Technische Hochschule Ingolstadt

# Bachelor Thesis

in the study program Autonomous Vehicle Engineering (B.Eng)
Faculty of Electrical Engineering and Information Technology

## Digital Twin Dataset vs Domain-specific randomized synthetic dataset to train 6d pose estimator of Industrial parts


First- und Last name : **Vishal Balaji**



Issued on : 07.12.2023

Submitted on : 21.01.2024



First Examiner : Prof. Dr. rer. nat. Lenz Belzner

Second Examiner : Prof. Alexander Schiendorfer

Company advisors : Mr. Daniel Jess & Mr. Philipp Quentin

# Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, that I have not presented it elsewhere for examination purposes, and that I have explicitly indicated all material which has been quoted either literally or by consent from the sources used. I have marked verbatim and indirect quotations as such.

Munich, 21. January 2024

Vishal Balaji

# Acknowledgments

This thesis is a milestone in my academic journey, and I owe a debt of gratitude to many who have supported me along the way.

Firstly, I extend my sincere thanks to my supervisors in BMW, Mr. Daniel Jess and Mr. Philipp Quentin, for the invaluable guidance, patience, and expertise. Your mentorship has been instrumental in shaping both this research.

I am also deeply grateful to Prof. Dr. rer. nat. Lenz Belzner, whose insights and assistance have been indispensable. Your constructive critiques and continuous engagement with my work have significantly contributed to its depth and rigor. I am profoundly grateful for your invaluable contributions and steadfast support.

My family, my friends and myself - Thanks for not giving up and believing in me. We got this !!

Vishal Balaji
January 2024

# Abstract

*This thesis explores the effectiveness of training the deep-learning-based 6D pose estimation algorithm, GDR-NPP, for the detection of automotive industrial parts using a photorealistic synthetic dataset generated from digital simulation, in comparison to a structured domain-randomized dataset in an industrial context.*

In the realm of industrial robotics, accurately detecting 6D poses of objects, essential for tasks like pick-and-place operations, presents significant challenges. Notably, GDR-NPP, an open-source deep-learning-based 6D pose algorithm highlighted in the BOP benchmark [40], has shown fast and robust results using single RGB camera inputs. However, collecting and annotating precise 6D pose data from RGB images is a daunting, time-consuming, and costly process, particularly for planned use cases that do not yet physically exist. Leveraging simulations from photorealistic, physically accurate engines like Nvidia IsaacSim, it is possible to realistically simulate the specific objects of interest and their environments.

This research establishes a pipeline for generating synthetic 6D pose data from such simulations, aiming to make data collection more cost-effective and efficient. The study involves creating and evaluating models trained on both digital twin datasets and structured domain-randomized datasets [37] within an industrial context. The findings reveal that while digital twin datasets offer accuracy benefits over industrial randomized datasets in the 6D pose estimation pipeline (Mask R-CNN + GDR-NPP), the integration of real images with synthetic data, as suggested in [45] and [38], enhances accuracy. Within the scope of this study, it was observed that models trained on industry domain-randomized datasets, when augmented with real images, exhibited the highest level of accuracy. Despite these advancements, the current state of the trained models necessitates further research to reach the reliability required for consistent industrial robotic applications.

# Glossary

**abs** Absolute operation.

# Acronyms

**6D** Six Degree-of-Freedom.

**ADD** Average Distance Difference.

**AI** Artificial Intelligence.

**AP** Average Precision.

**AR** Average Recall.

**BOP** Benchmark for 6D Object Pose Estimation.

**CAD** Computer-aided design.

**COCO** Common Objects in Context.

**D** Dimensions.

**FPS** Frames Per Second.

**GDR-NPP** Geometric Deep Learning Network for 6D Pose Prediction.

**ICP** Iterative Closest Point.

**IoU** Intersection over Union.

**MDD** Mean Distance Difference.

**RGB** Red Green Blue.

**RGB-D** Red Green Blue - Depth.

**ROS** Robot Operating System.

**SLAM** Simultaneous Localization and Mapping.

**STL** Stereolithography.

**ToF** Time of Flight.

# List of figures

# List of tables

# Table of contents

# 1 Introduction

6D pose estimation refers to determining the six degree-of-freedom (Position - x, y, z and Orientation - rotation along each of x, y, z axis) to accurately locate an object in world space. This is a relatively simpler task than 3D object detection, as the latter not only requires detecting the position and orientation of objects but also their size. 6D pose estimation is widely employed in the field of robotics, particularly for precise tasks such as object picking and placement, and also plays a crucial role in augmented and virtual reality applications. This section outlines the motivation behind this research, the specific research questions it addresses, and provides a structured overview of the thesis.

## 1.1 Motivation

In BMW factories, the use of robots is becoming more prevalent, particularly in internal logistics and on assembly lines. Many challenges arises in such situations - ranging from a large number of different parts present in a cluttered fashion within - in robotics sense - not very well defined environments. Due to highly optimized manufacturing processes used in the factories, there is no room for error and any small downtime for the robots can result in enormous financial losses. This particularly applies to the perception system as well, where the system is not only meant to be fast, scalable and maintainable, but also be robust against occlusions and errorneous sensor data [38].

Current state-of-the-art computer vision algorithms are based on deep-neural networks (especially open-source models) and significantly outperform traditional algorithms based only on hand-crafted features [40]. Training deep neural networks requires diverse and large quantities of labelled data. Collecting data inside a running factory and labelling them is a time, money and human -intensive task, especially for 6D pose estimation as the labels are in 3D world space and not in 2D image space like bounding box or semantic mask annotation. To tackle this problem, synthetic data generation provides the capacity to generate unlimited amount of data but with limitations of initial setup cost, availability of computational resources and a gap between rendered images with respect to real world (commonly referred to as Sim2Real gap). To this extent, many common synthetic image generators are available: 3D modelling software Blender based BlenderProc [13], realistic game engine based generation tools like Unity Perception package [43] or physically accurate simulation engines like Nvidia Omniverse based IsaacSim *(Replicator plugin)*, among others.

Creating a digital twin of factories is becoming a common practice to simulate a planned logistic process, trying out multiple configurations for optimizing the efficiency and reduce costs while ensuring high-quality output [14]. Given a precisely modelled scene of a planned usecase in simulation (essentially a digital twin), a very realistic synthetic data - exactly replicating the poses of the industrial parts could be generated. Common approach with synthetic data generation [42], [9], [44] involves generating domain-randomized dataset to provide significant variability in the data that the model learns to generalize effectively across a range of scenarios, even to those it has not encountered before.

## 1.2 Research objectives

The core research objectives of this thesis are listed below:

- The primary objective of this thesis is to evaluate the necessity of using physically accurate digital twin based synthetic dataset in comparison to industrial domain randomized synthetic dataset for training GDR-NPP 6D pose estimator. Digital Twin dataset here refers to the actual replica of the factory scene along with realistic physics and similar object poses as observed in the real world. Industrial domain randomized dataset refers to data generated in a industrial warehouse with standard industrial assets like barrels, forklifts, cones, boxes as distractors with completely randomized lighting and textures on the background wall and floor. The evaluation of the accuracy of the trained models are done on a subset of real labelled images.
- The secondary objective is to evaluate the accuracy gain by mixing real annotated images with each synthetic dataset. It is a proven theory that mixing real images with synthetic images helps in increasing accuracy of computer vision models [45] and [38] - This theory is specifically evaluated for our 6D pose estimation pipeline.

## 1.3 Outline

The present thesis is structured as follows:

- **Chapter 2: Background**
  This chapter provides an overview of the theoretical background relevant to 6D pose estimation and synthetic data generation. It also reviews previous work in the field, focusing on methods and technologies that have informed the current research.
- **Chapter 3: Methodology**
  This chapter details the methodology adopted in this research. It includes the process of generating synthetic data, real data collection and annotation, and the description of the training tools used for the 6D pose estimation models.
- **Chapter 4: Experiments**
  This chapter describes the experimental setup, including data processing, training details, and the specific experiments conducted to evaluate the effectiveness of the synthetic and real data integration.
- **Chapter 5: Results**
  This chapter presents the results of the experiments. It discusses the findings in detail, comparing the performance of models trained with different datasets and analyzing the effectiveness of mixing synthetic with real-world data, and highlights the limitations encountered.
- **Chapter 6: Conclusion**
  The final chapter summarizes the key findings of the research and suggests potential areas for future work.

# 2 Background

This chapter explains the theoretical background required to understand this thesis.

## 2.1 6D Pose estimation

6D pose estimation refers to the task of detecting the six-degrees of freedom of the object in 3D world space. This entails estimating the accurate translation vector (x, y, z) and rotation vector (indicates the orientation around three axes - x, y, z) of the object (See figure 1). This is a fundamental task in computer vision and robotics. In robotics, this is done using RGB or RGB-D (with depth) cameras to precisely identify the object of interest and to interact with them - in tasks like picking/placing, grasping and accurate semantic scene understanding.

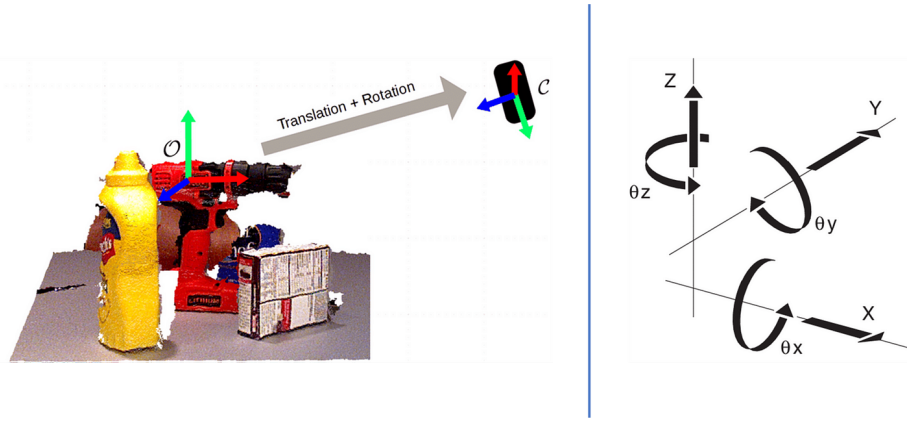



**Figure 1:** 6D pose estimation - Translation and rotation vector [32]

Depth information of the scene has proven to massively improve the accuracy of 6D pose estimations, it can be captured by specialized hardware like LiDAR, ToF sensors, structured light- or stereo- cameras [2]. Since these hardware are expensive, have highly specific depth operating window and a lower resolution than RGB cameras, the latter is preferred for common computer vision tasks like classification, object detection and segmentation. RGB cameras are affordable, widely available and capture data at high detail. Inspired by human binocular vision, where two eyes provide slightly different perspectives of the same scene to perceive depth, stereo cameras extend normal RGB cameras by adding one more camera to allow for depth perception. By comparing the images from the two cameras, the system can estimate the depth of various points in the scene. This process is known as stereo matching and is a computationally expensive process (see figure 2). Since stereo cameras are based out of normal camera sensors, they are relatively cost-effective and provide depth accuracy comparable to alternative depth sensors. The main weaknesses of stereo cameras arises due to the challenging task of stereo matching, especially in scenes with low light, repetitive patterns, lack of texture, or occlusions.

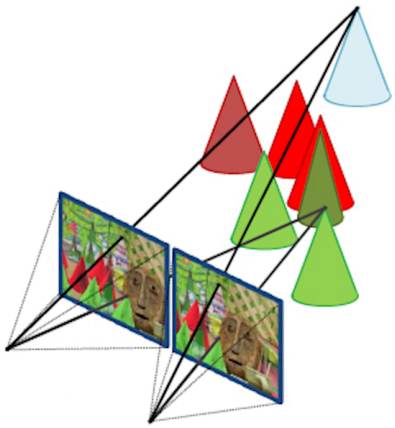


**Figure 2:** Stereo matching [1]

### 2.1.1 BOP Challenge

Benchmark for 6D Object Pose Estimation (BOP) challenge is continuously conducted every year to measure the progress in the field of pose estimation, with an implicit goal of finding the best synthetic-to-real domain transfer techniques [40]. In 2023, BOP challenge had 6 tasks: Model-based 6D localization, 2D segmentation and 2D detection of each seen and unseen objects. In this context, "seen" objects refers to class categories whose instance have the identical shape and form - like a scissors whose color and form always remain the same, while "unseen" objects refers to class categories whose instance have the different fixed shape and form - like cars. Each car might have different color and form, but they are still "car". The difference in technicality of them is similar to that of 6D pose estimation and 3D object detection, "unseen" object detection may usually need to also infer the size of the instances. Since "unseen" 6D pose localization is a more complex task, accuracy of models in this category are consistently less than those of "seen" categories [40]. The data used for training and evaluating the pose estimators

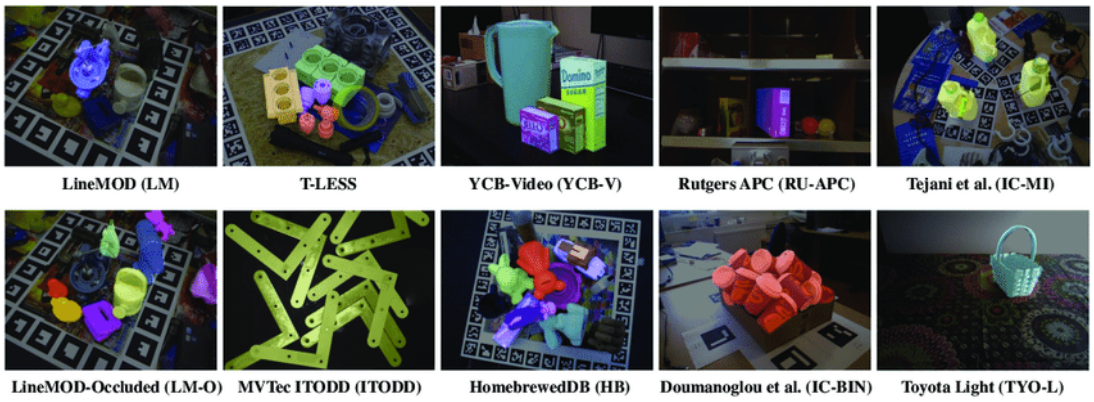



**Figure 3:** BOP datasets [25]

in the BOP challenge consists of seven core datasets: LM-O, T-Less, TUD-L, IC-Bin, ITODD, HB, YCB-V [40] (See figure 3). These datasets are highly varied in content and include 3D object models and annotated training/test RGB-D images with ground-truth

6D poses, 2D segmentation and detection data. The results from the BOP Challenge 2023 were instrumental in selecting a high-performing pose estimator for this bachelor thesis's specific use case. The object that the robots are designed to grasp is known and a 3D CAD model is available, categorizing it as a "seen" object. Additionally, given that the application is intended for a fast-paced automotive industrial factory, rapid inference times are crucial, ideally less than 1 second per detection of a single part. The 6D pose estimation model that best met these criterias, balancing high accuracy with the required swift inference time, is **GDRNPP-PBRReal-RGBD-MModel-Fast** - a derivative of GDR-Net pose estimation model.

