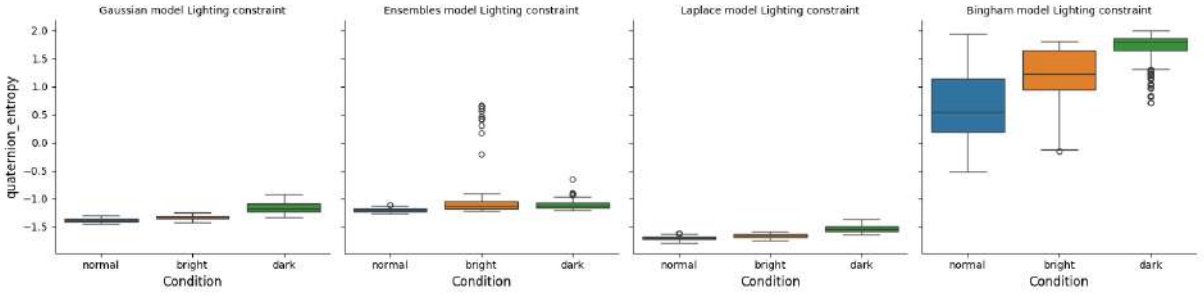


Figure 6.14: Box plot for quatenion entropy results from normal,far and near constraints

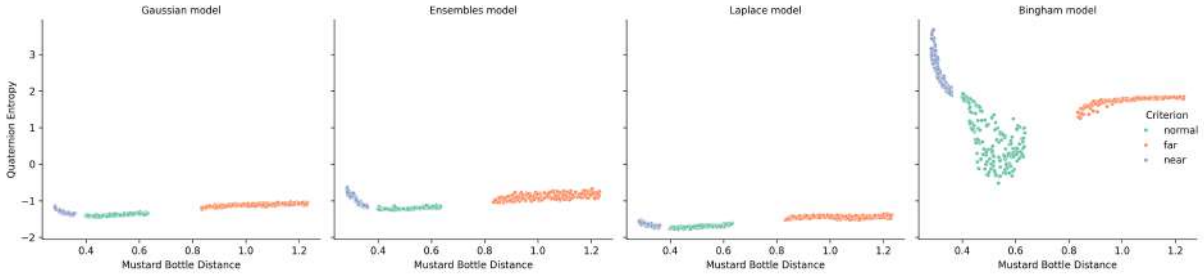


Figure 6.15: Scatter plot for quatenion entropy results from normal,far and near constraints

uncertainties of test constraints, the range or spread is very less. On the other hand, if we look the distribution in Bingham model, we can see clear separation in uncertainties and also we can observe that the uncertainty is increasing with variation in object's distance. From the figure6.14, we know that Bingham model also has low angular error values. Hence in distance constraint, we consider Bingham model as the best performing model.

In the second experiment, we have trained the DNN models, using YCB_Tomato_Soup_Can object under normal constraint settings and tested the performance on YCB_Tomato_Soup_Can object under far and near constraint settings. The datasets used and the results for the metrics in lighting constraint is provided in the Appendix-B A

From the figure 6.16, we can see that Ensemble model has low RMSE values compared to Gaussian,

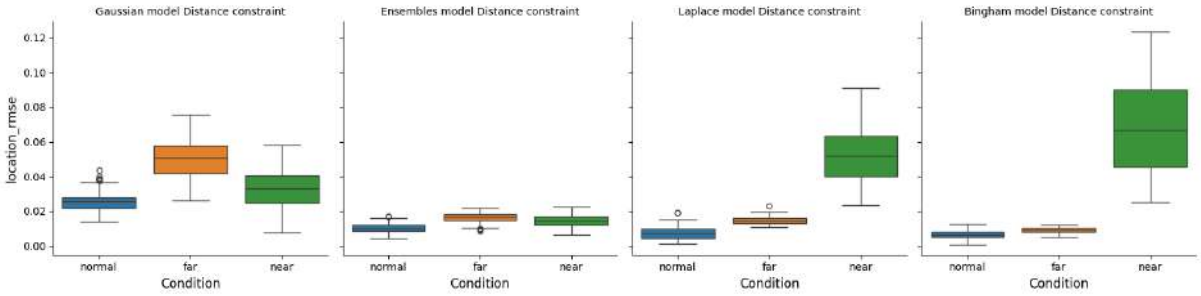


Figure 6.16: Box plot for location RMSE results from normal,far and near constraints for tomato soup can

