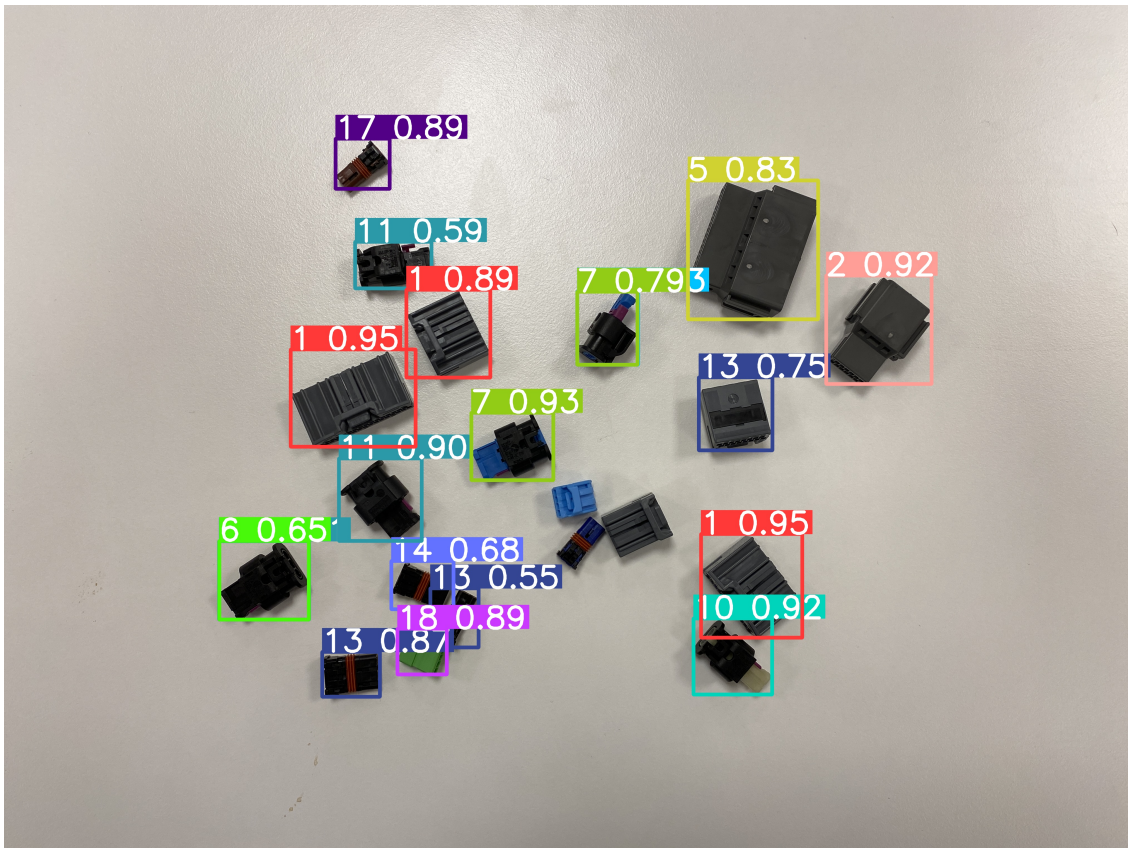
CHALMERS
UNIVERSITY OF TECHNOLOGY



Synthetic Data generation for Object Recognition and Pose Estimation

Master's thesis in Industrial and Materials Science

DESHENG ZHANG & XIANGBO KONG

Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Synthetic Data generation for Object Recognition and Pose Estimation

Desheng Zhang
desheng@student.chalmers.se

Xiangbo Kong
xiangb@student.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
Gothenburg, Sweden 2024

Synthetic Data generation for Object Recognition and Pose Estimation
DESHENG ZHANG & XIANGBO KONG

Supervisor: Hao Wang, Chalmers
Supervisor: Silvan Marti, Chalmers
Examiner: Björn Johansson, Chalmers

Master's Thesis 2024
Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: A result of object detection by detector trained on synthetic data.
Typeset in LaTeX
Gothenburg, Sweden 2024

Synthetic Data generation for Object Recognition and Pose Estimation
DESHENG ZHANG & XIANGBO KONG
Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology

Abstract

The automotive industry is looking for methods to promote a flexible robotic assembly on objects which are difficult for conventional robotic systems based on pre-programming. Specifically, it is difficult to automate wire harness assembly onto vehicles due to the limitation of current robotic assembly on pre-programed tasks and the deformability of wire harnesses. In order to achieve flexible wire harness assembly a well developed vision system. It is necessary for robots to perceive the spatial information of wire harnesses and adapt their actions to handle the objects. Computer vision techniques such as object recognition and pose estimation have been widely adopted in robotics to promote the perception capabilities of robots and the significant advancement in deep learning has remarkably promoted the research in computer vision. However, a vast amount of time and human effort are needed to collect and annotate the datasets for training deep learning models. In recent years, researchers apply synthetic dataset in computer vision tasks, which requires less human effort and promises good annotation quality. In this thesis, the synthetic datasets of connectors are created by using a physically based rendering method BlenderProc and procedures for data creation are provided for further research and investigations. Then, the performance of synthetic datasets are evaluated by object detection models(Yolov5 [22] and Yolov8 [23]) and a pose estimation model (Wide Depth Range [20]). Also, the influences of applying domain randomization methods(e.g. adding distarctors into the synthetic dataset) is discussed and evaluated. By evaluating the experiment results, the study finds that similar objects will cause mis-classification problems in connectors detection tasks, the domain gap will lead to a poor performance on real data and adding distractors into synthetic dataset can improve the robustness of the detectors. The study concludes with recommendations for future research, such as using Generative Adversarial Networks (GANs) to transfer the overall color and texture from the source images to the target images, or apply a bilevel optimization approach. These kinds of methods improve the domain gap between synthetic data and real data, thereby improving the performance of models trained on synthetic datasets in the future.

Keywords: Synthesis dataset, object recognition, pose estimation, computer vision, machine learning, neural network, domain randomization, domain gap

Acknowledgements

We are sincerely honored to have the opportunity to complete this thesis, and we are deeply thankful to everyone who supported and guided us during the process of this master's thesis.

Firstly, we would like to thank our supervisor at Chalmers University of Technology, Hao Wang, he provided valuable advice on study. And we deeply thank Daniel Magnfält, he introduced CAD models to us and its structure. Their support helped us overcome various challenges.

We are also very grateful to our supervisor Silvan Marti and the examiner Björn Johansson from the Department of Industrial and Materials Science at Chalmers University of Technology, they give us their feedback and critical insights, which enhanced the quality of this study.

Desheng Zhang, Xiangbo Kong, Gothenburg, 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CNN	Convolution Neural Network
FPN	Feature Pyramid Networks
PNP	Perspective-n-Points

Contents

List of Figures

List of Tables

1
Introduction

This section mainly introduces the background of this master thesis, and also will show the aim and research question of this master thesis, these are closely related to the following section part.

1.1 Background

Nowadays, many industrial applications require more robotic automation to address production problems regarding productivity, quality, safety, and ergonomics. But robotic automation is difficult to meet the requirements of the industry. For example, the automotive industry is facing problems related to wire harness assembly onto vehicles, it is difficult to automate due to the limitation of conventional robotic assembly systems on pre-programmed tasks and the deformability of wire harnesses [37]. In order for robots to process industrial tasks such as wire harness assembly, it is necessary to perform object recognition and pose estimation.

Recent research in object recognition and pose estimation involves the application of machine learning [45]. An example is to use 3D point clouds to obtain 3D information and point pair features to obtain pose estimation, in order to improve the flexibility of potential robot recognition [11].

However, applying machine learning techniques commonly involves large data sets for training, which usually involves collecting training data of target objects using mechanical equipment and annotating information manually. This kind of method has gradually become difficult because collecting and annotating datasets to train learning based models requires a large amount of time and human-work [30]. Therefore, using synthetic dataset has become an alternative choice, considering its potential as a faster and cheaper alternative method to collecting and generating large amounts of annotated data [28].

Although using synthetic data could solve the problem of the human-work and time consumption in manual annotation, there still is one problem that needs to be considered. Synthetic data always has a domain gap with real data [4, 31]. Therefore, it needs to use some method to improve the performance of training and testing synthetic data on real datasets, like domain randomization [7]. One example used as the method in this thesis is adding some distractors to compose a mixed synthetic data, this could be an effective domain randomization way to improve the

performance [40].

1.2 Aim and purpose

This master's thesis adopts the method of creating synthetic datasets to reduce manual work and time, and improve the quality of annotations.

The aim of this master's thesis is first to generate the synthetic datasets of small objects and some annotated data from 3D CAD models, and then apply the object detection models and pose estimation models. In the end, evaluate and compare the results of object detection models and pose estimation model on real clean dataset[1], distractor dataset[2], and real images of the objects.

An idea is to train networks using clean dataset or distractor dataset and test them using clean dataset, distractor dataset and real images of objects (or mixed dataset that include these three parts) [40]. In object recognition, the network will identify the target object's 2D bounding boxes. In pose estimation, an important part is to find the corresponding relationship between the 3D model points[3] and the 2D image points[4] [29], which we describe in detail in section 2, also discusses the Perspective-n-Point (PNP) algorithm [13] which will need to use these correspondences.

The purpose of this master's thesis is to show the adaptability of synthetic data to complex real-world environments by comparing the performance and applicability of small object detection and pose estimation in clean dataset, distractor dataset, and real images of the objects.

1.3 Research Questions

1. How could synthetic dataset for small objects be generated?

   Most of the synthetic dataset is used for the life scene like bop [17], YCB [44], these datasets normal use Blender to output, and we want to know if Blender also could output synthetic datasets of industrial connectors. And basically on the tasks and aim, this research question is necessary for considering.

   From [27, 28, 24], we could use theory of computer vision and deep learning to generate synthetic dataset by using blender, which is an open-source and support the entire 3D production process software.

[1] This means only include connectors

[2] This means include some connectors with some other objects

[3] 3D model points means the coordinates on the object model in 3D space.

[4] 2D image points means the points of an object in 2D image and the corresponding points in 3D object model.

2. What is performance of training on synthetic dataset and testing on real data?

   Based on the background, aim and purpose part, we would like to know the model performance on synthetic dataset for small objects when applied to real data. And we also want to know if adding distractor to the synthetic dataset could improve these model performances, to improve the issue of domain gap.

   We have read some literature [28, 19, 12] and conducted the experiments of object recognition and pose estimation.

   We have find two object recognition models: Yolov5 [22] and Yolov8 [23] and we basically use four different evaluation metrics Precision, Recall, mAP, IOU [38, 35].

   We also find one pose estimation model: Wide Depth Range (WDR) [20] and we will use Add10 as evaluation metrics [27].

3. How does using domain randomization methods (such as adding some distractors) affect performance?

   Basically from background and [40], adding some distractors into the clean synthetic dataset could be an effective domain randomization way. Therefore, datasets containing distractors usually could increase the model's ability to deal with complex environments, and this could improve its generalization ability in different environments. We could compare the evaluation metrics value and discuss which dataset has better effect.

1.4 Delimitation

This master's thesis will first focus on the steps of creating synthetic datasets, and then evaluate the performance of different types of datasets on object recognition models and pose estimation models. These datasets include clean dataset, distractor dataset, and real images of the objects. This thesis will provide detailed steps to how to construct these datasets, including methods for processing synthetic data, as well as how to use this synthetic data on training and testing models of object recognition and pose estimation.

In addition, this thesis will also discuss the adaptability and accuracy of object recognition models and pose estimation models in different environments, such as its performance in adding distractors and multi object scenes. By comparing the performance of the model on these different datasets, we can better understand the strengths of the models and different datasets, thereby providing guidance for future research and applications.

This master's thesis will not involve other topics of mechanical design, control systems, or creating some new deep learning algorithm, this thesis will only focus on creating synthetic datasets and evaluate the performance on object recognition models and pose estimation models based using the these different datasets.

It will not involve other topics of the robot system, such as mechanical design, control systems, or other sensory systems (such as tactile or sound perception).

1.5 Thesis Outline

The following section introduces the theory of synthetic dataset, object recognition and pose estimation. Section 3 talks about the detailed method steps on creating synthetic dataset, training and testing the model of object recognition and pose estimation. Section 4 shows the results of synthetic dataset, model of object recognition and pose estimation and discussion. Section 5 focuses on drawing conclusions and future work.

2

Synthetic dataset, object recognition and pose estimation

2.1 Synthetic dataset

In industrial environments, virtual models (3D CAD models) of many parts already exist from the design stage of the product, so this virtual model and rendering engine could be used to create synthetic dataset.

Now synthetic dataset is used to simulate real objects in many senses, the reason is for real objects, data collection and annotation manually will require too much effort and time [24, 27].

In our master's thesis, we created the synthetic dataset to collect a large amount of training data required for object recognition and pose estimation in later experiments.

2.1.1 Blender

Blender is an open-source 3D graphics and animation production software that provides full process 3D production functions from model creation, texture mapping, dynamic simulation, lighting mapping to final rendering output.

Blender has built-in rendering engines such as Cycles and Eevee, which utilize physics based rendering techniques to simulate how light interacts in the real world. For example, Cycles uses ray tracing algorithms to generate realistic images by simulating the interaction between light rays and the surface of an object [2].

2.1.2 BlenderProc

BlenderProc is a library of functions created based on Blender, used to automate the creation and rendering of 3D scenes, particularly for generating synthetic datasets for machine learning training.

BlenderProc could randomize various parameters of the scene, such as camera position, light color, light position, background texture, and object placement, and then generate a large amount of marker data. By using its built-in physical interactions, the position of objects in the dataset could be made more realistic [9, 10].

2.1.3 Domain gap

The domain gap refers to performance decrease when machine learning models trained in a source domain (training environment) apply their learned knowledge in a different domain (real environment). This performance decrease is mainly because of differences in data feature distribution between the source and target domains, especially in cross-domain learning or transfer learning [4].

Usually, models are trained in a well-annotated source domain and then tested and applied in a target domain where annotations are less or do not have annotations. The main differences between the source and target domains may include, but are not only these factors, in lighting color, lighting power or position, background texture, object scale, and viewing angles. These factors could lead to significant statistical differences between the data in the source and target domains, and then impact the model's predictive performance. Therefore, improving the gap between these two domains is the key point to increasing model performance [31].

The existence of a domain gap is still the problem. Figuring out the domain gap issue often requires introducing additional regularization or adversarial training methods during the model training, or by adjusting and optimizing the feature representations learned from the source domain, in order to make them more suitable for the feature distribution of the target domain. This helps to increase the adaptability and robustness of the model when faced with new domains [4, 31]. Figure 2.1 shows the example of source domain to target domain, which uses domain adaptation and semantic segmentation models learning from synthetic images (source domain) and then test on real images (target domain).

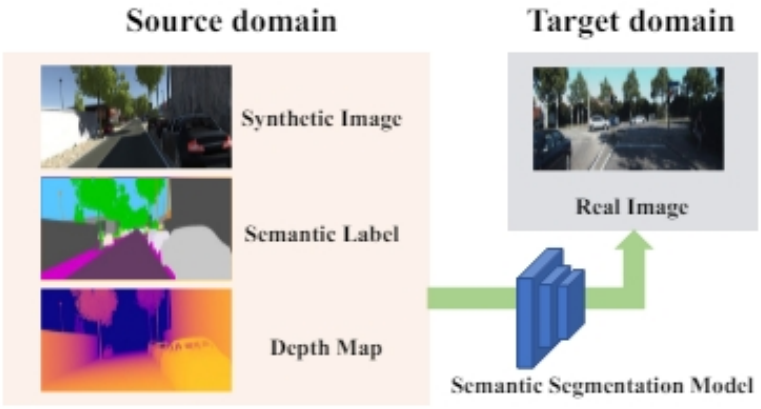


Figure 2.1: Example of source domain to target domain [8]

2.1.3.1 Domain Randomization

The theory basis of domain randomization: By increasing the diversity of the training environment, the model's adaptability to unknown environments could be improved. This method uses random environmental attributes during the training, allowing the training data to cover a wider range of senses. Therefore, the model which uses domain randomization could respond better to various changes in the environment, thereby improving the issue of domain gap [7].

Technique of the blenderProc randomization parameter is a type of domain randomization. Domain randomization introduces randomness into training data to simulate a different environment, then the trained model will be better adapted to real-world environments. It could also be used to increase the generalization ability of machine learning models.

2.1.4 Related work

In recent years, synthetic datasets have been proven as a useful tool in object detection and pose estimation. Synthetic dataset could easily get any number of annotated data, so it could solve the problem of the large amount of time and human-power in real data collection and annotation. In [41], the authors used synthetic human dataset to train pose estimation models and show the effectiveness of the model on real datasets. Although this is not the synthetic dataset for small objects, it still could prove the effectiveness of training by synthetic dataset and test on real datasets.

Although synthetic dataset has brought benefits, the authenticity of synthetic dataset is still the question that needs to be considered. In [21], the authors proposed a method to increase the realism of the synthetic dataset by adding noise to improve the model's performance in the real world.

2.2 Object recognition

2.2.1 Classification

In supervised learning, classification uses a convolutional neural network (CNN) to predict the probability of objects belonging to each category in each region.

There are usually two situations [32]: single label classification (each region only contains one main object category) and multi label classification (a region contain multiple objects); For single label classification, use the softmax function [32, 6]:

$$p(c \mid x) = \frac{e^{x_c}}{\sum_{i=1}^{N} e^{x_i}}.$$

For multi label classification, use the sigmoid function [32, 6],

$$p(c \mid x) = \frac{1}{1 + e^{-x_c}}.$$

2.2.2 Localization

Localization determines the specific position of the target object in the image, in order to complete the object detection task. Localization is achieved by predicting the center point coordinates (b_x, b_y), width b_w, and height b_h of the bounding box,

$$\text{Central coordinates: } b_x = \sigma(t_x) + c_x;\ b_y = \sigma(t_y) + c_y$$

$$\text{Width: } b_w = p_w e^{t_w}$$

$$\text{Height: } b_h = p_h e^{t_h}$$

where (t_x, t_y, t_w, t_h) are values predicted by the model, σ is sigmoid function, (c_x, c_y) is coordinates of the upper left corner of the bounding box, (p_w, p_h) is the original width and height of the bounding box [34].

