Evaluating 6D pose estimation accuracy with synthetic data

A comparative analysis of RGB based and RGB-Depth-based images in a chess piece picking task with a robotic arm

MSc Thesis

Max Waterhout

TU Delft

Evaluating 6D pose estimation accuracy with synthetic data

A comparative analysis of RGB based and RGB-Depth-based images in a chess piece picking task with a robotic arm

by

Max Waterhout

Student Name	Student Number
Max Waterhout	5384907

Supervisor:	Yke Bauke Eisma
Daily supervisor:	Renchi Zhang
Project Duration:	May, 2023 - January, 2024
Faculty:	Faculty of Mechanical, Maritime and Materials Engineering, Delft

TU Delft

Preface

This paper is a component of my master's thesis, representing the conclusion of my studies in Robotics at the Faculty of Mechanical, Maritime, and Materials Engineering of Delft University of Technology. The research for my master's thesis was carried out at TU Delft within the Cognitive Robotics department under the guidance of my supervisor, Yke Bauke Eisma.

The initial objective of the research was to make the existing chess robot more robust in picking and ultimately develop a pipeline enabling it to promote pieces but also set up the chessboard. Following this work, it becomes evident that this goal stands as a viable and reasonable goal for future research.

In conclusion, I would like to extend my gratitude to my supervisor Yke Bauke Eisma for insightful discussions and providing access to the GPU clusters. Special thanks to my daily supervisor Renchi Zhang, who consistently supported me by engaging in discussions and addressing any questions I had across various topics. Furthermore, I would like to thank Eline, my family, fellow students and friends for their encouragement and support throughout my time as a student at TU Delft.

I acknowledge the use of ChatGPT in refining the text in this work, but only that it was used for improving clarity and coherence, not for inventing content, not being used as a source or using sentences or words I would not use.

Max Waterhout
Delft, January 2024

Contents

Abstract

This study investigates the influence of synthetic data on the accuracy of 6D pose estimation in RGB images compared to RGB-Depth image-based methods. Additionally, it aims to examine how this performance varies across different types of small chess pieces during a picking task with a robotic arm. The methodology involves 3D scanning the chess pieces and generating a dataset of 61,910 synthetic images with diverse domain randomization. Using this synthetic dataset, six unique 6D pose estimation models were trained for each type of chess piece. The models were validated using a real-world 6D pose estimation dataset, and the obtained results were compared with those from the estimations on a synthetic dataset. It was observed that synthetic data can be used for bridging the visual simulation to reality gap. However there is superior performance on synthetic data compared to real-world data. This implies that results obtained in a synthetic environment cannot be directly projected to real-world scenarios. Also a noticeable decrease in accuracy in both object detection and 6D pose estimation was observed with respect to camera distance for real-world images, primarily linked to the reduced size of chess pieces in the images. Notably, each chess piece exhibited improvement after depth refinement, with accuracy closely linked to the performance of the depth camera. A picking experiment was also conducted, revealing that models relying solely on RGB data achieved a positive grasping rate of approximately 52%, while RGB-Depth-based methods reached around 70%. The findings underscore the potential for successfully picking chess pieces with and without depth refinement, emphasizing the feasibility of bridging the visual simulation to reality gap. Additionally, the study suggests several avenues for future research, including comparing synthetic data with real-world data, further exploring the training process, and introducing domain randomization in the picking experiment, such as background changes or different distractors. Furthermore, it suggests investigating diverse approaches to improve depth accuracy.

1 Introduction

Chess provides an interesting research domain, offering a simple yet multifaceted game that allows exploration across various factors. Research in this field delves into physiological aspects, as demonstrated in studies such as [1], where stress is examined through different chess problems. Furthermore, chess offers a platform for technical research, as seen in studies like [2], which delves into gaze and memory theory within a chess setting. In the domain of robotics research, particularly in human-robot interaction, the study conducted in [3] explores the application of nonverbal emotion expression to evaluate the interactions between humans and machines in a chess match. A fundamental aspect of chess playing also involves picking the game pieces. Some research papers have tackled the challenge of chess piece manipulation by employing methods such as detecting the chessboard with corner detection or identifying ArUco markers on a custom chessboard [4, 5, 6]. These methods typically concentrate on distinguishing the elements in the surrounding environment and assuming the initial positions of the chess pieces, rather than directly detecting the individual pieces themselves. While these methods are effective in scenarios where the chess pieces are already on the board, it's important to note that a crucial part of chess involves uncertain piece placement, like piece promotion. In this situation a chess piece is located outside the board. The task of picking is a significant challenge in the field of robotic manipulation and represents a key area of research within the broader robotics community. Many challenges, such as the Amazon Picking Challenge [7] and the DHL Robotics challenge [8] have been built around this task. The aim of these challenges is to enable robots to assist human workers with tedious tasks, leading to increased efficiency, improved throughput, and reduced costs.

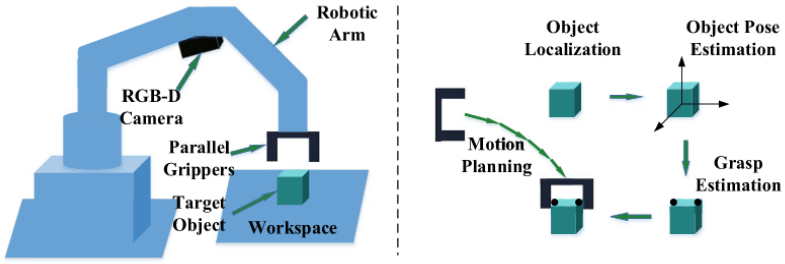


Figure 1.1: The picking problem pipeline

1.1. Problem statement

In many picking scenarios, it is essential to accurately estimate the transform of objects in the world. For example, consider a scenario where a robotic arm retrieves items from a moving assembly line, with objects situated in varying positions [9]. Similarly, in high-precision assembly tasks like the Siemens Innovation Challenge, the assembly of gears within industrial settings with millimeter-accurate placement [10]. The process of estimation the transform in the real world is known as six degree-of-freedom (6D) pose estimation. A process that transforms the object from its object coordinate system into the cameras coordinate system [11, 12]. The complete picking pipeline, which includes this critical step, is depicted in Figure 1.1. The 6D pose is a 4x4 matrix that is composed of a 3D rotation and a 3D translation of a frame with respect to another frame, as shown in equation (1.1). In equation (1.1), the 6D pose is represented as the transformation matrix ${}^{A}_{B}T$, which describes the position and orientation of frame B relative to frame A. In real life we do not know the exact pose, but we can estimate it, done with RGB or RGB-Depth based images. The position and rotation of a known object model can be obtained using various algorithms. Once the 6D pose is obtained and the position and rotation of

the camera is captured with respect to the gripper, it becomes possible to determine how to move the gripper towards the object for picking.

$$ {}^{A}_{B}T = \begin{bmatrix} {}^{A}_{B}R & {}^{B}P \\ 0 & 1 \end{bmatrix} \tag{1.1} $$

Figure 1.2: An example 6D pose from frame B to frame A

Modern object detection and 6D pose estimation algorithms mostly rely on neural networks. However, the efficiency of these algorithms is tightly linked to the availability and quality of training data. Achieving superior results often demands a substantial amount of data. In the early stages of object detection, benchmark datasets such as ImageNet [13] and COCO [14] played crucial roles, featuring extensive datasets with 1.2 million and 325,000 images, respectively. Recommendations for object detection emphasize the crucial role of data quality in ensuring good performance, including aspects such as accurate labeling [15]. In essence, the process of data collection poses a great challenge, demanding both time and effort. Creating a 6D pose estimation dataset is extra challenging as obtaining real 6D pose data for objects requires accurate annotation of object poses in three-dimensional space. The process involves capturing accurate information about the translation and rotation of objects as observed by the camera.

1.2. Relevant work

Traditionally, 6D pose estimation methods based on images have relied on matching RGB feature points between 3D models and images [16, 17], dating back to 1999 with [18]. These methods involve extracting features from images, which can be RGB colors or more recent approaches that employ neural networks. The goal is to match keypoints of objects in the image with keypoints from the object stored in the database. By establishing these keypoint matches, 6D pose estimation can be performed. However, one major limitation of this method is that objects require rich texture for detection of feature points. With the introduction of affordable depth cameras, such as the Microsoft Kinect and Intel RealSense, several new methods have emerged that that add depth information to match features of less texture-rich objects [19, 20, 21].

The review [22] categorized the various pose estimation methods from RGB based to RGB-Depth based methods. As outlined by the Benchmark for 6D Object Pose Estimation (BOP Challenge) [23], an open challenge which aims to capture the state-of-the-art 6D pose estimation models, geometric pose estimation techniques utilizing Pair Point Features (PPF) extracted from keypoints emerged as the leading performers from 2010 to 2019 [24, 25]. However, starting in 2020, Deep Neural Network (DNN) approaches that only utilized RGB inputs, trained on extensive datasets, and did not rely on keypoints, emerged as contenders [26]. By 2022, DNN-based methods had surpassed PPF-based methods in both speed and accuracy [27, 28]. In 2022 the best methods based on RGB even surpassed the best RGB-Depth methods from 2020. Recent works [29, 30] still consider depth information crucial for the task of 6D pose estimation, despite RGB methods becoming more competitive in recent years. This is because the 2D projection of an object in an image may lack vital geometric details that are preserved in depth information. In essence, the 3D representation contains richer information compared to an 2D image. Also state-of-the-art results are still achieved with depth refinement looking at results from the latest BOP challenge from 2022.

As already mentioned in Section 1.1, data collection for 6D pose estimation is challenging. There are various open-source datasets available for testing 6D pose estimation algorithms. In 2022, the BOP challenge incorporates twelve distinct datasets. The two most widely utilized datasets, namely YCV-Video and Linemod Dataset, feature household objects such as a bench vise or soap dispenser. These objects generally range from 150mm to 270mm in diameter. However, acquiring datasets for other objects can prove to be challenging. In this context, the importance of synthetic data has grown, allowing for the training of models using computationally generated data. This approach is particularly beneficial in scenarios where obtaining accurate ground truths in simulation is more feasible than in real-world. In delving into the realm of synthetic data, it becomes crucial to examine how simulation can effectively narrow the gap between simulated and real-world scenarios, known as the sim2real gap. Although simulated images may not be indistinguishable from real ones upon careful inspection, the

objective is to expand the simulation domain extensively until it encapsulates the reality domain. The overarching goal is to create a simulation environment so comprehensive that reality data points can be interpolated between two synthetic data points.
Early instances of utilizing synthetic data can be found in papers such as [31], where OpenGL, a graphics program for accelerated graphics, is employed to generate synthetic data for object detection. Additionally, in [32], images for 6D pose estimation are computed by integrating a 3D object into a randomly selected real photograph, also utilizing OpenGL. However, a significant drawback of employing OpenGL for synthetic data creation is that the resultant images lack realism, with inconsistent shading that deviates from real-world scenarios, as these images do not stick to the rules of the physical world. That is why [33] developed Blenderproc, which is a modular procedural pipeline in Blender [34], which helps in generating real looking images for training different networks. Blender, an open-source 3D computer graphics software, serves a wide range of applications, for example 3D modeling, animation, rendering, and shading. It utilizes a physically accurate light tracer for rendering to achieve realistic shading and collaborates with different frameworks such as OptiX, HIP and Metal (NVIDIA, AMD and Apple) to significantly accelerate computing times [35]. The paper [36] has demonstrated sim-to-real results for instance segmentation with BlenderProc, which exhibit favorable comparisons against the conventional render-and-paste method utilizing OpenGL renderings. The latest BOP challenge also provide 50,000 training images per dataset and states that sim2real has shrank further since earlier years [28]. The paper [37] illustrated that domain randomization, such as introducing distractors into the scene or altering lighting conditions, proves to be an effective method for narrowing the visual simulation-to-reality gap while the papers [38, 39] also train on simulated images for picking tasks.