### 2.1.2 GDR-Net

GDR-Net [46] is a novel deep learning framework for 6D pose estimation from single RGB images. GDR-Net stands for Geometric Distance Regression Network, and it consists of three main components (See figure 4): a feature extraction module, a geometric distance regression module, and a 6D pose regression module. The feature extraction module is

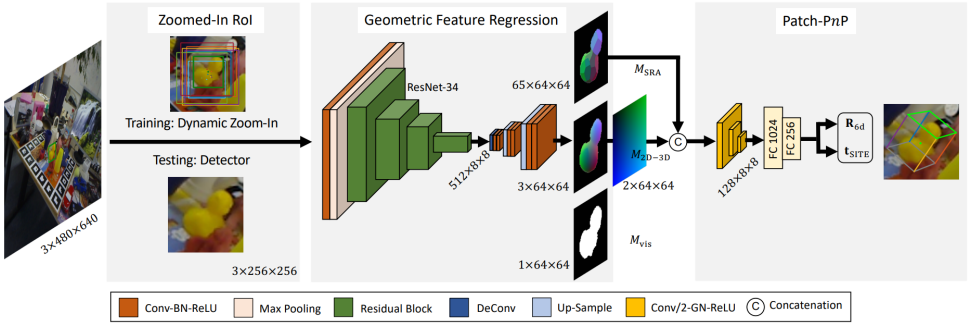



**Figure 4:** GDR-Net network architecture [46]

based on a convolutional neural network (CNN) that extracts high-level features from the input image. The CNN is pre-trained on a large-scale image classification dataset, such as ImageNet, and then fine-tuned on a specific object category for 6D pose estimation. The feature extraction module is responsible for capturing the appearance and texture information of the object, as well as its context and background.

The main idea of the geometric distance regression module is to use dense correspondence-based geometric features, such as the Dense Correspondences Map (M2D-3D), the Surface Region Attention Map (MSRA) and the Visible Object Mask (Mvis), to guide the direct regression of 6D object pose from a zoomed-in region of interest (RoI) of the input image. The final pose estimation module is based on Patch-PnP, which directly regresses the 6D object pose from the feature maps. The GDR-Net is inspired by CDPN [28], a state-of-the-art dense correspondence-based method for indirect pose estimation, but differs from it in several aspects. First, the GDR-Net does not require a disentangled translation head, which reduces the complexity and improves the efficiency of the network. Second, the GDR-Net introduces a new Surface Region Attention Map (MSRA) that encodes the surface region information of the object, which helps to filter out outliers and improve the accuracy of the pose estimation. Third, the GDR-Net employs a simple yet effective 2D

convolutional Patch-PnP module that directly regresses the 6D object pose from M2D-3D and MSRA, without relying on RANSAC or other post-processing steps.

One of the main advantages of GDR-Net is that it does not require any depth information, unlike many existing methods for 6D pose estimation. This makes it more suitable for scenarios where depth sensors are not available or reliable, such as outdoor environments or low-light conditions. Moreover, GDR-Net can handle occlusions, clutter, and varying lighting conditions, thanks to its robust feature extraction and pose refinement modules. GDR-Net can also handle multiple objects in a scene by applying a region proposal network (RPN) or any multi-object detector to generate candidate regions for each object category, and then applying GDR-Net to each region independently.

**Metrics** To evaluate the performance of GDR-Net, the common metric for 6D pose estimation: the average distance error (ADD) [18] is used. The ADD measures the average euclidean distance between the ground truth and the estimated object model points projected in 3D space. Lower values of ADD indicate higher accuracy. A more intuitive to measure this accuracy is by setting distance thresholds (usually in m) and verifying the percentage of all estimated poses that have an ADD value under this given threshold. In addition to ADD, this study incorporates the Maximum Distance Error (MDD) to ensure consistent detection and grasping accuracy. The thesis evaluates both ADD and MDD at thresholds of 0.7cm, 1cm, 2cm, 5cm, and 10cm. The thresholds of 0.7cm and 1cm are specifically selected for precise grasping tasks related to hinges, while the 2cm, 5cm, and 10cm thresholds align with the standard error benchmarks used in the Benchmark for 6D Object Pose Estimation (BOP) toolkits. Errors exceeding these thresholds are indicative of highly inaccurate pose estimations, rendering them unsuitable for industrial applications.

**GDRNPP-PBRReal-RGBD-MModel-Fast** [30] is a modified version of GDR-Net with the following changes:

- **Advanced Network Structure:** The usage of the more robust ConvNext as the backbone, replacing ResNet-34, coupled with the integration of dual mask heads for distinct predictions of amodal and visible masks.
- **Enhanced Data Augmentation:** Training incorporated more intensive domain randomization techniques compared to the initial configuration.
- **Diverse Refinements:** Precise adjustments in key hyperparameters like learning rate, weight decay, visible threshold, and the types of bounding boxes.

In the model name, "Fast" denotes a streamlined pose refinement stage that utilizes a depth map. Traditional pose refinement methods employ the Iterative Closest Point (ICP) algorithm [24] [41], which aligns three-dimensional ground truth and estimated poses. This process is computationally intensive and slow. However, in this model, refinement is limited to adjusting the translation vector, particularly in the z direction. This approach provides a quicker approximation compared to full ICP methods, trading some accuracy for speed. Consequently, it better meets real-time latency requirements, unlike comprehensive ICP-based refinement techniques. From this point onwards in this thesis, this modified version of GDR-Net would be abbreviated and addressed as GDRNPP.

### 2.1.3 Mask R-CNN

GDR-Net network inherently detects the poses of only one object at a time, but images usually contains multiple objects that needs to be detected. As a result, the GDR-Net pose estimation pipeline employs an object detector in the beginning, crop on to each detection's bounding box individually and then estimate the poses for them. For this specific research work, Mask R-CNN segmentation network is used - Bounding boxes are derived from min-max coordinates of the segmentation masks. This study forms a segment of a larger project, wherein the Mask R-CNN is employed for additional objectives pertinent to the perception system. These aspects, while not central to the scope of this thesis, necessitate the use of Mask R-CNN over the YOLO-X detection network used by GDR-NPP.

Mask R-CNN [16] is a robust network for object instance segmentation. It is an extension of Faster R-CNN, a popular object detection model, that adds a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

The basic idea of Mask R-CNN is to use a ResNet based *Region Proposal Network (RPN)* to generate regions of interest (RoIs) that may contain objects, and then apply a *RoIAlign* layer to extract features from each RoI using bilinear interpolation (See figure 5). The features are then fed into two separate branches: one for classifying the RoI and regressing the bounding box, and another for producing a binary mask for each RoI. The mask branch is a small fully convolutional network that outputs one mask for each class output. The final mask for each RoI is obtained by selecting the exact slice of the tensor corresponding to the predicted class. Mask R-CNN is suitable as an object detector

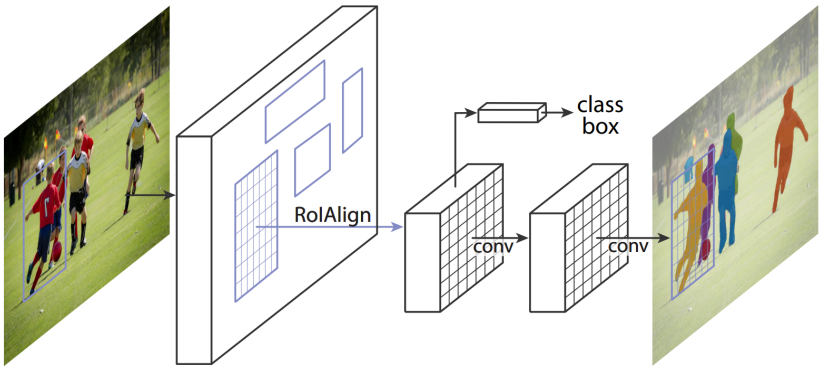



**Figure 5:** Mask R-CNN network architecture [16]

because it can efficiently detect objects in an image while simultaneously generating a high-quality segmentation mask for each instance and inference speed is fast enough for industrial applications. Since its debut in 2018, Mask R-CNN has been widely embraced, with an abundance of pretrained models on comprehensive datasets such as ImageNet readily available [49].

### 2.1.4 Data labelling

For training any DNN (deep neural network), large quantities of labelled data is required - GDR-NPP is no different. Human based labelling of 6D pose estimation data has following challenges:

- **High precision requirement:** 6D pose estimation demands highly accurate labeling of both orientation and position in 3D space. Even minor errors in labeling (See figure 6) can significantly affect the performance of the pose estimation model.
- **Complexity in annotation:** Unlike 2D labeling, which involves drawing boxes or segmentation masks on images, 6D pose annotation requires determining the object's exact world position and orientation relative to the camera - aligning in 3D space, which is inherently more complex.
- **Variability and occlusions:** In real-world scenarios, objects can appear in various orientations and can be partially occluded, complicating the annotation process.

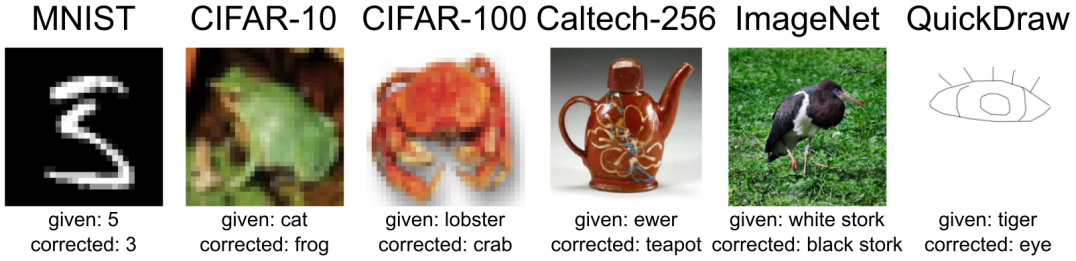



**Figure 6:** Average label error of 3.4% has been discovered across 10 most-cited datasets in a study led by computer scientists at MIT [34]

The complexities involved in 6D pose annotation make it a time-consuming, error-prone, and costly endeavor. This reality underscores the significance of employing synthetic data generation tools. Nonetheless, for the experiments as mentioned in this thesis, a modest amount of accurately labeled real data is essential. For this purpose, *Labelfusion* [31], a specialized tool for 6D pose annotation, has been utilized.

**Labelfusion** is an open-source research tool for generating high-quality ground truth labels for real RGB-D data of cluttered scenes, developed by the Robot Locomotion Group at MIT CSAIL. It can produce pixelwise labels and 6D object poses for multiple objects in scenes with occlusions and varying lighting conditions.

Labelfusion is specifically used for 6D pose estimation by leveraging dense RGB-D reconstruction to fuse together RGB-D images taken from different viewpoints, and labeling with ICP-assisted fitting of object meshes (see figure 7).

Labelfusion works as follows:

1. Raw data is initially collected using a RGB-D sensor. The raw data consists of RGB and depth images from an RGB-D sensor (such as Kinect or RealSense).

2. The raw data is processed by ElasticFusion [48], a dense SLAM method that reconstructs a 3D point cloud of the scene from the RGB-D data.
3. The reconstructed point cloud is annotated by a human using a graphical user interface (GUI) that allows the user to select and align object meshes to the point cloud. The GUI uses ICP algorithm to assist the user in fitting the meshes to the point cloud.
4. The annotated point cloud is then used to automatically generate per-pixel labels and object poses for each RGB-D image by projecting the object meshes back into the 2D images. The labels and poses are stored in appropriate format that can be used for training various computer vision algorithms.

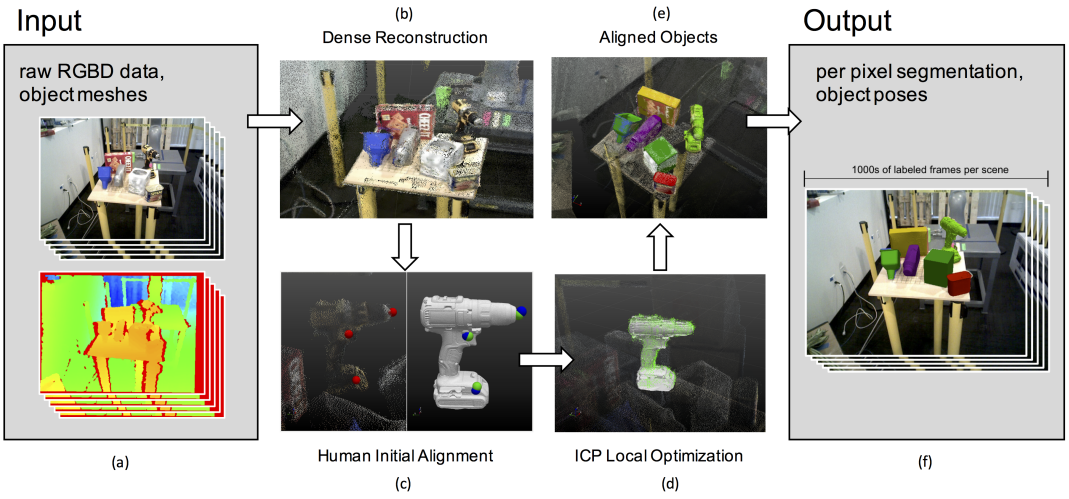



**Figure 7:** Labelfusion pipeline [31]

A notable advantage of the tool is that annotation with the object meshes has to be done only once per scene and then it is automatically reprojected back to every captured image - This approach effectively removes the need for the laborious and time-consuming process of manually annotating each individual image.
Some important observed drawbacks of the tool are:

- The efficacy of ElasticFusion's dense reconstruction, relying on simultaneous localization and mapping (SLAM), is closely tied to the quality of depth maps and the accuracy of visual feature matching. However, when recording RGB-D data manually, rapid movements and jerks often introduce blur, leading to inaccuracies in the SLAM algorithm. This can result in misaligned point clouds, causing improper ICP matching and, consequently, erroneous reprojection of poses. To mitigate these issues, employing smoother motion techniques, such as using a stabilizer or a robotic arm, is advised.
- Labelfusion does not effectively manage occlusions around the labelled objects. While it successfully reprojects poses and their associated mask labels onto the scenes, it fails to conceal parts of the mask that are not visible from the camera's perspective. It's important to note that while this limitation does not impact the

pose labels, it does affect the mask labels. Given that this research also incorporates the use of the mask-based object detector Mask R-CNN, training and evaluating it with such data could potentially compromise the accuracy of the results.

## 2.2 Synthetic Data Generation

This section describes the challenges faced with real world data collection and therefore, the necessity to generate synthetic data. It also explores the limitations of synthetically generated data and common tools used for this, specifically in computer vision.

### 2.2.1 Challenges with data collection

The importance of data in training machine learning (ML) or deep learning (DL) algorithms cannot be overstated. Data is fundamental to the functionality and effectiveness of these algorithms, as it directly influences their ability to learn, generalize, and perform tasks across various domains. Here's an overview of why data is crucial:

- **Learning from examples:** ML and DL algorithms learn by example. They require data to identify patterns, make decisions, and predict outcomes. The quality and quantity of this data significantly impact the learning process and the performance of the algorithm.
- **Generalization ability:** The primary goal of ML and DL models is to generalize well from the training data to unseen data. A diverse and representative dataset ensures that the model can handle various real-world scenarios and doesn't just memorize the training examples; essentially not overfitting.
- **Bias mitigation:** The data used in training can contain biases, which the model can inadvertently learn. Having a comprehensive dataset that accurately represents various facets of the problem domain is essential for mitigating biases in model predictions.