2.2.3 Yolov5 and Yolov8

The YOLO series of methods use CNNs and the Darknet architecture, more details are following. Overall, these two methods we use could perform efficient convolution operations and make multi-scale predictions, generating predictions at different levels (scales), allowing for simultaneous detection of both large and small objects.

Generating bounding box: YOLO will deal with the image as the input through convolutional layers. Then directly predicts the coordinates, categories, and confidence of the bounding box in the output layer. Each bounding box contains the coordinates of the center point, the width and height of the box, and a confidence score that represents the model's level of confidence in containing specific categories of objects within the box.

2.2.3.1 Yolov5

1. **Data Augmentation**:

   Data augmentation is a technique that increases the variability of the input images which aims to improve the robustness of the detection model when it deals with images obtained from different environments. Commonly used data augmentation methods are divided into two types: photometric distortion and geometric distortions. As for photometric distortion, researchers adjust the contrast, brightness, hue, saturation of the images. For geometric distortion, researchers apply random scaling, cropping, flipping and rotation on the images.

   One of the unique data augmentation methods frequently used in YOLO later versions is called Mosaic[3] which mixes multiple training images into one new image. To be specific, it first selects four random images from the training set. Then, each image is randomly cropped and combined into one image with each image occupying one quadrant of the image. The coordinates of each object's bounding box will also be adjusted according to the new image layout.

   One advantage of applying mosaic augmentation is that combining four images into one larger image could increase the variety of training samples and help the model learn and handle objects under various contexts and scales which can improve the robustness of the model. In addition, mosaic augmentation can increase the portion of small objects in the new images which let the model pay more attention to the small objects and improve the detection of small objects.

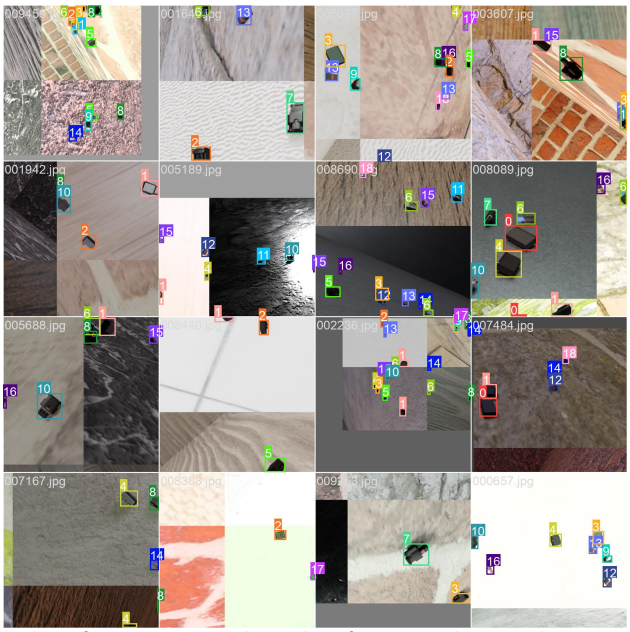


Figure 2.2: Example of a training batch after applying data augmentation.

2. **Network**:

   The model of YOLOv5 is structured into three components, the backbone, neck and head. The overview of the architecture is shown in Figure2.3 below.

   - **Backbone**:

     The backbone of YOLOv5 is CSPDarknet53 which focuses on feature extraction. CSPDarknet is a modified version of Darknet[33] which incorporates the ideas of Cross Stage Partial Network(CSPNet)[42] and Spatial Pyramid Pooling(SPPNet)[16].

     In CSPNet, the feature map of the base layer is divided into two parts and then merged through a cross stage hierarchy. The main purpose is to ensure the gradient flow propagates through two different paths of networks, thus the different parts of the network can learn complementary representations and reduce the impact of redundant gradient information. By applying the cross-stage partial connection, the number of computations is also reduced which improves the inference speed. In YOLOv5, the CSPNet is composed of three ConvModule and n numbers of DarknetBottleneck, the details of the network is shown in the Figure2.3. The feature map is divided into two parts, one part path through a ConvModule, another part path through a ConModule and n numbers of DarknetBottleneck, the computed features of each path are concatenated and followed with a ConvModule.

     In YOLOv5, a SPPBottleNeck module is applied at the end of the stage layer. The design of the SPPBottleNeck follows the idea of SPPNet which aims to generate a fixed-sized feature map regardless of input image size. Specifically, the input image is divided into sub-regions of different sizes, then the max pooling operations are applied on each sub-region. During

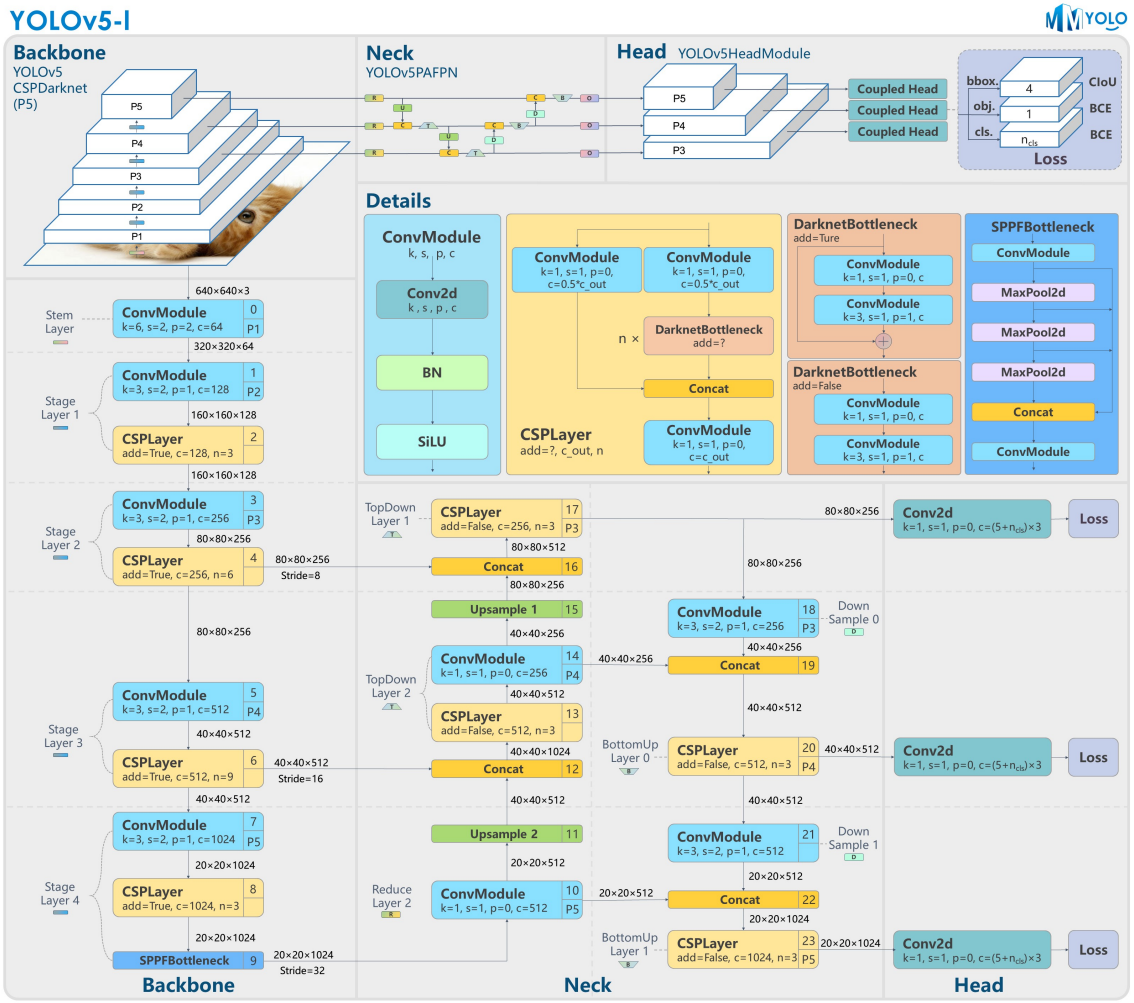


Figure 2.3: The architecture of the YOLOv5

the operations, the number of sub-region will be fixed, thus no matter how the size of input images changes the pooling result within each sub-region is fixed and the size of the final output will also be fixed.

- **Neck**

  The neck is responsible for integrating features from different scales. Path Aggregation Network(PANet)[26] is used in YOLOv5. In object detection networks, the features from low layers usually keep a higher resolution and more accurate localization signals than features from high layers. While, the features from high layers contain richer semantic information. PANet applies a bottom-up path augmentation to integrate the low level features to the high level features which enhance the entire feature hierarchy with accurate localization signals.

  In YOLOv5, the backbone and the neck altogether follow the structure in PANet. It integrates the semantic information from high level with features from low level in three different scales. Specifically, the feature map from the Stage Layer2 is integrated to the TopDown layer1, the feature map from the Stage Layer3 is integrated to the TopDown layer2 and the feature map from the Stage Layer4 is integrated to the Reduce layer2. The feature maps produced by the neck are the same as the backbone, each with size of $(80*80*256), (40*40*512), (20*20*1024)$.

- **Head**

  Three heads are used in YOLOv5, each predicts a different feature scale from the neck, whose shapes are (80∗80∗256), (40∗40∗512), (20∗20∗1024).

  In YOLOv5, each output layer has the different number of grid cells, with $80*80$, $40*40$, and $20*20$ separately, each grid cell predicts 3 bounding boxes. Each head produces bounding boxes, class probabilities and confidence scores, thus the output shapes of the head module are $(3*(4+1+C), 80, 80)$, $(3*(4+1+C), 40, 40)$ and $(3*(4+1+C), 20, 20)$, where C denotes the number of classes.

3. **Loss function** The loss function contains three parts, classification loss, objectness loss and localization loss.

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \tag{2.1}$$

$$L_{cls} = \sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in classes} [\hat{p}_i(c) log(p_i(c)) + (1 - \hat{p}_i(c)) log(1 - p_i(c))] \tag{2.2}$$

$$L_{obj} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} [\hat{C}_i log(C_i) + (1 - \hat{C}_i) log(1 - C_i)] \tag{2.3}$$

$$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{noobj} [\hat{C}_i log(C_i) + (1 - \hat{C}_i) log(1 - C_i)] \tag{2.4}$$

$$L_{loc} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{2.5}$$

where,

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2 \tag{2.6}$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{2.7}$$

The classification loss (2.2) measures the error between the predicted class probabilities $\hat{p}$ and the ground truth p, it is computed by the binary cross entropy loss.

The objectness loss (2.3) measures the difference of the predicted confidence score $\hat{C}$ and the ground truth C of the objects presented in the bounding box, it is computed by the binary cross entropy loss.

YOLOv5 uses the Complete Intersection over Union(CIoU) for the location loss(2.5). In equation (2.5), $\rho^2(b, b^{gt})$ represents the Euclidean distance between the center points b and b^{gt} of the bounding boxes, c is the diagonal length of the enclosing box covers both bounding boxes and v represents the difference of the aspect ratio between bounding boxes. Compared to the location purely applied to the Intersection over Union(IoU), CIoU incorporates the distance between the center points and the aspect ratio of the bounding boxes which is more comprehensive.

2.2.3.2 Yolov8

YOLOv8 is the latest version of the YOLO object detection model. The architecture is similar to the previous versions like YOLOv5, but it introduces many small improvements compared to the previous versions. In this part, we only show the improvement in the network aspect.

1. **Network** The overview of the architecture is shown in Figure2.4 below.

   - **Backbone** One of the biggest differences of the backbone compared to YOLOv5 is the CSPLayer module. The CSPLayer in YOLOv5 splits the feature map into two parts. In YOLOv8, the feature map is split into several parts with more skip connections. Specifically, the feature map passes through a ConvModule with kernel size of 1 and then applies the split operations with n number of Bottleneck modules following, the newly produced feature map will be concatenated with the residual and pass through a ConvModule. By concatenating features from different layers, the model will capture more diverse and richer feature representations. Meanwhile, the computation can be further decreased.
   - **Head** Another big change in YOLOv8 compared to YOLOv5 is the head part. In YOLOv8, a decoupled head is used which handles the classification and localization tasks separately. YOLOv8 also uses an anchor free

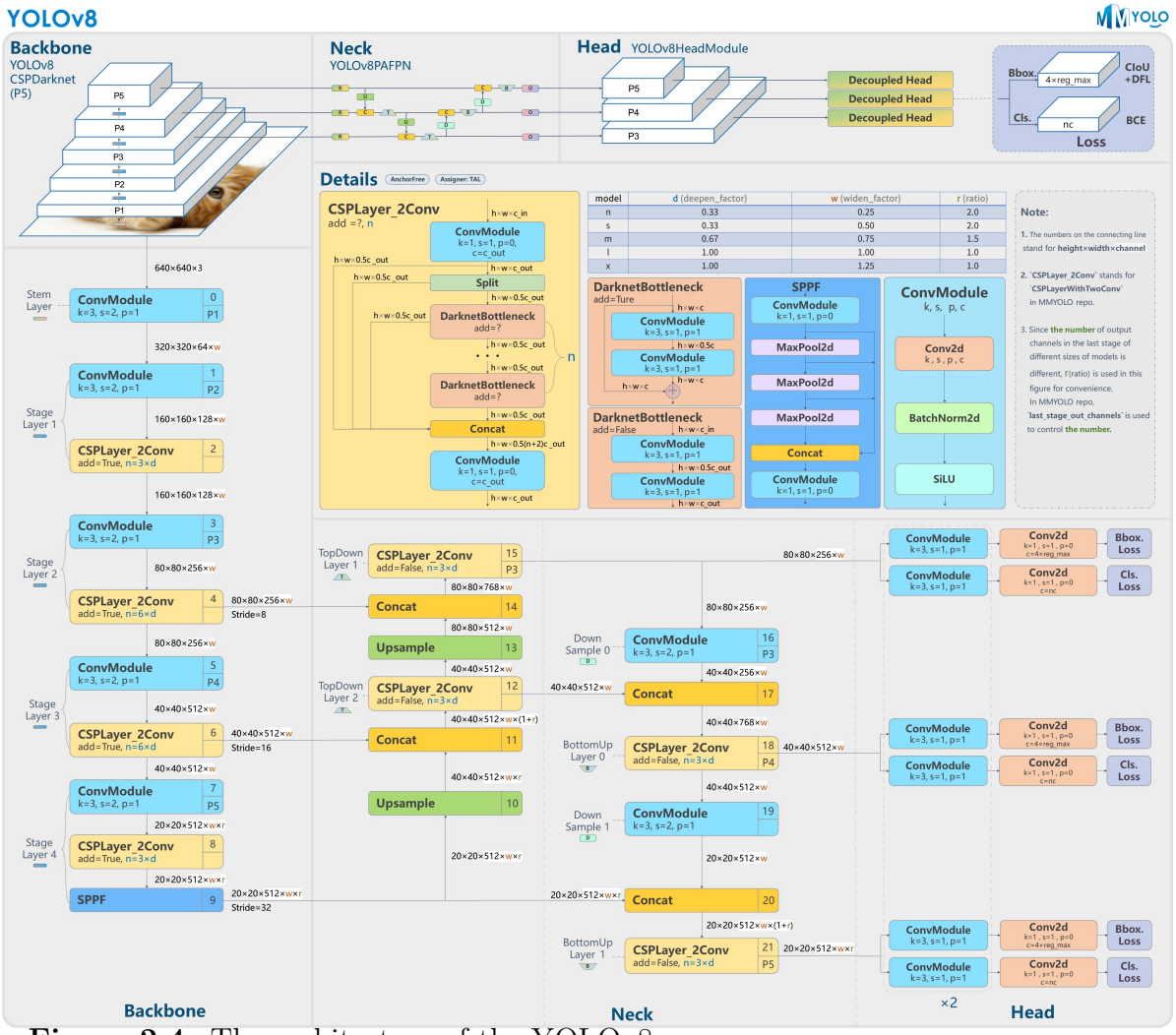


Figure 2.4: The architecture of the YOLOv8

detection mechanism which predicts the center of an object directly. By using the decoupled head and the anchor free detection mechanism, both the accuracy and the speed are improved.

2.2.4 Evaluation metrics

The evaluation metrics mainly involve Precision, Recall, mAP(mean Average Precision), IOU(Intersection over Union) [23, 34, 32, 33, 22].

1. Precision:
$$\text{Precision} = \frac{TP}{TP + FP},$$
where TP (True Positives) is the number of correct positive predictions, and FP (False Positives) is the number of incorrect positive predictions. That is the model

2. Recall:
$$\text{Recall} = \frac{TP}{TP + FN},$$
where FN (False Negatives) is the number of positive instances incorrectly predicted as negative.

3. mAP(mean Average Precision): mAP is the mean of the Average Precision scores for each class or across multiple IOU thresholds.

4. IOU(Intersection over Union): IOU is the ratio of intersection and union between predicted bounding boxes and real bounding boxes, used to evaluate the accuracy of bounding boxes,
$$\text{IOU} = \frac{\text{I}}{\text{U}}$$
where I means area of Intersection between predicted bounding boxes and real bounding boxes, U means area of Union between predicted bounding boxes and real bounding boxes [35, 38].

2.2.5 Related work

In object recognition, synthetic data has been used to improve the generalization ability of object detection models, it could increase the detection rate in unknown environments. In [36], the authors use images generated from video game engines to show the effectiveness of synthetic datasets in simulating the real world. To solve the occlusion problem in object recognition, synthetic data is used to simulate different degrees of occlusion, which helps the model learn to recognize objects in complex environments. In [40], adding distractors into the synthetic dataset and training on that, to improve the recognition rate of partially occluded objects in real-world scenes.

2.3 Pose estimation

2.3.1 Random Sample Consensus and Perspective-n-Point

Random sample consensus (RANSAC) is a robust estimation method for the datasets with a lot of outliers. In pose estimation, RANSAC fits the model by randomly selecting a subset of the dataset, then computes the consistency of this model with all other data points. The more consistent the number of data points, the more reliable the model is considered. This process is repeated multiple times to select the best model,

$$\hat{\theta} = \arg\max_{\theta} |\{x_i : d(x_i, \theta) < T\}|$$

where θ is model parameters, x_i is data point, d is distance measurement.