1.3. Research question

Despite ongoing research in 6D pose estimation and advancements in synthetic data, it remains unproven whether relying only on synthetic data, particularly in the RGB format, is accurate enough for real-world tasks such as picking the substantially smaller chess pieces. Consequently, a research gap exists, questioning the feasibility of this approach and investigating whether depth information is still necessary for refinement.
This study aims to investigate the applicability of synthetic data to chess pieces and assess whether it is enough to bridge the visual simulation-to-reality gap. The objective is to enable trained models for deployment in real-world picking tasks involving chess pieces while considering the potential requirement of incorporating depth information. The overarching research question is: **How does the utilization of synthetic data impact the accuracy of 6D pose estimation for RGB images compared to RGB-Depth images? Additionally, how does the performance vary across different types of chess pieces during a picking task?**

1.4. Main contributions

The main contributions in this work are as follows:

1. Generating 3D-scan models of chess pieces and constructing a synthetic dataset using them
2. Generating a real-world 6D pose estimation dataset using chess pieces
3. Conducting a comparative study of 6D pose estimation models with RGB and RGB-Depth based methods focusing on substantially smaller objects than current datasets
4. Comparing 6D pose estimation results on a synthetic and real world dataset
5. Evaluating a 6D pose model in the context of a real-world robotic chess picking task

2

Methods

This chapter provides an overview of the methods employed in the creation of a synthetic dataset. It covers the creation of 3D models and the selection of dataset attributes including domain randomization techniques. The chapter proceeds to explain the metrics used for evaluating the performance of various models utilized in this work. Furthermore, it delves into the explanation of the selected 6D pose estimation model. The chapter also discusses the advantages and disadvantages of various methods for generating real-world 6D pose estimation data and explaining the chosen method in this work. Finally, the methods for depth refinement are explained.

2.1. The synthetic dataset

2.1.1. The chess pieces

In this work only the white pieces are going to be analyzed. The objects are depicted in Figure 2.1a, also showcasing the range of heights that vary from 37mm for the smallest to 80mm for the largest. To generate CAD models of the objects, a 3D scanner is employed. Various methods exist for capturing 3D data, each with differences in accuracy and scanning distances. For instance, the Artec Ray II is designed for extended scanning ranges exceeding 30 meters. However, for more detailed tasks, handheld scanners such as the Artec Eva and Einscan H are preferred due to their greater accuracy, which can reach up to 0.1mm.
In this particular application, the Artec Eva is utilized. The Artec Eva 3D scanner uses structured light projection to capture the surface geometry of an object. An exemplary CAD model of the horse, captured by the handscanner, is showcased in Figure 2.1b, where it is visualized in the 3D mesh processing software Meshlab.

2.1.2. The synthetic dataset

For creating the synthetic dataset, the Blenderproc toolkit in blender is used, which streamlines a pipeline for photorealistic rendering. This toolkit helps creating ground truth poses and bounding boxes in the images. The images are generated using the 3D scanned chess pieces, but they also incorporate six different distractor objects selected from a collection of 36 various household products that are used in the YCB-Video and Linemod dataset. This inclusion of distractor objects highlights the variations in object size compared to the chess pieces. These distractor objects are placed in a manner that ensures they don't collide with each other, all within a specially crafted "box" featuring an unique texture. This

(a) The corresponding Chess Pieces. From left to right: king, queen, bishop, horse, rook, and pawn.

(b) Example display of CAD model of the horse

Figure 2.1: Chess Pieces and Horse Example

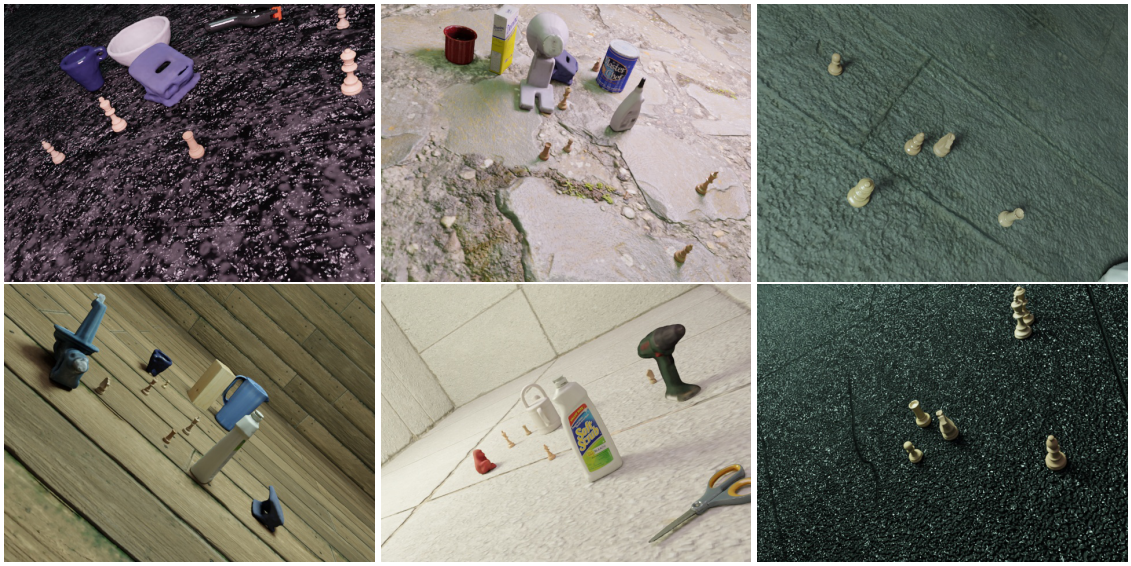


Figure 2.2: Synthetic example images

texture is picked from a selection of 1627 textures, offering a wide range of backgrounds, including options like sand, asphalt, leather, and brick. Exemplary images showcasing example scenes can be found at Figure 2.2. Each time a scene is created, it results in the generation of 10 images capturing that scene. The camera's viewpoint is determined by sampling from a range between two spheres, spanning from 200mm to 900mm away from the scene's center. The camera's elevation is also varied, with a minimum angle of 5 degrees and a maximum of 85 degrees. Furthermore, the light source is diversified for each scene by altering its placement within a spherical shell around the origin. Additionally, both the color and strength of the light source are randomized. And finally, the texture of the chess pieces is sampled from a pool of five distinct light wood textures, each assigned randomly for every new scene.
A total of 61,910 images are generated, and this entire process takes approximately 32 hours to complete when running on a Quadro T1000 GPU. It's worth noting that the number of generated images aligns with the benchmarks set by the BOP challenge papers, which utilize 50,000 training images in their experiments. The histogram of all the camera distances to the chess pieces can be found in Figure 2.3 where the distribution is around 900mm.

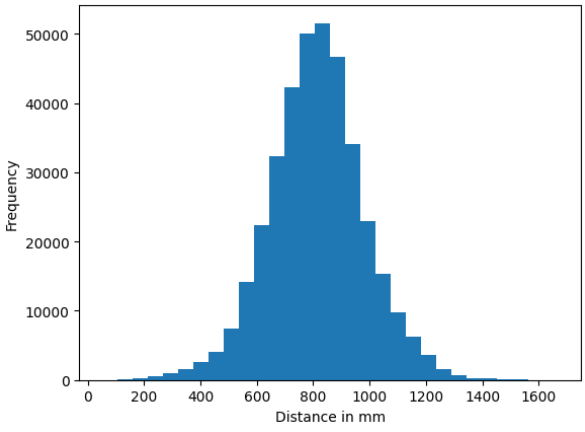


Figure 2.3: Histogram of camera distance to chess pieces

2.2. Evaluation metrics

2.2.1. Bounding box metrics

Two key metrics that are often used for bounding boxes are known as Mean Average Precision (mAP) at an IoU threshold of 0.50 (MaP@0.50) and Mean Average Precision across the range of Intersection over Union (IoU) thresholds from 0.50 to 0.95 (MaP@0.50-0.95) [40]. Mean Average Precision is a widely employed evaluation metric in bounding box evaluation. It calculates the Average Precision for each class and at a specific IoU threshold. The IoU concept, visually explained in Figure 2.4, involves defining true positives and false positives. A true positive occurs when the IoU between the predicted bounding box and the ground truth exceeds the set IoU threshold, while a false positive arises when the IoU falls below that threshold.

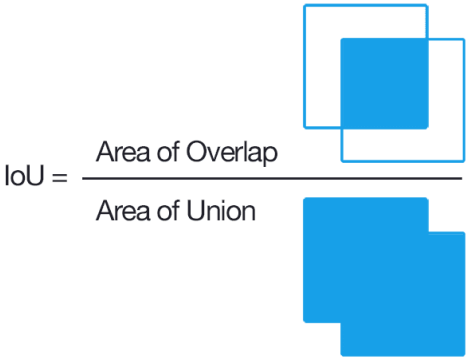


Figure 2.4: Intersection of Union explained

Precision is defined as the ratio of true positives to the sum of true positives and false positives, expressed by the formula $\frac{TP}{TP+FP}$. The mean Average Precision in the context of object detection involves averaging precision values across a range of Intersection over Union thresholds for each class. For mAP@0.50, only the IoU threshold of 0.50 is used. For mAP@0.50-0.95, the thresholds range from 0.50 to 0.95, with a step size of 0.05. The precision is determined for each class at the specified IoU thresholds. The mAP@0.50-0.95 is then obtained by averaging these class precision values, providing a comprehensive evaluation metric for bounding box performance across different IoU thresholds.

2.2.2. 6D pose estimation metrics

For non-symmetric objects, the commonly used evaluation metric is the Average Distance of Model Points (ADD), introduced in [19], which measures the mean pairwise distance between transformed points from the estimated and ground truth poses, see equation (2.1). In this notation, the symbols **R** and **T** represent the ground truth rotation and translation, respectively. Similarly, the symbols $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{T}}$ denote the estimated rotation and translation, respectively. $\mathcal{M}$ denotes the set of 3D points and m is the number of points.

$$ADD = \frac{1}{m} \sum_{x \in \mathcal{M}} ||(\boldsymbol{R}x + \boldsymbol{T}) - (\tilde{\boldsymbol{R}}x + \tilde{\boldsymbol{T}})|| \tag{2.1}$$

When dealing with symmetric objects, the evaluation metric used is the Average Closest Point Distance (ADD-S), introduced in [41]. This metric selects the average distance using the closest point distance because matching points between symmetric objects can be ambiguous, see equation (2.2).

$$ADD - S = \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} ||(\boldsymbol{R}x_1 + \boldsymbol{T}) - (\tilde{\boldsymbol{R}}x_2 + \tilde{\boldsymbol{T}})|| \tag{2.2}$$

In the same vein, the ADD(-S) is mainly used to evaluate 3D object tracking, as it considers both symmetric and non-symmetric objects, see equation (2.3). For non-symmetric objects, it calculates the ADD distance, while for symmetric objects, it calculates the ADD-S distance.

$$ADD(-S) = \begin{cases} ADD & \textit{if asymmetric,} \\ ADD - S & \textit{if symmetric.} \end{cases} \tag{2.3}$$

For pose estimation, accuracy is often determined by comparing the ADD(-S) with a pre-defined threshold. In [41, 26], the evaluation is considered correct if the ADD(-S) falls below the threshold of 10% of the object's diameter. However, [29, 30, 42] also vary the threshold values and plot an accuracy-threshold curve to visualize the performance of pose estimation over different thresholds relative to the objects diameter. Varying thresholds allows for assessing pose estimation methods across various difficulty levels, with lower thresholds used to evaluate more accurate estimations and higher thresholds for more tolerant evaluation. This approach provides a more comprehensive performance evaluation of pose estimation methods. The Area Under the Curve (AUC) is then computed as a metric for pose estimation accuracy. With this, a single metric can be obtained to quantify the performance of a model.

2.3. The 6D pose estimation model

The model that is chosen to estimate the accuracy of 6D pose estimation is EfficientPose [43]. EfficientPose is a model that proposes a scalable model based on the popular EfficientDet [44]. The network utilizes a convolutional neural network (CNN) architecture that takes RGB images as input and directly predicts the 6D poses of the objects in the scene. This is achieved by incorporating two additional subnetworks designed to predict the translation and rotation of objects, similar to the subnetworks handling classification and bounding box regression. The employed loss in this model combines classification, bounding box regression, and transformation losses, as outlined in equation (2.4). The loss serves as a metric to evaluate how closely the model's predictions align with the desired target values. It quantifies the error between the model's predicted outcomes and the actual targets. During training, the objective is to minimize this loss, striving for a closer match between the model's predictions and the ground truth. Within this function, the classification loss, often referred to as the focal loss [45], presents a modificated version of the cross-entropy loss [46]. This loss function is designed to tackle the issue of class imbalance in the dataset. Focal loss assigns higher weight to misclassified examples, which helps the model focus on hard-to-classify objects. In essence, it emphasizes correcting the most challenging cases during training. Additionally, the bounding box regression loss, also known as the Huber loss [47, 48] or smooth L1 loss, offers a loss function that is less sensitive to outliers. In scenarios where regression targets are unbounded for bounding boxes, other loss functions might need careful tuning of learning rates to prevent exploding gradients. The smooth L1 loss avoids this issue. The employed transformation loss is the ShapeMatch loss [41], which focuses on the 3D shape of the object. It calculates this loss for symmetric objects, as illustrated in equation (2.1), and asymmetric objects, as shown in equation (2.2). This loss plays a crucial role in accurately calculating the 6D pose error and training the model effectively by minimizing it. For training EfficientPose uses the Adaptive Moment Estimation (ADAM) optimizer [49]. ADAM combines the benefits of two other optimization methods, RMSprop and Momentum. ADAM maintains a moving average of the gradient and the square of the gradient's past values. This allows it to adapt its learning rates for each parameter individually, resulting in faster convergence and efficient training.

$$L = \lambda_{\text{class}} \cdot L_{\text{class}} + \lambda_{\text{bbox}} \cdot L_{\text{bbox}} + \lambda_{\text{trans}} \cdot L_{\text{trans}} \tag{2.4}$$

The model is designed to be efficient making it suitable for training on large datasets but also suitable for real time applications. The size of the model can be scaled with one hyperparameter ϕ to fit all different computational resources. This model is chosen based of a couple factors. Firstly, it demonstrated State of the Art performance in a prior study when tested on the Linemod dataset. Another advantage it holds over alternative models is its end-to-end training capability. This means that unlike other models that need separate networks for classification and/or bounding box detection, this model incorporates these functionalities during the training process for 6D pose estimation. The best results have been achieved with a separate model for each object, and that's also the approach adopted in this work.