As such, meticulous attention to data collection, preparation, and augmentation is vital in the development of robust and reliable ML/DL models like the ones used in this thesis. In the previous section 2.1.4, it has been explained why human 6D pose estimation annotation is a very challenging and error-prone task. Collecting the real world data for this annotation also has many challenges:

- **Scalability:**
  Manual data collection processes are inherently slow and labor-intensive. Achieving the necessary scale for training 6D pose estimation models, which requires capturing a wide variety of objects in numerous poses and environments, is impractical and often impossible due to these limitations.

- **Diversity:**
  Manually collecting data that covers all possible variations in object poses, lighting conditions, backgrounds, and occlusions is challenging. This diversity is crucial for training robust pose estimation models, but predicting and replicating every possible variation manually, especially in planned or hypothetical scenarios, is exceedingly difficult.
- **High cost and resource intensive:**
  The process involves significant expenses, including equipment, manpower, and time. In industrial environments, the cost of manual data collection is not just financial but also includes the potential for expensive downtime. Every second of operational pause to collect data can lead to significant revenue loss. Moreover, these environments can be hazardous, posing safety risks to personnel involved in data collection. The need to ensure uninterrupted and safe operations makes manual data collection in such settings exceptionally challenging and often unfeasible.
- **Inaccessibility of future or hypothetical scenarios:**
  In scenarios where the use case is still in the planning phase and has not yet materialized, manually collecting relevant data is impractical, if not impossible. This research aligns with the development of such a project, focusing on the development of a new automated logistics process for a certain automotive component. The project entails designing a robotic system for this purpose, equipped with a 6D pose estimation-based perception system, relying entirely on detailed, predefined plans rather than existing, tangible environments.

Given these challenges, especially the impossibility of collecting manual data for planned or hypothetical scenarios, synthetic data generation emerges as a necessary and effective solution. Synthetic data can be generated to represent any number of scenarios, objects, and environments, irrespective of whether they currently exist or are in the planning stages. This approach offers scalability, diversity, and the ability to create data that can closely mimics real-world complexities with precise labels. Additional benefits encompass include reduced privacy concerns, ability to prototype swiftly and with meticulous designing, the possibility of eliminating any inherent biases in the generated data.

### 2.2.2 Limitations of synthetic data generation

This form of data generation also presents its unique challenges, including the gap to realism (commonly referred to as Sim2Real gap), constrained diversity, and significant computational demands. The realism gap arises from the inherent differences between synthetic and real-world data in aspects like textures, lighting, and sensor noise, leading to data that may poorly reflect actual environments (see figure 8). Moreover, while synthetic data allows for controlled scenario creation, it often lacks the unpredictability and variability found in real-world settings, potentially compromising the robustness of data. Furthermore, the creation of high-quality synthetic data, particularly for intricate environments, demands extensive resources. This involves not only significant computational power and time for precise rendering and simulation, but also substantial human labor is needed to lay the groundwork for the initial setup essential for the data generation.

**Figure 8:** Sim2Real gap - Simulated (left) vs real (right) image [20]

### 2.2.3 Common synthetic data generation tools

This research work focuses on using such photorealistic synthetic data generation tool for generating ground truth data for training the pose estimation model. Here is a brief overview of popular tools used for this purpose:

**BlenderProc2**:

BlenderProc2 [13] is a procedural pipeline designed by DLR-RM (Deutsches Zentrum für Luft- und Raumfahrt - Institute für Robotik und Mechatronik) for generating photorealistic annotated images. The entire code is released open-source [12] and uses the popular open-source 3D graphics software Blender as it's simulation engine. It supports various label formats like bounding box, segmentation masks, depth maps, normal, and pose estimation. A key feature is its easy-to-use Python API, which is extendable and compatible with public datasets like 3D FRONT, ShapeNet and BOP. Furthermore specifically for BOP benchmark, joint team between BlenderProc team and BOP organizers resulted in BlenderProc4BOP, an open-source, light-weight, procedural and photorealistic (PBR) renderer. The renderer was used to render 50K training images for each of the seven core datasets of the BOP Challenge 2020 [12].

**NViSII**:

NViSII [33] is a python scriptable photorealistic rendering engine built by Nvidia Research for research in computer vision. This utilizes Nvidia's OptiX hardware-accelerated path tracing and AI denoiser C++/CUDA backend to generate high-quality synthetic data to reduce the sim-to-real transfer in situations that are challenging for traditional renderers. This tool facilitates the creation and modification of intricate dynamic 3D environments, encompassing elements like object meshes, materials, textures, and lighting, as well as complex features such as volumetric data. Additionally, it can generate various annotation types, including 2D and 3D bounding boxes, segmentation masks, depth and normal maps, material characteristics, and optical flow vectors (see figure 9).

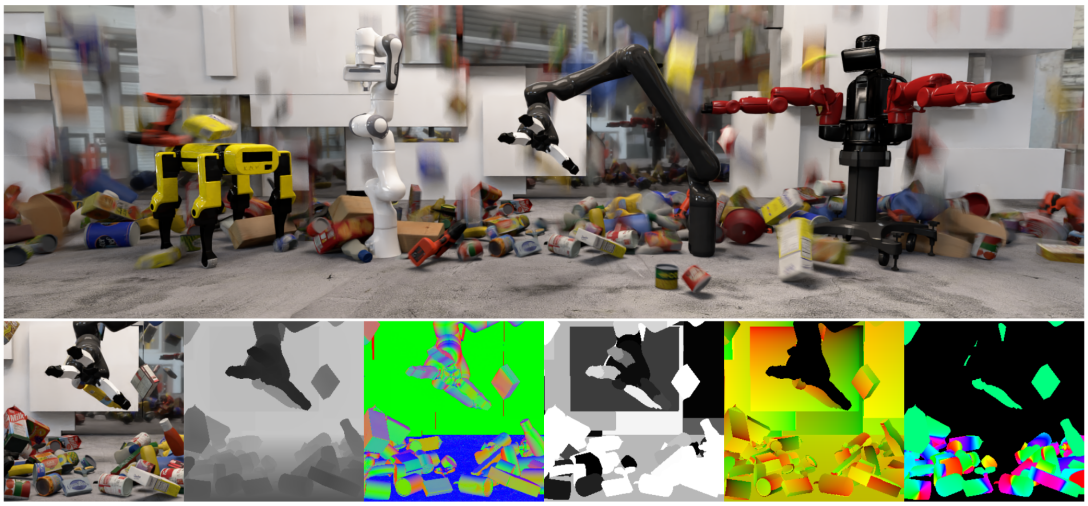


**Figure 9:** NViSII sample rendering [33]

Despite having very realistic rendering possibilities, to model different sensors and get annotated 6D label data with the above mentioned two tools, they are only renderers with very limited object manipulation capabilities. They cannot be used to build full-fledged simulations to create a digital twin, along with the necessary functions to interact with robots mimicking real world physics. This research works is based on concept of building the digital twin simulation and exploring if the data directly from the digital twin is enough for training pose estimation models. For this purpose, we are using the new and shiny simulation engine from Nvidia, the Isaac Sim.

**NVIDIA Isaac Sim**:

NVIDIA Omniverse Isaac Sim (hereby referred to as Isaac Sim) is an advanced robotics simulator that offers seamless integration with NVIDIA hardware and software, providing robust support for high-speed, GPU-accelerated simulation and AI training. Isaac Sim is based on NVIDIA Omniverse engine - A platform strongly connected with Pixar's open-source Universal Scene Description, or USD for short, to build virtual environments. This engine inherently provides realistic physics and photorealistic simulations that closely reflect the properties of the real world, and has software integration with popular robotic libraries like ROS1 and ROS2, making it an ideal simulator for the development of AI-enabled robotics applications.

There are three main ways of working with Isaac Sim: Standalone scripting using Python, Extensions or using the graphical interface. Graphical interface is the most convenient and intuitive way to interact with the simulation. One can create scenes, shapes, meshes, cameras and visually observe the changes that are being made. For more complex actions or behaviors, Isaac Sim provides a node based visual programming framework called OmniGraph. It provides a lot of pre-built nodes for many functions, like simulation timeline control, robotics controlers, sensor capture, among others. Custom nodes could be built as well, using the graphical interface or with Python. While this GUI and node based editing can tackle a lot of complex functions, not every possible function can be easily developed this way and there is no way to separate the code logic from the

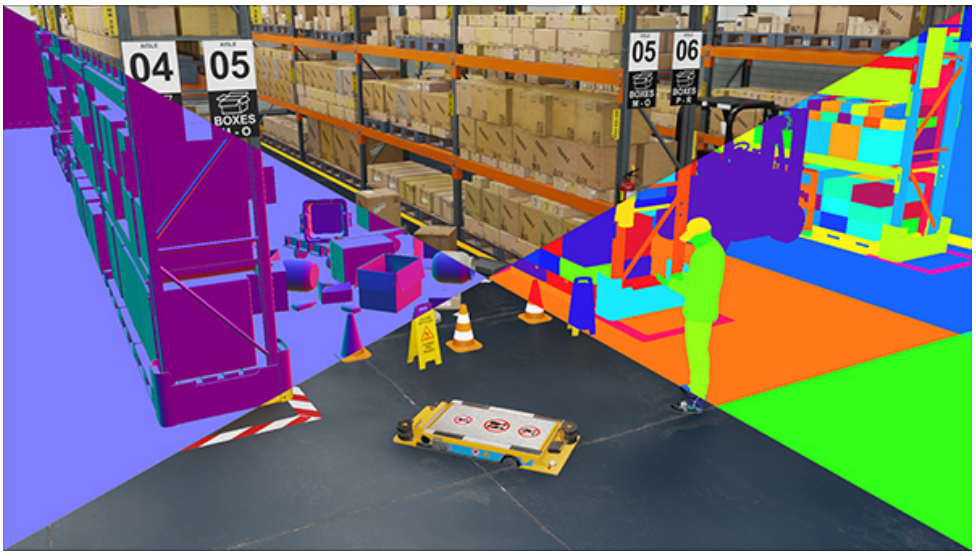



**Figure 10:** Nvidia Isaac Sim - Replicator visualization [35]

actual assets, since they are all combined in one, typically big, .usd file - This is not version control (like Git) friendly and hinders the collaborative working between different software developers working on the same scene.

As a result, Isaac Sim also provides an Python/C++ API (Application Programming Interface) called Omniverse Kit to script scenes and interact with objects, sensors as well extract properties from them during simulation. This is useful for moving object, cameras, changing lighting in the scene, change materials on objects and also log data from simulations robot's position or orientation. If required, every aspect of the simulation can be completely controlled via API alone.

The last but most essential workflow are extensions. All widgets and tools for Isaac Sim are built as extensions and they provide both scripting as well as GUI functionality to access Omniverse Kit API. They are inherently asynchronous and can run without blocking physics or rendering timelines. Every extension could designed with user-interfacebale widgets like checkboxes, buttons, dropdown menus or text boxes and are designed to one specific task only. NVIDIA provides enough sample examples of extensions to know about it's working and specifically for robotics and synthetic data generation as well. With custom extension creator, Isaac Sim automatically generates the basic template files for further development. Developing logic in extensions is more complex and involved process; also partly contributed to asynchronous program flow.

For the specialized task of generating synthetic data tailored to robotics, NVIDIA offers an array of extensions, Python APIs, workflows, and tools, collectively known as Replicator, as illustrated in figure 10. This tool allows to generate high-quality production ready synthetic data with structured domain randomization from multiple sensors and label formats. Replicator is the most vital tool from Isaac Sim used in this research work. This has been used extensively to load the object of interest, modify the position of cameras, lighting, adjusting the label format to our desired format, among others, completely through it's python API (see figure 11)

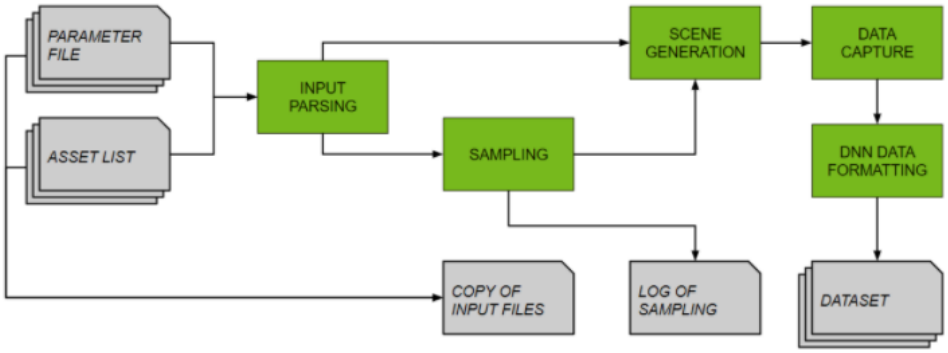



**Figure 11:** Sample workflow with Replicator [8]

## 2.3 Related works

Domain gap or realism gap, is a very well known problem with synthetic only simulations and computer vision models trained from purely synthetic data faces significant challenges adapting to realworld data. This section discusses the existing researches in these fields, with specific focus on industrial use cases.

### 2.3.1 Domain randomization

Domain randomization (DR) stems from the idea that training models on synthetic data with enough variability that the real world may appear to the model as just another variation. Extensive research has been conducted in this field, specifically for robotics. According to this paper from Tobin *et al.* [42], the important factors that help improve accuracy of object detection models trained purely from synthetic data are:

1. Using distractor objects
2. Significant number of different textures
3. Randomization in camera positions
4. Usage of pretrained backbone
5. Having more than 5000 images

Vanherle *et al.* [45] supplements the above findings and further establishes that having randomized lighting and poses of the objects help improve the object detection's accuracy, and including real labelled images along with synthetic data helps in improving the overall accuracy. Sundermeyer *et al.* [41] used a similar DR approach with Autoencoders for training object detection and 6D pose estimation models, achieving competitive results in T-LESS [21] and LineMOD [17] datasets. Quentin *et al.* [38] also utilized DR to train the GDR-Net 6D pose estimator for identifying automotive industrial parts, noting the approach's promise but also its current shortfall in meeting industry standards. One basic aspect of the above mentioned domain-randomized paradigms depends on using a lot of data to have sufficient variety for the model to learn from, and doesn't account for any

known aspect of the required scene. Shrinking the DR variety based on the structure and context of the scene, known as structured domain randomized (SDR) was introduced by [37] and performed better than techniques that only used DR synthetic data, and in some cases, techniques that utilized real data from a different domain. This research also established that amount of data to achieve good accuracy is significantly less than DR and most important factors that affect the accuracy are contrast, saturation and context.

Following these footsteps, the industrial domain-randomized dataset is based on the concept of SDR, whereby the distractors and the base environment resembles the expected industrial setting. The only deviation from [37] is the completely random poses used in the dataset, but all the other important observations from the above research findings are incorporated in generating this dataset.