The PnP problem means to estimate the camera pose from 3D space to the 2D image plane (already have several 3D points and their corresponding positions in the 2D image). At least three pairs of points are needed to solve this problem, and more points can provide more accurate solutions by minimizing the reprojection error [13].

$$\min_{R,t} \sum_{i} \|x_i - \text{proj}(K(RX_i + t))\|^2$$

2.3.2 3D-2D Correspondences

In order to use the PnP algorithm, it is first necessary to find the corresponding relationship between the 3D model points and the 2D image points. This usually uses computer vision technology to detect key points in an image, and then matching these key points with points in a 3D model. Deep learning methods can achieve this process by predicting the positions of key points in 2D images, which need to accurately correspond to the points in the 3D model.

2.3.3 Wide Depth Range Pose

WDR (Wide Depth Range) proposes an innovative 6D pose estimation method, used to solve the special challenges faced when observing objects from different depths in space. For example, effectively solving the solutions for different lighting and scale changes when observing objects from a long distance. Our master thesis needs to deal with small objects, so WDR is also applicable [20].

This method uses the Feature Pyramid Network, and FPN could catch features of objects with different sizes by dealing with different scale feature maps. More specifically, in pose estimation of WDR, lower-level feature maps are responsible for catching global information to determining the approximate position of objects; And higher-level feature maps are responsible for catching local information to determine the specific orientation and pose of objects; The combination of these two parts could obtain the basic idea of pose estimation [20, 25].

Secondly, FPN uses a PnP problem based on RANSAC in computer vision, this way could estimate the final pose of objects from the 3D to 2D point correspondence which is extracted from the feature maps. In WDR, using RANSAC to iteratively solve the PnP problem could find the best pose estimation from different possible solutions, make sure the accuracy and reliability of pose estimation [20].

The evaluation metrics mainly use the Average Distance of Discrepancy, it will compute the average distance between predicted 3D model points and real model points (the ground truth 3D points), and checks how many of these distances are less than or equal to $n\%$ of the model diameter. In later experiment, we use Add10, which means 10%

Assume we have $P = x_1, x_2, \ldots, x_n$ as the points in real model , $\hat{p} = \hat{x_1}, \hat{x_2}, \ldots, \hat{x_n}$ as the points in predictive pose of real model, then ADD could computed by

$$d_i = ||P(x_i) - P(\hat{x}_i)||.$$

Assume the model diameter is D, then check how many distances d_i less than or equal to $D \cdot 10\%$. Accumulate the amount of qualified points m, and $Add10 = \frac{m}{n}$. If Add10 closes to 1, this means almost all predicted points are very accurate, with errors within 10% of the model diameter [27][20].

2.3.4 Related work

In pose estimation, synthetic dataset provides an effective and fast iterative method for training deep learning models. In[39], show how to train object detection and 6D pose estimation models using fully computer-generated images, which could accurately estimate the pose of objects in complex real-world scenarios. In [44], the author makes a combination of synthetic data and real data to achieve accurate pose estimation with quaternion and convolutional neural networks.

3
Methods

In this section, we will introduce the steps and pipeline for making synthetic dataset. Then we will introduce the pipeline and methods used for object detection and pose estimation training on the synthetic dataset.

3.1 Synthetic Datasets

In section 3.1, we will explain the steps for making synthetic dataset basic on the theory of synthetic dataset and blender which we showed in section2.

3.1.1 Pipeline

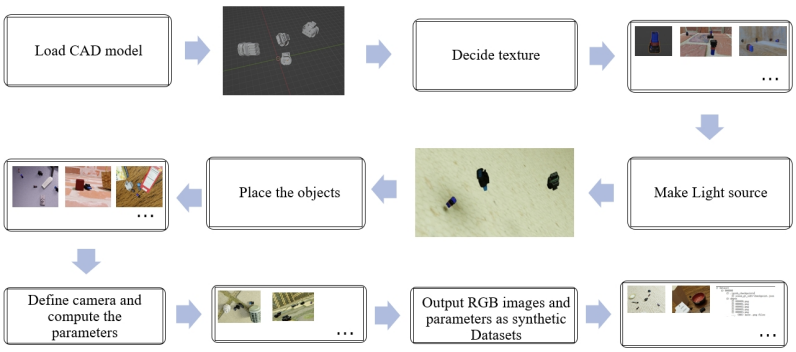


Figure 3.1: The pipeline for synthetic dataset

Based on the blender and blenderproc, Figure 3.1 shows that it is easy to import the CAD model and output parameters and images, our steps for making synthetic dataset will also follow the Figure 3.1.

3.1.2 Load the CAD models

CAD models usually come in many forms, and blender commonly use ***.ply** and ***.obj** formats when loading objects. Our objects are automotive wire harness connectors, so the best way to find the suit CAD models could be downloaded from this website *https* : *//www.te.com/en/home.html*, there are many CAD models of small parts or connectors in this website.

Figure 3.2 shows the connectors from industry. In Table 3.1, we choose 19 connectors and define the object class.

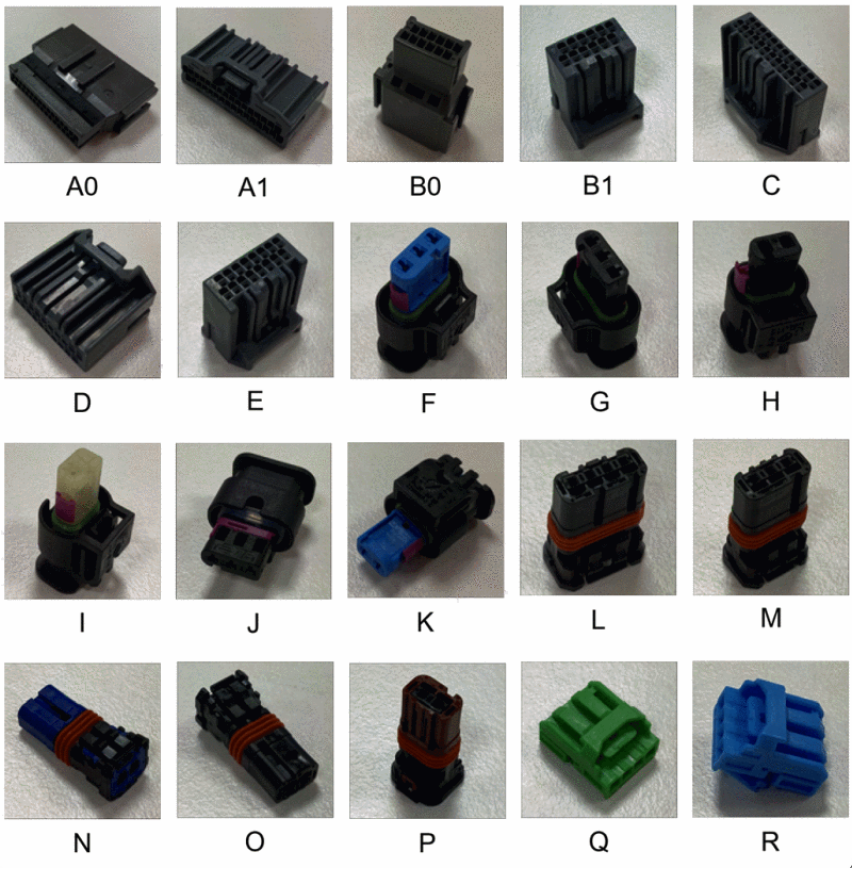


Figure 3.2: The twenty types of connectors from industry. The class of each connector is simplified and labeled below images.[43]

After download from website and choose the ***.ply** format [1], we could to import these ***.ply** file in blender, and it easy to change the Location, Rotation and Scale to adopt all the objects. Figure 3.3 shows the clean dataset that only contain some objects we are interested, these are some connectors in the cars which we mentioned before; Figure 3.6 shows some connectors with some other objects, that could be the distractors dataset.

[1]If some objects on the website may be not have the common format in blender such as ***.stp** or ***.igs**, it could easy to convert ***.ply** or ***.obj** by some software like CAD Assistant or some online features, it could be done very earlier

Object Class	Object letter
1	A1
2	B0
3	B1
4	C
5	D
6	E
7	F
8	G
9	H
10	I
11	J
12	K
13	L
14	M
15	N
16	O
17	P
18	Q
19	R

Table 3.1: Object class corresponding to object letters

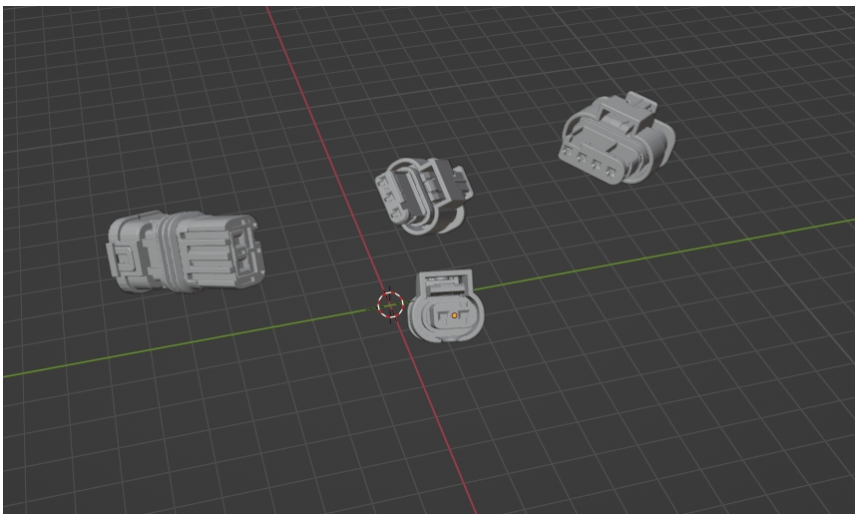


Figure 3.3: Only contain the connectors

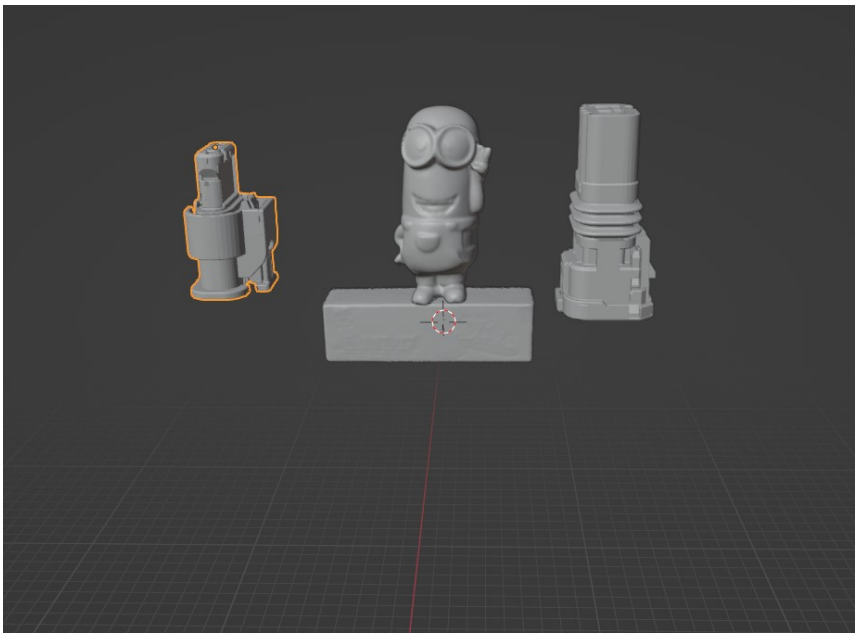


Figure 3.4: Contain the connectors with some with distractors

3.1.3 Decide texture

In order to make the dataset more realistic, it is necessary to apply some coloring on the surface of the object, rather than a single gray color. We could add the texture in blender, which could color on objects.

Usually in Blender, specific images could be used to get textures on CAD model, such as astronomical objects like the sun or earth; But the objects we interested are special connectors in the car, it difficult to create such texture images for these no-rule objects, so we could also color it ourselves by comparing it with the reference image. Figure 3.5 shows two connectors with texture; Figure 3.6 shows a part of the distractors dataset with texture.

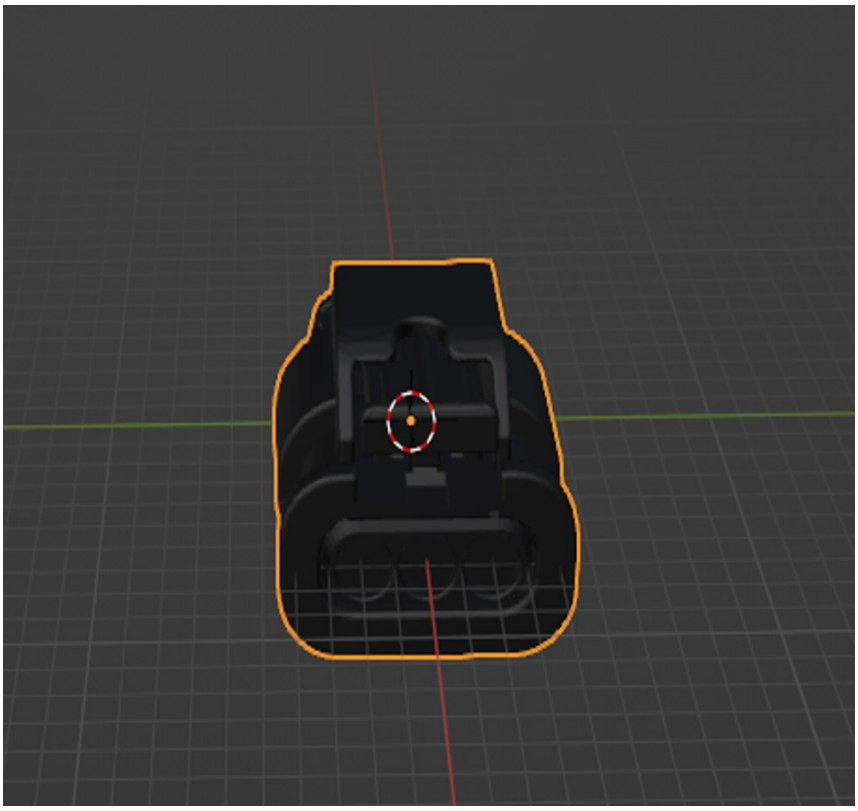


Figure 3.5: Texture in clean dataset

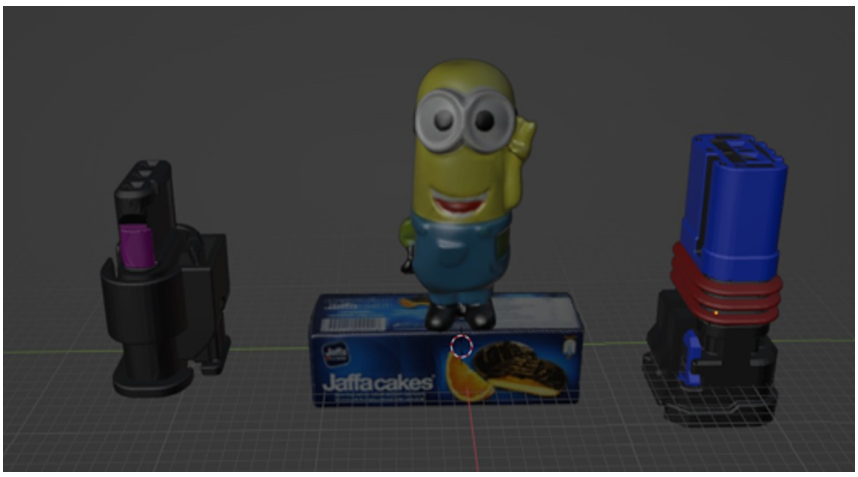


Figure 3.6: Texture in distractors dataset

Also, in order to get the better effect dataset, it is better to make the environment texture, here we create the cube box, and add the environment texture. If use the blenderproc, the environment texture is already included, and based on the randomness of blenderproc which we mentioned in section2, so it is easy to add and replace the required texture. First, read the directory where the background texture images are stored, then obtain random numbers through the numpy library [2], and

[2]NumPy is a library for the Python, we could use *random.choice*() to get random number

finally import the corresponding texture images based on the random numbers. When dealing with objects and backgrounds, first render a transparent background so that the next step is to overlay the rendering on a random background image, and then paste the object to be rendered onto the random background image. Figure 3.7 and 3.8 show two examples on different environment textures.