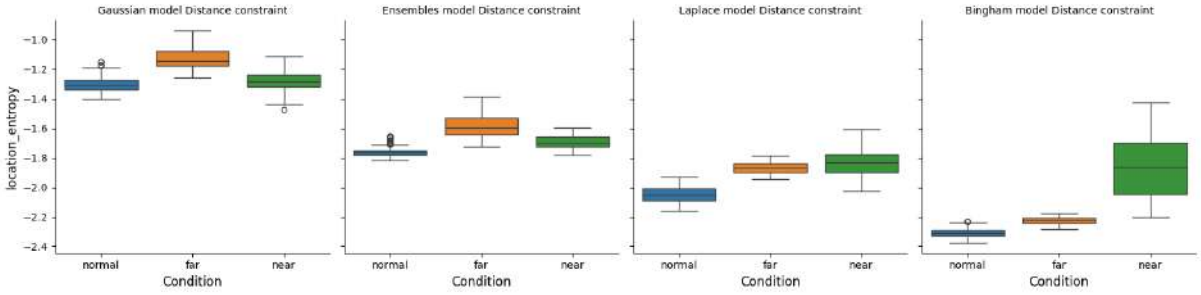


Figure 6.17: Box plot for location entropy results from normal,far and near constraints for tomato soup can

Laplace and Bingham models. From figure 6.17, we can see the separation in the distributions of normal,far and near constraints in all four models. Since Ensemble model has more test samples with low RMSE values, we consider Ensemble model as the best performing model for predicting location parameters in distance constraint.

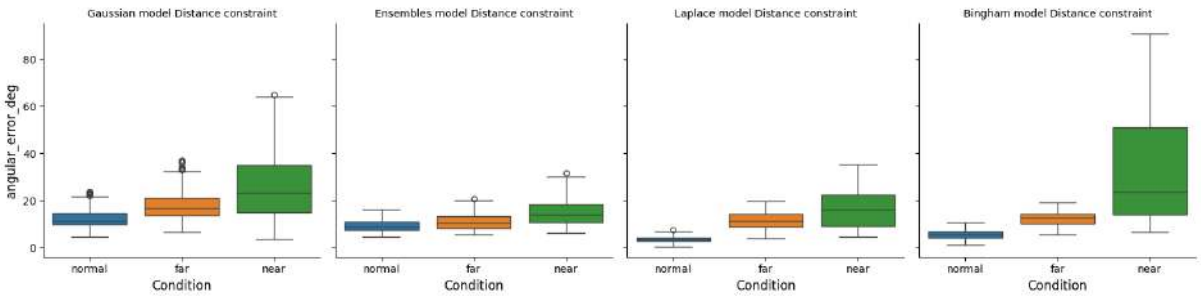


Figure 6.18: Box plot for angular error results from normal,far and near constraints for tomato soup can

From the figure 6.18, we can see that both Laplace and Bingham models have low angular error values compared to Gaussian, Laplace and Ensemble models in normal constraint. The performance of all four models is relatively bad in near and far constraint when compared to far and normal constraints. From figure 6.19, we can see all the entropy distributions of normal, far and near are squished and also there is