### 2.3.2 Photorealistic rendering

In contrast to DR, photorealistic rendering strives to closely mimic real-world environments, with limited randomization in parameters such as lighting and the virtual camera's position. According to Hodaň et al. [22], using highly photorealistic Physically Based Rendering (PBR) technique and producing realistic object poses, has been validated as an effective method for training object detectors - Hodaň is also a leading contributor of BOP challenge and used similar methodology as mentioned in the above paper with BlenderProc for generating PBR images for the BOP dataset. This approach demonstrates a significant enhancement in performance compared to the baseline established by Hinterstoisser *et al.* [19], who utilized OpenGL for rendering objects onto cluttered real background images. Specifically, the use of realistic scenes resulted in an up to 24% improvement in mean average precision (mAP). Additionally, the role of contextual relevance was assessed: placing objects in a scene that accurately reflects the test data setup yielded up to a 16% improvement in mAP, compared to scenarios where the context was misaligned with the test setup.

Focusing on industrial object detection, Eversberg, L *et al.* [15] conducted a comparison between Domain Accurate (DA) PBR dataset and Domain Randomization (DR) dataset in the context of a Faster R-CNN based detector. Their research indicated that while DR's randomized backgrounds and distractors significantly outperformed DA, suggesting that realistic backgrounds or distractors might not be essential, the realistic object textures and lighting conditions in DA were impactful. However, they also found that random textures in DR yielded similar results, highlighting the significance of variability in datasets. Ultimately, they concluded that datasets rendered with photorealistic DA do not show marked advantages over DR.

Conversely, Weisenböhler *et.al* [47] systematically demonstrated the superiority of scene engineered (or digital twin) dataset, specifically in industrial use cases with well known and comparatively static properties, in comparison to completely DR datasets across multiple known and unknown scenes for object detection. Significant gain in accuracy was observed by using realistic textures, realistic camera poses, physically plausible item placements. Furthermore, they observed the importance of having strong randomizations in lighting conditions, which resulted in gain in accuracy across all their evaluated datasets.

Building upon the findings of [47], the digital twin dataset developed in this research incorporates critical elements such as realistic textures, object poses, placements, and notably, randomized lighting intensity.

## 2.4 Research gap

Drawing insights from leading research in synthetic data generation, as outlined in section 2.3, there appears to be a gap in studies specifically exploring the use of digital twin datasets with realistic textures and objects for 6D pose estimation tasks. While [15] suggests that such datasets offer no significant advantage for industrial object detection, both [47] and [22] demonstrate notable improvements from these datasets. Although these findings were not exclusively for 6D pose estimation, they share core similarities with this task, being grounded in CNN-based computer vision networks and standard training methodologies. Leveraging the learnings and optimized parameters from these studies, this research utilizes an existing digital twin to generate a 6D pose estimation dataset, featuring realistic camera poses and randomized lighting intensity. The impact of this dataset is evaluated with our 6D pose estimation pipeline (Mask R-CNN + GDR-NPP), following the approach of [38]. Additionally, this study compares a semi-structured domain randomized (SDR) dataset, proven by [37] to outperform DR datasets, against the digital twin dataset in an industrial context with relevant distractors. As [45] and [38] indicates the potential of mixing a small number of real images with synthetic data for enhanced performance, a similar approach is adopted in this research. However, there remains a research gap in comparing the impact of real images between structured domain randomized and digital twin datasets. This work aims to fill that gap, developing a setup to evaluate and answer the above questions.

# 3 Methodology

This chapter delves into the systematic approach employed in this research. It starts by providing an in-depth description of the specific industrial use case under consideration. The chapter comprehensively outlines the processes involved in synthetic data generation, including the creation of an industrial domain randomized dataset and a digital twin dataset. It also covers the aspects of real data generation, highlighting the methods of data collection and labeling. Additionally, the chapter elaborates on the training tools utilized, specifically Mask R-CNN and GDR-NPP, and concludes with a section on the experiment setups designed to investigate the research question.

## 3.1 Usecase

The 6D pose estimator investigated as part of this research work is intended to be used a specific logistics usecase inside BMW factory and this section gives brief overlook about it. The perception system is a part of internal robotics platform based on ROS2 for colloborative robots specifically focussing on picking-placing and palletizing application. There are multiple modules covering different aspects of robots: Motion planning, grasping, connectivity, decision making, perception, among others. All the modules should be generic - it should be able to work with wide variety of hardware and industrial parts. As a result, the 6D pose estimator should also be developed in a way that it can be trained on new objects with relative ease. This entire stack is run on custom built PC with industrial housing with a higher-end Intel CPU and a powerful RTX 40 series graphics card.

This research work involves itself with a specific usecase utilizing this robotics platform to achieve the following goal: Grasp an automotive part from a sliding flowrack and place it on a conveyor belt. This automotive part is a rear door hinge found in certain BMW models and it is asymmetrical for left-side and right-side of the door (see figure 12). The hinge weighs around 2.5kg measuring 36.7 cm end-to-end. In it's entirety has 11 unique parts but could be grouped into two distinct groups: Body and the head. Body is fully cast iron with slight shade of blue and the head is shiny metallic part that rotates along a pivot - Allowing the rear door to close and open (see figure 13). Apart from the structural difference between left and right hinges, all their material properties are completely identical to each other - and identical between the hinges themselves. These hinges are loaded into a custom designed flowrack by humans at one end and there are steel rods located in a inclined orientation facing downwards, so that the hinges naturally slide down one-after-another. At the end of the rods, there are brackets to hold the final hinge in position suitable for grasping by the robot. Each flowrack holds equal number of left and right hinges (see figure 14).

A 6-axis colloborative robot (abbreviated as "cobot") from Universal Robots URx series [3] is being used with a custom gripper to grasp on the holes present on the lower body of the hinge. A Framos D435e, a Intel Realsense D435 based camera with industrial housing (see figure 15), with a custom mounting bracket is attached to the last joint of the robot (end-effector). This camera consists of an IR (Infrared) active stereo imagers for depth sensing and a separate RGB camera for capturing color data. The robot is placed within

**Figure 12:** Left and right hinge

one meter distance to the flowrack.
The robot is required to operate in sync with the high-speed cycle typical of logistics and factory settings. Consequently, detection of the part must be completed within one second or less. This constraint emphasizes the critical need for a fast and precise 6D pose estimation algorithm.

## 3.2 Synthetic data generation

This section explains the generation of Industrial domain randomized dataset and Digital twin dataset using Nvidia Isaac Sim.

A basic setup of the above mentioned scene was already developed in Isaac Sim and made available prior to the beginning of this research work (see figure 16). This setup was intended to create a digital twin for testing various pipeline aspects. It featured a simulated warehouse environment, including a conveyor belt with industrial fencing and a UR10 robot model atop a base column. The scene additionally included basic, unshaded models of the flowrack and the hinges.

### 3.2.1 Base setup

A local workstation with i9 13900K CPU, 128GB RAM and state-of-the-art machine learning GPU, RTX 6000 Ada graphics with 48GB VRAM, running on Linux (Ubuntu 20.04) is primarily used for this thesis. A standard version of Nvidia Isaac Sim (Version: 2023.1) was installed [5] and used. Isaac Sim comes with it's own packaged version of

**Figure 13:** Open hinge

python and Visual Studio Code (VSCode) was chosen as the code editor to work with the python files.

**Updating hinge model**

3D CAD file for the hinge model (both left and right) was readily made available in STL file format, a common file format for saving 3D models [7]. Even though this file format could be directly loaded into Isaac Sim, there were number of challenges:

- **Incorrect origin:** The STL CAD file's initial origin point wasn't located at the geometric center, leading to an offset during hinge movement. Subsequently, when local rotations are applied, the hinge rotates about this off-center point, resulting in abnormal rotations. This situation is both undesirable and counterintuitive. *Solution:* Using the open-source 3D modelling software Blender [10], the entire mesh of the model was easily centered to it's geometry.
- **Incorrect rotation:** The STL CAD file for the right hinge had a default rotation of 90 degree along x-axis (essentially lying on the x-y plane surface) while left hinge didn't have this. This is also undesirable, as the pose writer (later explained) relies on the data from the CAD model, and applies an offset of 90 degree to right hinge while saving the data, even though it looks perfectly fine in the camera images. *Solution:* Again using Blender to apply the counter-rotation along x-axis to match the left-hinge.
- **Lack of materials:** STL format only stores the surface geometry of 3D dimensional objects - No color or texture information are stored as part of it. The exact materials for the hinge were roughly known but no information was available about the exact color of the parts. The color information was approximated with the help of image color picker from pictures of the hinge. The STL file contains a lot of submeshes - As mentioned previously, the hinge is made up of 11 unique parts - including many invisible screws, nuts, bolts and washers between them. When inspected from all the different angles, only 6 visually distinct parts are observed (see figure 17):

**Figure 14:** Partial flowrack with few hinges


**Figure 15:** Framos Realsense D435

- Blue iron-cast hinge body
- Chromic reflective bob on the hinge body
- Shiny hammered metallic hinge head
- Pivot bolt between the hinge head and hinge body
- Golden translucent plastic tape on the hinge's head
- Screws holding the plastic tape to the hinge's head

All the submeshes are grouped into these 6 parts alone and the remaining meshes are not visible and hence merged with next closest meshes. After this, realistic materials were added to the submeshes. In the world of 3D rendering, PBR materials are a complex combination of multiple properties like color, reflectivity, roughness, texture, displacement maps, among others. Different 3D softwares have their own

**Figure 16:** Initial usecase scene in Isaac Sim

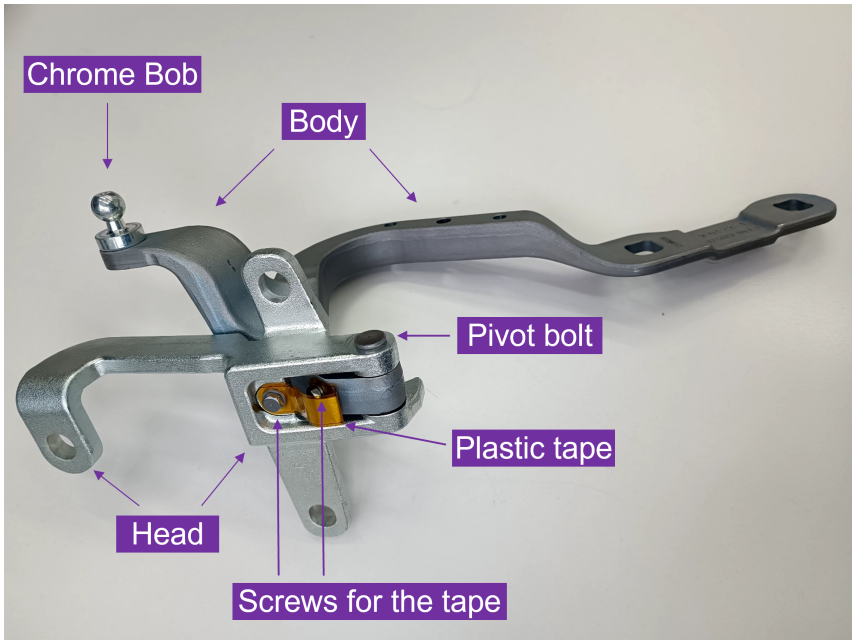



**Figure 17:** Different parts of the hinge

interpretations of material properties. The material properties were initially meticulously modelled in Blender but when exported as .usd and loaded back into Isaac Sim, these materials were not accurately reflected. Isaac Sim has a plugin for Blender to export materials properly in OmniPBR format (Isaac Sim's internal material representation) but this plugin was not available for Linux at that time. As a result, materials were once again applied in Isaac Sim itself. Here, thanks to the VRay Material library (see figure 18) - Available as a part of Isaac Sim's readily available Assets - had lot of realistic metal materials like cast iron, chrome, among others which closely reflected the hinge properties. Following numerous experiments with various materials from the VRay library, we ultimately achieved a highly realistic representation of the hinge (see figure 19).

- **Lack of physics:** As mentioned earlier, Isaac Sim has advanced physics capabilities and can reflect real world friction, weight, collisions, among other properties. Having

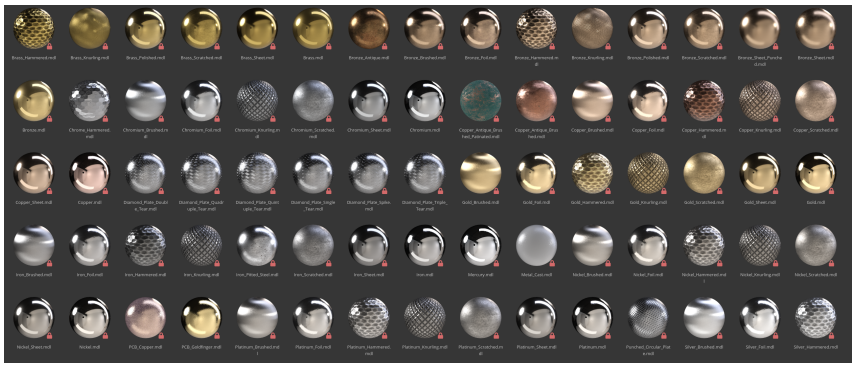


**Figure 18:** Different PBR materials in VRay material library

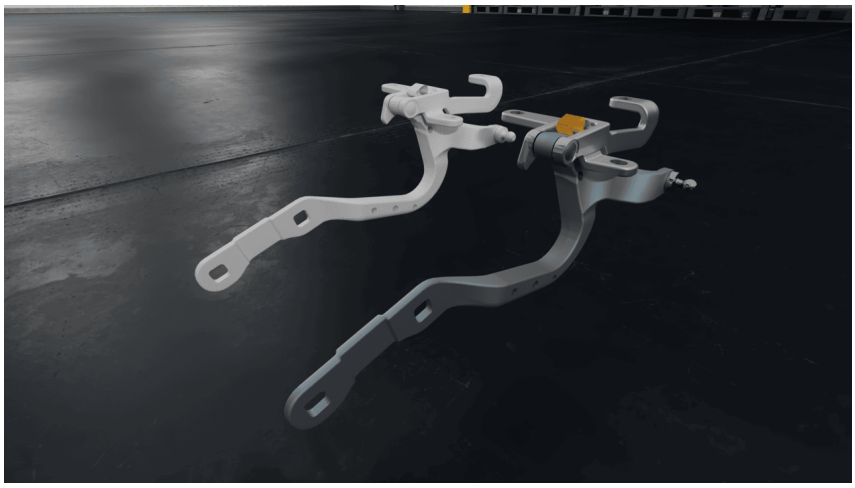


**Figure 19:** Hinge CAD model vs Final hinge with all materials

accurate physics is important for digital twin simulation and consequently for data generation out of this simulation. When objects are added to Isaac Sim simulation, they are only visual objects and no physics properties are associated with them. These properties must be explicitly defined for our custom object - Hinge. Through the object properties panel, "Rigid body physics with colliders preset" is added to the hinge. "Rigid body" refers to objects in simulation that have mass and are affected by external forces like gravity. "Colliders" adds collision meshes to the hinge and allow it to properly collide/interact with other objects it comes in contact with. They are multiple types of collider meshes available with varying degrees of precision. The more precise the collider meshes, the more accurate the collision behavior but at a significant computational cost. Due to the very unique and branch-like structure of the hinge, simple triangular mesh approximation is not accurate enough for precise collisions. As a result, more accurate SDF (Signed Distance field) meshes are being used. The mass of the object is set to 2.5 kg and since only the body part of the hinge will be sliding on the rods and interacted while grasping, material friction is only assigned to this submesh.