2.4. Real-world 6D Pose Estimation Datasets

A real-world dataset is essential to conduct a comparative analysis between synthetic and real-world data, providing insights into the sim2real gap. To generate a real-world dataset for evaluating the model's performance, it is crucial to obtain the ground truth poses of the chess pieces. The two most famous 6D pose estimation datasets both use a different method to capture the data. Linemod [19] creates its dataset where they fix the objects on a flat board with attached markers, serving correspond-

ing ground truth poses. In the case of the YCB-Video dataset [41], a video-based approach is utilized. Initially, only the first frame is manually annotated. Subsequently, with the aid of Signed Distance Function (SDF) representations for each object, the pose of each object in the initial depth frame is refined. As the video progresses, object configurations are continuously tracked. Ultimately, a global optimization process combines the camera trajectory and the relative object poses to yield accurate results.
Both of these methods come with their own advantages and disadvantages. The ArUco board that is used in the Linemod dataset is very useful due to their fast detection and their versatility. However, one of the problems of ArUco markers is that the accuracy of their corner positions is not too high, even after refinement. The YCB-Video approach offers greater flexibility in different objects scenes, whereas the accuracy can be prone to errors in the initial pose measurement or depth map.
In this dataset, the Linemod approach is adopted, but instead of using an ArUco board, a ChArUco board is employed [50]. A ChArUco board combines the advantageous features of both a chessboard pattern and an ArUco board. It allows for more precise refinement of the corners of chessboard patterns since each corner is enclosed by two black squares. However, it's worth noting that identifying a chessboard pattern is less versatile compared to locating an ArUco board, as it needs full visibility, and occlusions are not tolerated. Therefore, the combination of the two methods proves to be an excellent approach. By utilizing the ChArUco board, a predefined A3 board is generated to precisely dictate the placement of the chess pieces. Refer Section 3.2 for a detailed explanation of the generation of the real-world dataset used in this work.

2.5. Depth refinement

For the depth refinement the Iterative Closest Point (ICP) algorithm [51] is chosen. This classic method is still the most used method for depth refinement in the BOP challenge. ICP is a method for point cloud registration, which is a process that tries to allign two (or more) point clouds together. The idea is to refine the initial guess of the transformation that maps one point cloud to another by minimizing the distance between all corresponding points in the two clouds. An example can be seen in Figure 2.5 where the blue point cloud would be the (partial) depth map of the scene and the green point cloud the initial 6D pose estimation.
Incorporating ICP algorithms often involves the utilization of plane segmentation techniques. This is for instance useful, in the detection of ground planes [52]. The purpose of such segmentation is to mitigate overfitting to the ground, a situation that can occur in real-world scenarios where the point cloud only partially covers objects. The choice of Open3D's plane segmentation method is motivated by its ease of implementation, complementing the existing ICP algorithm, and the availability of great documentation for integration. This method segments geometric primitives from point clouds using the RANdom Sampling and Consensus (RANSAC) technique. RANSAC, as an iterative modeling method, effectively handles data with outliers. It identifies and removes outliers, estimates the model using outlier-free data, and extracts shapes based on minimal sets, a minimal number of data samples required to uniquely define a model (e.g., three points for a plane, four for a sphere). The process involves identifying points within a specified threshold of the plane represented by $ax + by + cz + d = 0$, referred to as inliers. The plane with the maximum number of inliers is considered the most accurate estimation for the ground.

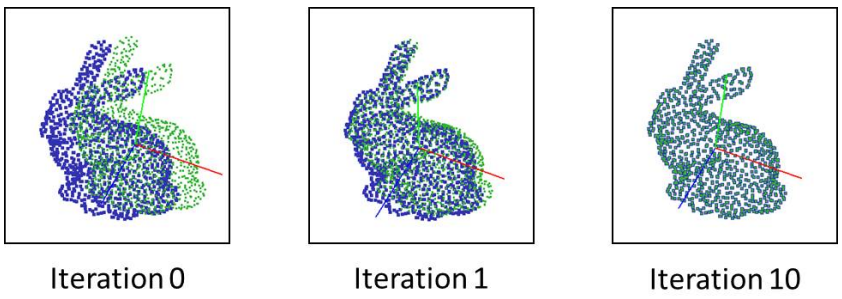


Figure 2.5: The Iterative Closest Point algorithm visualized [53]

3

Experiments

This chapter begins by detailing the training process for the 6D pose estimation model, providing insights into the made decisions. Subsequently, it delves into the creation of a real-world dataset designed specifically for validating the 6D pose models. Additionally, the chapter goes in detail on the depth refinement pipeline used in the experiments. Towards the end, the chapter offers an overview of the ROS-based picking experiment, exploring the picking capability. This section provides detailed explanations and visualizations of the ROS node network, the training process of a lightweight object detection model, and the execution of a hand-eye calibration process.

3.1. Training the 6D pose estimation model

3.1.1. Preparing the model

For each model, images are only chosen in which the object to be trained occupies a minimum of 30% of the frame. Subsequently, the synthetic dataset is randomly partitioned into training and validation sets, with a distribution of 97% for training and 3% for validation. Also a seperate test set is generated and held back for the validation of the final models. This set can be compared against the real-world dataset. The datasets size per object can be seen in Table 3.1. Most of the images are processed in each epoch, amounting to 3,000 steps x 16 images per batch (48,000 images). The hyperparameter ϕ is set to 0 to avoid computational resource constraints when running all six models in parallel at runtime. The initial learning rate is set at 0.0001 and is halved if the average point distance does not decrease during the last 35 evaluations. This adaptive learning rate strategy is implemented to address situations where the model's performance stagnates during training to help converging to a global minima and help with stability. Additionally, image augmentation is disabled due to the dataset's already substantial size and potential challenges associated with transitioning from augmented synthetic data to real-world images. Finally, the transformation loss, initially set at 0.02 as per the original paper's findings, has been adjusted through experimentation. In this particular experiment, a value of 1 has been determined to be more effective, following a process of trial and error. This modification resulted in a notable decrease in the transformation loss and an enhancement in the ADD(-S) metric performance over the validation set. The hypotheses of this outcome can possibly be attributed to the small scale of the objects being detected. A higher loss value helps the optimization process and enabling the model to converge to a more favorable minimum. Without this adjustment, the classification loss and bounding box regression loss could dominate the training process and hinder convergence for the transformation*.

Name	Train size	Validation size	Test size
Rook	51597	1596	1555
Queen	52983	1639	1547
Pawn	50808	1572	1400
King	49838	1542	1146
Horse	47547	1471	1112
Bishop	45587	1410	1036

Table 3.1: Size of the image sets per object

*During the writing process, additional research was undertaken, and it was found that these hypotheses may be incorrect. However, despite this discovery, the conclusions drawn in this work remain unchanged. For further details and additional results, please refer to Appendix A.5.

3.1.2. Training

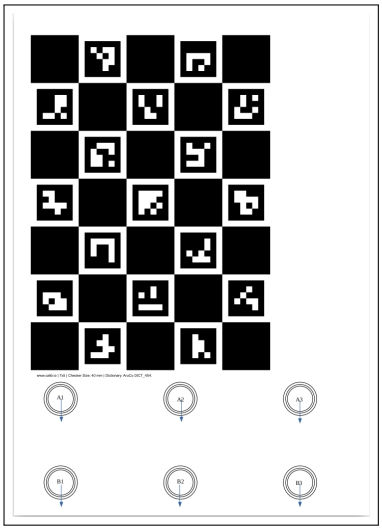
The model training is carried out using an Nvidia RTX 3090ti GPU in conjunction with an AMD Ryzen Threadripper PRO 83955WX 16-Cores CPU. All models undergo a minimum of 400 training epochs, equivalent to approximately 5 days of training. The training process has been saved through different logs with tensorboard. Evaluation of the validation set occurs following every 5 epochs, and if the ADD(-S) metric exhibits improvement on the validation set, the model is saved. The graphs of all the losses and the validation set metrics can be seen in Appendix A from Figures A.1 to A.6. In these graphs the epoch where the best model is saved is shown with a red dot. What can be seen from all these graphs is that while the model trains, the loss and mean distance decreases and the ADD(-S) increases. This means that the model is actually learning. What can be seen for some models is that at a later stage of training the model can become unstable where the loss diverges. This can be perfectly seen in the loss figure of Figure A.5.

3.1.3. The synthetic test set

This test set is made to validate the 6D pose estimation results of the trained models on unseen synthetic data. This test set is generated in addition to the training and validation sets. The only difference lies in the distribution of the objects distance to the camera, which centers around 500mm, as this is also the distribution for the real-world dataset, as can be read in section 3.2. Furthermore, only images where the object is visible in at least 70% of the image are used. The 6D pose estimation results can be found in chapter 4.

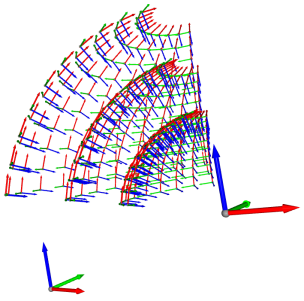
3.2. Real-world dataset creation

In the setup of dataset creation, the FRANKA RESEARCH 3 robotic arm is employed, and it operates with libfranka and franka_ros, both version 0.10. The choice of a robotic arm is driven by the capability to systematically arrange preprogrammed camera positions and maintain steady image quality. The imaging is facilitated by the ZED2 camera, which operates on ZED SDK 4.0. The captured images have dimensions of 1280x720 pixels, and the depth information is acquired using the neural depth mode. The dataset creation setting is shown in Figure 3.1b. The board is deliberately positioned atop a box at the end of the table, a decision made to prevent the robot arm's workspace from being constrained by joint limits. This arrangement guarantees that a wider range of poses can be accessed with few limitations. The images are acquired while the box remains stationary, and the robotic arm undergoes a series of movements to various locations in the environment, capturing images at each point. At each position, a defined pose is set for the end effector.



(a) The ChArUco board with predefined placements

(b) The dataset creation scene

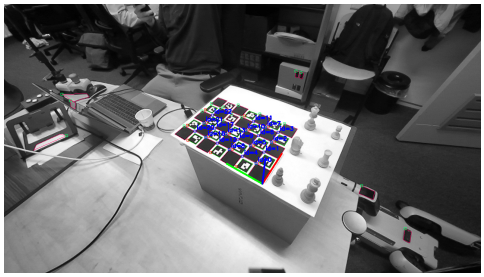


(c) The example hemisphere of the dataset creation

Figure 3.1: The dataset creation

These locations for the robotic end effector are determined by a hemisphere configuration with the theta angle ranging from 15 to 75 degrees and the phi angle spanning from 30 to 150 degrees. Each hemisphere contains 100 locations. The hemisphere is defined with radii of 300, 450, and 600 millimeters, with the top of the box serving as the central reference point. Example locations from the hemisphere can be seen at Figure 3.1c, showcasing both the central point and various end effector poses. An image will be captured when the ChArUco detection method can reliably and accurately locate the 6D pose of the board, as demonstrated in Figure 3.2b. However, it's important to note that certain poses may remain inaccessible due to restrictions imposed by the robot's joint limitations, and in such cases, images will also not be taken. For every reachable arm position, the camera takes a RGB image and saves the whole point cloud that will be used for depth refinement. Within each iteration of 300 poses, approximately 50% of the images are preserved, taking into account factors such as poses that cannot be reached or unsuccessful ChArUco board detection. This process is iterated six times, during which the entire board is rotated by 180 degrees, and the chess pieces are rearranged into different columns of the ChAruco board. For example, the piece on A1 goes to A2, A2 is moved to A3, and A3 is moved to A1. Furthermore, a standard deviation of 50mm is introduced for each pose in the [x, y, z] coordinates. This variation is deliberately introduced in each iteration to introduce an element of randomness and diversity in the dataset. The systematic introduction of randomness, the rotation of the box, and the reshuffling of chess pieces are incorporated to ensure comprehensive coverage of the entire set of chess pieces in the images that are captured. In total 944 images are taken in the dataset. Additionally, objects that are not visible in the images, either due to falling off the frame or experiencing at least 50% occlusion, are categorized as negative instances.