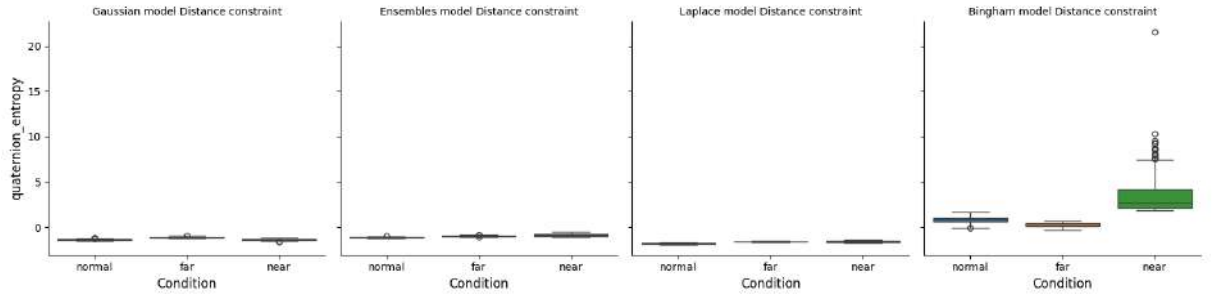


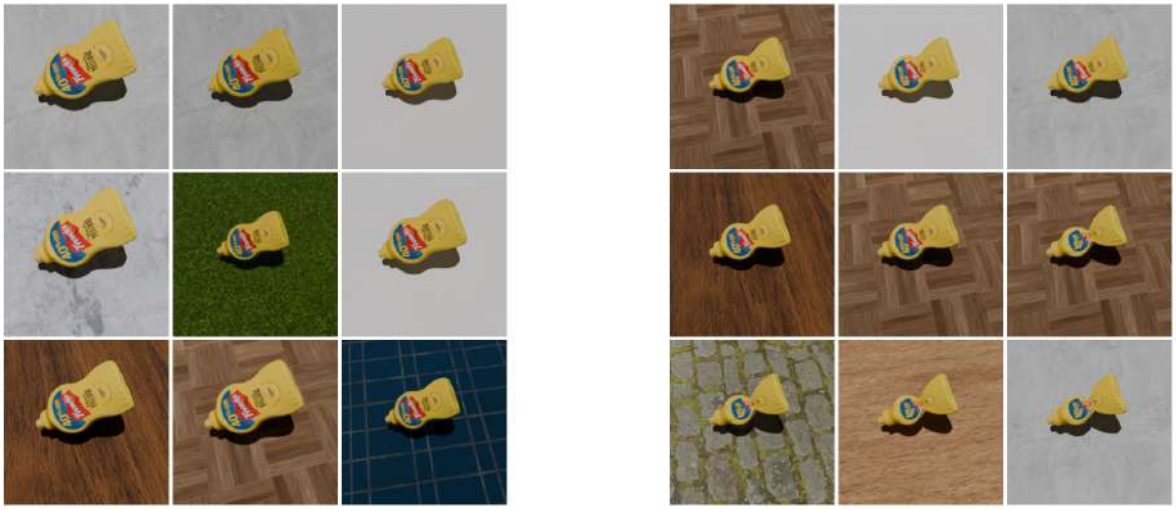
Figure 6.19: Box plot for quatenion entropy results from normal, far and near constraints for tomato soup can

not evident separation between the entropy distributions of normal,bright and dark. This shows that the performance of models is not good with symmetric objects.

6.3 Comparative evaluation of different uncertainty estimation methods on varying object's deformation conditions

RQ3: What is the impact of object's deformation on Uncertainty Estimation Methods in 6D Object Pose Estimation Task

6.3.1 Datasets Used



(a) Normal constraint

(b) Deformation constraint

Figure 6.20: Datasets generated for deformation constraint

6.3.2 Hypothesis

The uncertainty in deformation constraint is expected to be higher than the uncertainty in normal constraint.

Assumptions

1. All the DNN models are trained on normal constraint dataset in which the objects appear in proper shape without any deformation.

2. The appearance of the object present in deformation constraint is distorted by deforming the actual objects present in the scene.

Test Constraints

Test scenario :- The datasets generated under deformation constraint are tested with models trained on normal constraint dataset.

6.3.3 Observations on deformation constraint

In the case of deformation constraint, two experiments were performed in which YCB_Mustard_Bottle and YCB_Tomato_Soup_Can objects were trained on normal constraint separately and tested using normal and deformation constraints separately. Table 6.3 shows the results of test metrics for YCB_Mustard_Bottle under normal and deformation constraints. In the table, from the location RMSE results, it is observed that, the performance of Laplace model is good in estimating the location parameters compared to Gaussian Bingham and Ensembles models. From the Angular metric results, it is also observed that, optimal performance is achieved by the Laplace model in predicting the quaternions with low angular error. We can observe the same, from the ADD metric results. It is also noticed that there is no difference for the ADD metric in Bingham model for both normal and deformation constraint. An in-depth analysis on uncertainty estimation is required to select the best performing model.

Deformation Constraint					
Metric	Constraint	Gaussian Model	Laplace Model	Ensembles Model	Bingham Model
Location RMSE	Normal	0.016	**0.004**	0.01	0.009
	Deformation	0.013	**0.008**	0.01	0.007
Angular Error degrees	Normal	5.487	**2.664**	3.528	3.499
	Deformation	16.068	**14.13**	15.209	15.029
Location Interval Score	Normal	0.711	0.123	0.638	-
	Deformation	0.693	0.129	0.634	-
Quaternion Interval Score	Normal	0.968	1.1	1.059	-
	Deformation	1.133	0.951	1.143	-
ADD	Normal	0.028	0.008	0.018	**0.016**
	Deformation	0.025	0.02	0.021	**0.016**

Table 6.3: Metrics for YCB_Mustard_Bottle under deformation constraint

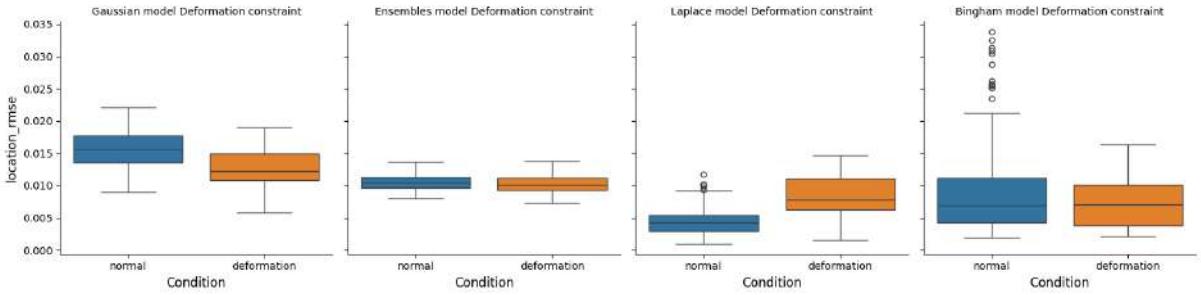


Figure 6.21: Box plot for location RMSE results from normal and deformation constraints