Apart from left-right asymmetric structures, both the hinges are identical in all the properties. Hence, this research work focuses only on using the right hinge (this will be referred

to simply as "hinge" for the remaining research work) data generation and testing purposes, with the understanding that all the below findings should similarly apply for the left hinge as well.

**Updating flowrack model**
The flowrack model present initially in the scene was originally imported from a STL CAD file and lacked any material information or physics, just like the hinge. Following the same process of adding material to hinge, the original flowrack was inspected visually, all the visually relevant parts (The frame, the steel rods, the holding brackets and their bolts) were assigned to the materials from the VRay library (see figure 20). For physics, the only interacting part with the flowrack are the steel rods and the final holding bracket at the end - Collision meshes and appropriate friction values were assigned only to these parts to reduce unnecessary computational overhead.

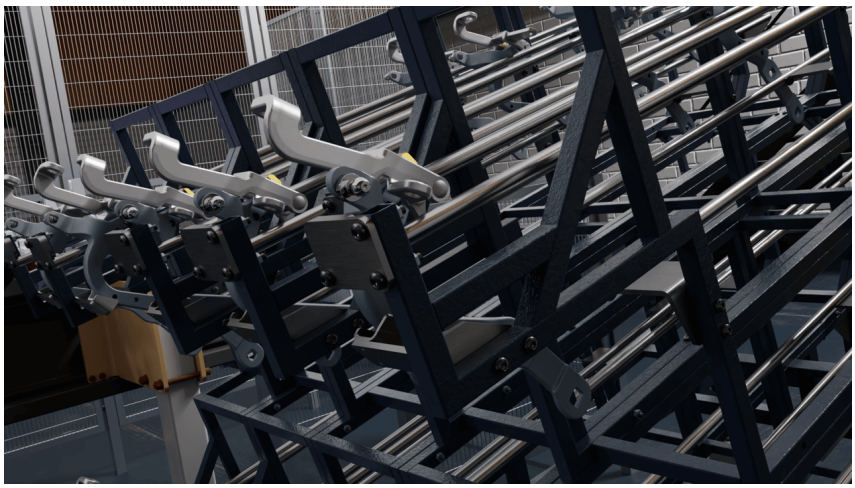


**Figure 20:** Flowrack with PBR materials

**Replicator pipeline**
The data generation for both the datasets takes place through offline scripts that interface through Isaac Sim's inbuilt python and not through any asynchronous extensions. To generate photorealistic images, the Path-traced Renderer is utilized, processing 40 subframes for each image. Although this renderer is slower compared to the standard real-time renderer, it excels in producing more realistic lighting effects within the scene. Subframes serve as a technique to aggregate multiple frames over time, which enhances the overall image quality. This accumulation method effectively reduces noise and artifacts, and contributes to achieving a more accurate representation of lighting. Scripts for generating Industrial domain randomized dataset and Digital twin dataset have slightly different program flow but overall follow the standard structure:

1. In the first part of the script, the Isaac Sim software is started and the necessary python libraries are loaded. This includes omni.Kit, openUSD and replicator library.
2. Then, all the necessary assets are loaded: The environment and required objects. The path of .usd files for the assets are given and then loaded through Isaac Sim's API.

3. Afterwards, a replicator camera with explicit camera properties (focal length, aperture, look-at points) is created. This is just like any other camera in the simulation but with an replicator wrapper. Multiple cameras can also be initiated. Once the cameras are initiated, they are assigned as render_product along with desired resolution - in our case 1280x720 (or commonly referred to as 720p) with focal length equivalent to the Realsense D435 camera. This signifies to the replicator the requirement for data generation using this particular camera, and hence, integrates the camera into the replicator's OmniGraph.

4. Once all the necessary items are created, a replicator pipeline (a type of OmniGraph) has to be initialized. In this pipeline, all the relevant parameters for synthetic data generation are defined: Number of frames, different types of randomizations and data writing format. Replicator's "Randomizers" offer multiple off-the shelf functionality to randomize the basic properties like position/orientation of an object, material properties, textures, lighting in the scene and any custom randomization could be written. With replicator group API, multiple objects could be grouped on the basis of their name or properties and these randomizations could be applied to all of them at once. These randomizations are especially useful for Industrial Domain Randomized dataset.

5. For saving the labelled data, replicator provides the "Writer" functionality. A writer specifies the format of data storage and what labels should be generated - RGB, semantic segmentation, instance, depth map, among others. There are multiple standard writers existing as part of replicator - BasicWriter, KiTTiWriter, YCBVWriter and DOPE Writer. Creating custom writers to output data in arbitrary types are also supported. YCBV Writer outputs data in YCB 6D pose estimation dataset format and it is one of the datasets used in BOP challenge. A modified version of this writer to including the following changes were used in this research work:

    - *Inclusion of instance segmentation maps*: The original writer had only semantic segmentation annotation data and this is not enough to uniquely identify the individual hinges. Consequently, Instance segmentation annotation format was added to the writer.

    - *Unnormalized depth maps*: The original writer output depth maps that were normalized between 0 to 255 for easy visualization purpose. However, the pipeline used in this research work requires depth values in millimeters. Consequently this normalization is removed and scaled from meters to millimeters.

When these randomizers and writer are defined in the code, they are registered individually into an OmniGraph. The number of frames to be generated is input as a CLI (Command line-input) argument and the OmniGraph is iterated in a loop accordingly.

### 3.2.2 Industrial domain randomized dataset

This dataset, as mentioned before, contains the hinges in an industrial background with industrial distractors randomly placed through the environment. Generation of dataset

was heavily inspired from Nvidia AI-IOT pallet-jack synthetic data generation tutorial [4]. The industrial background is "Warehouse with forklifts" (see figure 21) - One of the sample environments available as part of Isaac assets.

**Figure 21:** Warehouse with forklifts base environment

In this scene, the following distractors are spawned and randomly placed on the ground surface: *Palletjacks, cones, wet floor signs, barrels, bottles, cardboard boxes, pushcarts, crates and rackpiles (see figure 22).* No flowrack or robots are spawned in this specific scene.


**Figure 22:** Sample random assets spawned in the scene [4]

Furthermore, the materials for the floor and wall surfaces are randomly generated from a new OmniPBR material, featuring varying levels of roughness, metallic properties, diffusion, and emission characteristics. Additionally, the image textures for these surfaces are randomly selected from the Isaac Materials library. The entire warehouse is illuminated by one invisible rectangular light source - Color and intensity of this light source is also completely randomized. 5 hinges are spawned in the center of the warehouse in a square

region of 0.6x0.6m with height varying from 0.3 to 0.5m with random rotations between 0 and 360 degree (on all axes). The camera's position is randomized to be also within the center square region of 0.8x0.8m with it's orientation always pointing towards the center of the warehouse. This ensures that the camera is at maximum 1.4m from the hinge, similar to the planned usecase and it sufficiently focuses on the 5 hinges in most of the images.

No rigid body physics or collision interactions are enabled for the hinges; instead, they are positioned to float at randomized heights in various locations. This design choice stems from the dataset's primary goal: to enable the pose estimator network(s) to learn features pertinent to the industrial domain and its inherent randomizations. Enabling rigid body physics results in hinges frequently falling onto the ground or occasionally onto distractors. This limits the variety of images, as they predominantly feature hinges with the ground in the background, leading to an underutilization of the scene's potential diversity. Although randomizing ground surfaces introduces new features for the network to learn, this approach risks reducing the dataset to one with mere background variations and no occlusions, thereby deviating from this thesis's core focus.

See figure 23 for sample images from this dataset.

**Figure 23:** Sample images - Industrial domain randomized dataset

### 3.2.3 Digital twin dataset

This dataset is designed to closely reflect the real world usecase and consequently, uses the same digital twin simulation scene with the updated hinges and flowrack model. Rigid body collisions were enabled to allow the hinges slide into the flowrack naturally and

have authentic collision with other hinges. Given that simulating physics and collisions for numerous hinges is computationally intensive and time-consuming, a strategy was employed to mitigate this. Once the hinges settled into their final positions, these static locations were captured and saved in a new USD file. This file was then utilized for subsequent data generation, avoiding the need to repeat the simulation for each new frame.

All the randomizations were turned off except for the light intensity and camera positions. Given the inherent unpredictability of real-world lighting conditions, the experiment varied only the intensity of three fixed overhead lights, maintaining a consistent color. For camera placement, three strategic regions were identified using simulation. These included two regions to match expected data capture orientations, one at the top and another at the bottom row of the flowrack, and a third region along the side of the flowrack. Each region was defined as a 3-dimensional cuboid, characterized by eight vertices with specific coordinates along the x, y, and z axes. In these regions, a significant number of potential camera spawn points were generated to simulate a variety of realistic camera positions. To ensure comprehensive coverage, each region was captured by a separate camera, resulting in a total of three cameras capturing the scene in each frame. To ensure all relevant hinges fell within the camera's field of view, an arbitrary 'look-at' point was manually identified and assigned to each camera. During the generation of each frame, cameras are randomly positioned at one of the pre-determined spawn points, maintaining their respective look-at points. See figure 24 for sample images from this dataset.

**Figure 24:** Sample images - Digital twin dataset

## 3.3 Real data generation

This section dives into the process of collecting the data from the real world and annotating it with 6D pose and mask labels.

### 3.3.1 Data collection

To ensure consistency with the real world usecase, a Framos realsense D435 camera, akin to the one employed in the use case, was utilized for capturing real-world data. The resolution used for capturing RGB-D data matches that of the synthetic dataset, set at 1280x720. The data is captured using a modified version of lcm-logger present in Labelfusion [38]. This logger works with ROS2 and saves synchronized RGB-D data from the camera at 30 FPS (Frames per second). To make it easy to carry around and operate, the Framos camera was attached to a Ubuntu based laptop and the lcm-logger ran in a docker containerized environment in this laptop for data capture.

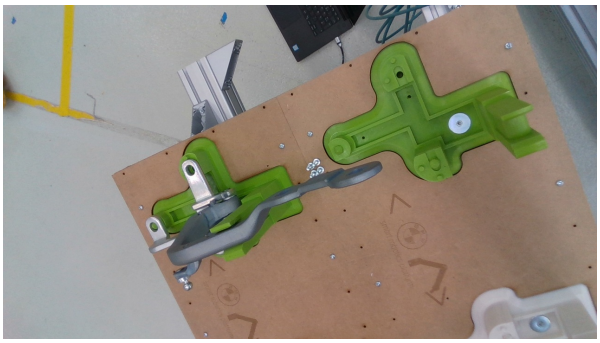
Since LabelFusion uses SLAM based reconstruction of the entire scene, it is crucial to capture smooth, jerk-free and stable footage from the camera. The chosen Framos Realsense camera, unlike modern smartphones [39], lacks built-in optical or electronic stabilization, thus requiring external means to achieve steadiness. An attempt was made to use the DJI Osmo Mobile 3 Gimbal [6], a well-known smartphone stabilizer, for this purpose. However, the camera's light weight, combined with disturbances caused by the movement of the attached USB-C cable, rendered the gimbal ineffective for stabilization in this context.

A total of 6 scenes were recorded. 5 of these scenes were obtained using a hand-held camera, employing slow and steady movements. In contrast, the 6th scene was captured with the camera mounted on the robot's end-effector, which enabled precise movements. Each scene was recorded for a duration ranging from 30 to 60 seconds, yielding approximately 900 to 1800 frames per scene. The captured scenes include:

1. **Scene 1:** This scene contained one hinge lying on the side on a work desk. The camera was steadily moved by hand to cover all possible angles of the hinge.
2. **Scene 2:** This scene contained one hinge placed inverted on custom 3D printed bracket, replicating the conveyor belt. A smooth motion surrounding all sides of the hinge was used to capture the footage.
3. **Scene 3:** This scene had a very basic replica of the flowrack constructed using aluminium tubes and steel pipes. This was constructed to have a sample flowrack that could be used to verify the expected orientation of the robot for possible grasping. A simple arc-like motion with camera looking on to the hinges from the bottom part was used for capturing the footage.
4. **Scene 4:** The flowrack matching the usecase was made available towards the later part of thesis work and one video capturing the hinges from the various angles were shot using hand movement. This scene contained 5 hinges in total

5. **Scene 5:** 2 footages were captured with camera attached to robot's end-effector and controlled to move in-front of hinges in a slow and steady fashion. These scenes also contained 5 hinges in total.
6. **Scene 6 (Test data):** This is also a footage from the flowrack but at a different position/lighting condition captured with hand. This scene also contained 5 hinges in total. This footage is exclusively used for evaluating the accuracy of the models.

**(a)** Scene 1

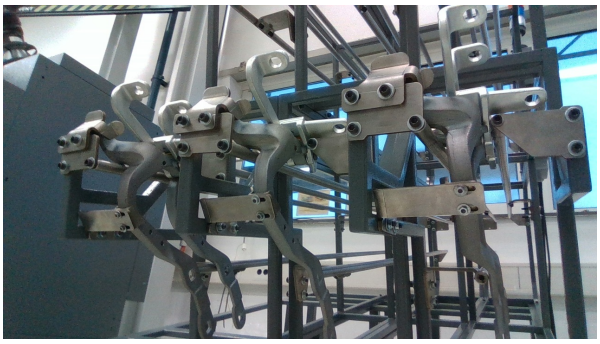


**(b)** Scene 2

**(c)** Scene 3

**(d)** Scene 4

**(e)** Scene 5



**(f)** Scene 6 (Test data)

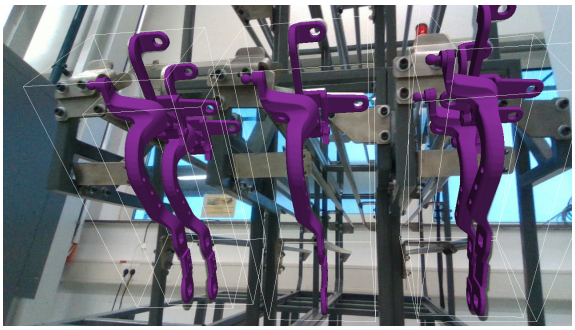
**Figure 25:** Sample images - Collected scenes

### 3.3.2 Data labelling

The collected footages (from section 3.3.1) are then trimmed to remove any unwanted or repetitive sections with no motion. Then, all the steps as mentioned in section 2.1.4 for labelfusion are followed:

1. The trimmed data is then processed by ElasticFusion, a SLAM-based point cloud reconstruction algorithm. Smooth motion is crucial for this algorithm as it depends on consistently tracking distinct environmental features over time to accurately estimate the system's motion and construct a map. Jerky movements disrupt this process by rapidly altering the camera's viewpoint, thereby hindering the reliable tracking of features across successive frames. Moreover, SLAM systems naturally accumulate errors over time, and abrupt, unpredictable movements amplify this, resulting in significant deviations from the actual path and map distortions. A critical component of SLAM is loop closure, which corrects these accumulated errors by identifying previously visited areas. However, jerky motions can obstruct effective loop closure by altering the perceived appearance of these locations due to sudden perspective shifts. In footages captured by hand, even minor jerkiness led to slight distortions in the reconstructed point cloud. Interestingly, attempting loop closure in these scenarios often exacerbated the distortion. Consequently, such footage was selectively trimmed to exclude segments where loop closure occurred.
2. Then the reconstructed pointcloud is loaded into labelfusion. Here, three key steps are done:
    a) *Orient the pointcloud:* The pointcloud from SLAM is rotated arbitrarily. This needs to be first rotated so that it faces the right side-up.
    b) *Segment the pointcloud:* When capturing a scene with the camera, all the background stuff in the scene may also get reconstructed and this makes the pointcloud cluttered to work with. Hence, only the relevant area of the pointcloud is segmented
    c) *Object Alignment:* Then the preloaded .stl file of the hinge is available as a 3D model and three distinct points should be matched between a single object in the pointcloud and the 3D model. After this, ICP algorithm is run in an attempt to try and fit the model to the pointcloud. This returns an initial 6D pose of the model which is further refined by manually adjusting the translation and rotational vectors. This step has to be done individually for every single object in the scene.
3. Then, the annotated segmentations masks and 6D pose estimation data is exported into a custom labelfusion data format with the corresponding RGB images and depth maps. Additionally, to facilitate the visualization of labeled poses, the 3D CAD model is projected onto these poses and saved for each image (see figure 26).

**(a)** Scene 1 labelled image



**(b)** Test data labelled image

**Figure 26:** Sample labelled images with pose projections

## 3.4 Training tools

This section describes the setting up of the tools for training the networks. Workstations with same configurations, as used in the synthetic data generation, were also used for training and testing these networks.

### 3.4.1 Data conversion

Since Mask R-CNN requires COCO dataset format and GDR-NPP requires BOP dataset format, both synthetic datasets were converted through python scripts into both the formats. Very small or highly occluded objects are harder to detect with simple CNN based networks and hence, a simple filtering strategy was employed to remove highly occluded objects from the scene based on their visibility ratio and area of their bounding boxes. Conversion to BOP format was complicated, due to the generation of visible and complete masks for each object in the scene, which involves projecting the labelled pose and comparing with depth maps for possible occlusions. For this purpose, an internally developed set of tools based on BOP toolkit was utilized for proper conversion into BOP and COCO formats.

### 3.4.2 Mask R-CNN

An internally developed code-base was used for training this network [38]. This was based on the popular deep learning framework PyTorch [36] with COCO dataset format [29] was required for training the network. The necessary tools for evaluating the network across the vital COCO metrics and for visualizing the outputs were also available.

### 3.4.3 GDR-NPP

The GitHub repository for the original GDR-NPP [30] was adapted specifically for the hinge dataset. This included adding the number of training scenes, testing scenes and

the specific resolution being used. This code-base was based on openMMLab Computer Vision toolkit (MMCV) [11] which internally depends heavily again on PyTorch framework and required dataset in BOP dataset format. All required dependencies for the GDR-NPP model were encapsulated within a Docker image. This image was utilized to conduct training sessions across multiple workstations, ensuring consistency in the base environment.

# 4 Experiments

In this section, the experiment setup required to answer the relevant questions for this thesis topics is defined:

1. Evaluate the necessity of using physically accurate digital twin based synthetic dataset in comparison to industrial domain randomized synthetic dataset for training GDR-NPP 6D pose estimator
2. Evaluate the accuracy gain by mixing real annotated images with the synthetic dataset

## 4.1 Data processing

Using the synthetic data generation pipeline mentioned in section 3.2.1, **25000 samples** (includes RGB, depth maps, instance/semantic segmentation masks and pose data) are generated for each Industrial domain-randomized dataset and Digital twin dataset. As for the real labeled dataset, it is bifurcated into two distinct sets: a real training set and the test set. The real training set comprises scenes 1-4 and a limited subset (approximately 10%) of actual flowrack images (scene 5). The foundational concept of this thesis focuses on leveraging simulation for a hypothetical use case that does not completely exist. Consequently, the data with actual flowrack is only available in limited quantity during the training phase, thereby relegating it mostly to the test set. The other scenes (scenes 1 - 4), serving as temporary solutions, can be rapidly prototyped in a factory setting and are therefore included in the real training set. As mentioned earlier in the section 3.3.1, the test set includes also images from actual flowrack but viewed from a different position and lighting condition, making it a viable choice for testing the performance of the networks. Owing to the minimal relative motion between consecutive frames, a sampling strategy was adopted where only every 10th frame was selected for further analysis for training data and every 30th frame was selected for test data. This results in **total of 376 images in real training set and 70 for the test set.**

## 4.2 Training details

To evaluate the first research question, Mask R-CNN and GDRNPP algorithms are each trained solely on Digital twin dataset, Industrial domain-randomized dataset. Then they are evaluated individually on the test set. To evaluate GDRNPP separately, the bounding box priors required for cropping into the objects are loaded from ground-truth annotation data. Then both Mask R-CNN and GDR-NPP are evaluated jointly together. Similarly, to evaluate the impact of mixing real world data, a comparable training strategy is used. The primary distinction in this approach is that each synthetic dataset is augmented by including the real training set as well. This results in 4 total experiments consisting of 8 trainings (see table 1).

To train Mask R-CNN, A pretrained ResNet50-FPN (Feature Pyramid Network) was chosen as the backbone. Since pretrained networks don't need a lot of training steps to achieve good accuracy, the network was only trained for 50 epochs with batch size 32 at 1280x720 resolution. This resulted in an average training time of 13 hours.

To train GDR-NPP, A pretrained ConvNext-Base [49] backbone was used. All the vital hyperparameters: learning rate, batch size, image augmentation, backbone, among others have been left untouched - The only change being the resolution: 1280x720 (Default: 640x480). Similar to the original repository, the network was trained for 50 epochs at a batch size of 48. This resulted in an average training time of 14 hours.

| **Exp. No.** | **Model** | **Train dataset** |
|---|---|---|
| 1 | GDR-NPP Mask-RCNN | Digital Twin |
| 2 | GDR-NPP Mask-RCNN | Industrial Ran. |
| 3 | GDR-NPP Mask-RCNN | Digital Twin + Real training set |
| 4 | GDR-NPP Mask-RCNN | Industrial Ran. + Real training set |

**Table 1:** Training experiments

# 5 Results

This section discusses the necessary metrics to understand the results from the experiments and then dives deep into the insights about it.

## 5.1 Metrics

To evaluate Mask R-CNN, the common segmentation metrics Average Precision (AP) and Average Recall (AR) for bounding boxes are being used. Precision refers to the accuracy of the positive predictions made by the model and Recall refers to the model's ability to identify all instances in the scene. Usually, there is a trade-off between precision and recall - Improving one of them can result in the degradation of another. For Mask R-CNN, the average precision is calculated by averaging the precision across different IoU thresholds (0.5 and 0.5 to 0.95) over all samples. Similarily, the average recall at 0.5-0.95 is average of recall values for 100 detections across different IoU thresholds (0.5 to 0.95). Detections with scores below the threshold of 0.4 were excluded, as this value was established as the minimum acceptable score.

For GDR-NPP, this precision and recall is calculated at the distance thresholds 0.7cm, 1cm, 2cm, 5cm and 10cm for ADD and MDD separately. A detection is considered to be true positive if distance error is below the threshold and matches with a ground truth pose. Conversely, if it matches with a ground truth pose and lies above the error threshold or if it does not match with any ground pose, it is considered false positive. If a ground truth pose is not matched at all or only match with detections above the error threshold, it is considered as false negative. These evaluations metrics were directly adopted from this paper [38].

## 5.2 Discussion

This subsection discusses the results as observed from the training runs. In each column of the table, the highest value is highlighted with a pale green background. The name of the training runs are abbreviated as follows:

| **Dataset** | **Abbreviation** |
|---|---|
| Industrial domain randomized | Ind.DR |
| Digital Twin | Dig.Twin |
| Industrial Ran. + Real training set | Ind.DR + R |
| Digital Twin + Real training set | Dig.Twin + R |

**Table 2:** Abbreviated dataset names

### 5.2.1 Only Mask R-CNN

In Table 3, we summarize all accuracy metrics, along with the respective epochs at which these accuracies were measured. Among the synthetic datasets evaluated, the Dig. Twin dataset achieves the highest accuracy by a considerable margin. This suggests that the hinge positions within the Dig. Twin dataset closely reflect those in real-world scenarios, allowing the Mask R-CNN to learn more representative features effectively. When the datasets are mixed with real images, the Ind.DR+R dataset exhibits higher accuracy in AP@0.5. However, for both AP@0.5-0.95 and AR@0.5-0.95 metrics, the Ind.DR + R dataset shows only a marginal increase in accuracy compared to the Dig.Twin + R dataset. The distinctions between these datasets in terms of performance can be more clearly visualized in figure 27. An intriguing observation emerges when considering the number of epochs required to attain the highest accuracy. For both synthetic datasets, peak accuracy is achieved as early as the first epoch. This indicates that the pre-trained features of the backbone contribute significantly to rapid generalization to the given dataset. In contrast, when the datasets are combined with real images, the network necessitates more epochs to adapt and learn the intricate features. This distinction is particularly noteworthy, given the potential for labeling errors in the mixed datasets.

| Dataset | Epoch | AP@0.5 | AP@0.5-0.95 | AR@0.5-0.95 |
|---|---|---|---|---|
| Ind.DR. | 1 | 0.149 | 0.044 | 0.111 |
| Dig.Twin | 1 | 0.199 | 0.112 | 0.127 |
| Ind.DR + R | 23 | **0.577** | **0.429** | **0.473** |
| Dig.Twin + R | 37 | 0.550 | 0.425 | 0.465 |

**Table 3:** Mask R-CNN training results

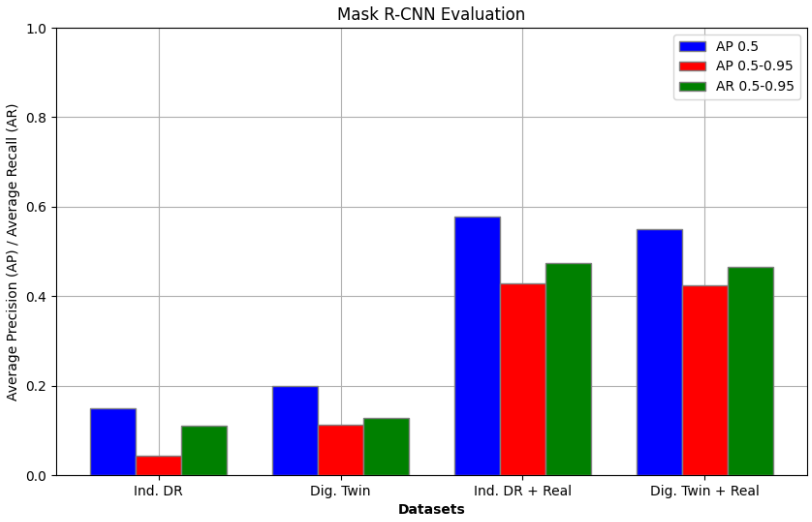



**Figure 27:** Mask R-CNN evaluation

However, the accuracy levels obtained from all the datasets currently do not meet the high standards required for effective and reliable implementation in industrial logistics.

### 5.2.2 Only GDR-NPP

In tables 4 and 5, the performance metrics for ADD and MDD evaluations at various thresholds are presented, focusing solely on the GDR-NPP model. As previously mentioned, these evaluations utilize ground-truth bounding boxes from the original annotation files to infer 6D poses. When considering results using synthetic datasets alone, the Ind.DR dataset significantly outperforms the Dig.Twin dataset. This superior performance can be attributed to the wider range of poses in the Ind.DR dataset, where hinges are represented in a broad spectrum of orientations, thereby providing a richer learning experience for the network. This trend persists when the datasets are mixed with real data, with Ind.DR + R outperforming Dig.Twin + R, for reasons consistent with the synthetic data analysis.

The evaluation of the MDD Error, which is inherently more challenging but particularly important for this usecase, reveals that even the best-performing dataset, Ind.DR+R, falls short of the required accuracy at 0.7cm and 1cm thresholds for applications in robotic grasping.

| **Dataset** | **0.7cm** | | **1cm** | | **2cm** | | **5cm** | | **10cm** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AP | AR | AP | AR | AP | AR | AP | AR | AP | AR |
| Ind.DR | 0.007 | 0.007 | 0.014 | 0.015 | 0.144 | 0.242 | 0.627 | 0.706 | 0.877 | 0.908 |
| Dig.Twin | 0.000 | 0.000 | 0.000 | 0.000 | 0.030 | 0.047 | 0.314 | 0.439 | 0.679 | 0.809 |
| Ind.DR + R | 0.011 | 0.024 | 0.060 | 0.133 | 0.387 | 0.569 | 0.880 | 0.934 | 0.981 | 0.990 |
| Dig.Twin + R | 0.000 | 0.000 | 0.000 | 0.000 | 0.113 | 0.208 | 0.833 | 0.914 | 0.946 | 0.966 |

**Table 4:** Only GDRNPP - ADD error threshold evaluation

| **Dataset** | **0.7cm** | | **1cm** | | **2cm** | | **5cm** | | **10cm** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AP | AR | AP | AR | AP | AR | AP | AR | AP | AR |
| Ind.DR | 0.000 | 0.000 | 0.000 | 0.000 | 0.017 | 0.022 | 0.353 | 0.487 | 0.798 | 0.845 |
| Dig.Twin | 0.000 | 0.000 | 0.000 | 0.000 | 0.003 | 0.006 | 0.084 | 0.131 | 0.315 | 0.434 |
| Ind.DR + R | 0.000 | 0.000 | 0.004 | 0.007 | 0.072 | 0.147 | 0.632 | 0.772 | 0.896 | 0.950 |
| Dig.Twin + R | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.364 | 0.520 | 0.864 | 0.918 |

**Table 5:** Only GDRNPP - MDD error threshold evaluation

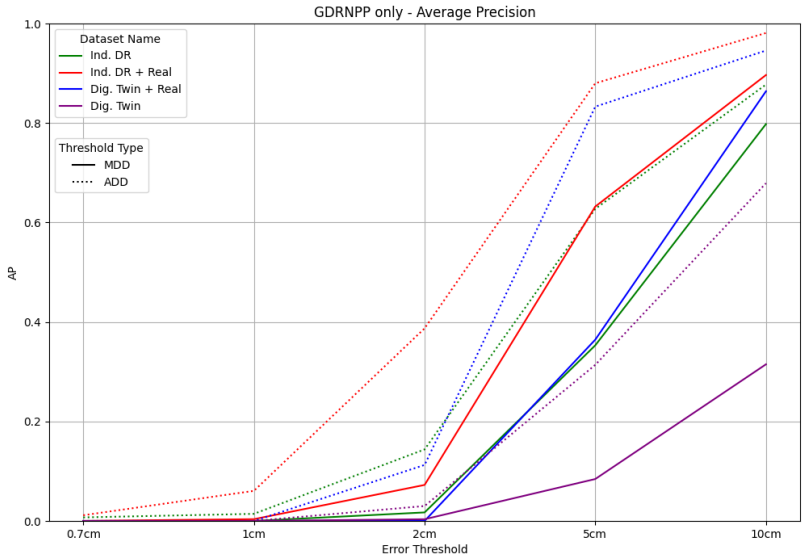



**Figure 28:** Only GDRNPP - Average precision (AP)

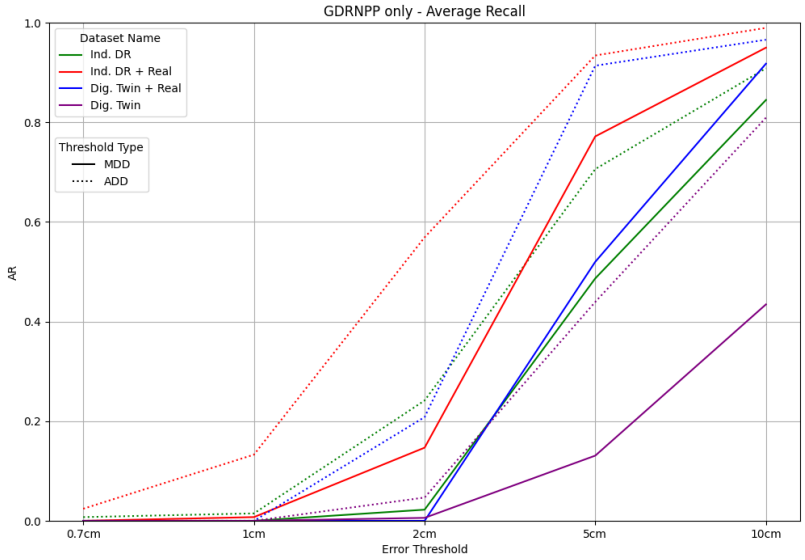



**Figure 29:** Only GDRNPP - Average recall (AR)

### 5.2.3 Complete pipeline

In tables 6 and 7, we present the performance metrics for the ADD and MDD evaluations, specifically focusing on the complete pipeline. In this context, the bounding box priors for GDR-NPP are derived from the predicted masks generated by Mask R-CNN.

