Gonçalo Tomaz Arsénio

6D Object Detection for Mobile Robotics Targeting Industrial Applications: a Case Study

Dissertation supervised by Professor Doctor Urbano José Carreira Nunes and Doctor Luís da Silva Garrote submitted to the Electrical and Computer Engineering Department of the Faculty of Science and Technology of the University of Coimbra, in partial fulfillment of the requirements for the Master's Degree in Electrical and Computer Engineering, specialization in Robotics, Control and Artificial Intelligence.

September of 2023

1 2 UNIVERSIDADE D COIMBRA 9 0

6D Object Detection for Mobile Robotics Targeting Industrial Applications: a Case Study

Gonçalo Tomaz Arsénio

Coimbra, September of 2023

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

6D Object Detection for Mobile Robotics Targeting Industrial Applications: a Case Study

Dissertation supervised by Professor Doctor Urbano José Carreira Nunes and submitted to the Electrical and Computer Engineering Department of the Faculty of Science and Technology of the University of Coimbra, in partial fulfillment of the requirements for the Master Degree in Electrical and Computer Engineering, specialization in Robotics, Control and Artificial Intelligence.

Supervisor:
Prof. Dr. Urbano José Carreira Nunes

Co-Supervisor:
PhD Luís Carlos Artur da Silva Garrote

Jury:
Prof. Dr. Jorge Manuel Moreira de Campos Pereira Batista
Prof. Dr. João Pedro de Almeida Barreto
Prof. Dr. Urbano José Carreira Nunes

Coimbra, September of 2023

Acknowledgments

I wish to extend my sincerest appreciation to all those who have contributed valuable contributions to successfully accomplishing my dissertation. This journey has presented both difficulties and rewards, and I would not have been able to achieve this significant milestone without numerous individuals' assistance, motivation, and direction.

First of all, I would like to express my sincere gratitude to my supervisor Dr. Professor Urbano for believing in me and trusting me with this project, along with all the tools needed and feedback throughout the development of this dissertation. Additionally, I extend my gratitude to my co-supervisor Doctor Luís Garrote for his support, encouragement, and patience in sharing his knowledge and constant assistance during this work. A word to Master Ricardo Pereira for his valuable insights and constructive feedback.

My biggest thanks to my family, particularly to my mother for all the unconditional support through this research, the culmination of six years of distance learning. Her dedication and sacrifices have been instrumental in enabling me to reach this significant milestone.

I also extend my appreciation to my friends ”Eletrões”, especially my housemates Diogo Soares, Duarte Cruz, Isidro Ribeiro, and João Castilho., who have made this academic journey all the more enjoyable with their camaraderie and support.

I would also like to express my sincere appreciation to my peers in the Human-Centered Mobile Robotics Lab. In particular, Gabriel Gonçalves and Ulisses Reverendo, for providing emotional and social support during this research.

Last, but certainly not least, I want to take a moment to extend my heartfelt gratitude to Lara Cardoso. Your love, patience, and encouragement have been a constant source of strength and motivation for me. I acknowledge that there were times when my focus on my research may have made me appear distant or preoccupied, and for that, I apologize. I genuinely appreciate your patience during those moments when my academic commitments required my prioritization.

This work has been supported by ”GreenAuto”, with reference C644867037, Portugal, and by ISR-UC FCT through grant UIDB/00048/2020.

Abstract

Mobile robotics can potentially revolutionize many aspects of industry, from material handling and manufacturing to augmented reality or autonomous driving. One of the key challenges in mobile robotics is enabling robots to perceive and manipulate objects in their environment accurately. Detecting the 6D pose of an object is a critical task for this purpose. It enables robots to classify and recognize objects, estimate their poses, and track them over time.

Perceiving an object's pose is also essential for building risk indicators that can be used in motion planning to help robots navigate safely and avoid collisions. However, most of the existing 6D object detection methods have been developed and evaluated in academic contexts, so further studies may be needed to demonstrate their feasibility and effectiveness in industrial environments. It is important to note that this context refers to the problem of determining an object's 6 Degrees of Freedom (6DoF) in 3D space, including its 3D position and 3D orientation. This can be achieved using various techniques, with point clouds, RGB, Depth, or RGB-D images as inputs.

This dissertation introduces several multi-stage frameworks for 6D object detection inspired in DenseFusion, encompassing key components such as object detection using Yolov5, image feature extraction, point cloud feature extraction, fusion network, and pose estimation. Validation on the LINEMOD dataset provided crucial insights into method effectiveness. Further evaluation on the KITTI dataset, unveiled nuanced behaviors based on object proximity. A multimodal analysis was also conducted to assess the influence of varying input data sources on the pose estimator performance, considering the trade-offs between cost-effective cameras and robust yet expensive LiDAR sensors.

Furthermore, a novel approach incorporating LiDAR sensor data was introduced, along with a custom loss function calculating the Chamfer distance between point clouds. This approach yielded a performance comparable to the one achieved with ground truth objects and the original loss function. The research culminated in the validation of all methods on a virtual industrially-focused dataset developed for this purpose, the presented results highlight the possible efficacy of the introduced methods in industrial applications.

Keywords: Deep Learning, 6D Object Detection, 6D Pose Estimation, Industrial Applications

Resumo

A robótica móvel pode potencialmente revolucionar muitos aspetos da indústria, desde o manuseamento de materiais e o fabrico até à realidade aumentada ou à condução autónoma. Um dos principais desafios da robótica móvel é permitir que os robôs percebam e manipulem objetos no seu ambiente com precisão. A deteção da pose 6D de um objeto é uma tarefa crítica para este fim.

A perceção da pose de um objeto é também essencial para a construção de indicadores de risco que podem ser utilizados no planeamento de movimentos para auxiliar os robôs a navegar em segurança e evitar colisões. No entanto, a maioria dos métodos de deteção de objetos 6D existentes foram desenvolvidos e avaliados em contextos académicos, pelo que poderão ser necessários mais estudos para demonstrar a sua viabilidade e eficácia em ambientes industriais. É importante notar que este contexto se refere ao problema de determinar os 6 graus de liberdade de um objeto no espaço 3D, incluindo a sua posição 3D e orientação 3D. Isto pode ser conseguido usando várias técnicas, tendo como entrada nuvens de pontos, imagens RGB, imagens de profundidade ou imagens RGB-D.

Esta dissertação introduz várias abordagens de múltiplas etapas para a deteção de objetos 6D inspiradas na DenseFusion, englobando componentes-chave como a deteção de objetos utilizando Yolov5, extração de características de imagem, extração de características de nuvens de pontos, rede de fusão e estimativa de pose. A validação no conjunto de dados LINEMOD forneceu informações cruciais sobre a eficácia dos métodos. Uma avaliação mais aprofundada no conjunto de dados KITTI revelou comportamentos diferenciados com base na proximidade do objeto. Também foi realizada uma análise multimodal para avaliar a influência de diferentes fontes de dados de entrada no desempenho do estimador de pose, considerando as vantagens e desvantagens entre câmaras económicas e sensores LiDAR robustos, mas dispendiosos.

Foi ainda introduzida uma nova abordagem que incorpora dados do sensor LiDAR, acompanhada por uma função de perda personalizada que calcula a distância Chamfer entre nuvens de pontos. Esta abordagem produziu um desempenho comparável ao obtido com objetos *ground truth* e a função de perda original. A investigação culminou com a validação de todos os métodos num conjunto de dados virtuais de cariz industrial desenvolvido para o efeito. Os resultados apresentados realçam a possível eficácia dos métodos introduzidos em aplicações industriais.

Palavras-chave: Aprendizagem Profunda, Deteção de Objetos 6D, Estimação de Pose 6D, Aplicações Industriais

“Education is the most powerful weapon which you can use to change the world.”
Nelson Mandela

Contents

Acknowledgments **i**

Abstract **iii**

Resumo **v**

List of Acronyms **xiii**

List of Figures **xvi**

List of Tables **xviii**

1 Introduction **1**

1.1 Context and Motivation 1
1.2 Proposed Framework 2
1.3 Objectives and Key Contributions 4

2 Background Material **6**

2.1 Deep Neural Network (DNN) 6
2.1.1 Convolution Neural Network (CNN) 6
2.1.2 YOLOv5 7
2.2 Point Cloud Clustering 8
2.3 Object Detection Algorithms 8
2.3.1 DenseFusion 11
2.3.1.1 Object Segmentation 11
2.3.1.2 RGB Feature Extractor 12
2.3.1.3 Point Cloud Feature Extractor 13
2.3.1.4 Pixel-wise Dense Fusion Network 14
2.3.1.5 Pose Estimator 15
2.3.1.6 Iterative Pose Refinement 17

3 State of the Art **19**

3.1 3D Object Detection 19
3.1.1 RGB Based Methods 19
3.1.2 RGB-D Based Methods 21

3.1.3 Point Cloud/Depth Based Methods 22
3.2 6D Object Detection 23
3.2.1 RGB Based Methods 24
3.2.2 RGB-D Based Methods 25
3.2.3 Point Cloud/Depth Based Methods 27
3.3 Datasets 27

4 Developed Work **29**
4.1 Segmentation-based Pose Estimation Methods 29
4.1.1 DenseFusion: Multimodal Early Fusion 29
4.1.2 DenseFusion: Multimodal Middle Fusion 29
4.1.3 DenseFusion: Depth 32
4.1.4 DenseFusion: Point Cloud 32
4.2 Detection-based Pose Estimation Methods 32
4.2.1 DenseFusion: Point Cloud 33
4.2.1.1 Depth Map Projection from Point Cloud 33
4.2.1.2 Point Cloud Projection from Depth Map 35
4.2.1.3 Object Selection 35
4.2.1.4 Upsampling and Downsampling of the Objects 36
4.2.1.5 Loss Function 36
4.2.1.6 Prediction Selection 37
4.3 Industrial Dataset 38

5 Software Tools and Hardware Materials **41**
5.1 Operating System 41
5.2 Python 41
5.2.1 Pytorch 41
5.3 Conda 42
5.4 Evaluation Setup 42

6 Results and Discussion **44**
6.1 Segmention-based Pose Estimation Methods 44
6.1.1 Evaluation Metrics 44
6.1.1.1 Average Distance (ADD) 44
6.1.1.2 Average Distance of Keypoints with Symmetry (ADD-S) 45
6.1.2 Multimodal Study 45
6.1.3 Feature Extraction Variations 47
6.1.4 Comparative Study with State-of-art 48
6.2 Detection-based Pose Estimation Methods 49
6.2.1 Distance-Based Accuracy Study 51
6.2.2 Multimodal Study 53
6.2.3 Prediction Selection Variations 55
6.2.4 DenseFusion: PointCloud 56

6.3 Validation on Industrial Dataset 58

7 Conclusion **61**

7.1 Future Work 62

Bibliography **63**

List of Acronyms

3DGP	3D Geometric Phrase
6DoF	6 Degrees of Freedom
AI	Artificial Intelligence
AUC	Area Under the Curve
BEV	Birds-Eye-View
CDPN	Coordinates-Based Disentangled Pose Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSP	Cross-Stage-Partial-connections
CUDA	Compute Unified Device Architecture
DL	Deep Learning
DNN	Deep Neural Network
EPOS	Estimating Pose of Objects with Symmetries
FLOPs	Floating Point Operations
GPU	Graphics Processing Unit
HHA	Horizontal Disparity, Height above Ground, and Angle with gravity
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
ML	Machine Learning
MLP	Multi-Layer Perceptron
PANet	Path Aggregation Network
PnP	Perspective-n-Point
PSPNet	Pyramid Scene Parsing Network

RANSAC	RANdom SAmple Consensus
ResNet	Residual Network
RoI	Region of Interest
RPN	Region Proposal Network
RSN	Range Sparce Net
SLOSS	ShapeMatch-Loss
SE(3)	Special Euclidean group in 3 dimensions
SE-SSD	Self-Ensembling Single-Stage Detector
SGD	Stochastic Gradient Descent
SPP	Spatial Pyramid Pooling
SPPF	Spatial Pyramid Pooling Fast
SSD	Single-Stage Detector

List of Figures

1.1 Pipeline for framework using RGB-D. 3

1.2 Pipeline for framework using Depth. 3

1.3 Pipeline for framework using Point Cloud only. 3

1.4 Pipeline for framework using RGB-D with Middle Fusion. 4

1.5 Taxonomy of the developed frameworks. 5

2.1 Overview of a Convolution Neural Network. 7

2.2 Representation of Yolov5 Network Architecture. Adapted from [1]. 8

2.3 Comparison between different clustering methods. Adapted from Scikit-Learn Plot Cluster Comparison. 10

2.4 DBSCAN clustering result. The clusters correspond to a car, pedestrian, car, and pedestrian, respectively. 10

2.5 Overview of DenseFusion 6D pose estimation model. Taken from [2]. 11

2.6 Representation of the Object Segmentation Network. Adapted from [3]. 12

2.7 Overview of the Residual Block. Adapted from [4]. 13

2.8 Representation of the RGB Feature Extractor Network. Adapted from [5]. 13

2.9 Representation of the Point Cloud Feature Extractor and Pixel-wise Dense Fusion Network. 15

2.10 Representation of the Pose estimator Network. 16

2.11 Overview of the Iterative Pose Refinement. Taken from [2]. 17

4.1 Overview of DenseFusion: Multimodal Early Fusion Architecture. 30

4.2 Overview of DenseFusion: Multimodal Middle Fusion Architecture. 31

4.3 Overview of DenseFusion: Depth Architecture. 32

4.4 Overview of DenseFusion: Point Cloud Architecture. 33

4.5 Overview of DenseFusion: Point Cloud Architecture. 34

4.6 Example of rendered RGB-D images, and respective Masks and Point Clouds from Industrial Dataset. 39

6.1 The accuracy-threshold curve of the multimodal methods on the LINEMOD dataset. 46

6.2 Box plot of the accuracy of the multimodal methods on the LINEMOD dataset. . 46

6.3 Visualization of the multimodal methods on the LINEMOD dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results of DenseFusion, DenseFusion: Depth, and DenseFusion: Point Cloud, respectively. 47
6.4 The accuracy-threshold curve of the methods with feature extraction variations on the LINEMOD dataset. 48
6.5 Box plot of the accuracy of the methods with feature extraction variations on the LINEMOD dataset. 48
6.6 Visualization of the methods with feature extraction variations on the LINEMOD dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Four columns show the results of DenseFusion, DenseFusion: Early Fusion, DenseFusion: Middle Fusion 1, and DenseFusion: Middle Fusion 2, respectively. 49
6.7 Box plot of the accuracy of the state-of-art methods on the LINEMOD dataset. . 51
6.8 Visualization of the method according to the distance on the KITTI dataset, considering only objects with more than 100 points. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results for <7m, 7-15m, 15-25m, respectively. 53
6.9 Visualization of the method according to the distance on the KITTI dataset, considering only objects with more than 50 points. Green dots indicate the ground truth poses, while red represents estimated poses. Four columns show the results for <7m, 7-15m, 15-25m, and 25-35m, respectively. 54
6.10 Visualization of the multimodal methods on the KITTI dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results of RGB-D, Depth, and Point Cloud, respectively. . . . 55
6.11 Visualization of the method according to the prediction selection on the KITTI dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results for argmax, weighted avg, and weighted avg 50%, respectively. 57
6.12 Confusion matrix of the 2D detection on the KITTI depth images. 58
6.13 Precision-Recall curve of the 2D detection on the KITTI depth images. 58
6.14 Visualization of the DenseFusion: Point Cloud on the KITTI dataset. Green dots indicate the ground truth poses, while red represents estimated poses. 59
6.15 Confusion matrix of the 2D detection on the Industrial dataset depth map images. 59
6.16 Precision-Recall curve of the 2D detection on the Industrial dataset depth map images. 59
6.17 Visualization of the proposed methods on the developed dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Four columns show the results for DenseFusion, DenseFusion: Depth, DenseFusion: PointCloud, DenseFusion: Middle Fusion respectively. 60

List of Tables

3.1 Summary of RGB 3D Object Detection Methods. 20
3.2 Summary of RGB-D 3D Object Detection Methods. 22
3.3 Summary of Point Cloud 3D Object Detection Methods. 24
3.4 Summary of RGB 6D Object Detection Methods. 25
3.5 Summary of RGB-D 6D Object Detection Methods. 26
3.6 Summary of Point Cloud 6D Object Detection Methods. 27
3.7 Datasets used for multiple object detection and pose estimation methods. 28

5.1 Python packages 41
5.2 NVIDIA GeForce RTX 3060 specifications 42

6.1 Accuracy (ADD(-S)) of the multimodal methods on the LINEMOD dataset. Objects in bold are symmetric. 46
6.2 Accuracy (ADD(-S)) of methods with feature extraction variations on the LINEMOD dataset. Objects in bold are symmetric. 48
6.3 Accuracy (ADD(-S)) of RGB state-of-art methods on the LINEMOD dataset. Objects in bold are symmetric 50
6.4 Accuracy (ADD(-S)) of RGB-D state-of-art methods on the LINEMOD dataset. Objects in bold are symmetric 50
6.5 Accuracy (ADD(-S)) of Depth state-of-art methods on the LINEMOD dataset. Objects in bold are symmetric 51
6.6 Accuracy (ADD) according to the distance of the objects on the KITTI dataset. Objects with less than 100 points were not considered. 52
6.7 Accuracy (ADD) according to the distance of the objects on the KITTI dataset. Objects with less than 50 points were not considered. 52
6.8 Accuracy (ADD) of the multimodal methods on the KITTI dataset. 55
6.9 Accuracy (ADD) for different prediction selection methods on the KITTI dataset. 56
6.10 Accuracy (ADD) for DenseFusion: Point Cloud on the KITTI dataset. 58
6.11 Accuracy (ADD) for the proposed methods on the developed dataset. 58

1 Introduction

In this chapter, lies discussions regarding what drove motivation for the project, objectives established for it, and contributions that have been made to fortify research in this area.

1.1 Context and Motivation

Since mobile robots are becoming an increasingly common sight in industrial setups for activities such as material handling or transportation of goods or assembly or inspections, they must be able to perceive and interact with their surroundings accurately [6]. This implies having accurate object detection abilities and accurately estimating an object's position (its six degrees of freedom), including its orientation within space. In many instances (e.g., pallet handling; bin picking; or object tracking), determining an object's pose plays a critical role. However, detecting an item's 6D pose against an industrial background can prove challenging given factors like unpredictability concerning visual obstructions, cluttered surroundings, and variable lighting conditions, amongst other demanding aspects unique to such environments.

The work of this dissertation is part of an ongoing research project known as "GreenAuto: Green Innovation for the Automotive Industry" [1]. The primary objective within this project is to develop and evaluate methods for detecting and precisely estimating the 6D pose of objects in industrial settings, such as warehouses and manufacturing facilities. The ultimate purpose of this detection system is to integrate it into an autonomous forklift operation. This autonomous forklift will navigate to designated drop-off and pick-up zones, and then detect objects of interest and manage their transportation. The deployment of this system is envisioned to significantly enhance warehouse efficiency and reinforce workplace safety by advancing the automation of forklift maneuvers.

The proposed research explored various Deep Learning-based object detection and pose estimation methods, focusing on addressing the specific challenges in industrial environments. Deep Learning (DL) [7] is a subfield of Artificial Intelligence (AI) that has revolutionized how machines process and assimilate information. The inspiration for this breakthrough lies in the intricate structure and functioning of the human brain, particularly its neural networks. DL models have demonstrated exceptional performance in computer vision tasks. These models can

[1]https://transparencia.gov.pt/pt/fundos-europeus/prr/beneficiarios-projetos/projeto/02-C05-i01.02-2022.PC644867037-00000013/

gain valuable insights and make accurate predictions by leveraging large datasets and computational power. Advances in this field have profound implications for various industries and hold immense potential for shaping the future of technology and innovation [8, 9, 10].

1.2 Proposed Framework

The main purpose of this dissertation was to develop and evaluate 6D pose estimation methods in industrial environments. Based on the DenseFusion framework [2], various methods were developed, using different input data and approaches for the pose estimation.

The 6D Object Detection pipelines depicted in Figs. 1.1, 1.2,1.3, 1.4, embody a sequence of stages that enable robust and accurate object detection in industrial environments. This multi-stage framework encompasses vital components, including but not limited to, object detection, image feature extraction (Figs. 1.1, 1.2, 1.4), point cloud feature extraction, fusion network (Figs. 1.1, 1.2, 1.4), and pose estimation. Each stage plays a crucial role in contributing to the overall performance of the pipeline.

At the forefront of the pipeline lies the object detection stage. This phase involves identifying objects of interest within the input data. The objective is to localize the objects and propose potential regions of interest precisely. To do so, the Yolov5 [11] was the object detection network chosen to recognize the objects present in the images. Following the object detection stage, the image feature extractor comes into play. By making use of a Convolutional Neural Network (CNN), the feature extractor processes the detected Region of Interest (RoI) and captures high-level features intrinsic to each object. This enables the 6D detection framework to recognize distinguishing characteristics even within cluttered or occluded scenes. Simultaneously, point cloud data provides a depth-rich representation of the environment in scenarios involving 3D perception. The point cloud feature extractor decodes the spatial coordinates, which then identifies significant features that enhance the knowledge acquired from 2D images. The true strength of the pipeline lies in its ability to fuse information from both image and point cloud data. The network fuses the extracted features, enabling the model to benefit from the complementary nature of the two data sources. The pose estimator stage uses deep learning techniques to accurately infer the object's position and orientation in 3D space.

In order to validate these frameworks, the development of a dataset was imperative. This necessity was originated from the absence of a realistic and accessible dataset online for validating the accurate detection of pallets within a warehouse setting. For instance, the PalLoc6D dataset [12], an RGB-D virtual dataset for the 6D detection of pallets, which lacks a realistic scenario because the pallets are generated randomly in various locations, surrounded by random objects, within a randomized background.

Along with the creation of this dataset, a comprehensive multimodal analysis was made to understand the influence of varying input data on the performance of the pose estimator. This analysis is relevant because the industry remains uncertain about which sensors to employ. While Light Detection and Ranging (LiDAR) presents unparalleled resilience to diverse weather conditions and lighting variations, it comes with the drawback of being very expensive

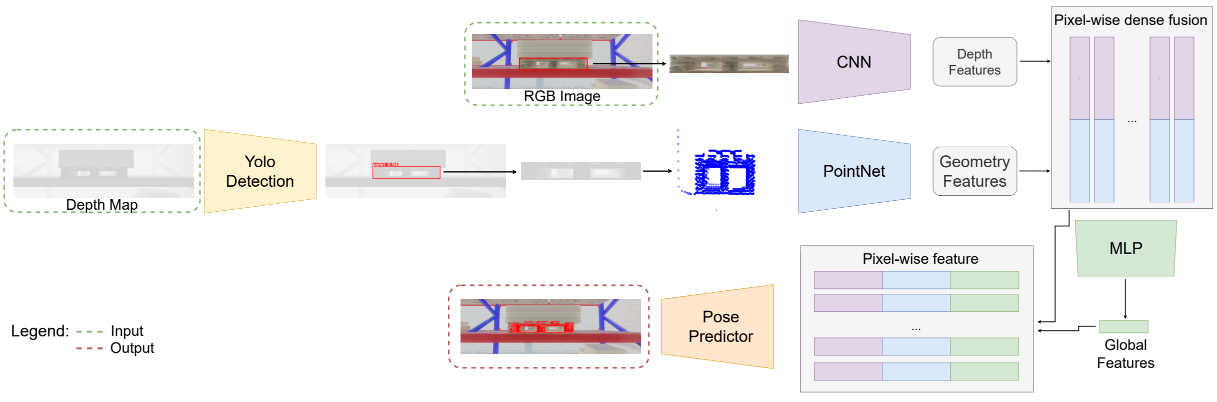

Figure 1.1: Pipeline for framework using RGB-D.

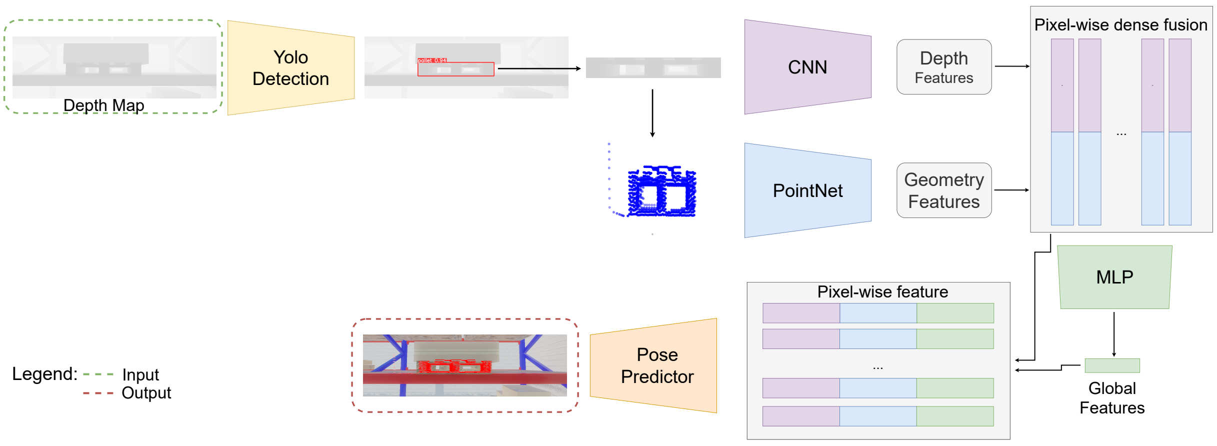

Figure 1.2: Pipeline for framework using Depth.