For evaluating the overall performance of the deep learning model, we need to consider both accuracy

and the predictive uncertainty. Figure 6.21 depicts the distribution of the RMSE values for normal and deformation constraints. Here we can observe that test samples in both Laplace model and Bingham model have lower RMSE values compared to other two models. But in Bingham model, we can clearly see that the RMSE values in deformation constraint are low when compared to the RMSE values of deformation constraint in Laplce model. So in Deformation constraint, we consider the performance of Bingham Model to be the best in predicting the location parameters. Next from figure 6.22, we need to investigate the behavior of entropy. Here we expect the models to have low uncertainty in normal constraint and high uncertainty in deformation constraint. This is because the models are trained on normal constraint dataset and the deformation constraint dataset was never seen by the models before.

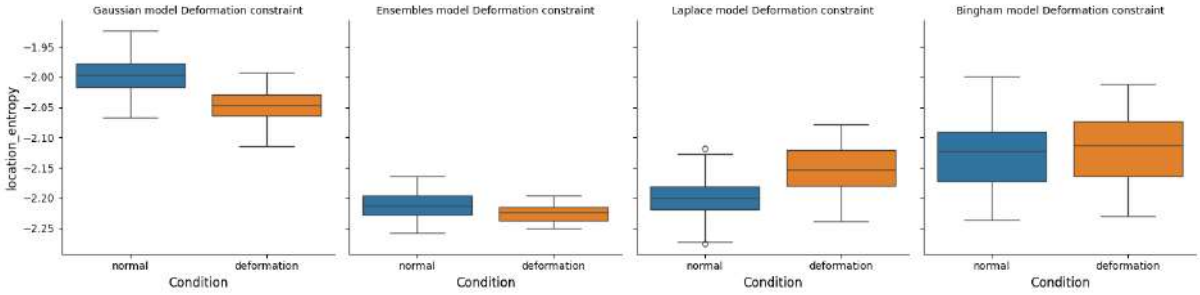


Figure 6.22: Box plot for location entropy results from normal and deformation constraints

From the figure 6.22, we observe that a clear separation of uncertainty distribution for normal and deformation constraint is seen in Laplace model. In Bingham model, despite having low RMSE values, we cannot see a clear separation in the distribution of uncertainties. Hence it does not satisfy our hypothesis. So in the case of deformation constraint, we select Laplace model as the best performing model for predicting the location parameters.

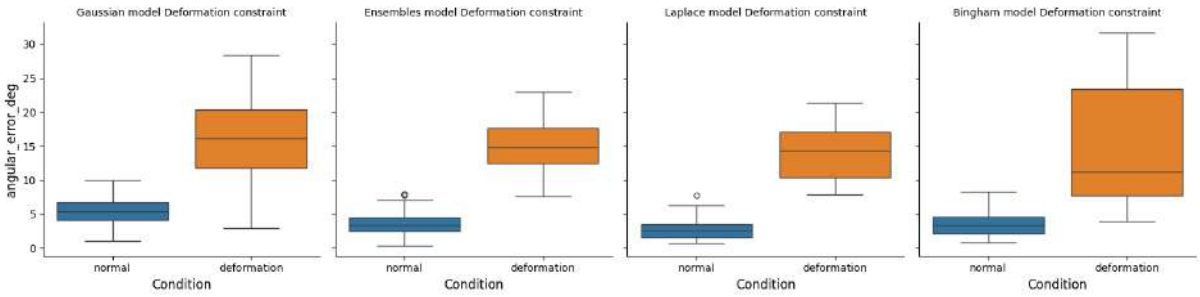


Figure 6.23: Box plot for angular error results from normal and deformation constraints

In the case of quaternions we consider both angular error and quaternion entropy for evaluating the best performing model. Figure 6.23 depicts the distribution of the angular error values for normal and deformation constraint. Here we can observe that all the four models have high angular error in deformation constraint. In numerical terms all the models have angular error more than 5 deg and their

performance in deformation constraint is bad. All the performance of all the four deep learning models is affected by the object's deformation. But comparatively Bingham model has little less angular error when compared to other three models. Let's investigate the behavior of uncertainty to get more insights.