(a) The RGB image of the board



(b) The ChArUco pose detection

Figure 3.2: An example image of the dataset

3.3. The depth refinement pipeline

For 6D pose estimation with depth refinement, the pointcloud is captured by the neural depth mode of the ZED2 camera. An important note is that only the visible portions of the scene is captured. To limit the ICP algorithm to points close to the object, only points within a 75mm radius of the initial estimation are taken into account. Additionally, the ground plane is excluded from the refinement. The plane is segmented with a distance threshold of 2.5mm, RANSAC is 3 and with 1000 refinement iterations. As explained in Section 2.5 the distance threshold defines the maximum distance a point can have to an estimated plane to be considered an inlier. The threshold is selected through a process of trial and error. A smaller threshold imposes strict criteria for points to be classified as inliers to the plane, potentially leaving some ground plane points unfiltered. In contrast, larger thresholds consider points further from the plane as inliers, which may result in the removal of points belonging to the object. The number of iterations defines how often a random plane is sampled and verified.
An example pipeline is depicted in Figure 3.3, showcasing the depth refinement process for the horse object in the provided image, as seen in Figure 3.3a. Figure 3.3b displays the point cloud of the image in red, with the initial estimation from an RGB image depicted in cyan. The estimated ground plane is represented in yellow in Figure 3.3c. Subsequently, the ICP algorithm is applied to the relevant depth points, with a maximum of 10 iterations. The resulting blue points denote the final pose estimation, while the ground truth is presented in green in Figure 3.3d.

(a) The provided RGB image **(b)** The point cloud of the image **(c)** Only the relevant points will be considered **(d)** The final estimations in cyan and blue, the ground truth in green

Figure 3.3: The depth refinement pipeline

3.4. The picking experiment

As the goal of the picking process goes beyond demonstrating the effectiveness of the pose estimation model, an experiment involving real picking data is conducted. Chess pieces are positioned on top of a box to create a more extensive range of joint movement, with a white paper underneath to enhance the contrast between the background and the chess pieces. For consistent evaluation, each attempt to pick up each chess piece is repeated 25 times, using a gripper with a width of 20mm, both with and without depth refinement. The chess pieces are positioned and rotated randomly on the box. The camera will determine the positions of the chess pieces from two predetermined joint states at the top corner of the box. The evaluation is solely focused on categorizing the picking outcomes as either ”POSITVE” or ”NEGATIVE”. Prior to identifying which grasp to request, a lightweight object detection model is employed to detect the presence of an object within the image. This choice is made to avoid the simultaneous execution of the six EfficientPose models in parallel, as it would not scale efficiently with computational costs. The lightweight classification model chosen for this purpose is YOLOv5 [54]. To accurately estimate the camera's position within the scene, MoveIt is employed for determining the transformation from the gripper to the camera. MoveIt [55] is an open-source software framework designed for motion planning and manipulation in robotic systems. It provides a set of tools, libraries, and APIs that streamline the development of robotic applications, particularly those involving robotic arms and manipulation. MoveIt includes a package that facilitates the Hand-Eye calibration process. The complete picking pipeline is illustrated in a simplified manner, outlining the steps in Figure 3.4. Additional details are provided in Section 3.4.

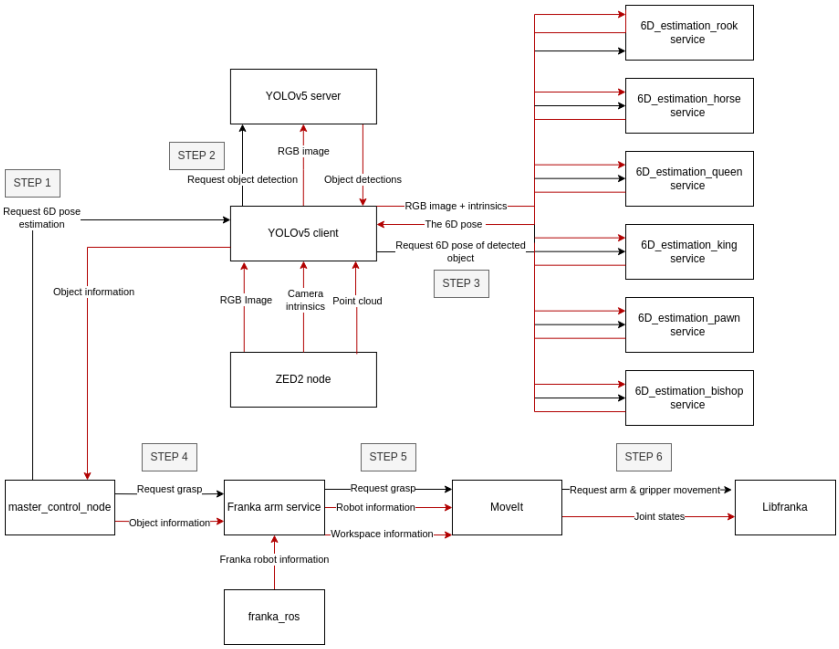


Figure 3.4: The picking pipeline

YOLOv5

YOLOv5 stands for the fifth iteration of the You Only Look Once [56] models. YOLO is a family of real-time object detection models in computer vision. YOLO performs regions of interest proposal and classifying objects within those regions both in a single step. The YOLO models are well-suited for real-time applications because of their efficiency. YOLOv5 stands out as one of the most widely adopted open-source classification models and has gained popularity within the computer vision community, evident from its 14,900 stars on its GitHub repository. In this training setting, a cluster of four Nvidia RTX 3090ti GPUs were employed alongside an AMD Ryzen Threadripper PRO 83955WX CPU featuring 16 cores. The training process utilized the data parallelization mode, which ensures gradient synchronization across the models. The training batch size was set to 64, and there were 49,527 training images compared to 12,382 test images, maintaining an 80-20 ratio. The weights of the yolov5x are used as a pretrained model. This is the biggest pretrained model provided. YOLOv5 uses three distinct loss components. The initial component relates to the loss associated with bounding box regression. The second element focuses on the confidence of an object's presence, termed the objectness loss, as described in [57]. Finally, the third component is the traditional classification loss, which is computed using Cross Entropy. The decrease of the losses can be seen in Figure 3.5 that show a stable learning process within 24 epochs.

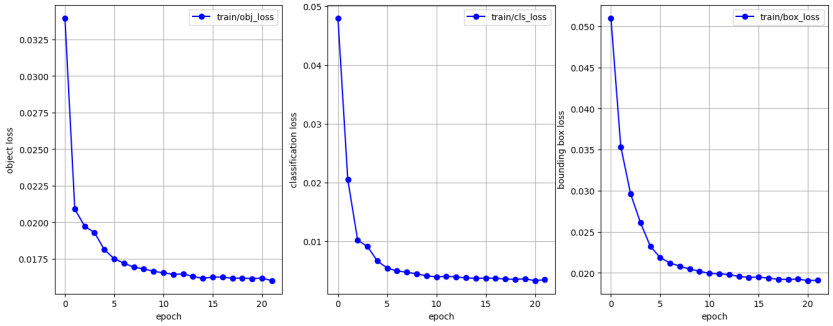


Figure 3.5: The losses of the YOLOv5 training

The model was trained within a single day and reached a stable minimum point where it didn't exhibit further improvements on the test set, as evident from the metrics displayed in Figure 3.6. What can be seen from the graphs is that after the very first epochs the MaP@0.50 is already nearly perfect and is stable at 4 epochs altough the MaP@0.50-0.95 is a stricter metric and shows that the model is only stable after around 15 epochs.

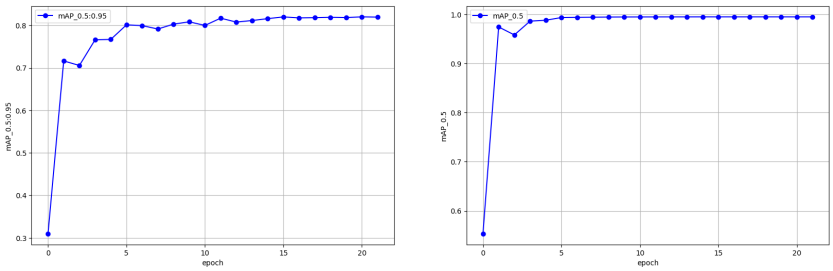


Figure 3.6: Two different metrics on the test set of YOLO training

Hand-Eye calibration

The calibration problem, expressed as $AX = BX$, requires solving for the transformation matrix X. Initial transforms A and B are known, including the base to the end effector through the robot's kinematics and the camera to the target through an ArUco board detection. The transformation matrix X can be determined through various poses. In Figure 3.7, the problem is visually depicted. In this work, a solver employing dual quaternions from [58] is utilized with 10 different poses.

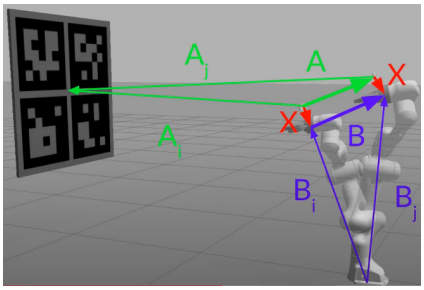


Figure 3.7: The hand eye calibration problem

The picking pipeline

The picking experiment employs a Robotic Operating System (ROS) network, and the simplified flow of the ROS node network pipeline is illustrated in Figure 3.4. In this figure the red lines represent the data transfers between nodes and the black lines represent requests to different nodes. The initial central point of the script is the master control node, responsible for two distinct actions. The first action involves requesting the 6D pose estimation, while the second action is executed after obtaining the 6D pose, focusing on grasping the object. The required data for object detection is received from the ZED2 camera node. Upon obtaining the 6D pose of a specific chess piece, a predefined grasp action is invoked to initiate the picking movement. The franka_ros package establishes a connection between Franka Emika research robots and the entire ROS ecosystem. It incorporates URDF models and intricate 3D meshes of both the robots and end effectors, facilitating visualization and kinematic simulations. When requesting a grasp action, MoveIt is employed to plan the motion of the Franka arm. The motion planning for the arm leverages MoveIt's standard planning library. Once a plan is generated, the specified joint states are transmitted to libfranka, the open-source C++ interface for the robot, facilitating the movement of the actual robot arm.

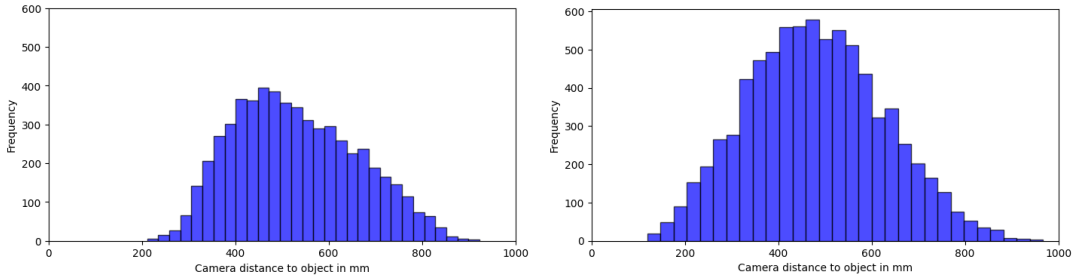
4

Results

This chapter begins with an analysis of the two datasets, the real world 6D pose estimation dataset and the synthetic test set, to explore the datasets data distribution. This is followed by a comparison of 6D pose estimation performance for each object. Following this comparison, the accuracy of depth refinement on real-world data is analysed. Finally, the picking capabilities are analyzed.

4.1. Analysing the datasets

Within the 944 images of the real-world dataset, a maximum of 39 negative instances per object were detected because they were not visible in the image. The size of the synthetic test set can be found in table 3.1. In both datasets chess pieces are placed at varying distances ranging from approximately 200mm to 900mm. The highest frequencies fall within the 400mm to 500mm range. The overall frequency distribution of ground truths distance for all chess pieces is depicted in Figure 4.1. The histograms for both datasets exhibit similar distributions.