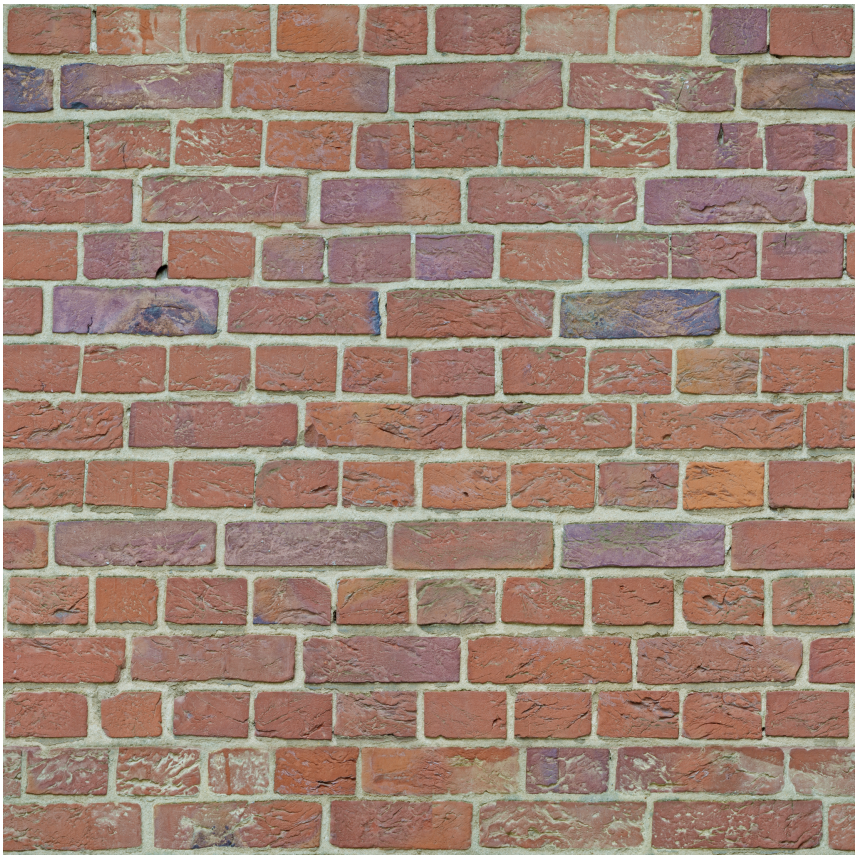


Figure 3.7: Environment texture1

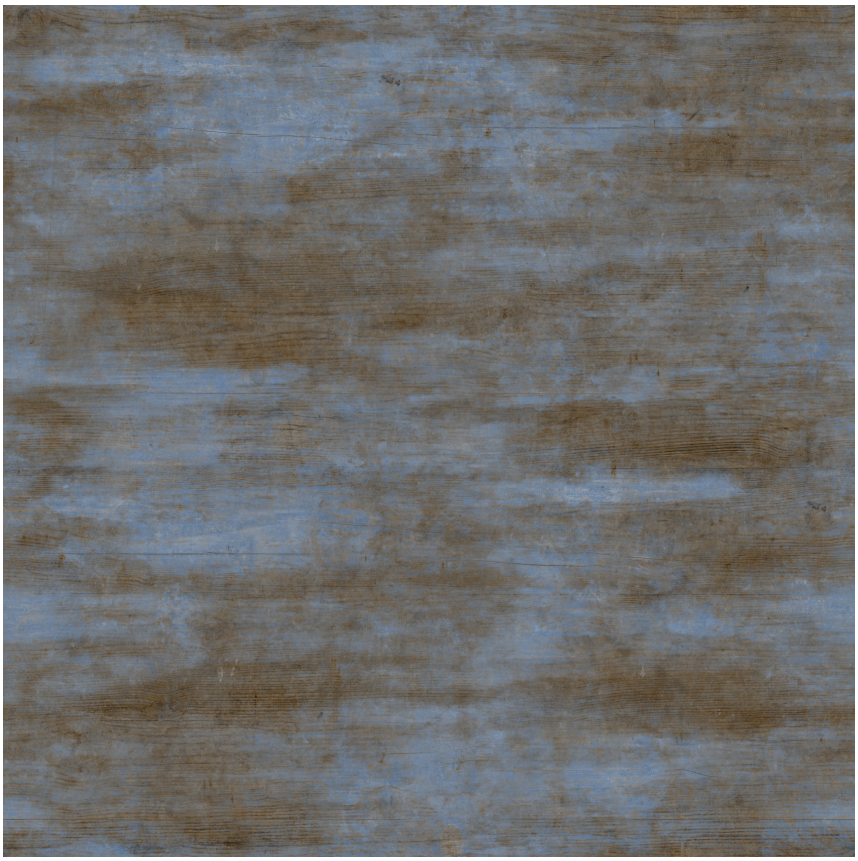


Figure 3.8: Environment texture2

3.1.4 Light source

The objects that we are interested in are connectors from cars, so we need to consider the lighting environment for factory assembly, in fact, there will be many single point lighting lights present indoors.

Basically from blenderproc and factory assembly, point light is a goode choice, because it only produces a moderate amount of shadows, not too much to make the object difficult to see, nor too little to make the surface of the object too reflective, and can make the model clearer and closer to reality, more important is the effect will be more adopt with the lighting environment for factory assembly.

BlenderProc could randomize the parameters of the scene, this is a type of domain randomization. In terms of lighting, blenderproc will give random choices on lighting position, lighting color and lighting power.

For lighting color and lighting power, almost the same as getting background texture random, basically use the numpy library to get the random number and give this random number to the corresponding lighting function in blenderproc. When focusing on lighting position, it also needs to consider the camera position, because the objects inside the lens could be illuminated only when the camera and lighting are combined. Blenderproc provides that the light is sampled in relation to the sampled camera poses. This light is outside the camera lens but has to intersect with it, to make sure that the objects inside the camera lens are illuminated. Figure

3.9 shows how objects look under the area light.

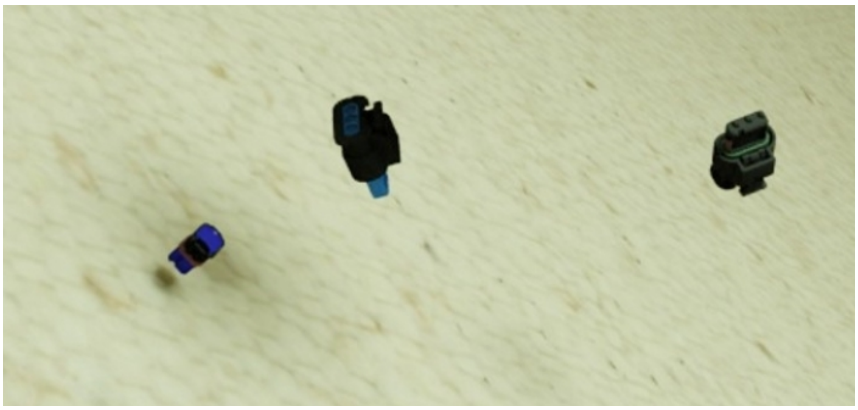


Figure 3.9: Example of objects under area light

3.1.5 Define the camera

When making the camera, we need to make sure that each object will be inside the lens, because the objects will be placed in the large cube, then we could create a camera that always faces the center (0,0), adjust the focal length, and obtain random positions for taking photos. Another idea is to have the camera move in a circular motion around the inscribed circle of the cube, always keeping it facing (0,0) during the motion. The advantage of this is that it can control the rotation of the camera, the interval between taking photos, and the rotation angle of the photography track.

When we use the blenderproc, its built-in function could determine whether a given 3D point is within the camera's field of view. We could make sure that the key points in the scene which we want are within the camera's visible range. We choose three objects to compute average coordinates of all 3D points coordinates in each object. Then we adjust camera information to ensure the camera lens will face three average positions. Figure 3.10 shows the camera parameters.


```
"cx": 320.0000000074506,
"cy": 240.00000000745058,
"depth_scale": 0.1,
"fx": 760.0,
"fy": 760.0,
"height": 480,
"width": 640
```


Figure 3.10: Camera information

3.1.6 Place the objects

It is valuable to think about how to place objects more reasonably. In figure 3.9, we could see some objects are floating in the air, this cannot happen in reality without external forces (such as hanging with thin wires). Therefore, we need to make the placement of objects more reasonable.

We could manually set the Location and Rotation, but this will need time and human working, especially when the amount of objects increases, using manual settings is not a wise choice.

It is a good idea to use the built-in physical interactions in blenderproc which we mentioned in section 2, it could simulate the physical processes. Based on Blender's steps and Blendeproc's ideas, we finally decided to establish a physical collision engine. First, each object will obtain the random height which also uses the numpy library to get the random number. When an object falls from a random height, collisions will occur between the landing plane and each object (each object will also have collisions). In the end, generating a random position that we can obtain for subsequent output operations. Figure 3.11 and 3.12 show some objects after using the physical collision engine basically from Blendeproc in the distractors dataset and clean dataset.

Figure 3.11: Example of clean dataset for placing objects randomly

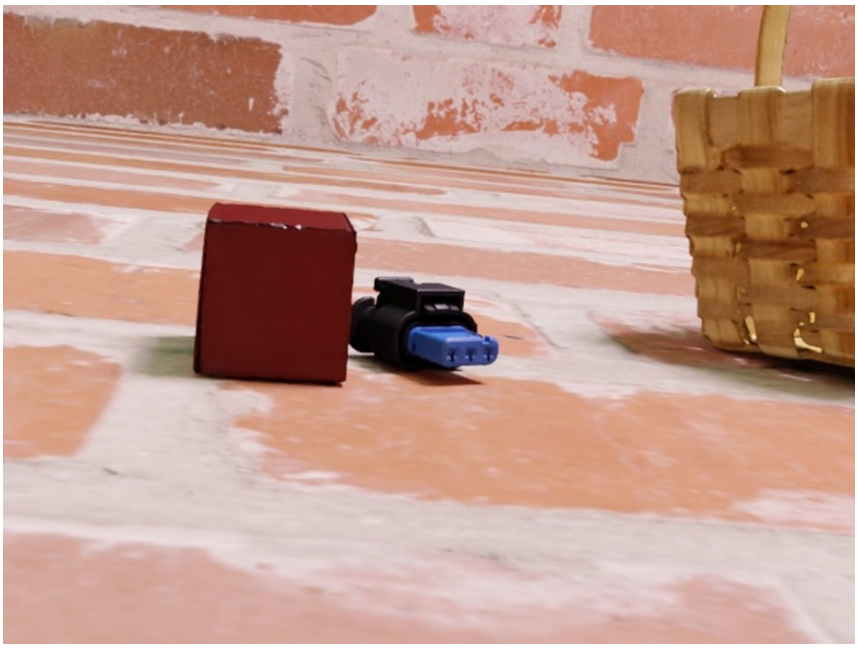


Figure 3.12: Example of distractors dataset distractors dataset for placing objects randomly

3.1.7 Compute the parameters

This part, we explain and show the meaning of the dataset, and parameters are shown in **json** file, these are very useful for object recognition and pose estimation in the later section.

1. Camera Parameters:
    - Principal point coordinates (cx, cy): useful for image correction and perspective transformation.
    - Focal lengths (fx, fy): Convert pixel coordinates into real spatial coordinates.
2. Pose Information:
    - Rotation matrix (cam_R_m2c): The object's three-dimensional rotation relative to the camera.
    - Translation vector (cam_t_m2c): The object's three-dimensional position relative to the camera.
    - Object identifier (obj_id): Objects present in the scene.
3. Object Visibility and Bounding Box Information
    - Object bounding box $(bbox\_obj)$: The object's position and size within the image.
    - Visible object bounding box $(bbox\_visib)$: The position and size of the visible parts of the object.

- Pixel counts ($px\_count\_all, px\_count\_valid, px\_count\_visib$): The amount of total, valid, and visible pixels of the object.
- Visibility fraction ($visib\_fract$): The extent of the object's visibility in the image.

4. Color and Depth Images

   - RGB images: Used for object detection, texture recognition, and color information extraction.
   - Depth images: Provide distance information between the object and the camera, could be used in the estimation of object size and reconstruction of the scene's three-dimensional structure.

5. Mask Information

   - Segmentation masks: Precise pixel-level shapes for each object.
   - Visible segmentation masks: Exact shapes of the visible parts of objects, could be used in dealing with occlusions and partially visible scenes.

3.2 Object detection and Pose estimation

3.2.1 Pipeline for object detection and pose estimation

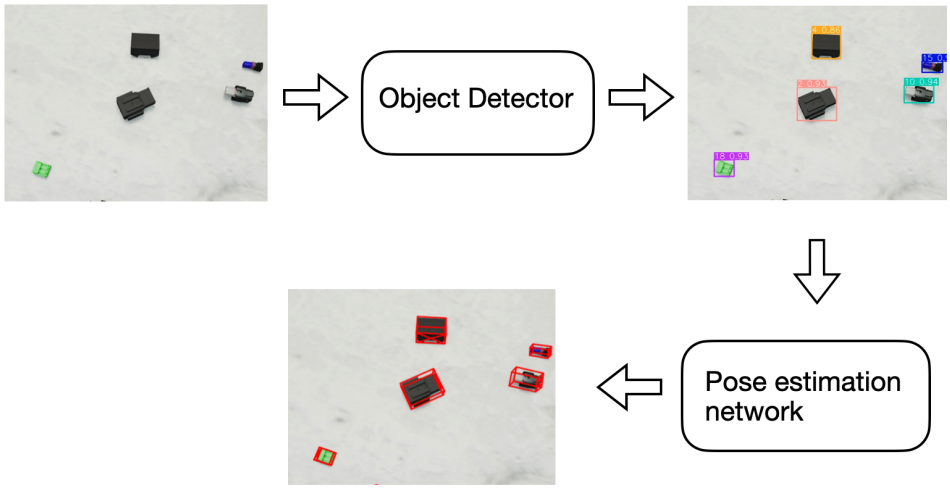


Figure 3.13: The pipeline for object detection and pose estimation

We apply a modular pipeline for object detection and pose estimation, the pipeline is shown in the Figure 3.13. There are two modules in this pipeline: a module for object detection and a module for pose estimation. For a given image, it will first go through the object detection module and output the bounding boxes and the labels of the objects. Then, the bounding boxes, the labels and the image will be sent to

the pose estimation network. The pose estimation network will estimate the object in the bounding boxes and output the corresponding poses.

There are two advantages of applying this modular pipeline. The first advantage is: it is flexible to change the methods in different modules. We use YOLOv5 and YOLOv8 in the object detection module and WDRNet in the pose estimation module. The second advantage is: we can apply more comprehensive and various experiments on the synthetic datasets. For example, we can examine how the change of the illuminance of the dataset influences the performance of detectors and pose estimators respectively.

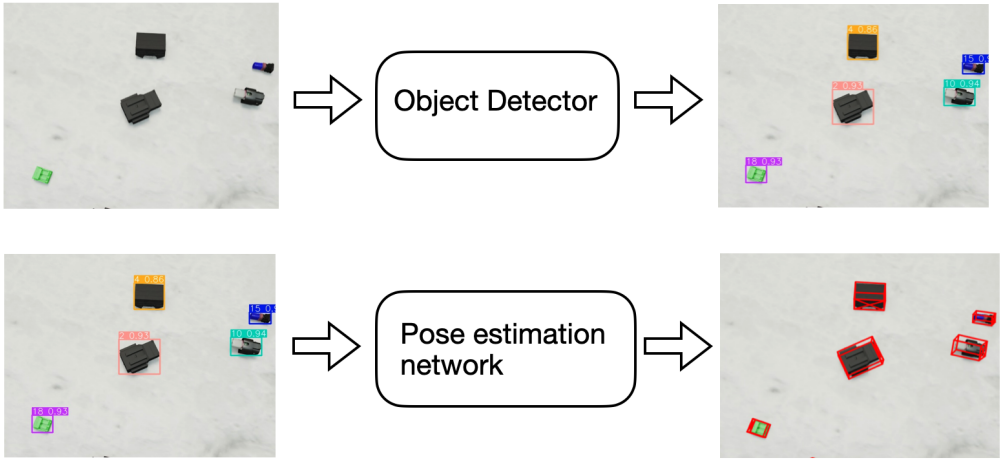


Figure 3.14: The procedure for training object detection and pose estimation

Since we apply the modular pipeline, we can train each module separately. The procedure for training is shown in the Figure 3.14. When we train the object detector we will send the images to the detector and the detector will output the bounding boxes of the objects. For pose estimators, we will send the images and the bounding boxes and output the estimated poses.

3.2.2 Object detection

Specifically, we use YOLOv5 and YOLOv8 as object detectors and we train the detectors on the Clean Dataset and Distract Dataset separately. Then, the well trained detectors will be tested on two types of test images: real and synthetic.

3.2.3 Training

In the work of Yolo, the network trained on the clean dataset and Distract Dataset. We use $10k$ images to train the model, also use some parameters from synthetic dataset.

3.2.4 Testing

The network after training basically uses the four different metrics that we mentioned in section 2: Precision, Recall, Mean Average Precision (mAP), Intersection over Union (IOU). We mainly show and discuss the mAP in the later section.

3.2.5 Pose estimation

In pose estimation, we choose WDR (Wide Depth Range) as pose estimation model and we train this model on the clean Dataset and distract Dataset separately. Then the same, this model will be tested on two types of test images: real and synthetic.

3.2.6 Training

In the work of WDR, the network also trained on the clean dataset and Distract Dataset. We use $10k$ images to train the pose estimation model, also use some parameters from synthetic dataset.

3.2.7 Testing

In pose estimation part, we test and compare clean dataset and distract Dataset on two networks which trained on clean dataset and distract Dataset. The network after training basically uses the metrics that we mentioned in section 2: Add10.

4

Results and Discussion

4.1 Data generation results

A various domain randomization methods are used for generating synthetic data, such as randomization of positions of objects, textures of backgrounds, color and positions of the lights and position and orientation of the camera. Adding distractors into the synthetic data is also an effective domain randomization method[40]. In order to examine how the adding of distractors influences the performance of the detectors and pose estimators, we applied the same settings(positions of objects, lighting conditions, textures of backgrounds etc...) to create two synthetic datasets: one clean dataset which only contain the objects we are interested and a datasets with distractors. The dataset contains 19 connectors from the industry and each dataset contains $10k$ training images, $2k$ validation images and $2k$ testing images. Examples of pictures from the Clean Dataset and the Distract Dataset are shown in the Figure 4.1 and 4.2 separately.

Figure 4.1: Examples of pictures from the clean dataset.

Figure 4.2: Examples of pictures from the dataset with distractors.

The datasets are generated for both object detection and pose estimation tasks, it contains RGB images along with depth images, segmentation images, 3D poses, 2D/3D bounding boxes, and also occlusion conditions. Some visualizations of the 2D bounding boxes and segmentation images are shown in the Figure 4.3. The visualizations of the 3D bounding boxes and the coordinates of the poses are shown in the Figure 4.4.

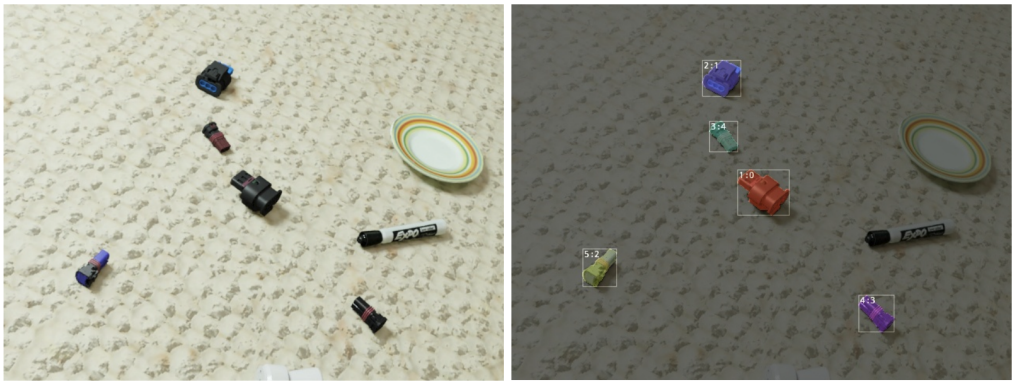


Figure 4.3: The left image shows the RGB image, the right image shows its segmentation images along with 2D bounding boxes.