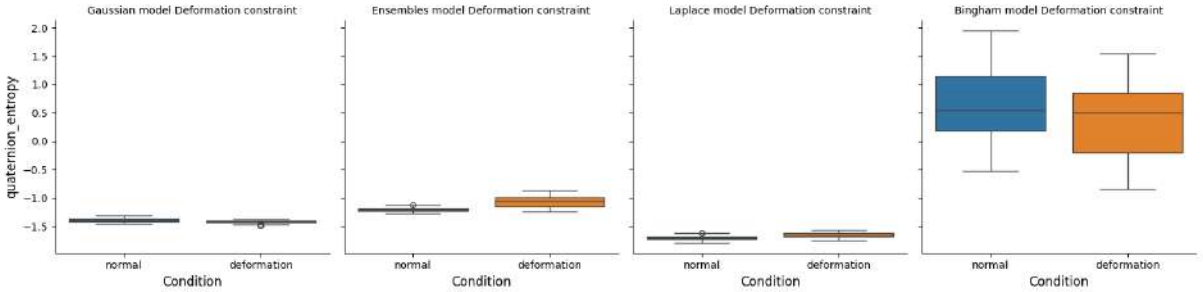


Figure 6.24: Box plot for quatenion entropy results from normal and deformation constraints

From the figure 6.24, we can observe that, in Gaussian, Laplace and Bingham models, the uncertainty in deformation constraint is less than the uncertainty in normal constraint which is not expected. Hence Gaussian, Laplace and Bingham models fail to satisfy the desired hypothesis. In Ensembles model, we see a separation in the distributions and also the uncertainty in deformation constraint is more than the uncertainty in normal constraint. So Ensembles model satisfies the desired hypothesis. But due to high angular error values in the deformation constraint, we do not consider ensembles model as the best performing model.

Further research is need and models capable of learning object's deformation is required for achieving good accuracy in estimating 6D object pose and also to satisfy the desired hypothesis.Overall, there is more impact of object's deformation on the predictions of rotation parameters compared to the predicted location parameters of the deep learning models.

In the second experiment, we have trained the DNN models, using YCB_Tomato_Soup_Can object under normal constraint settings and tested the performance on YCB_Tomato_Soup_Can object under deformation constraint settings. The datasets used and the results for the metrics in lighting constraint is provided in the Appendix-B A

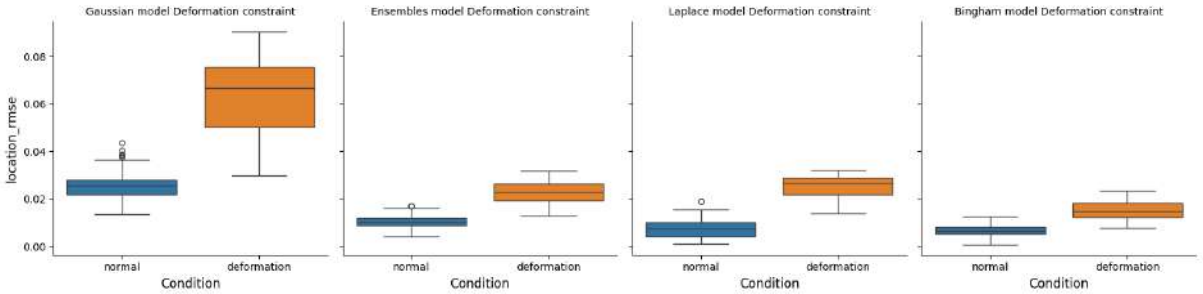


Figure 6.25: Box plot for location RMSE results from normal and deformation constraints for tomato soup can

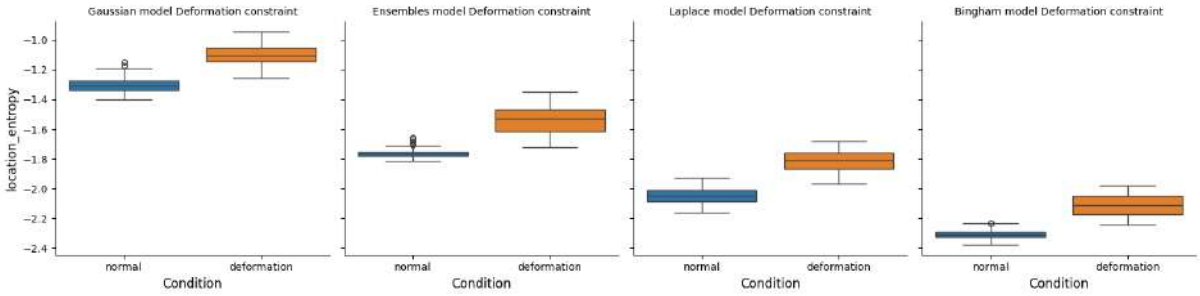


Figure 6.26: Box plot for location entropy results from normal and deformation constraints for tomato soup can

From the figure 6.25, we can see that Bingham model has low RMSE values compared to Gaussian, Laplace and Ensemble models. From figure 6.26, we can see the separation in the distributions of normal and deformation constraints in all four models. Since Bingham model has more test samples with low RMSE values, we consider Bingham model as the best performing model for predicting location parameters in deformation constraint.