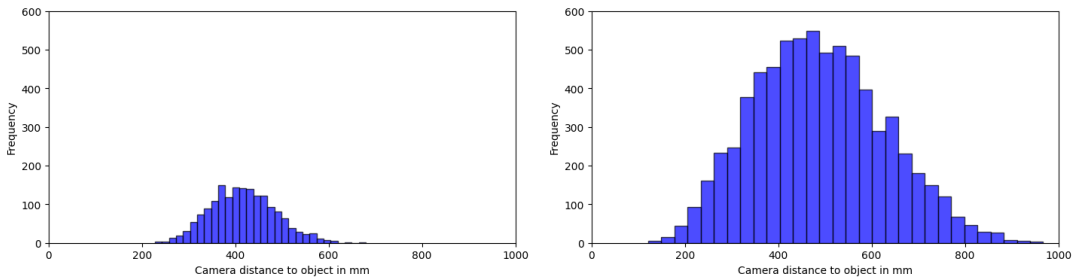


(a) The histogram of all ground truth distances combined for the real world dataset

(b) The histogram of all ground truth distances combined for synthetic dataset

Figure 4.1: The dataset distance distribution

The ground truth locations are entirely known. Following the completion of 6D pose estimations. False positives are identified when the ADD(-S) is greater than the diameter of the respective object, while true positives are recognized when it falls below that threshold. To delve into the distances where the model provides a true positive, for both datasets a histogram is generated for where the model gives a true positive prediction, as can be seen in Figure 4.2.



(a) The histogram of all True Positives distances combined for the real world dataset

(b) The histogram of all True Positives distances combined of the synthetic dataset

Figure 4.2: The datasets true positives distribution

Unlike the synthetic dataset, the ground truth histogram for the real-world dataset is noticeably much smaller than the combined ground truth distances shown in figure 4.1a. Specifically, the synthetic data had a true positive rate of 90%, whereas the real-world dataset showed a significantly lower rate of 30%. In the real-world dataset, there is also a significant decrease in the positive rate of the model beyond the 450mm mark. This decline is attributed to the objects becoming too small in the image. This finding and additional analysis on object detection for the real-world dataset can be found in appendix A.3.

4.2. Analysing the models performance on 6D pose estimation

To assess the 6D pose estimation across various objects, two distinct metrics are used, that are explained in Section 2.2.2. With the traditional ADD(-S) metric this translates to a threshold of a mean point distance below 3.9mm for the pawn and 7.9mm for the king, for instance. Additionally, the AUC is derived from multiple thresholds, where in this work the threshold is varied from 0 to 100% with steps of 0.1%. In Table 4.1, the various metric scores can be seen on synthetic and real-world data.

Chess Piece	Real / Synthetic	ADD(-S) metric [%]	ADD(-S) metric with ICP [%]	AUC [%]	AUC with ICP [%]
Rook	Real	1.85	66.42	65.27	86.18
	Synthetic	9.2	95.5	80.2	93.5
Queen	Real	42.36	74.65	81.54	88.37
	Synthetic	81.3	94.1	88.0	92.5
Pawn	Real	15.60	39.01	75.65	82.49
	Synthetic	47.0	97.0	83.3	95.0
King	Real	58.91	74.32	86.84	89.28
	Synthetic	76.4	92.1	86.8	91.9
Horse	Real	0.0	20.0	61.59	74.64
	Synthetic	22.5	74.6	68.7	85.2
Bishop	Real	23.13	66.67	75.93	89.26
	Synthetic	63.2	96.8	85.7	94.4

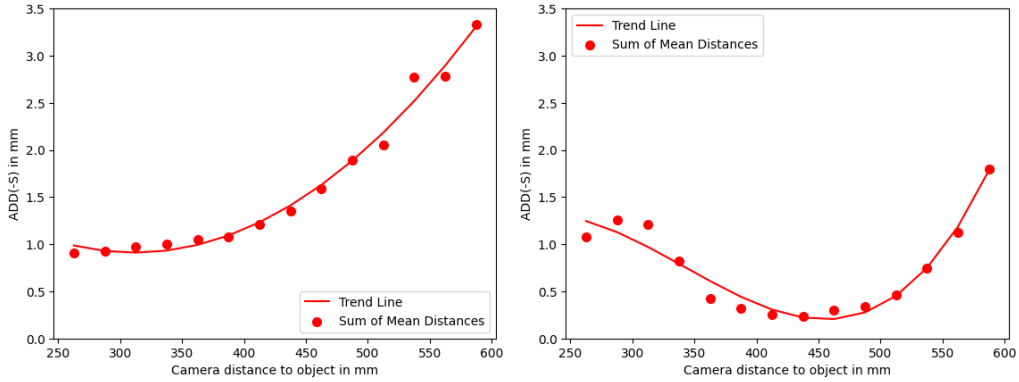
Table 4.1: Metrics for 6D pose estimation

Several observations stand out. Firstly, the synthetic data has much better metric scores, with the ADD(-S) metric having an improvement of 1.5-3 times for each model. The AUC scores also improve a lot, driven by better performance at stricter thresholds. Additionally, substantial enhancements are evident in all metrics following depth refinement. Comparing depth refinement with synthetic data poses a challenge due to the complete and perfect depth maps generated by the computer. This perfection is the reason for the high scores observed with depth refinement on synthetic data. Real-world data lack such perfect depth maps, still it shows good improvements in metrics with depth. The rook has in particular a remarkable improvement, soaring from 1.85% to 66% in ADD(-S), a thirtyfold increase. The horse, on the other hand, continues to pose challenges for the model in predicting rotation, reflected in its lower score on both the synthetic and real-world dataset. Despite an enhancement in score following depth refinement, it remains at the lower end of the scores.

4.3. The accuracy improvement of depth refinement

After comparing synthetic data with real-world data on 6D pose estimation, the accuracy improvement of depth refinement is further analysed. To ensure a fair analysis, only real-world data is considered. Two plots are generated where the ADD(-S) of all objects are plotted against their respective camera distances, both with and without depth refinement. Polynomial trend lines are incorporated into these plots. In figure 4.3a the values of the trend line is $3.15 * 10^{-5} * x^2 - 1.95 * 10^{-2} * x + 3.96$ and for figure 4.3b the value of the trend line is $1.92 * 10^{-7} * x^3 - 1.97 * 10^{-4} * x^2 + 6.03 * 10^{-2} * x - 4.44$. For better comparability, the ADD(-S) of each object is scaled between 0 and 1. The observation is that, without depth refinement, as illustrated in Figure 4.3a, the accuracy of the 6D pose estimation de-

creases as the distance increases. Notably, the decline in accuracy becomes more pronounced beyond 425-450mm, aligning intriguingly with the point where object detection experiences a drop in accuracy. Consequently, the 6D pose estimation model encounters difficulties when dealing with smaller objects as well. When examining the figure with depth refinement in Figure 4.3b, it becomes apparent that, before reaching 325mm, the estimations not only failed to improve but even demonstrated a decline in accuracy. Moreover, depth refinement also has a decline in accuracy, happening around the same range of 425-450mm.

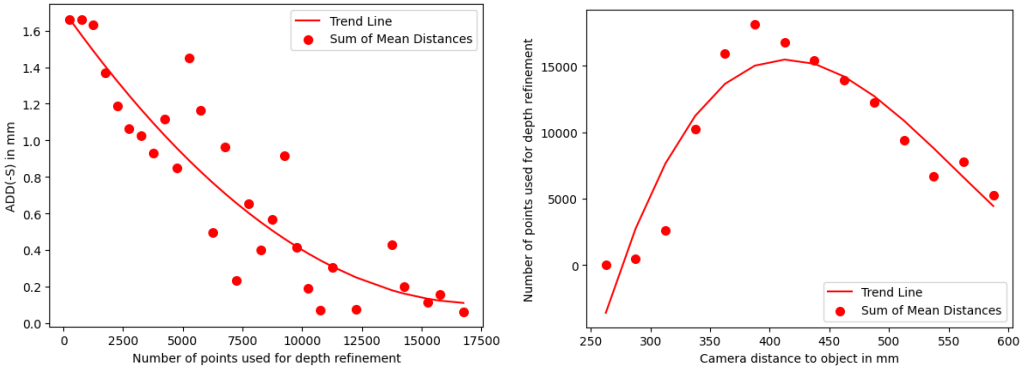


(a) Mean Distances without depth refinement

(b) Mean Distances with depth refinement

Figure 4.3: The mean distance plotted against the camera distance

Furthermore, a detailed analysis of the significance of the number of points used in depth refinement is conducted. The significance is showed in two visual representations as can be seen in Figure 4.4, accompanied by polynomial trend lines. In figure 4.4a the values of the trend line is $5.27 * 10^{-9} * x^2 - 1.84 * 10^{-4} * x + 1.71$ and for figure 4.4b the value of the trend line is $1.43 * 10^{-3} * x^3 - 2.39 * x^2 + 1.24 * 10^3 * x - 1.91 * 10^5$. In Figure 4.4a, the correlation between the number of points and ADD(-S) is illustrated. Although there is still some variation in ADD(-S), it is noteworthy to emphasize the trend of increasing accuracy associated with a greater number of points used in depth refinement. The greater number of points enables a more thorough capture of texture, contributing to enhanced refinement precision.
In Figure 4.4b, the mean number of points is graphed against the camera distance. This plot reveals a minimum camera distance of approximately 325mm and optimal camera performance at around 400mm. This trend aligns with the findings of Figure 4.3b, where depth refinement failed to yield improved results before 325mm, and the optimal refinement occurred around 400mm. Subsequently, the number of points decreases with an increase in camera distance. The observed behavior of a minimum camera distance is attributed to the characteristics of the ZED2 camera, which is documented to demonstrate optimal performance within the range of 300 to 1000mm. An important observation is that the accuracy of depth refinement is tied to the performance of the camera in use.



(a) Number of points plotted against the mean distance

(b) Number of points plotted against the camera distance

Figure 4.4: The significance of the number of points in depth refinement summarized in two plots

4.4. Analysing the picking

The sum of the outcome of the picking experiment is depicted in Table A.3 and detailed plots are available in Table 4.2. These experiments reveal the feasibility of picking chess pieces solely based on synthetic data. Notably, 52% of all detections were successful even without depth information. Furthermore, incorporating depth refinement significantly enhances the success rate, reaching approximately 70.5% for successful picking. Within the individual plots, it becomes evident that, for instance, picking the horse is notably simpler than predicting its 6D pose. This ease of picking can be attributed to the fault tolerance provided by the wide gripping range of the horse's head. Surprisingly, the smallest object, the pawn, exhibited above mean performance with only RGB.

Gripping results rook

	Negative	Positive
Without depth	13	13
With depth	12	13

Gripping results horse

	Negative	Positive
Without depth	9	18
With depth	5	21

Gripping results queen

	Negative	Positive
Without depth	10	15
With depth	6	20

Gripping results king

	Negative	Positive
Without depth	12	14
With depth	7	18

Gripping results pawn

	Negative	Positive
Without depth	9	16
With depth	10	15

Gripping results bishop

	Negative	Positive
Without depth	14	11
With depth	5	20

Table 4.2: Individual Gripping Results for Each Chess Piece

5

Conclusion

In recent years, significant progress has been made in RGB-only pose estimation methods and study to synthetic data. Various research studies indicate that the gap between simulation and reality has been diminishing over the years. Additionally, recent 6D pose estimation papers demonstrate achieving nearly perfect scores across various datasets. However is synthetic data accurate enough for bridging the visual simulation to reality gap for tasks as picking chess pieces? Or do we also still need depth for refinement? To adress this, this study aims to answer the following research questions: **How does the utilization of synthetic data impact the accuracy of 6D pose estimation for RGB images compared to RGB-Depth images? Additionally, how does the performance vary across different types of chess pieces during a picking task?**

To investigate these research questions, six distinct chess pieces underwent 3D scanning to generate CAD models, resulting in the creation of 61,910 synthetically generated images with diverse domain randomization. Using this synthetic dataset, a dedicated 6D pose estimation model was trained for each chess piece. Subsequently, a specific real-world dataset was generated. The models trained on synthetic data were then compared on 6D pose estimation performance between the synthetic and real-world datasets. Additionally, a real-world picking task was executed using a robotic arm. A ROS node network was established, and attempts were made to pick up the chess pieces in 25 trials, both with and without depth information.

5.1. Conclusion

In this work, it was observed that synthetic data can be used for bridging the visual sim2real gap. However, the superior performance on a synthetic test set compared to real-world data implies that results obtained in a synthetic environment cannot be directly projected to real-world scenarios.
Furthermore it was concluded that the models accuracy improves as the object gets closer. Beyond 450mm, there was a noticeable decline in accuracy in object detection, primarily attributed to the diminishing size of the chess pieces. Notably, each chess piece showed substantial improvement after depth refinement. The horse consistently posed challenges for the model in predicting rotation, as evidenced by its consistently lower score compared to other objects, even after depth refinement. Additionally, the analysis of the number of points used in depth refinement revealed a clear correlation between the number of points and accuracy. The ZED2 camera used, demonstrated suboptimal performance for closer distances, as its depth accuracy was insufficient for distances before 300mm. The optimal camera distance was found to be 400mm, where the maximum texture of the object is captured. After this distance, there was an apparent decline in the number of points with respect to camera distance.
The picking results reveal that objects could be successfully picked up approximately 52% of the time using RGB-only information. Surprisingly, even the smallest object, the pawn, demonstrated picking performance above the mean with RGB-only data. Particularly noteworthy is the horse, which demonstrated a notably high positive rate. Although its models achieved the lowest score, this can be attributed to the wide fault tolerance of its head's gripping range. The success rate of picking significantly improved to around 70% with the incorporation of depth refinement.
In conclusion, these findings highlight the potential to narrow the visual sim2real gap. They emphasize that while depth plays a crucial role in substantial improvement of 6D pose estimation, successfully picking various small chess pieces is achievable both with and without depth refinement.

5.2. Discussion and future work

The outcomes of this study open up several potential directions for future research, aiming to advance and expand the existing knowledge on synthetic data, 6D pose estimation and manipulation. Initially

centered on synthetic data training, a valuable extension involves training the 6D pose estimation model on the real-world dataset. This would enable a fair comparison between simulation-based and real-world pose estimation models.
Secondly, a more extensive investigation could be undertaken to explore the training process further. While Appendix A.5 already provides some insight into the current training configuration, a more comprehensive validation of these settings could significantly enhance the understanding of the training dynamics. Possible research directions could include varying the number of training images, like [39], to create an ablation study on how altering the dataset changes the performance on real-world data. Another potential research direction involves investigating if alternative object detection models, such as YoloV8, could be used instead of using EfficientDet as a backbone.
Furthermore, it's worth noting that the real-world dataset and picking experiment lacked substantial domain randomization, like different backgrounds or distractors. The gripping results might have been different with a more diverse domain that is used in real-world settings. An extension to the real-world dataset could also be annotating bounding boxes on the objects, enabling a better analysis of object detection.
This study demonstrates the feasibility of successful picking using synthetic data, as demonstrated by a total of 300 grasping attempts. To improve data quality and potentially gain different insights, there is a possibility to further automate the picking process. For instance, in [38], the authors trained on simulated grasping images and conducted a picking experiment with a significantly larger number of grasps, totaling 25,000.
Additionally, since the chess pieces were positioned upright, the synthetic data reflected this orientation. Future investigations could explore different grasping scenarios, such as chess pieces placed in a box or scenarios where they are positioned differently, including instances where they may have fallen.
Alternatively, considering the evident correlation between the number of points used in depth refinement and accuracy, an improved depth camera designed for shorter distances, like the ZED X MINI which has a minimum distance of 100mm, could potentially enhance the estimation. Exploring advanced methods for capturing the entire point cloud, such as fusing point clouds from different angles to generate additional points for depth refinement, similar to the approach employed by Open3D [59], may also present promising avenues for real-world applications.

References

[1] Juan Fuentes et al. “Psychophysiological stress response of adolescent chess players during problem-solving tasks”. In: *Physiology & Behavior* 209 (July 2019), p. 112609. DOI: 10.1016/j.physbeh.2019.112609.

[2] J. C. F. de Winter et al. “A Role of Peripheral Vision in Chess? Evidence from a Gaze-contingent Method”. en. In: *Journal of Expertise* 6.1 (2023). ISSN: 2573-2773. URL: https://repository.tudelft.nl/islandora/object/uuid%3Aac0ac081-fa22-4b7e-a1fa-5bc09719ac45 (visited on 11/13/2023).

[3] Ray Lc et al. “Power Chess: Robot-to-Robot Nonverbal Emotional Expression Applied to Competitive Play”. In: Oct. 2021, pp. 1–11. DOI: 10.1145/3483529.3483844.

[4] Nandan Banerjee. “A Simple Autonomous Robotic Manipulator for playing Chess against any opponent in Real Time”. In: Aug. 2012.

[5] Prabin Rath et al. “Autonomous Chess Playing Robot”. In: Oct. 2019, pp. 1–6. DOI: 10.1109/RO-MAN46459.2019.8956389.

[6] Jens Golz and Rolf Biesenbach. “Implementation of an autonomous chess playing industrial robot”. In: Nov. 2015. DOI: 10.1109/REM.2015.7380373.

[7] Nikolaus Correll et al. *Analysis and Observations from the First Amazon Picking Challenge*. arXiv:1601.05484 [cs]. Sept. 2017. URL: http://arxiv.org/abs/1601.05484 (visited on 04/04/2023).

[8] Post & Parcel. *Effidence wins DHL Robotics Challenge*. https://postandparcel.info/76645/news/effidence-wins-dhl-robotics-challenge/. Nov. 2016.

[9] Kinemetrix. *How to Make a Robot Pick Up a Part - Introduction to Pick and Place*. Nov. 2019. URL: https://www.youtube.com/watch?v=ZrquLlVzClo.

[10] Yuval Litvak, Armin Biess, and Aharon Bar-Hillel. *Learning Pose Estimation for High-Precision Robotic Assembly Using Simulated Depth Images*. arXiv:1809.10699 [cs]. Mar. 2019. URL: http://arxiv.org/abs/1809.10699 (visited on 05/24/2023).

[11] TUDelft, Dariu M. Gavrila. *3D Machine Vision*. PowerPoint slides. 2022. URL: https://brightspace.tudelft.nl/d2l/le/content/401407/viewContent/2537226/View.

[12] Yingzhao Zhu et al. “A Review of 6D Object Pose Estimation”. In: *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. Vol. 10. ISSN: 2693-2865. June 2022, pp. 1647–1655. DOI: 10.1109/ITAIC54216.2022.9836663.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (visited on 11/13/2023).

[14] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. arXiv:1405.0312 [cs]. Feb. 2015. URL: http://arxiv.org/abs/1405.0312 (visited on 11/13/2023).

[15] *IBM Documentation*. en-US. Sept. 2023. URL: https://www.ibm.com/docs/en/mas-cd/maximo-vi/continuous-delivery?topic=sets-data-set-considerations (visited on 11/13/2023).

[16] Fred Rothganger et al. “3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints”. en. In: *International Journal of Computer Vision* 66.3 (Mar. 2006), pp. 231–259. ISSN: 1573-1405. DOI: 10.1007/s11263-005-3674-1. URL: https://doi.org/10.1007/s11263-005-3674-1 (visited on 04/04/2023).

[17] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. “The MOPED framework: Object recognition and pose estimation for manipulation”. en. In: *The International Journal of Robotics Research* 30.10 (Sept. 2011), pp. 1284–1306. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364911401765. URL: http://journals.sagepub.com/doi/10.1177/0278364911401765 (visited on 04/04/2023).

[18] D.G. Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. Sept. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[19] Stefan Hinterstoisser et al. “Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes”. en. In: *Computer Vision – ACCV 2012*. Ed. by David Hutchison et al. Vol. 7724. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562. ISBN: 978-3-642-37330-5 978-3-642-37331-2. DOI: 10.1007/978-3-642-37331-2_42. URL: http://link.springer.com/10.1007/978-3-642-37331-2_42 (visited on 04/04/2023).

[20] Eric Brachmann et al. “Learning 6D Object Pose Estimation Using 3D Object Coordinates”. en. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Vol. 8690. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 536–551. ISBN: 978-3-319-10604-5 978-3-319-10605-2. DOI: 10.1007/978-3-319-10605-2_35. URL: http://link.springer.com/10.1007/978-3-319-10605-2_35 (visited on 04/04/2023).

[21] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. “Fast Point Feature Histograms (FPFH) for 3D registration”. In: *2009 IEEE International Conference on Robotics and Automation*. ISSN: 1050-4729. May 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[22] Guoguang Du et al. “Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review”. en. In: *Artificial Intelligence Review* 54.3 (Mar. 2021). Number: 3, pp. 1677–1734. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09888-5. URL: https://doi.org/10.1007/s10462-020-09888-5 (visited on 02/06/2023).

[23] Tomas Hodan et al. *BOP: Benchmark for 6D Object Pose Estimation*. arXiv:1808.08319 [cs]. Aug. 2018. URL: http://arxiv.org/abs/1808.08319 (visited on 04/05/2023).

[24] Bertram Drost et al. “Model globally, match locally: Efficient and robust 3D object recognition”. en. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. San Francisco, CA, USA: IEEE, June 2010, pp. 998–1005. ISBN: 978-1-4244-6984-0. DOI: 10.1109/CVPR.2010.5540108. URL: http://ieeexplore.ieee.org/document/5540108/ (visited on 04/11/2023).

[25] Joel Vidal, Chyi-Yeu Lin, and Robert Martí. “6D pose estimation using an improved method based on point pair features”. In: *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*. Apr. 2018, pp. 405–409. DOI: 10.1109/ICCAR.2018.8384709.

[26] Kiru Park, Timothy Patten, and Markus Vincze. “Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. arXiv:1908.07433 [cs]. Oct. 2019, pp. 7667–7676. DOI: 10.1109/ICCV.2019.00776. URL: http://arxiv.org/abs/1908.07433.

[27] Gu Wang et al. *GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation*. arXiv:2102.12145 [cs]. Mar. 2021. URL: http://arxiv.org/abs/2102.12145 (visited on 03/13/2023).

[28] Martin Sundermeyer et al. *BOP Challenge 2022 on Detection, Segmentation and Pose Estimation of Specific Rigid Objects*. arXiv:2302.13075 [cs]. Feb. 2023. URL: http://arxiv.org/abs/2302.13075 (visited on 04/11/2023).

[29] Yisheng He et al. *PVN3D: A Deep Point-wise 3D Keypoints Voting Network for 6DoF Pose Estimation*. arXiv:1911.04231 [cs]. Mar. 2020. URL: http://arxiv.org/abs/1911.04231.

[30] Chen Wang et al. *DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion*. arXiv:1901.04780 [cs]. Jan. 2019. DOI: 10.48550/arXiv.1901.04780. URL: http://arxiv.org/abs/1901.04780 (visited on 03/02/2023).

[31] Hao Su et al. *Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views*. arXiv:1505.05641 [cs]. May 2015. URL: http://arxiv.org/abs/1505.05641 (visited on 11/13/2023).

[32] Martin Sundermeyer et al. *Implicit 3D Orientation Learning for 6D Object Detection from RGB Images*. arXiv:1902.01275 [cs]. July 2019. URL: http://arxiv.org/abs/1902.01275 (visited on 11/13/2023).

[33] Maximilian Denninger et al. *BlenderProc*. arXiv:1911.01911 [cs]. Oct. 2019. URL: http://arxiv.org/abs/1911.01911 (visited on 11/13/2023).

[34] Blender Online Community. *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam: Blender Foundation, 2018. URL: http://www.blender.org.

[35] *GPU Rendering — Blender Manual*. URL: https://docs.blender.org/manual/en/latest/render/cycles/gpu_rendering.html (visited on 11/13/2023).

[36] Maximilian Denninger et al. "BlenderProc: Reducing the Reality Gap with Photorealistic Rendering". en. In: ().

[37] Jonathan Tremblay et al. *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*. arXiv:1804.06516 [cs]. Apr. 2018. URL: http://arxiv.org/abs/1804.06516 (visited on 11/13/2023).

[38] Konstantinos Bousmalis et al. *Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping*. arXiv:1709.07857 [cs]. Sept. 2017. URL: http://arxiv.org/abs/1709.07857 (visited on 12/15/2023).

[39] Stephen James, Andrew J. Davison, and Edward Johns. *Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task*. arXiv:1707.02267 [cs]. Oct. 2017. URL: http://arxiv.org/abs/1707.02267 (visited on 12/15/2023).

[40] Dillon Reis et al. *Real-Time Flying Object Detection with YOLOv8*. arXiv:2305.09972 [cs]. May 2023. URL: http://arxiv.org/abs/2305.09972 (visited on 11/08/2023).

[41] Yu Xiang et al. *PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes*. arXiv:1711.00199 [cs]. May 2018. URL: http://arxiv.org/abs/1711.00199 (visited on 11/08/2023).

[42] Yann Labbé et al. *CosyPose: Consistent multi-view multi-object 6D pose estimation*. arXiv:2008.08465 [cs]. Aug. 2020. DOI: 10.48550/arXiv.2008.08465. URL: http://arxiv.org/abs/2008.08465 (visited on 04/11/2023).

[43] Yannick Bukschat and Marcus Vetter. *EfficientPose: An efficient, accurate and scalable end-to-end 6D multi object pose estimation approach*. arXiv:2011.04307 [cs]. Nov. 2020. URL: http://arxiv.org/abs/2011.04307 (visited on 11/08/2023).

[44] Mingxing Tan, Ruoming Pang, and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. arXiv:1911.09070 [cs, eess]. July 2020. URL: http://arxiv.org/abs/1911.09070 (visited on 11/08/2023).

[45] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. arXiv:1708.02002 [cs]. Feb. 2018. URL: http://arxiv.org/abs/1708.02002 (visited on 11/08/2023).

[46] Zhilu Zhang and Mert R. Sabuncu. *Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels*. arXiv:1805.07836 [cs, stat]. Nov. 2018. URL: http://arxiv.org/abs/1805.07836 (visited on 11/08/2023).

[47] Peter J. Huber. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1 (Mar. 1964). Publisher: Institute of Mathematical Statistics, pp. 73–101. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177703732. URL: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-35/issue-1/Robust-Estimation-of-a-Location-Parameter/10.1214/aoms/1177703732.full (visited on 11/08/2023).

[48] Ross Girshick. *Fast R-CNN*. arXiv:1504.08083 [cs]. Sept. 2015. URL: http://arxiv.org/abs/1504.08083 (visited on 11/08/2023).

[49] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980 [cs]. Jan. 2017. URL: http://arxiv.org/abs/1412.6980 (visited on 11/08/2023).

[50] Gwon Hwan An et al. "Charuco Board-Based Omnidirectional Camera Calibration Method". en. In: *Electronics* 7.12 (Dec. 2018). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 421. ISSN: 2079-9292. DOI: 10.3390/electronics7120421. URL: https://www.mdpi.com/2079-9292/7/12/421 (visited on 11/16/2023).

[51] "An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)". In: vol. 81. Apr. 1981.

[52] Richard Honti, Ján Erdélyi, and Alojz Kopáčik. “Plane segmentation from point clouds”. en. In: *Pollack Periodica* 13.2 (Aug. 2018), pp. 159–171. ISSN: 1788-1994, 1788-3911. DOI: 10.1556/606.2018.13.2.16. URL: https://akjournals.com/doi/10.1556/606.2018.13.2.16 (visited on 11/23/2023).

[53] URL: http://www.sanko-shoko.net/note.php?id=3c5m.

[54] “ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation”. en. In: (). DOI: 10.5281/zenodo.7347926. URL: https://zenodo.org/records/7347926 (visited on 11/08/2023).

[55] David Coleman et al. “Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study”. In: *Journal of Software Engineering for Robotics* 5.1 (May 2014), pp. 3–16. DOI: 10.6092/JOSER_2014_05_01_p3.

[56] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. arXiv:1506.02640 [cs]. May 2016. URL: http://arxiv.org/abs/1506.02640 (visited on 11/08/2023).

[57] Zhaohui Zheng et al. *Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression*. arXiv:1911.08287 [cs]. Nov. 2019. URL: http://arxiv.org/abs/1911.08287 (visited on 11/08/2023).

[58] Konstantinos Daniilidis. “Hand-Eye Calibration Using Dual Quaternions”. In: *GRASP Laboratory, University of Pennsylvania* (1999).

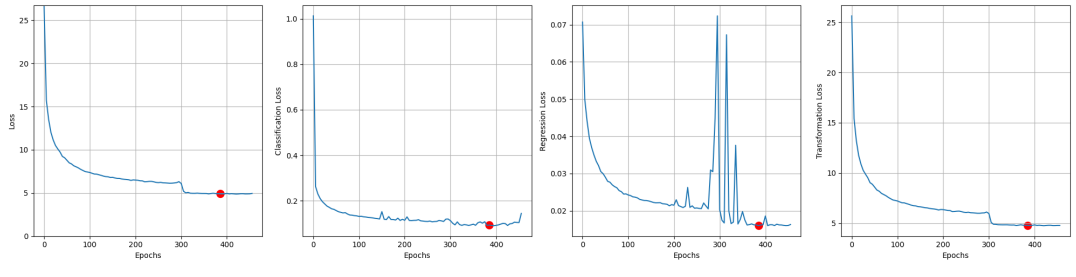
[59] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. “Robust reconstruction of indoor scenes”. en. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, June 2015, pp. 5556–5565. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7299195. URL: http://ieeexplore.ieee.org/document/7299195/ (visited on 12/18/2023).

[60] George Philipp, Dawn Song, and Jaime G. Carbonell. *The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions*. arXiv:1712.05577 [cs]. Apr. 2018. URL: http://arxiv.org/abs/1712.05577 (visited on 11/15/2023).