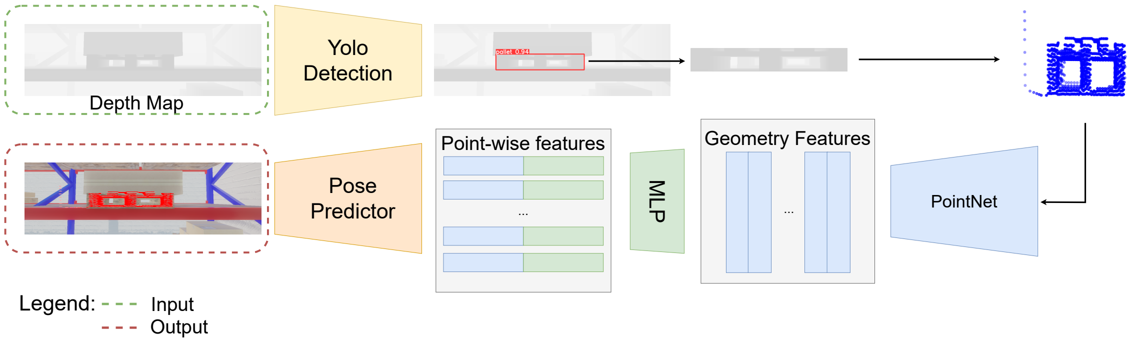

Figure 1.3: Pipeline for framework using Point Cloud only.

for large-scale implementation. Moreover, the point cloud data it generates, representing the spatial coordinates of object surfaces, introduces added complexity in automating object feature recognition and extraction. On the other hand, cameras offer a more cost-effective alternative, capable of discerning color in the environment and identifying objects at higher resolutions. However, they fail to maintain recognition accuracy when fluctuating illumination and struggle to identify distant objects within static images [13].

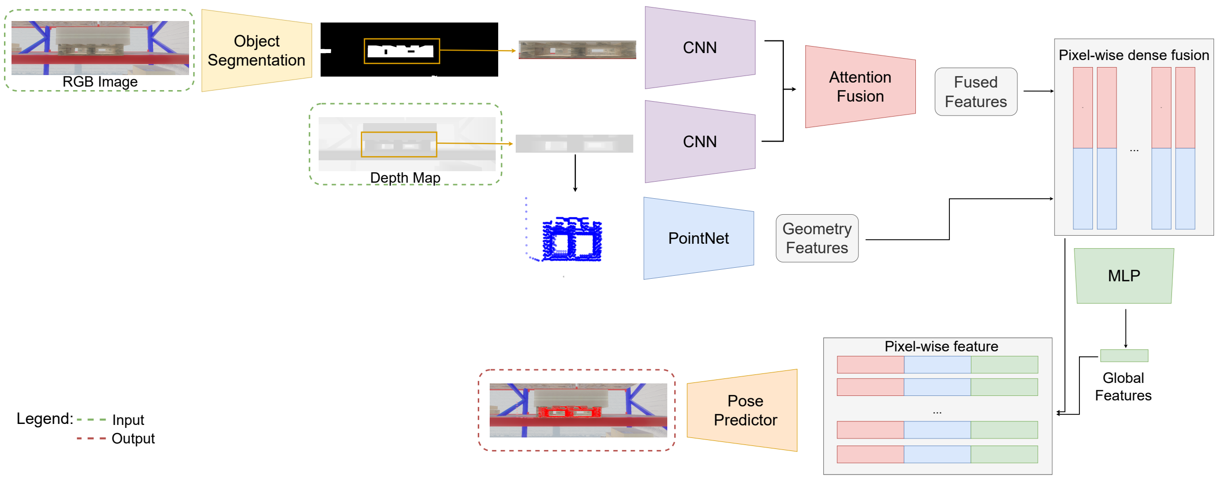

Figure 1.4: Pipeline for framework using RGB-D with Middle Fusion.

1.3 Objectives and Key Contributions

For the development of the 6D object detection in an industrial environment, several objectives were established:

1. Testing several 6D object detection methods;
2. Evaluation of the tested methods in benchmark datasets;
3. Detailed analysis of DenseFusion;
4. Development of multiple methods for 6D object detection based on DenseFusion (Fig. 1.5):
    (a) Masked-based DenseFusion: Depth;
    (b) Masked-based DenseFusion: Point Cloud;
    (c) Masked-based DenseFusion: Multimodal Middle Fusion;
    (d) Masked-based DenseFusion: Early Fusion;
    (e) Detection-based DenseFusion: Depth;
    (f) Detection-based DenseFusion: Point Cloud;
    (g) Detection-based DenseFusion;
5. Evaluation of the developed methods in benchmark datasets;
6. Development of a virtual dataset simulating an industrial environment;
7. Evaluation of the developed methods in the industrial dataset;

The succeeding chapters of this dissertation clarify on the main implementations and contributions of this study:

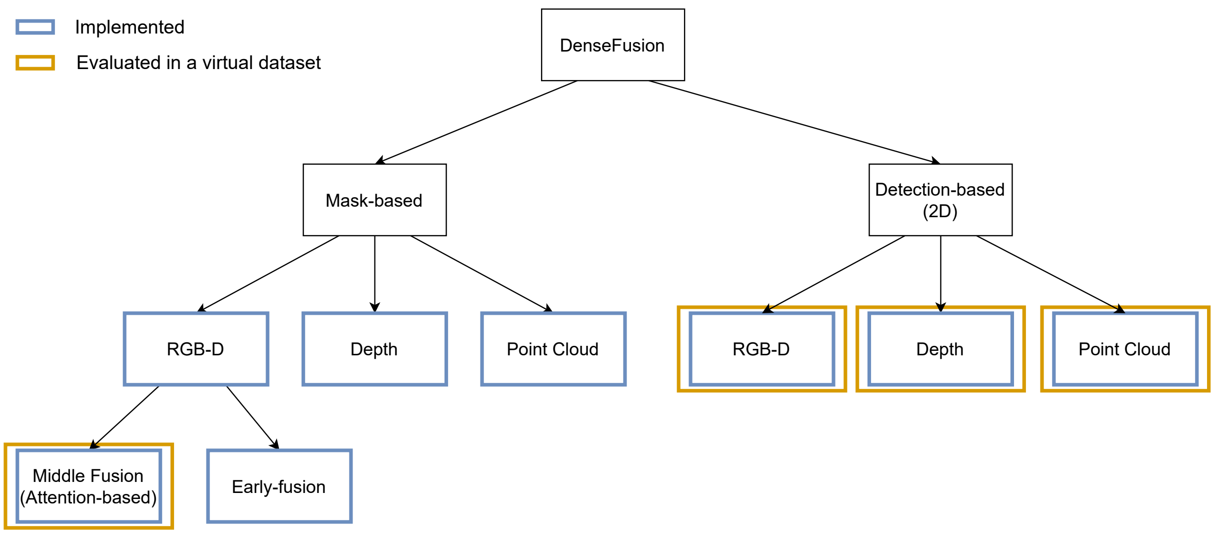

Figure 1.5: Taxonomy of the developed frameworks.

Developed Work (Chapter 4)

Presents an overview of developed methods for 6D pose estimation, including the developed dataset for industrial environment validation.

Software Tools and Hardware Materials (Chapter 5)

Describes the software and hardware components that were utilized in order to achieve the designated goals.

Results and Discussion (Chapter 6)

Evaluation and analysis of all the proposed methods, including the validation in an industrial environment.

2

Background Material

This chapter will elaborate upon a comprehensive description of the methodologies that support the progression of this dissertation.

2.1 Deep Neural Network (DNN)

Deep Neural Network (DNN) is a subdivision of Machine Learning (ML) that is concerned with the prediction of outputs, whether they are supervised, semi-supervised, or unsupervised [14]. In contrast to traditional ML techniques, which are often limited in their capacity to process raw data, the application of DL represents a highly sophisticated form of DNN that leverages multiple layers to extract high-level features from the input data. For instance, when applied to image processing, the lower layers of the DL model are capable of detecting basic edges, while the higher layers can distinguish specific letters, objects, or object features [15].

2.1.1 Convolution Neural Network (CNN)

The CNN possesses a deep feed-forward architecture and showcases a remarkable aptitude for generalizing to advanced networks that integrate fully connected layers. Primarily utilized for image analysis, particularly pattern recognition, the CNN can also be utilized to tackle other data analysis challenges, such as classification issues. Their notable ability to be trained properly without overfitting is particularly significant, along with their effortless application to large networks [16].

A CNN can be divided into the feature extraction and classification modules. Feature extraction is the process of identifying and extracting important features from an input image. Classification is the process of assigning a label to an input image. The label can be a category, such as “cat” or “dog”, or it can be a numerical value, such as the probability that the image is a cat.

The feature extraction and classification processes are represented in Fig. 2.1 and consist of the following:

1. The input image is subjected to a sequence of convolutional layers.
2. The convolutional layers extract spatial features from the image.
3. The spatial features are passed through a series of pooling layers.

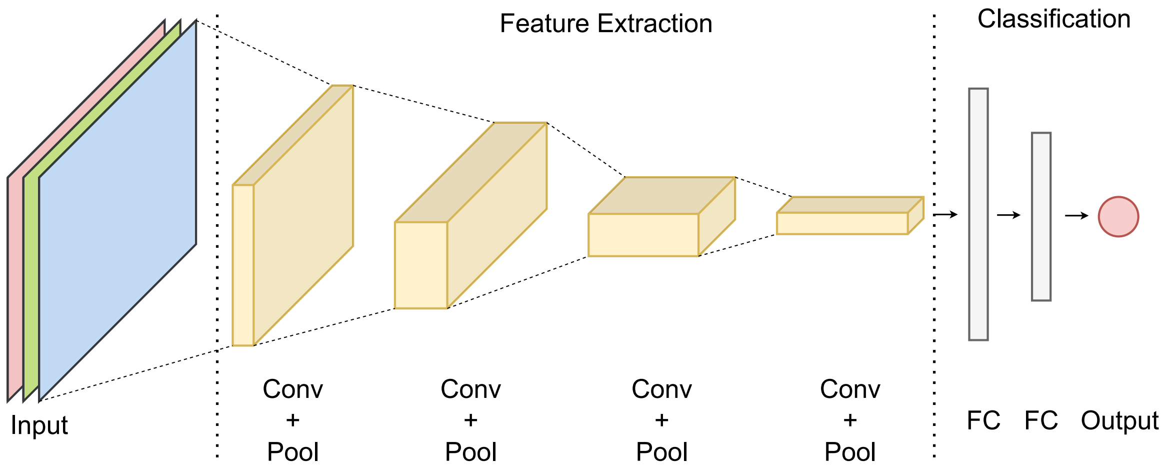

Figure 2.1: Overview of a Convolution Neural Network.

4. The pooling layers reduce the dimensionality of the spatial features.
5. The spatial features are passed through a series of fully connected layers.
6. The fully connected layers classify the image.

2.1.2 YOLOv5

YOLOv5 [11] is a state-of-the-art object detection model that was released in 2020. It comprises four different models: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The architecture of YOLOv5, represented in Fig. 2.2, is built upon CSP-Darknet53 [17] as the backbone, incorporating a Spatial Pyramid Pooling Fast (SPPF) [18] layer. The Neck of the model employs CSP-PANet [19], and the detection head follows the YOLOv4 [20] methodology.

CSP-Darknet53 fuses two popular CNN designs, Darknet and Cross-Stage-Partial-connections (CSP). The CSP framework reduces the network's computational cost while preserving its high accuracy. This is achieved by separating the feature maps into two distinct segments, processing them individually, and then integrating them. This process decreases the necessary computations and facilitates quicker training and inference durations.

Spatial Pyramid Pooling (SPP) is a pooling methodology commonly employed in CNNs to create a fixed-length representation regardless of image size or scale. SPPF, an optimized variant of SPP, was developed by the YOLOv5 creators and features a reduced number of Floating Point Operations (FLOPs).

Path Aggregation Network (PANet) is a neural network improvement technique that aims to enhance information propagation within the network by incorporating precise localization signals in the lower layers of the feature hierarchy. This is accomplished by implementing a bottom-up path augmentation strategy, effectively reducing the distance between the lower layers and the highest-level feature.

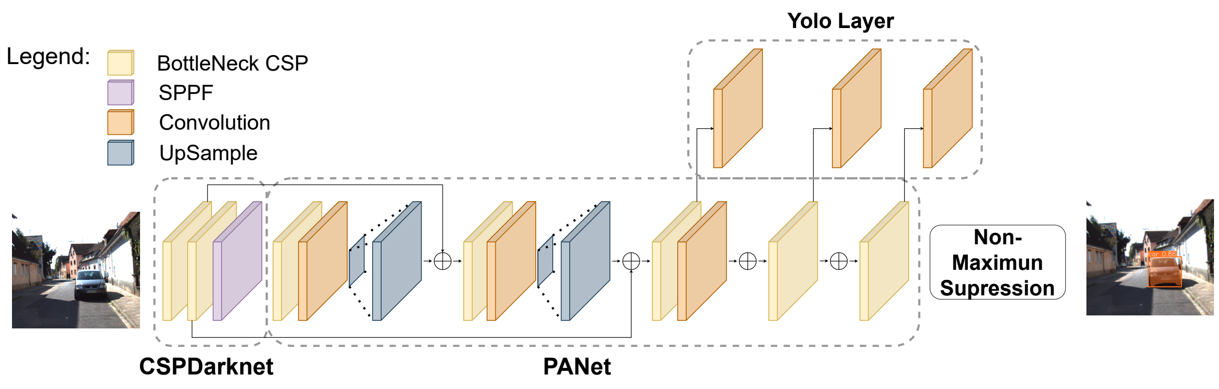

Figure 2.2: Representation of Yolov5 Network Architecture. Adapted from [1].

2.2 Point Cloud Clustering

The process of point cloud clustering refers to the method of categorizing points within a point cloud according to their similarities. This technique is commonly employed to partition a point cloud into distinct entities or to discern diverse characteristics within a particular object. Numerous clustering algorithms have been proposed in recent years, making the field of clustering techniques extensive.

Ester *et al.* [21] proposed DBSCAN, a density-grounded algorithm that discerns clusters based on the density of points in a specified area. The algorithm classifies points into three categories: core, border, and noise points. Core points are characterized by the presence of at least a predetermined number of points within a given range, as specified by the input parameter. In contrast, border points are those with fewer than the specified number of points but still within the range of a core point. Noise points are data points that neither qualify as core points nor as border points. Consequently, they do not belong to any cluster and are typically considered outliers or noise in the dataset. Ng *et al.* [22] introduced Spectral Clustering, which uses the top eigenvectors of a matrix obtained from the disparity between points to cluster them. Müllner *et al.* presented an Agglomerative Clustering Algorithm, which operates by merging the nearest clusters iteratively until all the data points are consolidated into a single cluster. The outcome of this clustering approach is a dendrogram, a diagram resembling a tree, that reveals the hierarchical connections among the clusters. These methods can be compared in Fig. 2.3.

Of all the clustering techniques discussed earlier, DBSCAN algorithm 1 was the most remarkable one owing to its efficient performance and superior clustering outcomes. The results of the implementation of the clustering algorithm can be seen in Fig. 2.4.

2.3 Object Detection Algorithms

To achieve the objectives proposed for this dissertation, a 6D object detection algorithm is needed. To do so, it is possible to implement and develop a new pose estimation algorithm or choose one from the multiple methods available and start from there. For the purpose of this study, the second option was the obvious choice. To that end, intensive research was done to

Algorithm 1: DBCSAN Algorithm

```
Data points $D$, neighborhood distance $\epsilon$, minimum points $MinPts$ Clusters $C$ and noise
points $N$
$C \leftarrow \emptyset$ ;                                   // Initialize empty set of clusters
$N \leftarrow \emptyset$ ;                               // Initialize empty set of noise points
foreach unvisited point $p$ in $D$ do
    Mark $p$ as visited;
    $N' \leftarrow$ getNeighbors($p, \epsilon$);
    if $|N'| < MinPts$ then
        Add $p$ to $N$;
    else
        Create a new cluster $C_i$;
        expandCluster(p, $N'$, $C_i$, $\epsilon$, $MinPts$);
        Add $C_i$ to $C$;
return $C$ (clusters) and $N$ (noise points);
expandCluster($p$, $N'$, $C_i$, $\epsilon$, $MinPts$):
Add $p$ to $C_i$;
foreach point $q$ in $N'$ do
    if $q$ is unvisited then
        Mark $q$ as visited;
        $N'' \leftarrow$ getNeighbors($q, \epsilon$);
        if $|N''| \geq MinPts$ then
            $N' \leftarrow N' \cup N''$;
    if $q$ does not belong to any cluster then
        Add $q$ to $C_i$;
getNeighbors($p$, $\epsilon$):
Initialize an empty set $N$;
foreach point $q$ in $D$ do
    if distance($p, q$) $\leq \epsilon$ then
        Add $q$ to $N$;
return $N$;
```

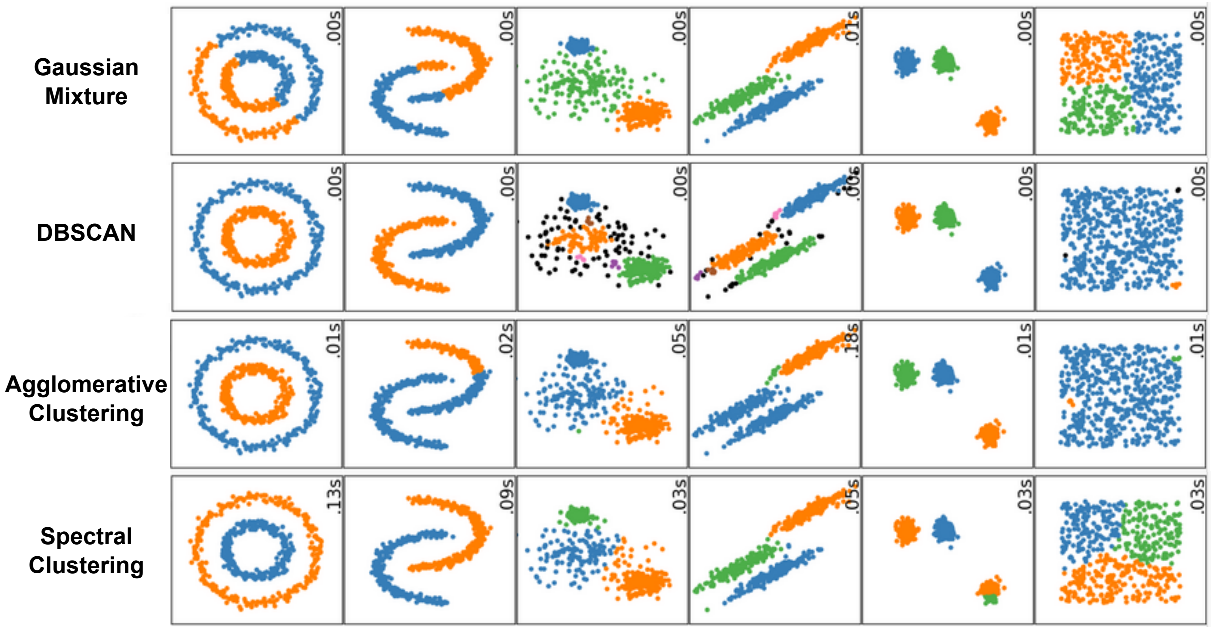

Figure 2.3: Comparison between different clustering methods. Adapted from Scikit-Learn Plot Cluster Comparison.

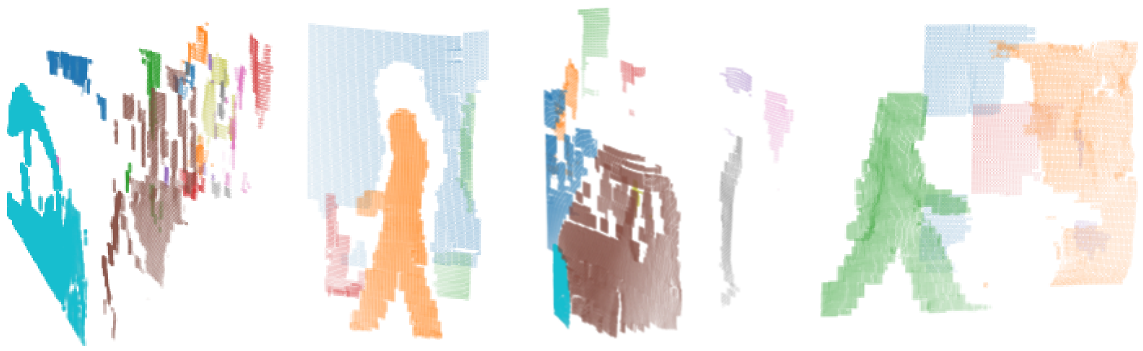

Figure 2.4: DBSCAN clustering result. The clusters correspond to a car, pedestrian, car, and pedestrian, respectively.

understand what method would be the best for what is intended.

Ideally, all the methods in chapter 3 would be implemented, tested and compared. However, most methods with good results and available code were developed with Compute Unified Device Architecture (CUDA) 10 or less, while the machine being used for the development of this research has a recent Graphics Processing Unit (GPU), section 5.4, and is only capable of running from CUDA 11 up.

Although this limitation, some methods were tested: CenterSnap [23], ShAPO [24], DenseFusion [2] and Point Pillars [25]. CenterSnap and ShAPO ended up being discarded because their code was developed too much around the NOCS [26] dataset, making it difficult to switch to a dataset with a different structure. Also, they used pre-training weights and files that were not explicit about how these could be generated.

While superior-performing methods exist, testing them was not possible due to incompat-

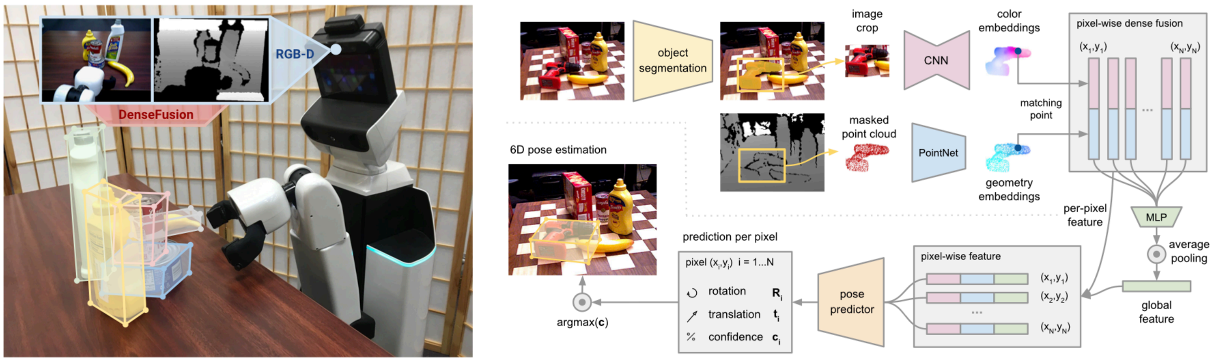

Figure 2.5: Overview of DenseFusion 6D pose estimation model. Taken from [2].

ibilities between CUDA and PyTorch. The selection of DenseFusion as the preferred approach was based on its ease of implementation and the availability of online support, which are crucial considerations when implementing such tools. Additionally, DenseFusion is currently one of the most precise estimators of 6D pose, and its architecture is not excessively complex. Therefore, this chapter presents the methodology of DenseFusion, covering its implementation details and its training and testing methods.

2.3.1 DenseFusion

DenseFusion is a framework for estimating the 6D orientation of known objects from RGB-D images. As the authors explain, this technique introduces an architecture that independently processes the two complementary data sources, RGB and depth. Moreover, a dense fusion network extracts pixel-wise dense feature embeddings, from which orientation is estimated. In addition, this approach incorporates an iterative pose refinement procedure that increases the precision of orientation estimation while enabling near real-time inference. The architecture of DenseFusion can be seen in Fig. 2.5.

2.3.1.1 Object Segmentation

The initial stage involves segmenting the objects of interest within the image. The semantic segmentation network is based on the PoseCNN [3] architecture, a convolutional neural network arranged in an encoder-decoder configuration to capture high-level and low-level features effectively. The encoder module comprises multiple convolutional layers, followed by batch normalization and ReLU activation. This encoding process progressively diminishes the spatial dimensions of the feature maps, enabling the network to capture increasingly complex and abstract representations. After each convolutional layer, max pooling operations were implemented to downsample the feature maps while preserving crucial local information.

The decoder module is designed to reestablish the spatial dimensions of the feature maps while enhancing the acquired representations. It commences by executing upsampling operations utilizing the max pooling indices from the corresponding encoder stage. The upsampled feature maps then undergo a series of convolutional layers, followed by batch normalization and ReLU activation. This method gradually improves the features and ensures the network learns to reconstruct segmentation maps accurately. This network can be seen in Fig. 2.6.