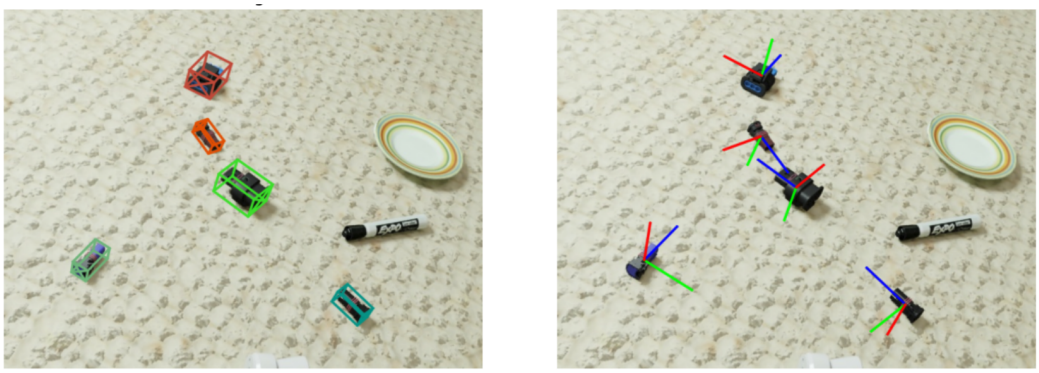


Figure 4.4: The left image shows the 3D bounding boxes, the right image shows the coordinates of the poses.

4.1.1 Clean Dataset

Figure 4.5 and 4.6 show the azimuth, elevation and distance to the camera and classes distribution of the 19 objects in the Clean Dataset. The distribution of the azimuth is concentrated at 90 degrees and 270 degrees due to the fact that after falling, the most of the objects lie on either the front or back side. The mean distances between objects and camera is around 450 cm, the minimum and maximum distances extend slightly beyond the intended range of 250 cm and 600 cm which are mainly caused by the roll or slide after falling.

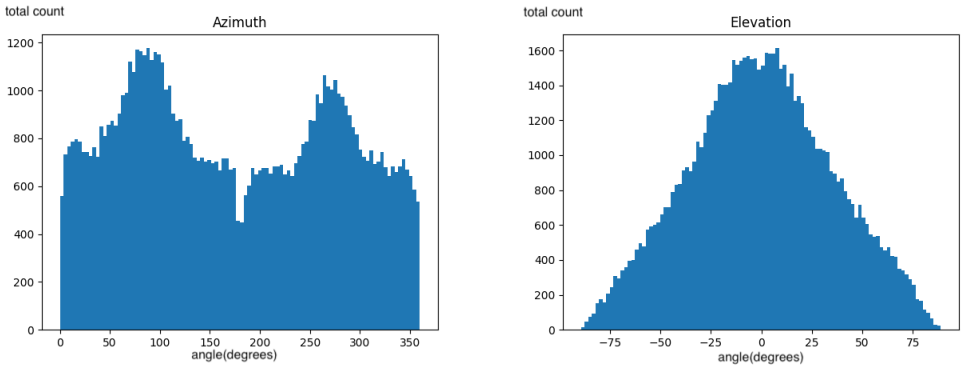


Figure 4.5: Distribution of azimuth and elevation of the objects in the Clean training dataset.

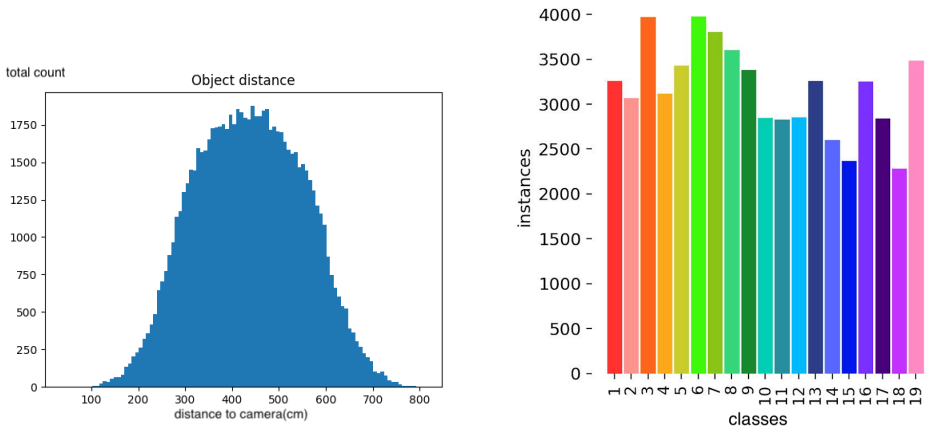


Figure 4.6: Distribution of object distances to the camera and the distribution of classes in the Clean training dataset.

4.1.2 Distract Dataset

Figure 4.7 and 4.8 show the azimuth, elevation and distance to the camera and classes distribution of the 19 objects in the Distract Dataset. In order to examine the influences from adding the distractors into the dataset, we need to make sure the distribution of the Clean Dataset and Distract Dataset are close to each other except for the distractors. As we can see, the distribution of azimuth, elevation and

distance to the camera are close to the Clean Dataset. Most of the classes from Distract Dataset and Clean Dataset are between 2000 and 3000, while the classes of 3, 6, 7 and 8 have more than 3500 classes. The classes are slightly unbalanced which need to be considered when analyzing the results.

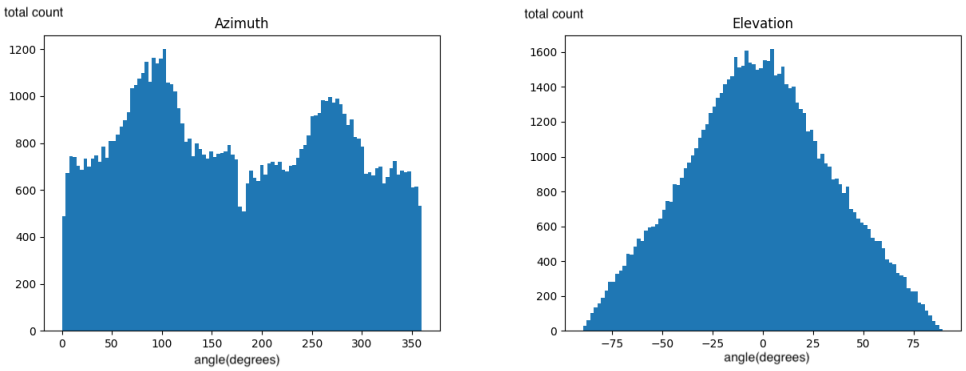


Figure 4.7: Distribution of azimuth and elevation of the objects in the Distract training dataset.

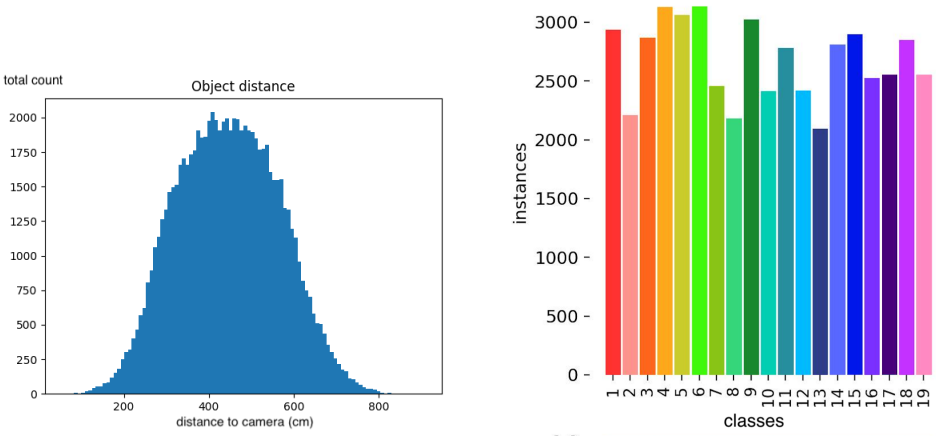


Figure 4.8: Distribution of object distances to the camera and the distribution of classes in the Distract training dataset.

4.2 Object detection experiment

We follow the procedure for training object detectors mentioned in the previous section and conduct several experiments to evaluate the effects of training with synthetic data. We train the detectors on Clean Dataset and Distract Dataset separately and test them on Clean test images, Distract test images and the real images. The mean Average Precision(mAP) of the YOLO models are shown in Table 4.1 and Table 4.2. In general, YOLOv8[23] shows better performance than YOLOv5[22] on both Clean and Distract test images, while YOLOv5 performs better on the real images.

mAP_{50}	on Clean test images	on Distract test images	on real images
YOLOv5	0.746	0.619	**0.574**
YOLOv8	**0.77**	**0.678**	0.506

Table 4.1: Detection Results of YOLO trained on the Clean Dataset

mAP_{50}	on Clean test images	on Distract test images	on real images
YOLOv5	0.71	0.675	**0.585**
YOLOv8	**0.742**	**0.745**	0.544

Table 4.2: Detection Results of YOLO trained on the Distract Dataset

When trained on the Clean Dataset and Distract Dataset, then tested on the Clean test data, the mAP decreases from 0.746 to 0.71 and 0.77 to 0.742 for YOLOv5 and YOLOv8 separately. When trained on the Distract Dataset and Clean Dataset, then tested on the Distract test data, the mAP decreases from 0.675 to 0.619 and 0.745 to 0.678 for YOLOv5 and YOLOv8 separately. Both cases show that the models tend to perform worse when they are tested on the dataset with different distributions from their training datasets. Also, we can notice that when the models trained on the Distract Dataset and tested on the Clean test data the decrease of mAP is around 3 percent. However, when the models trained on the Clean Dataset and tested on the Distract Dataset the decrease of mAP is around 7 percent. This shows that the Distract dataset is more robust. Specifically, the irrelevant objects in the dataset increase the noise and complexity of the training data which enable the models to focus more on robust and universal features and ignore the unimportant features.

From Table 4.1 and Table 4.2, we can see that when the models are tested on real images, the difference between the mAP on real images and synthetic images is quite big. For example, the difference of the mAP between Distract test images and real images for YOLOv8 is around 0.2. This shows there is a big domain gap between the real and synthetic data. The source of the domain gap could be from the exterior differences of the connectors: the connectors in the synthetic data are manually colored and do not contain all of the fine details of the real ones. Also, the differences from the objects' materials may cause a different reflective property of the objects under different lighting conditions.

It is worth noting that both models trained on Distract Dataset perform better on real images than the models trained on Clean Dataset. This shows adding distractors is a sufficient way to enhance the diversity of the training dataset and can improve the model performance.

It is also interesting to see YOLOv5 performs better on the real images than YOLOv8. From the previous theory part, we know that YOLOv8 uses a more powerful CSPNet which improves[15] the flow of information between different parts of the network. The improvement of YOLOv8 makes it overfit to some complex and granular features (like the colors, materials, reflection from lighting of the objects)

from the synthetic datasets which are quite different from the real data, thus leading to a poor performance on real data.

Then we pick the confusion matrix of YOLOv5 trained on Clean Dataset to further investigate the performance of the model. From Figure 4.9 we can see that the model fails to classify most of the classes from 1 to 6, classes of 11 and 14, for example, the connector 6 has almost the same chance to be classified as connector 5. Some studies have shown that classes with similar classes will lead to a mis-classification[18][5], thus one explanation of the classification failure is: the connectors from classes 1 to 6 have similar appearances to each other compared to the rest of the labels. Specifically, these classes are textureless objects with same colors and similar shapes, thus it is easy to mis-classify them.

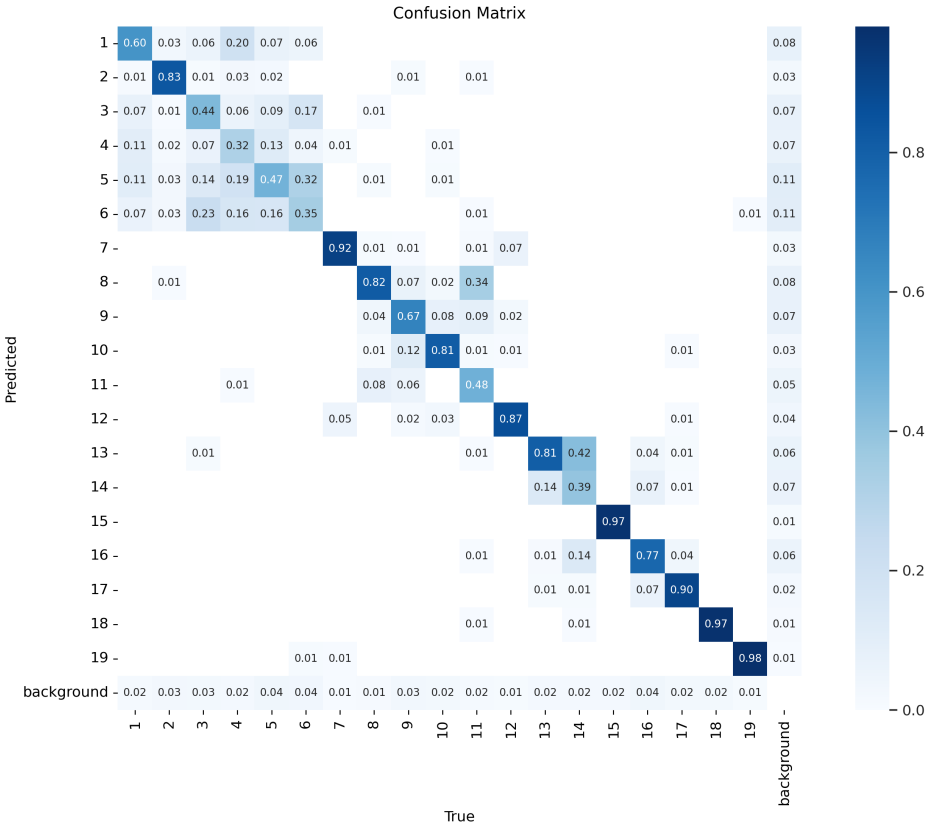


Figure 4.9: Confusion matrix of YOLOv5 trained on Clean Dataset test on synthetic data.

We also check the confusion matrix of YOLOv5 tested on real data. From Figure 4.10 we can see that the model still fails to classify most of the classes from 1 to 6. Also, there are some significant changes in some classes. For example, both true positive and false positive increase for class 1 which means the precision decreases and the recall increases and the model will mis-classify class 4 as class 1 on the real data. There is another example of class 7 which the false positive rate increases a lot and it mis-classifies with class 12. One hypothesis is: the model trained on the synthetic data obtains the general features from theses classes and overfits to some

features of class 4 and 12, thus the model is able to find class 1 and 7 but fails to find class 4 and 12 on the real data.

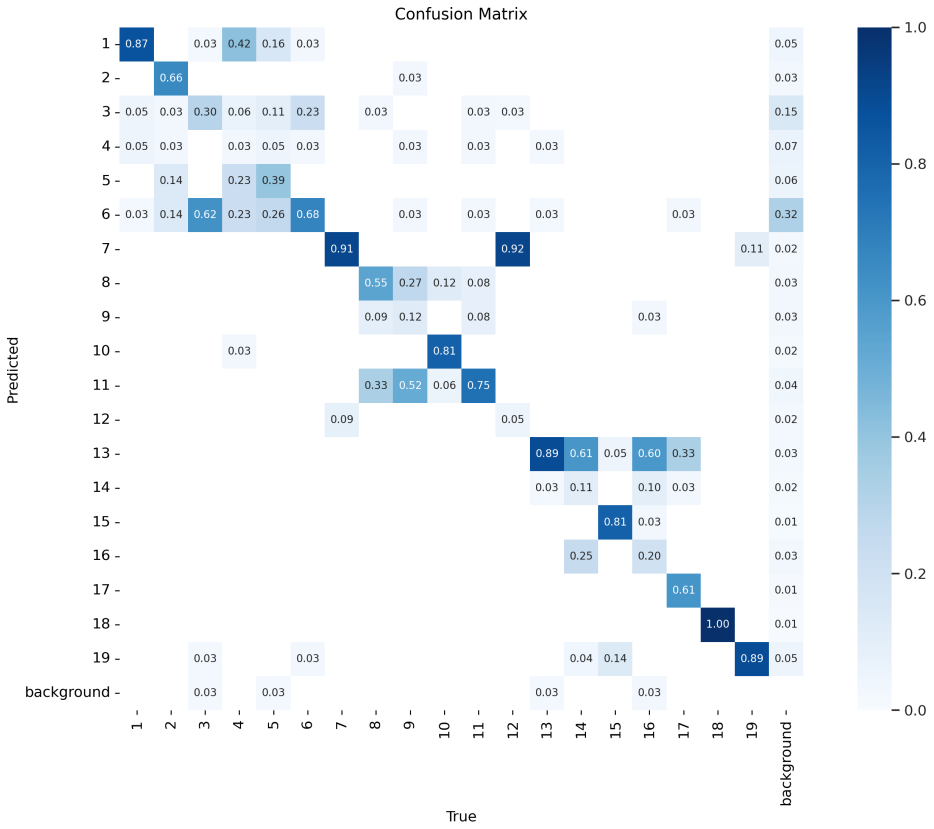


Figure 4.10: Confusion matrix of YOLOv5 trained on Clean Dataset test on real data.

Tables 4.3 and 4.4 present the quantitative test results of YOLOv5 and YOLOv8 on the synthetic and real data respectively. The precision of classes 4, 7, 8, 9, 12, 14, 16 and 17 drops more than 0.2, while the precision of classes 1, 10, 15, 18 and 19 only drop a little and still keep high. By observing the exteriors of 1, 10, 15, 18 and 19 we can observe that each of them has either a various shape or a unique texture. For example, class 18 is completely green and is quite different from others.