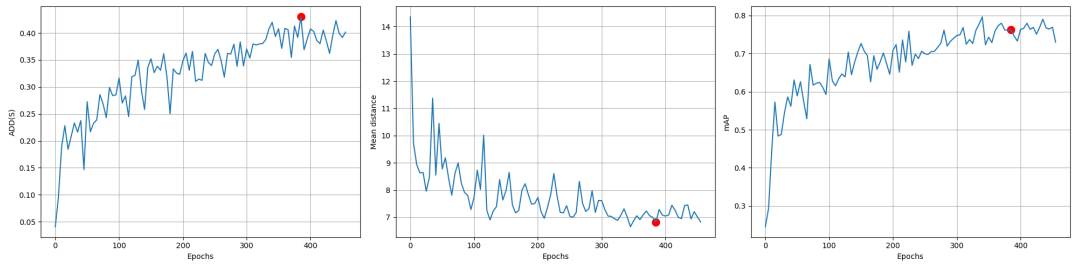
A

Appendix

A.1. Validation Metrics and Loss plots of 6D pose model training

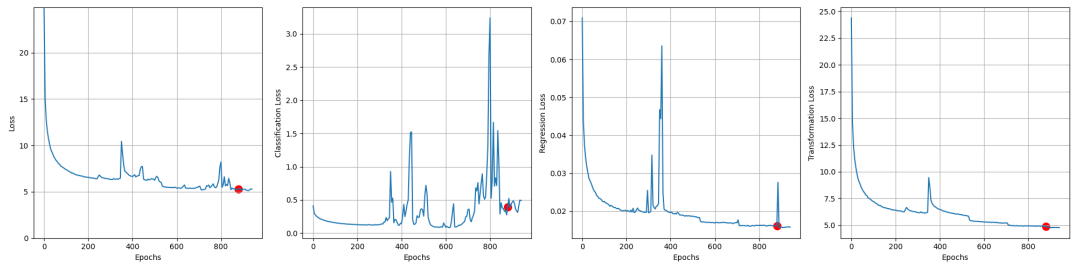


(a) Training Loss for the rook

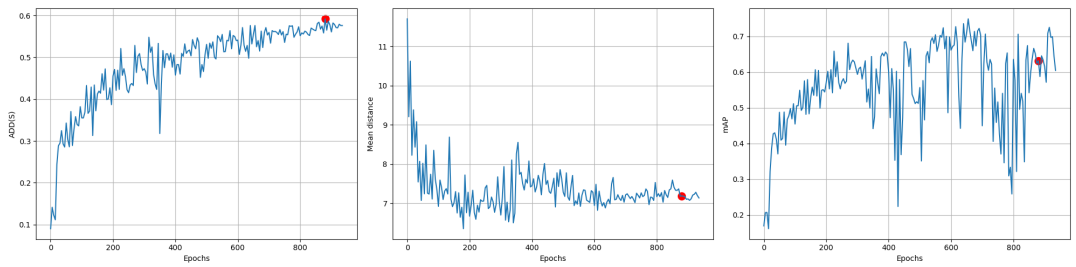


(b) Validation Metrics for the rook

Figure A.1: Validation Metrics and Loss for the rook

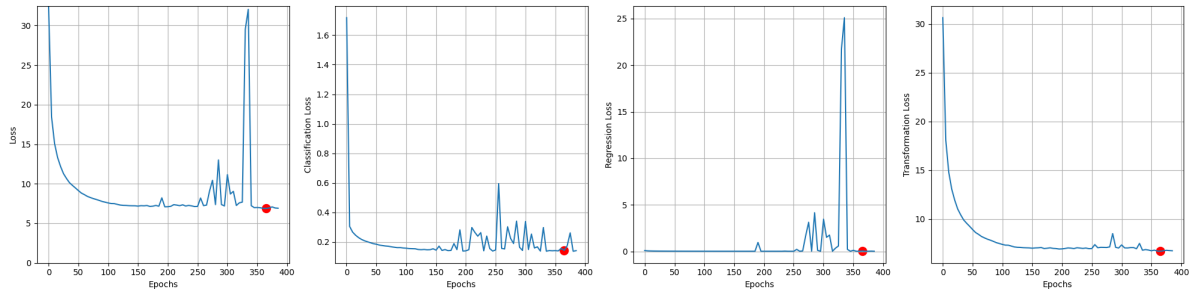


(a) Training Loss for the queen

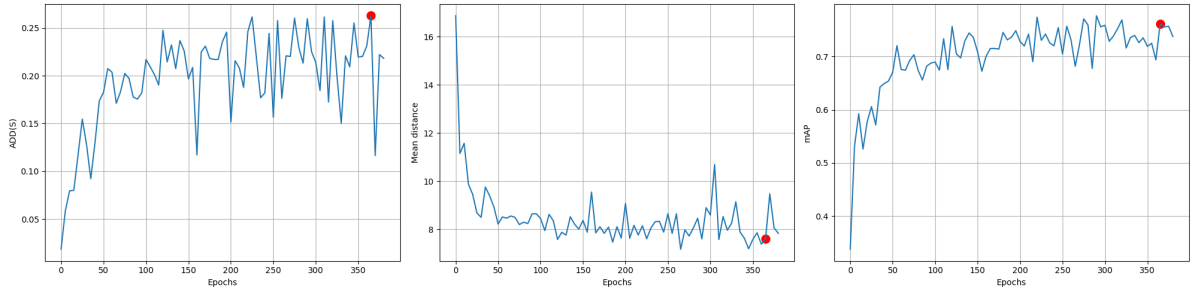


(b) Validation Metrics for the queen

Figure A.2: Validation Metrics and Loss for the queen

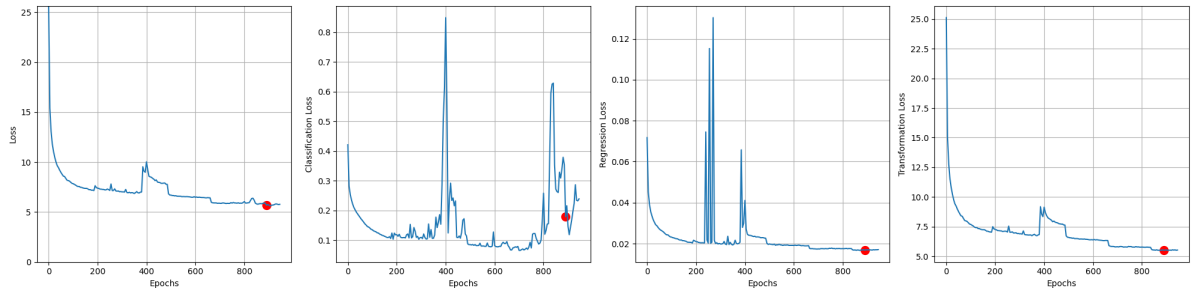


(a) Training Loss for the pawn

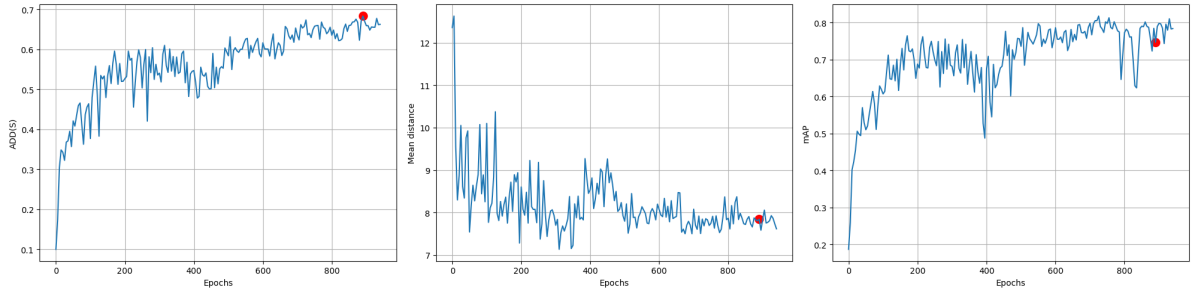


(b) Validation Metrics for the pawn

Figure A.3: Validation Metrics and Loss for the pawn

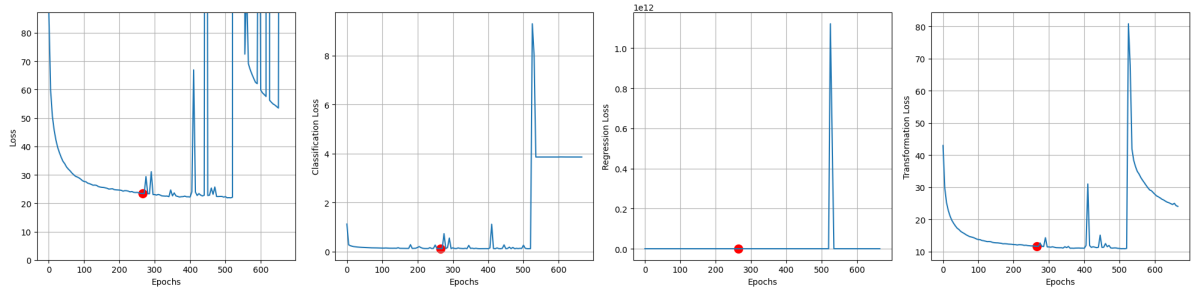


(a) Training Loss for the king

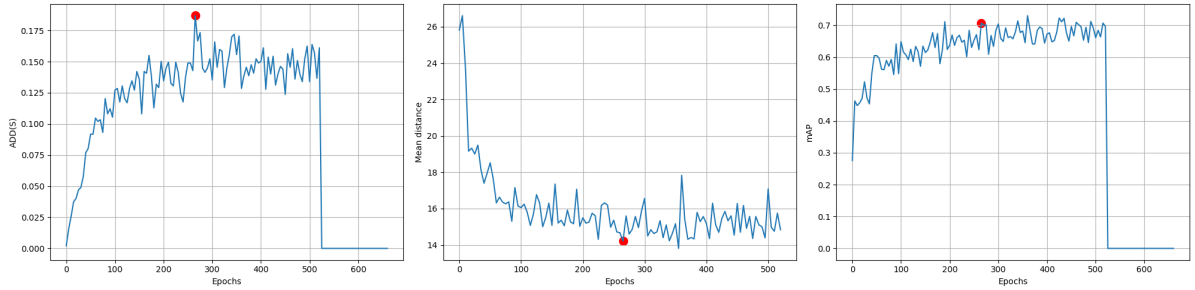


(b) Validation Metrics for the king

Figure A.4: Validation Metrics and Loss for the king

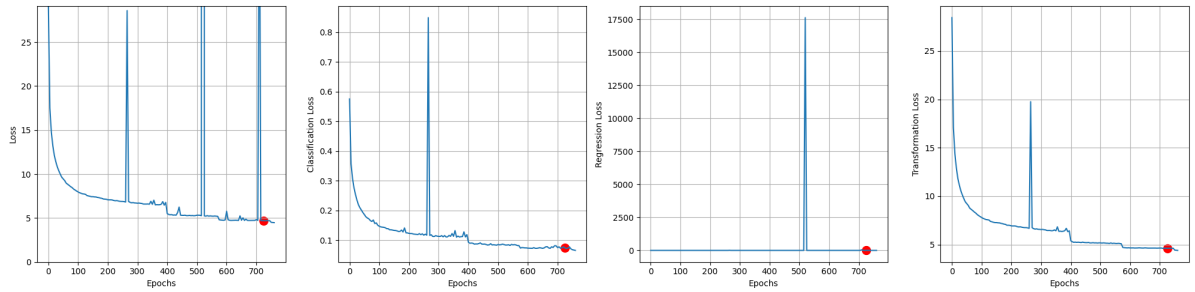


(a) Training Loss for the horse

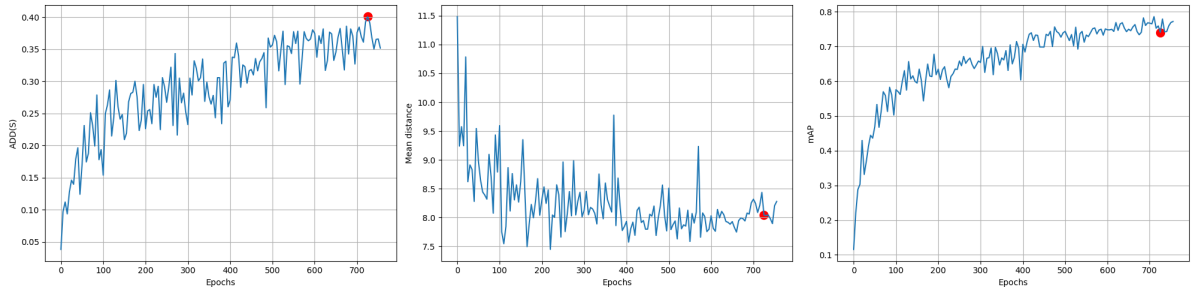


(b) Validation Metrics for the horse

Figure A.5: Validation Metrics and Loss for the horse



(a) Training Loss for the bishop



(b) Validation Metrics for the bishop

Figure A.6: Validation Metrics and Loss for the bishop

A.2. The picking experiment

Figure A.7: The picking test setting

A.3. Additional analysis on object detection for the real-world dataset

Following additional analysis, the focus is on the bounding boxes of true positives in relation to each camera distance. This exploration gives insights into the size of chess pieces within the images. As depicted in Figure A.8, The chess pieces are grouped according to camera distances of 50mm, and the average bounding box size is calculated for all chess pieces. The findings indicate that, at the closest camera distances, the identified objects have dimensions of 20mm along the x-axis and 27.5mm along the y-axis. However, beyond 450mm, their estimated size reduces to 10mm to 12.5mm along the x-axis and 15mm to 17.5mm along the y-axis. It becomes apparent that, beyond this threshold, the objects diminish in size, making their reliable detection too difficult as can be seen by the polynomial trend line.

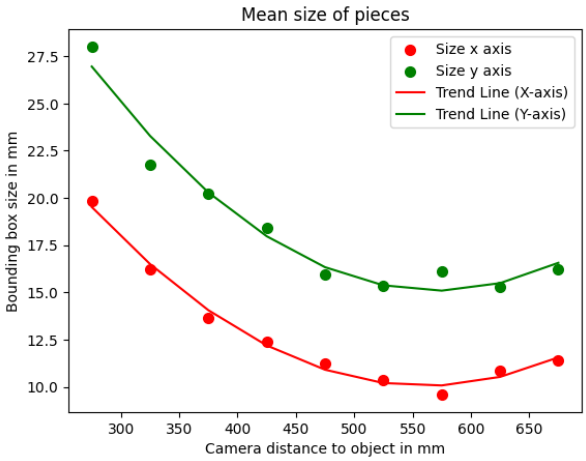


Figure A.8: Scatter plot of the size of chess pieces per each camera distance

Upon a more in-depth examination of the confusion matrix, essential metrics for object detection, including precision, recall, accuracy, and F1 score, are extracted, as depicted in Table A.1. The outcomes reveal a notable low score in recall, attributed to the multiple false negatives, potentially coming from

reduced detection performance beyond 450mm. Another noteworthy observation relates to the pawn: while the remaining objects exhibit relatively consistent scores with precision ranging from 0.7 to 0.8 and recall from 0.3 to 0.4, the pawn displays a higher recall but lower precision. This implies that the pawn model generates more predictions, resulting in fewer false negatives but more false positives. Consequently, the F1 score for the pawn even is the second-highest. Furthermore, the horse consistently demonstrates the lowest scores across all object detection metrics.

Chess Piece	Precision	Recall	Accuracy	F1 Score
Rook	0.78	0.32	0.3125	0.46
Queen	0.7978	0.3337	0.3136	0.4706
Pawn	0.5081	0.4399	0.3305	0.4716
King	0.7770	0.4086	0.3920	0.5356
Horse	0.7878	0.2816	0.2680	0.4149
Bishop	0.6869	0.3698	0.3273	0.4808

Table A.1: Metrics for Object Detection Models

		Predicted	
		Negative	**Positive**
Actual	**Negative**	8	66
	Positive	625	245

(a) Confusion Matrix of the horse

		Predicted	
		Negative	**Positive**
Actual	**Negative**	8	73
	Positive	575	288

(b) Confusion Matrix of the queen

		Predicted	
		Negative	**Positive**
Actual	**Negative**	39	95
	Positive	479	331

(c) Confusion Matrix of the king

		Predicted	
		Negative	**Positive**
Actual	**Negative**	15	134
	Positive	501	294

(d) Confusion Matrix of the bishop

		Predicted	
		Negative	**Positive**
Actual	**Negative**	30	273
	Positive	359	282

(e) Confusion Matrix of the pawn

		Predicted	
		Negative	**Positive**
Actual	**Negative**	24	43
	Positive	483	394

(f) Confusion Matrix of the rook

Table A.2: All Confusion Matrices

A.4. Picking results

	Gripping result	
	Negative	**Positive**
Without depth	74	80
With depth	45	107

Table A.3: Matrix of the sum of all gripping results

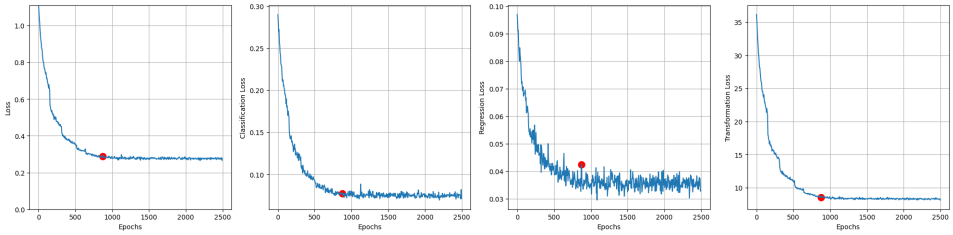
A.5. Further analysis of transformation loss factor

This section delves into the distinction between a transformation factor of 0.02 and a transformation factor of 1, as employed in this thesis. The comparison is illustrated with a focus on the rook and the original 0.02 transformation factor, given the time-intensive nature of training. The hypothesis is that the model will better learn the transformation, particularly due to the smaller size of the pieces. With a factor of 1, there were instances of achieving 0.45 ADD(-S) on the test set (see Figure A.1b), while with a factor of 0.02, the ADD(-S) hovered around 0.23 Figure A.9b. One notable distinction in training with

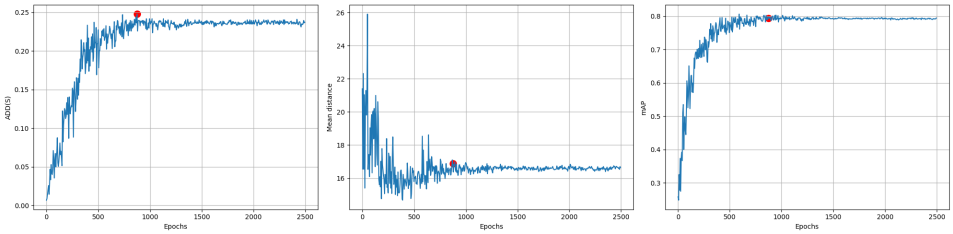
a factor of 0.02 is the increased stability evident in the graph (refer to Figure A.9a). Conversely, in the plots involving a transformation factor of 1, the stability at later epochs is notably lower with much losses that fluctuate. This can be from many different factors, for example exploding gradients [60] where the gradients of the loss function with respect to the model's parameters become extremely large, often leading to numerical instability.

Upon comparing the results of object detection between the original model and the 0.02 model in Appendix A.5 and Table A.5, it is evident that the 0.02 model outperforms the original across all categories, indicating a convergence to a better classification performance.

Now, the transformation scores are being compared in Table A.5, and surprisingly, even in this aspect, the factor of 0.02 performs slightly better, although the difference is marginal. This suggests that achieving convergence to an improved ADD(-S) score might simply be indicative of overfitting. However, since the differences in transformation scores are minimal, all the conclusions presented in this paper remain valid. The results might even be more favorable with a transformation factor of 0.02.



(a) Training Loss for the rook with 0.02 transformation loss



(b) Validation Metrics for the rook with 0.02 transformation loss

Figure A.9: Validation Metrics and Loss for the rook with 0.02 transformation loss

		Predicted	
		Negative	**Positive**
Actual	**Negative**	24	76
	Positive	573	271

(a) Confusion Matrix of the rook with transformation factor of 1

		Predicted	
		Negative	**Positive**
Actual	**Negative**	24	43
	Positive	483	394

(b) Confusion Matrix of the rook with transformation factor of 0.02

Table A.4: Confusion Matrices comparison of the rook

Chess Piece	Transf. factor	Precision	Recall	Accuracy	F1 Score
Rook	1	0.78	0.32	0.3125	0.46
Rook	0.02	0.9016	0.4493	0.4428	0.5997

Table A.5: Comparison of metrics for Object Detection Models

Chess Piece	Transf. factor	ADD(-S)	ADD(-S) with ICP	AUC	AUC with ICP	Diameter (mm)
Rook	1	1.85	66.42	65.27	86.18	47.76
Rook	0.02	3.30	63.20	68.60	86.24	47.76

Table A.6: Comparison of metrics for 6D pose estimation