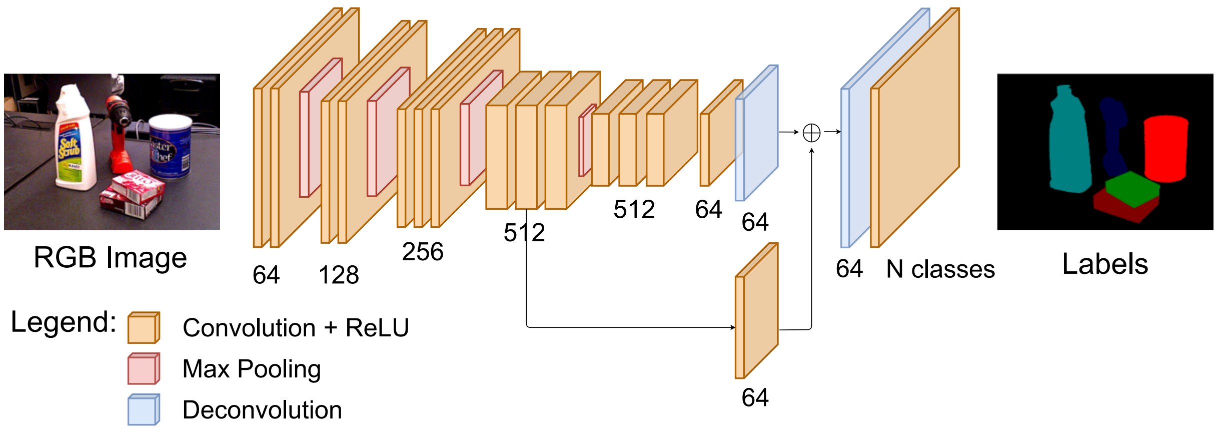

Figure 2.6: Representation of the Object Segmentation Network. Adapted from [3].

2.3.1.2 RGB Feature Extractor

The RGB feature extractor is a modified version of the Residual Network (ResNet) architecture integrated with the Pyramid Scene Parsing Network (PSPNet) module. This feature extractor is designed to extract rich and informative features from RGB images. At the core of the feature extractor lies the ResNet backbone, known for its ability to train deep models effectively.

ResNet [4] was first introduced to address a specific challenge in training very deep neural networks. As neural networks grow deeper, i.e., have more layers, they have the potential to capture more complex patterns and features from data, which is beneficial for many tasks, especially in computer vision.

However, as traditional deep neural networks became increasingly deeper, it was observed that utilizing additional layers in certain convolutional neural networks resulted in poorer performance than the same network with fewer layers. Unexpectedly, researchers also determined that this drop in performance couldn't be attributed to overfitting. This phenomenon defies intuition, as one would typically expect a larger network to perform at least as well as its smaller counterpart. However, Stochastic Gradient Descent (SGD) appears to encounter difficulties in converging to an optimal solution within expansive networks.

ResNet was specifically designed to overcome this. The introduction of residual blocks allowed the network to learn residual functions, that represent the difference between the input and output of a layer in a neural network, instead of unreferenced ones. This means the network can focus on learning the adjustments or "residuals" to the identity mapping of the input, rather than trying to learn the entire mapping from scratch. The residual block consists of two or three streams of convolutional neural networks, followed by an element-wise addition operation that combines the input with the output of the convolutional layers, as demonstrated in Fig. 2.7.

In this particular implementation, ResNet-18 serves as the backbone. The main difference in ResNet-18 compared to other ResNet architectures is its number of layers. ResNet-18 consists of 18 layers, while other ResNet architectures have more layers. When balancing the trade-off between accuracy and computational resource utilization, ResNet-18 emerges as the best choice.

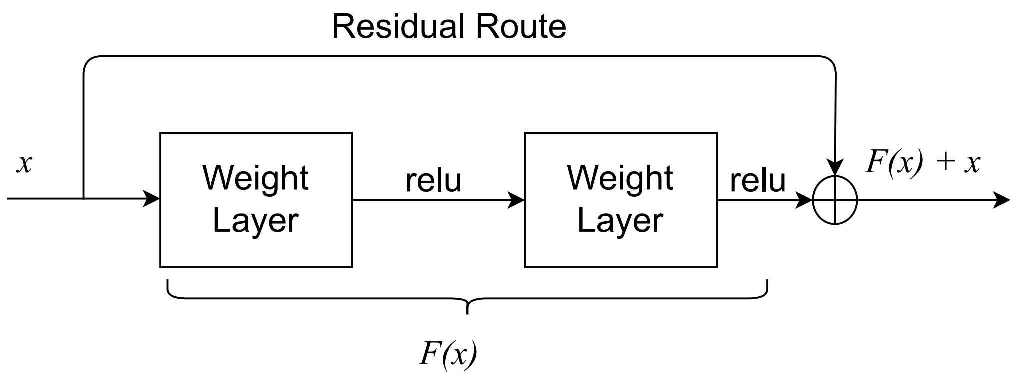

Figure 2.7: Overview of the Residual Block. Adapted from [4].

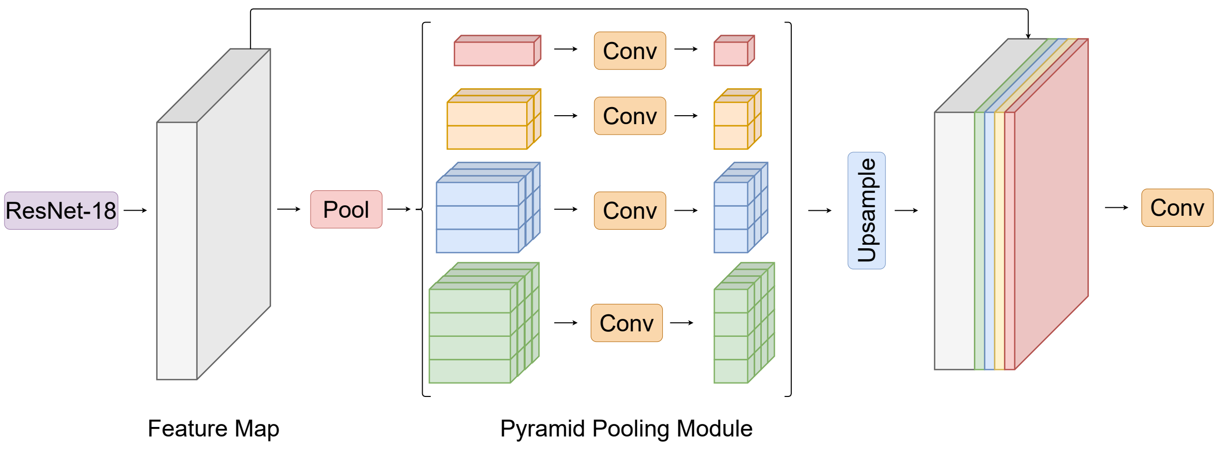

Figure 2.8: Representation of the RGB Feature Extractor Network. Adapted from [5].

The PSPNet [5] module is incorporated to enhance the feature extractor's capability. The PSPNet module leverages a pyramid pooling strategy to capture multiscale contextual information from the input image. It divides the input feature maps into multiple stages, each employing adaptive average pooling and convolution operations to extract features at different spatial resolutions. These features are then bilinearly upsampled and concatenated with the original features. A bottleneck convolutional layer reduces the dimensionality of the concatenated features to enhance efficiency.

The feature extraction process begins by passing the input RGB image through the ResNet backbone. The backbone network extracts both low-level and high-level features from the image. These features are further processed by the PSPNet module, which captures contextual information at multiple scales and incorporates it into the feature representation. By integrating this module, it is possible to have a pose estimation network that is more resilient for objects of varying scales. An overview of the architecture can be seen in Fig. 2.8.

2.3.1.3 Point Cloud Feature Extractor

Previous methodologies have utilized CNNs to process the depth image as a supplementary image channel. Nonetheless, such an approach fails to consider the depth channel's inherent three-dimensional (3D) structure. Instead, the DenseFusion framework involves initially converting the segmented depth pixels into a 3D point cloud by employing the known camera intrinsic parameters, followed by a PointNet-like architecture to extract the geometric characteristics.

The depth segmented object is converted into a 3D point cloud by the following equation:

$$\begin{cases} x_w = z_c \cdot (u - u_0) \cdot f_u \\ y_w = z_c \cdot (v - v_0) \cdot f_v \\ z_w = z_c \end{cases} \tag{2.1}$$

where u and v represent two coordinates of a given point within the depth map, $(u0, v0)$ serves as the central coordinate for the said map, z_c denotes the relevant depth value, and f is the focal length of the camera. Finally, x_w, y_w, and z_w denote the three coordinates within the world coordinate system.

PointNet introduced by Qi et al. [27] is a neural network architecture designed to process point cloud data, an essential type of geometric data structure. Unlike alternative techniques that convert point cloud data into regular 3D voxel grids or sets of images PointNet directly processes point clouds and upholds the permutation invariance of points within the input. This means that the network can handle point clouds of varying sizes and orders without the need for pre-processing.

PointNet uses max pooling to aggregate local features of points into global features. Max pooling is a common operation in CNNs that reduces the spatial dimensions of feature maps while retaining the most important information. In PointNet, max pooling is applied to the output of a Multi-Layer Perceptron (MLP) that processes the local features of each point. The resulting global feature vector represents the entire point cloud and is used for classification or segmentation tasks. Max pooling is used because it is a simple and effective way to reduce local features and capture the most salient information.

However, DenseFusion's approach introduces a modified version of the model, which replaces the max-pooling reduction function with an average pooling method. Using average pooling instead of max pooling can provide a more fine-grained encoding of information in the vicinity of each point and the point cloud as a whole. An overview of the architecture can be seen in Fig. 2.9.

2.3.1.4 Pixel-wise Dense Fusion Network

The concept behind the pixel-wise dense fusion network is to shift away from relying solely on the object's global features to determine its pose. Instead, it harnesses the power of local per-pixel features, allowing each of these features a global context and allowing them to make pose predictions independently. This approach enables the method to potentially pick the most reliable predictions based on the visible portion of the object, reducing the impact of problems like objects partially hidden from view or interference from background elements. An overview of the architecture can be seen in Fig. 2.9.

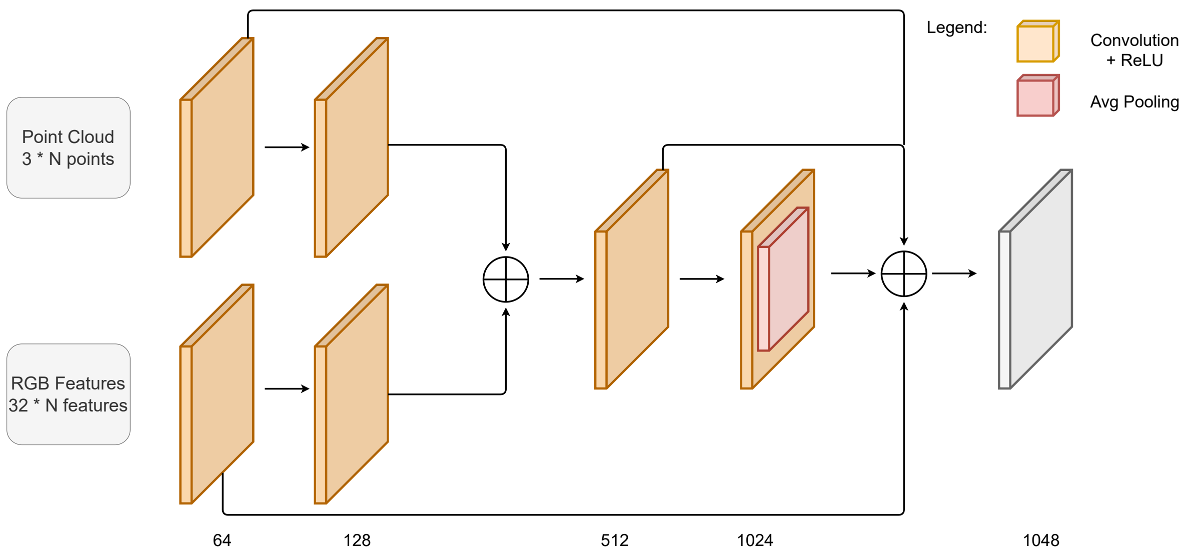

Figure 2.9: Representation of the Point Cloud Feature Extractor and Pixel-wise Dense Fusion Network.

2.3.1.5 Pose Estimator

The pose estimator block in the DenseFusion framework estimates the 6D pose of known objects from RGB-D images. The block takes as input the pixel-wise dense feature embedding generated by the dense fusion network and outputs the predicted pose of the object. The pose estimator block uses a residual-based approach to estimate the pose, which is trained jointly with the main network. The residual-based approach calculates the distance between the ground truth pose and the corresponding points transformed by the estimated pose, known as the pose estimation loss. This loss is quantified by the distance between said points. The pose estimation loss is defined as:

$$L = \frac{1}{N}\sum_{i=1}^{N}(L_i^p c_i - w\log(c_i)), \tag{2.2}$$

where N is the number of points sampled from the segmented object, c_i is the confidence coefficient, and w is incorporated as a secondary regularization term to balance the average distance loss and confidence. For asymmetric objects (objects where their halves are not symmetrical reflections of one another), L_i^p represents the average distance between the N corresponding points on the object after the ground truth pose and the estimated pose transformation. L_i^p is defined as:

$$L_i^p = \frac{1}{M}\sum_{j}\|(Rx_j + T) - (\hat{R}_i x_j + \hat{T}_i)\|, \tag{2.3}$$

where M is the number of points randomly selected from the 3D model of the object, x_j is the jth point in M, R and T are the ground truth rotation and translation, respectively. And $\hat{R}_i$ and $\hat{T}_i$ are the rotation and translation of the ith fused feature estimate in N, respectively.

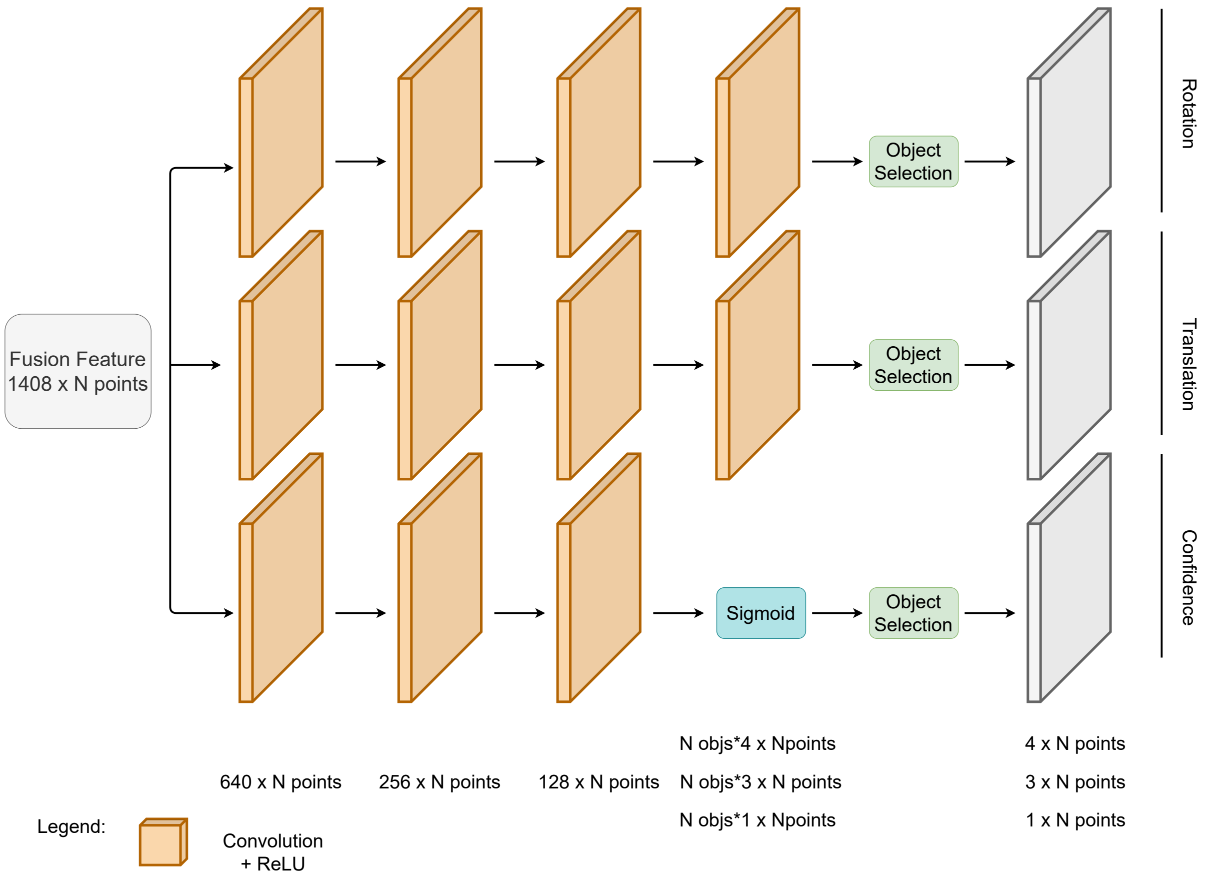

Figure 2.10: Representation of the Pose estimator Network.

Symmetric objects have more than one and possibly an infinite number of canonical frames, any orientation can be chosen for the object reference frame, and it will still exhibit the same symmetry, which leads to ambiguous learning objectives, therefore modified L_i^p is defined as follows when the object is symmetric:

$$L_i^p = \frac{1}{M} \sum_j \min_{0<k<M} \|(Rx_j + T) - (\hat{R}_i x_k + \hat{T}_i)\|. \tag{2.4}$$

The output of this network consists of n-point predictions, where n represents the total number of points in the point cloud. Each prediction includes the rotation quaternion, translation vector, and confidence score, all contributing to the estimated pose. Incorporating this final value allows the network to undergo a self-supervision process, whereby it can autonomously evaluate the quality of its predictions. Ultimately, the pose that is selected as the final prediction is the one that is associated with the highest score. The Pose Estimator architecture can be seen in Fig. 2.10.

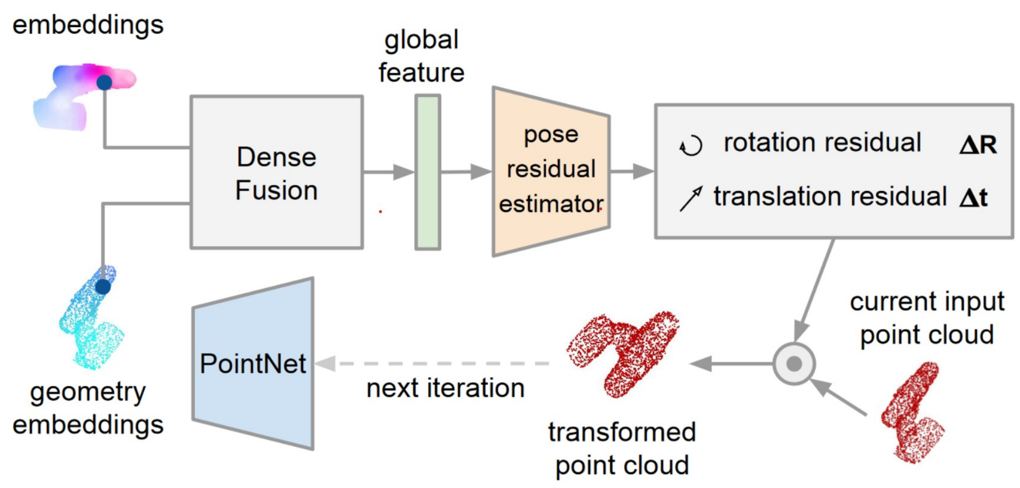

Figure 2.11: Overview of the Iterative Pose Refinement. Taken from [2].

2.3.1.6 Iterative Pose Refinement

The goal of the iterative pose refinement is to rectify its own errors in pose estimation by taking a step-by-step approach. The challenge lies in training the network to improve its initial prediction rather than generating entirely new ones. In order to overcome this, the previous projection is incorporated as part of the input for the next iteration. The fundamental concept involves regarding the formerly predicted pose as an approximation of the desired target entity's standard frame, and then transform the input point cloud to match this estimated frame. This way, the converted point cloud implicitly represents the estimated pose. This transformed point cloud is then reintroduced into the network, and a new pose is predicted based on the previous estimation. This iterative process can be repeated multiple times, potentially leading to more accurate pose estimations with each step. The final pose is obtained through K iterations, defined as follows:

$$RT = [R_k|t_k] \cdot [R_{k-1}|t_{k-1}] \cdots [R_0|t_0] \tag{2.5}$$

The Iterative Pose Refinement overview can be seen in Fig. 2.11.

3

State of the Art

In this chapter, a comprehensive summary of the literature on the task of object detection is presented.

In recent years, significant advancements have been made in the field of object detection, and numerous methodologies have been proposed to address this issue. Typically, these advanced techniques are categorized into three primary groups based on the type of input data: RGB-based methods, which exclusively rely on color information to determine the pose of an object; RGB-D methods, which use both color and depth data; and point-cloud-based methods, which utilize specialized algorithms to depict objects as sets of 3D points for feature extraction and pose estimation. Each approach has strengths and limitations depending on the specific requirements of each scenario or application task.

3.1 3D Object Detection

3D Object Detection is the process of recognizing items in a 3D setting and making predictions about their positioning (X, Y, Z) and the object's rotation around its center or origin. This particular methodology is typically employed in the fields of robotics, autonomous automobiles, and enhanced reality applications.

3.1.1 RGB Based Methods

Detecting three-dimensional objects from a single RGB image is challenging due to the inherent ambiguity involved. Existing approaches can be classified into three categories: geometry-based, learning-based, and deep learning-based. Geometry-based methods [28] use geometry and 3D world assumptions to estimate 3D bounding boxes. Learning-based methods incorporate category-specific 3D shape priors [29, 30] or additional 2.5D information [31] to detect 3D bounding boxes or reconstruct the shape of the object in 3D. Deep learning-based methods [32, 33] estimate 3D object bounding boxes directly from 2D object detection. A summary of these method can be found in Table 3.1.

Choi *et al.* [28] introduced a hierarchical scene model for understanding complex indoor scenes using a 3D Geometric Phrase (3DGP) model. The 3DGP captures the semantic and geometric relationships among objects that often coincide within the identical 3D spatial configuration, allowing the system to prefer certain configurations based on deformation cost and

Table 3.1: Summary of RGB 3D Object Detection Methods.

Method	Year	Category	Summary
Choi *et al.* [28]	2013	Geometry-based	The model captures semantic and geometric relationships between objects that frequently co-occur in the same 3D spatial configuration.
Huang *et al.* [29]	2018	Learning-based	The algorithm seeks to minimize the differences between the input image and the rendered images generated by the 3D representation, over the space of depth, surface normal, and object segmentation map.
Mono3d++ [30]	2019	Learning-based	The model detects the 3D pose and shape of vehicles by optimizing two-scale projection consistency between the generated 3D hypotheses and the 2D pseudo-measurements and integrating task priors into an overall energy function.
3D-RCNN [31]	2018	Learning-based	The algorithm is a deep convolutional neural network that learns to map image regions to the full 3D shape and pose of all object instances in the image. Produces a 3D representation of the scene.
Mousavian *et al.* [32]	2017	Deep Learning-based	The model estimates the 3D object orientation and dimension using a hybrid discrete-continuous loss and combines these with geometric constrains to recover an accurate pose.
Huang *et al.* [33]	2018	Deep Learning-based	The algorithm uses parametrization of targets and cooperative training across different modules to improve the prediction accuracy.

view-dependent biases. The spatial deformation of an object in a 3DGP node is determined by the difference between the object's instance location and the expected location relative to the centroid (average position of all the constituent objects).

Huang *et al.* [29] proposed a two-step method in which 3D object detection and 3D layout estimation are learned jointly, while Kundu *et al.* [31] presented a method for 3D object recognition and reconstruction using category-specific object shape priorities through a render-and-compare approach.

Mono3D++ was presented by He *et al.* [30]. The method optimizes the two-scale projection consistency between the generated 3D suppositions and their 2D pseudo measurements. In order to accomplish this, an overall energy function includes three task priors, which play a vital role in ensuring that the 2D projection of the 3D bounding box is contained within the truncated 2D box measurement.

Huang *et al.* in [33] proposed a method that uses parametrization of targets and cooperative training across different modules to improve prediction's accuracy. The parametrization helps maintain the consistency between the 2D image and the 3D world. The process of parametrization plays a crucial role in upholding the consistency between the 2D image and the 3D world.

Mousavian *et al.* [32] introduced a hybrid discrete-continuous loss. The discrete component divides the 360-degree range of possible orientations into a fixed number of bins, and the continuous component estimates the offset from the center of each bin. This approach allows the model to estimate the orientation with high precision while avoiding the limitations of purely discrete or continuous methods.

However, the precise localization of bounding boxes is difficult due to the lack of depth information, especially for monocular imagery. Methods based on RGB images have exhibited relatively poorer performance when compared to RGB-D methods, owing to the fact that pixel intensity can vary for different appearances. Nevertheless, these methods have the potential to provide useful range information thanks to the advantages of camera sensors.

3.1.2 RGB-D Based Methods

Various methods exist in the field of RGB-D (color and depth) techniques, and these can be classified into three main categories: 2.5D processing methods, 2D driven 3D methods, and 3D convolution-based methods. 2.5D processing techniques [34, 35] employ depth images as 2D images directly. In the case of 2D Driven 3D methods [36, 37], it is typical to maintain RGB constant and to transform depth images into point clouds. 3D convolution-based methods [38, 39] converts the depth maps into point clouds. A summary of these methods can be found in Table 3.1.

Gupta *et al.* [34] introduced a geometric embedding for depth images that encodes height above ground and angle with gravity for each pixel, a measurement of the angle that the local surface normal of a pixel makes with the inferred gravity direction, in addition to the horizontal disparity, called Horizontal Disparity, Height above Ground, and Angle with gravity (HHA).

3D-SSD [35], proposed by Luo *et al.*, consists of two components: hierarchical feature fusion and multi-layer prediction. Hierarchical feature fusion combines appearance and geometric features from RGB-D images to exploit 2.5D representations in a synergetic way, improving accuracy and efficiency. Multi-layer prediction utilizes multi-scale features for object detection. It addresses the issue of object sizes by attaching a set of 3D anchor boxes with varying sizes to every location of the prediction layers.

Qi *et al.* [36] presented F-PointNet, a method that takes a frustum of a point cloud and a 2D object detection bounding box as input to predict the 3D bounding box. The accuracy of the 3D prediction depends on the quality of the 2D prediction.

Frustum VoxNet [37], by Shen *et al.*, is a system that first detects objects in 2D and then detects the 3D objects within the 3D frustums these 2D detections define by voxelizing parts of the frustums. The main novelty of this method, was determining which parts of the frustums to voxelize, allowing for high-resolution representations around the objects of interest and reduced memory requirements.

PointFusion [38], by Xu *et al.*, is a method that uses a CNN and a PointNet [27] architecture to independently process the image and the point cloud data, respectively. The outputs are subsequently merged through a fusion network, which predicts multiple 3D boxes along with their confidences, employing the input 3D points as points of reference in space.

In 2020, Qi *et al.* introduced imVoteNet [39], the network applies 2D detectors on RGB images to generate 2D votes. These votes are produced by projecting 3D bounding boxes onto the 2D image plane and creating 2D heatmaps for each object category. The geometric signals from the 2D votes are lifted to 3D, along with semantic/texture cues. The deep Hough voting framework generates 3D votes, created by combining the features of the points within each 3D bounding box. Using gradient blending through a multi-tower architecture, the method blends 2D and 3D detection.

Table 3.2: Summary of RGB-D 3D Object Detection Methods.

Method	Year	Category	Summary
Gupta *et al.* [34]	2014	2.5D Processing	The model uses semantically rich image and depth features for object detection. They introduced HHA, a geocentric embedding for depth images that encodes height above ground and angle with gravity for each pixel in addition to the horizontal disparity.
3D-SSD [35]	2020	2.5D Processing	The algorithm uses a network that combines appearance and geometric features and utilizes multi-scale features for object detection.
F-PointNet [36]	2018	2D Driven 3D	The model operates directly on raw point clouds and leverages both 2D object detectors and advanced 3D deep learning for object localization.
Frustum Voxnet [37]	2020	2D Driven 3D	The model detects objects in 2D and then detects the 3D objects within the 3D frustums the these 2D detections define by voxelizing parts of the frustums.
PointFusion [38]	2018	3D Convolution-based	The algorithm combines image and point cloud information using a CNN and PointNet architecture. The outputs are combined by a fusion network which predicts multiple hypotheses and their confidences.
imVoteNet [39]	2020	3D Convolution-based	The model fuses 2D votes in images and 3D votes in point clouds, and explicitly extracts both geometric and semantic features from the 2D images.

3.1.3 Point Cloud/Depth Based Methods

3D Object Detection within point cloud data can be categorized into four main approaches: point-based methods, voxel-based methods, range-view-based methods, and multi-view-based methods. Point-based methods [40, 41] are input-wise permutation invariant, meaning they operate directly on individual points within the point cloud. Voxel-based methods [25, 42] use a grid-representation-based approach, dividing the point cloud into volumetric cells (voxels). Range-view-based methods [43, 44] capture multiple range views of the point cloud from different perspectives, converting the 3D detection problem into multiple 2D image-like detections. Multi-view-based methods [45, 46] combine the advantages of both range-view-based and voxel-based approaches. A summary of these methods can be found in Table 3.3.

PointRCNN [40], by Shi *et al.*, is a two-stage framework. The first stage generates a few 3D proposals from the point cloud in a bottom-up manner via segmenting the point cloud of the whole scene into foreground points and background. The proposals are refined in the canonical coordinates in the second stage to obtain the final detection results. The network of the second stage transforms the pooled points of each proposal to canonical coordinates to learn better local spatial features, which are combined with global semantic features of each point learned in the first stage for accurate box refinement and confidence prediction.

CenterPoint [41], by Yin *et al.*, consists of two stages. First, the framework detects the centers of objects using a keypoint detector, based on a 2D CNN feature map, and regresses to other attributes, including 3D size and 3D orientation. Secondly, the estimates from the first stage are refined using additional point features on the object.

Lang *et al.* introduced PointPillars [25], which works by encoding a point cloud into a sparse tensor representation, which reduces the number of computations required for object detection. The method employs PointNet for acquiring a depiction of point clouds that are structured in a vertical arrangement (referred to as pillars). The encoded characteristics can be applied in conjunction with any conventional 2D convolutional detection framework, and a lean downstream is proposed.

Proposed by Deng *et al.*, VoxelRCNN [42] is a two stage approach that uses voxel-based representation of 3D data. The first stage is a 3D backbone network that takes in the raw point cloud data and generates a set of voxel features. These features are computed by dividing the input data into a 3D grid of voxels and computing statistics (such as mean or max) for each voxel. The second stage is a 2D Birds-Eye-View (BEV) Region Proposal Network (RPN) that takes in the voxel features and generates a set of candidate object proposals. The BEV RPN operates on a 2D projection of the voxel grid onto the ground plane, which reduces the computational cost and allows for efficient processing.

RangeDet [43], by Fan *et al.*, is a single-stage LiDAR-based 3D object detector that uses the range view representation. It works by designing three components by analyzing the existing range-view-based methods and addressing the issues of scale variation and inconsistency between 2D range image coordinates and 3D Cartesian coordinates. These components include a range-aware feature extractor, a range-aware anchor-free head, and a range-aware post-processing module. The range-aware feature extractor extracts features from the range view representation and uses a range-aware convolutional layer to handle the scale variation issue. The range-aware anchor-free head predicts the 3D bounding boxes and class labels of objects using the range-aware feature maps. The range-aware post-processing module refines the predicted 3D bounding boxes by considering the objects' range information and geometric constraints.

Range Sparce Net (RSN) [44], by Sun *et al.*, works by predicting foreground points from range images and then processing the selected foreground points using sparse convolutions to detect objects. The implementation of lightweight 2D convolutions on dense range images leads to a notable reduction in the number of selected foreground points, thereby facilitating the subsequent operation of sparse convolutions in the RSN. The fusion of features derived from the range image has the potential to enhance the accuracy of detection.

Shi *et al.* proposed PointVoxel-RCNN [45], which is a framework that integrates 3D voxel CNN and PointNet-based set abstraction to learn more discriminative point cloud features for accurate 3D object detection from point clouds. The framework uses a 3D voxel CNN to condense a 3D scene into a few keypoints using a unique voxel set abstraction module. This reduces the need for additional computations and encodes important scene features. After generating 3D proposals with the voxel CNN, the framework employs RoI-grid pooling to extract proposal-specific features from the key points to the RoI-grid points through keypoint set abstraction.

Presented by Zheng *et al.*, Self-Ensembling Single-Stage Detector (SE-SSD) [46] contains a pair of teacher and student Single-Stage Detectors (SSDs). The teacher SSD generates soft targets, filtered using an Intersection over Union (IoU)-based matching strategy to obtain high-quality training samples for the student SSD. The SE-SSD employs an augmentation methodology in order to generate shape-aware augmented samples for instructing the student with the primary goal of optimizing the assimilated knowledge from the teacher model. This encourages the student to infer complete object shapes.

3.2 6D Object Detection

6D Object Detection is the process of recognizing 3D objects in a 3D space about their positioning (X,Y,Z) and orientation (roll, pitch, yaw). This task is important in various applications,

Table 3.3: Summary of Point Cloud 3D Object Detection Methods.

Method	Year	Category	Summary
PointRCNN [40]	2019	Point-based	The model is a two-stage framework. In the first stage, a sub-network generates a small number of 3D proposals via segmenting the point cloud of the whole scene. The proposals are refined in the canonical coordinates in the second stage.
CenterPoint [41]	2021	Point-based	The algorithm first detects center of objects using a keypoint detector and regresses to other attributes. In a second stage, refines these estimations using additional point features on the object.
PointPillars [25]	2019	Voxel-based	The model is encoder which utilizes PointNets to learn a representation of point clouds organizes in pillars. The encoded features can be used with any standard 2D convolutional detection architecture.
VoxelRCNN [42]	2021	Voxel-based	The algorithm consists of a 3D backbone network, a 2D Birds-Eye-View region proposal network, and a detect head.
RangeDet [43]	2021	Range-view-based	The model includes a range-aware feature extractor, range-aware anchor-free head, and a range-aware post-processing module to prevent issues of scale variation and inconsistency between 2D range image coordinates and 3D cartesian coordinates.
RSN [44]	2021	Range-view-based	The algorithm predicts foreground points from range images and applies sparse convolutions on the selected foreground points to detect objects.
PVRCNN [45]	2020	Multi-view-base	The model summarizes the 3D scene with a 3D voxel CNN into a small set of keypoints via a voxel set abstraction module to save follow-up computations and also to encode representative scene features.
SE-SSD [46]	2021	Multi-view-based	The algorithm contains a pair of teacher and student SSD's, and uses an IoU-based matching strategy to filter soft targets from the teacher and formulate a consistency loss to align student predictions with them.

including robotics, augmented reality, virtual reality, and autonomous vehicles.

3.2.1 RGB Based Methods

RGB methods for estimating an object's pose can be categorized into: holistic approaches, dense correspondence exploration, and 2D-keypoint-based methods. Holistic approaches [47, 3] directly extract pose parameters from RGB images. Dense correspondence methods [48, 49] establish correspondences between image pixels and mesh vertexes to recover poses using Perspective-n-Point (PnP) techniques. 2D-keypoint-based methods [50] detect 2D keypoints to establish the 2D-3D correspondence for pose estimation, although they may suffer from loss of geometry information due to perspective projections. A summary of these methods can be found in Table 3.4.

DeepIm [47], by Li *et al.*, uses an initial pose estimation to progressively enhance the pose accuracy through a process of matching the rendered image with the observed image. The training of the network involves learning to predict a relative Special Euclidean group in 3 dimensions (SE(3)) transformation by leveraging a disentangled representation of both 3D location and 3D orientation. This training process is conducted iteratively.

PoseCNN [3], by Xiang *et al.*, takes an RGB image as input and estimates the 3D translation and rotation of the object. The estimation of the translation is accomplished through the process of localizing the center of the object within the image and making predictions regarding its distance from the camera. On the other hand, the estimation of the rotation is achieved by regressing to a quaternion representation. The network uses a Hough voting layer to find the 2D object center and the ShapeMatch-Loss (SLOSS) function to handle symmetric objects.

Table 3.4: Summary of RGB 6D Object Detection Methods.

Method	Year	Category	Summary
DeepIm [47]	2018	Holistic	The model network takes an initial pose estimation of an object in a test image and predicts a relative SE(3) transformation that matches a rendered view of the object against the observed image.
PoseCNN [3]	2018	Holistic	The algorithm estimates the translation by localizing the objects center in the image and predicting its distance from the camera, and the rotation by regressing to a quaternion representation.
CDPN [48]	2019	Dense Correspondence	The algorithm disentangles the pose to predict rotation and translation separately. For rotation, a local region-based paradigm is used. For translation, it is directly estimated from local image patches.
EPOS [49]	2020	Dense Correspondence	The model uses an encoder-decoder network to predict correspondences between densely sampled pixels and compact surface fragments that represent the object.
PVNet [50]	2019	2D Keypoint-based	The algorithm regresses pixel-wise vectors pointing to the keypoints and uses these vectors to vote for keypoint locations, creating a flexible representation for localizing occluded or truncated keypoints.

Proposed by Li *et al.*, Coordinates-Based Disentangled Pose Network (CDPN) [48] approach untangles the pose to predict rotation and translation separately. For rotation, a well-designed local region-based paradigm is used to make the estimation more accurate and efficient. For translation, it is directly estimated from local image patches. These tasks are merged and solved in a unified network. The object size in the image can change arbitrarily along with the distance to the camera, so the object is zoomed in on to a fixed size according to the detection.

Introduced by Hodan *et al.*, Estimating Pose of Objects with Symmetries (EPOS) [49] uses an encoder-decoder network to predict correspondences between densely sampled pixels and compact surface fragments that represent the object. At every individual pixel, the network makes predictions regarding the likelihood of the presence of each object, the likelihood of the fragments given the presence of the object, and the precise 3D location on each fragment. A variable number of corresponding 3D locations, which are dependent on the data, are chosen for each pixel. The poses of potentially multiple instances of the object are then estimated using a robust and efficient variation of the PnP RANdom SAmple Consensus (RANSAC) algorithm.

Pixel-wise Voting Network [50], by Peng *et al.*, operates by performing regression on pixel-wise vectors that indicate the positions of keypoints. These vectors are then utilized to cast votes for the localization of the keypoints. This approach results in the establishment of a versatile representation that is capable of accurately localizing keypoints that may be occluded or truncated. Additionally, this method offers the ability to determine the uncertainties associated with the keypoint locations that the PnP solver can further leverage.

3.2.2 RGB-D Based Methods

The availability of depth cameras has led to an extensive investigation into estimating 6D object pose using RGB-D data. These methods can be divided into: template-based methods [51, 52], which rely on feature and shape-based template matching to locate the object in the image and coarsely estimate the pose, feature-based methods [53] that exploits point cloud to match 3D features and fit the object models into the scene, and deep learning-based methods [2, 24] that use deep learning to extract features from the RGB-D data and then use these

Table 3.5: Summary of RGB-D 6D Object Detection Methods.

Method	Year	Category	Summary
Hinterstoisser *et al.* [52]	2013	Template-based	The model uses templates of the object to detect it in the scene and estimate its pose. The templates are built from 3D models of the object using a rendering engine.
Cao *et al.* [51]	2016	Template-based	The algorithm uses a 3D model to render example poses of objects and transforms images to the Laplacian of Gaussian space to achieve invariance to illumination across an object.
Hinterstoisser *et al.* [53]	2016	Feature-based	The algorithm improves the Point Pair Feature method by proposing a novel sampling and voting schemes that reduce the influence of clutter and sensor noise.
DenseFusion [2]	2019	Deep Learning-based	The algorithm uses a heterogeneous architecture that processes the RGB and depth data separately and uses a dense fusion network to extract pixel-wise dense feature embeddings.
ShAPO [24]	2022	Deep Learning-based	The model uses a novel texture code unique to each object in the database and predicts object instance masks using a specialized head.

features to estimate the pose. A summary of these methods can be found in Table 3.5.

Hinterstoisser *et al.* [52], proposed a method that is mainly based on the LINEMOD [54] approach for object detection, which uses the templates to detect the object in the scene and estimate its pose accurately and in real-time. The method also uses color information to check the detection hypotheses and enhance the accuracy of the detection rate by 13% in comparison to the original LINEMOD.

Cai *et al.* [51] uses a 3D model to generate example poses of a textureless object and determine the closest match to the input image through the utilization of a GPU implementation. The approach transforms images to the Laplacian of Gaussian space to achieve invariance to illumination and appearance across an object. In order to facilitate real-time matching, the authors introduce an approach that involves the modification of the template set and the image, as well as the restructuring of the conventional normalized cross-correlation operation to harness the computational power of the GPU in performing rapid matrix-matrix multiplication.

In 2016 Hinterstoisser *et al.* [53] proposed some improvements to the Point Pair Features method [55]. The proposed improvements include sampling and voting schemes that reduce the influence of clutter and sensor noise. The sampling scheme selects pairs of points that are likely to belong to the same object, while avoiding pairs that are likely to belong to different objects or the background. The voting scheme aggregates the PPFs of all pairs of points that are likely to belong to the same object, while ignoring those that are likely to belong to different objects or the background.

ShAPO [24], by Irshad *et al.*, is a method that performs object-centric scene reconstruction from a single-view RGB-D observation. It infers 3D shape, 3D appearance, 6D pose, and sizes of multiple object instances. The approach employs a distinctive texture code associated with each object stored in the database and utilizes a specialized head to generate object instance masks. By means of accurate downstream optimization, the network predicts masks to ensure precision.

Table 3.6: Summary of Point Cloud 6D Object Detection Methods.

Method	Year	Description
PointVoteNet [56]	2020	The model performs point-to-point correspondence assignment, joint classification and segmentation within a point cloud system.
CloudPose [57]	2020	The algorithm uses a PointNet to extract features from point cloud segments, and directly regress to 3D rotation and 3D translation with two separate networks.
CloudAAE [58]	2021	The model uses an augmented autoencoder to learn latent embedding, which encodes object pose features.
OVE6D [59]	2022	The algorithm decomposes the 6-D pose into viewpoint, in-plane rotation, and translation, and estimates each component in a cascaded manner.

3.2.3 Point Cloud/Depth Based Methods

Recently, innovative approaches have been introduced in point cloud methods for 6D pose estimation. The summary of the following method can be found in Table 3.6.

Hagelskjaer *et al.* proposed PointVoteNet [56], which can detect and estimate the pose of rigid objects in point cloud data, using unordered point sets. This is achieved through performing point-to-point correspondence assignment joint classification and segmentation within a point cloud system.

CloudPose [57], introduced by Gao *et al.* uses supervised learning on point clouds to estimate the pose of a known 3D object. Supervised learning is a type of machine learning where an algorithm learns to map input data to output data based on a set of labeled examples. In the context of this method, the deep neural networks used for rotation and translation regression are trained using supervised learning on a dataset of known object poses and corresponding point clouds.

Later, Gao *et al.* introduced an improved method named CloudAAE [58] where rather than directly regressing to 6D pose from point clouds as CloudPose, they use the latent code from an Augmented Auto Encoder as the input to pose regressors. The method also includes a lightweight data synthesis pipeline that creates synthetic point cloud segments for training.

OVE6D [59] is a framework with a network consisting of a single shared backbone with three head branches and is trained end-to-end. The backbone extracts features from the input depth image and object mask. The three head branches estimate the viewpoint, in-plane rotation, and object translation. The network is trained using a loss function that combines three different losses with different weights.

3.3 Datasets

Numerous image datasets have been developed to facilitate the training and testing of methods. These datasets aim to portray significant scenarios and challenges that methods may face in practical applications, such as occlusion, clutter, multiple objects, similar objects, changes in lighting, street views, and indoor views. Table 3.7 presents the most commonly used datasets for 3D and 6D pose estimation.

Table 3.7: Datasets used for multiple object detection and pose estimation methods.

Dataset	**Pose Estimation**	**Data**	**Classes**	**Images**	**Environment**
Metric					
KITTI [60]	3D	RGB-D	3	14999	Street Views
SUN RGB-D [61]	3D	RGB-D	-	10335	Domestic
NYU-Depth V2 [62]	3D	RGB-D	894	1449	Domestic
PASCAL-VOC [63]	3D	RGB	12	14999	Indoor/Outdoor
RU-APC [64]	6D	RGB-D	25	10368	Warehouse
LINEMOD [52]	6D	RGB-D	12	18273	Indoor
T-LESS [65]	6D	RGB-D	30	117000	Industrial
YCB-Video [3]	6D	RGB-D	21	213827	Indoor
NOCS-REAL275 [66]	6D	RGB-D	6	300000	Indoor
JHUScene-50 [67]	6D	RGB-D	50	22520	Industrial

4

Developed Work

This chapter clarifies the 6D pose estimation methods developed (Fig. 1.5) within the context of this dissertation to fulfill the proposed objectives. All these methods are grounded in the foundational DenseFusion [2] framework, the specifics of which are detailed in Section 2.3.1. An overview of the crafted dataset designed to assess these methods performance within an industrial environment is also included.

4.1 Segmentation-based Pose Estimation Methods

This section comprises the developed pose estimation methods based on segmentation. These methodologies utilize an object segmentation technique, addressed in Section 2.3.1.1, to identify objects of interest in an image.

4.1.1 DenseFusion: Multimodal Early Fusion

This methodology proposes a modification to the DenseFusion framework, wherein the early fusion of RGB and depth information is employed to augment the accuracy and robustness of pose estimation. The primary objective of this approach is to leverage the distinct strengths of both modalities and enable the network to learn more discriminative features by jointly considering color and depth information. The goal of this fusion is to mitigate the challenges associated with occlusions, varying lighting conditions, and noise that are commonly encountered in pose estimation tasks.

To compute the early fusion approach, the architecture of DenseFusion was adapted by concatenating the RGB and depth modalities channel-wise to create a multi-channel input that retains both color and depth information. This joint input is then processed by subsequent network layers for feature extraction and pose estimation instead of performing separate feature extraction on RGB and depth images. An overview of the architecture can be seen in Fig. 4.1.

4.1.2 DenseFusion: Multimodal Middle Fusion

Similarly to the previous method, a multimodal middle fusion approach is proposed. While the early fusion method combines RGB and depth data at the input level, it may risk losing fine-grained details present in each modality. On the other hand, traditional RGB feature extraction methods often struggle with handling occlusions and varying lighting conditions. The middle

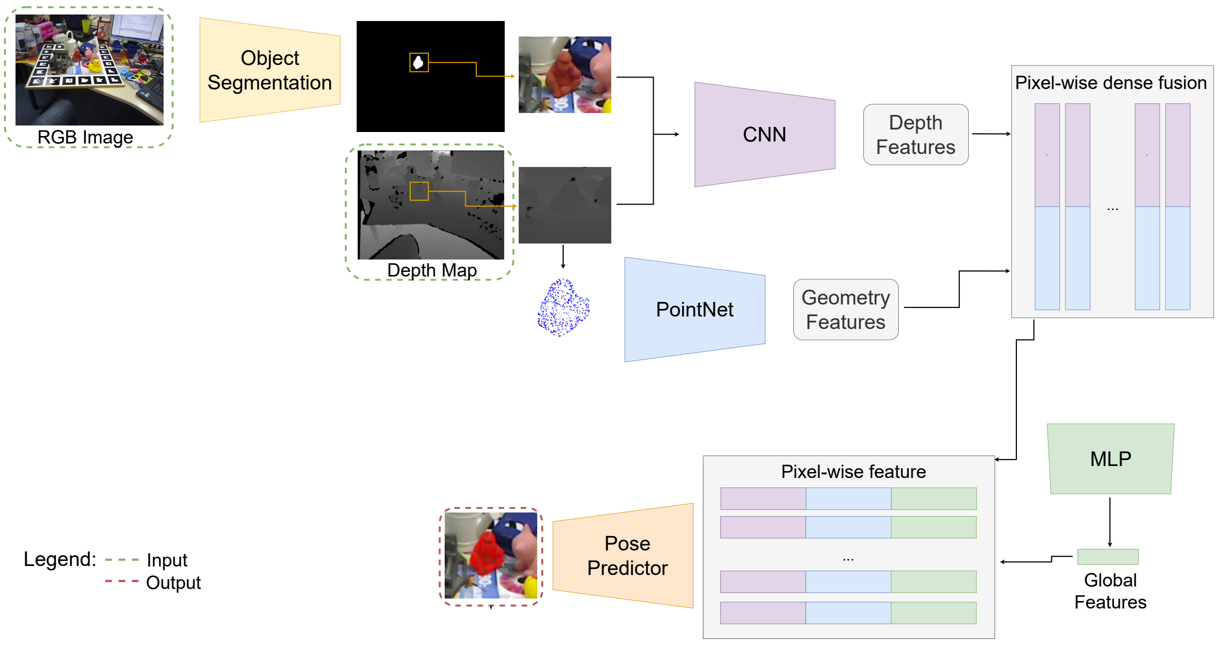

Figure 4.1: Overview of DenseFusion: Multimodal Early Fusion Architecture.

fusion with attention mechanisms approach is presented to address these challenges. Selectively fusing RGB and depth information at intermediate stages of the network using attention mechanisms, is an attempt to capture the most relevant and discriminative features from both modalities.

Attention mechanisms [68] allow the network to focus on salient features from each modality while suppressing noise and irrelevant information. This approach enables the network to decide when and how to integrate RGB and depth data. These mechanisms produce attention weights that highlight the most relevant features from each modality.