Class	**1**	**2**	**3**	**4**	**5**
Yolo v5	0.732	0.918	0.271	0.448	0.299
Yolo v8	0.68	0.923	0.436	0.389	0.269
Class	**6**	**7**	**8**	**9**	**10**
Yolo v5	0.361	0.96	0.841	0.768	0.87
Yolo v8	0.305	0.965	0.872	0.812	0.911
Class	**11**	**12**	**13**	**14**	**15**
Yolo v5	0.656	0.956	0.646	0.712	0.981
Yolo v8	0.76	0.955	0.738	0.807	0.983
Class	**16**	**17**	**18**	**19**	
Yolo v5	0.817	0.957	0.983	0.993	
Yolo v8	0.89	0.958	0.986	0.986	

Table 4.3: YOLOs trained on Clean Dataset and tested on the Clean test dataset

Class	**1**	**2**	**3**	**4**	**5**
Yolo v5	0.728	0.801	0.239	0.12	0.406
Yolo v8	0.7	0.317	0.239	0.0455	0.331
Class	**6**	**7**	**8**	**9**	**10**
Yolo v5	0.395	0.743	0.544	0.316	0.98
Yolo v8	0.259	0.683	0.586	0.327	0.758
Class	**11**	**12**	**13**	**14**	**15**
Yolo v5	0.612	0.309	0.534	0.389	0.979
Yolo v8	0.526	0.325	0.565	0.278	0.948
Class	**16**	**17**	**18**	**19**	
Yolo v5	0.414	0.799	0.995	0.979	
Yolo v8	0.0742	0.743	0.995	0.909	

Table 4.4: YOLOs trained on Clean Dataset and tested on real data

We also show some qualitative results of YOLOv5 test on real data. The Figure 4.11 shows the test results of YOLOv5 trained on Clean Dataset and Distract Dataset. From the figures, we can see that most of the objects can be detected. However, there are many mis-classification problems as what we have shown in the previous confusion matrix and tables.

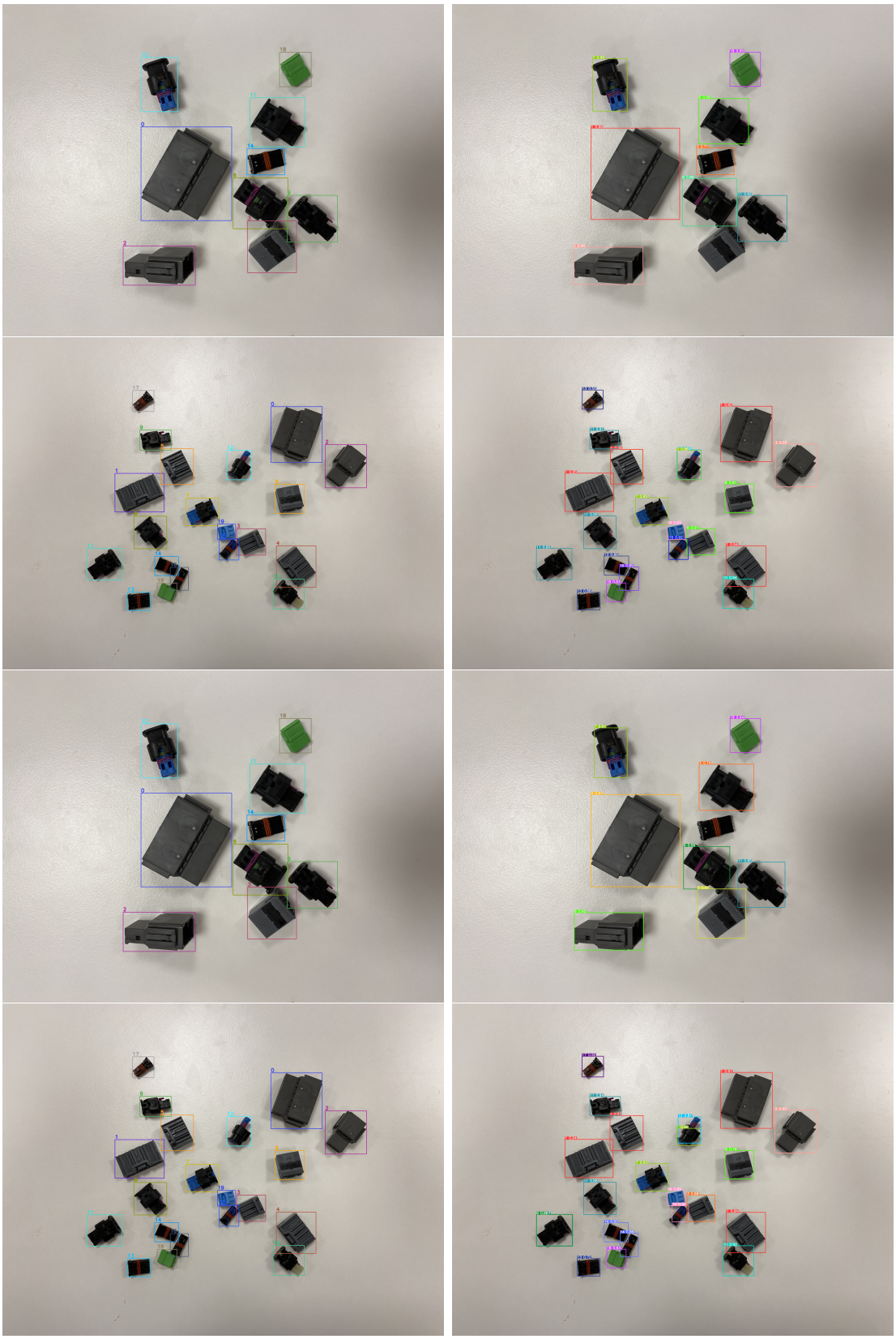


Figure 4.11: The upper four images show the result of YOLOv5 trained on Clean Dataset and tested on the real dataset. The lower four images show the result of YOLOv5 trained on Distract Dataset and tested on the real dataset. The left column shows the ground truth, the right column shows the predicted results.

Figure 4.12: The upper four images show the results of YOLOv5 trained on Clean Dataset, tested on the real dataset. The lower four images show the results of YOLOv5 trained on Distract Dataset, tested on the real dataset.

The effectiveness of adding distractors into the synthetic dataset can be shown in the Figure 4.12. The upper four images show the test results of YOLOv5 trained on Clean Dataset. As we can see, the yellow sticker in the first image is predicted as class 5, the object in the second image is captured by two bounding boxes with inaccurate positions and the object in the third image is not detected. The lower four images show the test results of YOLOv5 trained on Distract Dataset. Compared to the upper four images, the quality of localization gets improved which shows the model trained on Distract Dataset is more robust.

4.3 Pose estimation experiment

	on Distract test images	on Clean test images
Add_{10}	0.6346	0.5906

Table 4.5: Pose estimation results trained on the Clean Dataset

	on Distract test images	on Clean test images
Add_{10}	0.6016	0.5592

Table 4.6: Pose estimation results trained on the Distract Dataset

We followed the steps mentioned in section 3.2 to train a pose estimation model, we conducted some experiments to evaluate the effectiveness of using synthetic dataset for training. We trained the pose estimation models on both the clean dataset and the distract dataset, and tested them on clean test images and distract test images respectively. The Average Distance of Discrepancy (ADD) for the WDR model is shown in Tables 4.5 and 4.6. Since we don't have the ground truth of the real data for pose estimation, we can not examine the performance of the synthetic dataset quantitatively.

Table 4.7 and 4.8 show some specific results on each class. Figure 4.13 shows an example of the visualized test result on the synthetic image. Figure 4.14 and 4.15 show the visualized results of pose estimation on real data. As we can, some of the visualized 3D bounding boxes are not in the correct positions. For example, the bounding box from the first row does not locate correctly with the object and the bounding box from the second row has an obvious deviation from the object.

Class	**1**	**2**	**3**	**4**	**5**
occ	0.7561	0.5414	0.4418	0.6955	0.354
clean	0.784	0.6	0.4665	0.7375	0.3897
Class	**6**	**7**	**8**	**9**	**10**
occ	0.6973	0.4673	0.7436	0.6067	0.5339
clean	0.7268	0.521	0.759	0.633	0.5595
Class	**11**	**12**	**13**	**14**	**15**
occ	0.6186	0.5148	0.6404	0.626	0.3972
clean	0.6608	0.5676	0.649	0.6595	0.4261
Class	**16**	**17**	**18**	**19**	
occ	0.6542	0.5389	0.8042	0.7979	
clean	0.6832	0.5534	0.8241	0.8571	

Table 4.7: WDR trained on Distract Dataset tested tested on Distract and Clean test data.

Class	**1**	**2**	**3**	**4**	**5**
occ	0.6204	0.4467	0.3707	0.8666	0.4613
clean	0.6582	0.4994	0.3808	0.8746	0.4957
Class	**6**	**7**	**8**	**9**	**10**
occ	0.8374	0.4894	0.5159	0.5719	0.5258
clean	0.843	0.5745	0.5552	0.5987	0.5473
Class	**11**	**12**	**13**	**14**	**15**
occ	0.654	0.4804	0.4441	0.6618	0.3482
clean	0.6775	0.5278	0.4513	0.6782	0.3904
Class	**16**	**17**	**18**	**19**	
occ	0.5165	0.4185	0.64	0.7546	
clean	0.5436	0.4424	0.685	0.798	

Table 4.8: WDR trained on Clean Dataset tested tested on Distract and Clean test data.

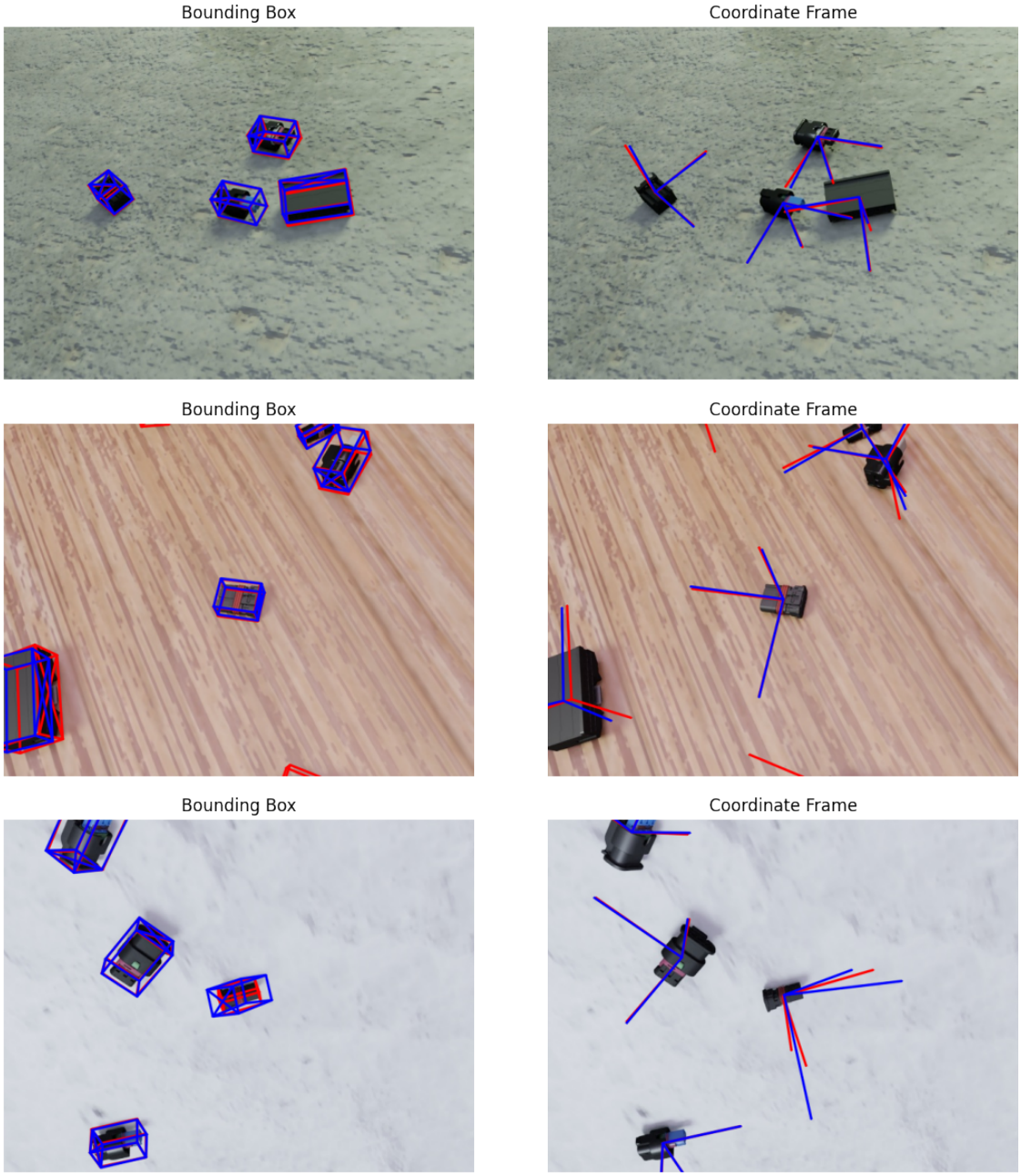


Figure 4.13: WDRNet trained on the Clean Dataset tested on the Clean Data. The red line is the predicted results and the blue line is the ground truth.

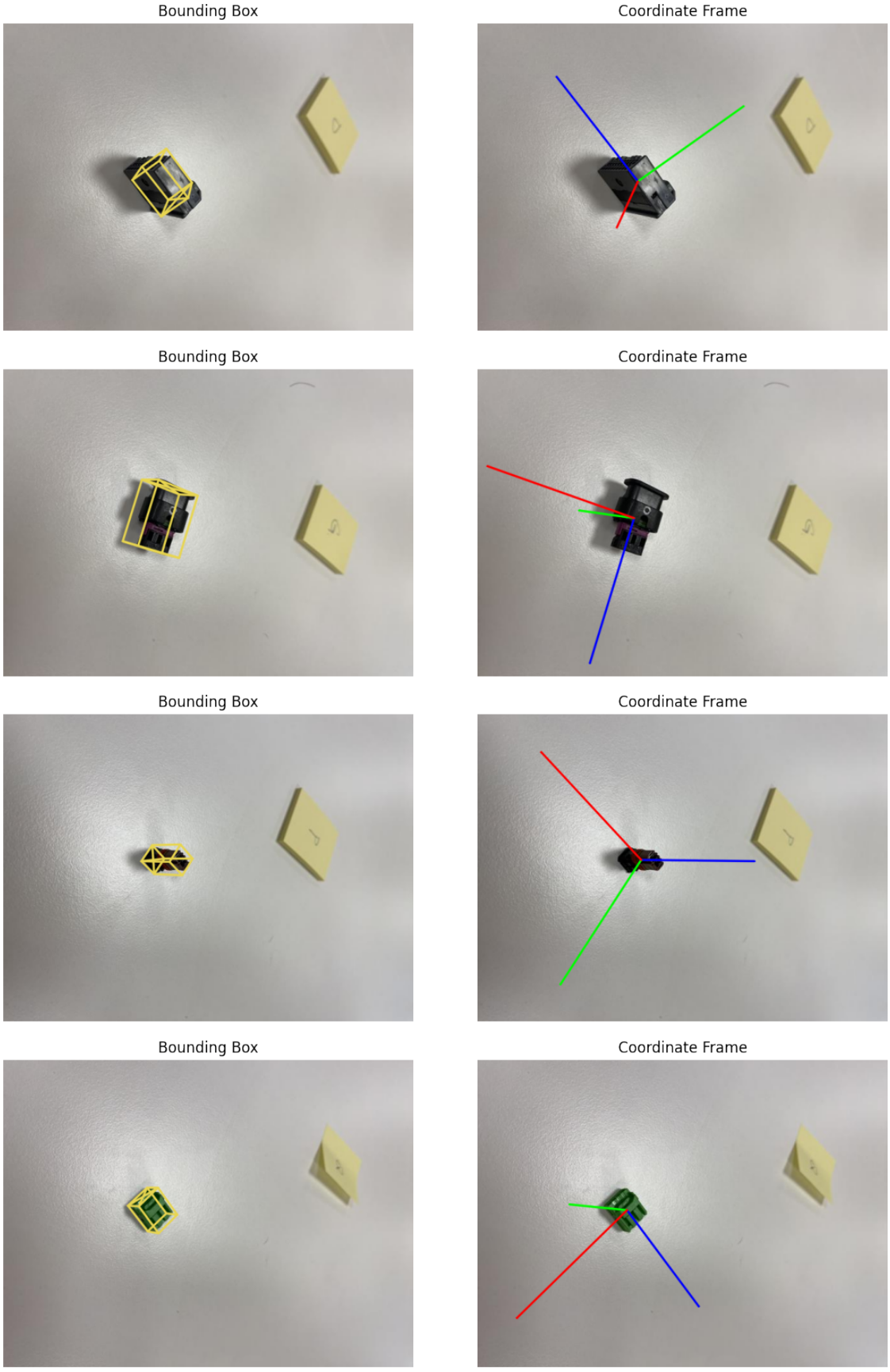


Figure 4.14: WDRNet trained on the Clean Dataset and tested on the real dataset.