When analyzing solely the synthetic dataset, it is observed that the Dig.Twin dataset outperforms the Ind.DR dataset at error thresholds below 2cm for both ADD and MDD metrics. However, Dig.Twin exhibits lower AR values at error thresholds beyond 2cm. This trend is consistent across both metrics.

In scenarios where the datasets are combined with real data (Ind.DR + R and Dig.Twin + R), a distinct pattern emerges. Ind.DR + R demonstrates significantly better performance than Dig.Twin + R at thresholds under 2cm, and continues to maintain higher AR values beyond this threshold. The superior performance of Ind.DR + R can be attributed to its incorporation of fully randomized elements such as lighting and textures, which enhances the network's ability to learn a more diverse range of features.

A critical observation is the recording of 0 AP and AR values at MDD thresholds ranging from 0.7cm to 1cm. This finding underscores a pivotal limitation of the current pipeline's applicability in robotic grasping applications; it cannot be reliably employed for precision tasks. While higher accuracies are noted at larger thresholds (5cm or 10cm), these are not viable for applications requiring high precision, such as in the case of a hinge measuring 36.7cm end-to-end, where such large errors cannot be deemed acceptable. The AP and AR data have been clearly visualized in the figures 30 and 31.

| **Dataset** | **0.7cm** | | **1cm** | | **2cm** | | **5cm** | | **10cm** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AP | AR | AP | AR | AP | AR | AP | AR | AP | AR |
| Ind.DR | 0.000 | 0.000 | 0.004 | 0.010 | 0.009 | 0.030 | 0.196 | 0.411 | 0.408 | 0.577 |
| Dig.Twin | 0.000 | 0.000 | 0.024 | 0.010 | 0.101 | 0.040 | 0.226 | 0.085 | 0.369 | 0.141 |
| Ind.DR + R | 0.000 | 0.000 | 0.051 | 0.078 | 0.281 | 0.386 | 0.684 | 0.858 | 0.789 | 0.912 |
| Dig.Twin + R | 0.000 | 0.000 | 0.008 | 0.010 | 0.186 | 0.252 | 0.733 | 0.754 | 0.892 | 0.863 |

**Table 6:** Complete pipeline - ADD error threshold evaluation

| **Dataset** | **0.7cm** | | **1cm** | | **2cm** | | **5cm** | | **10cm** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AP | AR | AP | AR | AP | AR | AP | AR | AP | AR |
| Ind.DR | 0.000 | 0.000 | 0.000 | 0.000 | 0.004 | 0.010 | 0.088 | 0.185 | 0.288 | 0.486 |
| Dig.Twin | 0.000 | 0.000 | 0.000 | 0.000 | 0.024 | 0.010 | 0.122 | 0.052 | 0.256 | 0.095 |
| Ind.DR + R | 0.000 | 0.000 | 0.000 | 0.000 | 0.040 | 0.081 | 0.517 | 0.655 | 0.711 | 0.873 |
| Dig.Twin + R | 0.000 | 0.000 | 0.000 | 0.000 | 0.009 | 0.011 | 0.390 | 0.453 | 0.809 | 0.802 |

**Table 7:** Complete pipeline - MDD error threshold evaluation

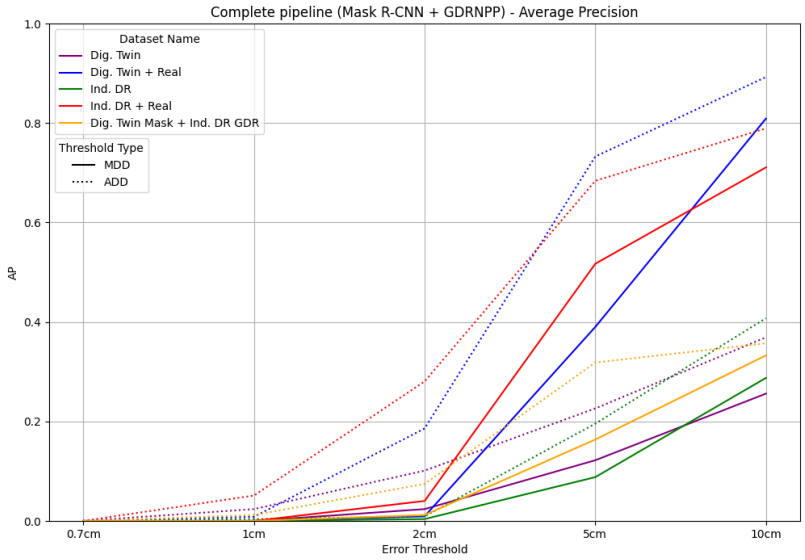



**Figure 30:** Complete pipeline - Average precision (AP)

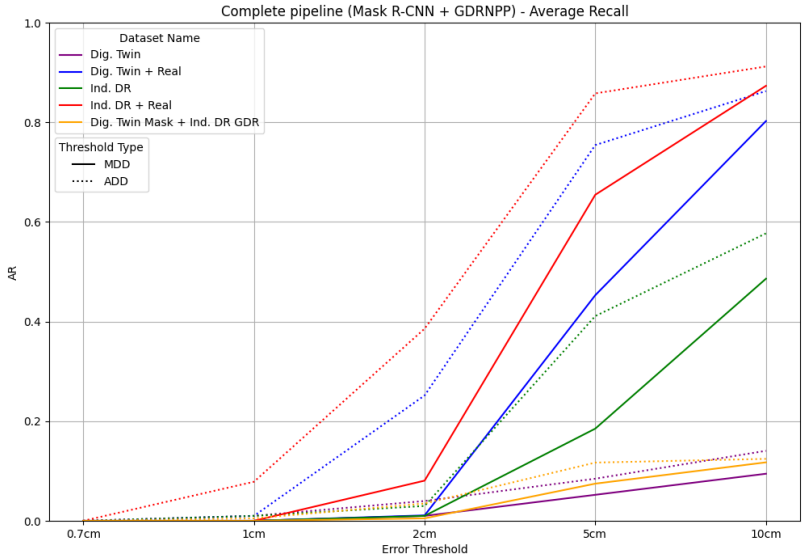
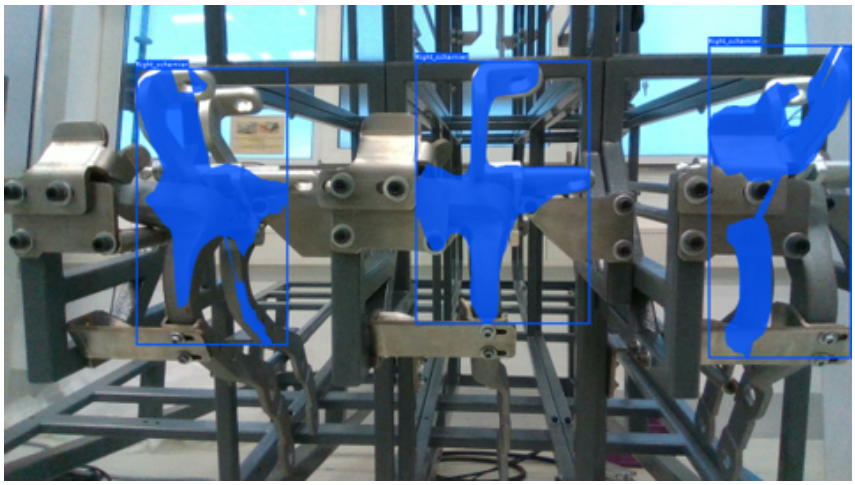
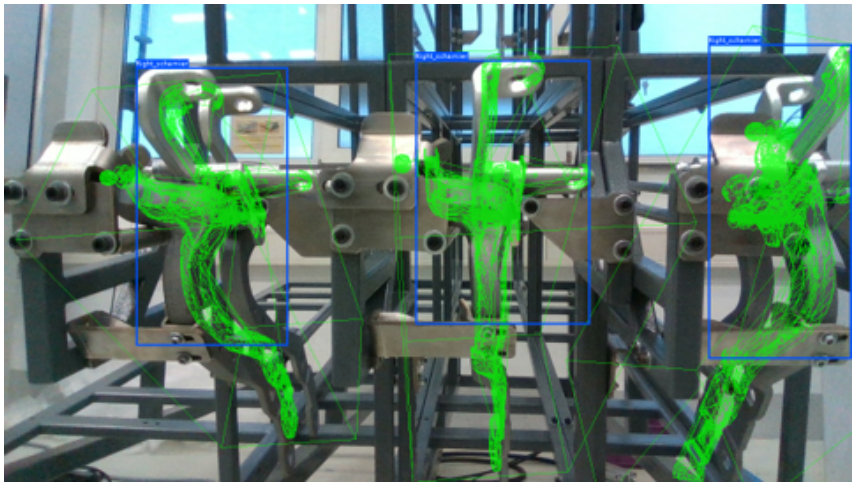



**Figure 31:** Complete pipeline - Average recall (AR)



**(a)** Mask R-CNN predictions



**(b)** GDR-NPP predictions

**Figure 32:** Sample visualization - Ind.DR + real

### 5.2.4 Dig.Twin Mask R-CNN and Ind.DR GDR-NPP

In the analysis of the synthetic datasets as detailed in subsections 5.2.1 and 5.2.2, it was observed that the Dig.Twin dataset yielded higher accuracy when used with Mask R-CNN, whereas the Ind.DR dataset showed better results with GDR-NPP. This led to the hypothesis that combining both networks in a single pipeline might result in enhanced accuracy, surpassing the performance of each individual pipeline.

To test this hypothesis, a specialized pipeline integrating both networks was developed and its performance was evaluated (refer to table 8). The results of this combined approach were intriguing. While the Average Precision (AP) values were higher than those achieved by the purely Ind.DR pipeline, they consistently fell short of the AP values for the pure Dig.Twin pipeline. On the other hand, the Average Recall (AR) values were superior to those of the Dig.Twin pipeline for error thresholds above 5cm.

This behavior suggests that the networks learn distinct features from each dataset. When combined, the more accurate bounding box predictions from one network potentially aid the GDR-NPP in achieving slightly more precise regression. However, the combined pipeline does not outperform the pure Dig.Twin pipeline, possibly because GDR-NPP is not as adept at handling the features specific to the Dig.Twin dataset. The analysis here is that while the integration of these networks leads to some improvements in certain metrics, it does not universally enhance performance across all parameters

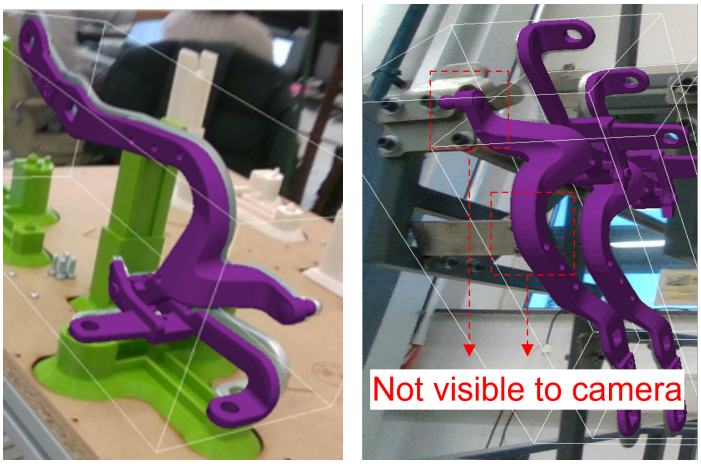
| Eval Type | 0.7cm | | 1cm | | 2cm | | 5cm | | 10cm | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AP | AR | AP | AR | AP | AR | AP | AR | AP | AR |
| ADD | 0.000 | 0.000 | 0.012 | 0.005 | 0.074 | 0.035 | 0.318 | 0.117 | 0.357 | 0.124 |
| MDD | 0.000 | 0.000 | 0.000 | 0.000 | 0.012 | 0.005 | 0.164 | 0.075 | 0.333 | 0.117 |

**Table 8:** Complete pipeline - Using Dig.Twin Mask R-CNN and Ind.Dr GDRNPP

## 5.3 Limitations

During the development and testing of this pipeline, many limitations were discovered and are summarized as follows:

1. **Data labelling errors:** As mentioned previously in subsection 2.1.4, the reconstruction of accurate pointcloud and subsequent reprojection of labelled poses (see figure 33a), heavily depends on the performance of the SLAM algorithm. This algorithm is prone shaky, blurred images, which are inherently created while moving the camera with hand. Such pose errors could reach up to 1cm, which compromises the evaluation of the critical detection threshold range of 0.7cm to 1cm. Furthermore, the mask projected by Labelfusion doesn't account for the occlusions caused by the static unlabelled structures in the environment, as a result parts of the hinge are marked as visible even though they are slightly occluded by the holding brackets (see figure 33b).




**(a)** Pose reprojection error **(b)** Occlusion error

**Figure 33:** Labelfusion errors

2. **Variable hinge head position:** The hinge's head is designed to move smoothly around its pivot, as illustrated in figure 13. Consequently, hinges loaded into the flowrack may exhibit minor positional variations. The CAD model of the hinge, used in this research work, represents the hinge in a closed state. This static model is utilized across various stages: labeling real-world data, generating synthetic data, and training the GDR-NPP network. Variations in the hinge head's position, particularly when it is not fully closed, can introduce discrepancies in both the labeled data and the predicted poses. The entire hinge, inclusive of its head, was selected based on the hypothesis that a larger surface area and distinctive colors would enhance the pipeline's ability to accurately detect the hinges. A potential resolution for these discrepancies is to focus exclusively on the static body of the hinge in future developments.

3. **Sim2Real gap:** Isaac Sim stands out as one of the leading simulators, known for producing photorealistic images and accurately simulating physical interactions. However, it encounters inherent challenges in perfectly mirroring the complexity and unpredictability of real-world physics and materials. Moreover, simulating more intricate physics scenarios can become prohibitively computationally demanding. For instance, in creating the digital twin dataset, computational and time constraints led us to simulate only one scene. This single scene formed the basis for generating all images in the dataset. This limitation increases the risk of model overfitting to the relative poses of hinges, potentially leading to the model learning an undesirable bias. With additional time and computational resources, it would be possible to simulate a greater variety of scenes, introducing subtle positional variations in the hinges. Such variations could potentially enhance the model's ability to learn more robust features. Additionally, real-world sensors, including cameras and depth sensors, often introduce noise and distortions, aspects that are not consistently replicated in Isaac Sim simulations. This discrepancy presents further challenges in bridging the gap between simulated and real-world data.

4. **Limited training runs:** The training runs for this project were conducted only once due to constraints in time and computational resources. Notably, despite initializing with pretrained weights, deep learning implementations often exhibit inherent randomness due to factors in libraries, software stacks [23], and hardware variations. Such randomness could potentially lead to fluctuations in accuracy across different training iterations even under identical parameter settings. This aspect of variability was not explored in the current research, and therefore, the results should be interpreted with this consideration in mind.

# 6 Conclusion

This thesis investigates the feasibility and challenges associated with employing synthetic data, derived from a digital twin, for 6D pose estimation. It assesses this approach by comparing its performance with an industrial domain-randomized dataset, with a specific focus on utilizing Nvidia IsaacSim. The study examines the 6D pose estimation pipeline involving Mask R-CNN and GDR-NPP, and additionally explores the accuracy improvements achieved by mixing a small proportion of real data into the synthetic dataset.

The key findings of this research can be summarized as follows:

1. **Benefits of using data generated from digital twin:** This research work shows that out of the models trained from only synthetic datasets, digital twin dataset performs better than just industrial domain-randomized dataset for only Mask R-CNN as well as for the combined pipeline (as seen in 3, 7 and in figure 34). In the presented graph, we focus solely on the most relevant MDD error thresholds of 0.7cm, 1cm, and 2cm, as previously stated, since larger errors are unacceptable for this specific use case. It's also crucial to note the scale of the y-axis; despite the apparent steep slope, the top-performing digital twin dataset achieves a maximum AP of just 2%.

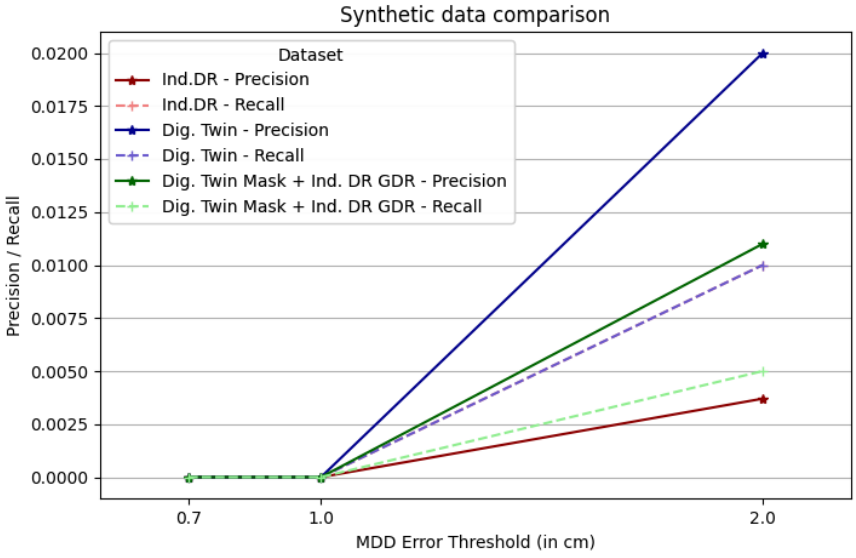



**Figure 34:** Synthetic dataset only comparison

2. **Gain in accuracy by mixing by real images:** By mixing only a small quantity of real images (exactly 1.5 %), a significant improvement in accuracy is noticed along both the datasets - Especially, Industrial domain randomized dataset mixed with real images consistently across most of the categories.

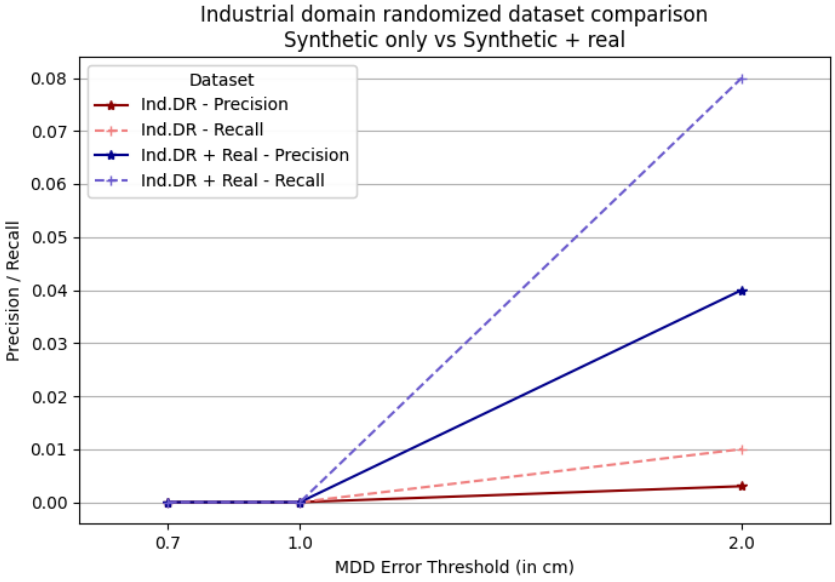



**Figure 35:** Ind.DR dataset - Synthetic only vs Mixed with real comparison

3. **Application to robotic grasping:** The study revealed a significant limitation in applying these models to high-precision tasks such as robotic grasping. The accuracy levels at lower error thresholds (0.7cm to 1cm) were not sufficient for reliable deployment in practical applications, particularly in logistics and manufacturing environments.
4. **Challenges and limitations:** Despite the advantages, several limitations were encountered. The most prominent is the Sim2Real gap, where discrepancies between the simulated and real-world environments led to challenges in model accuracy and generalization. Additionally, data labeling errors and the variability in hinge head positions further complicated the model's accuracy and reliability.

## 6.1 Future work

Deriving from the limitations and from general observations made during the research work, the following aspects could be improved in the future:

1. **Using better cameras:** The present system performs pose regression exclusively using RGB data. Due to the suboptimal quality of the stereoscopic depth data obtained from the camera, this information is not utilized for depth-based pose refinement. Implementing a superior RGB-D sensor, or employing AI-based stereo depth estimation algorithms [27, 50], could significantly enhance the quality of depth estimation - This could then be utilized for precise depth based pose refinement.
2. **Reducing the Sim2Real gap:** The digital twin of the scene has all the vital elements of the scene but misses considerable number of environmental assets and most importantly the accurate lighting information from the scene. Furthermore,

the Framos camera has an inherent level of noise in it's RGB sensor and this is not at all reflected in Isaac Sim. A simple-and-cheap trick to overcome this would be to augment the RGB data with noise speckle, or more complex but accurate method would be to replicate the physics based noise characteristic in Isaac Sim.

3. **Optimizing the network hyperparameters:** In this study, the hyperparameters for both the Mask R-CNN and GDR-NPP networks were kept constant across all datasets to ensure comparability. However, fine-tuning key hyperparameters such as learning rate, batch size, and more augmentations [26], along with experimenting with newer network backbones and more effective loss functions, could potentially lead to improvements in accuracy.

# Literature references


[1] 3D-Daten durch Stereo Vision | IDS. https://de.ids-imaging.com/technical-articles-details/whitepaper-depth-information-3d-images.html. [Accessed 10-01-2024].

[2] A Brief Analysis of the Principles of Depth Cameras: Structured Light, TOF, and Stereo Vision. | DFrobot. https://wiki.dfrobot.com/brief_analysis_of_camera_principles. [Accessed 23-12-2023].

[3] Mit Cobots Ihre Produktion automatisieren | Universal Robots. https://www.universal-robots.com/de/produkte/. [Accessed 04-01-2024].

[4] Nvidia AI-IoT: "Workflow for generating synthetic data and training cv models". https://github.com/NVIDIA-AI-IOT/synthetic_data_generation_training_workflow/tree/main. [Accessed 06-01-2024].

[5] NVIDIA Omniverse Licensing & Pricing | Nvidia. https://www.nvidia.com/en-us/omniverse/download/. [Accessed 05-01-2024].

[6] Osmo Mobile 3 | DJI. https://www.dji.com/de/osmo-mobile-3?site=brandsite&from=landing_page. [Accessed 07-01-2024].

[7] ALL3DP. What Is an STL File? – The STL Format Simply Explained | All3dp. https://all3dp.com/1/stl-file-format-3d-printing/, 2023. [Accessed 05-01-2024].

[8] G. Andrews. Using synthetic data to train autonomous mobile robots with nvidia omniverse replicator for isaac sim. *NVIDIA Developer Blog*, November 2023.

[9] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang. Understanding domain randomization for sim-to-real transfer. In *International Conference on Learning Representations*, 2022.

[10] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[11] M. Contributors. MMCV: OpenMMLab computer vision foundation. https://github.com/open-mmlab/mmcv, 2018.

[12] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, M. Elbadrawy, H. Katam, and J. Vidal. Blenderproc. https://github.com/DLR-RM/BlenderProc, 2023.

[13] M. Denninger, D. Winkelbauer, M. Sundermeyer, W. Boerdijk, M. Knauer, K. H. Strobl, M. Humt, and R. Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82), 2023.

[14] M. Ebni, S. M. Hosseini Bamakan, and Q. Qu. Digital twin based smart manufacturing; from design to simulation and optimization schema. *Procedia Computer Science*, 221, 2023. Tenth International Conference on Information Technology and Quantitative Management (ITQM 2023).

[15] L. Eversberg and J. Lambrecht. Generating images with physics-based rendering for an industrial object detection task: Realism versus domain randomization. *Sensors*, 21(23), 2021.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[17] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 International Conference on Computer Vision (ICCV)*, 2011.

[18] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, editors, *Asian Conference on Computer Vision (ACCV)*, 2013.

[19] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. In *Computer Vision – ECCV 2018 Workshops: Munich, Germany, September 8-14, 2018, Proceedings, Part I*, Berlin, Heidelberg, 2019. Springer-Verlag.

[20] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[21] T. Hodaň, P. Haluza, v. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[22] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha, and B. Guenter. Photorealistic image synthesis for object instance detection. In *2019 IEEE International Conference on Image Processing (ICIP)*, 2019.

[23] M. Kari. Reproducible ML Models using Docker — blog.mkari.de. https://blog.mkari.de/posts/reproducible-ml-models-using-docker/, 2020. [Accessed 19-01-2024].

[24] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10 2017.

[25] S.-h. Kim and Y. Hwang. A survey on deep learning based methods and datasets for monocular 3d object detection. *Electronics*, 10, 02 2021.

[26] C. Lei, B. Hu, D. Wang, S. Zhang, and Z. Chen. A preliminary study on data augmentation of deep learning for image classification. In *Proceedings of the 11th Asia-Pacific Symposium on Internetware*, 2019.

[27] J. Li, P. Wang, P. Xiong, T. Cai, Z. Yan, L. Yang, J. Liu, H. Fan, and S. Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] Z. Li, G. Wang, and X. Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *2014 European Conference on Computer Vision (ECCV)*, Cham, 2014. Springer International Publishing.

[30] X. Liu, R. Zhang, C. Zhang, B. Fu, J. Tang, X. Liang, J. Tang, X. Cheng, Y. Zhang, G. Wang, and X. Ji. Gdrnpp. https://github.com/shanice-l/gdrnpp_bop2022, 2022.

[31] P. Marion, P. Florence, L. Manuelli, and R. Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 05 2018.

[32] Merwansky. What is 6D Object Pose Estimation in Computer Vision? — just-merwan.medium.com. https://just-merwan.medium.com/what-is-6d-object-pose-estimation-in-computer-vision-21e8acf9e3e2, 2022. [Accessed 25-12-2023].

[33] N. Morrical, J. Tremblay, Y. Lin, S. Tyree, S. Birchfield, V. Pascucci, and I. Wald. Nvisii: A scriptable tool for photorealistic image generation. *ArXiv*, abs/2105.13962, 2021.

[34] C. G. Northcutt, A. Athalye, and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[35] Nvidia. Isaac Sim – developer.nvidia.com. https://developer.nvidia.com/isaac-sim. [Accessed 01-01-2024].

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NEURIPS)*, volume 32. Curran Associates, Inc., 2019.

[37] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[38] P. Quentin, D. Knoll, and D. Goehring. Industrial application of 6d pose estimation for robotic manipulation in automotive internal logistics. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023.

[39] R. T. Ryan-Thomas Shaw. Figure out what makes steady photos and videos possible. *Android Authority*, November 2023.

[40] M. Sundermeyer, T. Hodaň, Y. Labbé, G. Wang, E. Brachmann, B. Drost, C. Rother, and J. Matas. BOP challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023.

[41] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[42] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[43] Unity Technologies. Unity Perception package. https://github.com/Unity-Technologies/com.unity.perception, 2020.

[44] S. Z. Valtchev and J. Wu. Domain randomization for neural network classification. *J. Big Data*, 8(1), July 2021.

[45] B. Vanherle, S. Moonen, F. Van Reeth, and N. Michiels. Analysis of training object detection models with synthetic data. In *33rd British Machine Vision Conference 2022 (BMVC)*. BMVA Press, 2022. [Accessed 05-01-2024].

[46] G. Wang, F. Manhardt, F. Tombari, and X. Ji. GDR-Net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[47] M. Weisenböhler, B. Hein, and C. Wurll. On scene engineering and domain randomization: Synthetic data for industrial item picking. In I. Petrovic, E. Menegatti, and I. Marković, editors, *Intelligent Autonomous Systems 17*, Cham, 2023. Springer Nature Switzerland.

[48] T. Whelan, S. Leutenegger, B. Glocker, R. F. Salas-Moreno, and A. J. Davison. Elasticfusion: Dense slam without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2015.

[49] R. Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[50] G. Xu, X. Wang, X. Ding, and X. Yang. Iterative geometry encoding volume for stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.