Attention computations involve calculating a weighted sum of values based on the similarity between a query and a set of key-value pairs. The most commonly used attention mechanism employs a dot product and softmax function:

$$\vec{b} = \mathrm{softmax}\left(\vec{q} \cdot \mathbf{K}\right) = \mathrm{softmax}\left(\sum_j q_j k_{ij}\right), \tag{4.1}$$

where q is the query, a vector that represents the current input position that we want to focus on, K is the keys, a set of vectors representing all positions in the input sequence, index i is the position in the sequence and j is the index of the feature. The softmax function is defined as:

$$\mathrm{softmax}\left(\vec{x}\right) = \frac{e^{\vec{x}}}{\sum_i e_i^x}, \tag{4.2}$$

and ensures that $\vec{b}$ is normalized. Following the computation of $\vec{b}$, the weighted mean is calculated

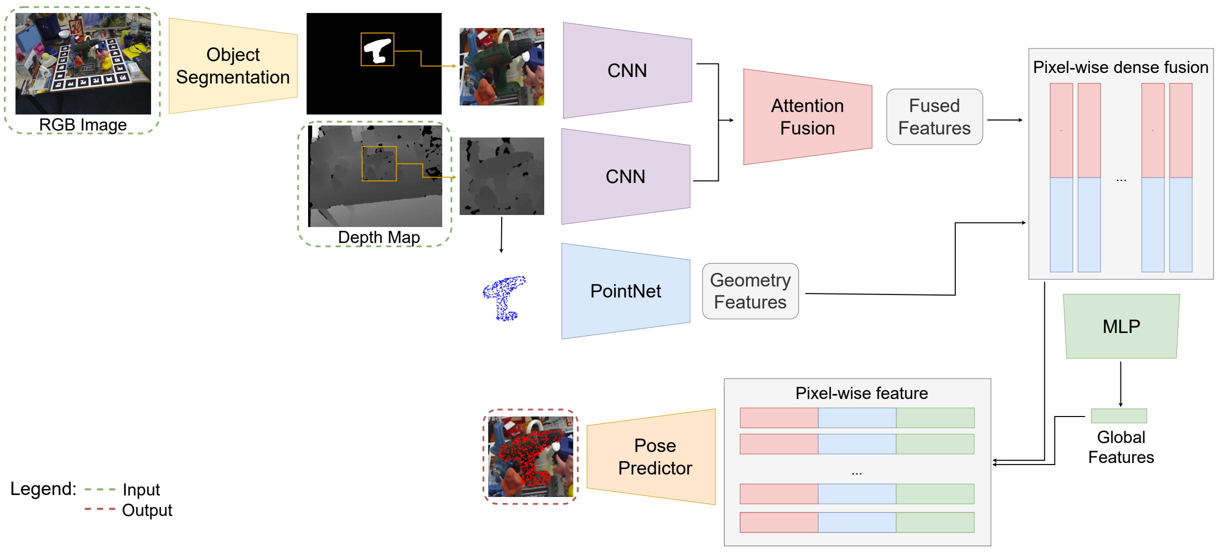

Figure 4.2: Overview of DenseFusion: Multimodal Middle Fusion Architecture.

as:

$$V \cdot \vec{b}, \tag{4.3}$$

where V is the values, a set of vectors containing information at each position in the input sequence.

Multi-head attention, used in this approach, extends the single-head attention concept by letting the model focus simultaneously on different parts of the input sequence. Instead of having a single set of learnable parameters for keys, queries, and values, multi-head attention introduces multiple sets (heads) of these parameters. Each head independently computes an attention mechanism, resulting in multiple output representations. Considering an attention layer defined by $A(\vec{q}, \mathbf{K}, \mathbf{V})$. The multi-head attention is expressed as:

$$\left[A(\mathbf{W}_q^0\vec{q}, \mathbf{W}_k^0\mathbf{K}, \mathbf{W}_v^0\mathbf{V}), A(\mathbf{W}_q^1\vec{q}, \mathbf{W}_k^1\mathbf{K}, \mathbf{W}_v^1\mathbf{V}), \ldots, A(\mathbf{W}_q^H\vec{q}, \mathbf{W}_k^H\mathbf{K}, \mathbf{W}_v^H\mathbf{V})\right]. \tag{4.4}$$

An overview of the architecture can be seen in Fig. 4.2. Two variations of this approach were employed, differing in the dimensionality of the attention module, which determines the dimensionality of the input and output embeddings in the sequence. For ease of comparison, we shall designate these as ”DenseFusion: Multimodal Middle Fusion 1” for the version with lower dimensionality, and ”DenseFusion: Multimodal Middle Fusion 2” for the version featuring higher dimensionality. Some alternative iterations of this approach incorporating attention mechanisms were explored. However, these variations yielded unsatisfactory preliminary outcomes that do not justify specific mention.

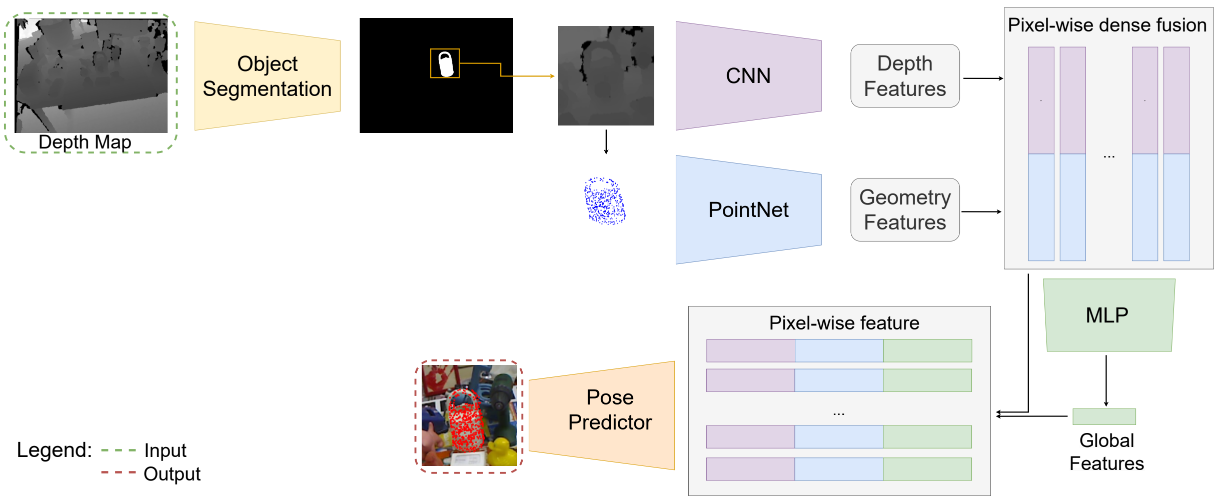

Figure 4.3: Overview of DenseFusion: Depth Architecture.

4.1.3 DenseFusion: Depth

In this section, a method that relies on only depth information is proposed. This approach offers some benefits to the field of computer vision and its applications. By focusing solely on depth data, this method provides a solution that surpasses some challenges associated with traditional techniques that rely on both color and depth information. The exclusive dependence on depth information eradicates the potential shortcomings that can arise from variations in color due to changes in lighting, occlusions, or other environmental factors.

The key idea was to fuse the strengths of DenseFusion while eliminating the reliance on RGB data. The original pipeline includes three main stages: feature extraction, feature matching, and pose refinement. This method retains these stages, but adapts them to accommodate depth information exclusively. An overview of the architecture can be seen in Fig. 4.3.

4.1.4 DenseFusion: Point Cloud

Like the previous one, this method relies exclusively on depth information, specifically the geometry features extracted from the point cloud generated by the depth map, using only PointNet to extract features. The method streamlines the data processing pipeline and reduces computational demands by forgoing the need for RGB data and focusing only on the point cloud. This efficiency translates to a faster and more responsive pose estimation algorithm. An overview of the architecture can be seen in Fig. 4.4.

4.2 Detection-based Pose Estimation Methods

This section comprises the developed Pose Estimation Methods based on detection. These methodologies utilize the Yolov5 [11] network to identify objects of interest in an image. Section 2.1.2 provides a detailed explanation of this object detector.

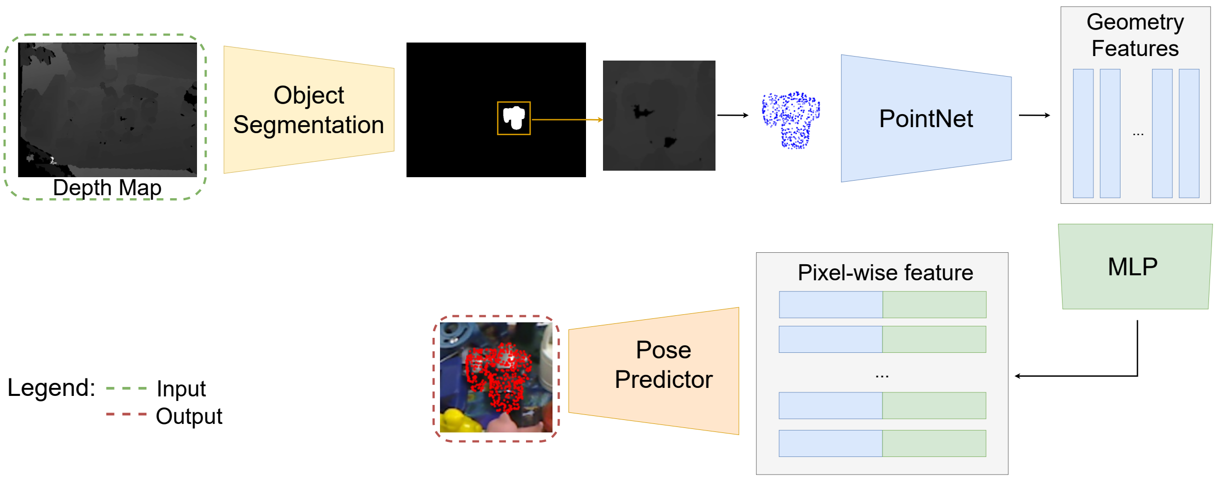

Figure 4.4: Overview of DenseFusion: Point Cloud Architecture.

4.2.1 DenseFusion: Point Cloud

Most 6D Pose Estimation methods available online use data from RGB or RGB-D cameras. These sensors have some limitations, such as uneven lighting, reflections, and occlusions, that often cause object detection accuracy to suffer. RGB-D cameras attempt to overcome these limitations by incorporating depth data to improve scene understanding. However, they also have their own constraints, such as struggling with transparent or reflective surfaces experiencing accuracy degradation in low-light conditions.

An alternative approach to RGB-based methods is to use lasers, LiDAR, or similar technologies to generate a 3D representation of the environment. A method that uses the data of these types of sensors can be more accurate in detecting objects, regardless of lighting conditions or surface properties, transcending the shortcomings of RGB-based approaches. This is especially crucial in industrial settings, where reliable object detection can directly impact operational efficiency and worker safety. With that in mind, a method using only LiDAR-like sensors information was developed. An overview of the architecture can be seen in Fig. 4.5.

4.2.1.1 Depth Map Projection from Point Cloud

Before diving into how the projection is made, we need to understand the calibration matrices that KITTI [60] provides. The camera projection matrix, often denoted as P, is a 3x4 matrix that maps 3D world coordinates to 2D image coordinates. It plays a central role in converting point cloud data to the image plane. Given calibration data, the P matrix is constructed and augmented to a 4x4 matrix:

$$P = \begin{bmatrix} P_{3\times 4} & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.5}$$

The rectification matrix, represented by R, is a 3x3 transformation matrix that rectifies stereo images, a process to merge images taken from multiple perspectives into a common map coordinate system. Like P, it is expanded to a 4x4 matrix:

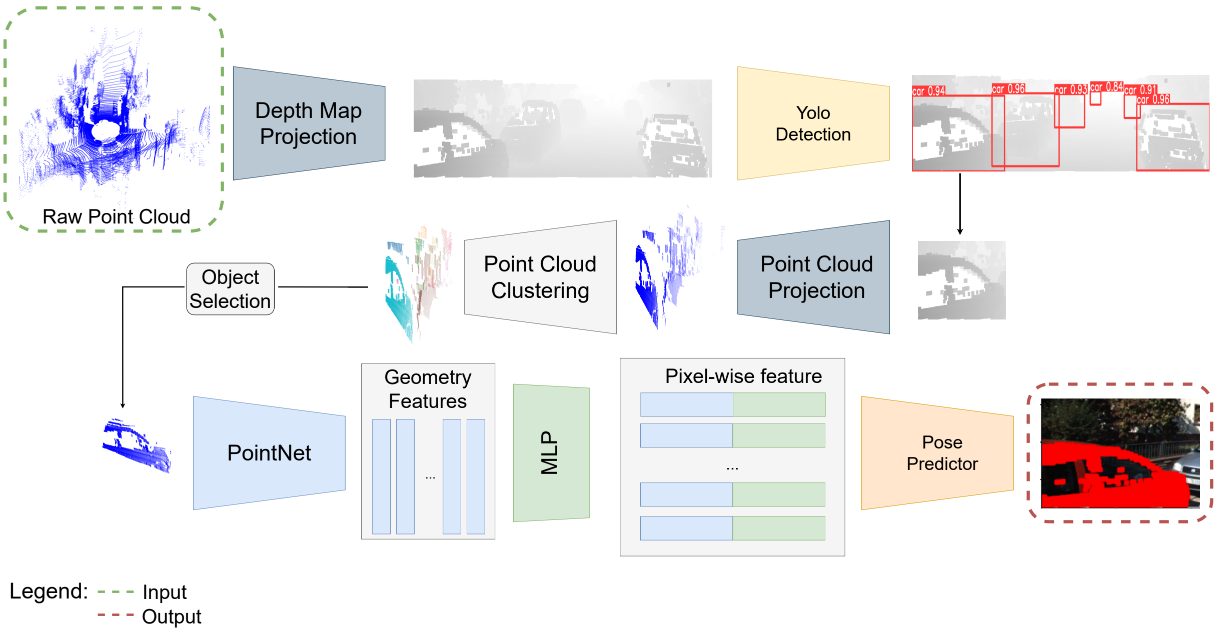

Figure 4.5: Overview of DenseFusion: Point Cloud Architecture.

$$R = \begin{bmatrix} R_{3\times3} & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.6}$$

The transformation matrix Tr converts the Velodyne LiDAR coordinates to camera coordinates.

$$Tr = \begin{bmatrix} Tr_{3\times4} & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.7}$$

We can now obtain the calibration matrix, Clb, by chaining the transformations as follows:

$$Clb = P \times R \times Tr. \tag{4.8}$$

The 3D point cloud is converted into a depth map by following the equation:

$$\begin{bmatrix} z_c \cdot v \\ z_c \cdot u \\ z_c \\ 1 \end{bmatrix} = Clb \times \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1, \end{bmatrix} \tag{4.9}$$

where u and v represent two coordinates of a given point within the depth map, z_c denotes the relevant depth value, and (x_w, y_w, z_w) represents the coordinates within the world coordinate system.

Given the inherent sparsity of point cloud data, it becomes crucial to enhance its density to facilitate more effective feature extraction. To achieve this, a bilateral filter [69] was employed.

This filter incorporates both background and foreground points projected onto the image plane, generating a denser representation of the point cloud.

The depth-mapping process involves a formulation that capitalizes on local spatial interpolation and leverages the Bilateral Filter technique. This fusion allows for accurate depth-map generation, which is essential in applications where understanding scene geometry is crucial.

Local spatial interpolation enables the estimation of range measurements in the depth map's sampled and unsampled areas. This interpolation is localized within a defined region or window, with the point of interest serving as the central reference. The primary objective is to predict the range-distance value at this central location based on a finite set of observed points situated within the region.

Incorporating the sliding window technique further refines local spatial interpolation. This methodology entails systematically shifting a window or mask across the depth map, centering it on the location of interest. The window's dimensions dictate the scope of neighboring points considered for the interpolation process. By adopting the sliding window technique, the algorithm effectively infers range measurements at unsampled positions, utilizing the values of proximate sampled points.

Central to this formulation is the Bilateral Filter, a technique used in image processing and computer vision to smooth images while preserving edges and details. The strength of the Bilateral Filter lies in its dual consideration of spatial proximity and intensity resemblance between pixels. The bilateral Filter's application is modified to enable depth-map upsampling. This adaptation emphasizes the preservation of edges and foreground-background disparities, ensuring that the resulting upsampled depth map retains crucial scene delineations.

4.2.1.2 Point Cloud Projection from Depth Map

The depth map is converted into a 3D point cloud by computing the inverse of the projection of a depth map from a point cloud:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = Clb^{-1} \times \begin{bmatrix} z_c \cdot v \\ z_c \cdot u \\ z_c \\ 1 \end{bmatrix}, \tag{4.10}$$

4.2.1.3 Object Selection

Once the output clusters are acquired, the subsequent task involves the strategic selection of a specific cluster that corresponds to the object of interest. The criteria for this selection are: density and proximity. The density of a cluster is estimated by the number of core points it comprehends, indicating a high concentration of data points in that region. Meanwhile, the proximity factor determines how close the cluster is to the sensor. In practice, selecting the cluster with the highest density and closest to the sensor ensures that the chosen cluster is the desired object.

Algorithm 2: K-means Clustering

```
Data: Input data $X$, number of clusters $k$
Result: Cluster assignments $C$
Initialize centroids $c_1, c_2, \dots, c_k$ randomly;
repeat
    foreach data point $x_i \in X$ do
        Assign $x_i$ to the nearest centroid: $c_i = \arg\min_j \text{dist}(x_i, c_j)$;
    end
    foreach centroid $c_j$ do
        Update centroid $c_j$ as the mean of assigned points: $c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$;
    end
until convergence;
```

4.2.1.4 Upsampling and Downsampling of the Objects

The object chosen from the point cloud clustering contains a specific count of points corresponding to the number of pixels depicting the object in the image. However, for the network's pose estimator to operate effectively, it's essential to maintain consistent object dimensions. Therefore, it becomes imperative to ensure uniform object sizes. In cases where the object exhibits a lower number of points than the target dimension, an upsampling procedure is necessary. Contrariwise, downsampling is required when the object comprises an excessive number of points. This process of upsampling and downsampling guarantees that the network's pose estimation functions optimally by standardizing object dimensions across the dataset.

For downsampling, the goal is to reduce the number of points while retaining the essential features of the point cloud. To do so, the K-means clustering algorithm 2 is applied with the desired number of clusters set to the desired number of points. The algorithm assigns each point in the cluster to one of these K clusters. The cluster centers are then computed and serve as the downsampled points for the point cloud.

For upsampling, the goal is to increase the number of points while preserving the overall distribution and characteristics of the point cloud. The number of points needed to reach the desired number of points is calculated by subtracting the current number of points in the object point cloud from the desired number. To generate new points, the algorithm employs random oversampling. This involves drawing random samples from the existing points in the point cloud. The generated new points are stacked (vertically) on top of the existing points in the point cloud.

4.2.1.5 Loss Function

The original approach employs 3D model point clouds of objects to calculate the positional disparity between the ground truth and the predicted pose. This involves applying the transformation poses of both the ground truth and the pose prediction to the 3D model. These transformations position the identical point cloud in distinct locations, enabling a point-to-point distance computation between the two configurations. Nevertheless, in the context of detection-based pose estimation methods, the datasets employed do not provide 3D object models. This means that the point cloud derived from detection and the point cloud representing the ground truth will differ. As a solution, the Chamfer distance emerges as a suitable metric for calculating the dissimilarity between these distinct point clouds.

The Chamfer distance is a metric used to measure the similarity between two point clouds by calculating the average nearest neighbor distance between points in one point cloud to the other point cloud. Representing the Chamfer distance between two 3D point clouds involves computing the Euclidean distances for each point in one point cloud to its nearest neighbor in the other point cloud and then averaging these distances. The Chamfer distance between two sets of points S_1 and S_2 is defined as:

$$CD(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2. \tag{4.11}$$

The pose estimation loss becomes:

$$L = \frac{1}{N} \sum_{i=1}^{N} (CD_i c_i - w \log(c_i)), \tag{4.12}$$

where N is the number of points sampled from the segmented object, c_i is the confidence coefficient, and w is incorporated as a secondary regularization term to balance the average distance loss and confidence.

4.2.1.6 Prediction Selection

As previously described, DenseFusion generates a set of N_{points} predictions, each with a confidence score associated. This method chooses the highest confidence score prediction as the ultimate pose prediction. However, a novel approach to prediction selection is introduced. Rather than solely relying on the single highest-confidence prediction, this new strategy considers all or a subset of the top predictions. The rationale lies in recognizing that the prediction with the utmost confidence may not invariably be the most accurate. A weighted average of the pose predictions is computed to address this, effectively combining the top predictions into a final, refined prediction. This technique aims to enhance the reliability and robustness of the pose estimation process.

However, averaging quaternions can be challenging because quaternion space is not linear, and simple averaging can lead to undesirable results. The algorithm 3 introduced by *Markley et al.* [70] aims to find the quaternion that represents the average rotation of a set of quaternions. This is particularly useful when we have a collection of rotations and want to find a smooth, interpolated rotation that is as close as possible to the average of the given rotations. The basic idea behind the algorithm is to represent the quaternions in a higher-dimensional space and then perform the averaging in that space. The algorithm involves finding the eigenvalues and eigenvectors of a matrix derived from the input quaternions. The eigenvector corresponding to the largest eigenvalue is then used to compute the averaged quaternion. This method tends to produce better results than simple linear averaging of quaternions, especially when dealing with rotations that are not close to each other in quaternion space. It helps avoid issues like gimbal lock and produces a more meaningful average rotation.

Algorithm 3: Quaternion Weighted Average (Markley)

```
Input:
    $Q$: an $M \times 4$ ndarray of quaternions.
    $weights$: a list of $M$ elements, a weight for each quaternion.
Output:
    An averaged quaternion.
Initialize:
$A$ as a $4 \times 4$ matrix of zeros;
$M$ as the number of rows in $Q$;
$wSum$ as 0;
for $i = 1$ to $M$ do
    $q = Q[i, :]$;
    $w_i = \text{weights}[i]$;
    $A = A + w_i \cdot (\mathbf{q} \otimes \mathbf{q}^\intercal)$;
    $wSum = wSum + w_i$;
Scale $A$ by dividing by $wSum$;
$A = A/wSum$;
Get the eigenvector corresponding to the largest eigenvalue of $A$;
$\text{eigenvalues}, \text{eigenvectors} = \text{eig}(A)$;
$\text{avgQuaternion} = \text{eigenvectors}[:, -1]$;
return $avgQuaternion$;
```

The translation vector follows a normal weighted average computed as:

$$T = \frac{\sum_{i=1}^{N} t_i \cdot c_i}{\sum_{i=1}^{N} c_i} \tag{4.13}$$

where N are the number of predictions considered, r and t are the rotation and translation of the prediction, and c the corresponding confidence. The final pose prediction is $[R|T]$.

4.3 Industrial Dataset

In pursuing the objectives of this dissertation, a comprehensive evaluation of existing methods in an industrial setting is essential. This requires a careful examination of their performance in such environments and an assessment of their robustness. Of course, this evaluation requires the availability of a suitable dataset.

This dataset was created to fulfill the project's prerequisites associated with the present dissertation. This necessity was originated from the absence of a realistic and readily available dataset online for validating of the accuracy of the detection of pallets within a warehouse setting. For instance, consider the PalLoc6D dataset [12], which serves as an RGB-D virtual dataset for the 6D detection of pallets. However, it lacks a realistic scenario because the pallets are generated randomly in various locations, surrounded by random objects, within a randomized background.

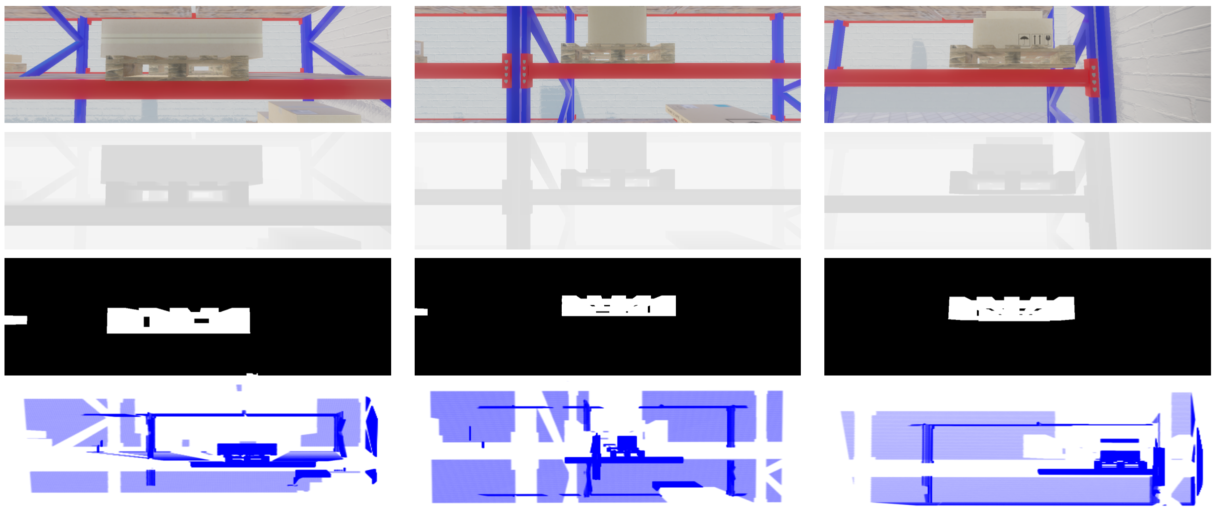

Figure 4.6: Example of rendered RGB-D images, and respective Masks and Point Clouds from Industrial Dataset.

The key idea revolves around an autonomous forklift capable of navigating towards designated pick-up and drop-off zones. Once positioned correctly, the robot must accurately identify the pallet's location, enabling seamless execution of the loading and unloading processes. To achieve this objective, the dataset simulates a virtual warehouse environment, consisting of carefully designed shelves populated with pallets and boxes, capturing data from the perspective of a mobile robot.

The resulting industrial dataset includes an extensive collection of 816 RGB-D raw images with a resolution of 1224x370, along with the corresponding point clouds, 2D and 3D bounding box annotations for every pallet object within the image, as well as the essential calibration matrices.

5

Software Tools and Hardware Materials

This chapter provides a brief overview of the various software tools and libraries used to develop the 6D object detection and classification system for this study.

5.1 Operating System

Ubuntu [71] is a free and open-source operating system that offers unlimited development privileges. It also has official long-term support, continuous maintenance, and a built-in firewall that mitigates potential security risks. The decision to use Ubuntu as the operating system for the study was primarily motivated by the observation that most online object detection methods have been developed and made available in this particular environment.

5.2 Python

Python [72], namely version 3.8, was the main programming language because it is a popular choice for machine learning and computer vision tasks. The packages implemented in developing the proposed pipeline can be found in Table 5.1.

5.2.1 Pytorch

PyTorch is a well-known open source machine learning library. It builds and trains deep learning models for object recognition and classification. PyTorch is based on the Torch library

Table 5.1: Python packages

Package	Version	Description
Pytorch [73]	1.7.1	Machine learning framework
Numpy [74]	1.22.0	Package for computation of arrays, matrices, and linear algebra
Matplotlib [75]	3.7.0	Library for creating static, animated, and interactive visualizations
Scikit-learn [76]	1.2.2	Data analysis library
Opencv [77]	4.5.2	Library that allows you to perform image processing and computer vision tasks
Ultralytics [78]	7.0.0	Package that provides YOLO models

Table 5.2: NVIDIA GeForce RTX 3060 specifications

Card Length	224mm x 116mm
CUDA Cores	3584
Video Memory	12GB GDDR6
Memory Bus	192-bit
Engine Clock	Boost: 1777 MHz
Memory Clock	15 Gbps
Power Consumption	170W
Supported OS	Windows, Linux

and provides effortless integration with Python. It also enables computations on the GPU using CUDA [79]. These tools are widely used for computer vision and DL in academia and industry. Consequently, PyTorch was a suitable option for developing a system for detecting 6D objects.

5.3 Conda

Conda [80] is an efficient package and environment management system widely used in code development due to its many advantages. It allows developers to create isolated environments so that each project gets its packages and dependencies without affecting other projects. This feature mitigates library conflicts and ensures consistent code execution on different systems. In addition, Conda offers a wide range of pre-built packages, including popular libraries for data science and machine learning, so developers can quickly set up their programming environments without installing packages manually. Conda has become an indispensable tool for developers who want to streamline code development and ensure the consistency of their code across multiple platforms.

5.4 Evaluation Setup

The study was conducted using a combination of hardware components, including an NVIDIA GeForce RTX 3060 [81] GPU and an AMD Ryzen™ 7 3800X 8-core processor Central Processing Unit (CPU), along with 64 GB of RAM. The key specifications of theGPU are outlined in Table 5.2.

6

Results and Discussion

This chapter unveils the performance of the proposed Pose Estimation methods and a comparative analysis of them.

6.1 Segmention-based Pose Estimation Methods

The evaluation of segmentation-based pose estimation methods will take place within the same dataset used for *DenseFusion*. This approach enables a direct comparison between the introduced techniques and the original method's performance. The chosen dataset, LINEMOD, is one of the most widely used benchmark datasets for 6D pose estimation, encompassing 13 objects (ape, bench vi, camera, can, cat, driller, duck, eggbox, glue, hole p., iron, lamp and phone) with minimal textures across a collection of 15,783 RGB-D images. Given the real-world acquisition of these images, challenges abound in the form of occlusions, limited textures, and lighting disparities within cluttered environments. It is noteworthy that the partitioning of training and test sets remains consistent with official dataset guidelines.

6.1.1 Evaluation Metrics

This section will delve into evaluation metrics used in the LINEMOD dataset: Average Distance of Keypoints (ADD) and Average Distance of Keypoints with Symmetry (ADD-S). These metrics offer a quantitative means of the quality of pose estimation outcomes.

6.1.1.1 Average Distance (ADD)

This metric measures the average Euclidean distance between predicted keypoint positions $(\hat{R}, \hat{T})$ and their corresponding ground truth positions (R, T). The final ADD score is obtained by averaging these distances over all keypoints, with a lower score indicating greater accuracy of the pose estimation algorithm, which is computed as follows:

$$ADD = \frac{1}{M} \sum_{x \in M} \|(Rx + T) - (\hat{R}x + \hat{T})\|, \tag{6.1}$$

where x represents one of the M sampled points belonging to the 3D model, x denotes one of the M points of the model, R and T are the ground truth rotation and translation matrices, respectively. And $\hat{R}$ and $\hat{T}$ are the predicted rotation and translation matrices, respectively. This metric can serve as a tool for both a loss function and an accuracy measurement. Predictions that register a score lower than a predetermined threshold are deemed correct. Generally, this threshold is established in the following manner:

$$\text{ADD} \leq k_m d. \tag{6.2}$$

6.1.1.2 Average Distance of Keypoints with Symmetry (ADD-S)

The ADD-S metric is an extension of the ADD metric that considers the symmetrical nature of certain objects. This is particularly relevant in human pose estimation, where certain keypoints possess symmetrical counterparts. If the ADD metric were used exclusively, wrongly predicted keypoints would be penalized, even if they were symmetric. To solve this problem, the ADD-S metric introduces a symmetrization step that verifies the correct placement of the predicted keypoint, based on a known symmetry transformation. In a misplaced keypoint, the metric identifies the corresponding symmetric keypoint on the opposite side and calculates the Euclidean distance between the predicted and ground truth keypoints. This adjustment guarantees that symmetric keypoints are evaluated impartially, which is computed as follows:

$$\text{ADD-S} = \frac{1}{M} \sum_{x_1 \in M} \min_{x_2 \in M} \|(Rx_1 + T) - (\hat{Rx_2} + \hat{T})\|. \tag{6.3}$$

6.1.2 Multimodal Study

In this Section, the performance of the proposed multimodal methods on the LINEMOD dataset is compared. Table 6.1 presents a comprehensive analysis of the performance of different multimodal methods for object detection and pose estimation on the LINEMOD dataset. The method based only on the geometric features achieves an accuracy really close to the original method, separated by only 0.5%, even though not provided with the RGB data. The *DenseFusion: Depth* disappoints slightly, with an accuracy of only 90.0%. This poor performance compared to *DenseFusion: Point Cloud* might be because using both depth image and point cloud features could introduce redundancy in the information. If the information extracted from the depth image is already present in the point cloud, combining them might not provide any additional benefit and introduce noise or inconsistencies. Interestingly, the inclusion of depth information proves beneficial for certain objects. Objects like *ape*, *bench vi*, *can*, *cat*, *driller*, and*lamp* exhibit improved accuracy scores when the *DenseFusion: Point Cloud* variant is employed. This suggests that depth data provides crucial information for these objects, enabling more accurate pose estimation.

In addition, to verify the robustness of the model for the different objects, the plot of the accuracy-threshold curve of ADD(-S) and box plots of ADD(-S) accuracy for the different multimodal methods is made, as shown in Figs. 6.1 and 6.2 respectively.

In Fig. 6.1, the threshold value is changed for evaluating ADD(-S) and calculating the corresponding accuracy of the estimated pose. The maximum threshold value is set to 10% of

Table 6.1: Accuracy (ADD(-S)) of the multimodal methods on the LINEMOD dataset. Objects in bold are symmetric.

	DenseFusion	**DenseFusion: Depth**	**DenseFusion: Point Cloud**
ape	89.3	77.8	**90.5**
bench vi	93.9	86.6	**94.6**
camera	**95.0**	84.5	89.4
can	95.7	89.2	**96.1**
cat	97.3	93.2	**97.4**
driller	91.6	89.1	**94.9**
duck	**92.7**	79.3	89.1
eggbox	**99.9**	99.8	**99.9**
glue	**99.7**	**99.7**	99.4
hole p.	**94.5**	90.4	93.5
iron	**97.2**	91.3	93.8
lamp	96.3	94.2	**97.7**
phone	**96.7**	94.4	95.9
Mean	**95.3**	90.0	94.8

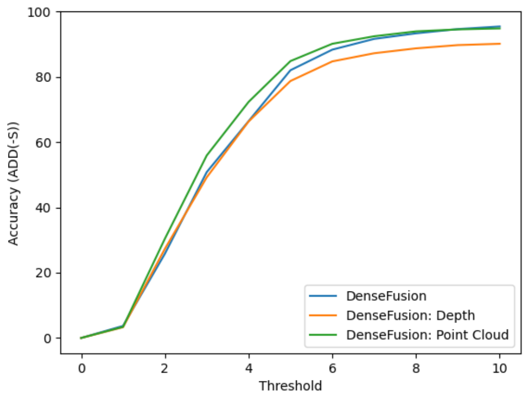

Figure 6.1: The accuracy-threshold curve of the multimodal methods on the LINEMOD dataset.

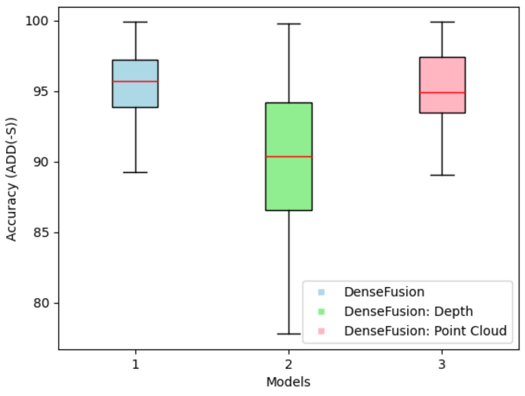

Figure 6.2: Box plot of the accuracy of the multimodal methods on the LINEMOD dataset.

the 3D model diameter. The graphic shows that *DenseFusion: Point Cloud* excels particularly at lower thresholds, reflecting enhanced robustness and stability in its performance.

Moving to Fig. 6.2, a set of three box plots corresponds to the methods enumerated in Table 6.1. Each box plot encapsulates five key numerical indicators: the upper whisker (depicted as a black horizontal line above the box), upper quartile (top edge of the box), median (highlighted by a red line within the box), lower quartile (bottom edge of the box), and lower whisker (black horizontal line below the box). Both *DenseFusion* and *DenseFusion: Point Cloud* exhibit a similar and small range between the upper and lower whisker, and the box height is relatively narrow in both, indicating a concentrated distribution of accuracy.

For a more tangible understanding, qualitative results of the multimodal methods are presented in Fig. 6.3.

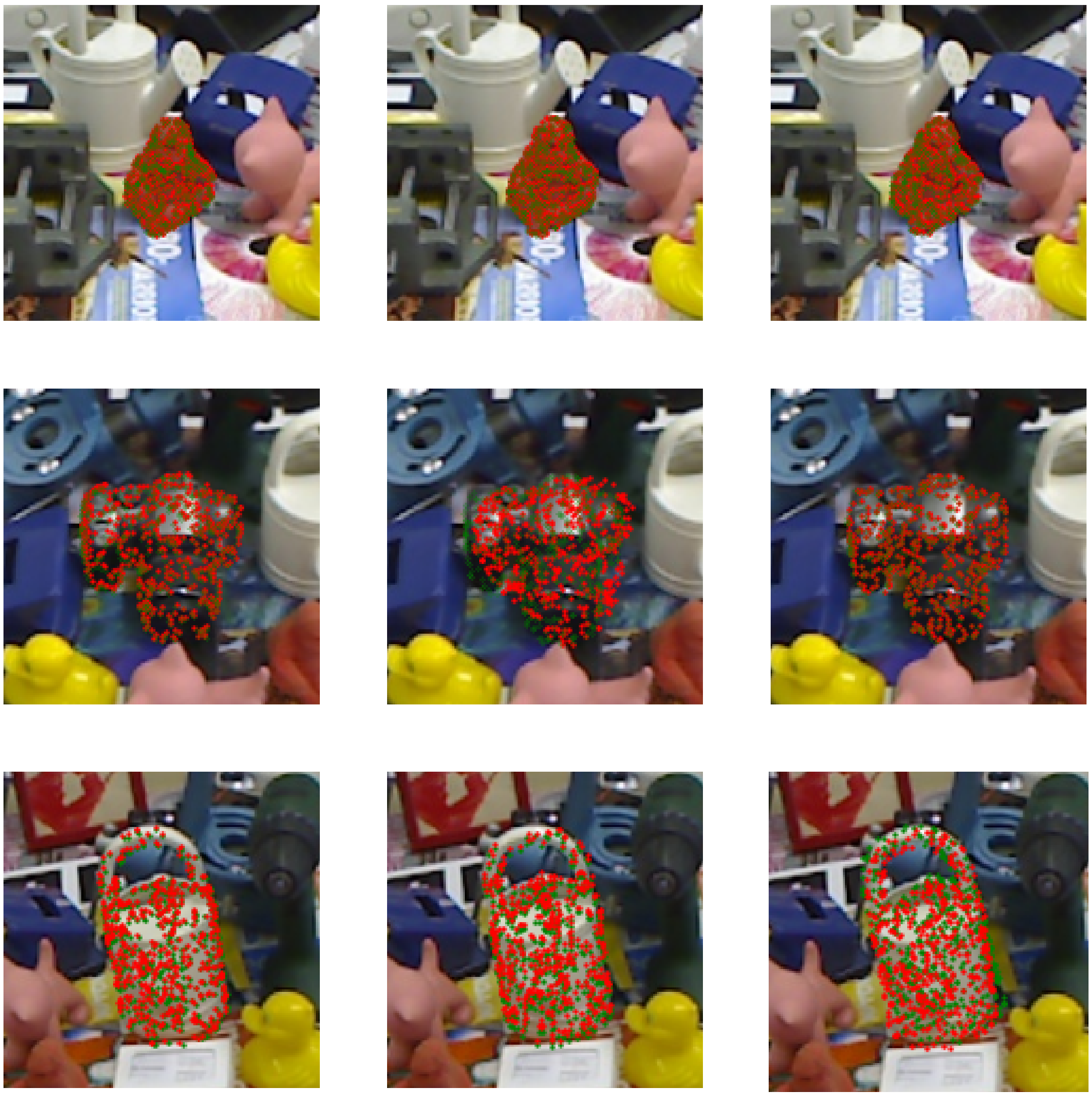

Figure 6.3: Visualization of the multimodal methods on the LINEMOD dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results of DenseFusion, DenseFusion: Depth, and DenseFusion: Point Cloud, respectively.

6.1.3 Feature Extraction Variations

This section assesses the method performances using various techniques for feature extraction from the input data. Table 6.2 clearly demonstrates that all the introduced methods increase the performance of the original one. *DenseFusion: Middle Fusion 2* stands out with the most remarkable performance. This variant enhances the overall method performance by 1.9 percentage points, showcasing the highest accuracy across nearly all objects within the dataset. By comparing *DenseFusion: Middle Fusion 1* and *DenseFusion: Middle Fusion 2*, it becomes evident that augmenting the dimensionality of the attention model leads to a notable improvement in overall accuracy.

Figure 6.4 exhibits that the variations of *DenseFusion: Middle Fusion* are way more robust and stable compared to the *DenseFusion: Early Fusion* and *DenseFusion* that behave similarly.

Figure 6.5 contains four box plots, corresponding to the methods in Table 6.2. The circles

Table 6.2: Accuracy (ADD(-S)) of methods with feature extraction variations on the LINEMOD dataset. Objects in bold are symmetric.

	DenseFusion	**DenseFusion: Early Fusion**	**DenseFusion: Middle Fusion 1**	**DenseFusion: Middle Fusion 2**
ape	89.3	91.1	95.2	**96.5**
bench vi	93.9	**95.2**	94.8	**95.2**
camera	95.0	95.2	98.6	**98.9**
can	95.7	**96.9**	95.3	96.2
cat	97.3	97.3	97.6	**99.2**
driller	91.6	94.2	93.2	**97.5**
duck	92.7	95.8	**96.0**	95.4
eggbox	99.9	**100.0**	97.1	96.5
glue	**99.7**	99.6	96.5	97.0
hole p.	94.5	93.2	95.1	**97.1**
iron	97.2	96.5	**99.6**	98.2
lamp	96.3	96.9	97.7	**98.4**
phone	96.7	96.3	97.8	**98.3**
Mean	95.3	95.8	96.5	**97.2**

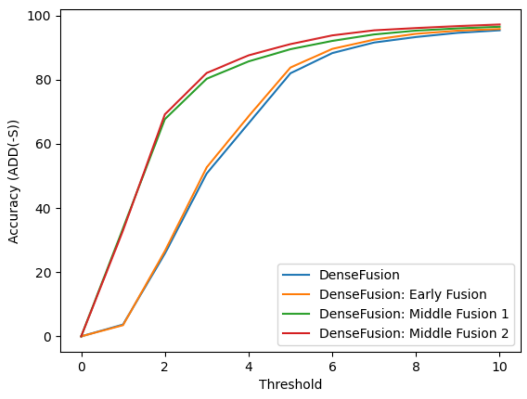

Figure 6.4: The accuracy-threshold curve of the methods with feature extraction variations on the LINEMOD dataset.

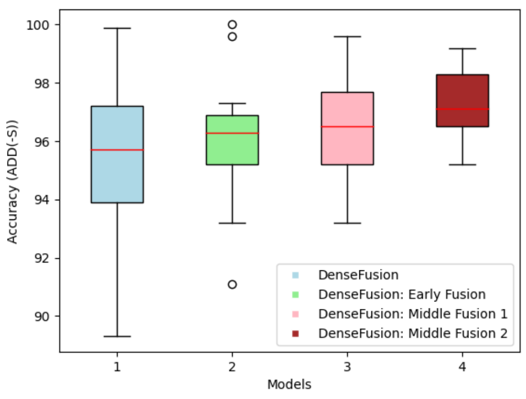

Figure 6.5: Box plot of the accuracy of the methods with feature extraction variations on the LINEMOD dataset.

in the graphic represent outliers, data points that are located outside the whiskers of the box plot. *DenseFusion: Early Fusion* is the only method with outliers indicating a more unstable behavior when trying to estimate the pose of different objects. *DenseFusion: Middle Fusion 2* has the smallest range between the upper and lower whisker, and the box height is relatively narrow, indicating that the accuracy distribution is relatively concentrated.

For a more visual understanding, qualitative results of the multimodal methods are presented in Fig. 6.6.

6.1.4 Comparative Study with State-of-art

From Tables 6.3, 6.4, and 6.5, and Fig. 6.7, it becomes evident that all the proposed techniques, with the exception to *DenseFusion: Depth*, exhibit enhanced performance when

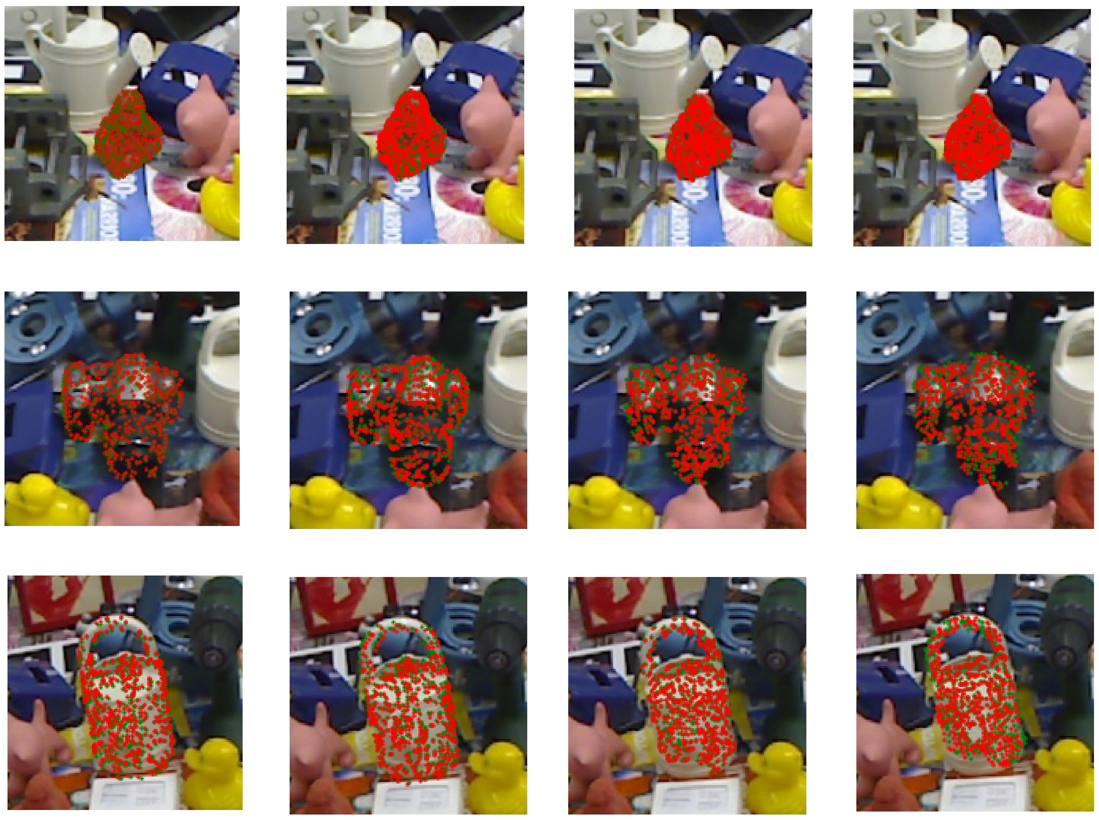

Figure 6.6: Visualization of the methods with feature extraction variations on the LINEMOD dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Four columns show the results of DenseFusion, DenseFusion: Early Fusion, DenseFusion: Middle Fusion 1, and DenseFusion: Middle Fusion 2, respectively.

evaluated on LINEMOD in comparison to the current state-of-the-art approaches that solely rely on RGB data for pose estimation. Delving into the realm of RGB-D techniques, the performance of *DenseFusion: Middle Fusion 2* emerges as strikingly competitive, nearly outpacing the state-of-the-art benchmarks. Shifting the focus to Depth-based methodologies, *DenseFusion: Point Cloud* surpasses the existing state-of-the-art techniques within this category, achieving the best performance for most object classes. This advancement in accuracy amounts to a notable increase of 2.3 percentage points. The methods proposed in this dissertation were trained for 100 epochs due to limited computational resources and time. This is less than the 500 epochs used to train state-of-the-art methods. However, the introduced methods still achieved a very good performance. This suggests that they have the potential to be even more effective with more training time.

6.2 Detection-based Pose Estimation Methods

The detection-based pose estimation methods will be evaluated on the KITTI dataset. This dataset is commonly used for 3D object detection, but it is possible to use it for 6D object detection and is well-supported by the research community. It is also relatively simple to work with and provides LiDAR data, which is necessary for evaluating models that only use point clouds as input data. Comprising a comprehensive collection of labeled images and point cloud

Table 6.3: Accuracy (ADD(-S)) of RGB state-of-art methods on the LINEMOD dataset. Objects in bold are symmetric

	DeepIM [47]	**CDPN** [48]	**HybridPose** [82]
ape	77.0	64.4	**77.6**
bench vi	97.5	97.8	**99.6**
camera	93.5	91.7	**95.9**
can	**96.5**	95.9	93.6
cat	82.1	83.8	**93.5**
driller	95.0	96.2	**97.2**
duck	77.7	66.8	**87.0**
eggbox	97.1	**99.7**	99.6
glue	99.4	**99.6**	98.7
hole p.	52.8	85.8	**92.5**
iron	**98.3**	97.9	98.1
lamp	97.5	**97.9**	96.9
phone	87.7	90.8	**98.3**
Mean	88.6	89.9	**94.5**

Table 6.4: Accuracy (ADD(-S)) of RGB-D state-of-art methods on the LINEMOD dataset. Objects in bold are symmetric

	DenseFusion	**DenseFusion: Early Fusion**	**DenseFusion: Middle Fusion 1**	**DenseFusion: Middle Fusion2**	**MaskedFusion** [83]	**FFB6D** [84]
ape	89.3	91.1	95.2	96.5	92.2	**98.4**
bench vi	93.9	95.2	94.8	95.2	98.4	**100.0**
camera	95.0	95.2	98.6	98.9	98.0	**99.9**
can	95.7	96.9	95.3	96.2	97.4	**99.8**
cat	97.3	97.3	97.6	99.2	97.8	**99.9**
driller	91.6	94.2	93.2	97.5	95.6	**100.0**
duck	92.7	95.8	96.0	95.4	94.0	**98.4**
eggbox	99.9	**100.0**	97.1	96.5	99.6	**100.0**
glue	99.7	99.6	96.5	97.0	**100.0**	**100.0**
hole p.	94.5	93.2	95.1	97.1	97.3	**99.8**
iron	97.2	96.5	99.6	98.2	97.1	**99.9**
lamp	96.3	96.9	97.7	98.4	99.0	**99.9**
phone	96.7	96.3	97.8	98.3	98.8	**99.7**
Mean	95.3	95.8	96.5	97.2	97.3	**99.7**

data, KITTI provides an extensive range of object categories, including cars, pedestrians, and cyclists. For this study, only cars and pedestrians will be considered.