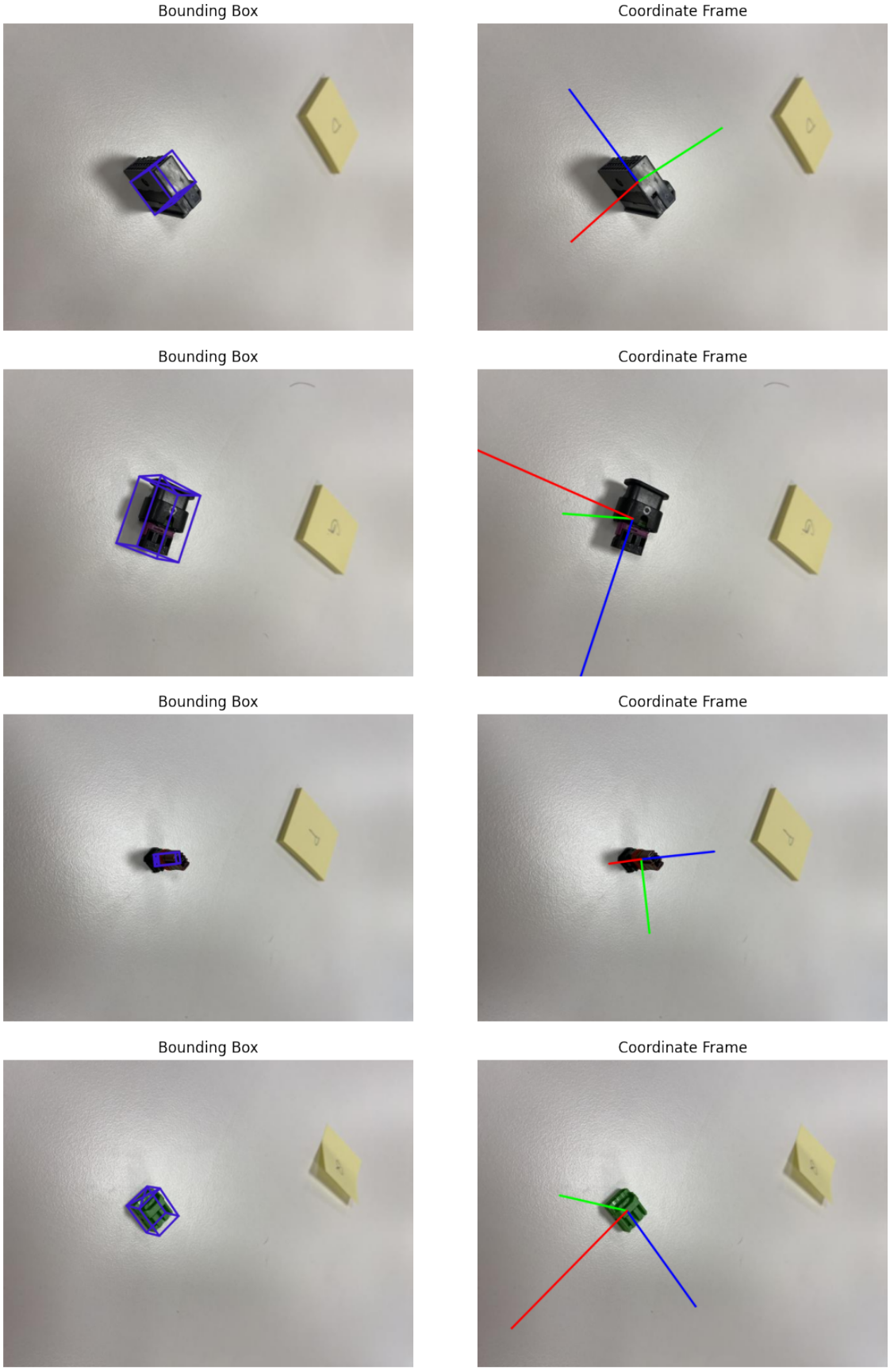


Figure 4.15: WDRNet trained on the Distract Dataset and tested on the real dataset.

4.4 Answers to the research questions

Based on the experiment above, we discussed the research questions in section 1.

1. How could synthetic dataset be generated?

   As mentioned Related work in section2, the application of synthetic dataset has become increasingly common [41, 36, 21]. Therefore, they have some different software and methods to create synthetic dataset.

   If we followed the steps in section 2, we could create the synthetic dataset easily by using blender and blenderproc. The objects that we are interested in are automotive wire harness connectors, thus the texture processing and lighting requires great care. The cycle rendering engine in blender could generate highly realistic images [2], so finally choose blender as the main software and method. [27, 24] also uses the blender to generate synthetic dataset. BlenderProc, as a programmatic pipeline for Blender, could efficiently deal with large datasets generation and could simulate physical processes to make object positions more random and flexibly define camera positions. Therefore, we also used BlenderProc [9, 10].

2. What is performance of training on synthetic dataset and testing on real dataset?

   Synthetic datasets play a crucial role in cases with limited data, but this method requires careful design to ensure the authenticity and diversity of the data. A main challenge is that models trained on synthetic datasets need to make sure of their generalization ability on real datasets.

   The results above show that although the model performs well on some synthetic dataset, it still falls if dealing with real dataset.

   We observed some misclassification issues on the real images, which may be due to the domain gap [4, 31] between synthetic dataset and real dataset.

   Because of the domain gap, the performance when training on synthetic dataset and testing on real dataset is poorer than when training and testing are done on synthetic dataset.

   The domain gap between synthetic and real dataset arises from differences in RGB images, the materials of real connectors, and different lighting conditions.

   Above results also show that testing similar objects on synthetic dataset and real images, the performance on real images is worse.

3. How does using domain randomization methods (such as adding some distractors) affect performance?

   Continuing with the previous question 2, domain gap is an issue that needs to be improved, we applied the method of adding some distractors [40] to improve the adaptability of the model to real-world cases.

The results above show that the dataset with added distractors is better than the clean dataset..

However, the improvement gained by adding some distractors is limited. Therefore, future research could add the use domain adaptation, this method could be further improved in future research, thereby enhancing the model's robustness.

5

Conclusion and future work

In this Master thesis, we provide pipelines for automatically generating annotated synthetic dataset based on clean and distract, object recognition, and pose estimation. Based on different pipelines, we introduced different input parameters to adjust the dataset and obtain better results, thereby improving the performance of the final pipeline.

In terms of evaluating model metrics, in the object recognition part, we mainly use mAP, this could measure the overall detection performance of the model on different classes and multiple confidence thresholds; We used Add10 in pose estimation part because it considers both translation and rotation of the estimated pose.

From results in section 4, by adding distractors into the synthetic dataset could increase the robustness of the detectors, so the distractor dataset has better performance than clean dataset.

In future work, also use real annotated datasets for training and testing and compare with training and testing on synthetic dataset could improve results, explore domain differences and apply domain adaptation to investigate the robustness of the dataset, thereby improving the performance of synthetic datasets and making them more realistic in simulating reality.

In order to improve the performance of the models trained on synthetic dataset, it is important to bridge the gap between the synthetic data and real data. There are different domain adaptation approaches for reducing the domain gap. One approach is applying Generative Adversarial Networks(GANs) [14], the discriminator is responsible to distinguish fake images from real images and the generator is responsible to generate realistic images to deceive the discriminator. By applying GANs, we can transfer the holistic color and texture of the source images to the target images. Another approach is applying bi-level optimization methods [1]. The objective is to find the optimal parameters for the data generator(like Blender) by minimizing the validation loss of detectors on real data and the training loss on synthetic data.

Bibliography

[1] Harkirat Singh Behl et al. *AutoSimulate: (Quickly) Learning Synthetic Data Generation*. 2020. arXiv: 2008.08424 [cs.CV].

[2] *Blender Manual*. https://docs.blender.org/manual/en/latest/getting_started/about/index.html. Accessed: 2024-05-013.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[4] *Bridging the Domain Gap for Neural Models*. https://machinelearning.apple.com/research/bridging-the-domain-gap-for-neural-models. Accessed: 2024-05-25.

[5] Danyang Cao, Zhixin Chen, and Lei Gao. "An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks". In: *Human-centric Computing and Information Sciences* 10 (2020), pp. 1–22. URL: https://api.semanticscholar.org/CorpusID:215724945.

[6] Leiyu Chen et al. "Review of Image Classification Algorithms Based on Convolutional Neural Networks". In: *Remote Sensing* 13.22 (2021). ISSN: 2072-4292. URL: https://www.mdpi.com/2072-4292/13/22/4712.

[7] Xiaoyu Chen et al. *Understanding Domain Randomization for Sim-to-real Transfer*. 2022. arXiv: 2110.03239 [cs.LG].

[8] Yuhua Chen et al. *Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach*. 2019. arXiv: 1812.05040 [cs.CV].

[9] Maximilian Denninger et al. *BlenderProc*. 2019. arXiv: 1911.01911 [cs.CV].

[10] Maximilian Denninger et al. "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering". In: *Journal of Open Source Software* 8.82 (2023), p. 4901. DOI: 10.21105/joss.04901. URL: https://doi.org/10.21105/joss.04901.

[11] Bertram Drost et al. "Model globally, match locally: Efficient and robust 3D object recognition". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 998–1005. DOI: 10.1109/CVPR.2010.5540108.

[12] Alvaro Figueira and Bruno Vaz. "Survey on Synthetic Data Generation, Evaluation Methods and GANs". In: *Mathematics* 10.15 (2022). ISSN: 2227-7390. DOI: 10.3390/math10152733. URL: https://www.mdpi.com/2227-7390/10/15/2733.

[13] M. Fischler and R. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395. URL: /brokenurl#%20http://publication.wilsonwong.me/load.php?id=233282275.

[14] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[15] Chenhao He and Pramit Saha. *Investigating YOLO Models Towards Outdoor Obstacle Detection For Visually Impaired People*. 2023. arXiv: 2312.07571 [cs.CV].

[16] Kaiming He et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 346–361. ISBN: 9783319105789. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[17] Tomas Hodan et al. *BOP: Benchmark for 6D Object Pose Estimation*. 2018. arXiv: 1808.08319 [cs.CV].

[18] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. “Diagnosing Error in Object Detectors”. In: *European Conference on Computer Vision*. 2012. URL: https://api.semanticscholar.org/CorpusID:6650709.

[19] Jiwei Hu et al. “Synthetic Data Generation Based on RDB-CycleGAN for Industrial Object Detection”. In: *Mathematics* 11.22 (2023). ISSN: 2227-7390. URL: https://www.mdpi.com/2227-7390/11/22/4588.

[20] Yinlin Hu et al. *Wide-Depth-Range 6D Object Pose Estimation in Space*. 2021. arXiv: 2104.00337 [cs.CV].

[21] Stephen James et al. *Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks*. 2019. arXiv: 1812.07252 [cs.RO].

[22] Glenn Jocher. *YOLOv5 by Ultralytics*. Version 7.0. May 2020. DOI: 10.5281/zenodo.3908559. URL: https://github.com/ultralytics/yolov5.

[23] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLO*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics.

[24] Jonas Lecerof and Torbjörn Opheim. “Generating Synthetic Data for Evaluation and Improvement of Deep 6D Pose Estimation”. MA thesis. Sweden: Chalmers University of Technology, 2019.

[25] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[26] Shu Liu et al. *Path Aggregation Network for Instance Segmentation*. 2018. arXiv: 1803.01534 [cs.CV].

[27] Tobias Löfgren and Daniel Jonsson. “Generating Synthetic Data for Evaluation and Improvement of Deep 6D Pose Estimation”. LiTH-ISY-EX–20/5339–SE. Sweden: Linköping University, 2020.

[28] Keith Man and Javaan Chahl. "A Review of Synthetic Image Data and Its Use in Computer Vision". In: *Journal of Imaging* 8.11 (2022). ISSN: 2313-433X. URL: https://www.mdpi.com/2313-433X/8/11/310.

[29] Uzair Nadeem et al. *Unconstrained Matching of 2D and 3D Descriptors for 6-DOF Pose Estimation*. 2020. arXiv: 2005.14502 [cs.CV].

[30] Filip Radenović, Giorgos Tolias, and Ondřej Chum. "Fine-Tuning CNN Image Retrieval with No Human Annotation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.7 (2019), pp. 1655–1668. DOI: 10.1109/TPAMI.2018.2846566.

[31] Ievgen Redko, Amaury Habrard, and Marc Sebban. *Theoretical Analysis of Domain Adaptation with Optimal Transport*. 2017. arXiv: 1610.04420 [stat.ML].

[32] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV].

[33] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[34] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[35] Szeliski Richard. *Computer vision algorithms and applications*. London; New York: Springer, 2011. ISBN: 9781848829343 1848829345 9781848829350 1848829353. URL: http://dx.doi.org/10.1007/978-1-84882-935-0.

[36] Stephan R. Richter et al. *Playing for Data: Ground Truth from Computer Games*. 2016. arXiv: 1608.02192 [cs.CV].

[37] Omkar Salunkhe et al. "Specifying task allocation in automotive wire harness assembly stations for Human-Robot Collaboration". In: *Computers Industrial Engineering* 184 (2023), p. 109572. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2023.109572. URL: https://www.sciencedirect.com/science/article/pii/S036083522300596X.

[38] F. Sultana, A. Sufian, and P. Dutta. "A Review of Object Detection Models Based on Convolutional Neural Network". In: *Intelligent Computing: Image Processing Based Applications*. Springer Singapore, 2020, pp. 1–16. ISBN: 9789811542886. DOI: 10.1007/978-981-15-4288-6_1. URL: http://dx.doi.org/10.1007/978-981-15-4288-6_1.

[39] Martin Sundermeyer et al. *Implicit 3D Orientation Learning for 6D Object Detection from RGB Images*. 2019. arXiv: 1902.01275 [cs.CV].

[40] Josh Tobin et al. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. 2017. arXiv: 1703.06907 [cs.RO].

[41] Gul Varol et al. "Learning from Synthetic Humans". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.492. URL: http://dx.doi.org/10.1109/CVPR.2017.492.

[42] Chien-Yao Wang et al. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. arXiv: 1911.11929 [cs.CV].

[43] Hao Wang and Björn Johansson. "Deep Learning-Based Connector Detection for Robotized Assembly of Automotive Wire Harnesses". In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. 2023, pp. 1–8. DOI: 10.1109/CASE56687.2023.10260619.

[44] Yu Xiang et al. *PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes*. 2018. arXiv: 1711.00199 [cs.CV].

[45] Jianghong Zhao et al. "The Fusion Strategy of 2D and 3D Information Based on Deep Learning: A Review". In: *Remote Sensing* 13.20 (2021). ISSN: 2072-4292. URL: https://www.mdpi.com/2072-4292/13/20/4029.

A

Appendix 1

Figure A.1 shows the clearly structure of dataset.

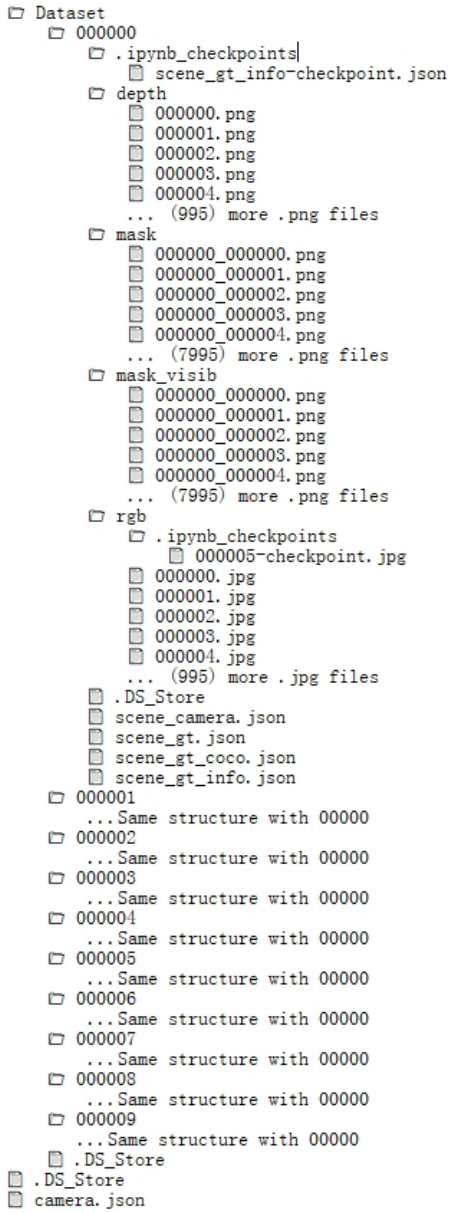


Figure A.1: Structure of dataset

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.549	0.577	0.544	0.461
1	0.327	0.947	0.741	0.689
2	0.63	0.411	0.564	0.409
3	0.212	0.297	0.223	0.189
4	0.166	0.323	0.0696	0.0637
5	0.266	0.324	0.31	0.277
6	0.223	0.419	0.194	0.167
7	0.428	0.971	0.658	0.596
8	0.622	0.273	0.386	0.336
9	0.412	0.576	0.388	0.322
10	0.91	0.812	0.882	0.683
11	0.314	0.635	0.416	0.362
12	0.729	0.289	0.479	0.402
13	0.601	0.514	0.62	0.537
14	0.282	0.786	0.347	0.3
15	0.983	0.892	0.935	0.792
16	0.505	0.167	0.304	0.247
17	0.935	0.806	0.889	0.754
18	0.965	1	0.995	0.842
19	0.919	0.814	0.93	0.797

Table A.1: Yolo v8 trained on occlusion and test by real dataset

Here provide some detailed tables of evaluation metrics for object recognition models.