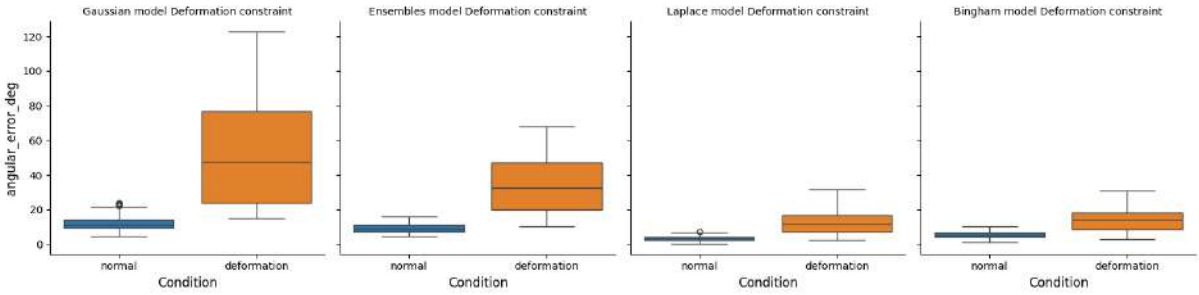


Figure 6.27: Box plot for angular error results from normal and deformation constraints for tomato soup can

From the figure 6.27, we can see that all four models have high angular error values in the deformation constraint. This shows that the performance of models is not good with symmetric objects. From figure 6.28, we can see the separation for the entropy distribution in Gaussian, Laplace and Ensemble models. But due to high angular errors we do not consider them as the best performing models. Deep learning models and loss functions which can account for object symmetries are required.

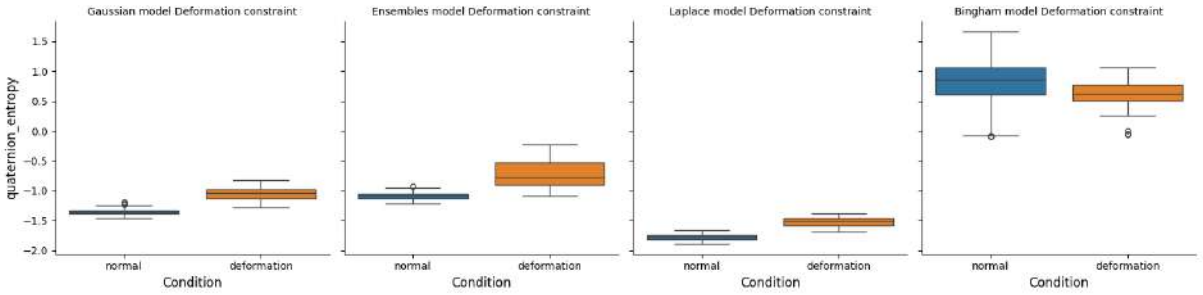


Figure 6.28: Box plot for quatenion entropy results from normal and deformation constraints for tomato soup can

7

Conclusions

7.1 Contributions

- Literature survey on state of the art "Synthetic Dataset Generation Tools", exploring the capabilities and limitations.
- In this thesis work, a blender based synthetic dataset generation tool was developed which streamlines the process of generating synthetic datasets ensuring the flexibility and ease of use. The developed tool is capable of generating synthetic datasets for diverse range of computer vision applications like classification, object detection, semantic segmentation,surface normal estimation and depth estimation.
- The developed tool is capable of generating datasets for various environmental conditions like, bright lighting, dark lighting, far distance objects, near distance of objects, blurry images, various background textures, deformation of objects and normal conditions.
- **Custom Lab Environment**: A custom lab environment for the C_025 laboratory of Hoshschule Bonn-Rhein-Sieg university of applied sciences is designed in Blender3D software from scratch.
- The developed tool is capable of generating datasets with different annotation formats like, BOP, YOLO and COCO.
- The developed tool allows the users to control lighting , camera paths, background textures, camera intrinsic parameters and objects deformation.
- Comparative evaluation of four Uncertainty estimation methods is performed for lighting, distance and deformation constraints.

7.2 Lessons learned

- In the overall experiments, it is observed that the the Bingham model is capable of learning rotational symmetries and it provides accurate rotation estimates.
- Since the Bingham distribution is considered as "the maximum entropy distribution on the hyper-sphere", it is hard to compute the cumulative distribution function (CDF).

- Since CDF is not available for Bingham distribution, computing metrics like, Interval Score, Uncertainty Calibration Score is not possible.
- In the overall experiments, Laplace model performed better in estimating the location parameters.
- In Bingham Loss, the models performance depends on the normalization constant which is generated using the pre computed look up tables and these look up tables needs to be generated based on the dataset.

7.3 Future work

- For Bingham model, during training instead of Gaussin NLL loss, Laplace NLL loss can be used for getting more robust predictions for the complete 6D pose of the object.
- Experiments on Symmetric objects can be conducted for evaluating the uncertainties in the predictions. A research question can be formed to check the impact of object symmetries on uncertainty estimation methods.
- For the dataset generation tool, a transformer based object/scene creation can be integrated for generating datasets with more environmental features.
- More constraints like, Noise (Gaussian noise) , Occlusions can be integrated into the dataset generation tool.
- An interactive tool can be developed for visualizing the parameters of blender while creating the configuration file.

A

Additional Metrics Results

Lighting Constraint					
Metric	Constraint	Gaussian Model	Laplace Model	Ensembles Model	Bingham Model
Location RMSE	Normal	0.025	0.008	0.01	0.007
	Bright	0.028	0.015	1.034	0.009
	Dark	0.079	0.024	0.026	0.019
Angular Error degrees	Normal	12.283	3.448	9.194	5.553
	Bright	18.367	6.716	24.459	4.445
	Dark	15.719	3.232	19.607	8.432
Location Interval Score	Normal	1.006	0.142	0.798	-
	Bright	1.062	0.152	3.16	-
	Dark	1.249	0.189	0.991	-
Quaternion Interval Score	Normal	0.982	1.061	1.122	-
	Bright	1.044	1.091	1.607	-
	Dark	1.24	1.215	1.226	-
ADDS	Normal	0.022	0.006	0.009	0.005
	Bright	0.025	0.011	0.02	0.007
	Dark	0.086	0.017	1.768	0.013

Table A.1: Metric results for tomato soup can under lighting constraints

Distance Constraint					
Metric	Constraint	Gaussian Model	Laplace Model	Ensembles Model	Bingham Model
Location RMSE	Normal	0.025	0.008	0.01	0.007
	Far	0.05	0.014	0.016	0.009
	Near	0.033	0.051	0.014	0.068
Angular Error degrees	Normal	12.283	3.448	9.194	5.553
	Far	17.943	11.372	10.892	12.279
	Near	25.853	16.885	14.994	32.002
Location Interval Score	Normal	1.006	0.142	0.798	-
	Far	1.096	0.171	0.873	-
	Near	1.02	0.177	0.828	-
Quaternion Interval Score	Normal	0.982	1.061	1.122	-
	Far	1.106	1.169	1.2	-
	Near	0.981	1.176	1.289	-
ADDS	Normal	0.022	0.006	0.009	0.005
	Far	0.041	0.009	0.012	0.007
	Near	0.032	0.046	0.013	0.068

Table A.2: Metric results for tomato soup can under distance constraints

Deformation Constraint					
Metric	Constraint	Gaussian Model	Laplace Model	Ensembles Model	Bingham Model
Location RMSE	Normal	0.025	0.008	0.01	0.007
	Deformation	0.025	0.062	0.022	0.015
Angular Error degrees	Normal	12.283	3.448	9.194	5.553
	Deformation	51.787	12.661	34.315	13.729
Location Interval Score	Normal	1.006	0.142	0.798	-
	Deformation	1.116	0.18	0.896	-
Quaternion Interval Score	Normal	0.982	1.061	1.122	-
	Deformation	1.16	1.202	1.369	-
ADDS	Normal	0.022	0.006	0.009	0.005
	Deformation	0.018	0.071	0.019	0.011

Table A.3: Metric results for tomato soup can under deformation constraint

B

Description of AI-Generated Content

In this work and AI application called Grammarly, was used to check for spelling mistakes and grammatical errors. It enhanced the transparency of the text and organized the sentences better. Grammarly itself was not used to generate any new content. The tool was used to check and improve the writing and accuracy. The main ideas, writing and analysis were done by hand.

References

[1] S. Panchangam, D. Nair, and N. Hochgeschwender, "Benchmarking uncertainty estimation of deep learning models using synthetic dataset," Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany, Tech. Rep., Jan. 2023, . Research and Development Project Report.

[2] D. Nair, S. Panchangam, M. A. Olivares-Mendez, and N. Hochgeschwender, "Embodied runtime monitoring of learning-enabled robot perception components," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2024, pp. 2983–2989.

[3] C. Sahin and T.-K. Kim, "Recovering 6D object pose: A review and multi-modal analysis," in *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018, pp. 0–0.

[4] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1513–1518.

[5] C. Sahin, G. Garcia-Hernando, J. Sock, and T.-K. Kim, "A review on object pose recovery: From 3D bounding box detectors to full 6D pose estimators," *Image and Vision Computing*, vol. 96, p. 103898, 2020.

[6] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis *et al.*, "Bop: Benchmark for 6D object pose estimation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 19–34.

[7] G. Marullo, L. Tanzi, P. Piazzolla, and E. Vezzetti, "6D object position estimation from 2D images: a literature review," *Multimedia Tools and Applications*, vol. 82, no. 16, pp. 24 605–24 643, 2023.

[8] Y. Su, J. Rambach, A. Pagani, and D. Stricker, "Synpo-net—accurate and fast cnn-based 6dof object pose estimation using synthetic training," *Sensors*, vol. 21, no. 1, p. 300, 2021.

[9] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann *et al.*, "Kubric: A scalable dataset generator," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 3749–3761.

[10] W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 909–916.

[11] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi, "Blenderproc: Reducing the reality gap with photorealistic rendering," in *16th Robotics: Science and Systems, RSS 2020, Workshops*, 2020.

[12] T. Löfgren and D. Jonsson, "Generating synthetic data for evaluation and improvement of deep 6d pose estimation," 2020.

[13] P. Sharma, "Benchmark for few-shot 6d pose estimation methods," Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany, Tech. Rep., Jan. 2023, . Master's thesis.

[14] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[15] K. Wursthorn, M. Hillemann, and M. Ulrich, "Uncertainty quantification with deep ensembles for 6d object pose estimation," *arXiv preprint arXiv:2403.07741*, 2024.

[16] I. Manivannan, "A comparative study of uncertainty estimation methods in deep learning based classification models," 2020.

[17] J. Glover and L. P. Kaelbling, "Tracking 3-D rotations with the quaternion bingham filter," 2013.

[18] "Free PBR Textures - Poliigon — poliigon.com," https://www.poliigon.com/textures/free, [Accessed 12-04-2025].

[19] D. Nair, "uncertainty-regression-robustness. devblog—github.com," https://github.com/deebuls/devblog/blob/main/posts/notebooks/2020-05-05-uncertainty-regression-robustness.ipynb, 2020, [Accessed 10-04-2025].

[20] W.-L. Huang, C.-Y. Hung, and I.-C. Lin, "Confidence-based 6D object pose estimation," *IEEE Transactions on Multimedia*, vol. 24, pp. 3025–3035, 2022.

[21] D. S. Nair, N. Hochgeschwender, and M. A. Olivares-Mendez, "Maximum likelihood uncertainty estimation: Robustness to outliers," p. 8, 2022.

[22] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[23] C.-I. Yang and Y.-P. Li, "Explainable uncertainty quantifications for deep learning-based molecular property prediction," *Journal of Cheminformatics*, vol. 15, no. 1, p. 13, 2023.

[24] R. Arun Srivatsan, M. Xu, N. Zevallos, and H. Choset, "Probabilistic pose estimation using a bingham distribution-based linear filter," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1610–1631, 2018.

[25] I. Gilitschenski, G. Kurz, S. J. Julier, and U. D. Hanebeck, "Unscented orientation estimation based on the bingham distribution," *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 172–177, 2015.

[26] I. Gilitschenski, R. Sahoo, W. Schwarting, A. Amini, S. Karaman, and D. Rus, "Deep orientation uncertainty learning based on a bingham loss," in *International Conference on Learning Representations*, 2020.

[27] Blender Online Community, "Blender - a 3d modelling and rendering package," https://download.blender.org/release/Blender4.0/, Blender Foundation, November 2023, accessed: 2025-04-07.

[28] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[29] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[30] S. Tyree, J. Tremblay, T. To, J. Cheng, T. Mosier, J. Smith, and S. Birchfield, "6-DoF pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark," in *International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[31] C. Papaioannidis and I. Pitas, "3D object pose estimation using multi-objective quaternion learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 8, pp. 2683–2693, 2020.

[32] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6D object pose estimation," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 606–619.

[33] D. Nair, "Comparing distributions: Normal, laplace and cauchy–devblog," https://deebuls.github.io/devblog/posts/notebooks/2022-12-15-comparing-normal-laplace-cacuhy-distirbutions-interval-score.html, Dec. 2022, accessed: 2025-4-12.