The approach employed to derive the outcomes in Sections 6.2.1 and 6.2.3 relies on utilizing the ground truth point cloud for pose prediction. In Section 6.2.2, the RGB-D and Depth methods similarly depend on ground truth 2D object detections for pose prediction. Consequently, in these instances, the processes of detection, point cloud projection, and point cloud clustering were not employed as part of the investigation detailed in the aforementioned sections. The loss used in these methods was the original one from DenseFusion. The assessment will be conducted based on the metrics outlined in Section 6.1.1.

Table 6.5: Accuracy (ADD(-S)) of Depth state-of-art methods on the LINEMOD dataset. Objects in bold are symmetric

	DenseFusion: Depth	**DenseFusion: Point Cloud**	**CloudPose** [57]	**CloudAAE** [58]
ape	77.8	90.5	58.3	**92.5**
bench vi	86.6	**94.5**	65.6	90.8
camera	84.5	**89.4**	43.0	85.7
can	89.2	**96.1**	84.7	95.1
cat	93.2	**97.4**	84.6	96.8
driller	89.1	94.9	83.3	**98.7**
duck	79.3	**89.1**	43.2	84.4
eggbox	99.8	**99.9**	99.5	99.2
glue	99.7	**99.4**	98.8	98.7
hole p.	90.4	**93.5**	72.1	85.3
iron	91.3	**93.8**	70.3	91.4
lamp	94.2	**97.7**	93.2	86.5
phone	94.4	95.9	81.0	**97.4**
Mean	90.0	**94.8**	75.2	92.5

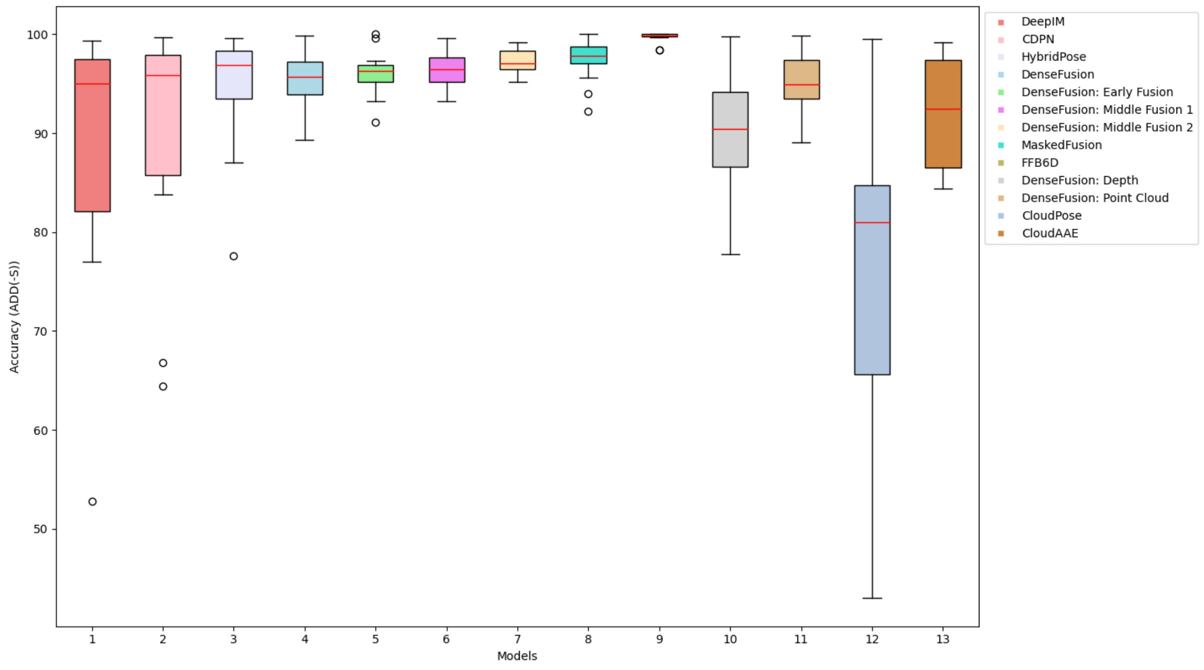

Figure 6.7: Box plot of the accuracy of the state-of-art methods on the LINEMOD dataset.

6.2.1 Distance-Based Accuracy Study

Within this section, an investigation will be done to comprehensively evaluate the network's ability to estimate the poses of objects situated at varying distances from the sensor. This study aims to provide a detailed understanding of how the network performs across different ranges and distances.

The accuracy (measured by ADD) of both vehicles and pedestrians can be found in Table 6.6 and Table 6.7, categorized by their respective ranges. Table 6.6 exclusively includes ground

Table 6.6: Accuracy (ADD) according to the distance of the objects on the KITTI dataset. Objects with less than 100 points were not considered.

Distance	**<7m**	**7-15m**	**15-25m**
Car	**95.9** (1114)	88.3 (969)	85.9 (1178)
Pedestrian	**89.7** (136)	85.8 (281)	61.1 (72)
Mean (1250)	**95.2**	87.8	84.5

Table 6.7: Accuracy (ADD) according to the distance of the objects on the KITTI dataset. Objects with less than 50 points were not considered.

Distance	**<7m**	**7-15m**	**15-25m**	**25-35m**
Car	**95.8** (1116)	84.9 (971)	87.7 (1090)	95.2 (1232)
Pedestrian	**89.6** (134)	74.2 (279)	74.4 (160)	66.7 (18)
Mean (1250)	**95.1**	82.5	86.0	91.2

truth objects that contained over 100 points before the upsampling/downsampling process, while Table 6.7 discards objects with less than 50 points.

This tables provide a clear representation of how object distance significantly influences detection accuracy. The accuracy falls with increasing separation between the Velodyne and objects. This holds true for both car and pedestrian detection scenarios. The decline in accuracy with distance underscores the intricate challenges that detection systems confront when attempting to identify distant objects precisely. One plausible explanation for this phenomenon could be attributed to the reduction in distinct points that distant objects exhibit, resulting in decreased object definition. It's worth noting an exception in Table 6.7, where accuracy is greater for the 25-35m range than the 7-15m and 15-25m ranges. This anomaly might be attributed to the dominance of cars at this particular distance interval, with only 18 pedestrians present. This scenario could enable the model to generalize more effectively for estimating car poses, subsequently leading to an overall accuracy boost.

A comparative analysis between *car* and *pedestrian* detection reveals a consistent trend: cars consistently achieve higher detection accuracy across all distance spans and point thresholds. This outcome aligns with expectations due to cars' greater prevalence and size, which naturally makes them more conspicuous in the detection process. However, the disparity in accuracy between cars and pedestrians extends as the distance increases. This disparity underscores that while both object classes have difficulty with distance-related challenges, pedestrian detection exhibits a heightened sensitivity to varying object distances.

The influence of the point threshold on accuracy also merits attention. Examination of objects containing a minimum of 100 points (Table 6.6) reveals a tendency toward heightened accuracy compared to instances where the threshold is set at 50 points (Table 6.7). This underscores the significance of dense point clouds in furnishing ample information for accurate detection, particularly when dealing with objects situated at greater distances. Qualitative results are presented in Figs. 6.17 and 6.9.

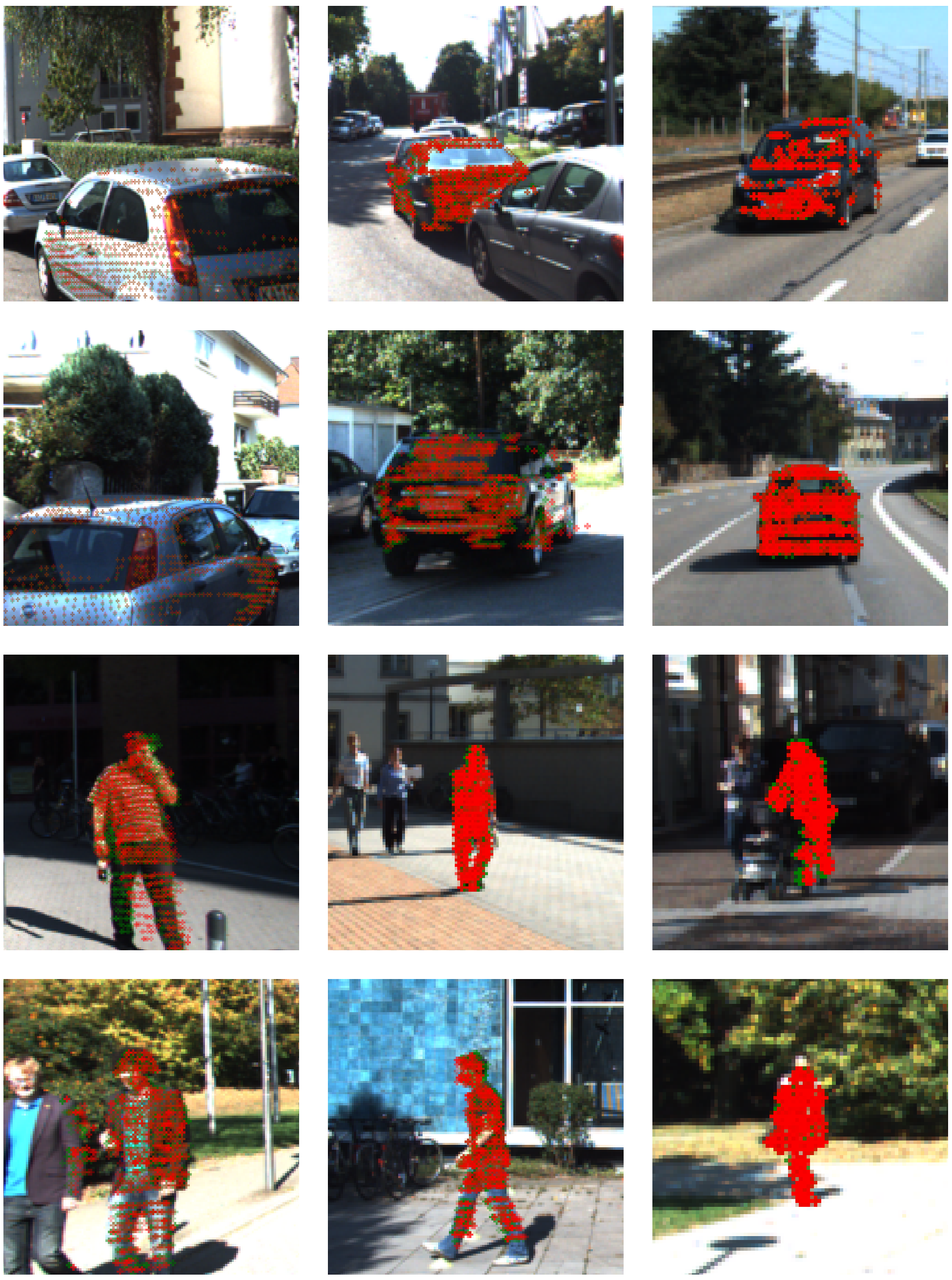

Figure 6.8: Visualization of the method according to the distance on the KITTI dataset, considering only objects with more than 100 points. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results for <7m, 7-15m, 15-25m, respectively.

6.2.2 Multimodal Study

In this section, we study how input data can significantly impact a model's performance when applied to a subset of the KITTI dataset, focusing on objects within a 7-meter range of the LiDAR sensor, filtering out objects with fewer than 100 data points. Table 6.8 presents the accuracy results of various multimodal methods. These methods include RGB-D, which extracts features from the RGB image and geometry features from the point cloud; Depth, which extracts features from the depth map and the point cloud; and Point Cloud, which relies

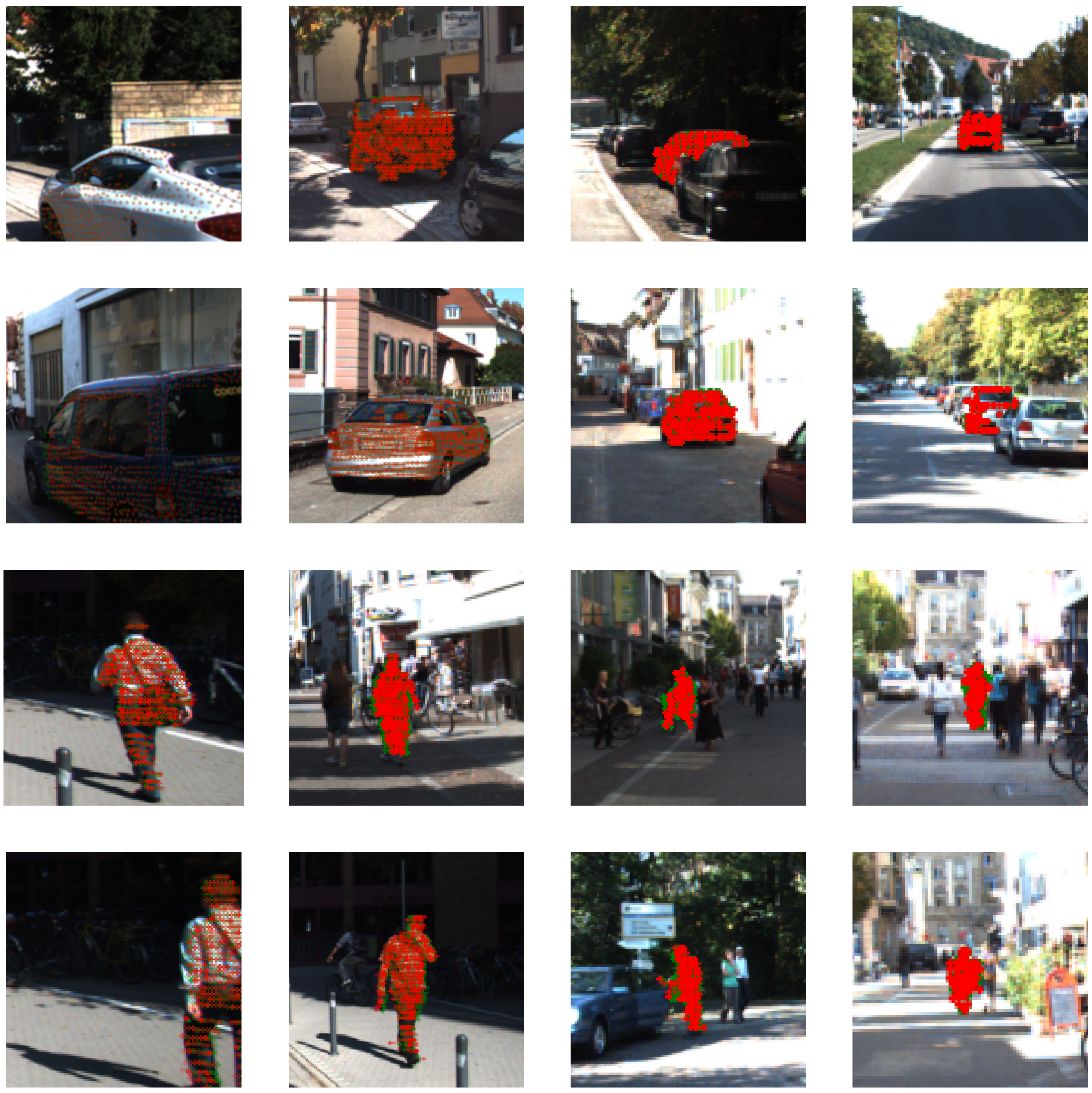

Figure 6.9: Visualization of the method according to the distance on the KITTI dataset, considering only objects with more than 50 points. Green dots indicate the ground truth poses, while red represents estimated poses. Four columns show the results for <7m, 7-15m, 15-25m, and 25-35m, respectively.

solely on geometry features extracted from the point cloud.

For the *car* detection, the Point Cloud modality achieves the highest accuracy with 95.9%. The RGB-D modality also performs well, achieving an accuracy of 90.9%, although it falls slightly short of the accuracy attained by the Point Cloud modality. Turning our attention to *pedestrian* detection, the Point Cloud modality again outperforms the rest with an accuracy of 89.7%. This further underscores the point that point cloud data is a powerful tool for detecting cars and pedestrians effectively.

These results indicate that point cloud data stands out as the most dependable and accurate modality for object detection tasks in the context of autonomous driving, closely followed by RGB-D data. Point clouds excel in capturing intricate spatial information, making them well-suited for the precise detection of both cars and pedestrians. Qualitative results are presented in Fig. 6.10.

Table 6.8: Accuracy (ADD) of the multimodal methods on the KITTI dataset.

	RGB-D	**Depth**	**Point Cloud**
Car (1114)	90.9	88.2	**95.9**
Pedestrian (136)	76.5	86.0	**89.7**
Mean (1250)	89.4	88.0	**95.2**

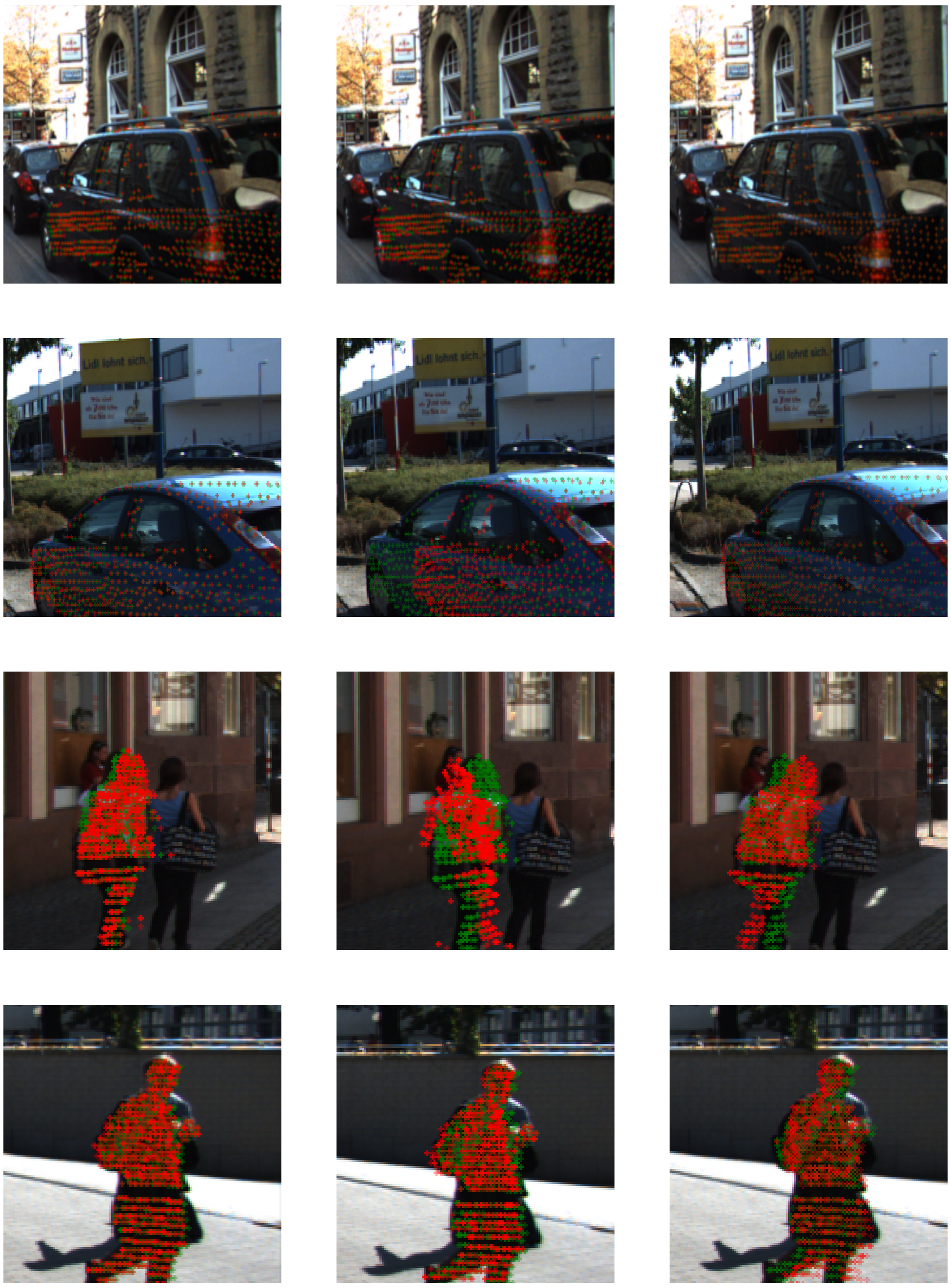

Figure 6.10: Visualization of the multimodal methods on the KITTI dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results of RGB-D, Depth, and Point Cloud, respectively.

6.2.3 Prediction Selection Variations

As introduced in Section 4.2.1.6, a new way of choosing the correct prediction for the pose estimation is proposed. This method involves computing a weighted average of predictions based

Table 6.9: Accuracy (ADD) for different prediction selection methods on the KITTI dataset.

	Argmax	**Weighted Avg**	**Weighted Avg 50%**
Car (1114)	**95.9**	81.7	79.9
Pedestrian (136)	**89.7**	53.0	61.8
Mean (1250)	**95.2**	78.6	77.9

on their associated confidence scores. The accuracy (ADD) on the KITTI dataset for various prediction selection techniques is presented in Table 6.9. Specifically, three distinct methods are compared: the argmax, which selects the prediction with the highest confidence; the weighted average, which takes into account all predictions; and the weighted averages considering a subset of predictions, the top 50% predictions with the highest confidence. A subset of KITTI was used, considering only objects within 7 meters of the LiDAR sensor, and excluding objects with less than 100 points.

From Table 6.9 it's possible to observe that none of the proposed methods manage to surpass the original approach's performance, Qualitative results are presented in Fig. 6.11.

6.2.4 DenseFusion: PointCloud

This section will assess the performance of the *DenseFusion: Point Cloud*. This method relies on object detection in a 2D space to estimate the pose of objects, making it crucial to have a robust detector capable of identifying all the objects of interest within a given image. In our case, the input is a depth image, which further complicates the task of 2D object detection.

In Figs. 6.12 and 6.13 are presented the results of the Yolov5 2D object detection. The confusion matrix and the precision-recall curve provide valuable insights into the model's performance. The confusion matrix is a tabular representation used to evaluate the model's performance. It categorizes the model's predictions into three key categories:

- **True Positives (TP):** Number of instances where the detection was an IoU with the ground truth higher than a certain threshold.
- **False Positives (FP):** Number of instances where the detection was an IoU with the ground truth inferior to a certain threshold.
- **False Negatives (FN):** Number of instances where the ground truth objects don't have any detection.

where IoU is the intersection over union, a number that quantifies the degree of overlap between two bounding boxes. A threshold of 50% was used. The precision-recall curve illustrates the trade-off between precision and recall at different thresholds. Precision represents the ratio of true positives to the total number of predicted positive instances, indicating how many predicted positive instances are accurate. It is calculated as TP / (TP + FP). Recall (also known as sensitivity or true positive rate) represents the ratio of true positives to the total number of actual positive instances, measuring how effectively the model identifies actual positives. It

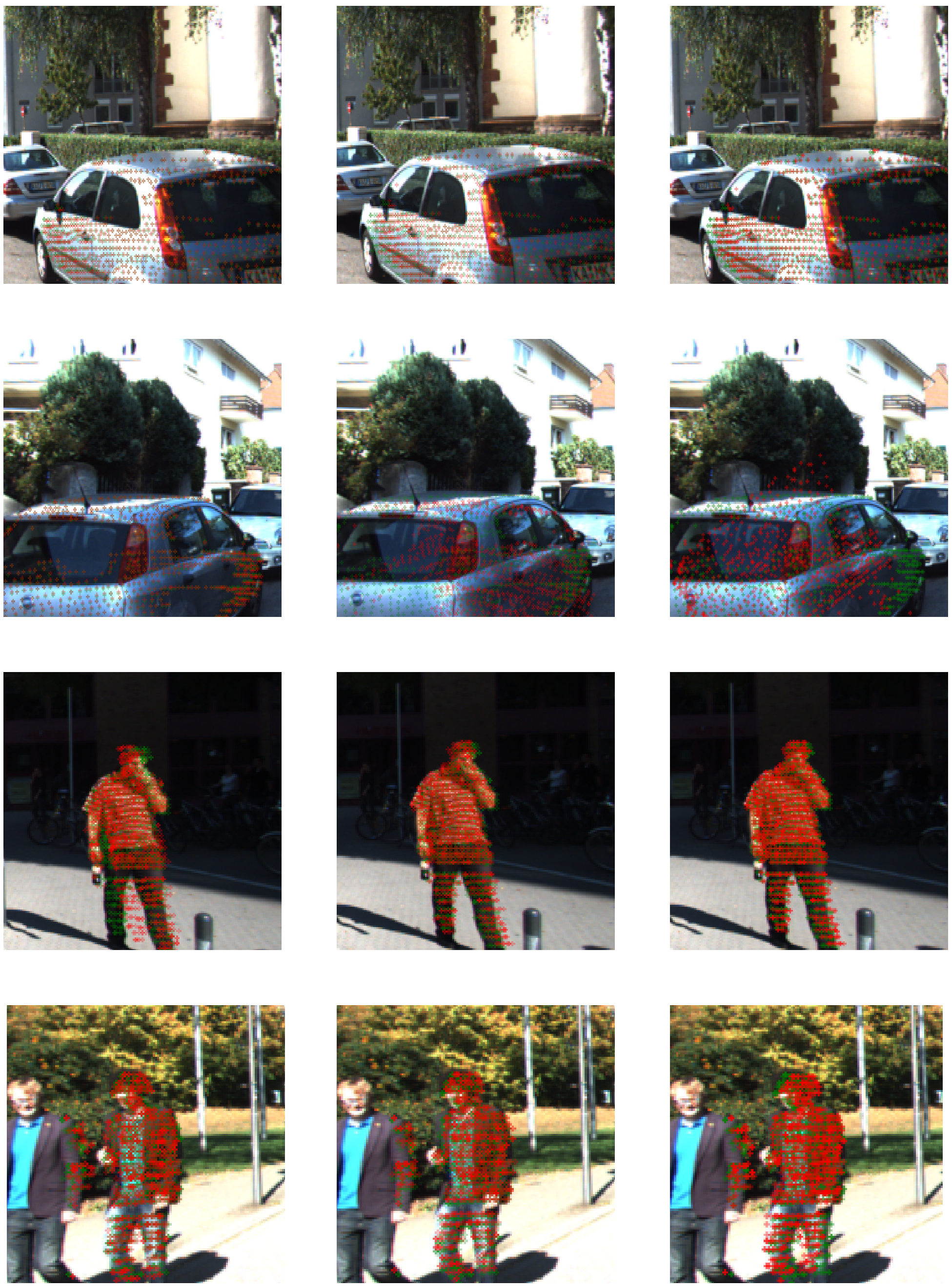

Figure 6.11: Visualization of the method according to the prediction selection on the KITTI dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Three columns show the results for argmax, weighted avg, and weighted avg 50%, respectively.

is calculated as TP / (TP + FN). A high Area Under the Curve (AUC) signifies both high recall and high precision, with high precision indicating a low false positive rate and high recall indicating a low false negative rate.

Analyzing the confusion matrix, we observe that for cars, the model achieved an 88% detection rate, implying that 12% of the cars went undetected. On the other hand, pedestrians had a detection rate of 79%, indicating that 21% of pedestrians were not detected. This suggests that detecting pedestrians is more challenging than detecting cars. Inspecting the precision-recall curve, we find that cars achieved an AUC of 91.9%, while pedestrians achieved 80.6%.

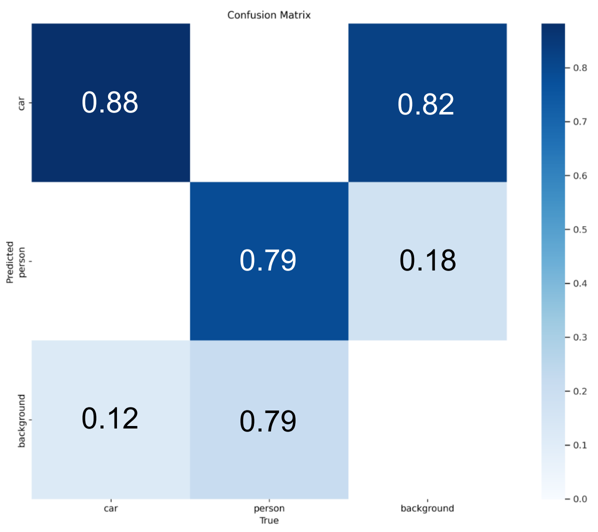

Figure 6.12: Confusion matrix of the 2D detection on the KITTI depth images.

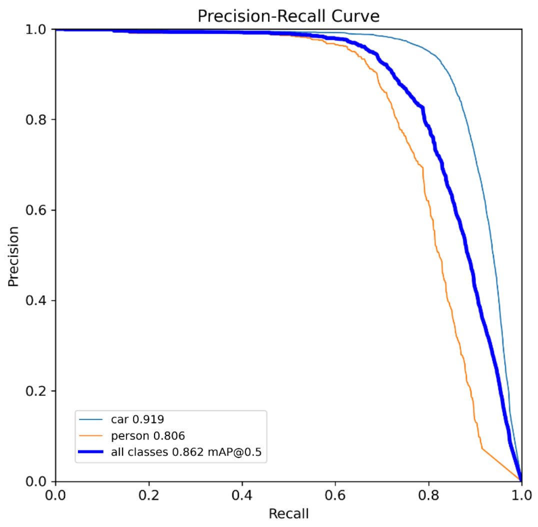

Figure 6.13: Precision-Recall curve of the 2D detection on the KITTI depth images.

Table 6.10: Accuracy (ADD) for DenseFusion: Point Cloud on the KITTI dataset.

	DenseFusion: PointCloud
Car (729)	94.5
Pedestrian (90)	98.9
Mean (819)	95.0

Table 6.11: Accuracy (ADD) for the proposed methods on the developed dataset.

	DenseFusion	**DenseFusion: Depth**	**DenseFusion: Point Cloud**	**DenseFusion: Middle Fusion**
Mean	100.0	100.0	100.0	100.0

These AUC values indicate the overall effectiveness of the model in correctly identifying cars and pedestrians, with cars being detected more accurately than pedestrians.

From Table 6.10 it is possible to observe the accuracy (ADD) of the model, having an overall accuracy of 95%. Impressively, the pedestrians class was able to have a higher accuracy than the cars class. Qualitative results are presented in Figure 6.14.

6.3 Validation on Industrial Dataset

In this section, we will assess the effectiveness of the proposed method using the newly created dataset. As illustrated in Figs. 6.15 and 6.16, it is evident that Yolov5 successfully detects all pallets while not producing any false positives. Additionally, referring to Table 6.11, we can observe that all methods exhibit flawless performance on the developed dataset, but it is important to mention that *DenseFusion: Middle Fusion* had the fastest converge rate.

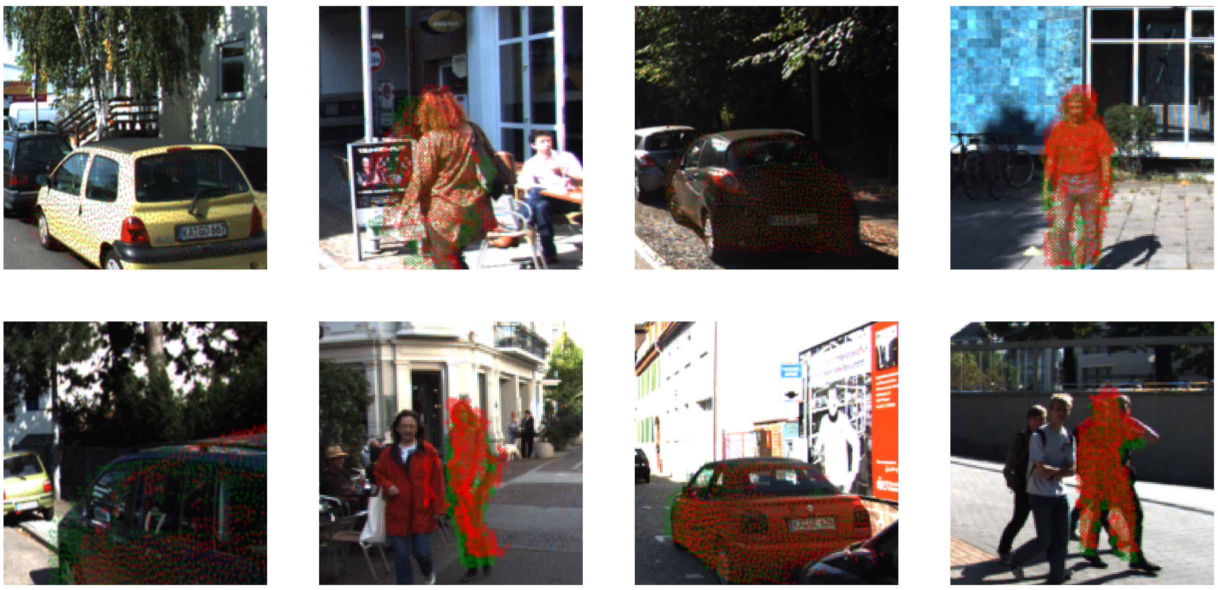

Figure 6.14: Visualization of the DenseFusion: Point Cloud on the KITTI dataset. Green dots indicate the ground truth poses, while red represents estimated poses.

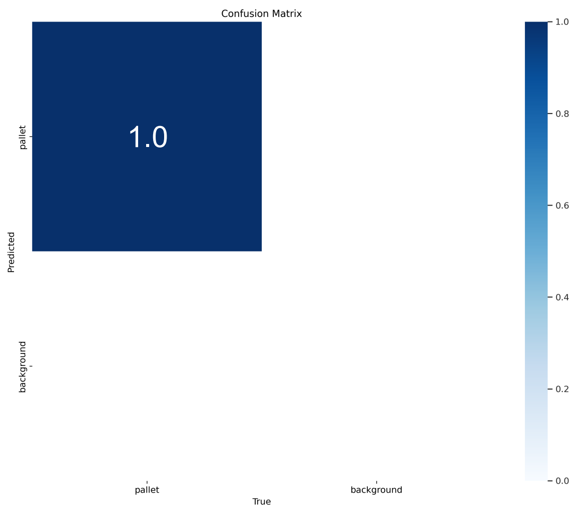

Figure 6.15: Confusion matrix of the 2D detection on the Industrial dataset depth map images.

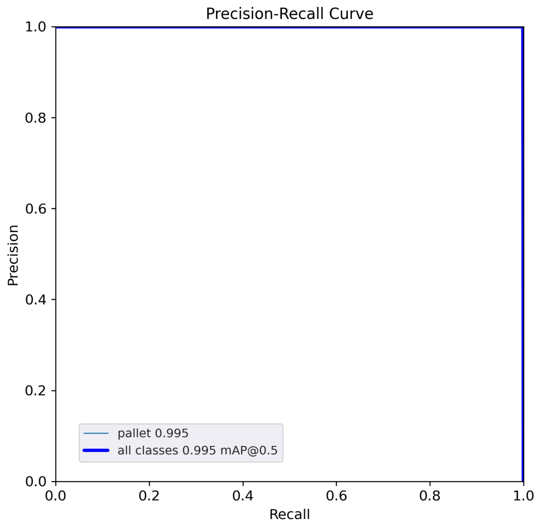

Figure 6.16: Precision-Recall curve of the 2D detection on the Industrial dataset depth map images.

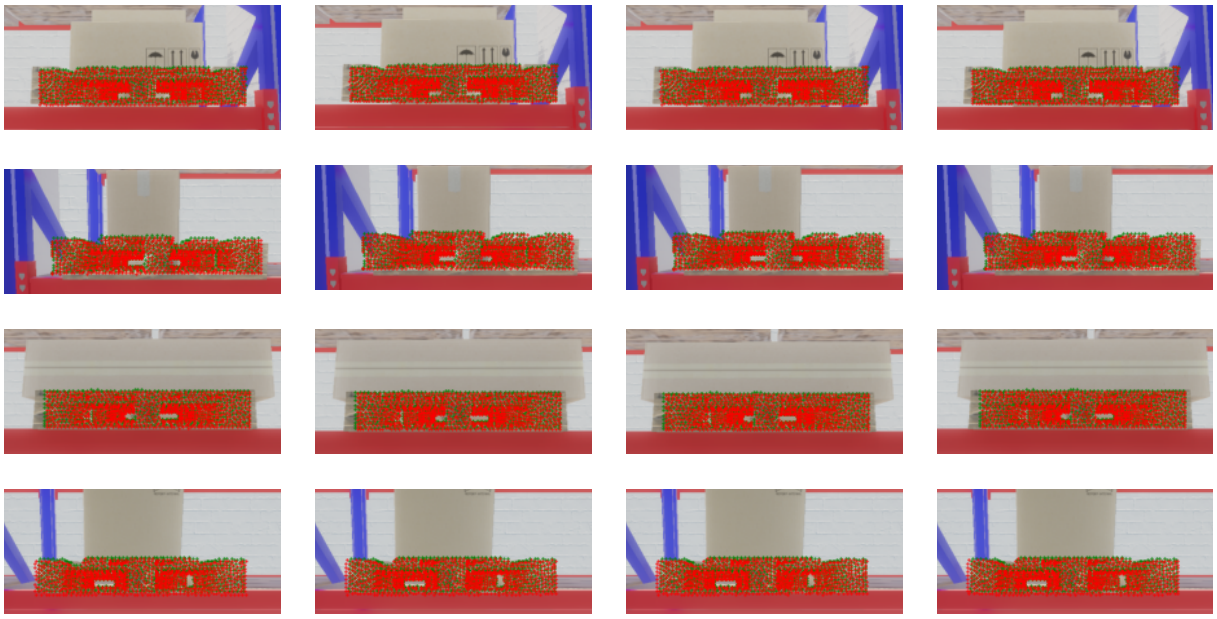

Figure 6.17: Visualization of the proposed methods on the developed dataset. Green dots indicate the ground truth poses, while red represents estimated poses. Four columns show the results for DenseFusion, DenseFusion: Depth, DenseFusion: PointCloud, DenseFusion: Middle Fusion respectively.

7

Conclusion

This dissertation represents a comprehensive exploration of 6D Object Detection algorithms. The primary objective was to develop an algorithm capable of accurately detecting the pose of pallets within shelves, with the intention of integrating it into an autonomous forklift for efficient object handling.

The development process started with the implementation of several state-of-the-art methods, drawn from publicly available codebases. With DenseFusion being the chosen framework to start from, a widely adopted technique in the community with substantial online support. Subsequently, a multimodal analysis of the framework was conducted to investigate how different types of input data impact the pose estimation accuracy. The original framework, utilizing RGB-D data, exhibited the best performance. Intriguingly, using only the point cloud of the object of interest yielded comparable results to the original method and outperformed it in several objects. In contrast, combining both the depth image and point cloud proved less effective, hinting at potential noise introduced into the system.

Following this analysis, some variations of the original method were introduced, including DenseFusion: Multimodal Early Fusion and DenseFusion: Multimodal Middle Fusion, aimed at enhancing performance. DenseFusion: Multimodal Early Fusion improved performance by fusing RGB and Depth images before feature extraction, while DenseFusion: Multimodal Middle Fusion improved performance by extracting features from RGB and Depth images separately and then fusing them using an attention model. Notably, DenseFusion: Multimodal Middle Fusion with a larger dimensionality stood out as a top performer, demonstrating performance close to some state-of-the-art methods.

After validating all introduced methods on the LINEMOD dataset, a distance-based study was conducted on the KITTI dataset, employing the point cloud of ground-truth objects as input data. This allowed to assess the model's behavior concerning the distance of objects, revealing that the framework more effectively estimated the pose of objects closer in proximity. A multimodal study was also conducted on this dataset, with DenseFusion: Point Cloud surprisingly having the best performance. Additionally, several variations in prediction selection methods were explored using a subset of the KITTI dataset. Instead of opting for the prediction with the highest confidence, a weighted average of predictions based on the best confidences was proposed. While this variation did not consistently outperform the original method, it did demonstrate potential improvements for specific objects, suggesting that the highest confidence prediction may not always be the most accurate.

Furthermore, a method was proposed utilizing input data from a LiDAR sensor. For this approach, a novel loss function was introduced by computing the Chamfer distance between two point clouds. This resulted in performance comparable to when using ground truth objects and the original loss function.

Finally, the introduced methods were validated on the developed industrial dataset. All validated methods achieved an impressive accuracy rate of 100%, unequivocally demonstrating the successful accomplishment of the dissertation's primary goal.

7.1 Future Work

The results presented in this dissertation demonstrate the full potential of 6D object detection in industrial applications. Nonetheless, it is recognized that there is still a wide range of improvements that can be made.

Longer Tests

Test the methods for 500 epochs, as the state-of-art methods.

Virtual Dataset

Improve virtual dataset with different lighting conditions and more occlusions.

Detection of Multiple Objects

Improve the model by considering panoptic segmentation models to deal with multiple objects.

Real-World Validation

Acquire data in a real industrial scenario and evaluate the models obtained from virtual data.

Simulation to Reality

Simulation to reality transfer and transfer learning of the models obtained with virtual data, considering data acquired in an industrial scenario.

Bibliography

[1] Renjie Xu, Haifeng Lin, Kangjie Lu, Lin Cao, and Yunfei Liu. A Forest Fire Detection System Based on Ensemble Learning. *Forests*, 2021.

[2] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3343–3352, 2019.

[3] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *Robotics: Science and Systems (RSS)*, 2018.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[6] Henrik Christensen, Nancy Amato, Holly Yanco, Maja Mataric, Howie Choset, Ann Drobnis, Ken Goldberg, Jessy Grizzle, Gregory Hager, John Hollerbach, et al. A Roadmap for US Robotics - From Internet to Robotics 2020 Edition. *Foundations and Trends® in Robotics*, 8(4):307–424, 2021.

[7] Zubair Md Fadlullah, Fengxiao Tang, Bomin Mao, Nei Kato, Osamu Akashi, Takeru Inoue, and Kimihiro Mizutani. State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. *IEEE Communications Surveys & Tutorials*, 2017.

[8] M Shamim Hossain, Muneer Al-Hammadi, and Ghulam Muhammad. Automatic Fruit Classification Using Deep Learning for Industrial Applications. *IEEE transactions on industrial informatics*, 2018.

[9] Roberto M Souza, Erick GS Nascimento, Ubatan A Miranda, Wenisten JD Silva, and Herman A Lepikson. Diagnosis of Multiple Faults in Rotating Machinery Using Ensemble Learning. *Computers & Industrial Engineering*, 2021.

[10] Javier Villalba-Diez, Daniel Schmidt, Roman Gevers, Joaquín Ordieres-Meré, Martin Buchwitz, and Wanja Wellbrock. Deep Learning for Industrial Computer Vision Quality Control in the Printing Industry 4.0. *Sensors*, 2019.

[11] YOLOv5. https://github.com/ultralytics/yolov5.

[12] Markus Knitt, Jakob Schyga, Asan Adamanov, Johannes Hinckeldeyn, and Jochen Kreutzfeldt. PalLoc6D-Estimating the Pose of a Euro Pallet with an RGB Camera based on Synthetic Training Data. 2022.

[13] Peide Wang. Research on Comparison of LiDAR and camera in Autonomous Driving. In *Journal of Physics: Conference Series*. IOP Publishing, 2021.

[14] George Stockman and Linda G Shapiro. *Computer Vision*. Prentice Hall PTR, 2001.

[15] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, 25, 2012.

[17] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2015.

[19] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path Aggregation Network for Instance Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-time Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[21] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *kdd*, 1996.

[22] Andrew Ng, Michael Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14, 2001.

[23] Muhammad Zubair Irshad, Thomas Kollar, Michael Laskey, Kevin Stone, and Zsolt Kira. CenterSnap: Single-Shot Multi-Object 3D Shape Reconstruction and Categorical 6D Pose and Size Estimation. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[24] Muhammad Zubair Irshad, Sergey Zakharov, Rares Ambrus, Thomas Kollar, Zsolt Kira, and Adrien Gaidon. ShAPO: Implicit Representations for Multi-Object Shape, Appearance, and Pose Optimization. In *European Conference on Computer Vision*. Springer, 2022.

[25] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[26] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[27] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[28] Wongun Choi, Yu-Wei Chao, Caroline Pantofaru, and Silvio Savarese. Understanding Indoor Scenes Using 3D Geometric Phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 33–40, 2013.

[29] Siyuan Huang, Siyuan Qi, Yixin Zhu, Yinxue Xiao, Yuanlu Xu, and Song-Chun Zhu. Holistic 3D Scene Parsing and Reconstruction from a Single RGB Image. In *Proceedings of the European conference on computer vision (ECCV)*, pages 187–203, 2018.

[30] Tong He and Stefano Soatto. Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8409–8416, 2019.

[31] Abhijit Kundu, Yin Li, and James M Rehg. 3d-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3559–3568, 2018.

[32] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[33] Siyuan Huang, Siyuan Qi, Yinxue Xiao, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Cooperative Holistic Scene Understanding: Unifying 3D Object, Layout, and Camera Pose Estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

[34] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*, pages 345–360. Springer, 2014.

[35] Qianhui Luo, Huifang Ma, Li Tang, Yue Wang, and Rong Xiong. 3D-SSD: Learning Hierarchical Features from RGB-D Images for Amodal 3D Object Detection. *Neurocomputing*, 2020.

[36] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[37] Xiaoke Shen and Ioannis Stamos. Frustum VoxNet for 3D object detection from RGB-D or Depth Images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1698–1706, 2020.

[38] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253, 2018.

[39] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. ImVoteNet: Boosting 3D Object Detection in Point Clouds with Image Votes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4404–4413, 2020.

[40] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[41] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-Based 3D Object Detection and Tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[42] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel R-CNN: Towards High Performance Voxel-Based 3D Object Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1201–1209, 2021.

[43] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. RangeDet: In Defense of Range View for Lidar-Based 3D Object Detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2918–2927, 2021.

[44] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. RSN: Range Sparse Net for Efficient, Accurate Lidar 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2021.

[45] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10529–10538, 2020.

[46] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. SE-SSD: Self-Ensembling Single-Stage Object Detector from Point Cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14494–14503, 2021.

[47] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. In *European Conference on Computer Vision (ECCV)*, 2018.

[48] Zhigang Li, Gu Wang, and Xiangyang Ji. CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6-DoF Object Pose Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[49] Tomáš Hodaň, Dániel Baráth, and Jiří Matas. EPOS: Estimating 6D Pose of Objects with Symmetries. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[50] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PvNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[51] Zhe Cao, Yaser Sheikh, and Natasha Kholgade Banerjee. Real-Time Scalable 6DoF Pose Estimation for Textureless Objects. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[52] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision*. Springer, 2013.

[53] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going Further with Point Pair Features. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 834–848. Springer, 2016.

[54] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal Templates for Real- Time Detection of Texture-Less Objects in Heavily Cluttered Scenes. In *2011 international conference on computer vision*. IEEE, 2011.

[55] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model Globally, Match Locally: Efficient and Robust 3D Object Recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*. Ieee, 2010.

[56] Frederik Hagelskjær and Anders Glent Buch. PointVoteNet: Accurate Object Detection and 6DoF Pose Estimation in Point Clouds. In *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020.

[57] Ge Gao, Mikko Lauri, Yulong Wang, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. 6D Object Pose Regression via Supervised Learning on Point Clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[58] Ge Gao, Mikko Lauri, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. CloudAAE: Learning 6D Object Pose Regression with On-line Data Synthesis on Point Clouds. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[59] Dingding Cai, Janne Heikkilä, and Esa Rahtu. OVE6D: Object Viewpoint Encoding for Depth-based 6D Object Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6803–6813, 2022.

[60] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012.

[61] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3583–3592, 2016.

[62] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RBD-D Images. *ECCV (5)*, 2012.

[63] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International journal of computer vision*, 2010.

[64] Magnus Burenius, Josephine Sullivan, and Stefan Carlsson. 3D Pictorial Structures for Multiple View Articulated Pose Estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3618–3625, 2013.

[65] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017.

[66] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[67] Chi Li, Jonathan Bohren, Eric Carlson, and Gregory D Hager. Hierarchical Semantic Parsing for Object Pose Estimation in Densely Cluttered Scenes. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016.

[68] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective Approaches to Attention-Based Neural Machine Translation. In *2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[69] Cristiano Premebida, Luis Garrote, Alireza Asvadi, A Pedro Ribeiro, and Urbano Nunes. High-Resolution Lidar-Based Depth Mapping using Bilateral Filter. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. IEEE, 2016.

[70] F Landis Markley, Yang Cheng, John L Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 2007.

[71] Mark G Sobell. *A Practical Guide to Ubuntu Linux*. Pearson Education, 2015.

[72] Python. https://python.org//.

[73] PyTorch. https://pytorch.org//.

[74] Numpy. https://numpy.org/.

[75] Matplotlib. https://matplotlib.org/.

[76] Scikit-learn. https://scikit-learn.org/.

[77] Opencv. https://opencv.org/.

[78] Ultralytics. https://ultralytics.com/.

[79] Cuda. https://developer.nvidia.com/cuda-zone.

[80] Conda. https://docs.conda.io/projects/conda/en/stable/.

[81] NVIDIA GeForce RTX 3060. https://www.zotac.com/pt/product/graphics_card/zotac-gaming-geforce-rtx-3060-twin-edge-0.

[82] Chen Song, Jiaru Song, and Qixing Huang. HybridPose: 6D Object Pose Estimation Under Hybrid Representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 431–440, 2020.

[83] Nuno Pereira and Luís A Alexandre. MaskedFusion: Mask-Based 6D Object Pose Estimation. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020.

[84] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. FFB6D: A Full Flow Bidirectional Fusion Network for 6D Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3003–3013, 2021.