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.72	0.714	0.746	0.684
1	0.709	0.609	0.732	0.698
2	0.896	0.844	0.918	0.874
3	0.331	0.462	0.271	0.254
4	0.533	0.335	0.448	0.425
5	0.273	0.499	0.299	0.279
6	0.42	0.385	0.361	0.32
7	0.868	0.919	0.96	0.916
8	0.687	0.816	0.841	0.799
9	0.727	0.682	0.768	0.718
10	0.786	0.813	0.87	0.817
11	0.703	0.484	0.656	0.608
12	0.92	0.887	0.956	0.891
13	0.492	0.859	0.646	0.578
14	0.791	0.386	0.712	0.623
15	0.991	0.965	0.981	0.878
16	0.631	0.771	0.817	0.714
17	0.931	0.907	0.957	0.858
18	0.975	0.97	0.983	0.864
19	0.962	0.976	0.993	0.882

Table A.2: Yolo v5 trained on clean and test by synthetic dataset

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.545	0.606	0.574	0.526
1	0.482	0.842	0.728	0.657
2	0.717	0.69	0.801	0.704
3	0.191	0.324	0.239	0.215
4	0.12	0.129	0.12	0.107
5	0.461	0.368	0.406	0.393
6	0.161	0.774	0.395	0.339
7	0.437	0.941	0.743	0.681
8	0.599	0.635	0.544	0.492
9	0.476	0.212	0.316	0.268
10	0.967	0.923	0.98	0.898
11	0.426	0.694	0.612	0.568
12	0.599	0.157	0.309	0.267
13	0.404	0.919	0.534	0.464
14	0.459	0.143	0.389	0.337
15	0.999	0.892	0.979	0.857
16	0.395	0.3	0.414	0.373
17	0.916	0.608	0.799	0.679
18	0.943	1	0.995	0.865
19	0.602	0.964	0.979	0.822

Table A.3: Yolo v5 trained on clean and test by real dataset

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.709	0.765	0.77	0.709
1	0.592	0.666	0.68	0.644
2	0.888	0.849	0.923	0.881
3	0.344	0.657	0.436	0.414
4	0.477	0.35	0.389	0.37
5	0.241	0.521	0.269	0.252
6	0.357	0.315	0.305	0.276
7	0.817	0.95	0.965	0.921
8	0.675	0.832	0.872	0.832
9	0.725	0.754	0.812	0.763
10	0.831	0.841	0.911	0.845
11	0.705	0.686	0.76	0.705
12	0.903	0.893	0.955	0.892
13	0.485	0.905	0.738	0.663
14	0.8	0.631	0.807	0.718
15	0.986	0.968	0.983	0.866
16	0.788	0.851	0.89	0.777
17	0.915	0.908	0.958	0.86
18	0.995	0.973	0.986	0.884
19	0.953	0.979	0.986	0.889

Table A.4: Yolo v8 trained on clean and test by synthetic dataset

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.523	0.485	0.506	0.434
1	0.377	0.868	0.7	0.647
2	0.399	0.276	0.317	0.247
3	0.331	0.189	0.239	0.218
4	0.0493	0.0323	0.0455	0.0422
5	0.29	0.526	0.331	0.315
6	0.285	0.226	0.259	0.229
7	0.418	0.912	0.683	0.619
8	0.587	0.606	0.586	0.458
9	0.682	0.0909	0.327	0.254
10	0.607	0.719	0.758	0.66
11	0.517	0.5	0.526	0.442
12	0.744	0.153	0.325	0.278
13	0.337	0.973	0.565	0.488
14	0.534	0.124	0.278	0.246
15	0.967	0.661	0.948	0.794
16	0.033	0.168	0.0742	0.0654
17	0.876	0.556	0.743	0.601
18	0.989	0.832	0.995	0.858
19	0.919	0.807	0.909	0.781

Table A.5: Yolo v8 trained on clean and test by real dataset

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.69	0.639	0.675	0.601
1	0.241	0.429	0.275	0.256
2	0.946	0.731	0.866	0.82
3	0.563	0.344	0.457	0.403
4	0.401	0.349	0.365	0.329
5	0.201	0.214	0.158	0.143
6	0.316	0.332	0.319	0.292
7	0.829	0.921	0.937	0.865
8	0.599	0.555	0.607	0.55
9	0.908	0.629	0.805	0.734
10	0.781	0.758	0.813	0.712
11	0.694	0.728	0.754	0.695
12	0.935	0.791	0.896	0.822
13	0.406	0.672	0.418	0.376
14	0.722	0.459	0.639	0.542
15	0.937	0.933	0.958	0.819
16	0.906	0.579	0.788	0.685
17	0.79	0.807	0.833	0.701
18	0.992	0.962	0.975	0.844
19	0.95	0.949	0.964	0.838

Table A.6: Yolo v5 trained on occlusion and test by synthetic dataset

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.537	0.626	0.585	0.518
1	0.426	0.895	0.828	0.774
2	0.59	0.517	0.582	0.518
3	0.276	0.432	0.339	0.306
4	0.13	0.194	0.133	0.124
5	0.311	0.474	0.366	0.333
6	0.28	0.388	0.281	0.247
7	0.415	0.853	0.698	0.652
8	0.606	0.333	0.481	0.412
9	0.377	0.633	0.393	0.352
10	0.872	0.938	0.961	0.882
11	0.403	0.639	0.458	0.42
12	0.65	0.553	0.529	0.443
13	0.546	0.595	0.624	0.562
14	0.312	0.821	0.47	0.421
15	0.897	0.811	0.935	0.786
16	0.599	0.532	0.27	0.236
17	0.909	0.836	0.906	0.795
18	0.976	1	0.995	0.849
19	0.62	0.929	0.868	0.734

Table A.7: Yolo v5 trained on occlusion and test by real dataset

Object Class	Precision	Recall	mAP50	mAP50-95
all	0.756	0.685	0.745	0.668
1	0.386	0.509	0.482	0.443
2	0.941	0.748	0.891	0.843
3	0.617	0.416	0.514	0.455
4	0.545	0.502	0.545	0.498
5	0.422	0.369	0.346	0.324
6	0.411	0.41	0.423	0.391
7	0.923	0.919	0.951	0.876
8	0.6	0.544	0.609	0.554
9	0.941	0.694	0.841	0.775
10	0.818	0.757	0.834	0.723
11	0.766	0.732	0.789	0.729
12	0.929	0.855	0.926	0.845
13	0.565	0.696	0.671	0.6
14	0.734	0.688	0.773	0.666
15	0.967	0.924	0.96	0.824
16	0.944	0.633	0.808	0.707
17	0.903	0.764	0.838	0.714
18	0.993	0.937	0.974	0.868
19	0.966	0.926	0.972	0.849

Table A.8: Yolo v8 trained on occlusion and test by synthetic dataset

Here provide some detailed tables of evaluation metrics Add10 for pose estimation models.

Class	**1**	**2**	**3**	**4**	**5**
Yolo v5	0.828	0.582	0.339	0.133	0.366
Yolo v8	0.741	0.564	0.223	0.0696	0.31
Class	**6**	**7**	**8**	**9**	**10**
Yolo v5	0.281	0.698	0.481	0.393	0.961
Yolo v8	0.194	0.658	0.386	0.388	0.882
Class	**11**	**12**	**13**	**14**	**15**
Yolo v5	0.458	0.529	0.624	0.47	0.935
Yolo v8	0.416	0.479	0.62	0.347	0.935
Class	**16**	**17**	**18**	**19**	
Yolo v5	0.27	0.906	0.995	0.868	
Yolo v8	0.304	0.889	0.995	0.93	

Table A.9: YOLOs trained on Distract Dataset and tested on real data

Class	**1**	**2**	**3**	**4**	**5**
Yolo v5	0.275	0.866	0.457	0.365	0.158
Yolo v8	0.482	0.891	0.514	0.545	0.346
Class	**6**	**7**	**8**	**9**	**10**
Yolo v5	0.319	0.937	0.607	0.805	0.813
Yolo v8	0.423	0.951	0.609	0.841	0.834
Class	**11**	**12**	**13**	**14**	**15**
Yolo v5	0.754	0.896	0.418	0.639	0.958
Yolo v8	0.789	0.926	0.671	0.773	0.96
Class	**16**	**17**	**18**	**19**	
Yolo v5	0.788	0.833	0.975	0.964	
Yolo v8	0.808	0.838	0.974	0.972	

Table A.10: YOLOs trained on Distract Dataset and tested on Distract test data

Table A.11: Yolo v5 trained on occlusion and test by synthetic dataset VS Yolo v5 trained on occlusion and test by real dataset

Class	**1**	**2**	**3**	**4**	**5**
Yolo v5 (syn)	0.275	0.866	0.457	0.365	0.158
Yolo v5 (real)	0.828	0.582	0.339	0.133	0.366
Class	**6**	**7**	**8**	**9**	**10**
Yolo v5 (syn)	0.319	0.937	0.607	0.805	0.813
Yolo v5 (real)	0.281	0.698	0.481	0.393	0.961
Class	**11**	**12**	**13**	**14**	**15**
Yolo v5 (syn)	0.754	0.896	0.418	0.639	0.958
Yolo v5 (real)	0.458	0.529	0.624	0.47	0.935
Class	**16**	**17**	**18**	**19**	
Yolo v5 (syn)	0.788	0.833	0.975	0.964	
Yolo v5 (real)	0.27	0.906	0.995	0.868	

Table A.12: Yolo v8 trained on occlusion and test by synthetic dataset VS Yolo v8 trained on occlusion and test by real dataset

Class	**1**	**2**	**3**	**4**	**5**
Yolo v8 (syn)	0.482	0.891	0.514	0.545	0.346
Yolo v8 (real)	0.741	0.564	0.223	0.0696	0.31
Class	**6**	**7**	**8**	**9**	**10**
Yolo v8 (syn)	0.423	0.951	0.609	0.841	0.834
Yolo v8 (real)	0.194	0.658	0.386	0.388	0.882
Class	**11**	**12**	**13**	**14**	**15**
Yolo v8 (syn)	0.789	0.926	0.671	0.773	0.96
Yolo v8 (real)	0.416	0.479	0.62	0.347	0.935
Class	**16**	**17**	**18**	**19**	
Yolo v8 (syn)	0.808	0.838	0.974	0.972	
Yolo v8 (real)	0.304	0.889	0.995	0.93	

Table A.13: Yolo v8 trained on clean and test by synthetic dataset VS Yolo v8 trained on clean and test by real dataset

Class	**1**	**2**	**3**	**4**	**5**
Yolo v8 (syn)	0.68	0.923	0.436	0.389	0.269
Yolo v8 (real)	0.7	0.317	0.239	0.0455	0.331
Class	**6**	**7**	**8**	**9**	**10**
Yolo v8 (syn)	0.305	0.965	0.872	0.812	0.911
Yolo v8 (real)	0.259	0.683	0.586	0.327	0.758
Class	**11**	**12**	**13**	**14**	**15**
Yolo v8 (syn)	0.76	0.955	0.738	0.807	0.983
Yolo v8 (real)	0.526	0.325	0.565	0.278	0.948
Class	**16**	**17**	**18**	**19**	
Yolo v8 (syn)	0.89	0.958	0.986	0.986	
Yolo v8 (real)	0.0742	0.743	0.995	0.909	

Table A.14: Yolo v5 trained on clean and test by synthetic dataset VS Yolo v5 trained on clean and test by real dataset

Class	**1**	**2**	**3**	**4**	**5**
Yolo v5 (syn)	0.732	0.918	0.271	0.448	0.299
Yolo v5 (real)	0.728	0.801	0.239	0.12	0.406
Class	**6**	**7**	**8**	**9**	**10**
Yolo v5 (syn)	0.361	0.96	0.841	0.768	0.87
Yolo v5 (real)	0.395	0.743	0.544	0.316	0.98
Class	**11**	**12**	**13**	**14**	**15**
Yolo v5 (syn)	0.656	0.956	0.646	0.712	0.981
Yolo v5 (real)	0.612	0.309	0.534	0.389	0.979
Class	**16**	**17**	**18**	**19**	
Yolo v5 (syn)	0.817	0.957	0.983	0.993	
Yolo v5 (real)	0.414	0.799	0.995	0.979	

Here are some confusion matrix for YOLOv8 on different datasets.

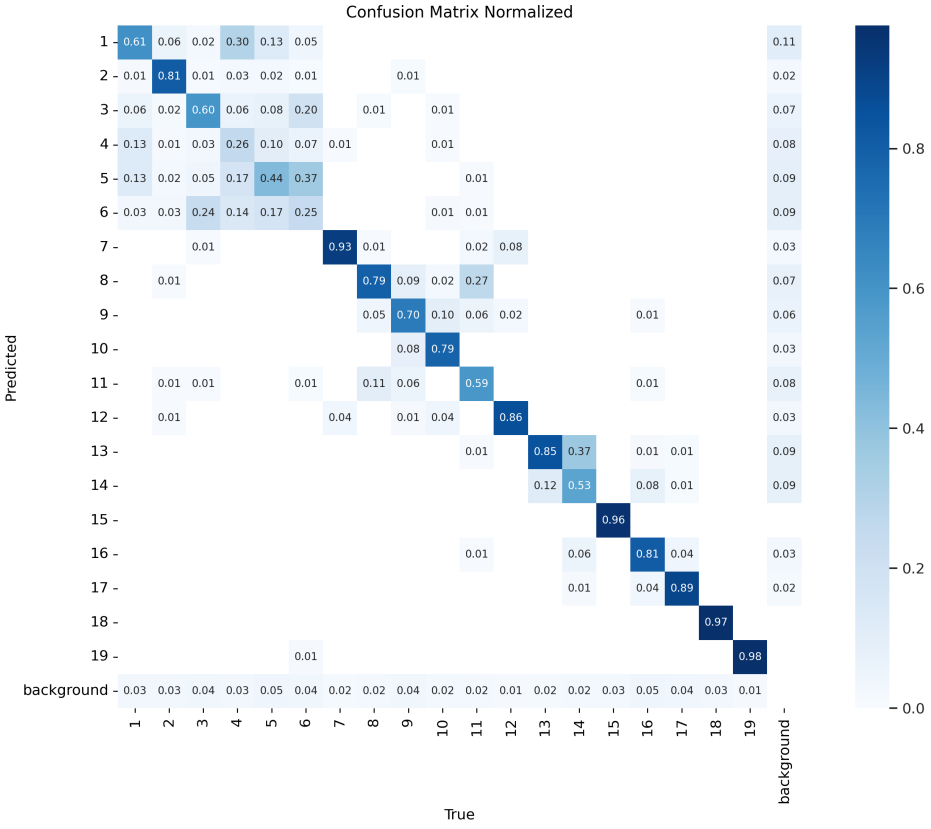


Figure A.2: Confusion matrix of YOLOv8 for the synthetic test dataset.

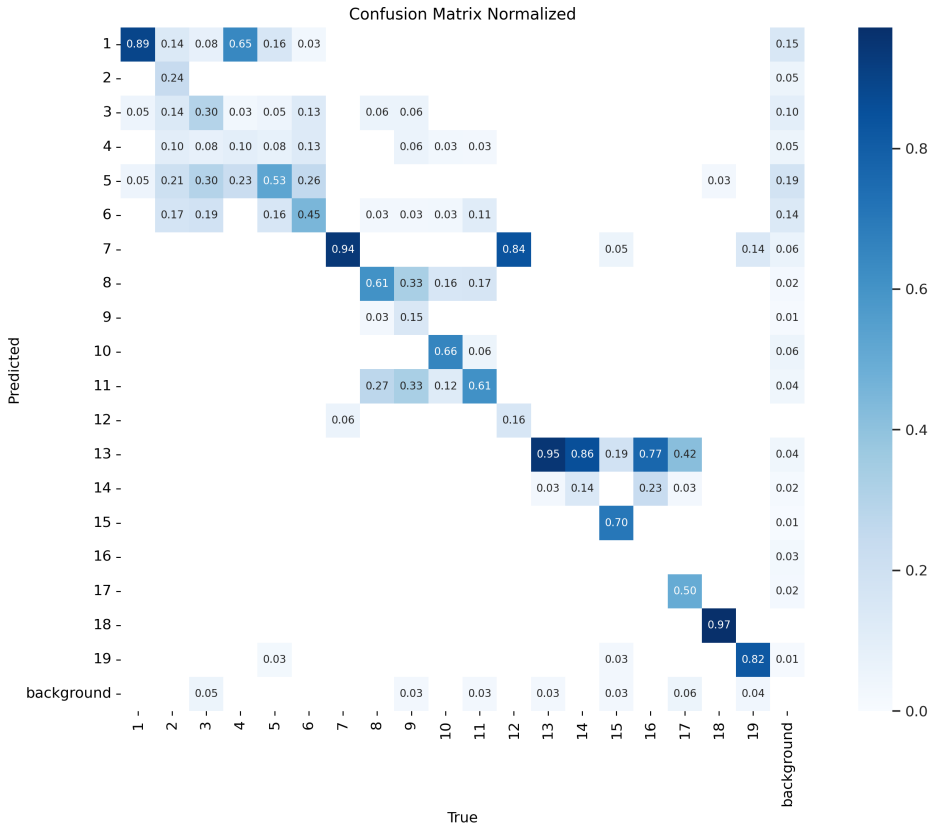


Figure A.3: Confusion matrix of YOLOv8 for the real dataset.

Here provide some detailed result figures for object recognition model YOLOv8 on different datasets.

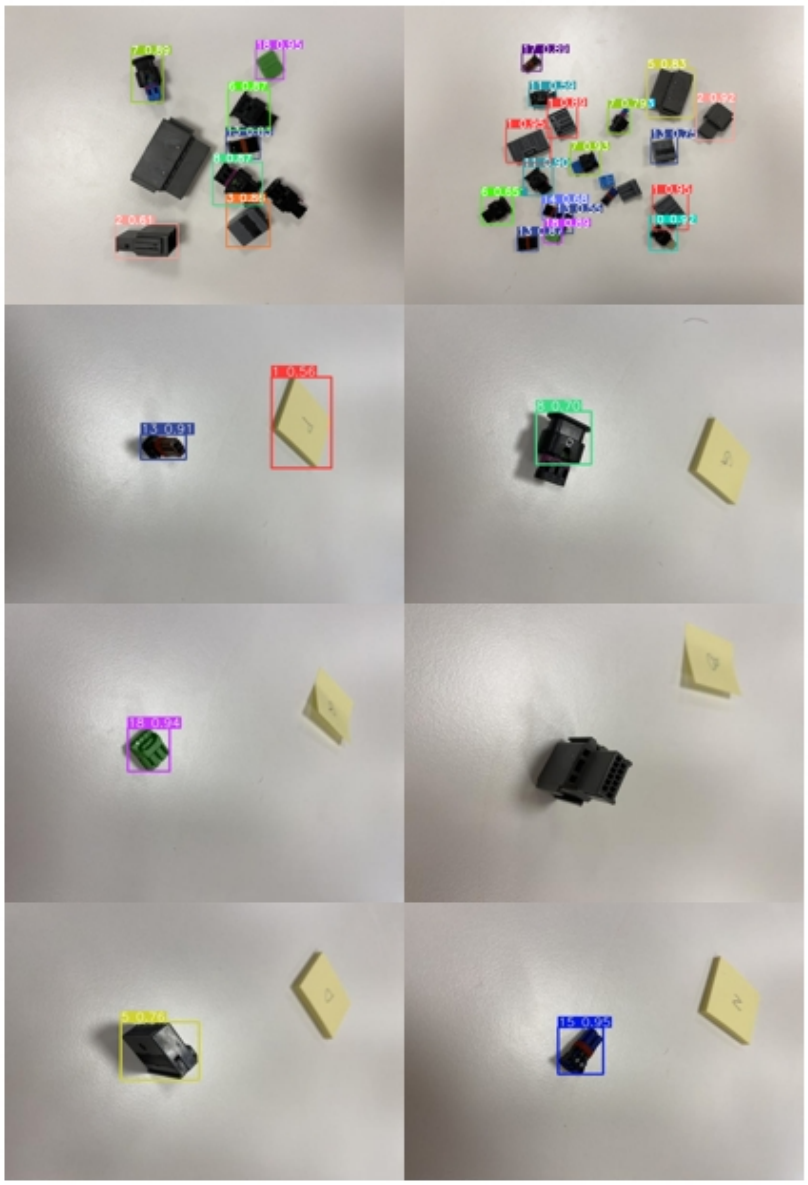


Figure A.4: YOLOv8 tests on the real dataset.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY