UNIVERSITY OF BIRMINGHAM

VISUAL OBJECT DETECTION AND POSE ESTIMATION OF TEXTURE-LESS OBJECTS IN UNSTRUCTURED ENVIRONMENTS

by

Ayush Aggarwal

A thesis submitted to the University of Birmingham for the degree of

DOCTOR OF PHILOSOPHY

Extreme Robotics Lab

School of Metallurgy and Materials

College of Engineering and Physical Sciences

University of Birmingham

August 2024

Abstract

Advancements in industries such as manufacturing, logistics, healthcare, and energy increasingly rely on autonomous robots to enhance reliability, efficiency, and safety. A critical factor in these advancements is the ability of these robots to accurately perceive and understand their environments, enabling effective interaction and decision-making. Visual data, including images and three-dimensional (3D) point clouds, serve as reliable sources of information for these systems. However, vision-based approaches encounter numerous challenges when faced with texture-less objects, articulated shapes, and unstructured environments with varying lighting conditions and occlusions, which often degrade performance.

To address these challenges, this thesis develops novel methods for visual object detection and pose estimation, specifically tailored for texture-less objects in unstructured environments. Firstly, a 3D edge detection method has been proposed to extract 3D edge features from a scene's depth map. The 3D edges, representing the geometrical boundaries of an object, are independent of the object's texture, making them well-suited for challenging scenarios.

The extracted 3D edges are subsequently utilised to estimate six-degree-of-freedom (6-DoF) pose of objects in unstructured environments. This task is approached as an alignment problem, where the object's model cloud is aligned with the observed 3D edge point cloud. The sparse nature of the 3D edges allows the network to effectively manage limited data, making it robust in handling occlusions and unstructured scenes where dense object information is often unavailable.

Furthermore, this thesis proposes a method for classifying articulated objects. Objects' articulation introduces additional complexities, which significantly reduce the performance of standard methods for pose estimation, grasping, and manipulation. To address this, a classifica-

tion approach is developed to determine whether an object is rigid or articulated by analyzing a video sequence that captures object manipulations. By identifying the object's properties within the scene, this approach enables the deployment of property-specific methods tailored to perform the corresponding tasks effectively.

Finally, a learning-based visual servoing method utilising the 3D edge point cloud is proposed to evaluate the effectiveness of 3D edges in practical scenarios. A deep learning-based network is implemented to learn latent features from 3D edges, which, together with the centroid of the 3D edge cloud, are utilised as visual cues to guide the servoing task.

The efficacy of the proposed vision-based methods has been validated through various performance metrics, including rotation error, translation error, F-measure, precision, recall, and average model distance (ADD) error. The performance of these methods has been analysed across multiple benchmark datasets and compared with state-of-the-art (SOTA) methods available in the literature. Results demonstrate competitive or superior performance compared to state-of-the-art methods, confirming the effectiveness of the proposed approaches for texture-less objects and unstructured environments.

Acknowledgements

I would like to express my deepest gratitude to my advisors, Prof. Rustam Stolkin and Dr. Naresh Marturi, for their unwavering support and guidance throughout the course of my PhD. Their patience, belief in my ideas, and insightful advice were instrumental in the completion of this thesis.

I am also immensely thankful to my colleagues, Cristiana and Max, for their continuous support and encouragement during this journey. Our refreshing discussions and your willingness to share your expertise in robotics have greatly enriched my understanding and work.

To my family, I extend my heartfelt thanks for their motivation and unwavering support in my pursuit of a PhD. Special thanks to my grandfathers for instilling in me a "do-it-yourself" attitude that has kept my curiosity alive. I am deeply grateful to my parents for always encouraging my experiments, and for their love and support in helping me follow my dreams. I also want to thank my mother-in-law for her prayers and especially for her assistance in recent months; without her help, managing everything would have been incredibly challenging.

A special thank you to my wonderful wife, Deepa, for always being there for me, for her constant encouragement, and for helping me overcome the challenges I faced along the way. Her support has been invaluable. Lastly, I want to express my love and appreciation for my little son, Ishir, who joined me on this journey last year and has been a joyful source of light amidst the chaos of PhD life.

Contents

List of Figures

List of Tables

List of Acronyms

2D 2-Dimensional. xii, 5, 12, 16, 20, 23, 27, 35, 37, 47, 69, 70, 77, 211, 212

3D 3-Dimensional. xi–xiii, xv, 2, 3, 5–9, 12, 13, 15–21, 23, 25, 27, 28, 33–40, 42, 45, 47, 52–55, 57–72, 74–82, 85, 91, 102, 109, 115, 126, 129–131, 133–135, 137, 143–147, 149, 151–155, 197

3D-EOPEN 3D Edge-based Object Pose Estimation Network. 78

6-DoF 6-Degree of Freedom. xvi, 19, 20, 25, 27, 28, 30, 72, 74, 75, 78, 79, 100, 102, 129, 152, 155

7-DoF 7-Degree of Freedom. 143, 149, 152

ADD Average Distance. xv, xvi, xxiv, 56, 58, 59, 94–100, 213

AM Articulated Motion. 115–117, 120, 122

AR Average Recall. 94–96, 98, 100, 212

BiT Big Transfer. 12

BOP Benchmark for 6D Object Pose Estimation. xv, xvi, 93–95, 97, 99

CAD Computer-Aided Design. 6, 23, 27, 79, 144, 152

CNN Convolutional Neural Network. 13, 16–18, 21, 23, 36, 37, 41

DCP Deep Closest Point. 22, 25, 76, 77, 94, 95

DeepGMR Deep Gaussian Mixture Registration. 22

Depth-CNN Depth-Aware Convolutional Neural Networks. 13

Depth-ResNet Depth-Aware Residual Networks. 13

DGCNN Dynamic Graph CNN. 14, 15, 18, 76

DLED Deep-Learning based Robust Edge Detection. 77

DNN Deep Neural Network. 44

DoF Degrees of Freedom. 111, 141

DPM Deformable Part-based Model. 16

DVS Direct Visual Servoing. 130, 131

EANet Edge-Attention Network. 78

EOPEN Edge-based Object Pose Estimation Network. 95–101

FCN Fully Convolutional Network. 17

FEA Finite Element Analysis. 111

FETNet Feature Erasing and Transferring Network. 13

FGR Fast Global Registration. 21, 75, 94, 95, 97–100

FPFH Fast Point Feature Histograms. 13, 17, 18, 76, 131

FPS Farthest Point Sampling. 81, 82, 85, 136

FuseNet Fusion Net. 13

GB Giga-Byte. 55

GCPose Geometry Correspondence Pose. 22, 25

GMM Gaussian Mixture Models. 42

Go-ICP Global ICP. 21

GPU Graphics Processing Unit. 55

HED Holistically Nested Edge Detection. 47, 48

HOG Histogram of Oriented Gradients. 12, 13, 16, 18, 74, 129

IBVS Image-Based Visual Servoing. 129–131

ICP Iterative Closest Point. 21, 25, 75, 76, 94–100, 116

IDAM Iterative Distance-Aware Similarity Matrix. 22, 76, 77, 94, 95

KL Kullback-Leibler. xxiv, 42

KL-Divergence Kullback-Leibler-Divergence. 50, 51

KPConv Kernel Point Convolutions. 14, 15, 18

LBVS Learning-Based Visual Servoing. 131

LiDAR Light Detection And Ranging. xi, 2, 3, 26

LM LineMod. xi, xii, xv, 29, 93, 95, 96, 101

LMO Occluded-LineMod. xii, xvi, 93, 97, 98, 101

LoG Laplacian of Gaussian. 49, 64

LQR Linear Quadratic Regulators. 128

mAP mean Average Precision. xv, xvi, 94–100, 213

MLP Multi-Layer Perceptron. 14, 79, 84, 88, 89

MPC Model Predictive Control. 129

MRG Multi-Resolution Grouping. 82

MSCNN Multi-Scale CNN. 13

MSE Mean Squared Error. 40, 48

MSG Multi-Scale Grouping. 82

MSPD Maximum Symmetry-Aware Projection Distance. 94, 211, 212

MSSD Maximum Symmetry-Aware Surface Distance. 94

NARF Normal Aligned Radial Feature. 13

NM No Motion. 115–117, 122

OctNet Octree Networks. 13

PBVS Position-Based Visual Servoing. 129–131

PCA Principal Component Analysis. 129

PFH Point Feature Histograms. 13, 17

PID Proportional-Integral-Derivative. 128

PPF Point Pair Features. 14, 15, 22, 76, 131

PPF-MEAM Point Pair Feature-based Pose Estimation with Multiple Edge Appearance Models. 77

PV-RCNN Point Voxel RCNN. 17

RAM Random Access Memory. 55

RANSAC RANdom SAmple Consensus. 20, 55, 77

RCNN Region based CNN. 16, 17

ReLU Rectified Linear Unit. 47, 48, 88

ResNet Residual Network. 12

RGB Red Green Blue. 2, 16, 18–22, 36, 37, 77, 94, 100, 108, 112–114

RGB-D Red Green Blue-Depth. 13, 16, 18–22, 27, 28, 30, 74, 75, 93, 108, 109

RM Rigid Motion. 115–117, 120, 122

RPM-Net Robust Point Matching Network. 22

RSD Radius-based Surface Descriptor. 13

SHOT Signature of Histograms of Orientations. 13, 18, 76, 131

SIFT Scale-Invariant Feature Transform. 12, 16, 18–20, 74, 129

SOTA State-Of-The-Art. 7, 40, 53, 55, 66, 78, 80, 81, 93, 95, 151, 152

SPPNet Spatial Pyramid Pooling Network. 16

SSD Single Shot Detector. 16

SURF Speeded-Up Robust Features. 12, 19

SVD Singular Value Decomposition. 76

TEASER Truncated Least Squares Estimation And Semidefinite Relaxation. 21, 75, 76, 94

VGGNet Visual Geometry Group Network. 12

VoxResNet Voxelwise Residual Networks. 13

VS Visual Servoing. 128–134, 136, 137, 140, 143, 149

VSD Visible Surface Discrepancy. 94, 211

YOLO You Only Look Once. 16, 26

List of Symbols

$\widehat{x}$	X-axis
$\widehat{y}$	Y-axis
$\widehat{z}$	Z-axis
$\mathbf{R}$	rotation matrix
$\mathbf{t}$	translation vector
$\mathbf{p}$	point
$\mathbf{G}$	gratient kernel
$\mathbf{0}$	zero vector
$\mathbf{D}$	organised point cloud
$\mathcal{D}$	depth map
$\widehat{\mathcal{D}}$	pre-processed depth map
λ	threshold
W	width of depth map
H	height of depth map
$\mathbf{E}_p$	set of edge points
e	single edge point
m	cluster mean
d	single depth value in depth map
$\mathbf{\Gamma}$	edge feature disctribution
δ	gradient feature
$\mathcal{M}$	shadow mask

M convolution mask size

p soft probability

α gain parameter

KL KL loss

T target probability distribution

C current probability distribution

L total loss

A pose error

$\mathbf{R}_{err}$ rotational error

$\mathbf{T}_{err}$ translation error

$\boldsymbol{Z}_E$ ADD error

$\mathbf{E}$ object edge point cloud

$\mathbf{M}$ object model point cloud

V number of object's edge points

U number of object's model points

$\mathbf{P}$ homogeneous transformation

$\mathbf{C}$ correspondence matching matrix

$\mathbf{B}$ point cloud

$\mathbf{S}$ correspondence score matrix

$\mathbf{r}$ 6D rotation vector

L^{corr} correspondence loss

β margin hyperparameter

$\mathbf{C}$ RGB image

$\mathbf{V}$ voxel grids

v number of voxel grids

$\mathbf{I}$ identity matrix

$\mathbf{q}$ quaternion form

$\mathcal{H}$ hash table

$\boldsymbol{\rho}$ list of in-frame classifications

$\boldsymbol{f}$ feature vector

$\boldsymbol{p}^*$ desired object pose

$\mathbf{L}$ interaction matrix

$\mathbf{J}$ robot jacobian

$\boldsymbol{v}$ robot's spatial velocity

$\boldsymbol{c}$ object centroid

$\boldsymbol{f}$ latent space feature representation

q anchor points

$Q(.)$ Chamfer distance

Z depth of the point in the camera frame

f focal length

$\mathbf{G}_x^l$ Prewitt kernel in the X directions

$\mathbf{G}_y^l$ Prewitt kernel in the Y directions

$\mathbf{G}_1^l$ LoG kernel in horizontal direction

$\mathbf{G}_2^l$ LoG kernel in vertical direction

$\mathbf{P}$ pose matrix

$Prec$ precision

Rec recall

θ distance threshold

$dist(\cdot)$ distance calculation function

c channels

n_f filter size

n_k kernel size

$\mathbf{K}$ camera intrinsic matrix

$\mathcal{M}$ set of model points

$\mathcal{S}$ symmetry transformations

n_k kernel size

List of Notations

$\circledast$	element-wise multiplication
$\circledplus$	element-wise addition
$*$	convolution operation
$\log(\cdot)$	logarithm function
$\|\| \cdot \|\|_F$	Frobenius norm operation
$(\cdot)^\top$	transpose operation
$O(\cdot)$	Big-O notation
$\sum$	summation operation
$\max(a, b)$	maximum of (a, b)
$\mathbf{X}^\dagger$	pseudo-inverse of matrix $\mathbf{X}$
$\frac{\partial(\boldsymbol{A})}{\partial \boldsymbol{x}}$	partial differentiation of $\boldsymbol{A}$ with respect to $\boldsymbol{x}$
$Tr(\cdot)$	trace operator
$\pi(\cdot)$	2D projection function
$\boldsymbol{c}$	object centroid

Chapter 1

Introduction

The integration of autonomous robots across various industries marks a pivotal advancement in the contemporary engineering landscape. Autonomous robots, characterised by their capability to perform tasks without continuous human intervention, are increasingly becoming instrumental in addressing the growing need for operational efficiency, precision, and safety. For example, the manufacturing sector requires autonomous robots to perform repetitive and hazardous tasks, thereby revolutionising production lines by enabling consistent and high-speed operations with minimal human error and workplace injuries [220]. Similarly, the energy industry, encompassing both renewable and non-renewable sectors, faces a critical need for autonomous robots to enhance operational efficiency, safety, and sustainability amid rising global energy demands and the complexities of production, distribution, and maintenance operations [42, 237]. Comparable requirements have been observed in other sectors such as logistics [53], healthcare [184], agriculture [239], and mining [228]. The trend of increasing demand for autonomous robots, as analysed and suggested in [220, 53, 184, 239, 228, 42, 237], is presented in Figure 1.1. From the figure, it can be observed that the demand has been increasing at a nearly exponential rate, with requirements in all industries more than doubling within a decade.

An autonomous robot comprises several critical components, each playing a significant role in enabling the robot to operate independently. The primary components include sensing, processing, and actuation. Sensing involves gathering information from the environment using

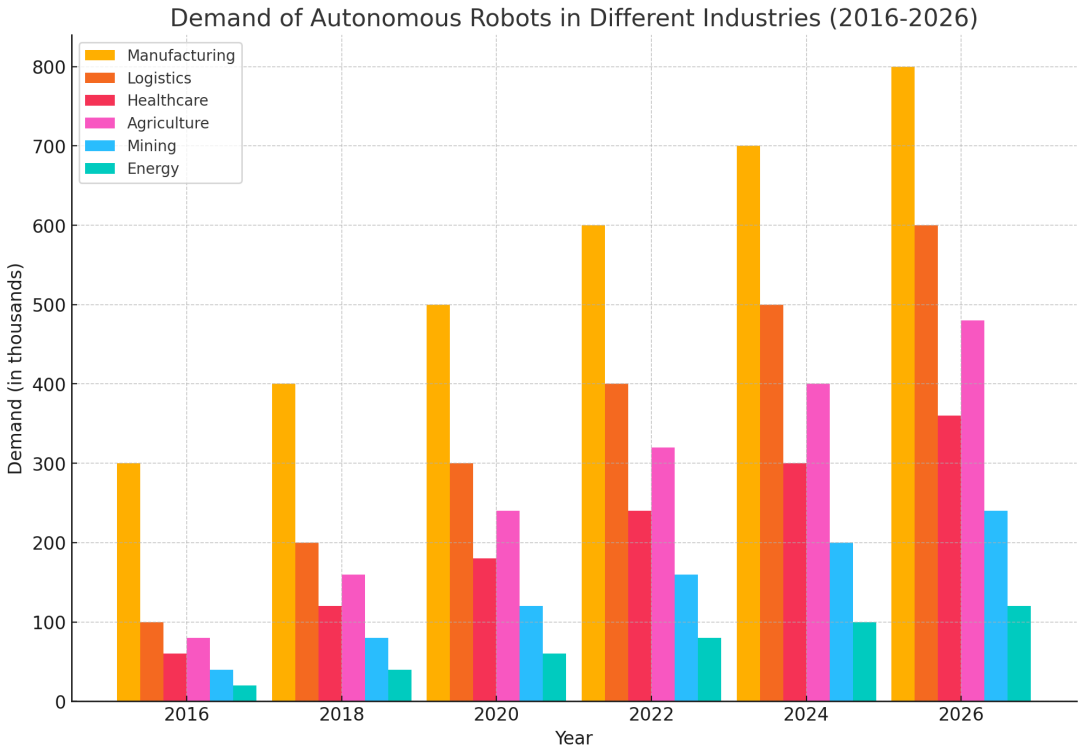

Figure 1.1: Observed and predicted trend for the demand of autonomous robots in manufacturing, logistics, healthcare, agriculture, mining and energy industry [220, 53, 184, 239, 228, 42, 237].

various sensors such as cameras, depth sensors, radars, and Light Detection And Ranging (LiDAR) [286, 134, 49], as shown in Figure 1.2. These sensors collect data in different formats, such as Red Green Blue (RGB) images, depth images, and 3-Dimensional (3D) point clouds, which provide a comprehensive understanding of the robot's surroundings [260, 81]. Processing, often executed by onboard computers, interprets this sensory data to understand the environment, identify work objects and/or obstacles, and determine appropriate actions [286, 180]. Finally, actuation is the component responsible for executing these actions, encompassing motors, servos, and other mechanisms that control the robot's movement and interactions with the environment [165, 183]. Each of these components is crucial, with sensing providing the necessary data, processing transforming the data into actionable insights, and actuation enabling the robot to carry out tasks autonomously [260, 165].

To enable robots to perform various tasks according to the requirements of different industries, specific processing algorithms need to be designed to tackle their respective challenges. Each industry presents unique obstacles that necessitate tailored solutions. For instance, in the

Figure 1.2: Different depth camera and LiDAR devices used in industries for 3D perception.

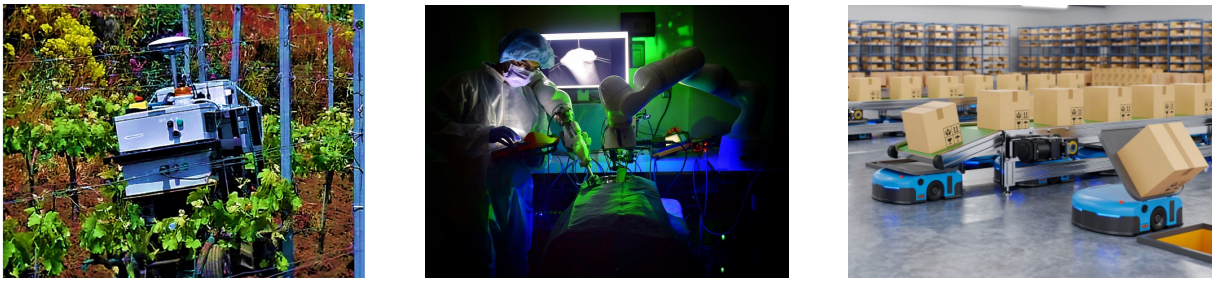

Figure 1.3: Application of autonomous systems in agriculture, healthcare, and logistic industries.

healthcare industry, surgical robots require advanced image processing and machine learning algorithms to interpret medical images and guide surgical instruments with precision [290]. The logistics sector also poses significant challenges for autonomous robots. Algorithms for this industry must manage dynamic environments with high traffic, optimise route planning for delivery robots, and ensure safe and efficient handling of packages. Robots in warehouses, for example, rely on sophisticated navigation and object recognition algorithms to move goods from storage to shipping areas [279]. The development and implementation of these algorithms are crucial for the successful integration of autonomous robots into various industries, ultimately enhancing efficiency, accuracy, and safety. These applications are visually represented in Figure 1.3.

1.1 Problem, challenges, and motivation

Developing methods to understand the environment and generate insights to perform specific tasks is accompanied by multiple challenges in various industries, which are listed below:

Understanding visual data

Visual data, encompassing images, videos, and point clouds captured by cameras and other sensors, is a primary source of information for autonomous systems to comprehend their environment. Accurate interpretation of this data is essential for making informed decisions and executing tasks effectively. However, visual data is often challenged by factors such as varying lighting conditions, occlusions, background clutter, and changes in perspective. These variations make it difficult to consistently extract useful information, necessitating the development of robust feature extraction techniques that can operate reliably despite these challenges. Achieving this level of consistency and accuracy in interpreting visual data remains a significant hurdle for autonomous systems.

Texture-less objects

Texture-less objects lack distinctive features such as colour, pattern, or texture that can help identify them (e.g., metal pipes, boxes, wires). These objects are particularly challenging for machine vision systems because traditional feature-based detection algorithms rely on visual cues to recognise and differentiate objects. In industrial settings, most of the objects utilised are industrial texture-less objects, making it difficult to separate them from the background or other objects. This poses a significant challenge for autonomous systems tasked with identifying, manipulating, and sorting these objects [170, 103].

Unstructured environments

Unstructured environments present a significant challenge due to their inherent unpredictability and variability. Unlike controlled, structured settings such as warehouses or power grids, unstructured environments lack predefined layouts and consistent conditions, e.g. in agriculture and mining sectors. Addressing these challenges remains critical to expanding the applicability and reliability of robotic systems in real-world, unstructured environments [274].

Articulation in objects

Articulated objects are characterised by their movable parts or joints, presenting significant

challenges in industrial settings. Many industrial objects, such as wires, chains, and pliers, exhibit varying degrees of articulation that must be accurately addressed. Furthermore, the complexity increases in cluttered or occluded environments commonly found in industrial applications, where parts may overlap or be partially hidden. Effective handling of articulated objects demands robust techniques to ensure precise detection and interaction, highlighting a crucial area for ongoing research and development in the field of industrial robotics and automation [203].

Unavailability of data

The unavailability of sufficient training data poses a significant challenge for developing robust vision-based methods in various industrial settings. In environments such as nuclear decommissioning, manufacturing, and hazardous material handling, where safety concerns are paramount, there is often a severe limitation on the availability of real-world object views. This scarcity of data hampers the ability to train and validate learning-based algorithms, e.g., supervised learning for classification, effectively before their deployment in actual applications [20].

In response to these challenges, there is a strong motivation to design methods specifically tailored to industrial environments. These methods must be capable of object classification, recognition, and pose estimation for texture-less objects, both rigid and articulated, in unstructured settings. Additionally, they should be robust to dynamic uncertainties and the unstructured organisation of objects commonly found in these environments. To mitigate the lack of real-world data, techniques such as Sim2Real [109] and domain transfer [254] are crucial, allowing for the development of models that can learn effectively without the need for extensive real-world datasets. Furthermore, 3D vision, which provides depth information, offers significant advantages over 2-Dimensional (2D) vision by allowing for a more accurate understanding of spatial relationships between objects. This depth information improves the system's ability to navigate cluttered environments and perform tasks such as grasping or assembling components with greater accuracy and reliability than 2D vision systems. As a result, this thesis

proposes advanced methods to tackle these challenges, enhancing the efficiency and flexibility of autonomous systems in industrial applications.

1.2 Contributions

Given these challenges and motivations, this thesis proposes various vision-guided methods targeting unstructured environments. Specifically, multiple vision-based tasks are explored, which are listed below.

Feature extraction For the first task, to extract texture-independent unique features from the scene, a novel unsupervised learning-based approach for detecting 3D edges is developed. This approach learn deep features at three different scales to capture coarse and fine details. Later, the method utilises learning-based clustering to differentiate between edge and non-edge points.

Object recognition In the context of object recognition, a region-to-region registration-based method for classifying both articulated and rigid objects is introduced. The method is designed to utilise a video sequence without any labelled data, where an object is being manipulated, to classify object as either rigid or articulated. This enables autonomous systems to detect whether an object is articulated and to respond appropriately.

Pose estimation Finally, for the pose estimation task, a self-supervised method for estimating the pose of rigid objects is proposed. This method utilises 3D edges, which correspond to approximately 5% of the original points, as input for scene objects along with the Computer-Aided Design (CAD) model of the object.

Applications Furthermore, to demonstrate the practical effectiveness of the proposed methods, an experiment is conducted. A 3D edge-based learning visual servoing pipeline is proposed to showcase real-world visual servoing experiments. The data used in these methods is primarily generated in a simulation environment, addressing the issue of dataset unavailability.

For all, point cloud data is used as the visual input, providing the spatial coordinates and point normals without any texture or colour to handle texture-less objects. The performance of all methods is evaluated using various performance metrics and compared with State-Of-The-Art (SOTA) methods over multiple benchmark datasets. This comprehensive evaluation ensures that the proposed methods are both effective and robust for real-world applications.

1.3 List of publications

Below is a list of papers associated with this thesis, along with a detailed breakdown of the contributions of each author:

1.3.1 Published work

- **Aggarwal, A.**, Stolkin, R. & Marturi, N., Unsupervised learning-based approach for detecting 3D edges in depth maps. Sci Rep 14, 796 (2024). https://doi.org/10.1038/s41598-023-50899-3

  The research was conceptualised and the method was implemented by A. Aggarwal, who also guided the experimental design and drafted the manuscript. The manuscript was reviewed and revised by R. Stolkin, while critical feedback and assistance in interpreting the results were provided by N. Marturi. The final manuscript was reviewed and approved by all authors.

- **A. Aggarwal**, R. Stolkin, & N. Marturi, Local Region-to-Region Mapping-based Approach to Classify Articulated Objects. 2023 20th Conference on Robots and Vision (CRV), Montreal, Quebec, Canada, 2023, pp. 177-183. https://doi.org/10.1109/CRV60082.2023.00030

  A. Aggarwal conceptualised the research, developed the methodology, and led the manuscript writing. Substantial input during manuscript revision was provided by R. Stolkin, while N. Marturi assisted in the experimental analysis

and manuscript revision. The final manuscript was reviewed and approved by all authors.

1.3.2 Unpublished work

- **Aggarwal, A.**, Stolkin, R. & Marturi, N., EOPEN: An Object Pose Estimation Method using 3D Edge Point Clouds.

  The research conceptualisation, methodology development, experimental validation and manuscript writing were led by A. Aggarwal. N.Marturi provided conceptual understanding, validation of experiments, and revised manuscript. Thorough manuscript revisions are performed by R.Stolkin.

- **Aggarwal, A.**, Farias, C., Stolkin, R. & Marturi, N., Learning-based Direct Visual Servoing with 3D Edges.

  A. Aggarwal conceptualised the research into two parts, *i.e.,* learning-based and control-based. The learning-based part is contributed by A. Aggarwal. The author also designed real-world experiments and led the manuscript writing. The control part of the methodology was developed by C. Farias, along with experiments and manuscript writing. N. Marturi validated the methodology, provided feedback on conceptual understanding, and performed manuscript revisions. R. Stolkin performed manuscript revisions, improving the quality of the work.

1.4 Thesis organisation

The rest of the thesis is organised as follows. Chapter 2 presents a survey of the latest advancements in vision-guided systems to process texture-less objects in unstructured environments. In Chapter 3, an unsupervised 3D edge feature detection method is described to extract 3D edges from point clouds. An object pose estimation method is discussed in Chapter 4, wherein 3D

edges are used to predict the pose of the object. This estimation method is based on a self-supervised deep learning model. A method to classify articulated objects using point clouds is presented in Chapter 5. Chapter 6 presents a practical application of 3D edge features utilised for a visual servoing task. Lastly, Chapter 7 presents the conclusion and future work.

Chapter 2

Literature study

In recent years, significant advancements have been made in the field of computer vision, particularly in the domains of visual object detection and pose estimation [174, 317, 155, 269, 113]. The challenge of accurately detecting and estimating the pose of texture-less objects in unstructured environments has gained increasing attention due to its relevance in real-world applications such as robotics, augmented reality, and autonomous systems. This chapter provides a comprehensive review of the existing literature, highlighting the key methodologies, algorithms, and techniques that have been proposed to address this challenge.

The literature review begins with an overview of feature extraction methods (Section 2.1), followed by a summary of object recognition techniques including detection and segmentation (Section 2.2). Object pose estimation methods are discussed in Section 2.3, and recent advancements for texture-less objects and unstructured environments are covered in Sections 2.4 and 2.5. Lastly, the datasets used in the thesis are detailed in Section 2.7.

2.1 Feature extraction

Visual features are distinctive elements or patterns present in an image or point cloud. These features may include inherent properties of an object, such as colour and texture, or may be extracted, such as edges and corners. These are crucial for any subsequent analysis of images or point clouds as they facilitate the extraction of insightful information from the data. This

information can then be directly applied in tasks such as object recognition and classification, thereby enabling machines to interpret and process visual information effectively.

Early research on feature extraction from images focused on designing handcrafted features that capture information like corners, edges, intensity maps, and image gradients. Prominent handcrafted features developed during this period include Harris Corner Detection [97], Histogram of Oriented Gradients (HOG) [56], Scale-Invariant Feature Transform (SIFT) [160], Speeded-Up Robust Features (SURF) [21], and Canny Edge Detection [33]. While these features have proven effective in extracting valuable information from raw data, they have several limitations, including limited robustness to variability, sensitivity to noise, the need for manual design and tuning, computational complexity, and inflexibility in diverse applications. The advent of deep learning popularised the extraction of deep features from images. Some of the most prominent image feature extractors include LeNet-5 [143], AlexNet [138], Visual Geometry Group Network (VGGNet) [234], Residual Network (ResNet) [99], Inception V3 [248], Big Transfer (BiT) [135], and DINO V2 [198]. These methods provide a fixed-size feature vector for each image, which can then be utilised in a wide variety of applications such as classification, recognition, and segmentation.

Using image-based feature extractors for point cloud data presents significant challenges due to the unstructured and higher-dimensional nature of point clouds, which differ substantially from the grid structure of 2-Dimensional (2D) images. Applying traditional image feature extractors directly to point clouds can lead to issues such as loss of geometric detail during voxelisation, inefficiency in handling sparse data, and increased computational complexity. 3D data is typically captured in two forms: depth maps and point clouds. A depth map is an organised representation of 3D data where two axes (usually $\widehat{x}$ and $\widehat{y}$) correspond to a 2D grid, and each grid cell contains the value of the third axis ($\widehat{z}$) wherever there is a point from the 3D data. It is an image-like representation for 3D data where each pixel has a depth value instead of pixel intensities. Due to its similarity to images, many handcrafted methods for images can be applied to depth maps to extract features. However, most of these methods struggle to capture the full depth information due to its complex nature. Some of the

most widely used feature extractors for depth maps include HOG [56], RGB-D SIFT [125], Normal Aligned Radial Feature (NARF) [241], Fast Edge Detection [29], and RGB-D Edge Detection [47]. Depth-Aware Convolutional Neural Networks (Depth-CNN) [92], Multi-Scale CNN (MSCNN) [69], Depth-Aware Residual Networks (Depth-ResNet) [267], Fusion Net (FuseNet)Fusion Net (FuseNet) [98], and FETNet [282] are among the pioneering works on feature extraction from RGB-D data using deep learning.

Point clouds generally represent an unorganised 3D data structure, where the same data is represented regardless of the order of points. Feature extractors developed for images and depth maps cannot be directly applied to this data structure as they struggle with the order invariance required for point clouds, potentially leading to inconsistent outputs if the input order of points is altered. To address the challenges of feature extraction from 3D point cloud data, several handcrafted and deep learning-based methods have been proposed in the literature [90, 95]. Prominent traditional point cloud feature extractors include Spin-Image [18], Point Feature Histograms (PFH) [223], Fast Point Feature Histograms (FPFH) [222], Signature of Histograms of Orientations (SHOT) [259], and Radius-based Surface Descriptor (RSD) [171]. Similar to image-based handcrafted feature extractors, these methods also face challenges such as sensitivity to sampling density, dependence on local geometry, lack of adaptability, and computational complexity.

In recent years, several deep learning-based point cloud feature extraction methods have been introduced in the literature [91, 159]. These methods are generally categorised based on the type of input received, *i.e.*, voxelised input-based and direct input-based. Voxelised input-based methods first transform the unordered point clouds into a 3D voxel grid where each grid cell may contain one or more points from the point cloud. The voxelised grid provides a structured way to process point clouds, enabling 3D Convolutional Neural Network (CNN) to extract features from point clouds. Prominent works in this category include VoxNet [176], VoxelNet [315], 3D ShapeNets [278], Octree Networks (OctNet) [218], Voxelwise Residual Networks (VoxResNet) [41], and SparseConvNet [72].

However, these methods are computationally intensive and dependent on the voxel grid

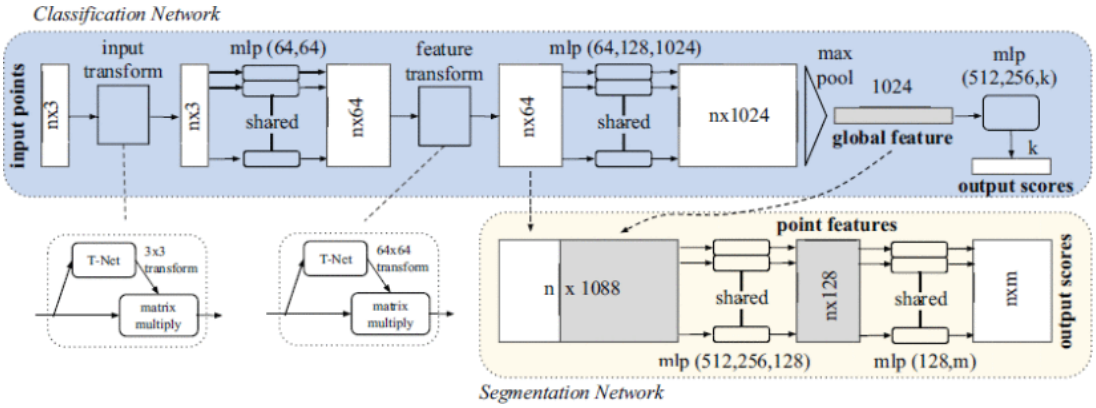

Figure 2.1: Deep learning-based network architecture for PointNet [212].

density. Direct input-based methods, on the other hand, take unordered point clouds directly as inputs. PointNet [212], as shown in Figure 2.1, was the first method to introduce a novel approach by processing raw point clouds directly. In PointNet, each point in the cloud is independently processed through a series of Multi-Layer Perceptron (MLP)s to generate individual point features. These features are then aggregated using a symmetric function, typically max pooling, to create a global feature vector that captures the overall geometry of the object. This global feature vector is used to make predictions, such as class labels or segmentation masks. The architecture is inherently invariant to the permutation of points due to the use of the symmetric function, making it well-suited for the unordered nature of point clouds. PointNet demonstrated that deep learning could be effectively applied to 3D data, setting the foundation for subsequent advancements in point cloud processing. Later PointNet++ [213] was developed by the same authors in which a hierarchical structure that captures local geometric features at multiple scales, improving its ability to handle complex and non-uniform point clouds. Following this, methods such as PointCNN [149], Dynamic Graph CNN (DGCNN) [272], SparseConvNet [85], Kernel Point Convolutions (KPConv) [256], and Point Pair Features (PPF)-FoldNet [61] have been introduced in the literature, further advancing the field of feature extraction from point clouds. A comparison of these feature extractors in terms of object recognition performance for ModelNet40 [278] data is presented in Table 2.1. From the comparison, it can be observed that these methods show good recognition performance and are able to learn useful features from the point clouds. In the subsequent sections, various methods that utilise these extracted features

Table 2.1: Comparative analysis of point cloud feature extractors in terms of object recognition performance on ModelNet40 [278] dataset.

Architecture	**Performance (Accuracy)**	**Key Features**
PointNet [212]	89.2%	Direct processing of raw point clouds; struggles with fine-grained details in complex scenes.
PointNet++ [213]	91.9%	Hierarchical feature learning; better handles complex and detailed geometries.
PointCNN [149]	92.5%	Convolutional operations on point clouds; effectively captures local structures.
DGCNN [272]	92.9%	Dynamic graph-based feature learning; excels in capturing both local and global features.
SparseConvNet [85]	93.2%	Sparse convolutional operations; efficient for processing large-scale 3D data.
KPConv [256]	92.9%	Flexible kernel point convolutions; adapts to varying geometries in point clouds.
PPF-FoldNet [61]	92.4%	Combines Point Pair Features with folding-based autoencoders; robust against partiality and occlusions.

for a range of applications are discussed, drawing on relevant literature.

2.2 Object recognition

Recognising objects in images and 3D data has been a topic of study in the literature for over three decades [317, 110, 269]. Early work in this field involved matching 3D models of objects with images to achieve recognition [167]. However, these methods required the availability of 3D CAD models, which were often not available for many common household items. As a result, much of this early work focused on recognising basic shapes within objects in images. The advent of handcrafted features and deep features, as discussed in the previous section, significantly improved the ability to detect and describe local features in images and point clouds. This advancement led to the widespread development of methods for recognising specific objects in images and point clouds. Object recognition has been categorised into two categories in literature i.e., detection and segmentation.

- **Object detection:** involves identifying and locating instances of objects within an image

or point cloud. The goal is to draw a bounding box (3D bounding box in point clouds) around objects of interest and classify them into predefined categories.

- **Object segmentation:** involves assigning a label to every pixel or point in an image or point cloud respectively, effectively separating the objects from the background. It is a more detailed task than detection. There are two main types: semantic segmentation, where all pixels/points belonging to the same object class are labelled identically, and instance segmentation, which distinguishes between different instances of the same class.

2.2.1 Object detection

The evolution of object detection in images began with methods that relied on handcrafted features, such as the Viola-Jones detector [262], HOG detector [39], SIFT detector [208, 193], and the Deformable Part-based Model (DPM) [77]. These early approaches laid the foundation for object detection by extracting distinctive features that enabled the identification of objects within images. As deep learning emerged, more advanced methods like RCNN [83], SPPNet [101], Fast RCNN [82], Faster RCNN [217], Single Shot Detector (SSD) [158], and You Only Look Once (YOLO) [122] were introduced, significantly improving both detection accuracy and prediction speed, leading to their deployment in real-world applications.

Similarly, object detection in RGB-D images has seen significant advancements, transitioning from traditional methods that extended 2D image processing techniques, like RGB-D extensions of HOG [240] and SIFT [141], to more sophisticated deep learning approaches. Early methods struggled to fully exploit depth data, but with the advent of deep learning, methods like RGB-D Faster RCNN [92] and Multi-Modal Faster RCNN [44] were developed, integrating depth as an additional channel to enhance detection accuracy. Approaches like Vote3Deep [73] and Deep Sliding Shapes [238] utilised both RGB and depth data to generate 3D bounding boxes, improving detection performance in cluttered environments. More recent advancements, such as FuseNet [98] and Depth-aware CNNs [69], leverage dual-stream architectures to process RGB and depth data separately before fusing them, further enhancing object detection by effectively combining visual and geometric cues.

In the domain of point clouds, early object detection methods also relied on handcrafted features. For instance, Johnson and Herbert [124] developed a surface matching algorithm using Spin-Image features, while Novotni and Klein introduced 3D Zernike moments [196] for shape matching. Rusu et al. [223, 222] proposed PFH and FPFH descriptors for local surface feature matching, which, although foundational, had limitations such as high computational costs, sensitivity to noise, and limited generalisation. In recent years, deep learning approaches like Frustum PointNets [211], VoteNet [64], PointRCNN [230], and Point Voxel RCNN (PV-RCNN) [231] have significantly advanced 3D object detection. These methods combine point-based and voxel-based techniques, enhancing both accuracy and efficiency for various applications, including autonomous vehicles and manufacturing.

2.2.2 Object segmentation

The evolution of object segmentation in images has followed a trajectory similar to that of object detection, beginning with methods based on handcrafted features. Early techniques such as thresholding [201], edge detection [33], and region growing [3] were foundational in dividing an image into meaningful regions. These methods were later refined by more sophisticated approaches like the Watershed algorithm [26] and Active Contours (Snakes) [129], which provided better accuracy in delineating object boundaries by considering both edge information and smoothness constraints. However, these methods had limitations in handling complex scenes with occlusions and varying illumination conditions.

With the advent of deep learning, object segmentation witnessed significant advancements. Fully Convolutional Network (FCN) [229] marked a turning point by adapting CNNs for pixel-wise prediction, enabling end-to-end learning for semantic segmentation tasks. This was followed by the introduction of U-Net [221] and SegNet [19], which utilised encoder-decoder architectures to improve the spatial resolution of segmentation maps, making them particularly effective for medical image segmentation and autonomous driving applications. The development of Mask RCNN [100] extended the Faster RCNN framework to simultaneously perform object detection and instance segmentation, further advancing the field by providing pixel-level

object masks along with bounding boxes.

The methods for object segmentation in RGB-D images generally follow the approach as object detection, using RGB-D adaptations of HOG [240] and SIFT [141]. However, these methods often struggled with the complexity of depth information and the need for precise alignment between RGB and depth data. Deep learning approaches, such as RGB-D SegNet [130] and FuseNet [98], introduced architectures that processed RGB and depth data in parallel streams before fusing the features, improving segmentation accuracy by effectively combining visual and geometric cues. More advanced models like Context-aware CNNs [34] further enhanced the capability of RGB-D segmentation by incorporating context and spatial relationships within the depth data.

In the realm of point clouds, early segmentation methods relied heavily on handcrafted features. Techniques using features like FPFH [309], and SHOT [259] were developed to capture local geometric features, enabling the segmentation of objects within 3D point clouds. These methods, however, faced challenges such as sensitivity to noise, varying point density, and computational inefficiency, particularly in large-scale scenes.

Deep learning has since revolutionised point cloud segmentation. PointNet [212] and PointNet++ [213] were the first methods to process raw point clouds directly for segmentation tasks. Other methods, such as PointCNN [149] and DGCNN [272], introduced new ways of handling the unordered nature of point clouds, using convolutional operations and dynamic graphs to better capture the spatial relationships between points. More recent approaches, like KPConv [256] and RandLA-Net [111], have further advanced the field by offering more efficient and accurate segmentation through innovative convolution operations specifically designed for point clouds and by optimising the processing of large-scale data.

2.3 Pose estimation

Once an object has been recognised in data, one of the key task for robotic applications is to estimate its pose. Pose refers to the orientation and position of an object or a camera in 3D

space. Mathematically, the pose of an object can be defined as a transformation that maps points from the object's coordinate frame to the world coordinate frame. This transformation typically consists of a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$.

$$\text{Pose} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathrm{SE}(3) \tag{2.1}$$

Here $\mathbf{R} \in \mathrm{SO}(3)$ is a 3×3 rotation matrix representing the orientation, $\mathbf{t} \in \mathbb{R}^3$ is a translation vector representing the position, SO(3) is the special orthogonal group representing rotations, and SE(3) is the special Euclidean group representing rigid body transformations (combining rotation and translation). The overall pose transformation can be applied to a point $\mathbf{p} \in \mathbb{R}^3$ in the object's local coordinate frame to obtain its position in the world coordinate frame as follows:

$$\mathbf{p}_{world} = \mathbf{R}\mathbf{p}_{local} + \mathbf{t} \tag{2.2}$$

In homogeneous coordinates, this can be written as:

$$\begin{bmatrix} \mathbf{p}_{world} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{local} \\ 1 \end{bmatrix} \tag{2.3}$$

This definition encapsulates both the orientation and the position (having 6-Degree of Freedom (6-DoF)) of the object or camera in 3D space, allowing for the calculation of how the object or camera is situated relative to a reference frame. In literature, methods for pose estimation have been proposed utilising different modalities of data *i.e.* RGB image, RGB-D data, Point Clouds.

2.3.1 Pose estimation using RGB images

Pose estimation using images has seen significant progress. Early methods for 6-DoF pose estimation were based on traditional feature matching techniques such as SIFT [144] and SURF [51]. These methods identified key points in the image and matched them to a 3D model, estimating

the pose by solving the Perspective-n-Point (PnP) problem [79]. However, these approaches often struggled with clutter, occlusions, and varying lighting conditions.

The introduction of deep learning significantly advanced 6-DoF pose estimation. PoseNet [131] was one of the first deep learning models to directly estimate camera pose from a single RGB image by regressing the position and orientation using a convolutional neural network. Later, methods like Deep6D [65] and BB8 [215] extended this approach to object pose estimation, predicting the 3D bounding box of the object and refining the pose through iterative processes.

More sophisticated methods, such as PVNet [204], RotationNet [127] and Pose Refinement by Particle Filtering (PoseRBPF) [63], have been developed to improve the robustness and accuracy of 6-DoF pose estimation in challenging scenarios. PVNet [204] uses a deep network to detect 2D keypoints and then applies a RANdom SAmple Consensus (RANSAC)-based PnP solver for pose estimation, which is particularly effective in handling occlusions. PoseRBPF [63] integrates particle filtering with deep learning to refine pose estimates iteratively, combining the strengths of probabilistic approaches with modern deep networks. RotationNet [127] is a deep learning-based model designed for 3D object recognition, which utilises multiple views of an object to predict its class and pose. By leveraging multi-view consistency, it achieves high accuracy in recognising objects across varying orientations.

2.3.2 Pose estimation using RGB-D data

The early approaches to pose estimation using RGB-D data primarily relied on traditional computer vision techniques. These methods often extended 2D image-based pose estimation techniques by incorporating depth information as an additional visual cue. One of the earliest methods was the use of feature matching techniques, such as the RGB-D extension of SIFT [251], which utilised both colour and depth features to find 3D objects in cluttered scenes. Similarly, the Point Pair Features (PPF) [89] technique was introduced to match 3D models to RGB-D data by extracting geometric features from pairs of points in the depth map, which significantly improved robustness in cluttered and noisy environments.

However, these early methods struggled with the complexity and variability inherent in

real-world scenes. The advent of Random Forests in methods like LINEMOD [104] allowed for efficient and robust 3D object detection and pose estimation by combining RGB and depth features in a discriminative manner. As deep learning gained prominence, methods [258, 117] were developed leveraging CNNs to directly regress object poses from RGB-D data. They improved the robustness and accuracy of pose estimation by learning deep features from both colour and depth channels, allowing it to handle occlusions and varying lighting conditions effectively.

More recent advancements in RGB-D-based pose estimation have focused on improving the integration of depth information. Approaches like DenseFusion [265] and DeepIM [150] use iterative refinement techniques to fuse RGB and depth data, enabling more accurate and robust pose estimation even in complex and cluttered environments. These methods apply deep learning models that process the RGB and depth information in parallel streams before merging the features, allowing the network to take full advantage of both modalities.

2.3.3 Pose estimation using point cloud data

Pose estimation have been extensively studied, with approaches ranging from traditional hand-crafted feature-based methods to modern learning-based techniques. Traditional methods, such as the Iterative Closest Point (ICP) algorithm [16], have been widely used for point cloud registration. ICP and its variants, such as Generalized-ICP [227] and Sparse ICP [30], require good initialisation to converge effectively. However, these methods are limited by their sensitivity to noise and mismatches. To address these limitations, global registration methods like Global ICP (Go-ICP) [292], Truncated Least Squares Estimation And Semidefinite Relaxation (TEASER) [291] and Super4PCS [177] have been introduced, offering globally optimal solutions at the cost of increased computational complexity. Fast Global Registration (FGR) [312] attempts to balance this trade-off by incorporating a local refinement strategy, though it still struggles with noise and outliers.

Learning-based methods have revolutionised pose estimation by leveraging deep networks designed to process unstructured sets. End-to-end registration frameworks like PointNetLK [15]

and Deep Closest Point (DCP) [271] have been proposed leveraging the point cloud processing enabled by methods like PointNet [212] and DeepSets [301]. However, these methods face challenges in handling partial-to-partial registration scenarios. To overcome this, PRNet [270] and Iterative Distance-Aware Similarity Matrix (IDAM) [145] introduced iterative refinement strategies and keypoint-based approaches to improve robustness. Robust Point Matching Network (RPM-Net) [297] and Deep Gaussian Mixture Registration (DeepGMR) [300] further extended these ideas, addressing issues related to outliers and partial observations.

In recent years, focus has shifted towards improving the integration of these methods with real-world data, which often presents challenges such as large rotations, occlusions, and varying object scales. To handle this, methods like MatchNorm [58], Geometry Correspondence Pose (GCPose) [308], CenterReg [120], and SE(3) Diffusion [121] have been proposed in literature which are specifically designed to handle the noise in the real world sensor data. Overall, the evolution from traditional to deep learning-based approaches in pose estimation and point cloud registration has led to significant advancements in accuracy, robustness, and efficiency, particularly in handling real-world challenges.

2.4 Texture-less objects

Texture-less objects, characterised by their lack of distinct surface patterns, colour variations, and visual features, pose significant challenges in computer vision tasks such as object detection, segmentation, and pose estimation. Their uniform appearances offer minimal visual cues, making traditional feature extraction methods less effective [140, 119, 205]. To address these challenges, various approaches have been developed, particularly leveraging RGB-D data and point clouds.

One effective strategy involves the use of depth information alongside RGB data to detect and estimate the pose of texture-less objects. Techniques like those proposed by Hinterstoisser et al. [102] and Drost et al. [68] employ depth data to enhance the robustness of detection algorithms against texture-less surfaces by matching geometric features or utilising PPF [119, 43]. Model-

based methods have also shown success, particularly in industrial applications where precise CAD models are available. For example, Liu et al. [157] developed a model-based approach for detecting and estimating the pose of texture-less objects using CAD models, which has proven effective in controlled environments [156]. Choi et al. [47] proposed a 3D edge based approach for pose estimation of texture-less objects utilising an efficient chamfer matching for 3D pose estimation from 2D edge templates, addressing the challenges of initialising particle filters for texture-less objects.

Deep learning has become a cornerstone in addressing the challenges posed by texture-less objects, especially when combined with depth sensors. Approaches by Tekin et al. [253] and Zakharov et al. [302] integrate depth data into CNNs to predict object coordinates directly from images, showing improved results in cluttered scenes. Additionally, frameworks like OctNet [218] and other multi-modal approaches have demonstrated the benefits of combining different types of data, such as depth and grayscale images, to enhance recognition accuracy in complex environments [205, 43].

Point cloud data offers a rich source of geometric information crucial for recognising texture-less objects. Methods like Frustum PointNets [211] and VoteNet [210] focus on the spatial arrangement of points to estimate object poses accurately, even in the absence of texture [43]. Probabilistic models have also been developed to handle uncertainties in pose estimation, offering robust solutions in industrial settings where objects may be partially obscured or placed in cluttered environments [289, 293, 249].

In practical applications, particularly in industrial automation, real-time processing is essential for the efficient handling of texture-less objects. The DOPE framework introduced by Tremblay et al. [295] demonstrates the application of deep learning for real-time 6D pose estimation in industrial settings, emphasising the need for scalable and efficient algorithms [310]. Recent advancements continue to focus on improving the generalisation of models to unseen objects and enhancing robustness in complex environments. For instance, new approaches combining synthetic and real-world data have shown promise in training models capable of handling a wide range of texture-less objects [45].

In this context, software solutions like MVTec HALCON [188], Cognex VisionPro [50], Matrox Imaging Library (MIL) [175], and Keyence Vision Systems [132] play a crucial role. HALCON is known for its robust 3D vision tools and surface-based matching, which are particularly effective for handling texture-less objects in industrial environments [188]. Cognex VisionPro's PatMax technology excels in geometric pattern matching [50], while MIL offers a user-friendly interface with strong pattern recognition capabilities [175]. Keyence integrates AI and auto-teach features, enhancing its ability to detect texture-less objects [132] efficiently. Each of these platforms brings unique strengths, making them well-suited for specific application requirements in the challenging realm of vision-based processing for texture-less objects.

2.5 Unstructured environments

Unstructured environments refer to settings that are inherently unpredictable and lack a defined or regular arrangement of objects, surfaces, or features. Unlike structured environments, where elements are typically organised in a consistent and repeatable manner—such as in factories or on roads—unstructured environments are characterised by their complexity, variability, and randomness. Examples include natural outdoor settings like forests or urban scenes, cluttered manufacturing plants, or object sorting at warehouses or any scenario where the spatial configuration is irregular and constantly changing.

In the fields of computer vision and robotics, unstructured environments present significant challenges for tasks such as object recognition, navigation, and pose estimation. These challenges arise from the absence of easily identifiable patterns, the presence of occlusions, varying lighting conditions, and the dynamic nature of such settings. As a result, more advanced algorithms and sensor systems are required to effectively interpret and interact with these surroundings. Researchers have developed innovative approaches to address issues like heavy occlusions, cluttered scenes, and the inherent uncertainty in object detection and pose estimation.

For example, Sui et al. [244] introduced SUM, a method that integrates sequential scene understanding with manipulation to estimate object poses in unstructured environments, while

accounting for uncertainty in object detection. Wong et al. [276] proposed SegICP-DSR, a real-time algorithm designed for dense semantic scene reconstruction and pose estimation. This method achieves high pose accuracy and successfully identifies object poses in the majority of test cases, making it well-suited for autonomous robotic manipulation in unstructured environments. Additionally, Feng et al. [78] proposed a Marker-Assisted Structure from Motion approach for 3D environment modelling and object pose estimation. By utilising fiducial markers in featureless environments, this method enhances pose accuracy by enabling the estimation of object poses attached to markers or cameras.

Furthermore, Collet et al. [52] developed a system for markerless 6-DoF pose estimation of complex objects in cluttered scenes, providing a solution that eliminates the need for markers or additional infrastructure. Li et al. [146] introduced a self-recognition grasping operation system that incorporates instance segmentation, pose estimation, and pose transformation for unstructured environments. This system is designed to autonomously perform object detection, location detection, and grasp planning, making it highly effective in unstructured settings. As discussed in the previous section, methods like MatchNorm [58], GCPose [308], CenterReg [120], and SE(3) Diffusion [121] have also been designed to perform pose estimation in real world unstructured environments.

2.6 Object occlusions

Occlusion poses a significant challenge in object recognition and pose estimation by obscuring key features, particularly in unstructured and dynamic environments. Various methods have been developed to address this issue. PointNett [212] and PointNet++ [213] process 3D point cloud data and remain robust under occlusions by learning local and global geometric features. ICP [16] and its deep learning extension, DCP [271]), align visible object surfaces to models, improving alignment despite partial visibility. PoseCNN [281] and PVNet [204] estimate object pose by focusing on visible regions, with PVNet [204] predicting keypoints that are robust to occlusion. LatentFusion [202] synthesises occluded object parts using latent features,

while GANs [179] infer missing regions by generating plausible reconstructions. Multi-view approaches, such as combining data from multiple cameras or LiDAR, reconstruct occluded regions to enhance recognition and pose estimation. Sparse feature-based methods, which focus on edges or corners, and advanced YOLO [122] models trained on occlusion-augmented datasets, also effectively handle partial visibility. These methods, leveraging robust feature extraction, multi-view data, and learning-based techniques, significantly mitigate the effects of occlusion, enabling more reliable performance in real-world scenarios.

2.7 Datasets

In the rapidly evolving field of computer vision, datasets are fundamental to driving research and development. They provide the essential groundwork upon which algorithms are trained, tested, and benchmarked, allowing researchers to measure progress in a systematic and standardised manner. In the domain of object recognition and pose estimation, the availability of high-quality datasets is critical. These datasets offer the necessary diversity and complexity, encompassing a wide range of object types, environmental conditions, and levels of occlusion. Such variety is vital for developing robust and generalisable models capable of performing well across different scenarios. Moreover, datasets enable the comparative evaluation of different methods, fostering innovation by highlighting the strengths and limitations of various approaches. As a result, the availability and quality of datasets significantly influence the pace of advancements in object recognition and pose estimation, making them indispensable tools for the computer vision research community.

To advance research in object recognition and pose estimation, numerous datasets have been developed, each tailored to address specific challenges such as occlusion, clutter, lighting variations, and texture-less-ness. In this thesis, a selection of significant datasets from the literature has been utilised, including the Tejani et al. [252], T-LESS [106], NYU [190], PARNET [181], MVTech-ITODD [67], LineMOD [103], Occluded LineMOD [31], TUD-L [107], YCB Video Dataset [281], and RBO Dataset [169]. These datasets were chosen for their provision of data

in both RGB-D and point cloud modalities. Additionally, many of the objects featured in these datasets are texture-less, reflecting unstructured and complex environments, thereby posing challenges that are essential for testing the robustness of the methods developed in this thesis. The properties of all the datasets is summarised in Table 2.2.

The dataset by Tejani et al. [252] is specifically designed for 3D object detection and pose estimation in environments characterised by clutter and occlusion. It includes RGB-D images of six household objects, namely a coffee cup, shampoo bottle, joystick, camera, juice carton, and milk carton. Each object is annotated with 6-DoF poses, 2D bounding boxes, and masks, making the dataset particularly useful for benchmarking in challenging scenarios. The T-LESS dataset [106] is aimed at industrial applications, featuring 30 textureless objects captured in RGB-D images. The absence of texture presents significant challenges for traditional feature-based methods, making T-LESS a standard benchmark for testing the robustness of pose estimation systems in industrial automation.

The NYU Depth Dataset (NYU v2) [190] provides RGB-D images of indoor environments, with dense annotations for semantic segmentation. The dataset's complexity, derived from the diversity of indoor scenes and varied lighting conditions, makes it invaluable for testing and developing algorithms that can handle real-world variations in indoor environments. PartNet [181] is a dataset focused on part-level object recognition and pose estimation, offering 3D CAD models of objects with detailed part segmentation. This dataset is particularly relevant for applications requiring fine-grained object understanding, such as those in robotics and augmented reality.

MVTech-ITODD [67] is similar to T-LESS with a focus on industrial applications and offers a broader range of texture-less objects. It provides high-resolution RGB-D images of 28 objects, along with precise ground truth annotations, making it suitable for benchmarking pose estimation algorithms in controlled environments. LineMOD [103] and its extension, Occluded LineMOD [31], are key benchmarks in the field of 6D pose estimation. LineMOD includes 15 household objects captured in RGB-D images, while Occluded LineMOD increases the complexity by introducing severe occlusion, challenging algorithms to maintain accuracy

Table 2.2: Summary of object pose estimation datasets.

Dataset	Data Type	Object Categories	Key Characteristics	Challenges
Tejani et al. [251]	RGB-D	Household objects	Cluttered and occluded environments, 6-DoF poses	High occlusion, clutter
T-LESS [108]	RGB-D	30 industrial, textureless objects	Textureless, symmetry	Lack of texture, rotational symmetries
NYU [190]	RGB-D	Indoor scenes	Dense semantic segmentation	Scene complexity, varied lighting
PartNet [181]	RGB-D	Part-level annotations	Focus on part-level recognition	Complexity and variability at the part level
MVTech-ITODD [67]	RGB-D	Industrial, textureless objects	High-resolution, precise ground truth	Texturelessness, high precision requirement
LineMOD [103]	RGB-D	15 household objects	Occlusion, clutter	Recognising objects under occlusion
Occluded LineMOD [31]	RGB-D	15 household objects	Extreme occlusion	Severe occlusion, clutter
YCB Video [280]	RGB-D Video	Household items, tools	Dynamic environments, video sequences	Real-time pose estimation, occlusion
TUD-L [107]	RGB-D	Common objects	Real-world environments, occlusion, clutter	Real-world variability, diverse objects
RBO [169]	RGB-D Video	Household objects	Multi-viewpoint, grasping points	Diverse poses, grasping points

under difficult conditions.

The TUD-L dataset [107] is designed for 3D object detection and pose estimation in real-world environments, featuring three common objects under varying conditions of occlusion and clutter. This dataset is particularly useful for testing algorithms in scenarios that closely mimic real-world applications. The YCB Video Dataset [281] provides RGB-D video sequences of 21 household items and tools, captured in cluttered scenes. It is widely used in robotics for real-time pose estimation and applications requiring dynamic environment interaction. Finally,

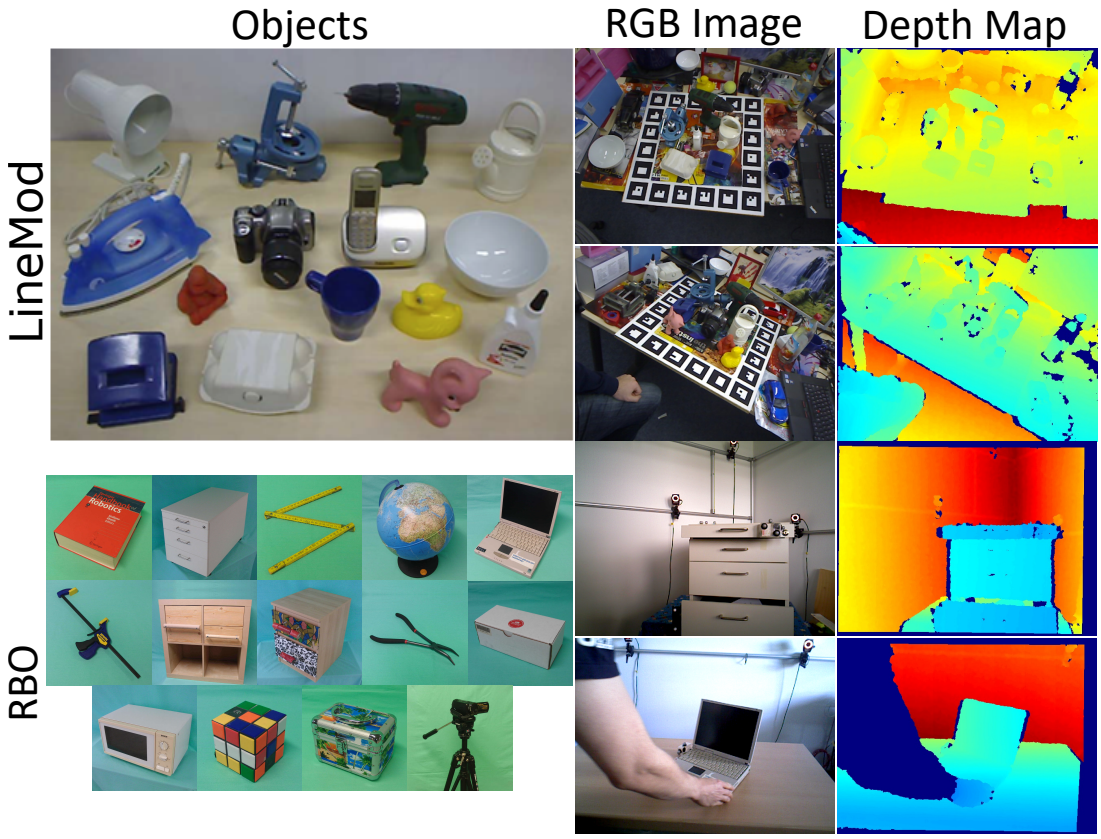

Figure 2.2: Objects and sample images from LineMod (LM) [103] and RBO datasets [169].

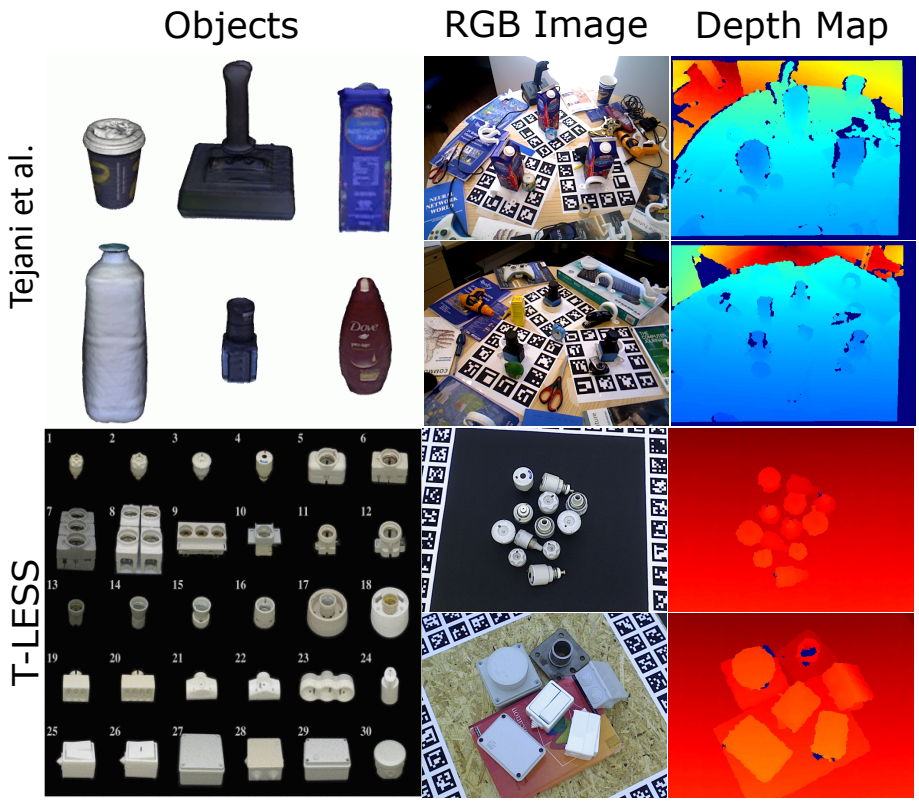

Figure 2.3: Objects and sample images from Tejani et al. [252] and T-LESS datasets [106].

the RBO dataset [169] focuses on robotic grasping and manipulation, offering multi-viewpoint RGB-D video sequences with annotations for 6-DoF poses and grasping points. This dataset is essential for developing and evaluating robotic systems that require precise object manipulation capabilities. The dataset consists of 15 objects, having articulation properties, with multiple video sequences demonstrating the manipulation of these objects.

In summary, these datasets collectively cover a wide range of scenarios and challenges that are critical for advancing object recognition and pose estimation research. Object models and some image samples for these dataset are shown in Figures 2.3 and 2.2. Each dataset contributes unique attributes that make it suitable for testing specific aspects of algorithm performance, whether it be handling occlusion, managing texture-less surfaces, or working in dynamic environments.

2.8 Camera Calibration for RGB and Depth Cameras

Camera calibration is considered a crucial process in computer vision to establish the relationship between 3D world coordinates and their 2D projections onto the camera image plane. The calibration process for both RGB and depth cameras is presented in this section, including intrinsic and extrinsic parameters, as well as depth-to-colour alignment.

2.8.1 Intrinsic Calibration

The intrinsic parameters, which define the internal characteristics of the camera such as focal length, principal point, and distortion coefficients, are determined. The intrinsic matrix $\mathbf{K}$ for an RGB camera is expressed as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

where f_x and f_y denote the focal lengths in the x and y directions, respectively, and (c_x, c_y) represents the principal point, typically located near the image centre. Lens distortion is corrected by estimating radial and tangential distortion coefficients $(k_1, k_2, k_3, p_1, p_2)$:

$$x_{\text{distorted}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2), \tag{2.5}$$

$$y_{\text{distorted}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y^2) + 2p_2 xy, \tag{2.6}$$

where $r^2 = x^2 + y^2$ denotes the radial distance.

For depth cameras, a similar intrinsic matrix is employed. However, additional calibration is required due to depth quantisation:

$$z_{\text{depth}} = \frac{d}{s}, \tag{2.7}$$

where z_{depth} is the real-world depth, d is the disparity, and s is the scale factor provided by the camera manufacturer.

2.8.2 Extrinsic Calibration

The extrinsic parameters, which describe the rigid body transformation between the camera coordinate system and the world coordinate system, are estimated. These parameters are represented by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$, forming the extrinsic matrix:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}. \tag{2.8}$$

The relationship between a 3D point in the world coordinate system $\mathbf{P}_w = [X_w, Y_w, Z_w, 1]^T$ and its projection on the image plane $\mathbf{p} = [u, v, 1]^T$ is given as:

$$\mathbf{p} \propto \mathbf{KTP}_w. \tag{2.9}$$

2.8.3 Depth-to-Colour Alignment

For systems involving both RGB and depth cameras, the alignment of the two modalities is essential. The transformation between the depth and RGB cameras is represented by:

$$\mathbf{P}_{\text{RGB}} = \mathbf{R}_{\text{align}}\mathbf{P}_{\text{Depth}} + \mathbf{t}_{\text{align}}, \tag{2.10}$$

where $\mathbf{R}_{\text{align}}$ and $\mathbf{t}_{\text{align}}$ represent the rotation and translation between the two cameras. The depth image is reprojected into the RGB camera's coordinate system using:

$$[u, v, 1]^T \propto \mathbf{K}_{\text{RGB}}\mathbf{R}_{\text{align}}\mathbf{K}_{\text{Depth}}^{-1}[x, y, z, 1]^T. \tag{2.11}$$

2.8.4 Calibration Procedure

The calibration process is conducted as follows:

- Multiple images of a known calibration pattern (e.g., a checkerboard) are captured from various angles and distances.
- The intrinsic parameters are estimated by analysing the known pattern dimensions and their projections in the images.
- The extrinsic parameters are computed by solving the Perspective-n-Point (PnP) problem.
- Depth and RGB images are aligned by estimating $\mathbf{R}_{\text{align}}$ and $\mathbf{t}_{\text{align}}$ through joint optimisation.

2.8.5 Error Metrics

The calibration accuracy is evaluated using the reprojection error:

$$e = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2, \tag{2.12}$$

where $\mathbf{p}_i$ and $\hat{\mathbf{p}}_i$ represent the observed and projected points, respectively.

This calibration framework ensures accurate 3D perception by minimising projection and alignment errors, which is crucial for applications in robotics and computer vision.

Chapter 3

Unsupervised 3D edge feature extraction

Preliminary remarks

The research in this chapter was published in the following paper:

- **Aggarwal, A.**, Stolkin, R. & Marturi, N., Unsupervised learning-based approach for detecting 3D edges in depth maps. Sci Rep 14, 796 (2024). https://doi.org/10.1038/s41598-023-50899-3

Note: Some parts of this chapter, including problem descriptions, specific equations, variable descriptions, interpretations of results, and figure/table captions, are adapted from [11].

3.1 Introduction

Among the features discussed in the previous chapter, edge features hold particular importance. Edges represent boundaries, making them highly informative for delineating the shape and structure of objects. This makes edges especially useful in scenarios where other features are difficult to extract. For example, monochrome, texture-less, and smooth metallic objects and surfaces can be challenging for systems to analyse using colour and texture information alone. In such cases, edges provide a robust alternative, enabling accurate detection and analysis of object boundaries and contours.

The advent and growing prevalence of 3D vision sensors have significantly expanded the capabilities of computer vision and robotics. These sensors, which capture detailed three-dimensional information about the environment, are increasingly being used in diverse industries such as manufacturing, energy, and medicine. To fully leverage the potential of 3D data, it is crucial to develop efficient techniques for feature extraction, particularly 3D edge estimation. 3D edges are characterised by noticeable variation or depth discontinuities in points, effectively marking the boundaries between different volumetric regions on a surface. The identification and analysis of 3D edges provide critical geometric information about objects within a scene, enabling systems to distinguish between useful foreground details and the background. Generally, there are three types of 3D edges:

- **Occluded edges**: These edges are found at boundary points on the background surface that are obscured or occluded by a foreground object. Occluded edges help in understanding the spatial relationships between objects and their environment, highlighting areas where one object blocks the view of another. Detecting occluded edges is particularly useful for tasks like object detection and scene reconstruction, where understanding the layering and depth of objects is essential.

- **Occluding edges**: These edges define the boundaries of objects present in the scene. They represent the silhouette or contour of an object as seen from a particular viewpoint, effectively outlining the shape and extent of the object. Occluding edges are critical for object recognition and segmentation, as they provide clear boundaries of where one object ends and the background or another object begins.

- **High curvature edges**: These edges occur along points of abrupt variation or high curvature on object surfaces. They signify areas where the surface undergoes a sharp change in direction, such as corners, edges of ridges, or intricate surface details. High curvature edges are valuable for capturing fine details and intricate shapes of objects, making them essential for accurate 3D modeling and surface analysis.

Each type of 3D edge carries valuable information about the geometric properties and

spatial relationships of objects within a scene. By analysing these edges, computer vision systems can gain a more nuanced understanding of the environment, facilitating more accurate and robust performance. Currently, there is relatively limited research on edge estimation in 3D data. Although 3D edge estimation shares similarities with 2D edge estimation, and numerous methods for 2D edge estimation exist [245] (as discussed in Section 2.1), 2D detectors cannot adequately handle the complexity of 3D data due to additional dimensions of depth and spatial relationships.

3.1.1 Related work

The depth information encoded in 3D vision data is generally larger in size compared to 2D images. Additionally, the process of calculating gradients and identifying local maxima in all three gradient directions for every point in 3D data has been found to be computationally intensive, as assessed by [182]. In early attempts, neighbour search and surface curvatures on object model meshes and point clouds were assessed by [197, 88] to detect 3D edges. Although refined edges of 3D models were extracted by these methods, the process was highly time-consuming. A fast edge detection method for organised point clouds was developed by Bormann et al. [29] by exploiting depth and surface discontinuities, and filtering the responses using multiple manually adjusted parameters. Hackel et al. [93] presented a contour detection algorithm for unorganised point clouds utilising the local neighbourhood surface properties of each point to detect edges.

A 3D edge detection method, utilising indexed neighbour properties of organised point clouds, was proposed by Choi et al.[46], where a point is classified as an edge if its neighbourhood angle and distance thresholds are met. A Canny-based edge detection method was developed by Sung et al.[246], in which 3D edges in the depth map were preserved by performing smoothing and morphological operations after Canny detection. Besides these, authors in [12, 195, 182, 2] have developed several other feature-based 3D edge detection methods. These methods typically identify significant variations in geometric features such as curvature, depth discontinuities, and surface normals within 3D data.

While these traditional feature-based methods are effective in many scenarios, they have some notable limitations. A major drawback is their reliance on manual parameter tuning. For instance, parameters such as thresholds for edge detection, neighbourhood size for calculating surface normals, and curvature metrics need to be carefully selected. If these parameters are not set optimally, the methods can become highly sensitive to noise present in the 3D data. This sensitivity often leads to the detection of spurious edges, which are false positives that do not correspond to actual object boundaries or features. To address these issues, this work develops a learning-based procedure to extract edges from depth maps constructed from organised point clouds. The learning-based approach leverages machine learning techniques to automatically learn and detect edges from the depth maps. The model can learn to identify complex patterns and features indicative of edges. This reduces the dependency on manual parameter tuning, as the model adapts to different conditions and types of noise through the training process. Consequently, the learning-based method is more robust and can generalise better to new, unseen data, thereby minimising the occurrence of spurious edges and improving the overall accuracy of 3D edge detection.

A refined 3D edge detection approach is developed in [112] by combining the semantic edge and segmentation features in a learning-based joint semantic edge detection and segmentation method. Local neighbourhood statistics are leveraged in the network developed by Bode et al. [27] to detect 3D edges in substantial geometric structures such as buildings. Kaneko et al. [126] utilised gradient depth images and their respective groundtruth to train the parameters of binary decision-tree for edge detection. A supervised encoder-decoder CNN-based depth contour prediction method was developed by Guerrero et al. [87] utilising a multi-modal input (RGB, depth, and normal). Sarkar et al. [225] presented an edge detection method wherein small patches in a given image are classified as edge or non-edge using deep learning. The learning-based methods discussed above require labelled ground truth data to train their networks, which is not always available. Furthermore, it is expensive and time consuming to create an acurrate labelled dataset. To address the unavailability of labelled data, the method described in this work utilises an unsupervised learning approach.

A refined 3D edge detection approach was developed in [112] by combining semantic edge and segmentation features in a learning-based joint semantic edge detection and segmentation method. Local neighbourhood statistics were leveraged in the network developed by Bode et al. [27] to detect 3D edges in substantial geometric structures such as buildings. The parameters of a binary decision tree for edge detection were trained by Kaneko et al. [126] using gradient depth images and their respective ground truth. A supervised encoder-decoder CNN-based depth contour prediction method was developed by Guerrero et al. [87], utilising a multi-modal input (RGB, depth, and normal). An edge detection method was presented by Sarkar et al. [225], in which small patches in a given image were classified as edge or non-edge using deep learning. The learning-based methods discussed above require labelled ground truth data to train their networks, which is not always available. Furthermore, the creation of an accurate labelled dataset is expensive and time-consuming. To address the unavailability of labelled data, the method described in this work utilises an unsupervised learning approach.

Unsupervised learning has been extensively studied in the literature for tasks like classification, clustering, and anomaly detection[123, 226, 187] due to its ability to understand intrinsic patterns within data. In the context of edge detection, it has also been explored for 2D edge estimation in literature [142, 147]. An encoder-decoder architecture based method is presented by Le et al. [142] which utilises a feedback loop for iterative enhancement of the predicted edges. Li et al. [147] consider a sequential data from video input to compute flow image and utilises its gradients for 2D edge detection.

3.1.2 Organisation

The rest of the chapter is organised as follows. Section 3.2 provides a detailed description of the proposed contributions in this chapter. A background on technical concepts utilised in this wor is given in Section 3.3. Section 3.4 describes the proposed network to estimate 3D edges as well as its components. Section 3.5 presents a comprehensive analysis of the results, accompanied by descriptions of the datasets and comparison methods. Section 3.6 discusses a practical analysis with an ablation study. Finally, Section 3.7 presents a summary.

3.2 Contribution

This work proposes an unsupervised 3D edge detection and parameter selection approach using deep learning [11]. The proposed approach integrates advanced deep learning techniques with traditional gradient-based methods to enhance the accuracy and robustness of 3D edge detection. A multi-scale encoder-decoder network is employed to learn features from depth maps. The encoder-decoder architecture allows the network to capture multi-scale contextual information by progressively downsampling and then upsampling the input depth maps. This multi-scale approach is crucial for effectively capturing both fine and coarse details in the depth maps, which are essential for accurate edge detection.

From the learned features, edge-specific features are extracted. These features are designed to capture the characteristics of edges in the 3D space, leveraging both the learned representations from the deep network and traditional gradient-based features. Once these edge features are extracted, a clustering algorithm is applied to classify each point in the depth map as an edge or non-edge. The clustering process groups points with similar edge features, enabling the identification of distinct edges without the need for labelled ground truth data. This unsupervised approach reduces the dependency on manually annotated datasets, which are often scarce and expensive to produce.

Of all the available ones, in this work, specific focus is given to estimating occluding and high curvature edges. These types of edges are of significant interest for various practical applications, including robotic grasping, manipulation, and segmentation. In robotic grasping and manipulation, accurate detection of object boundaries and surface details is crucial for planning and executing precise movements. High curvature edges, in particular, provide essential cues for identifying grasp points and understanding object shapes. In segmentation, distinguishing between different objects and their parts relies heavily on accurately identifying their boundaries, making occluding edges particularly important.

The main contributions of the proposed method are summarised as follows [11]:

- **A multi-stage network architecture**: The proposed architecture comprises four key

components:

1. **Encoder-decoder**: This component captures multi-scale features from the depth maps by progressively downsampling the input to encode contextual information and then upsampling to decode detailed features.
2. **Multi-size split**: This stage splits the encoded features into multiple scales or sizes, enabling the network to process information at different resolutions simultaneously. This multi-resolution approach helps in capturing both fine and coarse details in the 3D data.
3. **Edge feature extractor**: This module specifically focuses on extracting features that are indicative of edges. By combining learned features from the network with gradient-based features, it enhances the accuracy of edge detection.
4. **Cluster network**: The final stage involves a clustering network that classifies each point in the depth map as an edge or non-edge. This unsupervised clustering algorithm groups points based on their extracted edge features, facilitating robust edge detection without the need for labelled data.

- **A non-learning pre-processing algorithm**: To address the issue of missing depth information (often referred to as shadows) in the input depth data, a pre-processing algorithm is introduced. This algorithm filters noise and completes the missing depth information, ensuring that the depth maps used by the network are of high quality and free of artefacts that could affect edge detection performance.

- **An unsupervised training strategy**: The method employs an unsupervised training strategy that eliminates the need for labelled data. By leveraging clustering techniques and unsupervised learning algorithms, the method is capable of training without relying on manually annotated datasets. This not only reduces the cost and effort associated with data labelling but also makes the method more versatile and applicable to a wide range of scenarios where labelled data may not be available.

The proposed method has been compared with four State-Of-The-Art 3D edge detection methods from the literature. It has been evaluated on five benchmark datasets, which include both single and multi-object scenes. The results demonstrate that competitive performance is achieved by the proposed method, despite the absence of labelled data and the need for hand-tuning of key parameters. This highlights the effectiveness of the unsupervised approach and the robustness of the multi-stage network architecture in accurately detecting 3D edges from depth maps. Additionally, the evaluation showcases the method's ability to generalise across different types of scenes and objects, making it a promising solution for various applications in computer vision and robotics.

3.3 Background

3.3.1 Encoder-decoder networks

An encoder-decoder network is a type of neural network architecture widely used in deep learning applications, particularly those involving sequence-to-sequence tasks, image processing, and data transformation. This architecture consists of two main components:

- **Encoder:** The encoder processes the input data, compressing it into a fixed-size representation known as a latent space or bottleneck. This representation captures the essential features of the input while discarding irrelevant details [261].
- **Decoder:** The decoder reconstructs the compressed representation into the desired output format. This output could be a transformed version of the input, such as in image segmentation, or a completely new form of data, such as in machine translation [24, 105].

The encoder-decoder network is trained on a large dataset where input-output pairs are known. During training, network parameters are adjusted by minimising the difference between the predicted output and the actual output. This difference is measured by a loss function, such as Mean Squared Error (MSE) for regression tasks or cross-entropy loss for classification tasks. Once trained, the network can be used for inference on new, unseen data. The encoder processes

the new input to generate a latent representation, which the decoder uses to produce the final output. This process enables the network to generalise to new data not encountered during training [84].

The encoder-decoder architecture has been successfully applied in various domains, including image segmentation, machine translation, speech recognition, data compression, and anomaly detection [17]. Overall, this network structure is a powerful and flexible tool for efficient feature learning and data transformation across a wide range of deep learning applications. Its ability to handle complex data and generate meaningful outputs makes it a cornerstone of modern neural network design.

3.3.2 Deep unsupervised clustering

Deep unsupervised clustering is a rapidly growing area within the field of machine learning, where the primary objective is to group similar data points into clusters without relying on labelled data. Unlike supervised learning, which depends on a pre-defined set of labelled examples for model training, unsupervised clustering seeks to uncover the inherent structure within data by identifying patterns based solely on the data's intrinsic features [273]. This technique is particularly valuable in scenarios where labelled data is scarce or costly to obtain.

Deep unsupervised clustering leverages deep learning models, such as autoencoders, CNNs, and generative models, to extract high-level features from raw data [311]. These features are then used to perform clustering in a meaningful manner. The combination of deep learning with clustering has proven powerful, as it enables models to learn complex data representations that are more effective for clustering tasks compared to traditional methods that depend on hand-crafted features [224].

Key components of deep unsupervised clustering

- **Feature extraction:** The first step in deep unsupervised clustering involves extracting features from raw data using deep neural networks. For this purpose, autoencoders are commonly utilised, where the input data is compressed into a lower-dimensional latent

space by the encoder, and the original data is then reconstructed from this compressed representation by the decoder. The learned latent space captures the most relevant features of the data, which can then be employed for clustering.

- **Clustering layer:** After feature extraction, the latent representations are fed into a clustering layer, where algorithms such as k-means, Gaussian Mixture Models (GMM), or other clustering techniques are applied [233]. Some advanced methods integrate the clustering process with neural network training, allowing the model to simultaneously learn feature representations and cluster assignments. This integration fine-tunes the feature space specifically for clustering tasks.

- **Loss functions:** Specialised loss functions are designed to optimise clustering performance. These functions typically aim to minimise intra-cluster variance while maximising inter-cluster distance. A common approach is to use the Kullback-Leibler (KL) divergence, which measures the difference between the current cluster distribution and a target distribution. The network is trained to reduce this divergence, thereby enhancing clustering quality.

Applications in vision and 3D vision fields

Deep unsupervised clustering has become highly valuable in various vision and 3D vision tasks. Its successful applications span several areas, including image segmentation, anomaly detection, object detection and recognition, 3D point cloud segmentation, 3D shape analysis and reconstruction, and scene understanding. These techniques enable the organisation and analysis of complex visual data without the need for labelled datasets, making them particularly useful for tasks requiring scalability and adaptability [185].

3.4 Methodology

Features such as surface normals, X, Y, and Z directional gradients, and neighbourhood surrounding properties can be utilised to identify edges in point cloud data. Traditionally, manual

fine-tuning of multiple thresholds is required by the method to process these features and classify points in the point cloud as edge or non-edge. This process is not only laborious but also prone to errors. Moreover, the adaptability for diverse datasets is often affected by using fixed thresholds. Therefore, a clustering-based automated threshold estimation approach has been designed to improve the generalisation over different datasets. Before delving into the technical specifics of the method, the next subsection provides a detailed introduction to the problem.

3.4.1 Problem formulation

Given an organised point cloud with **D** points, in the form of a $W \times H$ size depth map $\mathcal{D}$, edge points $\mathbf{E}_p$ where $\mathbf{E}_p \subset \mathbf{D}$ are detected by finding the λ threshold. Then, each edge point $e \in \mathbf{E}_p$ in the depth map $\mathcal{D}$ is defined as:

$$e_{i,j} = \begin{cases} d_{i,j}, & \text{if } \delta_{i,j} > \lambda \\ 0, & \text{otherwise} \end{cases}, \tag{3.1}$$

where $d_{i,j}$ and $\delta_{i,j}$ represent the depth value and extracted edge feature at the i, j location with $i \in 1, 2, \cdots, W$ and $j \in 1, 2, \cdots, H$, respectively. The edge features are extracted at all the points **D** for all the depth maps in the dataset to obtain a distribution $\mathbf{\Gamma}$, as shown in Figure 3.1. A *k-means* clustering-based approach is employed utilising deep learning to learn means m_1 and

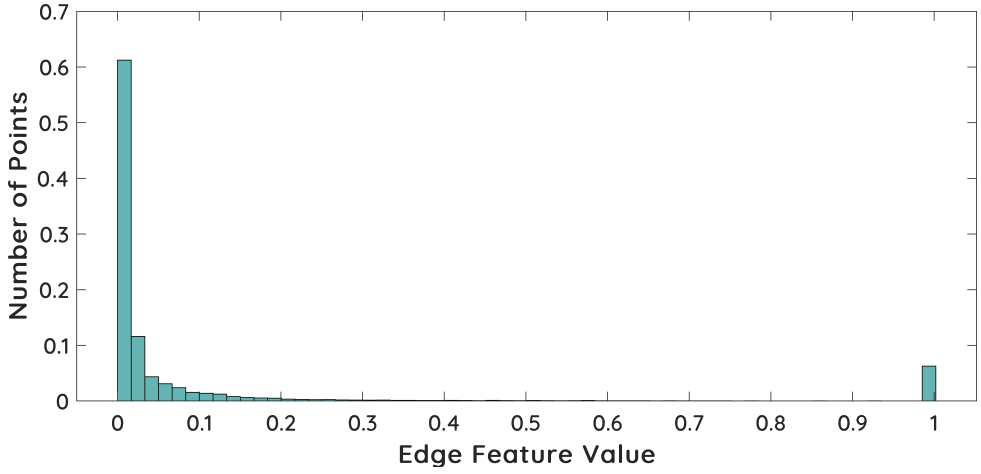

Figure 3.1: Normalised feature space distribution for clustering edge and non-edge points.

m_2 for two clusters in the distribution $\mathbf{\Gamma}$, with each mean representing the non-edge and edge regions respectively. Using these means, the optimal threshold λ is defined as:

$$\lambda = \arg\min_{\mathbf{\Gamma}} \{|(m_1 - \mathbf{\Gamma})| > |(m_2 - \mathbf{\Gamma})|\} \tag{3.2}$$

Gradient-based features are adapted in the proposed method to compute edge specific features. The gradient feature $\delta_{i,j}$ at i, j index in depth map is defined as in equation (3.3).

$$\delta_{i,j} = \sqrt{(\mathbf{G}_X * \mathbf{D}_{i,j})^2 + (\mathbf{G}_Y * \mathbf{D}_{i,j})^2} \tag{3.3}$$

Here, the gradient operators $\mathbf{G}_X$ and $\mathbf{G}_Y$ evaluate the change of depth in the depth map by applying a convolution operation $(*)$ on the local region matrix $\mathbf{D}_{i,j}$, defined as:

$$\mathbf{D}_{i,j} = \begin{bmatrix} d_{i-1,j-1} & d_{i-1,j} & d_{i-1,j+1} \\ d_{i,j-1} & d_{i,j} & d_{i,j+1} \\ d_{i+1,j-1} & d_{i+1,j} & d_{i+1,j+1} \end{bmatrix} \tag{3.4}$$

It is important to note that the gradient-based edge detection features utilised here can be substituted with any other edge detection features. To enable generalised unsupervised clustering, the edge feature extraction module is integrated with the Deep Neural Network (DNN)-based learning. The proposed network architecture for 3D edge detection in depth maps is illustrated in Figure 3.2.

The overall network architecture is divided into two main components: feature-based and learning-based. Pre-processing steps and an edge feature extractor are part of the feature-based component to calculate predefined features from the raw data. On the other hand, the learning-based component consists of an encoder-decoder network and a clustering sub-network. The encoder-decoder network is designed to learn complex, hierarchical features directly from the data, capturing intrinsic patterns and representations. The clustering sub-network further processes these learned features, organising the data into two groups *i.e.* edge or non-edge. Each of the components are described in detail in the following subsections in order of the flow

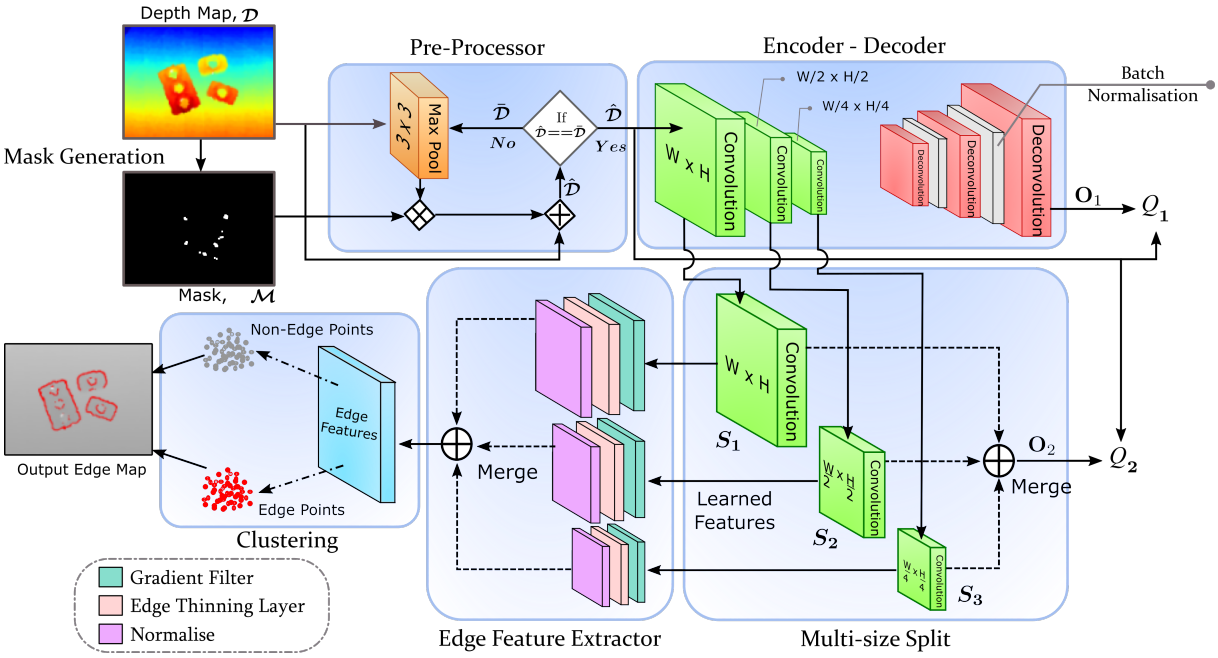

Figure 3.2: Proposed network architecture for 3D edge detection from depth maps [11].

of data in the network as shown in Figure 3.2.

3.4.2 Pre-processing

Empty information patches is an issue that arises in the depth data captures by most of depth sensors. Such an issue occurs from absorbing and reflective surfaces or shadows [57, 306]. Such empty patches can negatively impact the performance of edge detecting algorithms, since points surrounding these empty patches exhibit abrupt discontinuities which leads to difficulties in learning a threshold. To address this, a pre-processing algorithm has been designed that aims to eliminate empty patches from the input depth maps. It is to be noted that the final output of the network remains consistent with the original input depth map while excluding the empty regions.

In the literature a variety of shadow removal algorithms have been discussed. A deep learning-based method for completing a depth map was introduced by Zhang and Funkhouser in [306], however, to support the depth map completion in this task, they utilised color images. A low gradient regularisation algorithm for in-painting missing depth information was proposed by Xue et al. [288]. Furthermore, Yang et al. developed a depth map in-painting approach using

the geometry of light field epipolar plane images [294]. Despite their good performance, these techniques are too slow for online or real-time applications. In addition, a morphology-based algorithm was developed by Danciu et al. [57] to remove shadows from depth maps. This work proposes a similar yet simpler shadow removal method that has been proposed for pre-processing, which is outlined in Algorithm 1. The corresponding flow graph for the developed method is depicted in the pre-processor stage of Figure 3.2.

Algorithm 1 Shadow filtering in depth maps.

Require: $\mathcal{D}, \mathcal{M}$
Ensure: Filtered depth map $\widehat{\mathcal{D}}$
1: $\widehat{\mathcal{D}} = \mathcal{D}$
2: $\overline{\mathcal{D}} = \mathbf{0}$
3: **while** $\widehat{\mathcal{D}} \neq \overline{\mathcal{D}}$ **do**
4: $\quad \overline{\mathcal{D}} = \widehat{\mathcal{D}}$
5: $\quad \widehat{\mathcal{D}} = \text{MaxPool}(\overline{\mathcal{D}})$
6: $\quad \widehat{\mathcal{D}} = \widehat{\mathcal{D}} \circledast \mathcal{M}$
7: $\quad \widehat{\mathcal{D}} = \widehat{\mathcal{D}} \oplus \mathcal{D}$
8: **end while**

A max depth selection approach is designed for this method by employing a 3×3 size max-pooling layer to extract the maximum depth value in local region of the depth map. The pool size is designed to be analogous to the radius of the local region. The selection of pool size for pre-processing is critical as a smaller pool size may provide improved resilience to noise but may have a high processing time. Conversely, larger pool sizes perform faster operations but may be greatly affected by the random noise in the data. Through the experimental analysis as discussed in Section 3.6, the 3×3 filter size indicates a reasonable trade-off.

The inputs to the pre-processor include depth map and the shadow mask corresponding to depth map. All the empty regions in the depth map are generated into a shadow mask. Initially, the depth map goes through the max-pooling layer, and its output is element-wise multiplied ($\circledast$) with the mask. The element-wise addition ($\oplus$) is then performed on the resulting map. This process continues until no further changes occur. Once pre-processed, the depth map is passed on to the next stage as input. The sample results of the performance of the proposed pre-processor is shown in Figure 3.3, where the circles represent the shadow regions. The

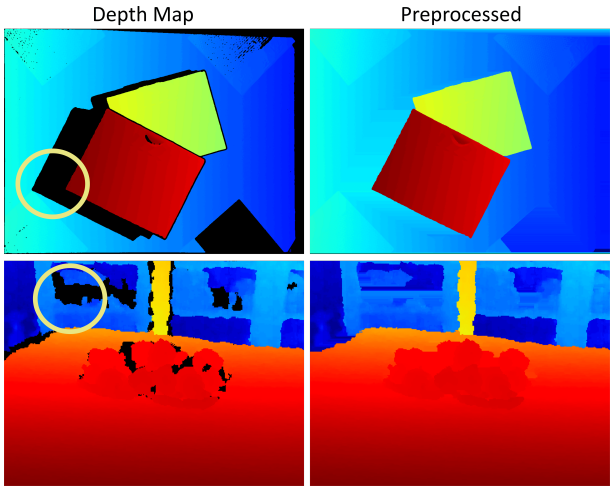

Figure 3.3: Affect of pre-processing process for 3D edge detection in two different scenes.

method achieves an average pre-processing time of roughly $20 - 30$ ms per depth map.

3.4.3 Encoder-decoder architecture

As discussed in Section 2.1, extracting intrinsic features from the data has been extensively done utilising an encoder-decoder deep network architecture [283, 142, 147]. In this work, a three-stage network has been designed to learn features from the input depth map, using convolution and deconvolution layers for the encoder and decoder respectively. Three side outputs, S_1, S_2, and S_3, have been considered from the encoder to extract features at different scales, as shown in Figure 3.2. The side output structure in the network is inspired from Holistically Nested Edge Detection (HED) [284] learning-based method designed for 2D edge extraction.

For the encoder, a convolution layer-based three-stage network is designed with the data size reducing the half in subsequent layers as compared to the previous layer. A kernel size of 3×3 with channel sizes of $(16, 16, 16)$ are configured for each encoding layer. As the depth value for the background is higher as compared to the foreground objects, pooling layers cannot be utilised to reduce the size of the depth map in subsequent layers; instead a stride of 2 is utilised after the convolution operation to achieve this task.

Each convolution layer is followed by a ReLU activation function [9] to introduce non-linearity. The output of each encoding layer is taken as a side output, which is then passed through a convolution operation to generate a single-channel feature map. Learned features at

three different scales, specifically $1\times, 0.5\times, 0.25\times$ of the original depth map, are provided by these side outputs. Edges at varying levels of abstraction are detected with the assistance of these side outputs. The abstraction improves the detection of edges around object boundaries, a benefit that has been analysed through experiments and discussed in Section 3.6.

A key distinction of the proposed method, compared to HED [284], is that this method does not train individual side outputs independently; instead, deconvolution upscaling layers are employed to fuse the side outputs, allowing the network to inherently learn essential features at each scale. The encoded features are then decoded back to a map of the same size as the input by upscaling through three deconvolution layers defined in the decoder. Each deconvolution layer is configured with a kernel size of 3×3 and channels $(16, 16, 1)$ in the corresponding layers. The first two deconvolution layers use ReLU activation functions, while a sigmoid activation function [189] is applied in the final layer.

To prevent overfitting and accelerate training, batch normalisation layers [116] are incorporated after the first and second deconvolution layers. As the expected output from the encoder-decoder is intended to match the pre-processed input, self-supervised learning strategy is employed to train the encoder-decoder network. MSE between the outputs $\mathbf{O}_1$ and $\mathbf{O}_2$ is calculated using the pre-processed depth map $\widehat{\mathcal{D}}$ as the target, to calculate errors $Q1$ and $Q2$, which is then minimised to perform the network learning. These two errors, also referred to as losses, are calculated as:

$$\begin{aligned} Q1 &= \frac{1}{WH} \sum_{i}^{W} \sum_{j}^{H} \left[\mathbf{O}_1(i,j) - \widehat{\mathcal{D}}(i,j) \right]^2 \\ Q2 &= \frac{1}{WH} \sum_{i}^{W} \sum_{j}^{H} \left[\mathbf{O}_2(i,j) - \widehat{\mathcal{D}}(i,j) \right]^2 \end{aligned} \tag{3.5}$$

The encoder-decoder block layers are trained using the $Q1$ loss, to ensure generalise learning from the training data. In contrast, the $Q2$ loss is employed to maintain the integrity of the input scene structure in the side outputs during the learning process. The error gradients are then back-propagated to train the encoder-decoder and multi-size split blocks.

3.4.4 Edge feature extractor

The edge feature extractor is a non-learning feature extraction layer within the proposed network. Specifically, it is employed to extract edge features from the multi-size side outputs of the encoder-decoder. This layer has been designed to serve as a dynamic component of the overall system, with the primary purpose of capturing application-specific features (such as corners, texture, etc.) in addition to the learned features within the model. In this work, the layers have been constructed to extract edge features.

A three-stage edge feature extractor is designed in the proposed method with stages: gradient filter, edge thinning layer, and normalisation layer. The gradient filter layer has been implemented as a convolution operation with two predefined kernel weights for height (H) and width (W) direction. Sun et al. [245] discusses the various types of gradient kernels studied in the literature to perform edge detection. In this work, four different kernels have been considered for edge feature extraction: Sobel, Roberts, Prewitt, and Laplacian of Gaussian (LoG). A detailed explanation of each of these kernels is provided in Appendix A. The proposed edge feature extractor consists of three stages: the gradient filter, the edge thinning layer, and the normalisation layer. The gradient filter layer is implemented as a convolution operation with predefined kernel weights for both the height (H) and width (W) directions in the depth map. In the literature, multiple types of gradient kernels have been studied to perform edge detection [245]. In this work, four different kernels are considered for edge feature extraction: Sobel, Roberts, Prewitt, and Laplacian of Gaussian (LoG). A detailed explanation for each of these kernels is provided in Appendix A.

The next stage of the feature extractor is the edge thinning layer. This layer has been designed to filter out noisy edges detected by the gradient filter and to thin down the calculated gradients around the edges. It has been implemented as a minimum-pooling operation with a kernel size of 2, selected after careful analysis through experimentation. Finally, *min-max* normalisation is applied to the extracted features to standardise the data. Normalisation is performed to facilitate the clustering of edge and non-edge points in the next stage of the network. Edge features are extracted for each of the side outputs of the encoder-decoder network to detect edges at different

scales. These edge features at each side output are upscaled to match the input map size and then fused together to produce a single edge-feature map as the output of this layer.

3.4.5 Learning-based clustering

The unsupervised classification of the network features into edge or non-edge points is facilitated by the deep embedded clustering approach, employed in the proposed method. Stochastic gradient descent-based learning is utilised to learn the cluster parameters by optimising the cluster objective function. In this work, the extracted edge features are classified into edge or non-edge points utilising *k-means* clustering, wherein features are associated with a cluster mean learned for each class. As there are two classes *i.e.* edge or non-edge, clustering can be categorised as a binary classification problem. Accordingly, the number of clusters has been defined as two. The decision for each point in the edge feature map to associate with one of the clusters is performed by calculating a soft probability utilising the Student's t-distribution [161] approach, as shown in the equation below.

$$p_{i,j}^{k} = \frac{(1 + ||\delta_{i,j} - m_k||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k'}(1 + ||\delta_{i,j} - m_{k'}||^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{3.6}$$

where i, j represents the coordinates of a point on the edge feature map and $p_{i,j}^{k}$ denotes the soft probability for point at i, j belonging to the k-th cluster. In the proposed method, $k \in 1, 2$, where let if $k = 1$, then $k' = 2$ and vice versa. The cluster means m_k and $m_{k'}$ are learned by this layer. α represents the degree of freedom component in the probability calculation, directly determining the impact of the Euclidean distance between the mean and a given point. In the experiments, α is set to 1. The unsupervised learning is performed to learn the cluster means using the Kullback-Leibler-Divergence (KL-Divergence) loss. KL-Divergence is a measure used in probability theory and information theory to quantify the difference between two probability distributions. Specifically, it measures how one probability distribution diverges from a second, reference probability distribution. To facilitate this, a temporary target probability distribution T is computed at the start of each epoch over the entire training set. The target distribution T

comprises the current probabilities for each point in each sample of the dataset. It is calculated at the beginning of each training epoch by accumulating the predictions of each sample without back-propagation, before the commencement of the training iterations. During each iteration within the epoch, the difference between the current distribution C and the target probability T is computed using the KL-Divergence loss [139], as given in equation below,

$$\mathrm{KL} = T * \log(T/C) \tag{3.7}$$

Minimising KL-Divergence loss ensures that the current distribution closely approximates the target distribution, thereby refining the model's predictions. In the context of edge detection, this helps in maximising the separation between edge and non-edge point distributions, improving the accuracy and robustness of the model. Gradients derived from the KL-Divergence loss are back-propagated through the network to update model parameters during training. Therefore, the total loss used for training the network is given by,

$$L = \mathrm{KL} + L_1 + L_2 \tag{3.8}$$

The clustering layer produces an edge map where each point is classified as either an edge or non-edge based on the soft probabilities. The deep embedded clustering technique is adopted from the work by [283], where the clustering of features is optimised by minimising the change in clusters over multiple epochs. Assuming that the objects are predominantly present in the centre of the sensor view, a centred masking layer is considered before the clustering layer. This masking layer selectively focuses on the central region of the depth map, where the objects of interest are most likely to be located, while ignoring the peripheral areas that are more likely to contain background noise. By doing so, unwanted background noise is removed from the features before performing clustering. This step helps to enhance the quality of the features that are fed into the clustering layer, leading to more accurate and robust edge detection.

It should be noted that this masking is optional and is mainly employed for experimental purposes to assess its effectiveness in various scenarios. The centred masking layer is not

an integral part of the network, and its inclusion or exclusion does not fundamentally alter the network's architecture or its core functionality. Instead, it serves as an additional tool to potentially improve performance in cases where the central region assumption holds true. During the experiments, it is observed that the use of the centred masking layer can lead to improved clustering results by reducing the impact of irrelevant background features, as discussed in Section 3.6. However, in real-world applications where objects may not always be centred, this layer may be omitted without significantly affecting the overall performance of the network. By providing flexibility in the network design, the optional nature of the centred masking layer allows for adaptability to different datasets and operational conditions. This flexibility ensures that the network can be tailored to specific use cases, maximising its effectiveness in diverse environments.

3.4.6 Network model discussion

The proposed network model was developed after conducting multiple experiments on various layer configurations to optimise performance and generalisation capabilities. Initially, a simple network architecture was considered with a single-layer feature extractor and clustering mechanism to detect edges in the depth maps. This straightforward network performed the task to some extent but failed to generalise for the variations in the 3D data. Specifically, edges were captured with less accuracy around the structure of objects, leading to suboptimal performance in edge detection.

To address these limitations, encoder-decoder layers with three side outputs were integrated before the feature extraction layer. The encoder-decoder architecture is well-known for its ability to learn hierarchical features at multiple scales, which is essential for handling the complexity of 3D data. The inclusion of three side outputs allowed the network to capture detailed edge information at different levels of abstraction, thereby improving the overall edge detection around object structures. Although the learning of the three side outputs could be performed individually, it was decided to combine them into the original image size to avoid additional processing overhead. This combination was achieved by experimenting with both upscaling

and deconvolution operations. During these experiments, it was observed that upscaling the smaller scale side outputs led to poor feature learning. Conversely, deconvolution proved to be effective in learning features from the side outputs, as it could better retain the spatial information necessary for accurate edge detection.

In the edge feature extractor, the kernel size of the edge thinning layer was selected empirically after testing multiple kernel sizes $1, 2, 5, 7$. Using a kernel size of 1 meant that no edge-thinning operation was applied. While this approach preserved all gradients, it made the network highly vulnerable to noise, leading to a decline in performance. Higher kernel sizes of 5 and 7 caused some essential gradients to vanish, leading to incomplete edge detection. Through this empirical testing, a kernel size of 2 was found to be optimal, balancing the reduction of noise and the retention of critical edge information.

The merging of edge features extracted from the side outputs was another crucial decision point. Both upscaling and deconvolution were considered for this purpose. Ultimately, upscaling was selected for merging edge features because it preserved the gradient values in the merged image, which is critical for subsequent clustering operations. By maintaining the integrity of the gradients, the network could perform more accurate clustering, leading to better edge detection results.

In summary, the final network architecture, as presented in Figure 3.2, was chosen after extensive experimentation and analysis. This design includes encoder-decoder layers with side outputs, deconvolution operations for upscaling, and an empirically chosen kernel size for the edge thinning layer. The next section discusses the detailed evaluation results, demonstrating the effectiveness of the proposed design in achieving robust and accurate edge detection in 3D data.

3.5 Experimental results

The effectiveness of the developed approach is demonstrated across noiseless and real-world noisy sensor data through experiments on five benchmark datasets. Four State-Of-The-Art

methods: Choi et al. [46], fast edge [29], Sung et al. [246], and JSENet [112] are considered for performance comparison. Major analysis is performed by evaluating the point cloud registration accuracy for the detected edge points onto the object model cloud. The performance analysis is systematically divided into two parts: single object analysis and multi-object analysis. For both parts, the accuracy and robustness of the detected edges is assessed through pose registration and F-measure comparison analysis.

3.5.1 Dataset description

Tejani et al. [251], T-LESS [108], PartNet [181], NYU [190], and MvTech-ITODD [67] are the five benchmark datasets utilised in this work for evaluation. The Tejani et al. [251] dataset provides noise-free synthetically generated depth maps for various household objects. Real-world depth sensor captured data for single and multi-object scenarios featuring industrial objects is provided by the T-LESS [108] dataset. The PartNet [181] dataset comprises object model clouds with part segmentation labels, which are then exploited to generate 3D edge ground truth labelled depth maps for experimental analysis. 464 multi-object indoor scene depth maps are provided in the NYU [190] dataset, including high occlusions and noise. The MvTech-ITODD [67] dataset provides high-definition real-world multi-object depth maps depicting industrial objects.

5000 and 500 depth maps distributed across different objects and scenes are utilised for training and testing respectively for the single object analysis from Tejani et al., T-LESS, and PartNet datasets. For multi-object scene analysis, 1200 and 250 depth maps are considered for training and testing from T-LESS multi-object and the NYU dataset. From MvTech-ITODD dataset, 800 depth maps were utilised for training and 150 depth maps are considered for testing. The training and testing sets are mutually exclusive to ensure unbiased evaluation.

For detailed analysis, specific objects were selected: from the Tejani et al. dataset Milk, Camera, Joystick, Juice Carton, Cup, and Shampoo; objects with ID-numbers 2, 5, 8, 17, and 27 from the T-LESS dataset; Bowl, Bottle, Knife, Scissors, and Mug from the PartNet dataset. Additionally, all 464 scenes from the NYU dataset and all 28 objects from the MvTech-ITODD

dataset are used.

3.5.2 Experimental setup

Tensorflow [1] library is utilised to implement the proposed network in Python. A single Nvidia Tesla P100 GPU is utilised to train the network. During training, 32 Giga-Byte (GB) of Random Access Memory (RAM) and 16 GB of Graphics Processing Unit (GPU) resources are utilised by the proposed network. While the inference model requires less than 2 GB of GPU memory and 4 GB of RAM. The high RAM requirements are due to the intensive calculation of the target distribution, which can increase with larger datasets. The network trains in approximately 4 hours for the depth map size 640×480. The network inference time (including the pre-processing) is about 60-80 milliseconds, without including the depth map loading time.

λ_c, λ_f, and λ_s represents the threshold values defined for Choi et al. [46], Fast Edge [29], and Sung et al. [246] methods, respectively. Each threshold comprises two values. Depth discontinuity and maximum neighbour search are the two values represented by λ_c. λ_f comprises depth step factor and minimum edge angle threshold. The first value in λ_s is sigma in colour space and the second value is sigma in coordinate space for the smoothing filter. The values for these thresholds is selected based on the empirical study of the performance of the respective methods on a subset of considered datasets. For this study, threshold values are selected from the range of values provided in their respective works [46, 29, 246]. The final thresholds are selected based on their average performance over all the objects in the respective datasets. JSENet [112] is trained for 200 epochs using the training parameters as provided in [112], on PartNet [181] and NYU [190] datasets.

3.5.3 Evaluation metrics

Pose registration and F-measure metrics are utilised to evaluate and compare the performance of the proposed method with SOTA approaches. Pose registration is achieved by aligning the detected edge points with the corresponding 3D object model, utilising the RANSAC algorithm. A homogeneous transformation matrix is obtained through registration, from which rotations

and translations are extracted for further evaluation.

The following performance metrics are computed for quantitative analysis:

1. $||A||_F$: Frobenius norm of the difference between the registered and ground truth transformation matrices.

2. $\mathbf{R}_{err}$: Rotation error.

3. $\mathbf{T}_{err}$: Translation error measured as the Euclidean distance.

4. $\boldsymbol{Z}_E$: Average Distance (ADD) of two model points transformed by the registered transformation and ground truth transformation, respectively.

Additionally, for the datasets with available labelled ground truth edge data *i.e.* PartNet [181] and NYU [190] datasets, the accuracy of the proposed method is assessed using the F-measure metric. F-measure is calculated utilising the soft precision and recall metrics, based on the distance thresholds between predicted and ground truth edge points. The distance threshold used to calculate soft precision and recall is set to one neighbour point distance in all directions.

Smaller values for the metrics $||A||_F$, $\mathbf{R}_{err}$, $\mathbf{T}_{err}$, and $\boldsymbol{Z}_E$ indicate better performance, demonstrating closer alignment to the ground truth pose. Whereas, larger values indicate higher accuracy for the F-measure, soft precision, and recall metrics. Appendix A presents the mathematical calculation for each of the metrics utilised for analysis of the proposed method. All metrics are calculated for each sample in the test set. It should be noted that the results presented for qualitative analysis are chosen based on the optimal performance of the proposed method, without considering the performance of the compared methods.

3.5.4 Comparison analysis: single object scenes

Tables 3.1, 3.2, and 3.3 summarises the pose registration analysis for the Tejani et al. [251], T-LESS [108], and PartNet [181] datasets, respectively. The effectiveness of the proposed method is further showcased in Figure3.4 for selected sample objects from all the datasets. The performance of JSENet is exclusively evaluated on PartNet dataset as it requires labelled data

for training. Table 3.4 presents the precision, recall, and F-measure score analysis of all the compared methods over the entire PartNet dataset.

Table 3.1: Analysis of 3D edge detection method with Tejani et al.[251] dataset for pose registration metrics.

M	Object	$\|\|A\|\|_F$	$\mathbf{R}_{err}$	$\mathbf{T}_{err}$	$\mathbf{Z}_E$
Choi et al.[46]	Joystick	**2.42**±0.64	2.13±0.74	0.66±0.26	0.11±0.05
	Juice	2.54±0.54	2.31±0.68	0.65±0.24	0.09±0.04
	Milk	2.27±0.73	2.01±0.84	0.58±0.26	0.10±0.06
	Camera	2.33±0.73	2.09±0.85	0.60±0.28	**0.05**±0.03
	Cup	2.30±0.63	**1.98**±0.72	0.59±0.24	**0.06**±0.03
	Avg.	2.37±0.65	2.10±0.76	0.62±0.26	0.08±0.04
Fast Edge[29]	Joystick	**2.42**±0.58	**2.08**±0.69	0.70±0.27	**0.10**±0.05
	Juice	2.48±0.64	2.25±0.77	0.67±0.26	0.09±**0.02**
	Milk	**2.24**±0.78	**1.99**±0.88	0.56±0.26	**0.09**±0.06
	Camera	2.58±0.56	2.39±0.70	0.68±0.26	0.06±0.03
	Cup	2.43±0.56	2.13±0.69	0.64±0.22	**0.06**±0.03
	Avg.	2.43±0.62	2.17±0.74	0.65±0.25	0.08±0.04
ung et al.[246]	Joystick	2.46±0.53	2.17±0.68	0.62±0.28	0.11±**0.03**
	Juice	**2.30**±0.56	**2.00**±0.70	**0.47**±0.23	**0.08**±**0.02**
	Milk	2.49±0.53	2.21±**0.67**	0.64±0.27	1.11±**0.03**
	Camera	**2.32**±0.59	**2.05**±0.74	**0.5**±0.23	**0.05**±**0.01**
	Cup	**2.29**±0.58	2.00±0.73	**0.45**±0.21	**0.06**±**0.02**
	Avg.	**2.33**±0.57	**2.03**±0.72	0.55±0.21	**0.06**±**0.02**
Proposed	Joystick	2.46±**0.47**	2.19±**0.64**	**0.53**±**0.20**	**0.10**±0.05
	Juice	2.44±**0.50**	2.18±**0.67**	0.53±**0.20**	**0.08**±0.04
	Milk	2.49±**0.52**	2.31± 0.71	**0.50**±**0.20**	0.11±0.05
	Camera	2.50±**0.48**	2.27±**0.66**	0.56±**0.17**	**0.05**±0.02
	Cup	2.47±**0.48**	2.24±**0.65**	0.47±**0.17**	**0.06**±**0.02**
	Avg.	2.47±**0.49**	2.23±**0.66**	**0.52**±**0.18**	0.07±0.03

Tejani et al. dataset The proposed method perform comparably to Choi et al. [46], Fast Edge [29], and Sung et al. [246], while achieving the best performance in translation error $\mathbf{T}_{err}$ for most of the objects. On average, a higher standard deviation is observed for the three compared methods, demonstrating that manually selected thresholds are sub-optimal for all the depth maps in the dataset and require fine-tuning. From Figure 3.4, it is observed that Choi et al. [46] method is unable to detect all the edges effectively. While Sung et al. [246] mistakenly identify surface points as edges for multiple objects. In contrast, the developed method detects

the object edges accurately, demonstrating its ability and efficiency in estimating 3D edges. For this test, the thresholds $\lambda_c = (0.05, 50)$, $\lambda_f = (0.05, 40)$, and $\lambda_s = (75, 75)$ were used.

Table 3.2: Analysis of 3D edge detection method with T-LESS [108] dataset for pose registration metrics.

M	Obj.ID	$\|\|A\|\|_F$	$\mathbf{R}_{err}$	$\mathbf{T}_{err}$	$\boldsymbol{Z}_E$
Choi et al.[46]	2	**2.53**±0.53	2.19±0.65	0.79±0.33	**0.02**±**0.01**
	5	**2.45**±0.77	**2.19**±0.87	**0.80**±0.37	**0.04**±0.02
	8	**2.43**±0.91	**2.28**±0.99	0.77±0.40	**0.08**±0.05
	17	**2.32**±0.81	2.03±0.89	0.74±0.40	0.05±0.03
	27	**2.47**±0.80	2.27±0.91	**0.80**±0.39	**0.08**±0.04
	Avg.	**2.44**±0.76	**2.19**±0.86	0.78±0.38	**0.05**±0.03
Fast Edge[29]	2	2.57±**0.47**	**2.18**±**0.61**	0.90±0.30	**0.02**±**0.01**
	5	2.55±**0.55**	**2.19**±**0.66**	0.87±**0.33**	**0.04**±**0.01**
	8	2.60±**0.79**	2.42±**0.86**	0.89±0.40	**0.08**±0.05
	17	2.46±**0.66**	2.12±0.76	0.84±**0.36**	0.05±0.02
	27	2.50±0.76	2.26±0.86	0.85±0.40	**0.08**±0.04
	Avg.	2.53±**0.65**	2.23±**0.75**	0.87±0.36	**0.05**±0.02
ung et al.[246]	2	**2.53**±2.72	3.14±3.0	**0.76**±2.74	0.03±**0.01**
	5	2.64±3.14	2.24±0.71	**0.80**±3.19	**0.04**±**0.01**
	8	2.52±1.65	3.14±3.0	**0.68**±1.62	**0.08**±**0.02**
	17	2.55±1.78	2.26±**0.70**	**0.69**±1.76	0.05±**0.01**
	27	2.65±3.68	**2.16**±**0.72**	0.86±3.74	**0.08**±**0.02**
	Avg.	2.58±2.59	2.59±1.63	**0.76**±2.61	0.06±**0.01**
Proposed	2	2.56±0.51	2.20±0.68	0.90±**0.28**	**0.02**±**0.01**
	5	2.51±0.63	2.20±0.75	0.84±**0.33**	**0.04**±0.02
	8	2.55±0.86	2.40±0.93	0.87±**0.38**	**0.08**±0.05
	17	2.33±0.80	**2.02**±0.88	0.80±0.37	**0.04**±0.02
	27	2.49±**0.72**	2.24±0.84	0.84±**0.36**	**0.08**±0.04
	Avg.	2.48±0.70	2.21±0.81	0.85±**0.34**	**0.05**±0.02

T-LESS dataset The proposed method demonstrate comparable pose registration performance to all the compared methods for all the objects. It achieved the best results with ADD $\boldsymbol{Z}_E$ metric. These results indicate that the developed method achieve satisfactory performance for noisy real-world data without any manual parameter tuning, as required by the compared methods. Figure 3.4 further verify this with the proposed method demonstrating substantially better edge detection and efficiently avoided detected edges at nearby noisy regions. Fast Edge [29] and Sung et al. [246] detects the object edges but also detects edges on the background noise, such as edges at the shadow boundary. While Choi et al. [46] is unable to detect edges on the lower

side of the objects. For this test, thresholds $\lambda_c = (0.02, 30)$, $\lambda_f = (0.02, 30)$, and $\lambda_s = (75, 75)$ were used. A zoom-in region is displayed at the top right corner of each image to highlight the important comparison details for T-LESS dataset.

Table 3.3: Analysis of 3D edge detection method with PartNet [181] dataset for pose registration metrics.

M	Object	$\|\|A\|\|_F$	$\mathbf{R}_{err}$	$\mathbf{T}_{err}$	$\mathbf{Z}_E$
Choi et al.[46]	Knife	3.06±1.14	2.15±0.97	1.84±1.27	2.49±0.56e1
	Scissors	2.75±1.19	2.04±1.09	1.54±1.13	7.01±**0.58e1**
	Bowl	3.05±0.81	2.16±**0.70**	1.74±1.06	9.78±**1.08e1**
	Bottle	2.99±1.07	2.09±0.81	1.69±1.29	**8.61**±**0.80e1**
	Mug	2.98±0.77	2.15±**0.70**	1.66±0.97	1.11e1±1.63e1
	Avg.	2.96±0.99	2.12±**0.70**	1.69±1.14	7.79±**0.93e1**
Fast Edge[29]	Knife	2.69±1.31	1.83±1.13	1.65±1.22	1.63±0.23e1
	Scissors	2.71±1.10	1.96±1.05	1.50±1.08	7.04±0.79e1
	Bowl	2.99±0.84	2.15±**0.70**	1.65±1.04	1.10e1±3.09e1
	Bottle	2.62±0.88	2.01±0.86	1.24±0.90	9.58±1.98e1
	Mug	2.87±0.80	2.09±0.75	1.55±0.93	1.35e1±2.81e1
	Avg.	2.77±0.98	2.00±0.89	1.52±0.70	8.55±1.78e1
JSENet[112]	Knife	1.47±1.36	1.54±1.50	0.14±0.24	0.75±0.17e1
	Scissors	1.74±1.27	1.77±1.41	0.24±0.32	6.27±0.95e1
	Bowl	1.89±0.86	1.64±0.92	0.25±0.2	8.31±1.42e1
	Bottle	2.03±0.87	1.85±0.99	0.24±0.22	1.24e1±2.68e1
	Mug	1.85±1.60	1.68±1.16	0.29±1.24	1.01e1±**1.14e1**
	Avg.	1.79±1.19	1.69±1.19	0.23±0.44	7.56±1.27e1
ung et al.[246]	Knife	2.23±**0.65**	2.02±**0.77**	0.40±0.17	5.3±2.7e1
	Scissors	2.31±**0.63**	2.14±**0.77**	0.4±**0.16**	6.2±2.4e1
	Bowl	2.29±**0.56**	2.08±0.71	0.38±**0.17**	9.4±2.7e1
	Bottle	2.26±**0.60**	2.03±**0.72**	0.39±**0.18**	**7.2**±2.6e1
	Mug	2.17±**0.68**	1.96±0.79	0.38±0.16	**8.0**±2.6e1
	Avg.	2.24±**0.63**	2.03±0.76	0.39±**0.17**	7.4±3.0e1
Proposed	Knife	**1.45**±1.37	**1.54**±1.50	**0.10**±**0.14**	**0.72**±**0.15e1**
	Scissors	**1.72**±1.22	**1.73**±1.38	**0.22**±0.24	**5.44**±0.62e1
	Bowl	**1.75**±1.00	**1.49**±0.95	**0.24**±0.55	**8.12**±1.38e1
	Bottle	**1.93**±0.96	**1.77**±1.06	**0.23**±0.26	1.09±3.76e1
	Mug	**1.53**±1.20	**1.46**±1.26	**0.16**±**0.14**	8.27±1.20e1
	Avg.	**1.67**±1.15	**1.59**±1.00	**0.19**±0.26	**6.69**±1.42e1

PartNet dataset The proposed method demonstrates outstanding performance against all the compared methods for mean error. It also achieve robust performance for translation and ADD errors with low standard deviation. As shown in Figure 3.4, Choi et al. [46] is not able to detect

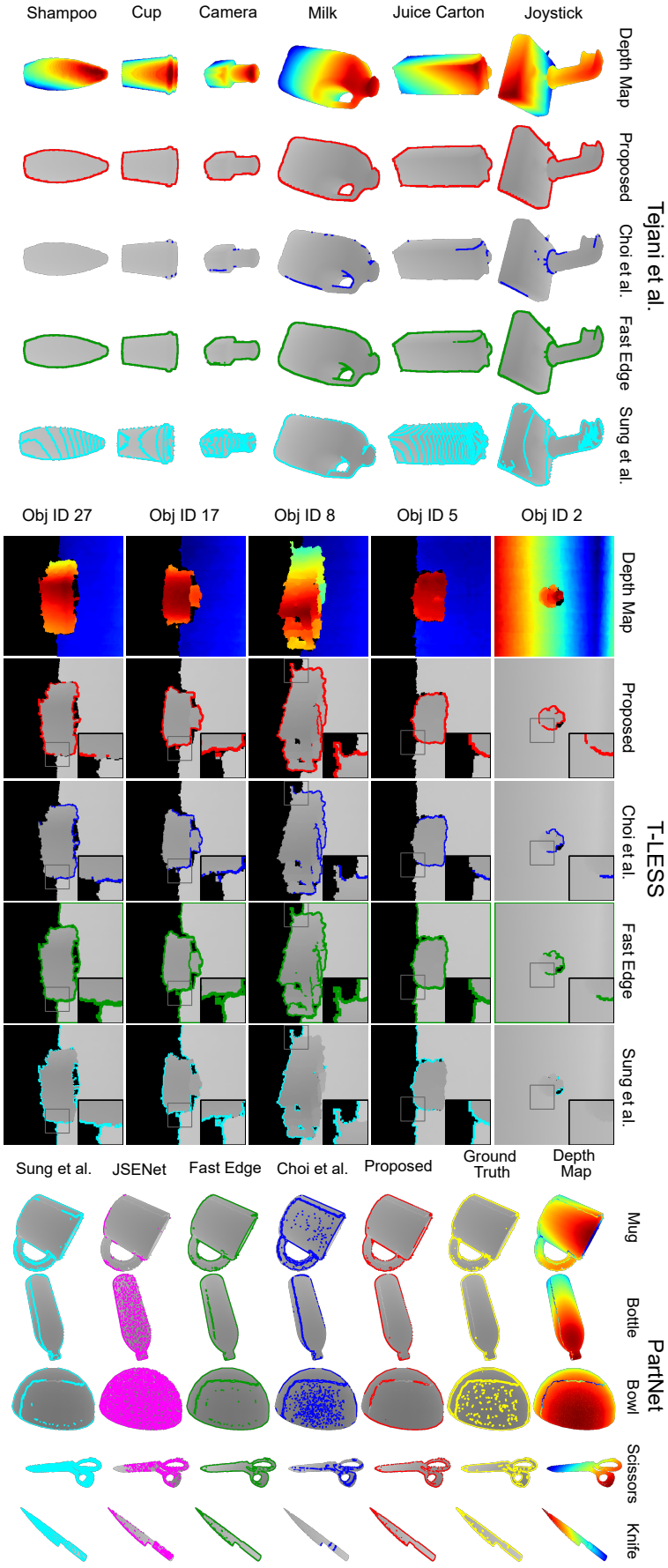

Figure 3.4: Single object scenes analysis of the 3D edge detection method with Tejani et al. [251], T-LESS[108], and PartNet [181] datasets.

Table 3.4: F-measure performance analysis with PartNet [181] dataset for 3D edge detection method.

Method	Choi et al.[46]	JSENet[112]	Fast Edge[29]	Sung et al.[246]	Proposed
Precision*	0.16	0.08	0.63	0.37	**0.82**
Recall*	0.19	0.20	**0.90**	0.84	0.84
F-measure*	0.14	0.10	0.72	0.48	**0.81**

* Larger values indicate better performance

all the edges. Further, non-edge point on the object surface are marked as edges by JSENet [112] and Sung et al. [246] methods. Fast Edge [29] exhibit similar performance to the proposed method and detect edges closely matching to the ground truth. This is further evidenced in Table 3.4, where high precision and F-measure accuracy is achieved by the proposed method while also showing comparable recall performance to the Fast Edge [29] method. The experiments were conducted using thresholds $\lambda_c = (0.02, 40)$, $\lambda_f = (0.02, 40)$, and $\lambda_s = (75, 75)$.

3.5.5 Comparison analysis: multi object scenes

Tables 3.5 and 3.6 presents the edge detection results for multi-object scenes from the T-LESS [108], MvTech-ITODD [67], and NYU [190] datasets. The detected 3D edges for all the compared methods (except JSENet [112]) are illustrated in Figure 3.5 and 3.6 for various scene from T-LESS and MvTech-ITODD datasets. The 3D edge detection performance for NYU [190] dataset are displayed in Figure 3.7 for all the methods. With the presence of multiple objects in a single scene, the pose registration analysis was performed by calculating the average performance of all the objects in a single scene. The edge estimation is performed for the whole scene and then individual object edges are segmented using the provided ground truth masks. These results clearly validate the suitability of the proposed method for (unstructured) multi-object scenarios. Experiments were conducted using $\lambda_c = (0.02, 40)$, $\lambda_f = (0.05, 40)$, and $\lambda_s = (75, 75)$.

T-LESS multi-object scenes The developed 3D edge detection method outperforms all the compared methods for all the pose registration metrics. Further from Figure 3.5, it is observed that Sung et al. [246] is affected by the shadow regions and only detects edges around them.

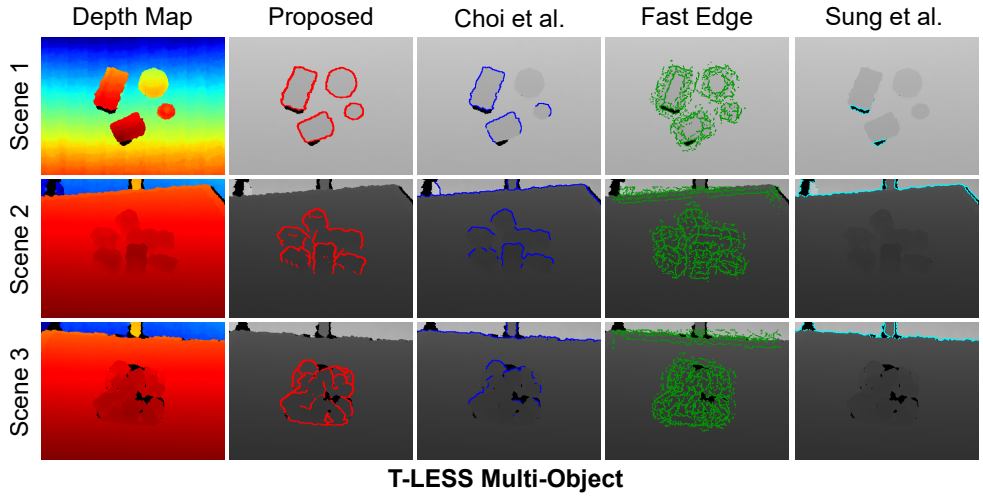

Figure 3.5: Multi-object scenes analysis with T-LESS[108] for 3D edge estimation method.

Choi et al. [46] only detects partial edges on the objects, while Fast Edge [29] show noise sensitivity and detects nearby noise as edges. In contrast, clear object 3D edges are detected by the developed method with robustness towards noise and shadow regions.

Table 3.5: Multi-object scenes performance analysis of 3D edge detection method with T-LESS[108] and MvTech-ITODD [67] datasets.

Method	T-Less[108]				MvTech ITODD[67]			
	$\|\|A\|\|_F$*	$\mathbf{R}_{err}$*	$\mathbf{T}_{err}$*	$\boldsymbol{Z}_E$*	$\|\|A\|\|_F$*	$\mathbf{R}_{err}$*	$\mathbf{T}_{err}$*	$\boldsymbol{Z}_E$*
Proposed	**2.21**	**1.85**	**0.82**	0.046	**2.45**	**2.10**	**0.69**	**0.044**
Choi et al.[46]	2.31	1.93	0.90	**0.044**	2.53	2.27	0.71	0.047
Fast Edge[29]	2.41	2.03	0.94	0.047	2.80	2.54	0.72	0.058
Sung et al.[246]	2.23	1.91	0.85	0.048	2.52	2.20	0.79	0.051

* Smaller values indicate better performance

MvTech-ITODD scenes Quantitatively, the developed method demonstrates outstanding performance across all metrics when compared to other methods. This superiority is evident in the numerical results, where our approach shows significant improvements, highlighting its effectiveness in various challenging scenarios. Qualitatively, as illustrated in Figure 3.6, the limitations of existing methods become clear. For instance, the method proposed by Choi et al. [46] fails to detect any edges in certain samples, which significantly compromises the reliability of edge detection in those cases. Similarly, the approach by Sung et al. [246] inaccurately detects edges within shadow regions. Fast Edge [29] struggles with noise in real-world data, often misidentifying surrounding noise as edges. In contrast, the proposed method excels in

edge detection and pose registration, delivering superior performance that is both noise-resilient and precise in delineating object boundaries.

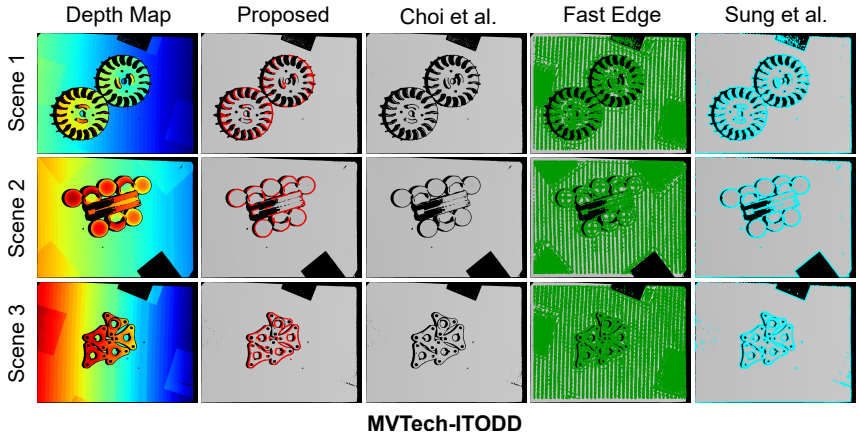

Figure 3.6: Multi-object scenes analysis with MVTech-ITODD [67] for 3D edge estimation method.

NYU scenes In NYU [190] dataset, ground truth pose information for the different objects in the scene is not provided. Hence, pose registration analysis is not performed for this dataset. However, F-measure based analysis is performed for all the methods. Table 3.6 summarises the performance of all the methods and it is observed that the proposed method outperforms all the other methods, indicating generalised learning for unseen samples with precise edge detection. The samples shown in Figure 3.7 further reflects the reliable performance of the proposed method. Choi et al. [46] and Fast Edge [29] are affected by the noise and mark flat regions also as edges. Sung et al.[246] detects few partial edges in the scene, while JSENet [112], despite being a supervised learning method, detects fragmented and incomplete edges. Overall, the proposed method outperforms all compared methods for multi-object scenes without requiring labelled data or parameter tuning.

Table 3.6: F-measure analysis using ground truth edges with NYU[190] dataset for 3D edge estimation method.

Method	Choi et al.[46]	JSENet[112]	Fast Edge[29]	Sung et al.[246]	Proposed
Precision*	0.31	0.48	0.33	0.61	**0.59**
Recall*	0.39	0.24	0.34	0.01	**0.44**
F-measure*	0.33	0.38	0.31	0.02	**0.49**

* Larger values indicate better performance

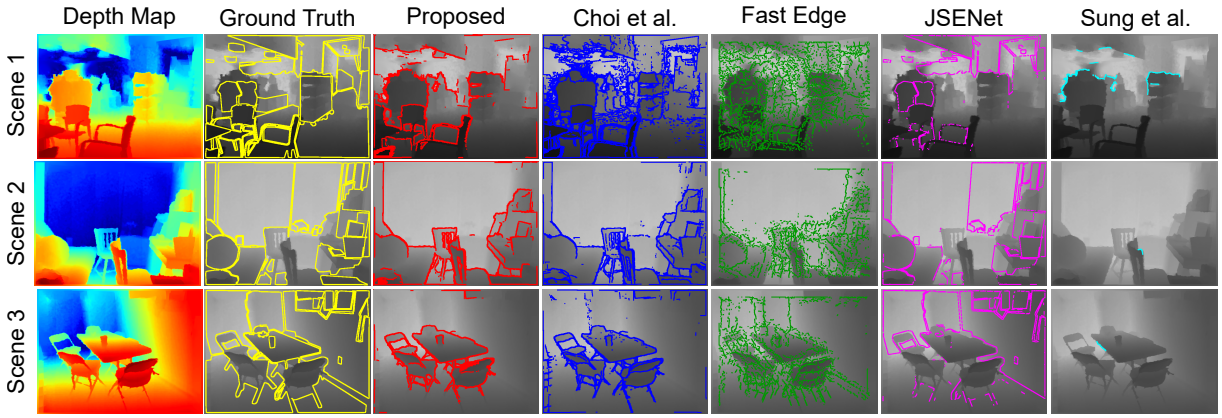

Figure 3.7: Multi-object scenes analysis with NYU[190] for 3D edge estimation method.

3.6 Contribution study

This section presents the effectiveness of different components of the proposed edge detection method through rigorous analysis and critical discussion. The network's computational complexity is also comprehensively evaluated to access its efficacy. Additionally, the practical applicability and robustness in diverse environments is highlighted via real-world performance of the method.

3.6.1 Generic network structure

In the proposed network architecture, the feature extraction layer is designed to be generic and can be modelled to identify different features. This generic property of the feature extraction layer is validated over a variety of edge feature extractors. Sobel, Roberts, Prewitt, and Laplacian of Gaussian (LoG) [13] are the four different gradient kernels considered in the proposed work as individual edge feature extractors and the performance is analysed. Figure 3.8 depicts the performance of the proposed method with these edge feature extractors. LoG filter is observed to be susceptible to noise and is unable to detect all edges. Overall, the best performance is demonstrated by the Sobel filter across all the considered datasets. In another test, a model was trained concatenating all the four kernel features for clustering. Through analysis, it is observed that there was no impact on the networks performance. It is worth noting that all the kernels extract gradients from the depth map and collectively provide similar types of features, hence

concatenating them had no impact on the prediction.

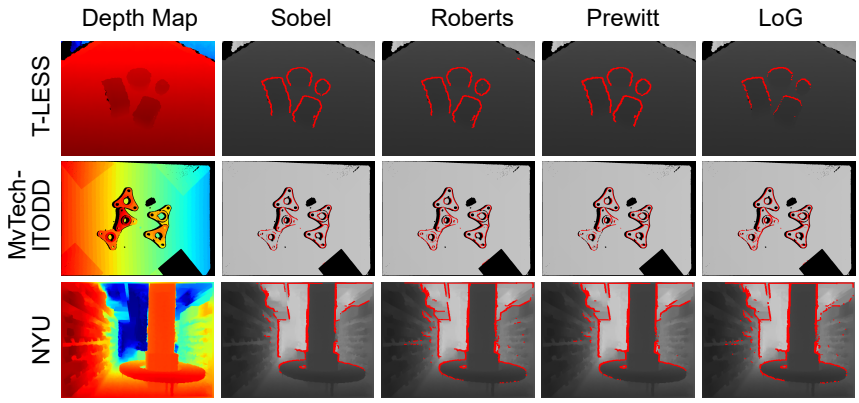

Figure 3.8: Qualitative analysis of the 3D edge detection method with different edge feature extractors.

3.6.2 Benefit of automatic parameter Selection

Automatic threshold selection is one of the key features of the developed method which supports towards achieving generality to the data. To validate this, the effect of threshold on the performance of Choi et al. [46] and Fast Edge [29] is evaluated for two different threshold values for each method respectively. $\lambda_{c,1} = (0.05, 50)$ and $\lambda_{c,2} = (0.02, 30)$ are used for Choi et al. and for Fast Edge, $\lambda_{f,1} = (0.05, 40)$ and $\lambda_{f,2} = (0.02, 30)$ are used. The threshold values for both the methods are selected based on the results reported in [46] and [29], respectively. From Figure

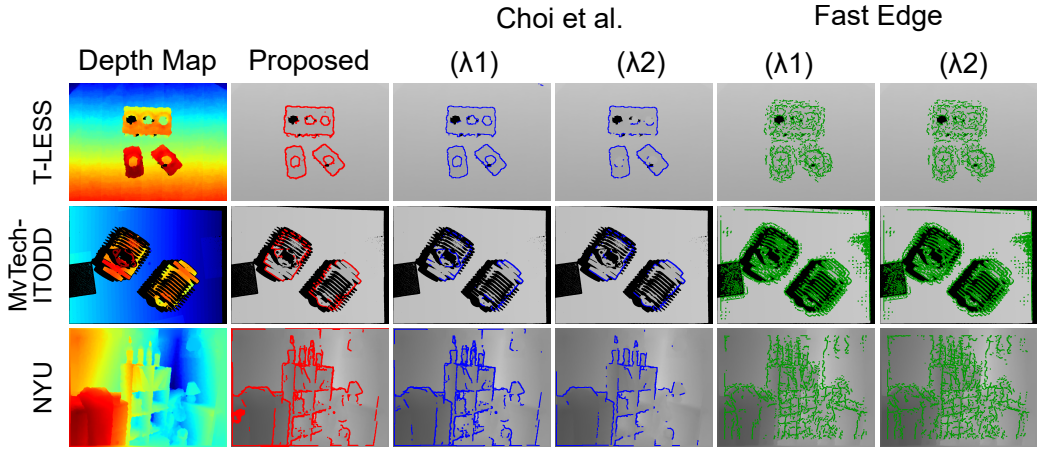

Figure 3.9: Affect of manual threshold selection on the performance of Choi et al. [46] and Fast Edge [29] as compared to the proposed 3D edge detection method.

3.9, it is observed that both the Choi et al. is significantly affected by small variations in the threshold values. Whereas, Fast Edge has no effect in its performance due to threshold change and still detects noise as edges in the scene. The proposed method, after learning the threshold

via unsupervised learning, showcased comparable performance to the best results of the SOTA methods.

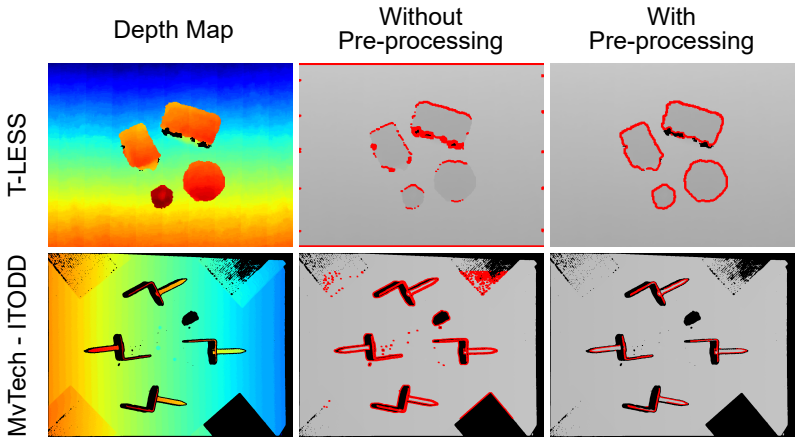

Figure 3.10: Affect of Pre-processing on the performance of the proposed 3D edge detection method.

3.6.3 Need for pre-processing

As previously discussed in Section 3.4.2, empty shadow regions in the depth maps impacts the edge extraction process. This particularly is observed when data is captured using a real depth sensor. The effect on the 3D edge detection performance due to shadows and effectiveness of the pre-processing approach are discussed here. From Figure 3.10, it is observed that when edges are detected without pre-processing, they are primarily centered along the shadow regions, due to the high depth variations at these points. However, actual object boundary edges are not detected. Conversely, with pre-processing, the proposed method detects correct edges on the object boundary rather than in the shadow regions.

3.6.4 Effects of edge thinning

A key operation in the edge feature extractor layer is edge thinning. In this, a minimum pooling kernel is applied to the edge features to thin down the extracted gradients around the edges and to remove unwanted noise. As shown in Figure 3.11, when a larger kernel size is utilised in edge thinning, the detected edges are incomplete or borken. While when smaller kernel size is considered, background noise affects the performance of the system. After multiple tests by experimentation, a kernel size of 2 was selected for all previously reported experiments.

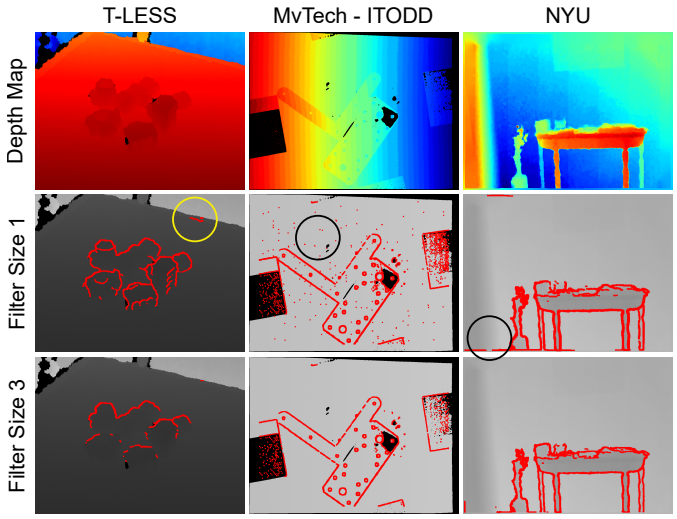

Figure 3.11: Impact of the kernel size in edge thinning layer on the 3D edge detection network performance.

3.6.5 Encoder-decoder requirement

An encoder-decoder-based feature extraction is performed in the proposed method, before extracting the edge features. It is utilised to learn intrinsic features from the depth maps, with an

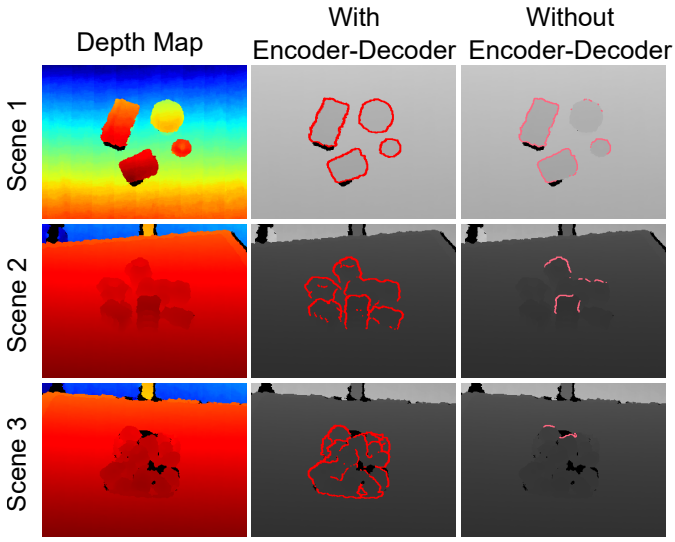

Figure 3.12: 3D edge detection performance analysis with and without encoder-decoder module evaluated on T-LESS [108] multi-object dataset.

aim to make the system robust towards input variations. To validate the utilisation of feature extractor, a separate network, without the encoder-decoder is trained for T-LESS [108] multi-object dataset. Analysis of the results, as shown in Figure 3.12, reveals that the method detects partial edges and have inconsistent performance when encoder-decoder is not utilised. This is

due to the variations in dataset which caused the method to learn inconsistent thresholds. This underscores the importance of incorporating the encoder-decoder in the proposed method for generic feature learning.

3.6.6 Analysing the effect of central masking

As discussed in Section 3.4.5, central masking is an optional layer utilised before clustering to filter out the background noise which may contain abrupt depth variations. The effect of central masking on the network performance is analysed by performing experiments with and without this layer. From the experiments, it is observed that for MvTech-ITODD [67] dataset, no variation in the proposed methods performance is observed, with or without central masking. This is due to filling out of all the noise in the depth map by the pre-processor. However, for the T-LESS [108] dataset, abrupt depth changes due to background noise were detected as edges in few instances, when central masking is not utilised. Nevertheless, removing central masking had only a minimal effect on the overall edge detection performance.

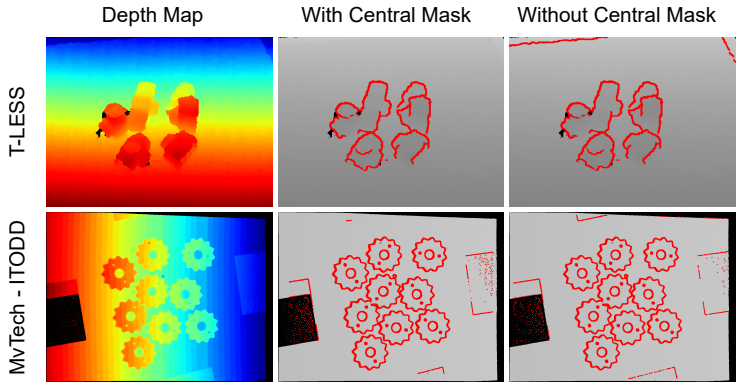

Figure 3.13: Effects on the 3D edge detection performance for central masking layer.

3.6.7 Generalisation to new data

The adaptability of the trained model to entirely new scenes is verified by analysing the performance of a trained model using the Stefan et al. [103] dataset. Two set of experiments were conducted for real-world multi-object scenes: one without fine-tuning ("Edge – NF") and the other with fine-tuning ("Edge – F"). 100 randomly selected depth maps are utilised for evaluation in each case. For "Edge – F", the model was fine-tuned for 20 epochs. The results for

four different scene are shown in Figure 3.14. It is observed that both "Edge – NF" and "Edge – F" performed comparably, with "Edge – F" demonstrating slightly better performance due to the additional fine-tuning. These results showcase the generalisation capability of the proposed method, to detect edges for a completely new dataset, without requiring any additional training or parameter tuning.

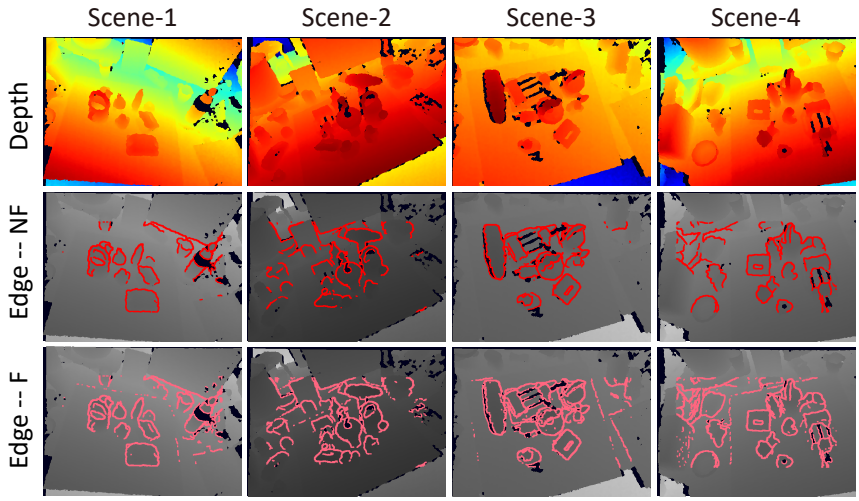

Figure 3.14: 3D edge detection performance with previously unseen Stefan et al. [103] dataset for four different scenes. "Edge – NF" represents the results when no fine-tuning is performed, while "Edge – F" are with fine-tuning of a trained model.

3.6.8 Why 2D edge detectors are not suitable?

A common question arises when depth maps are used for edge detection: why are traditional 2D edge detectors unsuitable for this task? To clarify this, tests were conducted using popular 2D edge detectors, specifically Canny [33] and Sobel [128], and their performance was compared with the proposed 3D edge detector. The results, as shown in Figure 3.15, clearly demonstrate that 2D detectors do not perform effectively on depth maps. The primary reason for this is that 2D edge detectors interpret pixel values in an image as intensity values. When applied to depth maps, they treat the depth values as intensities and thus detect edges where there are visible intensity discontinuities, rather than focusing on depth discontinuities. Since 2D edge detectors do not account for the 3D organisation of points, they tend to incorrectly identify points outside the object boundary as edge points. This behaviour is evident in the figure, where the Sobel 2D detector primarily detects edges around empty regions rather than along the actual object

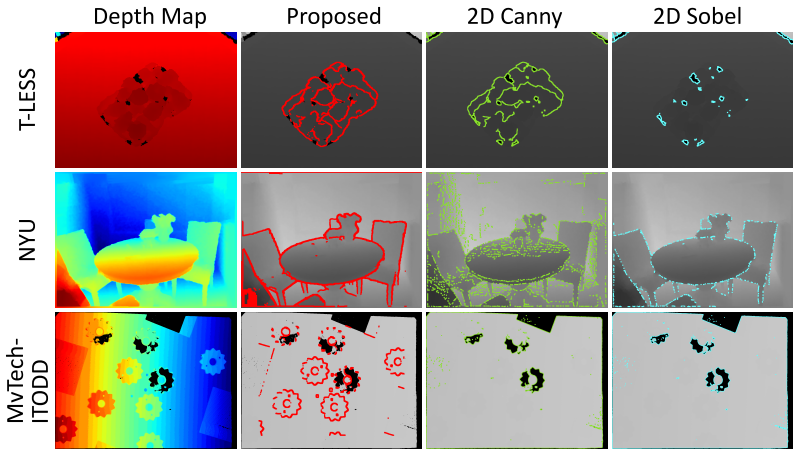

Figure 3.15: Performance comparison of the proposed 3D edge detection method with Canny [33] and Sobel [128] 2D edge detectors.

boundaries.

While the Canny 2D edge detector performed better than Sobel, the detected edges were often disconnected, and surface intensity variations were sometimes mistakenly identified as edges. In cases where surface variations were indistinguishable, such as with the MvTech-ITODD dataset, the 2D methods failed to detect object edges altogether.

This analysis clearly indicates that 2D methods do not consider the 3D information embedded in a depth map, resulting in broken edges along object boundaries or the detection of edges where depth information is absent. This underscores the necessity for a reliable 3D edge detector, such as the one proposed in this paper.

3.6.9 Complexity analysis

Conventional 3D edge detectors typically work by convolving a mask of size M to estimate the gradients in all directions. For 3D data of size $(X \times Y \times Z)$, convolving with a mask of size $(M \times M \times M)$ requires $O\left(M^3 * \text{size}(X) * \text{size}(Y) * \text{size}(Z)\right)$ operations [182]. If the data is dense, this complexity is even higher, which restricts the size of the mask to be used for different sizes of data and compromises the accuracy of edge estimation. In contrast, the computational complexity of the proposed inference model is calculated to be $O(HW + H + W)$, where W and H represent the width and height of the input depth map, respectively. The full derivation can be referenced in Appendix A. This calculation considers the total number of operations performed

in different layers, and since the complete model is not required at the time of inference, the complexity is calculated using only the layers necessary for inference, including the encoder layers, multi-size split layers, edge feature extractor layers, upscaling and merge, and clustering layer.

Table 3.7: Inference times for 3D edge detection methods.

Method	Time (ms)
Choi et al. [46]	100 – 120
Fast Edge [29]	60 – 80
JSENet [112]	80 – 100
Sung et al. [246]	70 – 90
Proposed	60 – 80

Table 3.7 compares the inference times of the proposed method with those of compared approaches. The results indicate that Choi et al. [46] and JSENet [112] exhibit relatively high inference times, rendering them less suitable for real-time applications. In contrast, Sung et al.[246] and Fast Edge[29] achieve real-time performance in edge detection. The proposed method demonstrates an inference time of approximately 60 – 80ms, aligning with the speed of Fast Edge and confirming its suitability for live imaging applications. To validate this, the method was used to detect edges for live acquisition of depth maps using an Intel RealSense 3D camera. Figure 3.16 shows screenshots captured for two different scenes during this process.

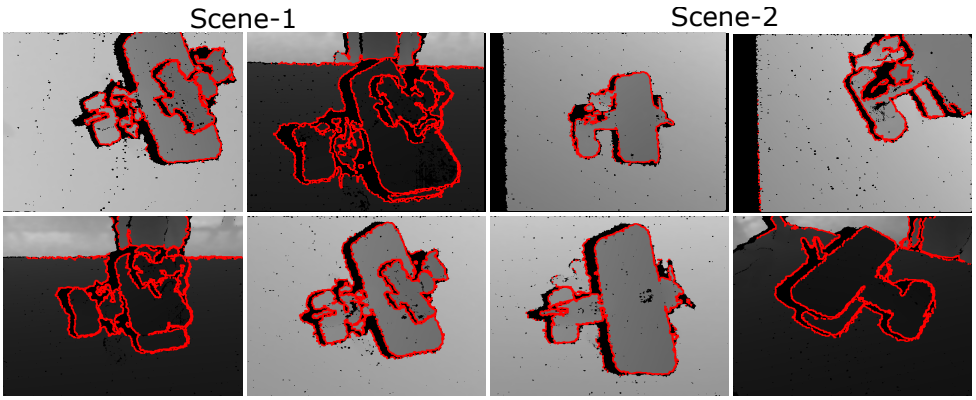

Figure 3.16: Screenshots representing the 3D edge detection from a live camera feed.

3.7 Summary

In this chapter, a method for 3D edge detection in depth maps is proposed. A deep learning-based network architecture is designed, consisting of three stages: i.e., encoder-decoder, edge feature extractor, and clustering. Initially, intrinsic features are learned from the input at three different scales by the encoder-decoder. The output from each of these scales is then passed to the edge feature extractor layers, which learn edge-specific features at these scales. The extracted edge features are subsequently combined and passed to the clustering layer, where each point in the depth map is clustered into either edge or non-edge classes. The overall network is trained in an unsupervised manner by learning probability distributions for both classes, thereby achieving independence from labelled data requirements.

It was observed that the input data from real-world sensors contained empty information patches, which affected the performance of the edge detector. To address this issue, a max-pool-based pre-processing algorithm is proposed to in-paint the missing depth information. The performance of the proposed method is evaluated on five benchmark datasets with both single and multi-object scenes. Pose registration and F-measure metrics are used for quantitative analysis, while visual results are provided for qualitative analysis. The proposed method is compared with four state-of-the-art methods from the literature, demonstrating competitive performance without the need for manual threshold fine-tuning or labelled data. A contribution study is also conducted, analysing the different aspects of the proposed method along with a computational complexity analysis.

The detected 3D edge features offer distinct geometrical characteristics of the objects, which can be utilised for numerous tasks. One of the key requirements in autonomous robots is the ability to understand the pose of objects in a scene. In the next chapter, a method for estimating the 6-DoF pose of an object is presented. The method is designed as an unsupervised approach that utilises detected 3D edges as input to predict the pose of an object in the scene.

Chapter 4

3D Edge-based object pose estimation

4.1 Introduction

Pose estimation is a critical component in understanding and interacting with the environment, playing a vital role in a variety of applications such as robotic grasping [6], visual servoing [4], augmented reality [166], autonomous driving [136], and object manipulation [118]. Essentially, pose estimation involves determining an object's spatial position and orientation, encompassing both its location and angular alignment in space. These parameters are organised into a transformation matrix, which mathematically represents the relationship between the object's own coordinate system and the world frame

The importance of pose estimation spans numerous industries and sectors. In manufacturing, for example, precise pose estimation allows automated systems to handle objects with exceptional accuracy, ensuring smooth and efficient assembly line operations. In healthcare, it facilitates the exact positioning of medical instruments or devices during surgeries or diagnostic procedures, thereby enhancing precision and safety. The energy sector also reaps benefits, particularly in the maintenance and operation of complex machinery, where accurate pose estimation is essential for ensuring operational integrity.

Accurately estimating the pose of an object provides a comprehensive understanding of its current state, which is crucial for making informed decisions about subsequent actions.

This capability enables precise repositioning necessary adjustments, or the execution complex manipulation tasks. Thus, pose estimation not only informs the current status of an object but also plays a vital role in planning and executing future operations, making it indispensable in modern technological and industrial applications.

Pose estimation, while highly advantageous, encounters substantial challenges, particularly in handling occlusions and varying lighting conditions. Ensuring accuracy in diverse, dynamic environments remains difficult. However, research is continually advancing, overcoming these challenges and establishing pose estimation as a pivotal tool in modern computer vision.

This chapter introduces a novel approach to object pose estimation using 3D edge-based techniques, enhancing accuracy and robustness in challenging environments. By utilising 3D edge features—highlighted in Section 3.1—the method effectively captures geometric contours, making it particularly useful for texture-less objects and less sensitive to lighting variations and occlusions. These 3D edges offer a more reliable information source in complex settings compared to other feature types like RGB-D HOG [56] and RGB-D SIFT [125], which are more susceptible to environmental factors.

The utilisation of 3D edges for 6-DoF pose estimation presents several significant advantages:

- The geometric nature of 3D edges enables more precise alignment of object models with real-world data, resulting in higher accuracy in pose determination.
- These edges provide strong, distinct features that can be matched with greater confidence across varying viewpoints, thereby reducing the ambiguity often encountered with other feature types.
- 3D edges facilitate the estimation process in environments with low texture where traditional methods may fail. This robustness enhances the reliability of pose estimation in real-world scenarios, making it a versatile tool for various domains.

Despite these advantages, using 3D edges for pose estimation presents inherent challenges. One of the primary challenges is the sparse nature of 3D edges requires sophisticated algorithms capable of effectively processing and interpreting this sparse information. Additionally, while

3D edges are generally robust, they can still be affected by noise and inaccuracies in depth sensing, which may lead to potential errors in pose estimation. Addressing these issues requires ongoing improvements in both hardware, such as more precise depth sensors, and in algorithmic techniques for edge detection and matching. Continued research in these areas is essential to fully harness the potential of 3D edges in 6-Degree of Freedom (6-DoF) pose estimation.

4.1.1 Related work

6-DoF pose estimation has been extensively studied in the literature, with methods generally classified into traditional and learning-based approaches [316]. In Section 2.3, a comprehensive survey on object pose estimation has been presented, covering various data modalities, including image-based approaches [144, 51, 131, 63], RGB-D methods [251, 265, 150], and point cloud-based techniques [227, 271, 145, 58].

In this section, the focus shifts to the specific domain of pose estimation using point cloud data, with an emphasis on 3D edge-based methods as explored in the literature.

Traditional methods

Several traditional methods have been proposed in the literature for performing pose estimation on point clouds. ICP [16] and its variants [227, 30] provided a robust framework for local pose refinement between two point clouds. These algorithms operate by iteratively refining the alignment of two point clouds, initially selecting the closest points between the two sets as correspondences, followed by estimating a transformation that minimises the distance between these corresponding points. This process is repeated until convergence is achieved, resulting in the optimal alignment of the point clouds. However, these methods require a good prior estimated pose for effective convergence.

To address this limitation, global pose estimation methods such as FGR [312], Super4PCS [177], and TEASER [291] have been proposed. FGR [312] optimises a dense robust objective based on direct feature matching and correspondence creation, eliminating the need for iterative sampling or local refinement, and does not require initialisation. Super4PCS [177] employs smart index-

ing and an efficient data structure to achieve optimal performance, enabling rapid and accurate alignment of point clouds without requiring prior pose information. TEASER decouples the estimation of scale, rotation, and translation, making it both efficient and accurate even with up to 99% outliers. This method leverages semidefinite programming relaxation to ensure global optimality and introduces TEASER++, a faster variant that uses graduated nonconvexity and Douglas-Rachford splitting [255].

Additionally, several other traditional methods have been proposed in the literature [104, 89, 292, 264, 207, 47], utilising features such as 3D edges [11], PPF [89], FPFH [222], and SHOT [259].

Deep learning methods

Recent advances in deep learning have significantly improved the accuracy and robustness of pose estimation. Over the past decade, numerous learning-based solutions have been proposed in the literature [131, 65, 204, 265, 15, 271, 145, 94, 58, 154, 307] for object pose estimation. In the context of point clouds, deep learning techniques have enabled transformative end-to-end learning frameworks that directly estimate poses from raw 3D data [15, 271, 145, 94, 58].

One notable method is DCP [271], which redefines the traditional ICP algorithm through a deep learning perspective. The model comprises three key components: first, input point clouds are mapped to rigid transformation (translation and rotation) invariant embeddings using a feature extraction network, such as PointNet [212] or DGCNN [272]. Next, a soft matching between the point clouds is predicted using an attention-based module, which helps in identifying correspondences. Finally, a differentiable Singular Value Decomposition (SVD) [242] layer is employed to compute the optimal rigid transformation, ensuring end-to-end trainability.

Another advanced method is IDAM [145], an iterative deep learning-based approach that integrates both geometric and distance features into the matching process. This integration helps resolve ambiguities and improves performance. IDAM introduces a learnable similarity matrix convolution module to compute similarity scores based on concatenated features of point pairs. Additionally, a two-stage point elimination technique is proposed to enhance computational

efficiency, with a novel mutual-supervision loss allowing for effective end-to-end training.

Furthermore, a learning-based method proposed in [94] employs keypoint matching to predict the pose of objects within a scene. This method combines object and scene features to predict keypoints and segment the scene cloud, with RANSAC used to estimate the final pose. To address the challenges of real-world noise and its variations between input clouds, Dang et al. introduced a noise-normalising layer called MatchNorm [58]. This layer is integrated within feature learning processes in DCP and IDAM based architectures to improve feature matching and pose estimation. Experimental results demonstrated that these approaches significantly enhance performance on real-world data.

3D edge-based methods

A subset of these methods specifically focus on the utilisation of 3D edges of objects for pose estimation [264, 207, 47, 154, 307]. A robust edge-based 3D object tracking approach was proposed by Wang et al. [264], where direction-based pose validation was utilised. The method was enhanced by incorporating both distance and direction cues, improving upon traditional distance-based pose estimation. Additionally, a particle filtering technique was integrated for non-local searching, with a failure recovery mechanism implemented using a 2D detection method. In [207], a novel approach was presented for detecting, localising, and reconstructing transparent objects using two calibrated views. A structured edge detector was trained to identify transparent edges, and the detected contours were utilised to hypothesise object shapes and poses.

A method for 3D texture-less object detection and tracking using an edge-based approach within a particle filtering framework on the SE(3) group was presented by Choi et al. [47]. Particles were initially aligned through chamfer matching with 2D edge templates, followed by an annealing process to avoid local optima. Edge correspondences between the projected model edges and image edges were refined using RANSAC, thereby enhancing tracking accuracy. Lui et al. proposed a Deep-Learning based Robust Edge Detection (DLED) method for detecting edges from RGB images to improve Point Pair Feature-based Pose Estimation with Multiple Edge Appearance Models (PPF-MEAM), particularly under varying lighting condi-

tions. Additionally, an Edge-Attention Network (EANet) was proposed by Zhang et al. [307] for 6-DoF pose estimation of texture-less objects. EANet employed a multitask learning strategy, combining edge reconstruction and pose estimation to enhance accuracy and robustness against varying lighting conditions. The network featured a shared-weight edge extractor, which boosted performance during pose refinement.

Overall, while traditional methods for object pose estimation using point clouds have established a solid foundation, they remain constrained by sensitivity to noise, outliers, and computational demands. Although edge-based approaches exhibit greater resilience in some aspects, further refinement is necessary to effectively address these limitations.

4.1.2 Organisation

The remainder of this chapter is structured as follows: In Section 4.2, novel contributions of the proposed method are described. A background on PointNet++ [213] point cloud feature extractor is discussed in Section 4.3. Section 4.4 provides an in-depth description of the proposed method, including a detailed discussion of the network architecture and the specific contributions made in this work. In Section 4.5, the datasets utilised for training and evaluation are introduced, followed by a comprehensive presentation of the experimental results. This section highlights the performance of the proposed method across various metrics and compares it with SOTA approaches. Finally, the chapter is concluded in Section 4.6, where the key findings are summarised, and the implications of the results are discussed.

4.2 Contributions

In this chapter, the 3D Edge-based Object Pose Estimation Network (3D-EOPEN) method is introduced for object pose estimation by aligning the object model point cloud with the 3D edge points extracted from a scene's point cloud. The process begins with the segmentation of the scene point cloud using the DFormer segmentation method [298] to identify an object within the scene. This object id is selected before the process starts. This step isolates the object of

interest from its surrounding environment, enabling focused analysis. Once segmented, the 3D edge point cloud is detected using the 3D edge detector developed in the previous chapter [11], which extracts the significant geometric contours of the object. These edges are critical as they provide a sparse yet informative representation of the object's structure, making the subsequent pose estimation process more efficient and robust.

Simultaneously, the object model point cloud is obtained through two possible approaches: sampling a CAD model or performing surface registration [172]. The extracted 3D edge and model point clouds are then fed into the proposed deep learning-based network to estimate the 6-DoF pose of the object. This network leverages a modified PointNet++ [213] architecture, which is specifically adapted to handle sparse edge data. The architecture extracts features at three different levels of abstraction, capturing both local and global geometric information.

Feature correspondences are subsequently established between the edge and model features, identifying keypoints with the highest matching scores. These correspondences are crucial as they anchor the relative positioning between the observed data and the reference model. The keypoint features are then used to estimate the object's rotation and translation through a fully connected Multi-Layer Perceptron (MLP) regression network. This final step generates the precise pose of the object, allowing for accurate alignment within the scene, which is essential for tasks such as robotic manipulation, augmented reality, and more.

The core contributions of this work include:

- A novel self-supervised learning-based method is introduced, specifically designed for the robust alignment of 3D edge and model point clouds for 6-DoF pose estimation. This method leverages the inherent structure of 3D edges to guide the alignment process, making it particularly effective in scenarios where full point cloud based approaches may struggle due to noise or sparsity. The self-supervised nature of this method reduces the need for extensive labelled datasets, allowing the model to learn effective representations and alignments directly from the input data.

- The architecture is based on a modified version of PointNet++ [213], tailored to extract meaningful features from sparse 3D edge point clouds. This modification allows the

network to handle the unique challenges posed by sparse data, such as limited information and higher susceptibility to noise, while still maintaining high performance in feature extraction. By capturing multi-scale local features, the architecture ensures that both fine and coarse details of the object edges are preserved, facilitating more accurate pose estimation.

- An online data generation pipeline is developed to support the self-supervised learning process. This pipeline dynamically generates training samples that incorporate various real-world challenges such as noise, occlusions, and variations in rotation and translation. By simulating these conditions during training, the pipeline ensures that the model is exposed to a wide range of scenarios, enhancing its ability to generalise to unseen data. This approach not only improves robustness but also significantly reduces the risk of overfitting to specific datasets.

- A multi-point variant of the triplet loss [114] is employed to improve the learning of matching correspondences between 3D edge point clouds and their corresponding model features. This variant extends the traditional triplet loss by considering multiple points simultaneously, which enhances the network's ability to discriminate between correct and incorrect matches. This approach encourages the model to focus on the most informative parts of the edge clouds, leading to more reliable and accurate pose estimations.

In contrast to conventional approaches that require full point clouds or dense depth images, the proposed method operates with a remarkably sparse set of points, less than 5% of the total, making it highly efficient and less dependent on data density. This method exclusively utilises spatial information from the 3D edge cloud, thereby eliminating the need for rich textures or dense point clouds, which are often required in traditional methods.

The performance of this method is rigorously evaluated using the LineMOD [103], Occluded-LineMOD [31], and TUD-L [107] benchmark datasets, which are well-known for their challenging multi-object, real-world scene data. Various performance metrics are employed to assess the effectiveness of the proposed approach, and these results are directly compared with SOTA

methods in the literature. Experimental outcomes reveal that the proposed method show competitive performance to existing SOTA methods but does so while using significantly less data, highlighting its efficiency and robustness in practical applications. This advancement showcases the method's potential in environments where data sparsity is a constraint, yet high accuracy in pose estimation is critical.

The method proposed in this chapter is under submission to a renowned peer reviewed journal in the community.

4.3 Background on PointNet++ feature extraction

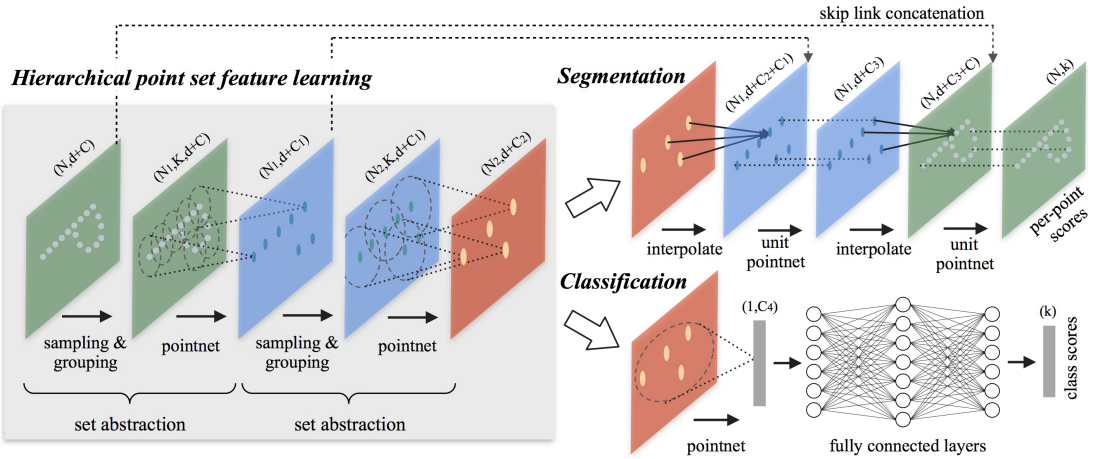

Figure 4.1: Deep learning-based network architecture for PointNet++ [213]. Image taken directly from published work.

PointNet++ [213] is an extension of the original PointNet [212] architecture designed to address the challenges associated with varying point densities in 3D point clouds. Unlike PointNet, which processes each point independently, PointNet++ introduces a hierarchical learning approach that effectively captures both local and global features as shown in Figure 4.1. The network is structured around set abstraction layers that progressively extract local features by grouping neighbouring points and applying PointNet-like operations within these groups. This hierarchical approach allows the network to capture local geometric details such as edges and corners, while also maintaining an understanding of the global structure of the point cloud.

The network employs FPS and ball query to group points, ensuring that local regions

are uniformly sampled and effectively processed. Within each group, features are extracted using PointNet, and then aggregated through max-pooling, which is invariant to the order of points. This makes the model robust to the inherent unordered nature of point clouds. Additionally, PointNet++ offers variants like Multi-Scale Grouping (MSG) and Multi-Resolution Grouping (MRG). MSG captures features at multiple scales by using different radius values during grouping, while MRG maintains features at different resolutions, which is particularly beneficial when dealing with sparse data. The extracted features are then propagated back to the original point set through feature propagation layers, ensuring that the final output is enriched with both local and global context. The network's final layers typically perform classification or segmentation tasks, depending on the specific application.

PointNet++ is particularly effective for sparse point clouds due to its hierarchical structure, which captures fine-grained local features and broader global structures. Its robustness to noise and outliers, achieved through max-pooling and FPS, makes it suitable for real-world 3D data. The network's invariance to point order and density, coupled with its ability to process data at multiple resolutions, ensures that it handles varying point densities effectively. This versatility has made PointNet++ widely used in various 3D computer vision tasks, including object classification, part segmentation, and scene segmentation, demonstrating its significant impact on deep learning for 3D data.

4.4 Methodology

In this section, the proposed deep learning-based pose estimation method is detailed. The object edge point cloud $\mathbf{E} \in \mathbb{R}^{V\times 3}$ with V points, extracted from the scene, and the corresponding object model point cloud $\mathbf{M} \in \mathbb{R}^{U\times 3}$ with U points are provided as inputs to the network, with the goal of estimating a homogeneous transformation $\mathbf{P} \in \mathbb{R}^{4\times 4}$ that aligns the model to the edge cloud. A three-stage network architecture is proposed, as illustrated in Figure 4.2, which comprises feature extraction, correspondence matching, and pose prediction stages.

For the feature extraction stage, a modified PointNet++ [213] architecture is employed. Point-

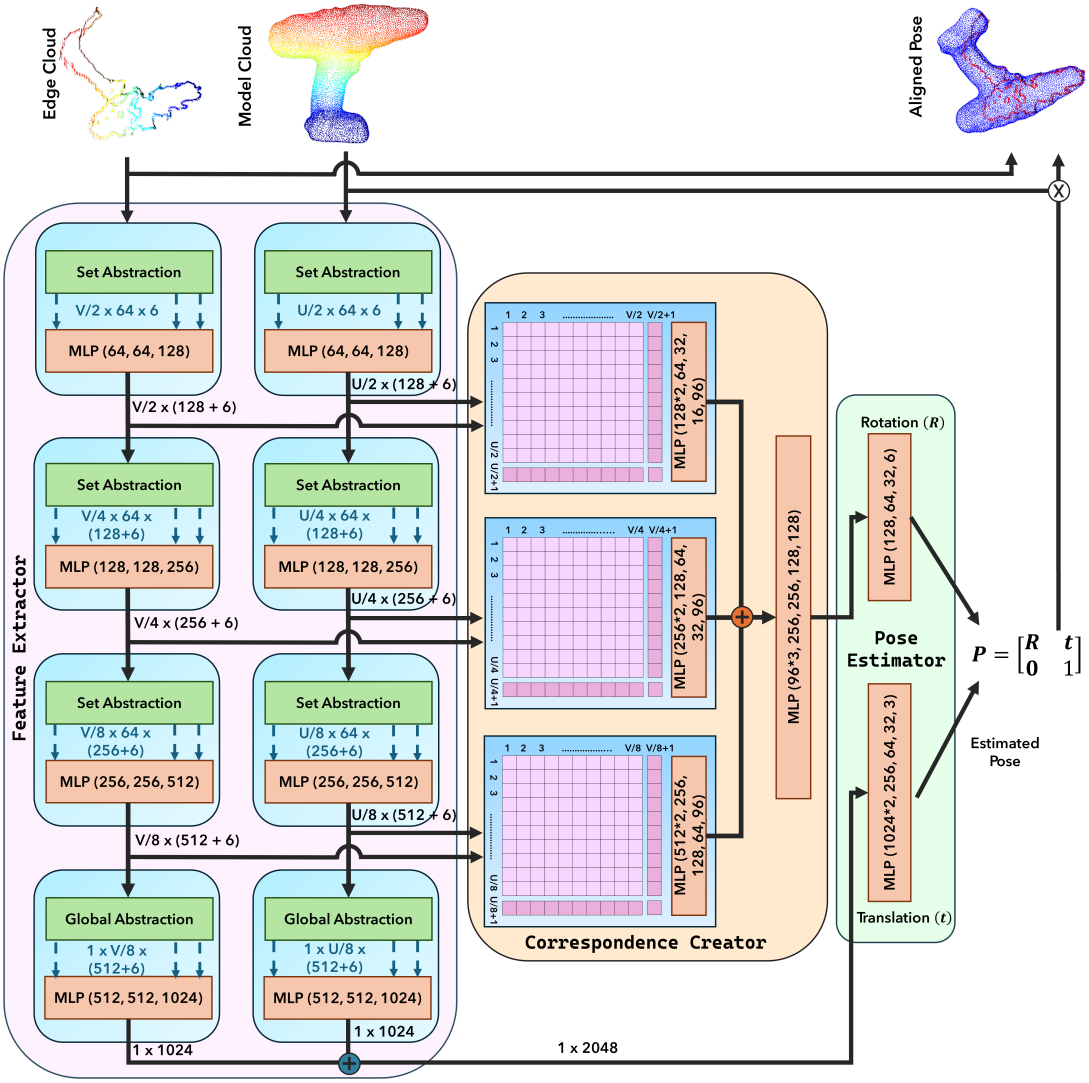

Figure 4.2: The proposed deep learning-based network architecture of edge-based object pose estimation method.

Net++ [213] offers significant advantages over other point cloud feature extractors by employing a hierarchical structure that captures both local and global geometric features. This approach enhances its ability to handle complex, non-uniform point clouds and improves performance in tasks involving fine-grained details, making it particularly suitable for texture-less objects and unstructured environments as highlighted in the thesis. This architecture is designed to learn features $\mathbf{E}_f^i$ and $\mathbf{M}_f^i$ (where i represents the layer number in the feature extractor) from $\mathbf{E}$ and $\mathbf{M}$, respectively through mutually exclusive layers. Following feature extraction, correspondences are established between the learned features $\mathbf{E}_f^i$ and $\mathbf{M}_f^i$ at all the abstraction layers to identify k_i matching keypoints between the two inputs. These keypoint features are then aggregated into

a 128-dimensional feature vector $\boldsymbol{f}$, which is subsequently passed to the pose predictor.

In the final stage, the feature vectors are processed through two MLPs to predict the rotation $\mathbf{R}_p$ and translation $\mathbf{t}_p$, which together form the transformation matrix $\mathbf{P}_p$. Each of these modules is discussed in detail in the following subsections.

4.4.1 Problem formulation

The primary challenge addressed in this study is determining the optimal transformation to align two point clouds, the edge cloud $\mathbf{E}$ and the model cloud $\mathbf{M}$. Building on the definition of object pose as discussed in Section 2.3, where the edge cloud is defined as $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_V\}$ and the model cloud as $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_U\}$, with $\mathbf{e}_j, \mathbf{m}_j \in \mathbb{R}^3$, the goal is to determine the transformation matrix $\mathbf{P}$, which comprises a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$, that best aligns $\mathbf{M}$ to $\mathbf{E}$. This pose estimation problem is mathematically formulated as a distance minimisation task, where the goal is to find the optimal $\mathbf{R}$ and $\mathbf{t}$ that minimise the sum of the minimum squared distances between each point $\mathbf{m}_j$ in the model cloud and its closest point $\mathbf{e}_k$ in the edge cloud, as shown below:

$$\min_{\mathbf{R}\in SO(3),\mathbf{t}\in\mathbb{R}^3} \sum_{j=1}^{U} \min_{k=1}^{V} \|\mathbf{R}\mathbf{m}_j + \mathbf{t} - \mathbf{e}_k\|^2 \tag{4.1}$$

In this case $\mathbf{R}$ is a valid rotation matrix, i.e., it is orthonormal with a determinant of 1, and $\mathbf{t} \in \mathbb{R}^3$ represents the translation vector. The transformation matrix $\mathbf{P}$ is thus defined as:

$$\mathbf{P} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{4.2}$$

The objective of this optimisation is to minimise the mean squared distance between corresponding points in the transformed model cloud and the edge cloud. The learning process is driven by this optimisation, where the deep neural network is trained to predict the transformation parameters that achieve this minimisation. In the following sections, each module of the network is described.

4.4.2 Feature extractor

Feature extraction is a critical component in enhancing the reliability of deep learning applications, as discussed in Section 2.1. Due to the sparse nature of 3D edge point clouds, extracting useful information from them is challenging. To tackle this, a deep learning-based network architecture is designed, inspired from PointNet++ [213] to extract fine detailed features from the input point clouds. The presented architecture employs two feature extractors to derive features $\mathbf{E}_F$ and $\mathbf{M}_F$ from edge cloud $\mathbf{E}$ and model cloud $\mathbf{M}$, respectively. Each extractor contains four layers ($i \in 1, 2, 3, 4$), with each layer consisting of set abstraction and learning mechanisms. Several enhancements over the standard set abstraction method have been introduced to optimise feature extraction from edge point clouds $\mathbf{E}$:

- Original methods utilise FPS [71] to select anchor points, which often fail in sparse, noisy conditions such as those found in edge clouds. The proposed approach randomly selects q anchor points, increasing the inclusion probability of critical edge points.

- Traditionally, features for each anchor point are calculated using a fixed-radius query, which performs poorly in edge areas due to unreliable proximity data. A random selection from the entire edge cloud is proposed to compute features, thereby increasing the extraction reliability.

- Typically, only the relative coordinates (x, y, z) between anchor points and nearby points are utilised for feature computation. In contrast, the developed method also integrates the differences in point norms and other attributes, to enrich the feature set for subsequent layers.

For each layer $i \in 1, 2, 3$, the number of anchor points u_i and v_i from $\mathbf{E}$ and $\mathbf{M}$ is set to 0.5× the number of points from the previous layer. For instance, in layer 1, $u_1 = 0.5 \times U$ and $v_1 = 0.5 \times V$. A random selection of 64 points is made to learn features for each anchor point. In the final layer, global features are derived by treating a zero vector as the anchor point and considering all remaining points. These features from the first three layers are utilised for

correspondence matching, and the global features from the last layer are directly applied to pose estimation.

4.4.3 Correspondence creator

The scene captured using depth sensors typically includes only a partial view of the object. Extracting the edge cloud from this partial view results in a sparse representation of the complete model cloud. To address this challenge, correspondences between the points in the edge cloud and the points in the model cloud is established using features learned across all hierarchical layers. Various methods for correspondence matching have been explored in the literature, often based on feature similarity and matching metrics [70]. The proposed approach, employ direct feature matching to construct a matching matrix $\mathbf{C}_i \in \mathbb{R}^{e_i \times m_i}$ for each layer $i \in \{1, 2, 3\}$. The value at each index j, k in $\mathbf{C}_i$ is obtained by performing a dot product between the features of edge point j and features of model point k at the ith layer. Given that some points in the edge cloud may not correspond to any points in the model cloud and vice versa, the matching matrix is expanded by introducing an additional dimension along both axes. This extension accommodates unmatched points by assigning them to this extra dimension, resulting in an initial score matrix $\mathbf{S}_i^{\text{init}} \in \mathbb{R}^{(e_i+1)\times(m_i+1)}$. The scores for these added dimensions are learned during training to facilitate reliable correspondence matching.

After computing the score matrix, a softmax [86] function is applied along each axes to estimate the probabilities of correspondence from edge points to model points and vice versa, resulting in $\mathbf{S}_i^{e2m}$ and $\mathbf{S}_i^{m2e}$ respectively. These probability matrices are then averaged to obtain the final score matrix $\mathbf{S}_i$. Considering the noise inherent in the edge cloud, not all points may establish correct correspondences. To mitigate this effect, k_i keypoints are extracted from the edge cloud, selecting points with the highest matching probabilities. For each keypoint, the best corresponding point in the model cloud is determined. The features from these keypoints in both the edge and model clouds are then aggregated and processed through a series of feature extractor layers, culminating in the extraction of a 96-dimensional feature vector.

The proposed network learns to establish these correspondences during the training process.

To facilitate this, a multi-point triplet loss is computed on the correspondence matrix at all layers. This loss function is designed to help the network learn similar features for corresponding points and dissimilar features for non-corresponding points. Most studies in the literature assume a one-to-one correspondence between points in different sets [145, 70, 285], and their methods learn mappings accordingly. However, due to the sparse nature of the edge cloud, our method considers that a single point in the edge cloud may correspond to multiple points in the model cloud, resulting in a many-to-many correspondence relationship, which adds complexity to the problem. To address this, we introduce a multi-point triplet loss that assumes each point in the edge cloud can correspond to multiple points in the model cloud, and vice versa.

To establish these correspondences, the model cloud is first transformed using the ground truth pose to align it with the edge cloud. Then, for each point in the edge cloud, corresponding points in the model cloud are identified as positive samples if they fall within a threshold distance α from the edge point. These points are then considered as positive sample indices. Let p denote the maximum number of nearby points selected. If no point in the model cloud falls within the threshold distance, a dummy point is assigned as the positive sample index for the edge point. All other points in the model cloud are considered as negative sample indices for that edge point.

Using these indices, the respective score values from the score matrix $\mathbf{S}_i$ are extracted to form the set of score values for the points considered as positive samples $\mathbf{s}_i^p$ and the negative samples $\mathbf{s}_i^n$.For the dummy point, the score associated with the last added dimension is used. The loss is calculated as follows:

$$L_{e2m}(\mathbf{s}_i^p, \mathbf{s}_i^n) = \max\left(0, \sum \mathbf{s}_i^p - \sum \mathbf{s}_i^n + \beta\right), \tag{4.3}$$

where β is a margin hyperparameter that defines the minimum required separation between the positive and negative samples. The same loss calculation is performed for the model-to-edge cloud correspondences to obtain L_{m2e}. The final correspondence loss is then computed as $L^{corr} = (L_{e2m} + L_{m2e})/2$. The overall algorithm is detailed in Algorithm 2.

Algorithm 2 Multi-point triplet loss

Input: $\mathbf{M}$, $\mathbf{E}$, $\mathbf{R}_g$, $\mathbf{t}_g$, k, α, $\mathbf{C}$, β
Output: Loss
$\mathbf{M}_g \leftarrow \mathbf{R}_g\mathbf{M}^T + \mathbf{t}_g$
$\mathbf{D} \leftarrow \sqrt{\sum(\mathbf{S} - \mathbf{M}_g)^2}$
$\mathbf{topK}_s \leftarrow \text{TopK}(\mathbf{D} < \alpha, k, \text{axis} = 2)$
$\mathbf{topK}_m \leftarrow \text{TopK}(\mathbf{D} < \alpha, k, \text{axis} = 1)$
$\mathbf{C}_s \leftarrow \log(\text{softmax}(\mathbf{C}, \text{axis} = 2))$
$\mathbf{C}_m \leftarrow \log(\text{softmax}(\mathbf{C}, \text{axis} = 1))$
$\mathbf{Pos}_s \leftarrow \mathbf{topK}_s \neq V$
$\mathbf{Pos}_m \leftarrow \mathbf{topK}_m \neq U$
$\mathbf{Neg}_s \leftarrow \mathbf{Pos}_s == False$
$\mathbf{Neg}_m \leftarrow \mathbf{Pos}_m == False$
$\mathbf{loss}_s \leftarrow \text{mean}(\max(0, \mathbf{C}_s[\mathbf{Pos}_s] - \mathbf{C}_s[\mathbf{Neg}_s] + \beta))$
$\mathbf{loss}_m \leftarrow \text{mean}(\max(0, \mathbf{C}_m[\mathbf{Pos}_m] - \mathbf{C}_m[\mathbf{Neg}_s] + \beta))$
$Loss \leftarrow \frac{1}{2}(\mathbf{loss}_s + \mathbf{loss}_m)$

4.4.4 Pose estimator

The pose estimator module leverages the accumulated features from all hierarchical layers, extracted based on the correspondences between the edge cloud and model cloud, to predict the pose transformation. It is divided into two branches: one dedicated to estimating the rotation and the other to estimating the translation components. Together, these branches form the predicted pose $\mathbf{P}$. The rotation branch consists of a MLP equipped with batch normalisation [116] and ReLU activation [9]. The rotation is represented as a 6D vector $\mathbf{r} \in \mathbb{R}^{1\times6}$, which corresponds to the vector representation of the rotation matrix $\mathbf{R}_p \in \mathbb{R}^{3\times3}$, as presented by Zhou et al. [314]. This representation is derived as follows:

$$
\begin{aligned}
\mathbf{v}_1 &= \mathbf{r}_p[:3], \quad \mathbf{v}_2 = \mathbf{r}_p[3:] \\
\mathbf{e}_1 &= \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} \\
\mathbf{e}_2 &= \mathbf{v}_2 - (\mathbf{e}_1 \cdot \mathbf{v}_2)\mathbf{e}_1 \\
\mathbf{e}_2 &= \frac{\mathbf{e}_2}{\|\mathbf{e}_2\|} \\
\mathbf{R}_p &= [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \mathbf{e}_1 \times \mathbf{e}_2]
\end{aligned}
\tag{4.4}
$$

Rotation is estimated solely using the feature vector $\mathbf{c}_f$, which is obtained through corre-

spondence. The translation branch, also an MLP, processes the global features $\mathbf{E}_f^4$ from the edge cloud and $\mathbf{M}_f^4$ from the model cloud, along with the correspondence feature $\mathbf{c}_f$. These features are concatenated and fed into the translation branch to estimate the translation vector $\mathbf{t}_p$ in the x, y, and z directions. Given that both the rotation and translation vectors are real-valued, no activation function is used in the final layers. The predicted pose matrix $\mathbf{P}_p$ is then formulated as:

$$\mathbf{P}_p = \begin{bmatrix} \mathbf{R}_p & \mathbf{t}_p^T \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{4.5}$$

This approach ensures that the pose estimator accurately determines the orientation and position of objects. By effectively processing feature vectors for rotation and translation, the estimator handles complex variations in object poses, enhancing system reliability.

4.4.5 Pre and post processing

In real-world scenarios, objects can vary widely in scale and may be positioned at any distance within a 3D space, presenting challenges for pose estimation tasks. To ensure the network remains robust to these variations and provides reliable performance, specific pre-processing steps are applied before feeding data into the network, and post-processing steps are carried out after the final transformation is obtained.

During the pre-processing phase, a set of transformation, known as standardisation transformations, are applied to both the edge and model clouds, to bring the difference between them to a common scale, before they are passed to the network. Initially, both clouds are translated to the frame origin by subtracting their respective centroid coordinates, denoted as $\mathbf{b}_e$ and $\mathbf{b}_m$, as follows:

$$\begin{aligned} \mathbf{E}' &= \mathbf{E} - \mathbf{b}_e, \\ \mathbf{M}' &= \mathbf{M} - \mathbf{b}_m. \end{aligned} \tag{4.6}$$

This translation normalisation makes the method robust to variations in object position within the scene. To address variations in object size, a scaling operation is performed on both clouds

by dividing each by the object's radius r_m, ensuring consistent scale:

$$\begin{aligned} \mathbf{E}'' &= \mathbf{E}'/r_m, \\ \mathbf{M}'' &= \mathbf{M}'/r_m. \end{aligned} \tag{4.7}$$

The standardised clouds $\mathbf{E}''$ and $\mathbf{M}''$ are then fed to the network for pose estimation. In the post-processing phase, the predicted pose $\mathbf{P}_p$ is transformed to derive the final transformation $\mathbf{P}$:

$$\mathbf{P}' = \mathbf{S}_m^* \mathbf{P}_p \mathbf{S}_m, \qquad \text{and} \qquad \mathbf{P} = \mathbf{B}_e \mathbf{P}' \mathbf{B}_m, \tag{4.8}$$

where,

$$\mathbf{S}_m = \begin{bmatrix} 1/r_m & 0 & 0 & 0 \\ 0 & 1/r_m & 0 & 0 \\ 0 & 0 & 1/r_m & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{S}_m^* = \text{inv}(\mathbf{S}_m),$$

$$\mathbf{B}_e = \begin{bmatrix} 1 & 0 & 0 & b_e^x \\ 0 & 1 & 0 & b_e^y \\ 0 & 0 & 1 & b_e^z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{B}_m = \begin{bmatrix} 1 & 0 & 0 & -b_m^x \\ 0 & 1 & 0 & -b_m^y \\ 0 & 0 & 1 & -b_m^z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here, $\mathbf{S}_m^*$ represents a scale-up operation to revert the predicted pose to the original scale of the object. $\mathbf{B}_e$ and $\mathbf{B}_m$ are translation transformations that adjust the predicted pose based on the original positions of the edge cloud and model cloud centroids, respectively. These pre-processing and post-processing steps are essential for enabling the pose estimation network to effectively handle real-world variabilities. By normalising translation and scale prior to the network input and appropriately transforming the predicted pose afterward, the system is equipped to deliver robust and precise pose estimation across diverse scenarios. These methodologies significantly enhance the practical applicability of the pose estimation system, making it a reliable tool in dynamic environments where accuracy is crucial.

4.4.6 Training data generation

The proposed method has been designed as a self-supervised learning-based approach, allowing the network to learn without the need for labelled data. To facilitate this, a synthetic data generation pipeline has been developed, mimicking the properties of real-world data. The overall pipeline is designed as below

1. **Initiating clouds:** Object model clouds are used as the starting point to generate edge point cloud and model cloud for each data sample.

2. **Rotation:** Model cloud is rotated using a generated rotation matrix. For training data: Rotation matrix is formed by uniformly sampling Euler angles within the $[0, 2\pi]$ range for the $\{x, y, z\}$ axes. For validation data: Rotation matrix is generated by randomly selecting Euler angles within the $[0, 2\pi]$ range for each axes.

3. **Translation:** Rotated model cloud is translated randomly within a specified distance range along all three axes. Smaller translation range is used along the $\{x, y\}$ axes, and a larger range along the $\{z\}$ axis to simulate depth camera data [209].

4. **Depth map generation:** Transformed model cloud is projected onto an image plane using simulated depth camera intrinsic parameters (refer Appendix B) to form a depth map. Partial cloud, reprojected back into point cloud space, simulates a real-world captured point cloud.

5. **Occlusion simulation:** A partial cloud is generated by randomly cropping the original point cloud using a cuboid defined by its bounding parameters. This cropping process simulates an occluded cloud by removing specific regions, resulting in a partial cloud that closely resembles a scene with obscured parts of the object.

6. **Edge cloud derivation:** An edge cloud is derived from the occluded cloud using a 3D edge detector. The edge cloud is processed to include all the points within a certain distance threshold from the edge points to have wider edges, which help in improving estimation accuracy.

7. **Pre-processing:** Pre-processing manipulations, as discussed in the previous subsection, are applied to the edge cloud and initial model cloud to obtain normalised clouds. The normalisation help to make the sample independent from translational and scale variations.

8. **Unordered point clouds:** Points in the pre-processed clouds are randomly ordered to resemble unordered point clouds typical in real-world scenes.

9. **Real world noise:** Gaussian random noise, with randomised mean and variance, is added to both point and normal coordinates to obtain noisy point clouds, simulating a real-world depth sensor data.

10. **Random selection:** V and U points are randomly selected from the edge and model clouds to create a fixed size input. to be passed to the network. This also simulates missing information in real world depth sensors.

This comprehensive data simulation approach bridges the gap between synthetic training environments and real-world application conditions, enabling robust neural network training without relying extensively on large labelled datasets. Real-world challenges, such as shadows and occlusions caused by other objects, are simulated directly within the 3D point cloud space through the introduction of noise and occlusion effects. The pipeline's versatility in replicating diverse environmental conditions ensures that the pose estimation model generalises effectively to new, unseen scenarios. By simulating realistic sensor data and occlusion patterns, the approach prepares the system to handle practical deployment challenges with greater efficiency. This method not only enhances model performance but also significantly reduces the reliance on costly and time-intensive data collection and annotation processes.

4.5 Experimental results

To evaluate the effectiveness of the proposed pose estimation framework, the accuracy and reliability of the pose estimator were assessed using multiple real-world benchmark datasets. The experimental setup included a detailed analysis of performance metrics such as precision,

recall, and the overall error rate of the estimated poses when compared to ground truth data. In this section, the datasets employed are described, along with the experimental setup, and a comprehensive comparative analysis with SOTA methods is presented.

4.5.1 Dataset description and experimental setup

Three benchmark datasets were considered: LineMod (LM) [103], Occluded-LineMod (LMO) [31], and TUD-L [107]. These datasets provide real RGB-D data captured by depth sensors, featuring objects in highly complex and unstructured environments. More detailed descriptions of these datasets can be found in Section 2.7. For each dataset, the provided object model clouds were utilised to generate the training and validation data, as outlined in Section 4.4.6. The test performance was assessed using the challenging real scene data provided by these datasets. Experiments were conducted in accordance with the pose testing protocols specified by the Benchmark for 6D Object Pose Estimation (BOP) Challenge 2018 [107]. In the proposed work, the object cloud was first segmented from the scene point cloud before processing and passing it to the network. To prevent errors introduced by the segmentation method during experimentation, the object clouds were extracted from the scenes using the ground truth masks provided in the datasets for the intended objects.

The network was implemented using the PyTorch [14] library in Python and trained with a distributed data learning strategy. The model was trained with a batch size of 20 for 1001 epochs. For the training data, rotations were uniformly sampled at intervals of 12°, generating 27,000 samples for each object. For validation, 1,000 randomly rotated samples were generated. The translation ranged between $[-50, 50]$mm in the $\{x, y\}$ directions and $[50, 1000]$mm in the $\{z\}$ direction, and was sampled randomly for both training and validation. The point cloud was voxel downsampled with a voxel size of 0.02 times the object diameter. Noise was added to the point coordinates with a mean of 0 and a variance of 10% of the downsample voxel size. Additionally, noise in the point cloud normals was introduced with a mean of 0 and a variance of 0.02. Finally, 256 edge points and 2048 object model points were selected to be passed to the network. The weighted ADAM optimiser was utilised with a step learning rate starting at

0.001 to train the model. ICP is applied for fine refinement of pose, utilising the predicted pose as initial input, for maximum of 1000 iterations.

The effectiveness of the proposed method was rigorously evaluated using the error metrics defined by the BOP Challenge [107], which include Average Recall (AR), Visible Surface Discrepancy (VSD), Maximum Symmetry-Aware Surface Distance (MSSD), and Maximum Symmetry-Aware Projection Distance (MSPD). Additionally,mean Average Precision (mAP) of the Average Distance of Model Points ADD error, with 10% of object diameter threshold, was calculated for both non-symmetric and symmetric objects as a further measure of performance [281]. To provide a comprehensive analysis, the was computed for rotation errors at thresholds of $\{5°, 10°, 20°\}$ and translation errors at thresholds of $\{1cm, 2cm, 5cm\}$. The mathematical formulations for these error calculations can be found in Appendix B. Furthermore, qualitative assessments were conducted by visualising the estimated and ground truth poses projected onto real-world scene RGB images, offering an intuitive comparison of the method's accuracy.

4.5.2 Compared methods

To thoroughly evaluate the performance of the proposed pose estimation framework, a comparative analysis with both traditional and learning-based methods from the literature was conducted. The methods selected for comparison were those that rely solely on point cloud information, excluding any texture details, to perform pose estimation. Among traditional approaches, ICP [16], FGR [312], TEASER++ [291], and Super4PCS [177] were considered. For learning-based methods, DCP [271], IDAM [145], and MatchNorm [58] were selected. It is understood that none of the edge-based methods in literature have been designed to utilise only spatial information for pose estimation. The evaluation of these methods was carried out using the parameters and training strategies presented in their respective works to ensure a fair comparison. This approach allows for a comprehensive understanding of the strengths and limitations of the proposed method in relation to established techniques.

4.5.3 Comparison analysis: LineMod dataset

The quantitative performance of the proposed method on the LineMod dataset is presented in Table 4.1 and Table 4.2 across all metrics. The analysis of these results has led to several important insights.

Method	BOP Metrics				ADD
	AR	VSD	MSSD	MSPD	0.1d
ICP [16]	0.044	0.092	0.014	0.027	0.01
FGR [312]	0.026	0.068	0.000	0.010	0.00
TEASOR++ [291]	0.094	0.108	0.076	0.098	0.03
Super4PCS [177]	0.165	0.117	0.178	0.201	0.10
DCP(v2) [271]	0.044	0.057	0.025	0.049	0.00
IDAM [145]	0.067	0.050	0.070	0.081	0.02
MatchNorm-IDAM [58]	0.225	0.170	0.248	0.258	0.13
MatchNorm-DCP [58]	0.559	0.427	0.608	0.644	0.54
EOPEN	0.607	0.530	0.650	0.644	0.783
EOPEN-ICP	**0.620**	**0.546**	**0.664**	**0.650**	**0.800**

Table 4.1: Quantitative performance comparison for LineMod (LM) [103] dataset with ADD and BOP challenge metrics.

Traditional methods: EOPEN significantly outperformed traditional methods such as ICP, FGR, TEASOR++, and Super4PCS across all BOP metrics and ADD. While ICP and FGR demonstrated minimal performance with AR values of 0.044 and 0.026, respectively, EOPEN achieved a robust AR of 0.607, indicating a substantial improvement in pose estimation accuracy. Similarly, in terms of ADD, EOPEN's score of 0.783 was notably higher than those of traditional methods, with Super4PCS being the closest competitor at only 0.10. Regarding rotation mAP, EOPEN significantly surpassed traditional methods, with Super4PCS achieving 0.02 at the 5° threshold. For translation mAP, EOPEN demonstrated strong performance across all thresholds, indicating precise translation estimation.

Deep learning-based methods: When compared to deep learning-based methods such as DCP, IDAM, and MatchNorm variations, EOPEN demonstrated SOTA performance across multiple metrics. While MatchNorm-DCP achieved strong results with an AR of 0.559 and an ADD

Method	Rotation mAP			Translation mAP		
	5°	10°	20°	1cm	2cm	5cm
ICP [16]	0.00	0.01	0.01	0.04	0.27	0.82
FGR [312]	0.00	0.00	0.00	0.05	0.31	0.89
TEASOR++ [291]	0.01	0.03	0.05	0.03	0.21	0.73
Super4PCS [177]	0.02	0.09	0.15	0.04	0.31	0.89
DCP(v2) [271]	0.00	0.00	0.01	0.05	0.24	0.83
IDAM [145]	0.00	0.01	0.05	0.03	0.16	0.71
MatchNorm-IDAM [58]	0.01	0.07	0.15	0.13	0.38	0.87
MatchNorm-DCP [58]	0.10	0.27	0.49	0.26	0.60	0.95
EOPEN	0.12	0.39	0.61	0.63	0.886	0.99
EOPEN-ICP	**0.20**	**0.59**	**0.75**	**0.70**	**0.94**	**1.00**

Table 4.2: Quantitative performance comparison for LineMod (LM) [103] dataset with rotation and translation mAP metrics at different thresholds.

score of 0.54, EOPEN surpassed it with an AR of 0.607 and an ADD score of 0.783, showcasing its robust edge feature utilisation for pose estimation. In rotation accuracy, EOPEN achieved competitive scores, with a rotation mAP of 0.61 at 20°, outperforming MatchNorm-DCP (0.49). Furthermore, EOPEN excelled in translation accuracy, achieving the highest translation mAP scores of 0.63, 0.886, and 0.99 at the 1cm, 2cm, and 5cm thresholds, respectively.

The combination of EOPEN with ICP (EOPEN-ICP) further enhanced its performance across all metrics. EOPEN-ICP achieved the highest AR (0.620) and ADD (0.800) scores, along with substantial improvements in rotation and translation accuracy. At the 5°, 10°, and 20° thresholds, EOPEN-ICP reached rotation mAP scores of 0.20, 0.59, and 0.75, respectively, significantly outperforming MatchNorm-DCP and other methods. In translation tasks, EOPEN-ICP achieved mAP scores of 0.70, 0.94, and 1.00 at the 1cm, 2cm, and 5cm thresholds, demonstrating unparalleled robustness in handling translational variations.

Overall, EOPEN and EOPEN-ICP consistently delivered top-tier results, with EOPEN-ICP excelling in both rotation and translation accuracy. This highlights the effectiveness of ICP in refining alignments and the capability of EOPEN to achieve high accuracy using only 5% of the points compared to other methods. Moreover, the proposed method's reliance on synthetic training data, without the need for labelled datasets, further underscores its efficiency and practicality for pose estimation in complex environments. These results establish EOPEN-ICP

as a highly competitive method for precise pose estimation under challenging conditions.

4.5.4 Comparison analysis: Occluded-LineMod dataset

The performance of EOPEN on the LMO dataset, assessed across all quantitative metrics, is provided in Table 4.3 and Table 4.4. The detailed analysis is provided below.

Method	BOP Metrics				ADD
	AR	VSD	MSSD	MSPD	0.1d
ICP [16]	0.044	0.085	0.014	0.032	0.01
FGR [312]	0.021	0.055	0.000	0.009	0.00
TEASOR++ [291]	0.083	0.096	0.060	0.093	0.02
Super4PCS [177]	0.080	0.054	0.072	0.113	0.03
DCP(v2) [271]	0.044	0.055	0.018	0.059	0.00
IDAM [145]	0.090	0.063	0.088	0.119	0.02
MatchNorm-IDAM [58]	0.187	0.142	0.191	0.229	0.09
MatchNorm-DCP [58]	0.410	0.289	0.439	0.501	0.33
EOPEN	0.472	0.389	0.500	0.533	0.55
EOPEN-ICP	**0.494**	**0.417**	**0.510**	**0.557**	**0.58**

Table 4.3: Quantitative performance comparison for Occluded-LineMod (LMO) [31] dataset with ADD and BOP challenge metrics.

Method	Rotation mAP			Translation mAP		
	5°	10°	20°	1cm	2cm	5cm
ICP [16]	0.01	0.01	0.01	0.07	0.36	0.85
FGR [312]	0.00	0.00	0.00	0.08	0.43	0.85
TEASOR++ [291]	0.01	0.02	0.05	0.04	0.26	0.77
Super4PCS [177]	0.01	0.03	0.06	0.06	0.31	0.83
DCP(v2) [271]	0.00	0.00	0.01	0.03	0.30	0.83
IDAM [145]	0.00	0.02	0.06	0.07	0.26	0.76
MatchNorm-IDAM [58]	0.02	0.08	0.18	0.15	0.44	0.84
MatchNorm-DCP [58]	0.07	0.19	0.36	0.24	0.57	0.88
EOPEN	0.04	0.24	0.48	0.43	0.80	0.96
EOPEN-ICP	**0.20**	**0.38**	**0.66**	**0.58**	**0.95**	**0.99**

Table 4.4: Quantitative performance comparison for Occluded-LineMod (LMO) [31] dataset with rotation and translation mAP metrics at different thresholds.

Traditional methods: EOPEN demonstrated significant improvement over traditional methods such as ICP, FGR, TEASOR++, and Super4PCS across various metrics. Specifically, in

the BOP metrics, EOPEN achieved an AR of 0.472 and an ADD of 0.55, clearly outperforming traditional methods like ICP (AR: 0.044, ADD: 0.01) and FGR (AR: 0.021, ADD: 0.00). This result highlights the superior capability of EOPEN in handling complex pose estimation tasks, especially in scenarios with large occlusions. Additionally, EOPEN demonstrated strong rotation and translation accuracy, achieving an mAP of 0.48 at the 20° threshold and 0.96 at the $5cm$ threshold, significantly outperforming traditional methods like TEASOR++ and ICP. These results underscore EOPEN's robustness in challenging scenarios with occlusions and sparse data.

Deep learning-based methods: When compared to deep learning-based methods, EOPEN demonstrated significant improvements across multiple metrics. MatchNorm-DCP, a strong baseline, achieved an AR of 0.410 and an ADD score of 0.33, but EOPEN surpassed these results with an AR of 0.472 and an ADD score of 0.55, highlighting its effective use of edge features. In terms of rotation mAP, EOPEN achieved 0.48 at the 20° threshold, outperforming MatchNorm-DCP's score of 0.36. For translation mAP, EOPEN excelled across all thresholds, particularly at the $5cm$ threshold, where it achieved a score of 0.96, significantly higher than MatchNorm-DCP's 0.88, demonstrating its robust handling of translational variations.

The integration of EOPEN with ICP (EOPEN-ICP) further enhanced its performance, achieving the highest AR (0.494) and ADD (0.58) scores among all methods. EOPEN-ICP outperformed other methods in rotation accuracy, achieving mAP scores of 0.20, 0.38, and 0.66 at the 5°, 10°, and 20° thresholds, respectively. For translation accuracy, EOPEN-ICP achieved mAP scores of 0.58, 0.95, and 0.99 at the $1cm$, $2cm$, and $5cm$ thresholds, respectively, showcasing its strong ability to align objects even in highly occluded scenarios.

Overall, EOPEN consistently delivered competitive results, excelling in both rotation and translation metrics. Its edge-based approach proved particularly effective in overcoming challenges posed by occlusions in the LMO dataset. The combination with ICP further refined its performance, making EOPEN-ICP a robust and competitive method for pose estimation in complex and highly occluded environments. This performance, achieved while utilising only

5% of the points compared to other methods, underscores its efficiency and practicality.

4.5.5 Comparison analysis: TUD-L dataset

The performance of the EOPEN method on the TUD-L dataset, as presented in Tables 4.5 and 4.6, is analysed below.

Method	BOP Metrics				ADD
	AR	VSD	MSSD	MSPD	0.1d
ICP [16]	0.056	0.117	0.023	0.027	0.02
FGR [312]	0.029	0.071	0.007	0.008	0.01
TEASOR++ [291]	0.188	0.175	0.196	0.193	0.17
Super4PCS [177]	0.418	0.265	0.500	0.488	0.54
DCP(v2) [271]	0.038	0.025	0.051	0.039	0.01
IDAM [145]	0.091	0.067	0.108	0.099	0.05
MatchNorm-IDAM [58]	0.427	0.306	0.500	0.475	0.43
MatchNorm-DCP [58]	**0.884**	**0.806**	**0.924**	**0.923**	0.92
EOPEN	0.765	0.648	0.791	0.846	0.84
EOPEN-ICP	0.835	0.685	0.900	0.921	**0.95**

Table 4.5: Quantitative performance comparison for TUD-L [107] dataset with ADD and BOP challenge metrics.

Method	Rotation mAP			Translation mAP		
	5°	10°	20°	1cm	2cm	5cm
ICP [16]	0.02	0.02	0.02	0.01	0.14	0.57
FGR [312]	0.00	0.01	0.01	0.04	0.25	0.63
TEASOR++ [291]	0.13	0.17	0.19	0.03	0.22	0.56
Super4PCS [177]	0.30	0.50	0.56	0.05	0.40	0.92
DCP(v2) [271]	0.00	0.01	0.02	0.02	0.07	0.55
IDAM [145]	0.03	0.05	0.10	0.02	0.08	0.49
MatchNorm-IDAM [58]	0.36	0.46	0.53	0.23	0.47	0.57
MatchNorm-DCP [58]	**0.70**	0.81	0.87	**0.71**	0.86	0.97
EOPEN	0.16	0.62	0.86	0.58	0.90	0.98
EOPEN-ICP	0.34	**0.84**	**0.92**	0.62	**0.93**	**0.99**

Table 4.6: Quantitative performance comparison for TUD-L [107] dataset with rotation and translation mAP metrics at different thresholds.

Traditional methods: EOPEN has been shown to outperform traditional methods such as ICP, FGR, TEASOR++, and Super4PCS across all BOP metrics and ADD. For instance, EOPEN

achieved an AR of 0.765 and an ADD of 0.84, which is significantly higher than the performance of ICP (AR: 0.056, ADD: 0.02) and FGR (AR: 0.029, ADD: 0.01). While Super4PCS performed relatively well with an AR of 0.418 and an ADD of 0.54, it still lagged behind EOPEN. This highlights EOPEN's superior ability to handle complex pose estimation tasks.

Deep learning-based methods: EOPEN also demonstrated competitive performance when compared to other deep learning-based methods. Although MatchNorm-DCP achieved the highest AR (0.884) and ADD (0.92), EOPEN remained strong with an AR of 0.765 and an ADD of 0.84. Additionally, EOPEN outperformed MatchNorm-IDAM (AR: 0.427, ADD: 0.43), indicating its robustness in leveraging edge features for improved pose estimation. EOPEN's rotation mAP of 0.86 at the 20° threshold and its high translation mAP of 0.98 at the 5cm threshold further demonstrated its effectiveness. EOPEN-ICP show competitive performance to the top performing MatchNorm-DCP method, achieving an AR of 0.835 and an ADD of 0.95. The rotation mAP of EOPEN-ICP at the 20° threshold was 0.92, and its translation mAP at the 5cm threshold was 0.99, indicating its competitive performance. This combination strikes a balance between accuracy and computational efficiency, making it particularly effective in scenarios where dense scene points are difficult to acquire. Overall, EOPEN consistently demonstrated competitive results across the TUD-L dataset, particularly excelling in translation accuracy and performing strongly in rotation accuracy when combined with ICP.

4.5.6 Qualitative performance analysis

A qualitative analysis of the EOPEN method has been conducted by examining Figure 4.3 and Figure 4.4, which depicts the predicted poses across datasets—LineMod, Occluded LineMod, and TUD-L. In Figure 4.3, the predicted poses have been overlaid on the RGB images of the scenes, allowing to visually assess the accuracy of EOPEN's 6-DoF pose estimations. The alignment of the object model with the actual object in the scene indicates that EOPEN's predicted poses closely match the ground truth in most cases. Minor deviations have been observed in some challenging scenarios, such as those involving occlusions. However, EOPEN's

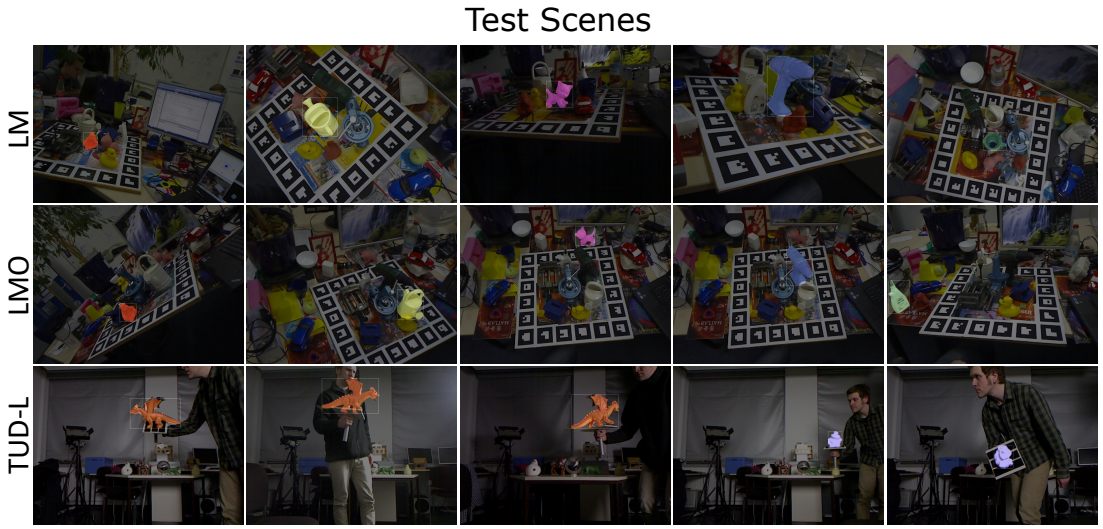

Figure 4.3: Qualitative performance of EOPEN method for LineMod (LM), Occluded-LineMod (LMO), and TUD-L datasets across different scenes.

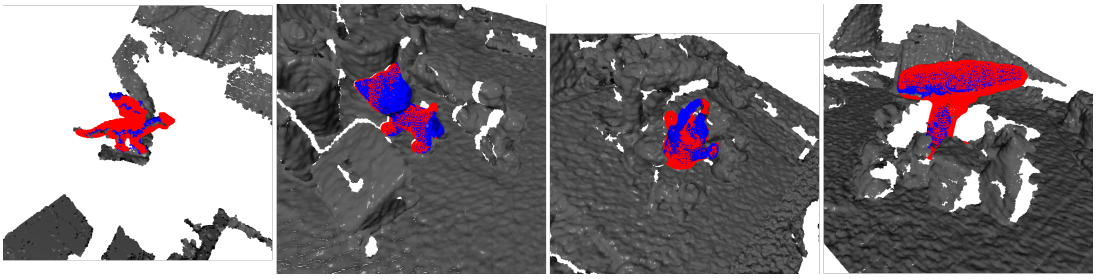

Figure 4.4: Qualitative performance of EOPEN method in 3D view for LineMod (LM), Occluded-LineMod (LMO), and TUD-L datasets across different scenes.

robustness is demonstrated by its strong performance in cases where traditional methods often struggle, particularly with texture-less objects or highly cluttered scenes. The results of 3D point cloud alignment are presented in Figure 4.4, illustrating the high precision achieved in both rotational and translational alignment. This precision is evident from the transformed model cloud (displayed in red) closely overlaying the object scene cloud (shown in blue).

Overall, this qualitative analysis confirms that EOPEN provides reliable and precise pose estimation, with the predicted poses closely corresponding to the ground truth in most scenarios. The minimal discrepancies observed further highlight the method's robustness and its applicability to real-world conditions.

4.5.7 Inference time analysis

The inference time of a method refers to the duration required for processing the input and generating the output. Table 4.7 presents a comparison of inference times between the developed method and other approaches. The results reveal that traditional methods, such as ICP [16], FGR [312], TEASOR++ [291], and Super4PCS [177], exhibit high computational times, rendering them less viable for real-time applications. Conversely, deep learning-based approaches, including IDAM [145], DCP(v2) [271], and MatchNorm [58] variants, demonstrate real-time performance for pose estimation tasks. The developed method achieves inference times of 15ms directly and 18ms when executed with 1000 iterations of ICP, underscoring its suitability for live robotic applications.

Table 4.7: Inference times for 6-DoF pose estimation methods.

Method	Time (ms)	Method	Time (ms)
ICP [16]	720	DCP(v2) [271]	14.2
FGR [312]	220	IDAM [145]	38
TEASOR++ [291]	213.5	MatchNorm-IDAM [58]	17.3
Super4PCS [177]	3285	MatchNorm-DCP [58]	18.8
EOPEN	15	**EOPEN-ICP**	18

4.6 Summary

A deep learning-based 6-DoF pose estimation method has been proposed in this chapter, which utilises the 3D edge point cloud of an object, along with its model cloud, to estimate the rotation and translation that align the two clouds. The method employs a hierarchical PointNet++ architecture to extract both fine and coarse features from the data at multiple scales. These point features are then used to identify keypoint correspondences between the edge cloud and model cloud through feature matching, which are subsequently regressed to obtain the rotation and translation parameters. The proposed method is designed as a self-supervised learning approach. An online data generation pipeline is introduced, where synthetic edge cloud data is generated using object model point clouds. Additional augmentations, such as cropping,

scaling, and noise, are applied to better reflect real-world data conditions. The data is generated over a wide range of rotations and translations, which is then used to train the network, ensuring generalised learning.

The effectiveness of the proposed method has been evaluated using three benchmark datasets: LineMod, Occluded-LineMod, and TUD-L. The performance is compared against both traditional and deep learning-based pose estimation methods from the literature. The analysis shows that the proposed method outperforms almost all the compared methods, demonstrating robust performance while using only 5% of the points compared to other methods.

Chapter 5

Articulated object classification

Preliminary remarks

The research in this chapter was published in the following paper:

- **A. Aggarwal**, R. Stolkin, & N. Marturi, Local Region-to-Region Mapping-based Approach to Classify Articulated Objects. 2023 20th Conference on Robots and Vision (CRV), Montreal, Quebec, Canada, 2023, pp. 177-183. https://doi.org/10.1109/CRV60082.2023.00030

Note: Some parts of this chapter, including problem descriptions, specific equations, variable descriptions, interpretations of results, and figure/table captions, are adapted from [10].

5.1 Introduction

Understanding the properties of objects is fundamental for various applications in robotics and automation. Each type of object presents unique challenges that must be addressed through specialised modelling, recognition, and control techniques. By accurately categorising and understanding these properties, more effective and versatile robotic systems can be developed, leading to advancements in automation, manufacturing, and human-robot interaction e.g., a robot tasked with clearing unknown objects in case of hazardous decommissioning tasks as

discussed by [173].

In the context of articulated objects, the complexity of object recognition and classification increases significantly. Unlike rigid objects, which maintain a fixed shape and structure, articulated objects possess joints or segments that can move relative to one another. The classification of such objects poses significant challenges due to the variability in their configurations and the complex relationships between their components. Historically, the classification of articulated objects relied on traditional computer vision techniques that often struggled with the dynamic nature of these objects. Early approaches focused on detecting key features and landmarks, but these methods were typically limited by their sensitivity to variations in pose, occlusion, and lighting conditions [199, 200]. As a result, these methods required substantial manual intervention and fine-tuning, making them less practical for real-world applications.

In response to these challenges, a specialised method for articulated object classification has been developed to detect if an object is either rigid or articulated. Articulated objects, as previously discussed, consist of multiple rigid segments connected by joints. The complexity of these objects lies in the diverse range of motions that each segment can exhibit, depending on the type of joint that links them. The relative motion of these segments is crucial in determining the overall behaviour of the articulated object, especially during manipulation and interaction with other objects or environments. These segmental movements are generally classified into three primary categories: revolute, prismatic, and free-form, each of which plays a vital role in the kinematic modelling of articulated objects.

- **Revolute joints:** Revolute joints, also known as rotational joints, allow for rotational movement around a single axis. This type of motion is commonly found in mechanical systems such as robotic arms and human joints like elbows or knees. In kinematic modelling, revolute joints are essential for calculating angular displacements and velocities, which are critical for tasks requiring precision and control in rotational motion.

- **Prismatic joints:** Prismatic joints, also referred to as linear or sliding joints, permit movement along a single translational axis. This linear motion is typical in applications where a straight-line movement is required, such as in sliding doors, piston mechanisms,

or linear actuators in robotic systems. Kinematic modelling of prismatic joints involves the calculation of linear displacements and velocities, which are vital for operations that necessitate accurate and smooth translational motion.

- **Free-form movements:** Free-form movements are more complex, encompassing a combination of rotational and translational motions along multiple axes. These movements occur in joints that do not conform to a single, predefined axis of motion, allowing for greater flexibility and range of movement. Free-form joints are commonly found in biological systems, such as the shoulder joint in humans, and in advanced robotic systems designed to mimic human-like dexterity. Kinematic modelling of free-form movements is more intricate, often requiring advanced algorithms and simulations to predict and control the wide range of possible motions.

Understanding these categories of movement is essential for the kinematic modelling of articulated objects. Accurate modelling enables the prediction of how each segment will move in response to external forces or control inputs, which is critical for tasks such as robotic manipulation, motion planning, and the development of autonomous systems. Vision-based classification of articulated objects plays a pivotal role in this process by allowing systems to visually differentiate between rigid and articulated objects, as well as to identify the specific joints and segments involved. By leveraging vision-based techniques, engineers and roboticists can design more efficient, adaptable, and capable robotic systems that interact seamlessly with complex environments. This approach not only enhances the accuracy of kinematic modelling but also improves the ability of autonomous systems to predict and respond to the dynamic behaviour of articulated objects in real-time.

5.1.1 Related work

In the literature, the modelling, pose estimation, and tracking of articulated objects have been extensively studied by various researchers [243, 199, 200, 48, 74, 214, 304, 178, 168, 115, 206, 23]. A method for learning a kinematic model of an articulated object from a video

sequence was presented by Sturm et al. [243]. In this approach, an initial object articulation model is learned, followed by verification of the learned model on a testing sequence. The method is designed to handle free-form movements but is limited to objects consisting of only two rigid parts. Additionally, the method relies on markers to accurately determine the object's position, which may not be feasible in real-world scenarios. A vision-based method for estimating robot configuration and kinematics was proposed by Valerio et al. [199], in which the configuration of a multi-joint robot was estimated by tracking visual markers, without the use of any proprioceptive sensors.

A marker-less object skeleton estimation method from a multi-view point cloud was proposed by Chun et al. [48], where skeleton curvatures are utilised to align and form the object model. Although this method eliminates the need for markers, it requires a complete point cloud of the object. In contrast, the proposed method can operate on single-view (marker-less) point clouds derived from a video sequence to classify articulated objects. A marker-less, single image-based system for robot state tracking and configuration estimation was proposed by Valerio et al. [200], in which individual parts of the robot are identified and their poses are estimated. However, the system relies on RGB images, and the accuracy of part estimation is dependent on the presence of texture on the object.

Several deep learning-based methods for estimating object articulation types and articulation axes have been introduced in the literature [74, 214, 304]. The approach proposed by Fan et al. [74] predicts object and hand models to understand object-hand interactions. An articulated object detection method using RGB-D video sequences was introduced by Qian et al. [214], where object parts, planes, and axes are detected in each image, and the temporal relation between frames is then utilised to predict the object bounding box, articulation planes, and axes.

A method for predicting part kinematic constraints using object RGB-D images and corresponding part segmentations was proposed by Zeng et al. [304]. While learning-based methods demonstrate good performance and robustness to noise, they require accurately labelled training data, which may not always be available in practical real-world scenarios. Additionally, with free-form articulated objects, numerous possible object states exist, making the labelling process

time-consuming and labor-intensive. To address this limitation, an online classification method has been developed that can directly infer the object's type without the need for labelled data.

A part segmentation-based model creation and tracking system in images was proposed by Meyer et al. [178], where shape-based models are created for each rigid component of the object, and these models are tracked to form a complete articulation model. Finally, a three-step articulation state estimation method was proposed by Martín et al. [168], where RGB-D images are processed over an interaction sequence to predict the object's state.

5.1.2 Organisation

The rest of the chapter is organised as follows. Section 5.2 presents the key contribution in the proposed method. Background information required to understand the development of this method is given in Section 5.3. Section 5.4 describes the proposed algorithm to classify objects. Section 5.5 presents a detailed analysis of results, along with a description of datasets. Finally, Section 5.6 presents a summary.

5.2 Contribution

The aforementioned methods are designed to estimate the model and kinematics of objects without first determining whether the object is rigid or articulated. However, it is equally important to initially analyse whether the object in the scene is articulated before attempting to model or track its movements. To address this, a temporal local region-to-region registration is performed on 3D point clouds, which are obtained from an observed sequence of the object's movements or interactions.

In the proposed approach, the process begins by generating a mask for the object of interest in each video frame, using an off-the-shelf object localisation method [185]. This is accomplished by utilising the video corresponding to the object's movements. Following the generation of the mask and the extraction of the corresponding depth information, a point cloud of the object is created for each frame. Subsequently, local region registrations are performed on each set of

consecutive frame point clouds, resulting in local transformations from one frame to the next. These local transformations are then quantised, a step that reduces the effect of point cloud noise, and clustered to obtain the final set of unique transformations between the local regions of the two frames. For a rigid object, these transformations remain consistent across all local regions, whereas for articulated objects, the local transformations may vary, with different articulated parts exhibiting distinct rotations or translations.

The key contributions of this work are summarised as follows:

- A novel model-free object classification method is introduced to identify whether an object is rigid or articulated by analysing a sequence of its movements.
- A constraint-free, registration-based approach is developed, which employs local region-to-region mapping to detect and classify objects with any type of articulation.

The primary advantage of the proposed method lies in its constraint-free nature, allowing it to classify any type of articulated object without being restricted to specific types of articulations. Additionally, the method is parametric and does not require any training, making it independent of labelled data requirements.

The performance of this approach has been evaluated using two publicly available benchmark datasets, which contain a mixture of rigid and articulated objects. The results demonstrate the effectiveness of the proposed method in accurately classifying articulated objects.

5.3 Background

In the fields of robotics, computer vision, and industrial automation, understanding the physical properties of objects is essential for accurate modelling, recognition, and manipulation. Objects can generally be classified into three categories based on their physical characteristics: rigid, deformable, and articulated objects. Each of these categories presents unique challenges and opportunities in terms of object interaction, recognition, and control.

- **Rigid objects:** Rigid objects maintain a constant shape and volume, irrespective of

external forces acting upon them. Mathematically, a rigid object is defined as one for which the distance between any two points remains constant over time. This implies that rigid objects do not bend, stretch, or compress, and their internal structure remains unchanged during motion. The movement of rigid objects can be accurately described using a combination of linear and rotational transformations, making it possible to model their behaviour using classical mechanics. The kinematics of rigid objects are relatively straightforward, facilitating the easy prediction of their future positions based on their current state. Common examples of rigid objects include metal parts, tools, and most household items such as cups and plates.

- **Deformable objects:** In contrast to rigid objects, deformable objects can undergo changes in shape, size, and structure when subjected to external forces. These objects do not have a fixed geometry and can be compressed, stretched, bent, or twisted. Depending on their material properties, deformable objects may exhibit elastic behaviour, where they return to their original shape after the removal of forces, or plastic behaviour, where permanent deformation occurs. The kinematics of deformable objects are complex, requiring sophisticated modelling techniques, such as Finite Element Analysis (FEA), to predict their deformation patterns. Examples of deformable objects include rubber bands, sponges, fabrics, and biological tissues.

- **Articulated objects:** Articulated objects are composed of multiple rigid segments connected by joints, allowing for a wide range of motion. These joints provide the flexibility needed to perform complex movements, distinguishing articulated objects from purely rigid or deformable objects. The motion of articulated objects is characterised by multiple Degrees of Freedom (DoF), with each joint contributing to the overall DoF of the object. The movement of articulated objects is often hierarchical, where the motion of one segment influences the position and orientation of connected segments. Examples of articulated objects include human limbs, robotic arms, folding furniture, and multi-part tools.

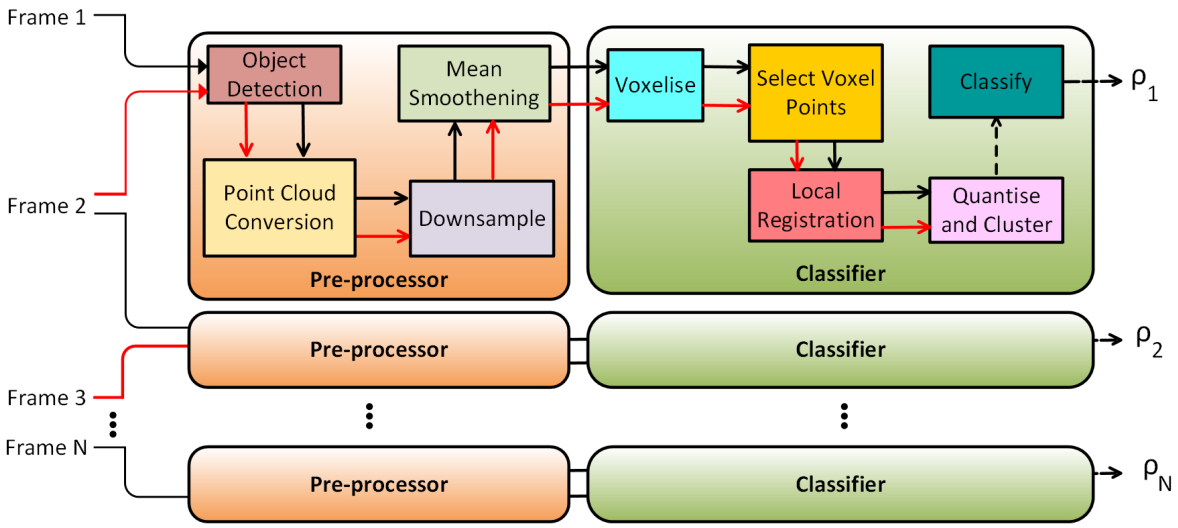

Figure 5.1: The overall pipeline for the articulated object classification method.

5.4 Methodology

In this section, the developed framework for articulated object classification is described. For demonstration, an N-frame video where an object is being manipulated is considered where each frame consists of a depth map and an RGB image. Then for each set of consecutive frames $i, i+k \in N$, registration is performed between local regions of corresponding point clouds in order to classify the motion between them as ***no motion, rigid motion, or articulated motion***. It is worth noting that the point cloud of a frame is generated using the depth information of intended object regions. This process is presented in detail in Section 5.4.2. To perform this task, initially, each frame is passed through a set of pre-processing steps for noise and background filtering. Afterwards, the corresponding filtered point clouds of the frames are passed through the classifier. The complete pipeline is shown in Figure 5.1. Majorly, the pipeline is divided into two modules, i.e., the pre-processor and the classifier. Each of these modules is described in the following subsections.

5.4.1 Method overview

In this subsection, the developed framework for articulated object classification is described. For demonstration purposes, an N-frame video, in which an object is being manipulated, is

considered. Each frame consists of a depth map and an RGB image. For each set of consecutive frames, $i, i + k \in N$, registration is performed between local regions of the corresponding point clouds to classify the type of motion occurring between them as either no motion, rigid motion, or articulated motion.

These classifications are critical for understanding the underlying behaviour of the object:

- **No Motion (NM):** This classification indicates that the object remains stationary between frames i and $i + k$. No significant changes in position or orientation are detected within the local regions of the point clouds, suggesting that the object is either at rest or not being manipulated during this interval.

- **Rigid Motion (RM):** This classification is applied when the object undergoes uniform motion, where all parts of the object move in unison without any relative movement between them. In this case, the motion is characterised by consistent translations and/or rotations across the entire object, indicative of a rigid structure. This type of motion is typical for objects that maintain a constant shape and internal configuration, such as solid tools or mechanical components.

- **Articulated Motion (AM):** This classification identifies the presence of more complex movements where different parts of the object move relative to each other. Such motion is characterised by varying transformations within different local regions of the object's point clouds, indicating that the object consists of multiple rigid segments connected by joints. This type of motion is common in articulated objects such as robotic arms, human limbs, or mechanical linkages, where the motion involves bending, rotating, or translating individual segments independently.

It should be noted that the point cloud for each frame is generated using the depth information of the intended object regions. This process is described in detail in Section 5.4.2. To perform this classification task, each frame is initially subjected to a series of pre-processing steps to filter out noise and background elements. Following this, the corresponding filtered point clouds of the frames are processed through the classifier to determine the type of motion. The complete

pipeline is illustrated in Figure 5.1. The pipeline is primarily divided into two modules: the pre-processor and the classifier. Each of these modules is described in detail in the following subsections.

5.4.2 Pre-processing

In a typical scenario, the depth map $\mathbf{D}_i$ and RGB image $\mathbf{C}_i$ of a given frame i may contain various items other than the object of interest, such as walls, tables, and other background elements. For the proof of concept, it is assumed that in any sequence of frames, only a single object is being manipulated, while all other objects remain stationary. Consequently, these additional objects can be considered as unwanted noise elements that may adversely affect the performance of the classifier. Moreover, the presence of these stationary objects introduces a logical challenge in achieving accurate classification, which is further discussed in the next subsection.

Algorithm 3 Pre-processing of input frame depth map and RGB image.

Require: Depth map $\mathbf{D}_i$, RGB image $\mathbf{C}_i$ for ith frame, object id O, Camera intrinsic $\mathbf{c}_K^i$, outlier noise standard deviation s, downsample size v, smoothening radius r
Ensure: Cropped Point Cloud $\mathbf{B}_i^d$

$p, bb = ObjectDetection(\mathbf{C}_i, O)$
if $p \neq None$ **then**
    $\mathbf{D}_i^c = Crop(\mathbf{D}_i, bb)$
    $\mathbf{B}_i = ConvertToCloud(\mathbf{D}_i^c, \mathbf{c}_K^i)$
    $\mathbf{B}_i^o = OutlierNoiseRemoval(\mathbf{B}_i, s)$
    $\mathbf{B}_i^v = VoxelDownsample(\mathbf{B}_i^o, v)$
    $\mathbf{B}_i^d = MeanSmoothening(\mathbf{B}_i^v, r)$
else
    Skip Frame
end if

In this pre-processing stage, each frame is processed to remove unwanted elements and noise from the point cloud. Initially, object localisation is performed on the RGB image $\mathbf{C}_i$ of the frame using the Mask-RCNNv2 mask generation model developed by Li et al.[148]. Pre-trained weights from the Microsoft COCO dataset[153] are utilised for this task, without any fine-tuning on the specific objects under consideration. The Mask-RCNNv2 model produces a list of detection scores and masks (bb) for the objects in the frame, although no classifications

are utilised in this process. From these predictions, mask instances for each object are created.

To select the intended object, the mask instances between two consecutive frames are examined. If an object is being manipulated, the corresponding mask is retained. However, if no object is detected in the frame, that frame is skipped, and the process continues with the next set of frames. When the intended object is detected in $\mathbf{C}_i$, the object is cropped, and all other information is removed from the depth map, resulting in $\mathbf{D}i^c$.

This procedure is repeated for all frames. Once the region corresponding to the intended object is identified, the object depth map is converted to a 3D point cloud $\mathbf{B}_i$ using the depth camera's intrinsic parameters $\mathbf{c}_K^i$. The point cloud $\mathbf{B}_i$ is then further processed to remove outliers and noise through statistical noise removal, producing $\mathbf{B}_i^o$. This processed point cloud is subsequently downsampled using a voxel size v. The downsampled point cloud $\mathbf{B}i^v$ is then passed through a mean smoothness filter with a radius r to eliminate sensor noise. The smoothened point cloud $\mathbf{B}i^d$ is finally passed to the classifier module.

The pre-processing steps are summarised in Algorithm 3. It should be noted that any other object localisation method [28, 133, 268, 263] could also be employed in place of Mask-RCNNv2 to obtain the object mask. Additionally, the learning methods are solely used for mask generation.

5.4.3 Classifier

The corresponding filtered point clouds of the pre-processed frames i and $i+k$ are utilised by this module to classify the motion as belonging to one of three categories: *i.e.*, NM, RM and AM. This classification is performed for all frame sets within the given video, a process henceforth referred to as ***in-frame classification***. The proposed in-frame classification method is summarised in Algorithm 4.

In this method, the point clouds $\mathbf{B}_i^d$ and $\mathbf{B}_{i+k}^d$ are first divided into voxel grids $\mathbf{V}$ with a voxel size x. This division results in v voxel grids containing points. For each voxel grid $\mathbf{V}_m, m \in M$, the points from $\mathbf{B}_i^d$ and $\mathbf{B}_{i+k}^d$ within the voxel grid $\mathbf{V}_m$ are identified as $\mathbf{B}_i^{d,m}$ and $\mathbf{B}_{i+k}^{d,m}$, respectively.

Assuming that the object's motion between two frames is minimal, registration based on the ICP algorithm [25] is performed between $\mathbf{B}_{i+k}^{d,m}$ and $\mathbf{B}_{i}^{d,m}$. Given the assumption that the points within both voxel grids are close, the initial transformation for ICP is provided as the identity matrix $\mathbf{I}$. This registration process results in a homogeneous transformation matrix $\mathbf{P}_m$, comprising rotation $\mathbf{R}_m$ and translation $\mathbf{t}_m$ components for the vth voxel grid. The rotation matrix is subsequently converted to its quaternion form $\mathbf{q}_m$ for further processing.

For each $m \in M$, $\mathbf{q}_m$ and $\mathbf{t}_m$ are quantised using thresholds $\overline{q}$ and $\overline{t}$ to mitigate noise errors in the registration process. These quantised values are then concatenated and hashed into a hash table $\mathcal{H}$, where the respective voxel grids are stored under each key. Due to the potential articulated nature of the objects, it is observed that some portions of an object may become occluded or move out of the frame between consecutive frames (e.g., the top portion of a box may become occluded when the box is opened). To address this issue, voxel grids that do not have any corresponding match during ICP are skipped in the proposed method. Additionally, registrations with low confidence scores are also excluded. If more than α% of the voxel grids are successfully matched, the algorithm proceeds to the classification step; otherwise, those frames are considered unreliable, and the process moves to the next set of frames.

Utilising the prepared hash table $\mathcal{H}$, in-frame classification is performed according to the following rules:

- If the number of keys in $\mathcal{H}$ is greater than 1, the motion is identified as articulated and classified as AM.
- If the number of keys in $\mathcal{H}$ equals 1, and the key corresponds to a rotation or translation, the motion is classified as RM.
- If the number of keys in $\mathcal{H}$ equals 1 and the key represents no motion, *i.e.*, $\mathbf{q} \in 0^{1\times4}$ and $\mathbf{t} \in 0^{1\times3}$, the motion is classified as NM.

The in-frame classification method is illustrated in Figure 5.2. All in-frame classifications are stored in a list $\boldsymbol{\rho}$ to determine the overall classification of the object. A moving average filter with a window size of w is then applied to $\boldsymbol{\rho}$. After filtering, if any of the elements in the list

correspond to AM, the object is classified as articulated. If only RM motion is present in the list, without any AM motion, the object is classified as rigid. Finally, if only NM motion is present, the classification remains nondeterministic due to the absence of motion in the provided video.

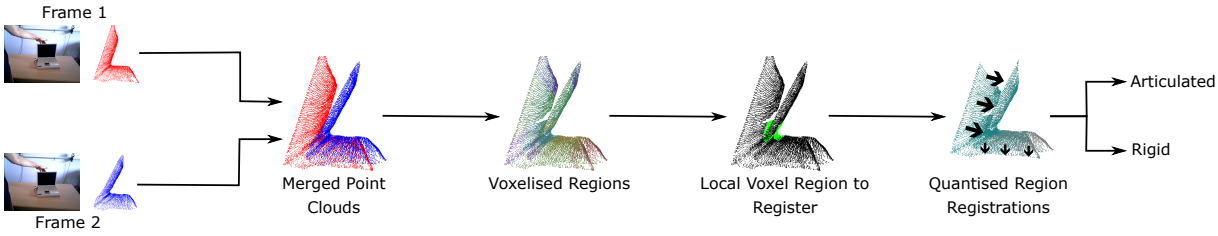

Figure 5.2: The in-frame classification process for the articulated object classification method.

From Algorithm 4, it can be observed that the proposed method relies heavily on the type of motion in the local regions of $\mathbf{B}_i^d$. Consequently, if significant noise or background elements are present in $\mathbf{B}_i^d$, the registration of these elements will indicate no motion, as they are assumed to be stationary. In the case of rigid objects, the presence of such background elements will generate two hash keys, potentially leading to the erroneous classification of the object as articulated.

One potential solution to mitigate this issue would be to exclude the NM key from consideration in the hash map. However, this approach presents a challenge when articulated objects are present in the frame. Specifically, if some components of an articulated object remain static throughout the entire video sequence, this static component would also be excluded, which could result in the mis-classification of the object as rigid.

To address this problem, the intended object is masked during the pre-processing stage, as described in the previous subsection. By doing so, only the point cloud information of the intended object is considered in the proposed classification algorithm, ensuring that background elements and static components of articulated objects do not interfere with the classification process.

5.5 Experimental results

Experiments were conducted on two benchmark datasets from the literature to demonstrate the capability of the proposed method in classifying objects as either articulated or rigid. The

Algorithm 4 Articulated and rigid object classification using point clouds.

Require: source point cloud $\mathbf{B}_i^d$, target point cloud $\mathbf{B}_{i+k}^d$, voxel size x, quaternion quantisation $\overline{q}$, translation quantisation $\overline{t}$, moving window size w

Ensure: In-frame object class ρ_i

$\mathbf{V} = Voxelise(\mathbf{B}_i^d, \mathbf{B}_{i+k}^d, x)$
$M = |\mathbf{V}|$
$m = 0$
$\mathcal{H} = \{\}$
repeat
  $\mathbf{B}_i^{d,m} = SelectVoxelPoints(\mathbf{B}_i^d)$ ▷ Local voxel region selection
  $\mathbf{B}_{i+k}^{d,m} = SelectVoxelPoints(\mathbf{B}_{i+k}^d)$
  $\mathbf{P}_m = ICPRegistration(\mathbf{B}_i^{d,m}, \mathbf{B}_{i+k}^{d,m})$
  $\mathbf{R}_m = \mathbf{P}_m[:3, :3]$
  $\mathbf{t}_m = \mathbf{P}_m[:3, 3]$
  $\mathbf{q}_m = ToQuaternion(\mathbf{R}_m)$
  $\tilde{\mathbf{q}}_m = \lfloor \frac{\mathbf{q}_m}{\overline{q}} \rfloor * \overline{q}$ ▷ Quantisation operation
  $\tilde{\mathbf{t}}_m = \lfloor \frac{\mathbf{t}_m}{\overline{t}} \rfloor * \overline{t}$
  if $[\tilde{\mathbf{q}}_m | \tilde{\mathbf{t}}_m] \in \mathcal{H}$ **then** ▷ Hash table is generated with unique keys by clustering
    $\mathcal{H}[\tilde{\mathbf{q}}_m | \tilde{\mathbf{t}}_m] = [\mathcal{H}[\tilde{\mathbf{q}}_m | \tilde{\mathbf{t}}_m]; \mathbf{V}_m]$
  else
    $\mathcal{H}[\tilde{\mathbf{q}}_m | \tilde{\mathbf{t}}_m] = [\mathbf{V}_m]$
  end if
  $m = m + 1$
until $m \leq M$
$K = |\mathcal{H}|$
if $K \geq 2$ **then**
  $\rho_i = AM$
else if $K = 1$ **then**
  $\rho_i = RM$
else
  $\rho_i = NM$
end if

performance was evaluated based on the mean accuracy score of the classification. Additionally, qualitative results were presented to illustrate the articulation regions detected by the proposed method from the given observations.

5.5.1 Dataset description

At the time of this work, no interaction-based hybrid datasets containing both rigid and articulated objects were available in the literature. To effectively evaluate the performance of the proposed method in such hybrid scenarios, it was tested using the RBO [169] and YCB video datasets [280]. The RBO dataset provides a collection of interaction videos and corresponding point clouds for multiple articulated objects. These interaction videos were captured in a controlled environment with objects in motion. In literature, no interaction-based datasets with point clouds for rigid objects were available. However, to assess the performance of the proposed method for rigid objects, the YCB video dataset was utilised, which provides multi-object video sequences along with point clouds. From the RBO dataset, interaction sequences for four articulated objects—a globe, a laptop, a cabinet, and a microwave—were considered. In contrast, four rigid objects—a banana, a gelatin box, a mustard bottle, and a power drill—were selected from the YCB video dataset for the experiments. For each object, at least 10 video sequences, with a maximum of 200 frames per sequence, were used for experimentation.

5.5.2 Experimental setup

As discussed in Section 5.4, the proposed method relies on multiple parameters during the pre-processing and classification steps. The values of these parameters were determined through multiple trials and based on the statistical properties of the data. For downsampling and smoothing, the values of parameters such as v and r were determined by analysing the properties of the point clouds. The downsample voxel size v was calculated as $\frac{1}{20}$ of the object's diameter within the scene, and r was set to $5 \times v$. The statistical outlier removal parameter s was set to 0.5, with a nearby points threshold of 0.1 times the total number of points in the object point cloud. The voxel size x in the classifier was set to $\frac{1}{5}$ of the maximum distance between the points in the

point cloud. Other parameters, such as $\overline{q}$, $\bar{t}$, and k, were determined empirically after multiple trials on a subset of the considered data. In this study, the values $\overline{q} = 0.1$, $\bar{t} = 0.1$, and $k = 5$ were utilised for the experiments.

The experimental setup was meticulously designed to ensure that the chosen parameters optimally reflect the characteristics of the datasets, thereby enhancing the accuracy and reliability of the proposed method in classifying articulated and rigid objects.

5.5.3 Performance analysis

The performance of the developed method is thoroughly evaluated through both quantitative and qualitative analyses. For the quantitative analysis, two key metrics are employed: classification accuracy and average classification probability. Classification accuracy is assessed on a per-object basis, calculated as the ratio of the total number of correctly classified video sequences to the total number of video sequences processed for each object. This metric provides a direct measure of the method's ability to correctly identify whether an object is articulated or rigid across different sequences. The average classification probability is determined as the mean probability of each frame set being classified as AM or RM. This probability reflects the confidence level of the classifier in its decisions, offering insights into the robustness and reliability of the method across varying scenarios.

Qualitative analysis is conducted to complement the quantitative results by providing visual interpretations of the method's performance. In this analysis, the direction and magnitude of the transformations obtained from the registration between frames are displayed. These transformations are visualised using arrows, where the direction of each arrow corresponds to the direction of movement between frames, and the length of each arrow indicates the magnitude of the transformation. This visual representation allows for a more intuitive understanding of how the motion within the point clouds is tracked and classified, offering a clear depiction of how articulated and rigid objects are distinguished in practice.

Through this combination of quantitative and qualitative analyses, a comprehensive evaluation of the method's performance is provided, highlighting both the accuracy in classification

tasks and the ability to visually and effectively capture the motion dynamics of the objects under study.

Table 5.1: Performance analysis of the articulated classification method for RBO dataset.

Object	Accuracy %
Laptop	88.00
Globe	76.00
Microwave	100.00
Cabinet	100.00
Average	91.00

Table 5.2: Performance analysis of the articulated classification method for YCB dataset.

Object	Accuracy %
Power Drill	89.47
Gelatin Box	86.36
Banana	95.45
Mustard Bottle	91.30
Average	90.65

5.5.4 Accuracy analysis

The accuracy performance for both the RBO and YCB datasets is presented in Table 5.1 and Table 5.2. From the results in Table 5.1, it is observed that the proposed method successfully classified the Microwave and Cabinet objects as articulated with 100% accuracy. For the Laptop object, an accuracy of 88% was achieved. The decrease in accuracy for the Laptop is attributed to the presence of a mirror-like screen, which adversely affected the depth sensor, introducing significant noise and leading to registration failures. Additionally, the Globe object was classified with 76% accuracy, which is considered notably good given that colour information was not utilised in the registration process, and the symmetric nature of the Globe posed challenges. Overall, the proposed method demonstrated an accuracy of 91% in correctly classifying articulated objects within the RBO dataset.

For the rigid objects from the YCB dataset, the proposed method achieved a classification accuracy of 90.65%, as shown in Table 5.2. The Power Drill, Banana, and Mustard Bottle objects were classified with accuracies of 89.47%, 95.45%, and 91.30% respectively. The Gelatin Box was classified with an accuracy of 86.36%. The slightly lower accuracy for the Gelatin Box is attributed to its symmetric shape, which negatively impacted local registration accuracy.

These results demonstrate that the proposed method is capable of classifying objects with high accuracy, regardless of whether the objects are articulated or rigid. Importantly, this

Table 5.3: Per scene sequence classification probabilities of the articulated classification method for RBO dataset.

Object	Probability of rigid	**Probability of articulated**
Laptop	0.1472	0.8528
Globe	0.0041	0.9959
Microwave	0.0251	0.9749
Cabinet	0.0543	0.9457
Average	0.0577	0.9423

Table 5.4: Per scene sequence classification probabilities of the articulated classification method for YCB dataset.

Object	**Probability of rigid**	Probability of articulated
Power Drill	0.8744	0.1256
Gelatin Box	0.7380	0.2620
Banana	0.9847	0.0153
Mustard Bottle	0.8433	0.1567
Average	0.8601	0.1399

classification was achieved without the need for pre-existing object models, labelled data, or articulation constraints, highlighting the robustness and versatility of the proposed approach.

5.5.5 Probability analysis

Tables 5.3 and 5.4 present the probabilities of a set of frames being classified as either articulated or rigid. As discussed in Section 5.4, each frame set is classified into one of three categories: AM, RM, or NM, with the final classification being determined by the max-count filtered output.

In Table 5.3, the performance of the proposed method for objects from the RBO dataset is displayed. It can be observed that the probability of a set of frames being classified as AM is consistently higher than the probabilities for RM and NM. This indicates that the proposed method effectively recognises and classifies articulated objects with a high degree of confidence across multiple frames. Similarly, for rigid objects from the YCB dataset, Table 5.4 demonstrates that the probability of a set of frames being classified as RM is significantly greater than the probabilities for AM and NM. This observation confirms that the proposed method reliably identifies rigid objects, with the classification being supported by strong probability values.

This analysis indicates that the proposed algorithm is capable of identifying the type of objects

(a) RBO dataset

(b) YCB dataset

Figure 5.3: Local region registration results of the articulated classification method for RBO and YCB dataset with two consecutive frames.

with high probabilities, even when evaluating a single set of frames. The consistent assignment of higher probabilities to the correct classification category demonstrates the effectiveness and reliability of the proposed method in distinguishing between articulated and rigid objects.

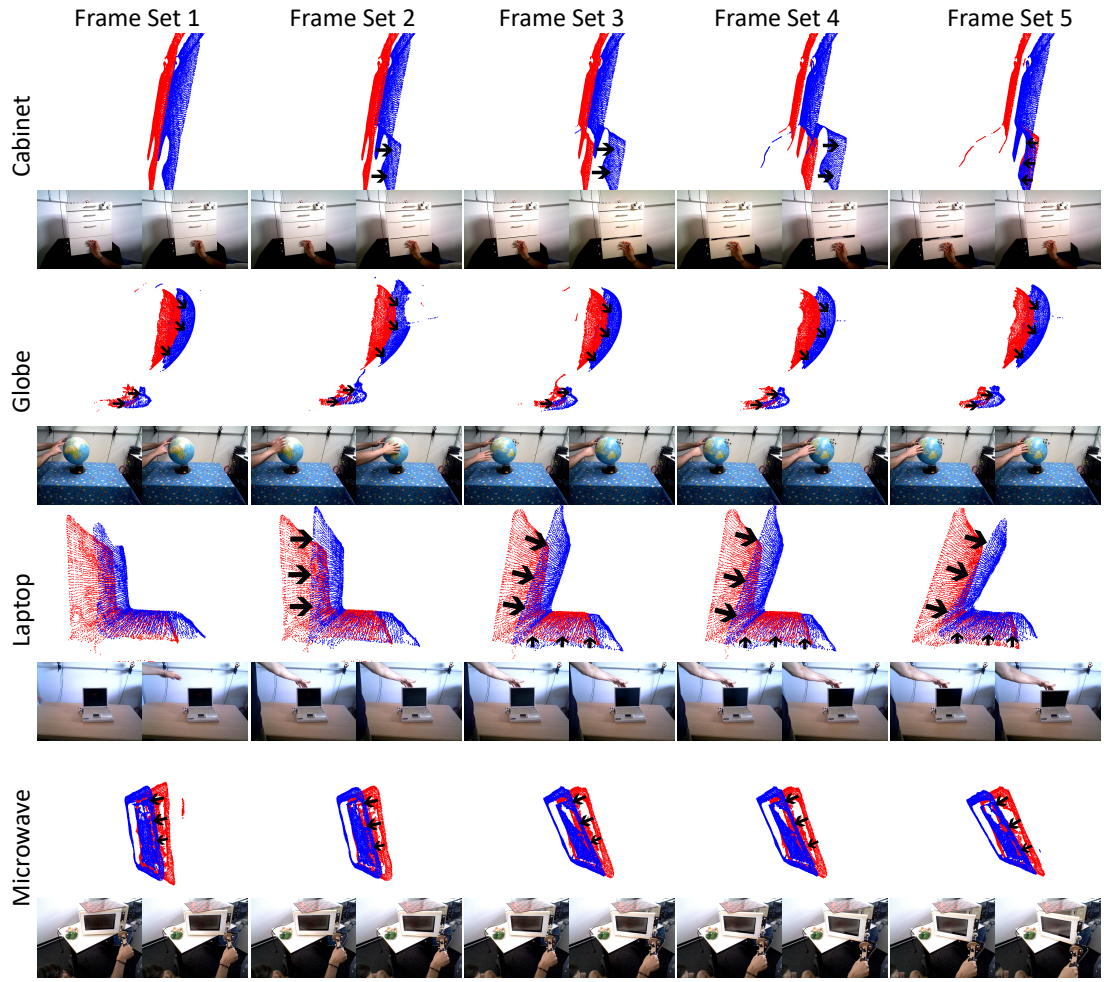

Figure 5.4: Local registration results of the articulated classification method for RBO dataset with 5 consecutive frames.

5.5.6 Qualitative analysis

The local registration results for the RBO and YCB datasets, along with the corresponding RGB images for two consecutive frames, are presented in Figures 5.3a and 5.3b. In Figure 5.3a, it can be observed that more than one unique registration is obtained between the frames, leading to the classification of these objects as articulated. This outcome demonstrates the effectiveness

of the proposed method in detecting and correctly classifying multiple distinct transformations occurring within different parts of the articulated objects. Similarly, in Figure 5.3b, it is observed

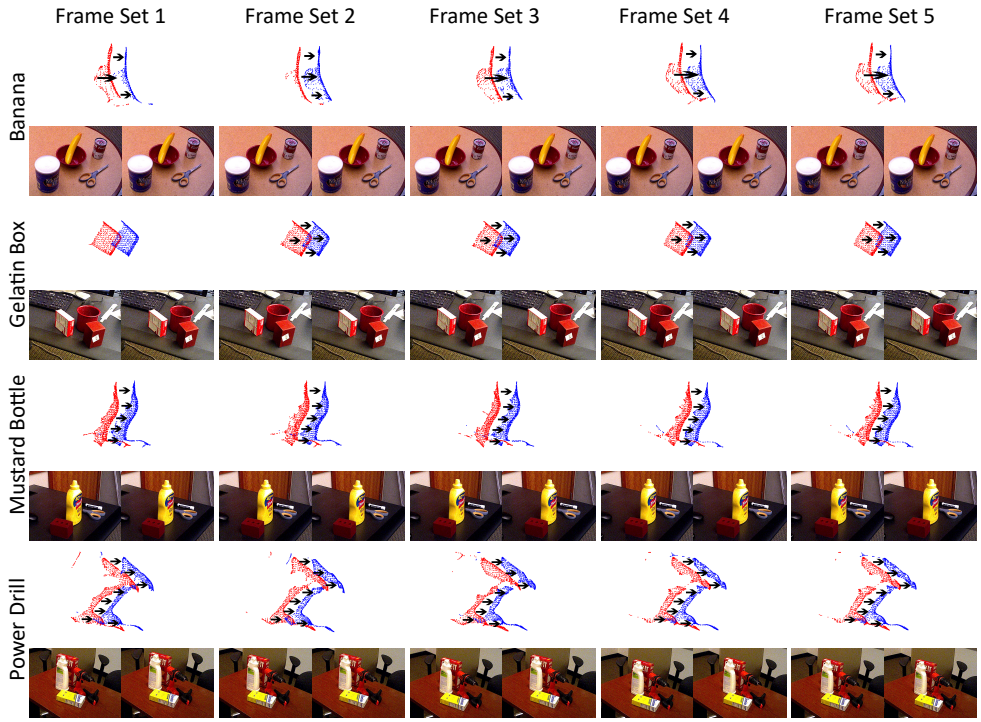

Figure 5.5: Local registration results of the articulated classification method for YCB dataset with 5 consecutive frames.

that a single unique transformation is consistently obtained across all local regions between the frames, resulting in the classification of these objects as rigid. This consistency in the registration across the entire object confirms the rigid nature of the objects, further validating the accuracy of the proposed method in distinguishing between rigid and articulated objects.

Figures 5.4 and 5.5 illustrate the local registrations between the voxel grids of five consecutive frames for the RBO and YCB datasets, respectively. These figures provide a more extended visual analysis, showcasing the continuity and reliability of the registration process across multiple frames. For the RBO dataset, the presence of multiple unique transformations across consecutive frames consistently supports the classification of the objects as articulated. In contrast, for the YCB dataset, the uniformity in the transformations across the frames consistently leads to the classification of the objects as rigid.

These results emphasise the robustness of the proposed method in accurately identifying

and classifying the motion characteristics of objects within both articulated and rigid categories, even when evaluated across multiple consecutive frames.

5.6 Summary

In this chapter, a registration-based local region-to-region mapping approach for articulated object classification was proposed. The classification of object articulation was formulated as a temporal movement detection method, where the point cloud of an object from two consecutive time frame observations was analysed to identify unique local motions between the frames. An object is classified as articulated if more than one unique motion is detected during the observations. If only a single motion is detected, the object is classified as either undergoing rigid motion or no motion.

The proposed method is characterised by its model-free nature, allowing it to be applied to a wide variety of articulated objects, making it more versatile than state-of-the-art methods in the literature. Furthermore, it does not require labelled data, enhancing its applicability in scenarios where annotated datasets are not available.

Experimental results using two publicly available benchmark datasets, consisting of articulated and rigid objects, demonstrated that the proposed method achieves high accuracy in classifying articulated objects. This performance underscores the method's effectiveness in diverse settings, especially in cases where other methods may struggle due to the complexity or variability of the objects being analysed. In the next chapter, visual servoing task for robotic application has been discussed, wherein the designed 3D edge features are utilised as visual features. The method is designed as a learning-based direct visual servoing approach where 3D edge features are directly utilised to control the robot.

Chapter 6

Learning-based visual servoing with 3D edges

6.1 Introduction

In various robotics applications, such as manufacturing, assembly, and grasping, it is essential for the robot to interact with an object at a specific pose. Achieving this precise interaction is particularly critical when tasks demand high accuracy and repeatability, as is often the case in industrial settings. However, in real-world environments, which are frequently unstructured and unpredictable, objects may be present in random and varying poses. These unstructured environments pose significant challenges for robots, as the variability in object positioning requires the robot to adapt continuously to align itself correctly with the object.

Consequently, before any interaction can take place, it is essential for the robot to adjust its pose to achieve the desired alignment with the object. This alignment is particularly challenging in dynamic environments, where objects may be randomly positioned or in motion. For example, if a robot is welding on a moving assembly line, such as a car body, the object remains in continuous motion. Similarly, in tasks like grasping and packing items from a moving conveyor belt, the robot must continuously adapt its pose to align with the moving objects and perform the task effectively. These dynamic conditions pose significant challenges for robots in executing

tasks with precision. Moreover, the processes must operate in real time to ensure the interaction is completed before the object is lost or moves out of reach.

To address these challenges, vision-guided motion control of robots, also called as Visual Servoing (VS) has been widely used in the literature as an effective method for ensuring that a robot can dynamically adjust and maintain its pose relative to an object. VS leverages visual information of the environment, such as images, depth maps, or point clouds [162]. This visual data acquired from sensors is used to regulate the robot's control system, enabling real-time adjustments to the robot's movements.

Technically, VS has been employed to accomplish a variety of robotic tasks, e.g., end-effector positioning, where precise placement of the robot's tool is required; adaptive robotic grasping, where objects must be securely held; tracking, where the robot follows a moving target; and manipulation, where complex interactions with objects are necessary [6, 7, 59, 5]. Essentially, VS can be conceptualised as an optimisation problem, with the primary objective being to minimise the error between the current visual features detected by the robot and the desired features corresponding to the target object or scene. By continuously reducing this visual feature error, the robot is guided towards the desired pose. This error, representing the discrepancy between the robot's current visual perception and the target state, serves as the critical input for the control loop. The controller plays a crucial role in ensuring that the robot's motion aligns with the computed velocity to minimise the feature error, thereby driving the robot towards the desired pose. The choice of controller directly impacts the performance of the servo loop. A well-designed controller must balance accuracy, robustness, and computational efficiency to ensure that the robot can successfully converge to the desired pose, even in the presence of environmental disturbances and sensor noise.

The literature categorises robot control strategies into three primary types: classic control techniques, modern control techniques, and intelligent control techniques [163].

Classic control techniques: classic methods, like Proportional-Integral-Derivative (PID) control [250] and Linear Quadratic Regulators (LQR) control [232], are favoured for their simplicity and ease of implementation. However, they may struggle in non-linear or

dynamic environments, where complex disturbances are common.

Modern control techniques: modern techniques, such as Model Predictive Control (MPC) [8] and robust control [152], address the limitations of classic methods by handling multi-variable systems and ensuring stability despite uncertainties. These methods are powerful but often computationally demanding.

Intelligent control techniques: intelligent control methods, including fuzzy logic, deep learning-based control, and reinforcement learning [194], adapt to variable and uncertain environments. They offer flexibility and adaptability but typically require significant computational resources and training data.

Overall, visual servoing offers a robust framework for addressing the inherent uncertainties and dynamics of real-world environments, making it a crucial component in the development of autonomous and adaptable robotic systems.

6.1.1 Related work

VS can be primarily categorised into Image-Based Visual Servoing (IBVS) and Position-Based Visual Servoing (PBVS). In IBVS, features extracted from the captured image are utilised to control the robot's motion [60, 96, 76, 37, 151, 299, 137, 216]. These features may encompass entire images, hand-crafted descriptors such as SIFT and HOG, or learned features derived from methods such as Principal Component Analysis (PCA) and autoencoders. IBVS operates by minimising the error between the current and desired image features, thereby driving the robot's movements towards the target pose. However, despite its ability to achieve high positioning accuracy, in case of 3D tasks, IBVS-based systems are often challenged using depth information and the complexities associated with a strongly coupled image Jacobian matrix. These challenges can lead to suboptimal performance in certain scenarios, particularly in environments with significant depth variations or occlusions [37].

On the other hand, PBVS involves the estimation of the 6-DoF pose of the object using point information, which is then employed to perform the servoing task [219, 257, 275]. By

incorporating 3D information, PBVS offers a more generalised framework that can effectively handle a wider range of tasks, especially those requiring precise depth and pose control. However, the accuracy of PBVS is highly dependent on the precision of the object pose estimation, which can be affected by factors such as sensor noise and calibration errors. The reliance on accurate 3D models and the potential for errors in pose estimation represent significant limitations in PBVS, particularly in dynamic or unstructured environments [164].

A recent development in the field is Direct Visual Servoing (DVS), which has been proposed as an alternative to the traditional IBVS and PBVS approaches. DVS directly utilises the entire image, without the need for explicit feature extraction or pose estimation, to control the robot's motion [236, 266]. This method avoids the complexities associated with feature extraction and matching, as well as the need for accurate 3D models, making it particularly suitable for applications where these tasks are challenging or infeasible. DVS approaches have demonstrated robustness in scenarios with significant visual noise or in cases where objects lack distinct features. Moreover, DVS can provide a more direct mapping between the image space and the control commands, potentially leading to faster and more efficient servoing [192]. Nevertheless, these methods are highly sensitive to image noise and variations in lighting conditions. Further, their performance is limited in case of dense or heavy information, which is often the case while using 3D cameras.

To address the inherent limitations of both IBVS and PBVS, hybrid visual servoing approaches have been proposed in the literature [80, 40, 287]. Hybrid visual servoing combines elements of both IBVS and PBVS to leverage their respective strengths while mitigating their weaknesses. Typically, these approaches utilise image features to maintain robustness against calibration errors and partial occlusions, as seen in IBVS, while also incorporating 3D pose estimation to ensure accurate depth and pose control, akin to PBVS. This integration of methods allows for more reliable and accurate control in dynamic and unstructured environments, providing a balanced solution where pure IBVS or PBVS might fall short. Hybrid VS methods have demonstrated significant improvements in handling complex tasks that involve both large position displacements and precise alignment requirements [191].

In addition to IBVS, PBVS, and DVS, a fourth category of VS has been explored in recent literature, known as Learning-Based Visual Servoing (LBVS). LBVS methods leverage advances in deep learning to perform the servoing task, utilising image or point-based inputs to predict features for matching or velocities directly [186, 75, 247, 277]. These approaches have shown promise in scenarios where traditional VS methods struggle, particularly in handling complex, high-dimensional feature spaces or learning robust feature representations from large datasets. However, LBVS also introduces challenges, such as the need for extensive training data and the difficulty in interpreting the learned features, which may limit its applicability in certain contexts [235].

6.1.2 Organisation

The rest of the chapter is organised as follows. The main contribution of this chapter are discussed in Section 6.2. In Section 6.3, the auto-encoder network and VS control design are presented. Experimental results and problem stability are discussed in Section 6.4. Finally, the proposed method is concluded in Section 6.5.

6.2 Contribution

In this chapter, a visual servoing approach is proposed, which utilises 3D information as visual cues. As discussed in Section 2.1, various types of features such as PPF, FPFH, and SHOT can be extracted from 3D Point Clouds. Among these, 3D edge point clouds are particularly interesting due to their texture independence and sparsity in 3D space, while preserving the geometrical properties of objects, as discussed in Chapter 3.4. One method for utilising 3D edges is to consider the entire point set as an input feature [305]. However, in unstructured 3D settings, the feature representation can be affected by the order and number of 3D points, which may degrade VS performance. To address this, a latent space representation of the 3D edge point set is learned using autoencoders, which is a popular method to extract robust information from raw data [55, 22]. A hierarchical autoencoder network for 3D points is designed to extract

features that are robust to the unstructured and irregular properties of point clouds.

Various deep learning networks have been proposed in the literature for feature extraction from point clouds [61, 213, 62, 303, 296, 212], as discussed in Section 2.1. In the proposed work, deep autoencoder network (inspired from PointNet++ [213]) has been designed to learn the latent features from 3D edge clouds. PointNet++ is utilised due to its hierarchical nature, which enable to capture both local and global relationships within a given cloud. This latent feature enhances the convergence area for VS optimisation and also improves robustness against external noise [186].

There are a few studies in the literature that use latent space representation for VS tasks [247, 75, 186], but they primarily focus on image embeddings. For example, [75] combines image and pose details to learn a latent space representation for visual servoing on images, and [247] proposes a VS system encoding tactile sensor information. However, these methods suffer from lack of spatial information for the VS task, which impact the system performance in case of occlusions. Further, they rely on texture information in the images to extract meaningful information, which may not always be available in many industrial settings. This work addresses these limitations by utilising 3D edge cloud data to extract latent space features.

Alongside the latent space features, the centroids of the captured object scene cloud are utilised as direct features for the VS task. The centroid provides a concise and robust representation of the object's spatial distribution, which can be used within the robot control loop. This approach ensures that even in cluttered or partially occluded environments, where individual feature points may be unreliable or difficult to track, the overall spatial structure of the object remains effectively represented in the control process. As the robot moves, the centroid of the point cloud dynamically shifts, offering continuous feedback for real-time adjustment of the robot's pose relative to the object.

Two types of errors, namely deep feature-based and centroid-based, are calculated by comparing the current information with the desired information to determine the necessary camera spatial velocity. A hybrid controller [36] has been designed, wherein the deep feature error is utilised to compute the angular velocity, while the centroid-based error is used to calculate

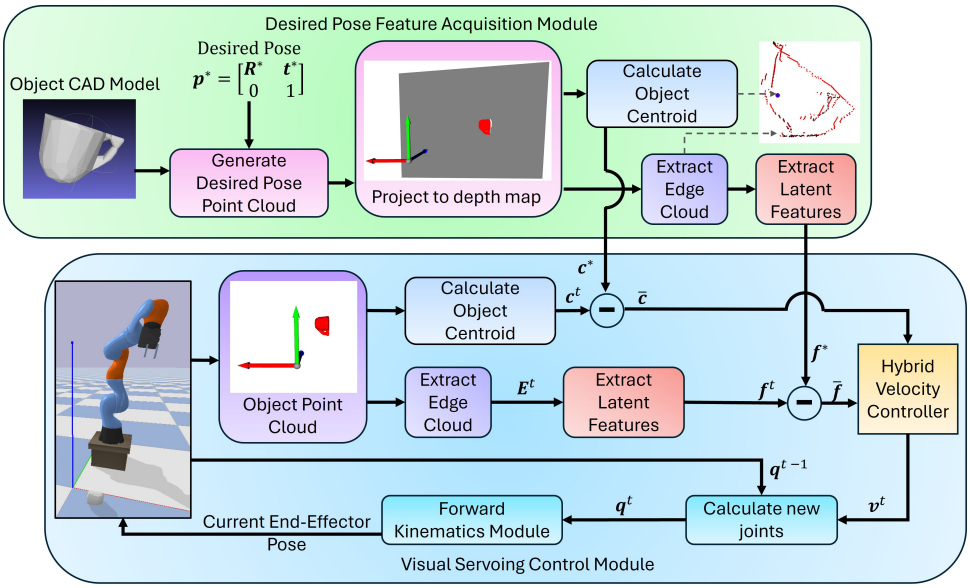

Figure 6.1: Control flow of the proposed 3D edge-based visual servoing method.

the translational velocity. This computed velocity is subsequently applied to the robot using a classical control strategy [59], enabling the robot arm to move to the new position.

The overall closed-loop pipeline for the proposed VS method is illustrated in Fig. 6.1. This pipeline integrates the computed camera velocity with the selected control strategy to actuate the robot's movements, thereby iteratively minimising the feature error. The proposed method has been validated through both simulation and real-robot experiments across various complex scenarios. The results have demonstrated that the proposed servo control achieves smooth convergence with fewer iterations, indicating its effectiveness and efficiency in dynamic and unstructured environments.

6.3 Methodology

A VS task is typically formulated as an optimisation problem aimed at minimising the difference between the features $\boldsymbol{F}^t$ and $\boldsymbol{F}^*$, which are respectively extracted from the current object edge cloud $\mathbf{E}^t$ at time t and the desired object edge cloud $\mathbf{E}^*$ corresponding to the desired object pose $\boldsymbol{p}^*$. The feature error, denoted as $\overline{\boldsymbol{F}}$, can therefore be expressed as

$$\overline{\boldsymbol{F}} = \boldsymbol{F}^t - \boldsymbol{F}^*. \tag{6.1}$$

The optimisation problem is thus defined as finding the feature set $\boldsymbol{f}$ that minimises the error, formally written as:

$$\mathcal{L} = \min_{\boldsymbol{F}} \|\boldsymbol{F}^t - \boldsymbol{F}^*\|. \tag{6.2}$$

To solve this optimisation problem, the robot must be moved in a direction that reduces the feature error. This is accomplished by estimating the robot's velocity $\boldsymbol{v}$ based on the feature error $\overline{\boldsymbol{F}}$. To achieve this, a relationship must be established between the changes in features and the corresponding robot motion. In the literature, this transformation is commonly referred to as the interaction matrix $\mathbf{L}$ [38]. The interaction matrix $\mathbf{L_E}$, also called as image Jacobian matrix, contains the partial derivatives of the features with respect to the spatial velocity. Thus, the change in features with respect to robot motion can be expressed as:

$$\overline{\boldsymbol{F}} = \mathbf{L}_{\boldsymbol{F}}\boldsymbol{v}, \tag{6.3}$$

where $\boldsymbol{v}$ denotes the robot's spatial velocity.

By substituting (6.3) into the optimisation problem (6.2) and solving for the velocity $\boldsymbol{v}$, the following expression is obtained:

$$\boldsymbol{v} = -\lambda \mathbf{L}_{\boldsymbol{F}}^{\dagger}\overline{\boldsymbol{F}}, \tag{6.4}$$

where λ represents a gain parameter, and $\mathbf{L}_{\boldsymbol{F}}^{\dagger}$ denotes the pseudo-inverse of the interaction matrix $\mathbf{L}_{\boldsymbol{F}}$.

This equation describes the robot's velocity as a function of the feature error and the interaction matrix, thereby guiding the robot to minimise the discrepancy between the current and desired object poses during the VS task. From the discussion, it can be inferred that the overall VS module comprises three stages within a closed-loop system. In the first stage, visual information is captured and processed from the scene. Specifically, in this work, 3D point cloud data are captured from the scene, from which the ground plane is segmented out resulting in just object cloud. This segmented object point cloud is then used to derive the 3D edge cloud $\mathbf{E}$, from which a feature vector $\boldsymbol{f}$ is extracted. Additionally, the centroid $\boldsymbol{c}$ is calculated from the object point cloud. In the second stage, the interaction matrices $\mathbf{L}_{\boldsymbol{f}}$ and $\mathbf{L}_{\boldsymbol{c}}$ are computed for the

extracted features $\boldsymbol{f}$ and the 3D edge cloud $\mathbf{E}$, respectively. These interaction matrices provide the necessary relationship between the feature changes and the corresponding robot motion.

Finally, in the third stage, the feature error and the interaction matrices are utilised to control the robot by calculating the updated robot joint configurations. These joint configurations are then sent to the robot, enabling it to move to the new position. Each of these stages is discussed in detail below.

6.3.1 Feature extraction

In the presented work, two types of features have been extracted from the captured 3D point cloud $\mathbf{P}$: the object centroid $\boldsymbol{c}$ and a K-dimensional latent space feature representation $\boldsymbol{f} \in \mathcal{R}^K$. Initially, the object point cloud $\mathbf{P}_o$ is segmented from the captured 3D point cloud, effectively filtering out the background, ground plane, and any other extraneous objects. Following segmentation, the centroid of the object point cloud is calculated using the equation:

$$\boldsymbol{c} = (c_x, c_y, c_z) = \left(\frac{1}{N} \sum_{i=1}^{N} x_i, \frac{1}{N} \sum_{i=1}^{N} y_i, \frac{1}{N} \sum_{i=1}^{N} z_i \right) \tag{6.5}$$

where $\boldsymbol{x} = [x_1, x_2, \cdots, x_N]^T$, $\boldsymbol{y} = [y_1, y_2, \cdots, y_N]^T$, $\boldsymbol{z} = [z_1, z_2, \cdots, z_N]^T$, and $\mathbf{P}_o = [\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]^T$.

The latent space representation is extracted by employing a deep encoder-decoder network, denoted as $\mathcal{H}$. As discussed in Chapter 3, 3D edges provide a sparse representation of the object that is robust against texture variations and occlusion while effectively capturing the geometrical properties of the object. Therefore, in this work, the 3D edge point cloud is utilised as input to the feature extractor. The 3D edges are extracted using the method developed in Chapter 3. It is worth noting that any other 3D edge extraction techniques can be used as alternates.

A modified PointNet++ [213] based feature extractor has been utilised in this work to obtain the latent space features. The network architecture used is a point segmentation-based model, as illustrated in Figure 4.1, where the output corresponds to the input. To effectively handle the sparse nature of the edge point clouds, the same modifications as discussed in Section 4.4.2 have been applied. These modifications include:

- Random selection of q anchor points, as opposed to using Farthest Point Sampling (FPS) [71].
- Random selection of points from the entire edge cloud to compute the anchor point features.

The network consists of three encoding layers, with the number of selected keypoints reduced by half at each successive layer. Following the encoding layers, a grouping layer is employed to aggregate features from all points in the previous layer, yielding a 512-dimensional latent feature vector. To ensure that the network's feature learning is independent of large variations in translation, the edge clouds are centred before being processed by the network. For training, the decoder is structured as an interpolator, where both points and features are interpolated from the previous layer and concatenated with the learned features from the corresponding encoder layers. The decoder is expected to produce a point cloud $\mathbf{D}$ as its output.

The overall learning process is conducted by minimising the Chamfer Distance between the input point cloud $\mathbf{E}$ and the predicted point cloud $\mathbf{D}$, which is defined as:

$$Q(\mathbf{E}, \mathbf{D}) = \sum_{x \in \mathbf{E}} \min_{y \in \mathbf{D}} ||x - y||_2^2 + \sum_{y \in \mathbf{D}} \min_{x \in \mathbf{E}} ||x - y||_2^2. \tag{6.6}$$

These extracted features, including the centroid and latent space representation, form the basis for driving the VS process.

6.3.2 Interaction matrix design

As stated earlier, the interaction matrix is a fundamental component in visual servoing, as it establishes the relationship between variations in features and the corresponding robot motion. This matrix is typically calculated by determining the partial derivatives of the features with respect to the robot's spatial coordinates. Mathematically, the interaction matrix $\mathbf{L}_{\boldsymbol{F}}$ for a feature vector $\boldsymbol{F}$ is expressed as:

$$\mathbf{L}_{\boldsymbol{F}} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{p}}, \tag{6.7}$$

where $\boldsymbol{p} = [x, y, z, \theta_x, \theta_y, \theta_z]^T$ represents the robot's pose, including both translational (x, y, z) and rotational $(\theta_x, \theta_y, \theta_z)$ components. These partial derivatives quantify how small changes in the robot's position and orientation affect the observed features, thus enabling the translation of feature errors into robot velocities.

The interaction matrix is often derived analytically for simple features such as points, lines, or centroids [35]. For instance, in the case of a single image point feature, the interaction matrix is represented as:

$$\mathbf{L}_{\boldsymbol{F}} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{u}{Z} & \frac{uv}{f} & -(f + \frac{u^2}{f}) & v \\ 0 & -\frac{1}{Z} & \frac{v}{Z} & (f + \frac{v^2}{f}) & -\frac{uv}{f} & -u \end{bmatrix}, \tag{6.8}$$

where Z denotes the depth of the point in the camera frame, f is the focal length, and (u, v) are the coordinates of the point in the image plane.

However, for more complex features, such as those derived from a 3D edge cloud or latent space representations, numerical methods or machine learning approaches can be used to estimate the interaction matrix [36]. In such cases, the matrix may be approximated using finite differences:

$$\mathbf{L}_{\boldsymbol{F}} \approx \frac{\boldsymbol{F}(\boldsymbol{p} + \Delta\boldsymbol{p}) - \boldsymbol{F}(\boldsymbol{p})}{\Delta\boldsymbol{p}}, \tag{6.9}$$

or by training a model to predict the Jacobian based on empirical data [186]. The accuracy of the interaction matrix is crucial, as it directly influences the precision of the robot's movements during the VS task. In this method, where 3D point information is directly utilised, specifically the object centroid $\boldsymbol{c}$, the interaction matrix for a 3D point with respect to spatial velocity is defined as [36]:

$$\mathbf{L}_{\boldsymbol{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 & c_z & -c_y \\ 0 & 1 & 0 & -c_z & 0 & c_x \\ 0 & 0 & 1 & c_y & -c_x & 0 \end{bmatrix}. \tag{6.10}$$

For the latent feature $\boldsymbol{f}$, the derivation of the interaction matrix is inspired by [186], where

a detailed analysis is provided. Hence, the feature Jacobian is calculated as:

$$\begin{aligned}\mathbf{L}_f &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}} \\ \mathbf{L}_f &= \frac{\partial \mathcal{H}(\mathbf{E})}{\partial \boldsymbol{v}} \\ \mathbf{L}_f &= \frac{\partial \mathcal{H}(\mathbf{E})}{\partial \mathbf{E}} \frac{\mathbf{E}}{\partial \boldsymbol{v}} \\ \mathbf{L}_f &= \mathbf{L}_{\mathcal{H}} \mathbf{L}_{\mathbf{E}},\end{aligned} \tag{6.11}$$

where $\mathbf{L}_{\mathbf{E}}$ is the interaction matrix that defines the relationship between edge points and spatial velocity, and is defined as:

$$\mathbf{L}_{\mathbf{E}} = [\mathbf{L}_{e^1}, \mathbf{L}_{e^2}, \cdots, \mathbf{L}_{e^N}]^T, \tag{6.12}$$

where $\mathbf{L}_{e^i}$ for $i \in \{1, 2, \cdots, N\}$ represents the rate of change of a point $\tilde{e}^i$ with respect to spatial velocity, and is derived similarly to (6.10) as follows:

$$\mathbf{L}_{e^i} = \begin{bmatrix} 1 & 0 & 0 & 0 & e^i_z & -e^i_y \\ 0 & 1 & 0 & -e^i_z & 0 & e^i_x \\ 0 & 0 & 1 & e^i_y & -e^i_x & 0 \end{bmatrix}. \tag{6.13}$$

The matrix $\mathbf{L}_{\mathcal{H}}$ represents the interaction matrix containing the rate of change of features with respect to the change in edge points. In the considered network, set-abstraction (denoted as $\mathbf{S}()$) and feature extractor (denoted as $\mathbf{W}()$) modules are used as three consecutive sequential layers, followed by a linear layer $\mathbf{A}^f$ to obtain the feature vector. Let the superscript $l1, l2, l3$ represent each set of modules, respectively. From (6.11), $\mathbf{L}_{\mathcal{H}}$ can be expressed as:

$$\begin{aligned}\mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathcal{H}(\mathbf{E})}{\partial \mathbf{E}} \\ \mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(\mathbf{W}^{l1}(\mathbf{S}^{l1}(\mathbf{E})))}{\partial \mathbf{E}} \\ \mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(\mathbf{W}^{l1}(\mathbf{E}'_{k1,m}))}{\partial \mathbf{E}},\end{aligned} \tag{6.14}$$

where $\mathbf{E}'_{k1,m}$ are the selected $k1$ anchor points with m points randomly selected to calculate anchor point features, performed by the set-abstraction layer $\mathbf{S^{l1}}$.

Each feature extractor $\mathbf{W}$ consists of three 2D convolutions (W_1, W_2, W_3) with kernel size $(1, 1)$ and ReLU activation $a()$. Let the weight matrix for each convolution operation be $\mathbf{A}_1^{l1}, \mathbf{A}_2^{l1}, \mathbf{A}_3^{l1}$ and the biases be $\mathbf{b}_1^{l1}, \mathbf{b}_2^{l1}, \mathbf{b}_3^{l1}$. Hence, (6.14) can be extended as:

$$
\begin{aligned}
\mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(\mathbf{W}^{l1}(\mathbf{E}'_{k1,m}))}{\partial \mathbf{E}} \\
\mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(a(W_3^{l1}(a(W_2^{l1}(a(W_1^{l1}(\mathbf{E}'_{k1,m})))))))}{\partial \mathbf{E}} \\
\mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(a(W_3^{l1}(a(W_2^{l1}(a(\mathbf{A}_1^{l1}\mathbf{E}'_{k1,m} + \mathbf{b}_1^{l1})))))))}{\partial \mathbf{E}} \\
\mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(a(W_3^{l1}(a(W_2^{l1}(a(\mathbf{A}_1^{l1}\mathbf{E}'_{k1,m} + \mathbf{b}_1^{l1})))))))}{\partial (\mathbf{A}_1^{l1}\mathbf{E}'_{k1,m} + \mathbf{b}_1^{l1})} \frac{\partial (\mathbf{A}_1^{l1}\mathbf{E}'_{k1,m} + \mathbf{b}_1^{l1})}{\partial \mathbf{E}} \\
\mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(a(W_3^{l1}(a(W_2^{l1}(\mathbf{O}_1^{l1})))))}{\partial (\mathbf{O}_1^{l1})} \frac{a(\mathbf{A}_1^{l1}\mathbf{E}'_{k1,m} + \mathbf{b}_1^{l1})}{\partial (\mathbf{A}_1^{l1}\mathbf{E}'_{k1,m} + \mathbf{b}_1^{l1})} \mathbf{A}_1^{l1} \\
\mathbf{L}_{\mathcal{H}} &= \frac{\partial \mathbf{L2}(a(W_3^{l1}(a(W_2^{l1}(\mathbf{O}_1^{l1})))))}{\partial (\mathbf{O}_1^{l1})} a(\mathbf{A}_1^{l1}),
\end{aligned}
\tag{6.15}
$$

where $\mathbf{O}_i^{lj}$ represents the output of the ith layer in the ljth feature extraction module. Following the same steps for the remaining two layers in the feature extraction module, the network interaction matrix for layer $l1$ is obtained as:

$$
\mathbf{L}_{\mathcal{H}} = \frac{\partial \mathbf{L2}([\mathbf{O}^{l1}|\mathbf{E}'_{k2,m}])}{\partial ([\mathbf{O}^{l1}|\mathbf{E}'_{k2,m}])} a(\mathbf{A}_3^{l1} a(\mathbf{A}_2^{l1} a(\mathbf{A}_1^{l1}))) \tag{6.16}
$$

By partially differentiating all the module set layers in a similar manner, the final interaction matrix is derived as:

$$
\begin{aligned}
\mathbf{L}_f &= \mathbf{A}^f a(\mathbf{A}_3^{l3} a(\mathbf{A}_2^{l3} a(\mathbf{A}_1^{l3} \mathbf{A}^{''}))) \\
\mathbf{A}^{''} &= \left[a(\mathbf{A}_3^{l2} a(\mathbf{A}_2^{l2} a(\mathbf{A}_1^{l2} \mathbf{A}^{'}))) | \mathbf{L}_{\mathbf{E}'_{k3,m}} \right] \\
\mathbf{A}^{'} &= \left[a(\mathbf{A}_3^{l1} a(\mathbf{A}_2^{l1} a(\mathbf{A}_1^{l1} \mathbf{L}_{\mathbf{E}'_{k1,m}}))) | \mathbf{L}_{\mathbf{E}'_{k2,m}} \right],
\end{aligned}
\tag{6.17}
$$

The interaction matrix is calculated for each input at each iteration in the visual servoing loop. The interaction matrices $\mathbf{L}_c$ and $\mathbf{L}_f$ are then utilised in a velocity controller to estimate

the end-effector velocity that minimises the feature error.

6.3.3 Controller design

The proposed method utilises a hybrid approach where two distinct visual cues are employed to perform the VS task. To implement this approach effectively, a velocity control strategy has been designed to estimate the camera's spatial velocity by appropriately weighting the two visual features. In the proposed control method, the robot's translational velocity is determined using the centroid interaction matrix and the corresponding centroid feature error. The interaction matrix for the centroid, $\mathbf{L}_{\boldsymbol{c}}$, establishes a linear relationship between changes in the centroid position and the required translational velocity, as expressed in (6.10). To focus solely on the translational component, the interaction matrix is decoupled into translational and angular components as follows:

$$\mathbf{L}_{\boldsymbol{c}} = \begin{bmatrix} \mathbf{L}_{\boldsymbol{c}}^{trans} & \mathbf{L}_{\boldsymbol{c}}^{angular} \end{bmatrix},$$

where,

$$\begin{aligned} \mathbf{L}_{\boldsymbol{c}}^{trans} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ \mathbf{L}_{\boldsymbol{c}}^{angular} &= \begin{bmatrix} 0 & c_z & -c_y \\ -c_z & 0 & c_x \\ c_y & -c_x & 0 \end{bmatrix}. \end{aligned} \tag{6.18}$$

The translational velocity $\boldsymbol{v}_t$, computed using Equation (6.4), is then given by:

$$\boldsymbol{v}_t = -\lambda \mathbf{L}_{\boldsymbol{c}}^{trans^{\dagger}} \overline{\boldsymbol{c}}, \tag{6.19}$$

where λ is the control gain, $\mathbf{L}_{\boldsymbol{c}}^{trans^{\dagger}}$ is the pseudoinverse of the interaction matrix, and $\overline{\boldsymbol{c}}$ represents the centroid feature error.

For angular movements, the latent space feature representation is utilised. The interaction matrix for the latent features, $\mathbf{L}_f$, relates the changes in the latent feature vector to the required spatial velocity. The error in the latent space features, $\overline{f}$, is used to compute the angular velocity by partitioning the feature interaction matrix into translational and angular components, i.e., $\mathbf{L}_f = \begin{bmatrix} \mathbf{L}_f^{trans} & \mathbf{L}_f^{angular} \end{bmatrix}$. The control law, as defined in Equation (6.3), can be written as:

$$\begin{aligned} \overline{f} &= \mathbf{L}_f \boldsymbol{v}, \\ \overline{f} &= \begin{bmatrix} \mathbf{L}_f^{trans} & \mathbf{L}_f^{angular} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_t \\ \boldsymbol{v}_\omega \end{bmatrix}, \\ \overline{f} &= \mathbf{L}_f^{trans} \boldsymbol{v}_t + \mathbf{L}_f^{angular} \boldsymbol{v}_\omega. \end{aligned} \tag{6.20}$$

Using Equation (6.4), the angular velocity is formulated as:

$$\boldsymbol{v}_\omega = -\mathbf{L}_f^{angular^\dagger} (\lambda \overline{f} + \mathbf{L}_f^{trans} \boldsymbol{v}_t), \tag{6.21}$$

where $\mathbf{L}_f^{angular^\dagger}$ is the pseudoinverse of the angular component of the latent feature interaction matrix $\mathbf{L}_f$, and $\boldsymbol{v}_t$ is the translational velocity calculated using Equation (6.19). By decoupling the control tasks between translational and rotational DoF, the control strategy allows for independent optimisation of each movement type. The translational DoF are managed effectively by the centroid-based control, ensuring robust movement towards the target. Simultaneously, the rotational degrees of freedom are governed by the latent space features, providing a detailed representation of the object's orientation and enabling precise adjustments to the robot's pose.

Once the velocity has been calculated using the controller defined in Equations (6.19) and (6.21), it is necessary to apply this velocity to the robot's joint space to move to the next position. Since the visual information used to calculate the velocity is in the camera frame, the resulting velocities are also expressed in the camera frame. Before these velocities can be utilised to calculate the new joint positions, they are transformed into the robot's world frame. The transformation from the camera frame to the world frame is performed by applying a homogeneous transformation matrix $\mathbf{T}_{cw}$, which represents the pose of the camera with respect

to the world frame. Estimated camera velocity $\boldsymbol{v}$ in world frame $\boldsymbol{v}'$ is calculated as

$$\begin{aligned}
\mathbf{r}_w^c, \mathbf{t}_w^c &= [\mathbf{T}_w^c[:3, :3], \mathbf{T}_w^c[:3, 3] \\
\boldsymbol{v}_t, \boldsymbol{v}_\omega &= \boldsymbol{v}[3:], \boldsymbol{v}[:3] \\
\boldsymbol{v}_\omega^w &= \mathbf{r}_w^c \cdot \boldsymbol{v}_\omega \\
\boldsymbol{v}_t^w &= \mathbf{r}_w^c \cdot \boldsymbol{v}_t - \boldsymbol{v}_\omega{}^w \times \mathbf{t}_w^c \\
\boldsymbol{v}' &= [\boldsymbol{v}_t^w \boldsymbol{v}_\omega^w]^T
\end{aligned} \tag{6.22}$$

where $\mathbf{r}_w^c$ and $\mathbf{t}_w^c$ are the rotational and translational components of the camera pose, respectively. $\boldsymbol{v}_t$ and $\boldsymbol{v}_\omega$ are the translational and angular component respectively, of the velocity in camera frame, and $\boldsymbol{v}_t^w$ and $\boldsymbol{v}_\omega^w$ are the respective components of velocity in world frame. To convert the world frame velocity into joint space velocity, the robot's Jacobian matrix is utilised. The Jacobian matrix $\mathbf{J}_r$ relates the velocities in the Cartesian space to the velocities in the joint space. The general form of the Jacobian is given by:

$$\boldsymbol{v}_w = \mathbf{J}_r(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{6.23}$$

where $\boldsymbol{v}_w$ is the velocity in the world frame, $\boldsymbol{q}$ is the vector of joint angles, and $\dot{\boldsymbol{q}}$ is the joint velocity vector. The joint velocity is then calculated as,

$$\dot{\boldsymbol{q}} = \mathbf{J}_r(\boldsymbol{q})^\dagger \boldsymbol{v}_w, \tag{6.24}$$

Using the computed joint velocities $\dot{\boldsymbol{q}}$ from Equation (6.24), the new joint positions can be updated using the following integration:

$$\boldsymbol{q}_{new} = \boldsymbol{q}_{current} + \dot{\boldsymbol{q}}\Delta t, \tag{6.25}$$

where $\boldsymbol{q}_{new}$ represents the updated joint positions, $\boldsymbol{q}_{current}$ is the current joint position, and Δt is the time step. By applying these equations, the calculated velocities in the camera frame can be

effectively transformed into joint movements, ensuring accurate robot motion that aligns with the VS objectives.

6.4 Experimental results

The performance of the proposed method was validated in a simulated and real-world environment using a 7-DoF robotic arm. Multiple experiments were conducted with different object settings, and the convergence of the visual servoing control was thoroughly evaluated.

6.4.1 Experimental setup

The simulation environment was established using the PyBullet physics engine, which provides a reliable and efficient platform for robotics simulations [54]. The setup consisted of a 7-Degree of Freedom (7-DoF) robotic manipulator equipped with a parallel jaw gripper. A virtual 3D camera was simulated within the environment to capture the necessary visual information, including the object's point cloud. The camera was simulated in an eye-in-hand configuration [313], meaning that it was attached to the end-effector of the robotic arm. Note that the camera was calibrated with respect to the robot's end-effector to ensure accurate transformations between the camera and world coordinates. An equivalent real world environment was setup using a Kuka 7-Degree of Freedom (7-DoF) robot equipped with a parallel jaw gripper. An ensenso depth camera is placed in an eye-in-hand configuration [313]. The camera is calibrated using the mathematical formulations as discussed in Section 2.8.

The learning-based feature extractor was implemented using PyTorch [14] library in Python and distributed data learning strategy was utilised for training. The model was trained for 1001 epoch with a batch size of 100. A similar training data generation strategy is utilised as discussed in Section 4.5.1. Finally, 256 edge points are selected to be passed to the network. The 3D edges cloud before the addition of noise is considered as the expected output from the network. Overall, the feature extraction inference time of the network is $\approx$ 60ms, while the interaction matrix calculation time is $\approx$ 90ms for each sample. Overall pipeline time for each iteration in

the control loop was observed to be $\approx$ 200ms.

Three different objects were selected, a plastic water bottle, a cup, and a master chef can, all from the YCB objectset [32]. Notably, no texture information was utilised in the experiments, ensuring that the results were purely based on the geometric features of the objects.

In each experiment, a random pose of the object was chosen as the desired pose. To extract the features and centroid of this pose, the CAD model of the object was first transformed to the selected random pose. It was then projected onto a depth map using the virtual camera's intrinsic parameters. The 3D edge point cloud of the object at the desired pose was subsequently extracted from the depth map. The random pose was generated by selecting an orientation angle randomly within the range of $[-45°, 45°]$ along all three axes. Additionally, the object was translated randomly within the range of $[-0.5, 0.5]$ metres in the x and y axes, and between $[0.3, 1.8]$ metres along the z axis. From the captured point cloud, the desired centroid and 3D edge features were extracted and subsequently processed to generate the desired feature vector for control.

The experiments were conducted with a control gain $\lambda = 0.01$ and a maximum number of iterations set to 3000. Furthermore, a threshold of 0.001 was imposed on the feature error, and a threshold of 0.0001 was set on the velocities to determine the convergence of the proposed method. The interaction matrices considered to estimate the spatial velocity, as in equation (6.19) and (6.21), is computed utilising the desired pose latent features $\boldsymbol{f}^*$ and centroid $\boldsymbol{c}^*$.

6.4.2 Convergence analysis

The convergence of the visual servoing control was analysed across various experimental setups. The analysis focused on the reduction in feature error over the iterations and the stability of the robot's end-effector as it approached the desired pose.

Figures 6.2, 6.3, and 6.4 present the experimental results for the master chef can, plastic water bottle, and cup objects, respectively. For each experiment, the robot's motion, the respective object point cloud as viewed from the camera, compared against the desired object view, and the velocity convergence have been depicted. The green point cloud represents the object at the

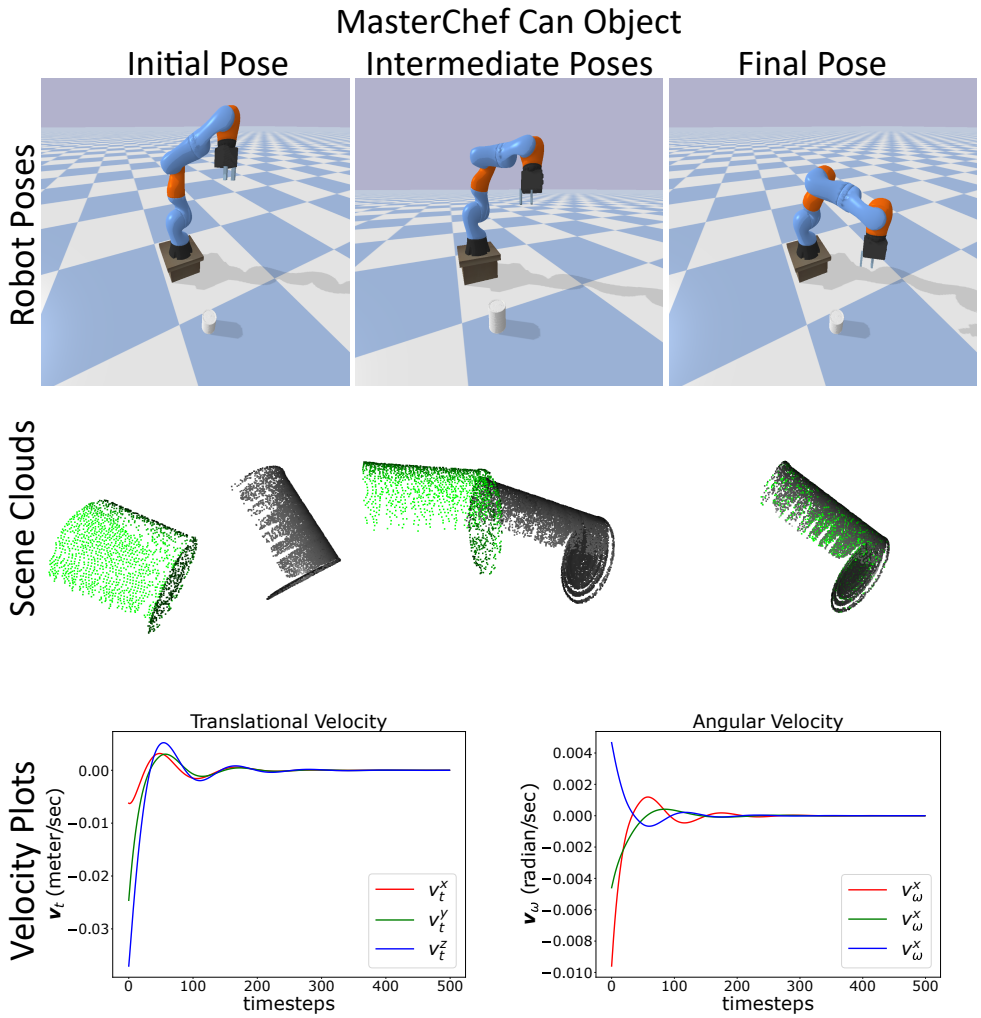

Figure 6.2: Illustration of the proposed visual servoing robot movements and velocity convergence for master chef can object.

robot's current pose, while the grey point cloud corresponds to the object at the desired pose. It should be noted that the entire object point cloud is used for display purposes only, while the servoing was performed using only the 3D edges.

From Figure 6.2, it was observed that smooth and consistent convergence was achieved by the proposed method. The transition of the robot from the initial to the final pose, along with the respective point clouds, further validates that the proposed method successfully minimises the feature error and aligns the current pose with the desired pose. The method demonstrated rapid convergence, requiring fewer than 500 iterations to reach the desired pose. Nevertheless, it is important to note that the selected master chef can is a symmetric object. The symmetry in such objects presents significant challenges in pose recognition due to nearly identical visual

features appear across different orientations. At times, this similarity can cause confusion in determining the object's exact pose, leading to difficulties in accurately aligning or manipulating the object during servoing tasks. As a result, achieving precise control and interaction with symmetric objects becomes problematic. To address this, the system can incorporate additional visual cues such as colour patterns, or small distinctive markers on the object's surface to break the symmetry and provide unique features that complement the 3D edges.

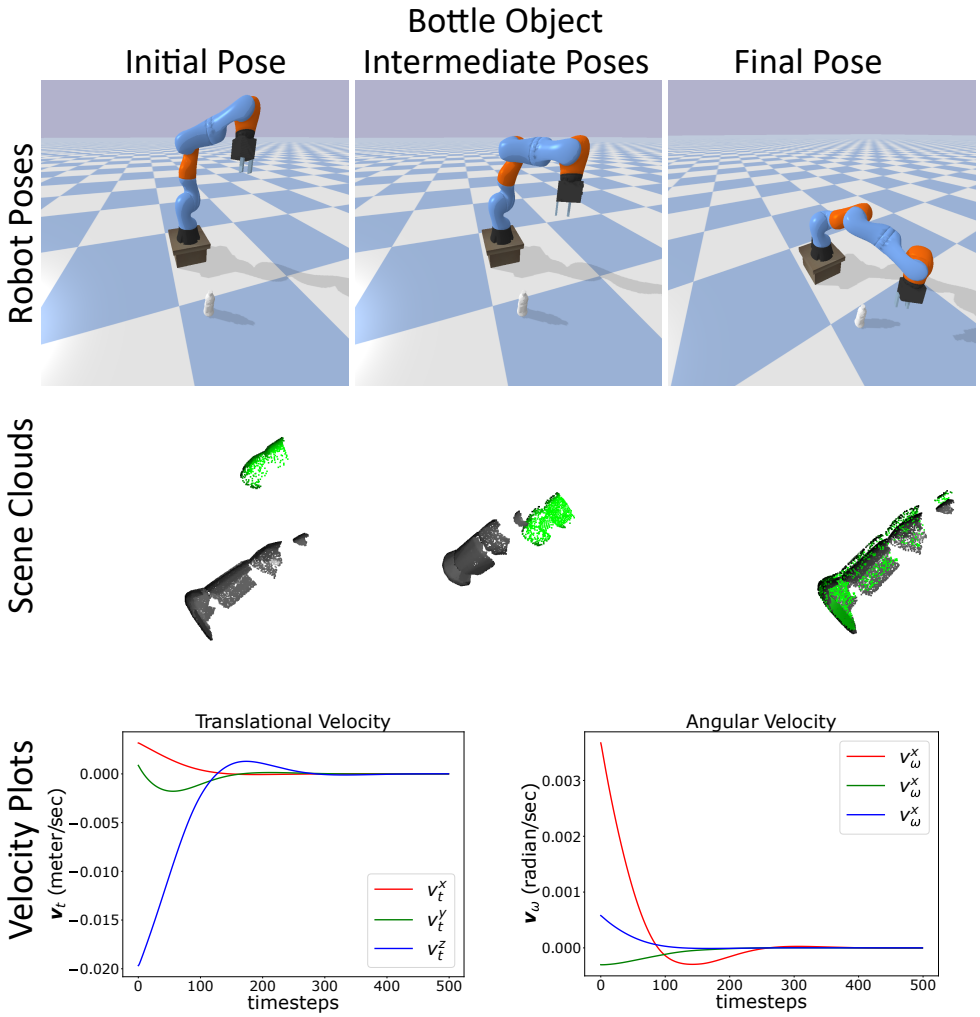

Figure 6.3: Illustration of the proposed visual servoing robot movements and velocity convergence for plastic water bottle object.

Next, the servoing performance for the plastic water bottle, as illustrated in Figure 6.3, indicates that the proposed method converged effectively in these experiments as well. The sequence of robot poses and the alignment of the point clouds further validate the success of the experiment achieved during the servoing task. It was observed that the method effectively

minimised the feature error and guided the robot's end-effector to align with the desired pose, even in the absence of texture information on the object. These results demonstrate that the proposed method is robust and reliable when dealing with texture-less objects, as it relies solely on geometric 3D edge features for the servoing process. The ability to perform precise alignment based on edge features alone highlights the method's effectiveness in scenarios where traditional visual features, such as texture and colour, are unavailable or unreliable. This capability is particularly valuable in real-world applications e.g. in industrial manufacturing where objects are metallic and shiny.

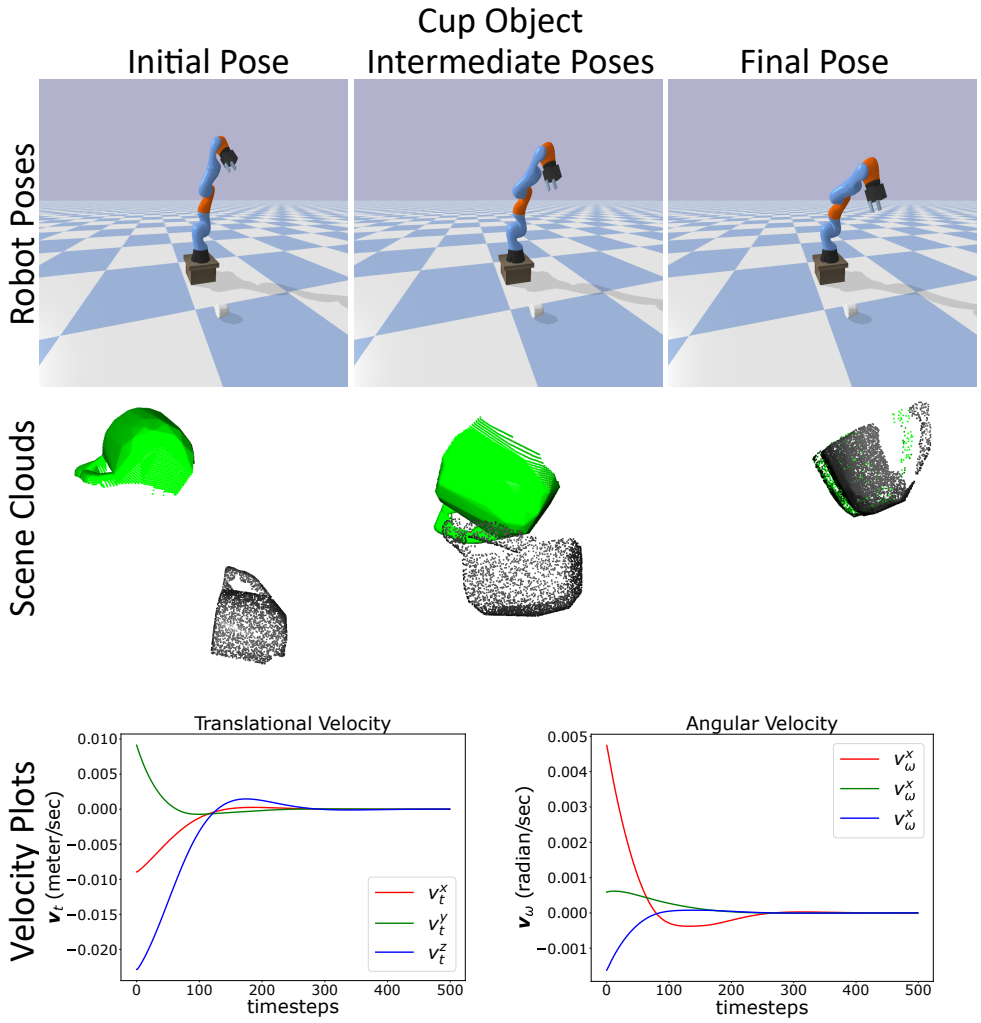

Figure 6.4: Illustration of the proposed visual servoing robot movements and velocity convergence for cup object.

Finally, the visual servoing performance for the cup object is shown in Figure 6.4. It was observed that the proposed method, similar to its performance with other objects, achieved

smooth convergence for the cup object. This smooth convergence is reflected in the robot's pose and the corresponding point cloud samples. However, it should be noted that the robot's initial pose was near a singularity, which reduced the available movement space and introduced additional challenges. A potential solution is to use a secondary control task operating in the robot's null space, particularly for robots with redundant joints.

The convergence anaylsis of the proposed method on real world robot experiments are provided in Figure 6.5 and Figure 6.6. The two figures present the translational and angular velocity convergence for when the target pose is only a variation in z-axis and when the target pose is moved in all directions, respectively. For both the scenarios, the convergence is achieved in less than 50 iterations. The initial fluctuations in the velocities stabilises quickly, obtaining a smooth control.

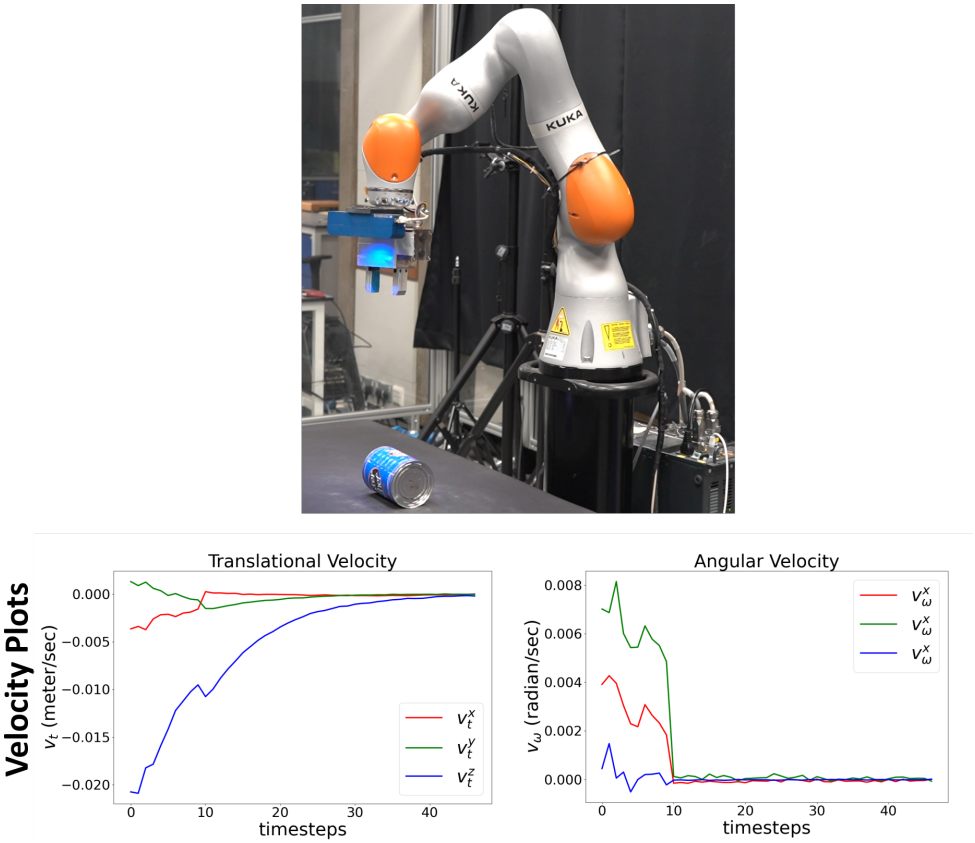

Figure 6.5: Illustration of the proposed visual servoing robot movements and velocity convergence for master chef can object in real world experiment with movement in only z-axis.

Overall, the experimental results demonstrated that the proposed control method achieved smooth and reliable convergence within the specified thresholds. The method effectively guided

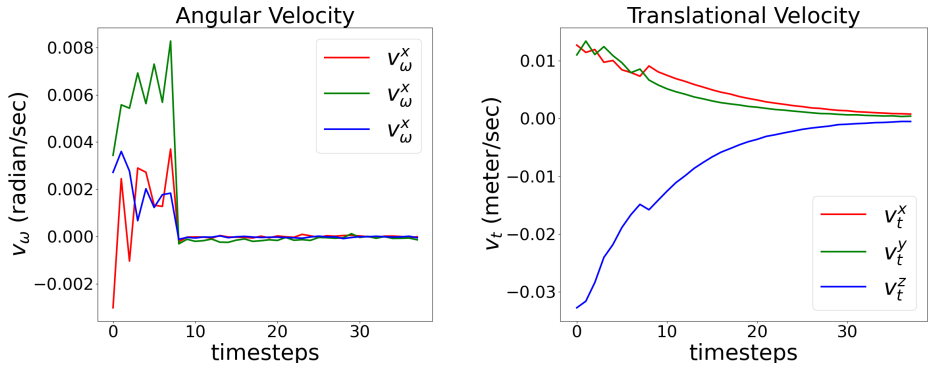

Figure 6.6: Illustration of the proposed visual servoing robot movements and velocity convergence for master chef can object in real world experiment with movement along all the axis.

the robot to the target pose in the simulated and real world environment, confirming its efficacy in dynamic and complex scenarios.

6.5 Summary

This chapter proposed a learning-based visual servoing approach that utilises both centroid-based and latent space feature-based control strategies to achieve precise robot motion. The proposed VS scheme leverages centroid features for robust translational control and latent space features for detailed rotational control. Visual information is first extracted from a 3D point cloud captured by a simulated camera. The object's centroid and a K-dimensional latent feature vector are computed as primary inputs for the control strategy. The centroid interaction matrix calculates the translational velocity, while the latent space interaction matrix determines the angular velocity. These velocities, initially computed in the camera frame, are then transformed into the world frame using the camera-to-world transformation matrix. The robot's joint velocities are determined by applying the pseudoinverse of the robot's Jacobian matrix to the transformed velocities, and the joint positions are updated iteratively to guide the end-effector towards the desired pose. The method was validated in a simulated environment using a 7-DoF robotic arm and objects from the YCB dataset, demonstrating smooth and reliable convergence in fewer than 500 iterations. The results confirmed that the considered control strategy effectively reduced feature errors and achieved the desired pose with high accuracy. The proposed method

shows promise in handling complex and dynamic environments, making it a robust solution for autonomous robotic tasks in unstructured settings.

Chapter 7

Conclusion

The widespread deployment of autonomous robots in industries like manufacturing, logistics, healthcare, and energy represents a major leap forward in robotics. Achieving full autonomy allows these robots to operate with greater efficiency, reliability, and safety. The components of sensing, processing, and actuation must seamlessly work together to ensure a reliable autonomous system. Among these, the ability to sense and process visual information is particularly important for accurately perceiving and interacting with the environment. However, challenges such as varying object textures, lighting conditions, environmental complexity, and intricate object details often hinder the effectiveness of these tasks.

This thesis presents methods for processing visual data that are resilient to variations in object texture, object complexity, and environmental complexity. An unsupervised learning-based method was initially proposed to detect 3D edges from depth maps. These 3D edges, which capture the geometric contours of objects, are robust features that can be extracted even in challenging scenarios, such as when dealing with texture-less objects. The extraction process involves encoding a scene's depth map at multiple scales, followed by the extraction of edge specific features at each scale. These per-point features are then clustered into edge or non-edge points through unsupervised deep clustering. The effectiveness of the detected 3D edges was evaluated using point cloud registration and F-score-based metrics across five benchmark datasets from the literature. This method was compared with four State-Of-The-Art (SOTA)

methods and demonstrated competitive 3D edge detection performance while not requiring manual threshold tuning or labelled data, both of which are necessary for other methods in the literature.

Building upon the detected 3D edges, a pose estimation method was developed to estimate the 6-DoF pose of objects in unstructured environments. This method was designed as a self-supervised deep learning-based network, that aligns the CAD model of an object with the detected object's 3D edges through deep feature correspondence. When compared to traditional and learning-based SOTA methods from the literature, the proposed method outperformed them on three benchmark datasets while utilising only 5% of the data points in the point cloud. The method proved effective for reliable pose estimation in complex, unstructured environments, utilising texture-independent 3D edge features, making it particularly suitable for texture-less objects.

To further enhance the understanding of object properties and facilitate the use of specific visual processing techniques, an articulated object classification method was introduced. This method was developed as a local region registration-based motion estimation approach, wherein a video sequence of an object being manipulated was used as input. Within this video, each pair of frames was classified as rigid motion, articulated motion, or no motion. This was achieved by extracting the object's point cloud from each frame and performing registration on local patches. The final classification was determined by accumulating the classifications across all frame sets in the video sequence. The method demonstrated high accuracy in classifying objects without the need for any object model or texture information.

Finally, the practical reliability of the detected 3D edges was evaluated through a visual servoing task. To accomplish this, a learning-based visual servoing approach was designed, wherein the 3D edge information was utilised to perform the servoing task. The effectiveness of the method was evaluated through simulations involving multiple texture-less objects, using a 7-DoF robot equipped with a depth camera in an eye-in-hand configuration. The results indicated that the task converged quickly, with the overall feature error decreasing smoothly.

In conclusion, the research presented in this thesis offers a robust framework for the process-

ing of 3D visual data in autonomous robotic systems, particularly in challenging environments. The developed methods have shown resilience to variations in object texture and complexity, demonstrating their applicability across a wide range of scenarios. These advancements contribute significantly to the field of robotics, paving the way for further research and development in autonomous systems capable of operating in diverse and complex environments.

7.1 Research impact

This research significantly advances the field of vision-guided robotics by addressing core challenges in object detection and pose estimation, particularly for texture-less objects in unstructured environments. The methods proposed in this work have implications across multiple industries and academic fields.

Across various industries, the proposed methods for 3D edge detection and pose estimation have been shown to enhance the precision, reliability, and flexibility of automated systems. In manufacturing and logistics, the handling of texture-less components, which often lack distinct visual features, has been significantly improved. The advancements allow robotic systems to perform grasping and manipulation tasks with greater accuracy, thereby streamlining operations in assembly lines and warehouses. In the healthcare sector, these innovations could contribute to enhanced precision in surgical robotics, enabling safer and more effective robotic-assisted procedures. Similarly, in the energy sector, tasks such as pipeline maintenance, wind turbine repairs, and nuclear facility operations are expected to benefit from improved autonomous capabilities, thereby reducing human exposure to hazardous environments while increasing operational reliability. In agriculture and mining, where robots are required to function in dynamic and unpredictable settings, these solutions facilitate more effective navigation and object manipulation, improving productivity while reducing the need for human intervention in high-risk activities.

Beyond industry-specific applications, this research makes valuable contributions to the fields of computer vision, robotics, and artificial intelligence. The introduction of unsupervised and

self-supervised learning methods has reduced reliance on extensive labelled datasets, making the proposed solutions more accessible to a wide range of applications. By employing 3D vision and point cloud data instead of traditional 2D methods, the thesis advances the understanding of spatial relationships and object geometry, establishing new benchmarks for vision-guided systems. The performance of these approaches has been validated against benchmark datasets, ensuring their relevance and applicability to real-world challenges.

In conclusion, the work conducted in this thesis addresses critical challenges in vision-guided robotics, making significant contributions to the development of autonomous systems. The methodologies proposed enhance flexibility, reliability, and efficiency, providing a strong foundation for the integration of such systems into diverse industries. The research not only advances academic understanding but also lays the groundwork for practical and impactful applications in industrial and real-world contexts.

7.2 Future research directions

Numerous unresolved research challenges related to the practical utilisation of autonomous robots in unstructured environments remain. However, within the scope of this thesis, the following open problems are proposed for future investigation.

3D Edge detection from unorganised point clouds In this thesis, the 3D edge detection method presented in Chapter 3 was performed on organised point clouds structured as depth maps. However, in real-world scenarios, point cloud data may not always be organised. Therefore, extending 3D edge detection to unorganised point clouds is suggested as an area for future research.

Joint modelling for articulated objects In Chapter 5, the classification of objects as articulated or rigid was addressed. However, the method could be extended in future research to also model the motion of different components during classification, thereby providing additional information about the type and number of articulations within the object.

Task-oriented application of visual servoing The visual servoing method proposed in Chapter 6 could be further developed in future research to incorporate additional tasks such as grasping, assembly, and manipulation, alongside servoing. This would allow for the study of servoing performance and task accuracy when using 3D edge features.

Task-Oriented Application with Pose Estimation The 6-DoF pose estimation method presented in Chapter 4 could be combined with tasks such as bin picking, grasping, assembly, and manufacturing in future research to evaluate its efficacy across different applications. Additionally, this approach could assess the performance accuracy of these tasks in texture-less and unstructured environments.

Non-Visual Object Recognition In this thesis, visual features, in the form of 3D point clouds, have been utilised in all the proposed methods. However, visual data may not always be available. For such scenarios, tactile sensors could be employed to gather the necessary information. Therefore, as future work, the methods proposed in this thesis could be extended to utilise tactile sensor data to perform the respective tasks.

Handling object deformation Objects in the real world may undergo deformations due to external pressures, which can adversely affect the performance of object recognition and pose estimation methods designed specifically for rigid objects. In future work, the potential of 3D edges could be explored to evaluate their effectiveness in recognising and estimating the pose of deformed objects.

7.3 Dataset bias and impact

Datasets for visual object detection and pose estimation, as discussed in Section 2.7, often suffer from biases in fairness, inclusiveness, and representation, limiting their generalisability to global applications. Many datasets are skewed toward specific object categories, regions, or environments, such as industrial or structured indoor settings, while under representing

dynamic, unstructured environments like agricultural fields or disaster zones. This leads to challenges in recognising objects with diverse shapes, textures, and cultural variations, such as deformable objects or region-specific designs. Furthermore, the dominance of objects sourced from particular regions introduces representation bias, introducing challenges to the methods effectiveness when deployed in different global contexts.

The methods proposed in the thesis address challenges in dataset bias, representation, and inclusiveness through the use of unsupervised and self-supervised learning approaches, which reduce reliance on extensive labelled datasets. This allows for improved generalisation to unseen objects and environments. By utilising 3D edge-based features and point clouds, the methods are designed to be texture-independent, enabling robust performance on diverse object surfaces. In this way, biases associated with datasets that predominantly feature visually distinctive, textured objects are mitigated.

Additionally, the methodologies have been validated on multiple benchmark datasets, which include unstructured environments and objects with varying complexity. This ensures that the models are exposed to a range of diverse scenarios, thereby reducing the effects of representation bias. The inclusion of techniques to handle both articulated and rigid objects enhances inclusiveness, allowing the methods to generalise to a wider variety of object types encountered across different industries and regions.

7.4 Social and ethical impacts

The thesis introduces advancements that could have significant social and ethical implications across various industries. The improved precision and reliability of autonomous systems could enhance workplace safety by reducing human involvement in hazardous tasks, particularly in sectors such as mining, nuclear decommissioning, and agriculture. However, concerns regarding job displacement due to automation must be addressed through ethical measures such as retraining and upskilling affected workers.

Ethical considerations related to dataset bias and representation must also be taken into

account. Although the methods focus on texture-less objects and unstructured environments, insufficiently diverse datasets may lead to systems that perform poorly in underrepresented scenarios. Furthermore, privacy concerns may arise in applications such as surveillance, healthcare, and personal robotics, where the handling of sensitive data must comply with ethical and regulatory standards. The environmental impact of deploying autonomous systems, including energy consumption and electronic waste, must also be managed to ensure sustainability.

To address these issues, the equitable deployment of these technologies must be prioritised to avoid exacerbating inequalities between regions or industries. Transparency in the development of datasets and methodologies should be encouraged, along with the establishment of regulatory frameworks to monitor and prevent misuse. These measures would help ensure that the social and ethical impact of the research remains positive and widely beneficial.

References

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Syed Mohammad Abid Hasan and Kwanghee Ko. "Depth edge detection by image-based smoothing and morphological operations". In: *Journal of Computational Design and Engineering* 3.3 (Feb. 2016), pp. 191–197. ISSN: 2288-5048. DOI: 10.1016/j.jcde.2016.02.002. eprint: https://academic.oup.com/jcde/article-pdf/3/3/191/33134158/j.jcde.2016.02.002.pdf. URL: https://doi.org/10.1016/j.jcde.2016.02.002.

[3] Ralph Adams and Loren Bischof. "Seeded region growing". In: *IEEE Transactions on pattern analysis and machine intelligence* 16.6 (1994), pp. 641–647.

[4] Maxime Adjigble et al. "3D Spectral Domain Registration-Based Visual Servoing". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 769–775. DOI: 10.1109/ICRA48891.2023.10160430.

[5] Maxime Adjigble et al. "An assisted telemanipulation approach: combining autonomous grasp planning with haptic cues". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 3164–3171.

[6] Maxime Adjigble et al. "Model-free and learning-free grasping by Local Contact Moment matching". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 2933–2940. DOI: 10.1109/IROS.2018.8594226.

[7] Maxime Adjigble et al. “SpectGRASP: Robotic Grasping by Spectral Correlation”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 3987–3994.

[8] Abdul Afram and Farrokh Janabi-Sharifi. “Theory and applications of HVAC control systems – A review of model predictive control (MPC)”. In: *Building and Environment* 72 (2014), pp. 343–355. ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2013.11.016. URL: https://www.sciencedirect.com/science/article/pii/S0360132313003363.

[9] Abien Fred Agarap. “Deep Learning using Rectified Linear Units (ReLU)”. In: *CoRR* abs/1803.08375 (2018). arXiv: 1803.08375. URL: http://arxiv.org/abs/1803.08375.

[10] Ayush Aggarwal, Rustam Stolkin, and Naresh Marturi. “Local Region-to-Region Mapping-based Approach to Classify Articulated Objects”. In: *2023 20th Conference on Robots and Vision (CRV)*. 2023, pp. 177–183. DOI: 10.1109/CRV60082.2023.00030.

[11] Ayush Aggarwal, Rustam Stolkin, and Naresh Marturi. “Unsupervised learning-based approach for detecting 3D edges in depth maps”. In: *Scientific Reports* 14.1 (Jan. 2024), p. 796. ISSN: 2045-2322. DOI: 10.1038/s41598-023-50899-3. URL: https://doi.org/10.1038/s41598-023-50899-3.

[12] S. M. Ahmed et al. “Edge and Corner Detection for Unorganized 3D Point Clouds with Application to Robotic Welding”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 7350–7355.

[13] N. Anandakrishnan and S. S. Baboo. “An Evaluation of Popular Edge Detection Techniques in Digital Image Processing”. In: *2014 International Conference on Intelligent Computing Applications*. Mar. 2014, pp. 213–217. DOI: 10.1109/ICICA.2014.53.

[14] Jason Ansel et al. “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation”. In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS ’24. La Jolla, CA, USA: Association for Com-

puting Machinery, 2024, pp. 929–947. DOI: 10.1145/3620665.3640366. URL: https://doi.org/10.1145/3620665.3640366.

[15] Yasuhiro Aoki et al. "PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[16] K. S. Arun, T. S. Huang, and S. D. Blostein. "Least-Squares Fitting of Two 3-D Point Sets". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.5 (1987), pp. 698–700. DOI: 10.1109/TPAMI.1987.4767965.

[17] Ahmad Asadi and Reza Safabakhsh. "The Encoder-Decoder Framework and Its Applications". In: *Deep Learning: Concepts and Architectures*. Ed. by Witold Pedrycz and Shyi-Ming Chen. Cham: Springer International Publishing, 2020, pp. 133–167. ISBN: 978-3-030-31756-0. DOI: 10.1007/978-3-030-31756-0_5. URL: https://doi.org/10.1007/978-3-030-31756-0%5C_5.

[18] Jrgen Assfalg et al. "Content-Based Retrieval of 3-D Objects Using Spin Image Signatures". In: *IEEE Transactions on Multimedia* 9.3 (2007), pp. 589–599. DOI: 10.1109/TMM.2006.886271.

[19] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495. DOI: 10.1109/TPAMI.2016.2644615.

[20] Ms. Aayushi Bansal, Dr. Rewa Sharma, and Dr. Mamta Kathuria. "A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications". In: *ACM Comput. Surv.* 54.10s (Sept. 2022). ISSN: 0360-0300. DOI: 10.1145/3502287. URL: https://doi.org/10.1145/3502287.

[21] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and

Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[22] Khaled Bayoudh et al. "A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets". In: *The Visual Computer* 38.8 (Aug. 2022), pp. 2939–2970. ISSN: 1432-2315. DOI: 10.1007/s00371-021-02166-7. URL: https://doi.org/10.1007/s00371-021-02166-7.

[23] A. Beinglass and H.J. Wolfson. "Articulated object recognition, or: how to generalize the generalized Hough transform". In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1991, pp. 461–466. DOI: 10.1109/CVPR.1991.139736.

[24] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[25] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[26] Serge Beucher and Christian Lantuéjoul. "Use of watersheds in contour detection". In: *International workshop on image processing: real-time edge and motion detection/estimation*. 1979, pp. 17–21.

[27] Lukas Bode, Michael Weinmann, and Reinhard Klein. "BoundED: Neural boundary and edge detection in 3D point clouds via local neighborhood statistics". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 205 (2023), pp. 334–351. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2023.09.023. URL: https://www.sciencedirect.com/science/article/pii/S0924271623002642.

[28] Ali Borji et al. "Salient object detection: A survey". In: *Computational Visual Media* 5.2 (June 2019), pp. 117–150. ISSN: 2096-0662. DOI: 10.1007/s41095-019-0149-9. URL: https://doi.org/10.1007/s41095-019-0149-9.

[29] R. Bormann et al. “Fast and accurate normal estimation by efficient 3d edge detection”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 3930–3937. DOI: 10.1109/IROS.2015.7353930.

[30] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. “Sparse iterative closest point”. In: *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*. SGP ’13. Genova, Italy: Eurographics Association, 2013, pp. 113–123. DOI: 10.1111/cgf.12178. URL: https://doi.org/10.1111/cgf.12178.

[31] Eric Brachmann et al. “Learning 6D Object Pose Estimation Using 3D Object Coordinates”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 536–551. ISBN: 978-3-319-10605-2.

[32] Berk Calli et al. “Yale-CMU-Berkeley dataset for robotic manipulation research”. In: *The International Journal of Robotics Research* 36.3 (2017), pp. 261–268. DOI: 10.1177/0278364917700714. eprint: https://doi.org/10.1177/0278364917700714. URL: https://doi.org/10.1177/0278364917700714.

[33] J Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8.6 (June 1986), pp. 679–698. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851.

[34] Siddhartha Chandra et al. “Context Aware 3D CNNs for Brain Tumor Segmentation”. In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Ed. by Alessandro Crimi et al. Cham: Springer International Publishing, 2019, pp. 299–310. ISBN: 978-3-030-11726-9.

[35] Francois Chaumette and Seth Hutchinson. “Visual servo control. I. Basic approaches”. In: *IEEE Robotics & Automation Magazine* 13.4 (2006), pp. 82–90. DOI: 10.1109/MRA.2006.250573.

[36] Francois Chaumette and Seth Hutchinson. “Visual servo control. II. Advanced approaches [Tutorial]”. In: *IEEE Robotics & Automation Magazine* 14.1 (2007), pp. 109–118. DOI: 10.1109/MRA.2007.339609.

[37] François Chaumette and Seth Hutchinson. “3D Visual Servoing Based on the Estimation of the Depth by the Image Jacobian Matrix”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2003, pp. 2605–2611.

[38] François Chaumette and Seth Hutchinson. “Visual Servoing and Visual Tracking”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 563–583. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_25. URL: https://doi.org/10.1007/978-3-540-30301-5%5C_25.

[39] A. Chayeb et al. “HOG based multi-object detection for urban navigation”. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014, pp. 2962–2967. DOI: 10.1109/ITSC.2014.6958165.

[40] Meriem Chebbi et al. “A Novel Hybrid Visual Servoing Approach for Safe Human-Robot Interaction Using Eye-in-Hand Configuration”. In: *IEEE Access* 7 (2019), pp. 165916–165928.

[41] Hao Chen et al. “VoxResNet: Deep voxelwise residual networks for brain segmentation from 3D MR images”. In: *NeuroImage* 170 (2018). Segmenting the Brain, pp. 446–455. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2017.04.041. URL: https://www.sciencedirect.com/science/article/pii/S1053811917303348.

[42] Heping Chen et al. “Opportunities and Challenges of Robotics and Automation in Offshore Oil & Gas Industry”. In: *Intelligent Control and Automation* 05No.03 (2014), p. 10.

[43] Jian Chen and Lei Wang. “Multi-scale geometric features for point cloud-based object detection”. In: *Proceedings of the International Conference on 3D Vision*. Springer. 2022, pp. 201–209.

[44] Xinlei Chen and Abhinav Gupta. “Spatial Memory for Context Reasoning in Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[45] Zhe Chen, Jiaqi Wang, and Xiaofei Li. "Generalizing Object Pose Estimation with Synthetic and Real-World Data". In: *arXiv preprint arXiv:2403.06107* (2023).

[46] C. Choi, A. J. B. Trevor, and H. I. Christensen. "RGB-D edge detection and edge-based registration". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2013, pp. 1568–1575. DOI: 10.1109/IROS.2013.6696558.

[47] Changhyun Choi and Henrik I Christensen. "3d textureless object detection and tracking: An edge-based approach". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2012, pp. 3877–3884.

[48] Chi-Wei Chun, O.C. Jenkins, and M.J. Mataric. "Markerless kinematic model and motion capture from volume sequences". In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. 2003, pp. II–II. DOI: 10.1109/CVPR.2003.1211505.

[49] L. Claussmann et al. "A review of motion planning for highway autonomous driving". In: *IEEE Transactions on Intelligent Transportation Systems* 21 (2020), pp. 1826–1848.

[50] Cognex Corporation. *Cognex VisionPro*. VisionPro Vision Software. 2024. URL: https://www.cognex.com/products/machine-vision/vision-software/visionpro.

[51] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. "The MOPED framework: Object recognition and pose estimation for manipulation". In: *The International Journal of Robotics Research* 30.10 (2011), pp. 1284–1306. DOI: 10.1177/0278364911401765. eprint: https://doi.org/10.1177/0278364911401765. URL: https://doi.org/10.1177/0278364911401765.

[52] Alvaro Collet et al. "Object recognition and full pose registration from a single image for robotic manipulation". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 48–55. DOI: 10.1109/ROBOT.2009.5152739.

[53] McKinsey & Company. "Automation in logistics: Big opportunity, bigger uncertainty". In: *McKinsey & Company* (2019).

[54] Erwin Coumans and Yunfei Bai. *Pybullet, a python module for physics simulation for games, robotics and machine learning*. 2016.

[55] Jinrong Cui et al. "Learning robust latent representation for discriminative regression". In: *Pattern Recognition Letters* 117 (2019), pp. 193–200. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.04.016. URL: https://www.sciencedirect.com/science/article/pii/S0167865518301363.

[56] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[57] G. Danciu, S. M. Banu, and A. Căliman. "Shadow removal in depth images morphology-based for Kinect cameras". In: *2012 16th International Conference on System Theory, Control and Computing (ICSTCC)*. Oct. 2012, pp. 1–6.

[58] Zheng Dang et al. "Match Normalization: Learning-Based Point Cloud Registration for 6D Object Pose Estimation in the Real World". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.6 (2024), pp. 4489–4503. DOI: 10.1109/TPAMI.2024.3355198.

[59] Cristiana De Farias et al. "Dual quaternion-based visual servoing for grasping moving objects". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2021, pp. 151–158.

[60] Alessandro De Luca, Giuseppe Oriolo, and Fabrizio Caccavale. "Dual-armed Visual Servoing Using Image Moments: Theory and Experiments". In: *IEEE Transactions on Robotics* 37.3 (2021), pp. 896–910.

[61] Haowen Deng, Tolga Birdal, and Slobodan Ilic. "PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[62] Haowen Deng, Tolga Birdal, and Slobodan Ilic. "PPFNet: Global Context Aware Local Features for Robust 3D Point Matching". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[63] Xinke Deng et al. "PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking". In: *IEEE Transactions on Robotics* 37.5 (2021), pp. 1328–1342. DOI: 10.1109/TRO.2021.3056043.

[64] Zhipeng Ding, Xu Han, and Marc Niethammer. "VoteNet: A Deep Learning Label Fusion Method for Multi-atlas Segmentation". In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*. Ed. by Dinggang Shen et al. Cham: Springer International Publishing, 2019, pp. 202–210. ISBN: 978-3-030-32248-9.

[65] Thanh-Toan Do et al. *Deep-6DPose: Recovering 6D Object Pose from a Single RGB Image*. 2018. arXiv: 1802.10367 [cs.CV]. URL: https://arxiv.org/abs/1802.10367.

[66] Ivan Dokmanic et al. "Euclidean Distance Matrices: Essential theory, algorithms, and applications". In: *IEEE Signal Processing Magazine* 32.6 (2015), pp. 12–30. DOI: 10.1109/MSP.2015.2398954.

[67] Bertram Drost et al. "Introducing MVTec ITODD — A Dataset for 3D Object Recognition in Industry". In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 2200–2208. DOI: 10.1109/ICCVW.2017.257.

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

[69] Andreas Eitel et al. "Multimodal deep learning for robust RGB-D object recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 681–687. DOI: 10.1109/IROS.2015.7353446.

[70] Amal El Kaid and Karim Baïna. "A Systematic Review of Recent Deep Learning Approaches for 3D Human Pose Estimation". In: *Journal of Imaging* 9.12 (2023). ISSN:

2313-433X. DOI: 10.3390/jimaging9120275. URL: https://www.mdpi.com/2313-433X/9/12/275.

[71] Y. Eldar et al. "The farthest point strategy for progressive image sampling". In: *IEEE Transactions on Image Processing* 6.9 (1997), pp. 1305–1315. DOI: 10.1109/83.623193.

[72] Erich Elsen et al. "Fast Sparse ConvNets". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[73] Martin Engelcke et al. "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1355–1361. DOI: 10.1109/ICRA.2017.7989161.

[74] Zicong Fan et al. *Articulated Objects in Free-form Hand Interaction*. 2022. DOI: 10.48550/ARXIV.2204.13662. URL: https://arxiv.org/abs/2204.13662.

[75] Samuel Felton, Elisa Fromont, and Eric Marchand. "Deep metric learning for visual servoing: when pose and image meet in latent space". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 741–747. DOI: 10.1109/ICRA48891.2023.10160963.

[76] Samuel Felton et al. "Visual Servoing in Autoencoder Latent Space". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3234–3241. DOI: 10.1109/LRA.2022.3144490.

[77] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587597.

[78] Chen Feng, Vineet R. Kamat, and Carol C. Menassa. "Marker-Assisted Structure from Motion for 3D Environment Modeling and Object Pose Estimation". In: *Construction Research Congress 2016*, pp. 2604–2613. DOI: 10.1061/9780784479827.259. eprint: https://ascelibrary.org/doi/pdf/10.1061/9780784479827.259. URL: https://ascelibrary.org/doi/abs/10.1061/9780784479827.259.

[79] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[80] Nate Gans, Seth Hutchinson, and François Chaumette. “A Hybrid Approach to Visual Servoing: Combining Image-Based and Position-Based Techniques”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2003, pp. 317–322.

[81] K. Ghosh et al. “Simulation of density based traffic control system using Proteus 7.1 Professional”. In: *Proceedings of 3rd International Conference on Artificial Intelligence: Advances and Applications: ICAIAA 2022* (2023), pp. 493–504.

[82] Ross Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[83] Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR ’14. USA: IEEE Computer Society, 2014, pp. 580–587. ISBN: 9781479951185. DOI: 10.1109/CVPR.2014.81. URL: https://doi.org/10.1109/CVPR.2014.81.

[84] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Available online: http://www.deeplearningbook.org. MIT Press, 2016. ISBN: 978-0262035613. URL: http://www.deeplearningbook.org.

[85] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. “3D Semantic Segmentation With Submanifold Sparse Convolutional Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[86] Jiuxiang Gu et al. *Exploring the Frontiers of Softmax: Provable Optimization, Applications in Diffusion Model, and Beyond*. 2024. arXiv: 2405.03251 [cs.LG]. URL: https://arxiv.org/abs/2405.03251.

[87] Paul Guerrero et al. “DepthCut: improved depth edge estimation using multiple unreliable channels”. In: *The Visual Computer* 34.9 (Sept. 2018), pp. 1165–1176. DOI: 10.1007/s00371-018-1551-5.

[88] Stefan Gumhold, Xinlong Wang, and Rob MacLeod. “Feature Extraction from Point Clouds”. In: *Proceedings of 10th international meshing roundtable* 2001 (Nov. 2001).

[89] Jianwei Guo et al. “Efficient Center Voting for Object Detection and 6D Pose Estimation in 3D Point Cloud”. In: *IEEE Transactions on Image Processing* 30 (2021), pp. 5072–5084. DOI: 10.1109/TIP.2021.3078109.

[90] Yulan Guo et al. “A Comprehensive Performance Evaluation of 3D Local Feature Descriptors”. In: *International Journal of Computer Vision* 116.1 (Jan. 2016), pp. 66–89. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0824-y. URL: https://doi.org/10.1007/s11263-015-0824-y.

[91] Yulan Guo et al. “Deep Learning for 3D Point Clouds: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.12 (2021), pp. 4338–4364. DOI: 10.1109/TPAMI.2020.3005434.

[92] Saurabh Gupta et al. “Learning Rich Features from RGB-D Images for Object Detection and Segmentation”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 345–360. ISBN: 978-3-319-10584-0.

[93] T. Hackel, J. D. Wegner, and K. Schindler. “Contour Detection in Unstructured 3D Point Clouds”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 1610–1618. DOI: 10.1109/CVPR.2016.178.

[94] Frederik Hagelskjær and Rasmus Laurvig Haugaard. *KeyMatchNet: Zero-Shot Pose Estimation in 3D Point Clouds by Generalized Keypoint Matching*. 2023. arXiv: 2303.16102 [cs.CV]. URL: https://arxiv.org/abs/2303.16102.

[95] Xian-Feng Han et al. “3D point cloud descriptors: state-of-the-art”. In: *Artificial Intelligence Review* 56.10 (Oct. 2023), pp. 12033–12083. ISSN: 1573-7462. DOI: 10.1007/s10462-023-10486-4. URL: https://doi.org/10.1007/s10462-023-10486-4.

[96] Tiantian Hao, De Xu, and Fangbo Qin. "Image-Based Visual Servoing for Position Alignment With Orthogonal Binocular Vision". In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–10. DOI: 10.1109/TIM.2023.3289560.

[97] Chris Harris, Mike Stephens, et al. "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[98] Caner Hazirbas et al. "FuseNet: Incorporating Depth into Semantic Segmentation via Fusion-Based CNN Architecture". In: *Computer Vision – ACCV 2016*. Ed. by Shang-Hong Lai et al. Cham: Springer International Publishing, 2017, pp. 213–228. ISBN: 978-3-319-54181-5.

[99] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[100] Kaiming He et al. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[101] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

[102] Stefan Hinterstoisser et al. "Gradient response maps for real-time detection of textureless objects". In: *IEEE transactions on pattern analysis and machine intelligence*. Vol. 34. 5. IEEE. 2011, pp. 876–888.

[103] Stefan Hinterstoisser et al. "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes". In: *Proc. Computer Vision – ACCV 2012*. Berlin, Heidelberg, 2013, pp. 548–562. ISBN: 978-3-642-37331-2.

[104] Stefan Hinterstoisser et al. "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes". In: *2011 International Conference on Computer Vision*. 2011, pp. 858–865. DOI: 10.1109/ICCV.2011.6126326.

[105] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647. eprint: https://www.science.org/doi/pdf/10.1126/science.1127647. URL: https://www.science.org/doi/abs/10.1126/science.1127647.

[106] Tomáš Hodan et al. "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 880–888. DOI: 10.1109/WACV.2017.103.

[107] Tomáš Hodaň et al. "BOP: Benchmark for 6D Object Pose Estimation". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, 2018, pp. 19–35. ISBN: 978-3-030-01249-6.

[108] Tomáš Hodaň et al. "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017).

[109] Sebastian Höfer et al. "Sim2Real in Robotics and Automation: Applications and Challenges". In: *IEEE Transactions on Automation Science and Engineering* 18.2 (2021), pp. 398–400. DOI: 10.1109/TASE.2021.3064065.

[110] Sabera Hoque et al. "A Comprehensive Review on 3D Object Detection and 6D Pose Estimation With Deep Learning". In: *IEEE Access* 9 (2021), pp. 143746–143770. DOI: 10.1109/ACCESS.2021.3114399.

[111] Qingyong Hu et al. "RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[112] Zeyu Hu et al. "JSENet: Joint Semantic Segmentation and Edge Detection Network for 3D Point Clouds". In: *Computer Vision – ECCV 2020*. 2020, pp. 222–239.

[113] Xiaoshui Huang et al. *A comprehensive survey on point cloud registration*. 2021. arXiv: 2103.02690 [cs.CV]. URL: https://arxiv.org/abs/2103.02690.

[114] Zhenyu Huang et al. “Learning with Noisy Correspondence for Cross-modal Matching”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 29406–29419. URL: https://proceedings.neurips.cc/paper%5C_files/paper/2021/file/f5e62af885293cf4d511ceef31e61c80-Paper.pdf.

[115] Grinnell Jones III and Bir Bhanu. “Recognizing articulated objects in SAR images”. In: *Pattern Recognition* 34.2 (2001), pp. 469–485. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(99)00218-6. URL: https://www.sciencedirect.com/science/article/pii/S0031320399002186.

[116] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[117] Omid Hosseini Jafari et al. *iPose: Instance-Aware 6D Pose Estimation of Partly Occluded Objects*. 2018. arXiv: 1712.01924 [cs.CV]. URL: https://arxiv.org/abs/1712.01924.

[118] Maximilian Jaritz, Jiayuan Gu, and Hao Su. “Multi-View PointNet for 3D Scene Understanding”. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019, pp. 3995–4003. DOI: 10.1109/ICCVW.2019.00494.

[119] Chao Jiang et al. “A Deep Learning Approach for Textureless Object Recognition Using RGB-D Data”. In: *Sensors* 20.18 (2020), p. 5098.

[120] Haobo Jiang et al. “Center-Based Decoupled Point-cloud Registration for 6D Object Pose Estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 3427–3437.

[121] Haobo Jiang et al. “SE(3) Diffusion Model-based Point Cloud Registration for Robust 6D Object Pose Estimation”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=Znpz1sv4IP.

[122] Peiyuan Jiang et al. “A Review of Yolo Algorithm Developments”. In: *Procedia Computer Science* 199 (2022). The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital

Economy after COVID-19, pp. 1066–1073. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.01.135. URL: https://www.sciencedirect.com/science/article/pii/S1877050922001363.

[123] L. Jing and Y. Tian. "Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. DOI: 10.1109/TPAMI.2020.2992393.

[124] A.E. Johnson and M. Hebert. "Using spin images for efficient object recognition in cluttered 3D scenes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.5 (1999), pp. 433–449. DOI: 10.1109/34.765655.

[125] Sam Johnson and Mark Everingham. "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation". In: *Proceedings of the British Machine Vision Conference*. doi:10.5244/C.24.12. BMVA Press, 2010, pp. 12.1–12.11. ISBN: 1-901725-40-5.

[126] M. Kaneko et al. "Fast 3D edge detection by using decision tree from depth image". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 1314–1319.

[127] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. "RotationNet for Joint Object Categorization and Unsupervised Pose Estimation from Multi-View Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43.1 (2021), pp. 269–283. DOI: 10.1109/TPAMI.2019.2922640.

[128] N. Kanopoulos, N. Vasanthavada, and R. L. Baker. "Design of an image edge detection filter using the Sobel operator". In: *IEEE Journal of Solid-State Circuits* 23.2 (1988), pp. 358–367.

[129] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. "Snakes: Active contour models". In: *International journal of computer vision* 1.4 (1988), pp. 321–331.

[130] Alex Kendall and Roberto Cipolla. "Geometric Loss Functions for Camera Pose Regression With Deep Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[131] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[132] Keyence Corporation. *Keyence Vision Systems*. Vision System Software. 2024. URL: https://www.keyence.com/products/vision/vision-systems/.

[133] Alexander Kirillov et al. "Segment Anything". In: *arXiv:2304.02643* (2023).

[134] Gregor Klančar, Marija Seder, and Sašo Blažič. "Advanced Sensors Technologies Applied in Mobile Robot". In: *Sensors* 23 (2023), p. 2958.

[135] Alexander Kolesnikov et al. *Big Transfer (BiT): General Visual Representation Learning*. 2020. arXiv: 1912.11370 [cs.CV]. URL: https://arxiv.org/abs/1912.11370.

[136] Hendrik Königshof, Niels Ole Salscheider, and Christoph Stiller. "Realtime 3D Object Detection for Automated Driving Using Stereo Vision and Semantic Information". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 1405–1410. DOI: 10.1109/ITSC.2019.8917330.

[137] Danica Kragic and Henrik I. Christensen. "Hybrid Visual Servoing for Object Manipulation: Combining Image-Based and Position-Based Approaches". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2002, pp. 2552–2557.

[138] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[139] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. ISSN: 00034851. URL: http://www.jstor.org/stable/2236703.

[140] Chandra Kushal, Lei Wang, and Fan Yang. "3D Object Recognition Using Template Matching for Textureless Objects". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.12 (2022), pp. 987–999.

[141] Kevin Lai et al. “A large-scale hierarchical multi-view RGB-D object dataset”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1817–1824. DOI: 10.1109/ICRA.2011.5980382.

[142] Truc Le, Yuyan Li, and Ye Duan. *RED-NET: A Recursive Encoder-Decoder Network for Edge Detection*. 2019. arXiv: 1912.02914 [cs.CV]. URL: https://arxiv.org/abs/1912.02914.

[143] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[144] V. Lepetit and P. Fua. “Keypoint recognition using randomized trees”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.9 (2006), pp. 1465–1479. DOI: 10.1109/TPAMI.2006.188.

[145] Jiahao Li et al. “Iterative Distance-Aware Similarity Matrix Convolution with Mutual-Supervised Point Elimination for Efficient Point Cloud Registration”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 378–394. ISBN: 978-3-030-58586-0.

[146] Tong Li et al. “Self-Recognition Grasping Operation with a Vision-Based Redundant Manipulator System”. In: *Applied Sciences* 9.23 (2019). ISSN: 2076-3417. DOI: 10.3390/app9235172. URL: https://www.mdpi.com/2076-3417/9/23/5172.

[147] Y. Li et al. “Unsupervised Learning of Edges”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 1619–1627. DOI: 10.1109/CVPR.2016.179.

[148] Yanghao Li et al. *Benchmarking Detection Transfer Learning with Vision Transformers*. 2021. DOI: 10.48550/ARXIV.2111.11429. URL: https://arxiv.org/abs/2111.11429.

[149] Yangyan Li et al. “PointCNN: Convolution On X-Transformed Points”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper%5C_files/paper/2018/file/f5f8590cd58a54e94377e6ae2eded4d9-Paper.pdf.

[150] Yi Li et al. “DeepIM: Deep Iterative Matching for 6D Pose Estimation”. In: *Proceedpings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[151] Xinwu Liang, Hesheng Wang, and Weidong Chen. “Adaptive image-based visual servoing of wheeled mobile robots with fixed camera configuration”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 6199–6204. DOI: 10.1109/ICRA.2014.6907773.

[152] Feng Lin. *Robust control design: an optimal control approach*. John Wiley & Sons, 2007.

[153] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755.

[154] Diyi Liu et al. “Deep-Learning based Robust Edge Detection for Point Pair Feature-based Pose Estimation with Multiple Edge Appearance Models”. In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019, pp. 2920–2925. DOI: 10.1109/ROBIO49542.2019.8961752.

[155] Jian Liu et al. *Deep Learning-Based Object Pose Estimation: A Comprehensive Survey*. 2024. arXiv: 2405.07801 [cs.CV]. URL: https://arxiv.org/abs/2405.07801.

[156] Shijie Liu and Qian Zhang. “Model-based detection and pose estimation of textureless objects using CAD models”. In: *Proceedings of the International Conference on Industrial Informatics*. Springer. 2021, pp. 22–29.

[157] Shijie Liu et al. “A model-based approach for robust textureless object detection and pose estimation in industrial applications”. In: *International Conference on Industrial Informatics*. Springer. 2021, pp. 15–22.

[158] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46448-0.

[159] Weiping Liu et al. “Deep Learning on Point Clouds and Its Application: A Survey”. In: *Sensors* 19.19 (2019). ISSN: 1424-8220. DOI: 10.3390/s19194188. URL: https://www.mdpi.com/1424-8220/19/19/4188.

[160] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[161] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of Machine Learning Research* 9 (Nov. 2008), pp. 2579–2605.

[162] Zakariae Machkour, Daniel Ortiz-Arroyo, and Petar Durdevic. “Classical and Deep Learning based Visual Servoing Systems: a Survey on State of the Art”. In: *Journal of Intelligent & Robotic Systems* 104.1 (Dec. 2021), p. 11.

[163] Ezio Malis. “Survey of vision-based robot control”. In: *ENSIETA European Naval Ship Design Short Course, Brest, France* 41 (2002), p. 46.

[164] Ezio Malis and François Chaumette. “Position-Based Visual Servoing with a Motion Estimation Architecture”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 1999, pp. 247–252.

[165] Liao Maosheng. “Research and Implementation of Autonomous Navigation for Mobile Robots Based on SLAM Algorithm under ROS”. In: *Sensors* 24 (2024), p. 3080.

[166] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. “Pose Estimation for Augmented Reality: A Hands-On Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 22.12 (2016), pp. 2633–2651. DOI: 10.1109/TVCG.2015.2513408.

[167] D Marr and H K Nishihara. “Representation and recognition of the spatial organization of three-dimensional shapes”. en. In: *Proc. R. Soc. Lond.* 200.1140 (Feb. 1978), pp. 269–294.

[168] Roberto Martín Martín and Oliver Brock. "Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 2494–2501. DOI: 10.1109/IROS.2014.6942902.

[169] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. *The RBO Dataset of Articulated Objects and Interactions*. 2018. eprint: arXiv:1806.06465.

[170] Manuel Martinez, Alvaro Collet, and Siddhartha S. Srinivasa. "MOPED: A scalable and low latency object recognition and pose estimation system". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 2043–2049. DOI: 10.1109/ROBOT.2010.5509801.

[171] Zoltan-Csaba Marton et al. "General 3D modelling of novel objects from a single view". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 3700–3705. DOI: 10.1109/IROS.2010.5650434.

[172] Naresh Marturi et al. "Dynamic grasp and trajectory planning for moving objects". In: *Autonomous Robots* 43.5 (June 2019), pp. 1241–1256. ISSN: 1573-7527. DOI: 10.1007/s10514-018-9799-1. URL: https://doi.org/10.1007/s10514-018-9799-1.

[173] Naresh Marturi et al. "Towards advanced robotic manipulation for nuclear decommissioning: A pilot study on tele-operation and autonomy". In: *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. IEEE. 2016, pp. 1–8.

[174] Giorgia Marullo et al. "6D object position estimation from 2D images: a literature review". In: *Multimedia Tools and Applications* 82.16 (July 2023), pp. 24605–24643. ISSN: 1573-7721. DOI: 10.1007/s11042-022-14213-z. URL: https://doi.org/10.1007/s11042-022-14213-z.

[175] Matrox Imaging. *Matrox Imaging Library (MIL)*. Version 11.1. 2024. URL: https://www.matrox.com/en/imaging/products/software/matrox-imaging-library.

[176] Daniel Maturana and Sebastian Scherer. "VoxNet: A 3D Convolutional Neural Network for real-time object recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481.

[177] Nicolas Mellado, Dror Aiger, and Niloy J. Mitra. "Super 4PCS Fast Global Pointcloud Registration via Smart Indexing". In: *Computer Graphics Forum* 33.5 (2014), pp. 205–215. DOI: https://doi.org/10.1111/cgf.12446. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12446. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12446.

[178] D. Meyer, J. Denzler, and H. Niemann. "Model based extraction of articulated objects in image sequences for gait analysis". In: *Proceedings of International Conference on Image Processing*. Vol. 3. 1997, 78–81 vol.3. DOI: 10.1109/ICIP.1997.631988.

[179] Luca Minciullo et al. "DB-GAN: Boosting Object Recognition Under Strong Lighting Conditions". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 2938–2948. DOI: 10.1109/WACV48630.2021.00298.

[180] R. Miyata et al. "Object search using edge-AI based mobile robot". In: *Proceedings of the 6th international conference on intelligent informatics and biomedical sciences (ICIIBMS)* (2021), pp. 198–203.

[181] Kaichun Mo et al. "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[182] O. Monga and R. Deriche. "3D edge detection using recursive filtering: application to scanner images". In: *Proceedings CVPR '89: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 1989, pp. 28–35. DOI: 10.1109/CVPR.1989.37825.

[183] J. Moreno, E. Clotet, R. Lupiañez, et al. "Design, implementation and validation of the three-wheel holonomic motion system of the assistant personal robot (APR)". In: *Sensors* 16 (2016), p. 1658.

[184] Ahmed Ashraf Morgan et al. “Robots in Healthcare: a Scoping Review”. In: *Current Robotics Reports* 3.4 (Dec. 2022), pp. 271–280. ISSN: 2662-4087. DOI: 10.1007/s43154-022-00095-4. URL: https://doi.org/10.1007/s43154-022-00095-4.

[185] Takayasu Moriya et al. “Unsupervised segmentation of 3D medical images based on clustering and deep representation learning”. In: *Medical Imaging 2018: Biomedical Applications in Molecular, Structural, and Functional Imaging*. Ed. by Barjor Gimi and Andrzej Krol. Vol. 10578. International Society for Optics and Photonics. SPIE, 2018, p. 1057820. DOI: 10.1117/12.2293414. URL: https://doi.org/10.1117/12.2293414.

[186] Anastasios I. Mourikis and Stergios I. Roumeliotis. “Latent Space Autoencoders for Visual Servoing”. In: *Journal of Field Robotics* 38.8 (2021), pp. 1156–1174.

[187] M. Munir et al. “DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series”. In: *IEEE Access* 7 (2019), pp. 1991–2005. DOI: 10.1109/ACCESS.2018.2886457.

[188] MVTec Software GmbH. *MVTec HALCON*. Version 22.11. 2024. URL: https://www.mvtec.com/products/halcon/.

[189] Sridhar Narayan. “The generalized sigmoid activation function: Competitive supervised learning”. In: *Information Sciences* 99.1 (1997), pp. 69–82. ISSN: 0020-0255. DOI: https://doi.org/10.1016/S0020-0255(96)00200-9. URL: https://www.sciencedirect.com/science/article/pii/S0020025596002009.

[190] Pushmeet Kohli Nathan Silberman Derek Hoiem and Rob Fergus. “Indoor Segmentation and Support Inference from RGBD Images”. In: *Computer Vision – ECCV 2012*. 2012.

[191] Tan Nguyen et al. “Hybrid Visual Servoing: Advances and Applications in Complex Robotics Tasks”. In: *IEEE Transactions on Robotics and Automation* 39.4 (2023), pp. 2345–2358.

[192] Thanh Nguyen, Binh Le, and An Pham. “Advancements in Direct Visual Servoing for Dynamic and Unstructured Environments”. In: *IEEE Access* 11 (2023), pp. 56485–56500.

[193] Thao Nguyen et al. “Object Detection Using Scale Invariant Feature Transform”. In: *Genetic and Evolutionary Computing*. Ed. by Jeng-Shyang Pan, Pavel Krömer, and Václav Snášel. Cham: Springer International Publishing, 2014, pp. 65–72. ISBN: 978-3-319-01796-9.

[194] Duy Nguyen-Tuong and Jan Peters. “Model learning for robot control: a survey”. In: *Cognitive Processing* 12.4 (Nov. 2011), pp. 319–340. ISSN: 1612-4790. DOI: 10.1007/s10339-011-0404-1. URL: https://doi.org/10.1007/s10339-011-0404-1.

[195] Huan Ni et al. “Edge Detection and Feature Line Tracing in 3D-Point Clouds by Analyzing Geometric Properties of Neighborhoods”. In: *Remote Sensing* 8.9 (2016). DOI: 10.3390/rs8090710.

[196] Marcin Novotni and Reinhard Klein. “3D zernike descriptors for content based shape retrieval”. In: *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*. SM ’03. Seattle, Washington, USA: Association for Computing Machinery, 2003, pp. 216–225. ISBN: 1581137060. DOI: 10.1145/781606.781639. URL: https://doi.org/10.1145/781606.781639.

[197] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. “Ridge-Valley Lines on Meshes via Implicit Surface Fitting”. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 609–612. DOI: 10.1145/1015706.1015768.

[198] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV]. URL: https://arxiv.org/abs/2304.07193.

[199] Valerio Ortenzi et al. “Vision-Based Framework to Estimate Robot Configuration and Kinematic Constraints”. In: *IEEE/ASME Transactions on Mechatronics* 23.5 (2018), pp. 2402–2412. DOI: 10.1109/TMECH.2018.2865758.

[200] Valerio Ortenzi et al. “Vision-guided state estimation and control of robotic manipulators which lack proprioceptive sensors”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 3567–3574. DOI: 10.1109/IROS.2016.7759525.

[201] Nobuyuki Otsu. “A threshold selection method from gray-level histograms”. In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[202] Keunhong Park et al. “LatentFusion: End-to-End Differentiable Reconstruction and Rendering for Unseen Object Pose Estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[203] Petra Pejić et al. “Articulated Objects: From Detection to Manipulation—Survey”. In: *Intelligent Autonomous Systems 17*. Ed. by Ivan Petrovic, Emanuele Menegatti, and Ivan Marković. Cham: Springer Nature Switzerland, 2023, pp. 495–508. ISBN: 978-3-031-22216-0.

[204] Sida Peng et al. “PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[205] Xin Peng and Shuang Li. “Fusion of RGB-D and point cloud data for textureless object recognition and pose estimation”. In: *Proceedings of the International Conference on 3D Vision*. Springer. 2021, pp. 111–118.

[206] Zachary Pezzementi, Sandrine Voros, and Gregory D. Hager. “Articulated object tracking by rendering consistent appearance parts”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 3940–3947. DOI: 10.1109/ROBOT.2009.5152374.

[207] Cody J. Phillips, Matthieu Lecce, and Kostas Daniilidis. “Seeing Glassware: from Edge Detection to Pose Estimation and Shape Recovery”. In: *Proceedings of Robotics: Science and Systems*. AnnArbor, Michigan, June 2016. DOI: 10.15607/RSS.2016.XII.021.

[208] Paolo Piccinini, Andrea Prati, and Rita Cucchiara. “Real-time object detection and localization with SIFT-based clustering”. In: *Image and Vision Computing* 30.8 (2012). Special Section: Opinion Papers, pp. 573–587. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2012.06.004. URL: https://www.sciencedirect.com/science/article/pii/S0262885612000923.

[209] Ana Puljčan, Domagoj Zoraja, and Tomislav Petković. "Simulation of Structured Light 3D Scanning using Blender". In: *2022 International Symposium ELMAR*. 2022, pp. 215–220. DOI: 10.1109/ELMAR55880.2022.9899809.

[210] Charles R Qi et al. "Deep Hough voting for 3D object detection in point clouds". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9277–9286.

[211] Charles R. Qi et al. "Frustum PointNets for 3D Object Detection From RGB-D Data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[212] Charles R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[213] Charles Ruizhongtai Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper%5C_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[214] Shengyi Qian et al. "Understanding 3D Object Articulation in Internet Videos". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 1599–1609.

[215] Mahdi Rad and Vincent Lepetit. "BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects Without Using Depth". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[216] R. Raja and S. Kumar. "A Hybrid Image Based Visual Servoing for a Manipulator Using Kinect". In: *Proceedings of the Advances in Robotics*. AIR '17. New Delhi, India: Association for Computing Machinery, 2017. ISBN: 9781450352949. DOI: 10.1145/3132446.3134916. URL: https://doi.org/10.1145/3132446.3134916.

[217] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper%5C_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[218] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[219] Patrick Rives and Eric Marchand. “Position-Based Visual Servoing Using a Robust 3D Tracker”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 1997, pp. 1224–1230.

[220] International Federation of Robotics. “World Robotics Report 2020”. In: *IFR* (2020).

[221] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

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

[223] Radu Bogdan Rusu et al. “Aligning point cloud views using persistent feature histograms”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 3384–3391. DOI: 10.1109/IROS.2008.4650967.

[224] Artsiom Sanakoyeu, Miguel A. Bautista, and Björn Ommer. “Deep unsupervised learning of visual similarities”. In: *Pattern Recognition* 78 (2018), pp. 331–343. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2018.01.036. URL: https://www.sciencedirect.com/science/article/pii/S0031320318300293.

[225] Soumik Sarkar et al. *Occlusion Edge Detection in RGB-D Frames using Deep Convolutional Networks*. 2015. arXiv: 1412.7007 [cs.CV]. URL: http://arxiv.org/abs/1412.7007.

[226] Lars Schmarje et al. *A survey on Semi-, Self- and Unsupervised Learning for Image Classification*. 2021. arXiv: 2002.08721 [cs.CV]. URL: https://arxiv.org/abs/2002.08721.

[227] Aleksandr V. Segal, Dirk Hähnel, and Sebastian Thrun. "Generalized-ICP". In: *Robotics: Science and Systems*. 2009. URL: https://api.semanticscholar.org/CorpusID:231748613.

[228] Hooman Shariati et al. "Towards Autonomous Mining via Intelligent Excavators". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.

[229] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 640–651. DOI: 10.1109/TPAMI.2016.2572683.

[230] 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 (CVPR)*. June 2019.

[231] Shaoshuai Shi et al. "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[232] Mohamad Kazem Shirani Faradonbeh, Ambuj Tewari, and George Michailidis. "On adaptive Linear–Quadratic regulators". In: *Automatica* 117 (2020), p. 108982. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2020.108982. URL: https://www.sciencedirect.com/science/article/pii/S0005109820301801.

[233] Zhenyu Shu et al. "Unsupervised 3D shape segmentation and co-segmentation via deep learning". In: *Computer Aided Geometric Design* 43 (2016). Geometric Modeling and Processing 2016, pp. 39–52. ISSN: 0167-8396. DOI: https://doi.org/10.1016/j.cagd.2016.02.015. URL: https://www.sciencedirect.com/science/article/pii/S0167839616300164.

[234] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[235] John Smith, Wei Zhang, and Xueyang Li. "Challenges and Opportunities in Learning-Based Visual Servoing: A Review". In: *Journal of Intelligent & Robotic Systems* 106.2 (2022), pp. 321–340.

[236] Robert Smith, James Anderson, and Steven Taylor. "Direct Visual Servoing for Robotics: A Survey". In: *IEEE Transactions on Robotics* 35.4 (2019), pp. 927–944.

[237] Osama B. Bahwal Soliman A. Al-Walaie and Sarah S. Alduayj. "Emerging Robotic Technologies for Oil and Gas Operations". In: *Journal of Petroleum Technology* (2021).

[238] Shuran Song and Jianxiong Xiao. "Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[239] Robert Sparrow and Mark Howard. "Robots in agriculture: prospects, impacts, ethics, and policy". In: *Precision Agriculture* 22.3 (June 2021), pp. 818–833. ISSN: 1573-1618. DOI: 10.1007/s11119-020-09757-9. URL: https://doi.org/10.1007/s11119-020-09757-9.

[240] Luciano Spinello and Kai O. Arras. "People detection in RGB-D data". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 3838–3843. DOI: 10.1109/IROS.2011.6095074.

[241] Bastian Steder et al. "NARF: 3D range image features for object recognition". In: *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 44. Citeseer. 2010, p. 2.

[242] Gilbert Strang. *Introduction to Linear Algebra*. Fourth. Wellesley, MA: Wellesley-Cambridge Press, 2009.

[243] Jürgen Sturm et al. "Towards understanding articulated objects". In: *Proc. of the Workshop on Robot Manipulation at Robotics: Science and Systems Conference (RSS)*. 2009.

[244] Zhiqiang Sui et al. “SUM: Sequential scene understanding and manipulation”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 3281–3288. DOI: 10.1109/IROS.2017.8206164.

[245] Rui Sun et al. “Survey of Image Edge Detection”. In: *Frontiers in Signal Processing* 2 (2022). ISSN: 2673-8198. DOI: 10.3389/frsip.2022.826967. URL: https://www.frontiersin.org/articles/10.3389/frsip.2022.826967.

[246] Thai Leang Sung and Hyo Jong Lee. “Depth edge detection using edge-preserving filter and morphological operations”. In: *International Journal of System Assurance Engineering and Management* 11.4 (Aug. 2020), pp. 812–817. ISSN: 0976-4348. DOI: 10.1007/s13198-019-00881-y. URL: https://doi.org/10.1007/s13198-019-00881-y.

[247] Giovanni Sutanto et al. “Learning Latent Space Dynamics for Tactile Servoing”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 3622–3628. DOI: 10.1109/ICRA.2019.8793520.

[248] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV]. URL: https://arxiv.org/abs/1512.00567.

[249] B. Tamadazte et al. “CAD Model-based Tracking and 3D Visual-based Control for MEMS Microassembly”. In: *The International Journal of Robotics Research* 29.11 (2010), pp. 1416–1434. DOI: 10.1177/0278364910376033. eprint: https://doi.org/10.1177/0278364910376033. URL: https://doi.org/10.1177/0278364910376033.

[250] Kok K Tan, Qing-Guo Wang, and Chang C Hang. *Advances in PID control*. Springer Science & Business Media, 2012.

[251] Alykhan Tejani et al. “Latent-class hough forests for 3d object detection and pose estimation”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 462–477.

[252] Alykhan Tejani et al. “Latent-Class Hough Forests for 6DoF Object Pose Estimation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.1 (2018), pp. 119–132. DOI: 10.1109/TPAMI.2017.2665623.

[253] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. "Real-time seamless single shot 6D object pose prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 292–301.

[254] Hasan Tercan, Alexandro Guajardo, and Tobias Meisen. "Industrial Transfer Learning: Boosting Machine Learning in Production". In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Vol. 1. 2019, pp. 274–279. DOI: 10.1109/INDIN41052.2019.8972099.

[255] Andreas Themelis, Lorenzo Stella, and Panagiotis Patrinos. "Douglas–Rachford splitting and ADMM for nonconvex optimization: accelerated and Newton-type linesearch algorithms". In: *Computational Optimization and Applications* 82.2 (June 2022), pp. 395–440. ISSN: 1573-2894. DOI: 10.1007/s10589-022-00366-y. URL: https://doi.org/10.1007/s10589-022-00366-y.

[256] Hugues Thomas et al. "KPConv: Flexible and Deformable Convolution for Point Clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[257] B. Thuilot et al. "Position based visual servoing: keeping the object in the field of vision". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, 1624–1629 vol.2. DOI: 10.1109/ROBOT.2002.1014775.

[258] Meng Tian et al. "Robust 6D Object Pose Estimation by Learning RGB-D Features". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 6218–6224. DOI: 10.1109/ICRA40945.2020.9197555.

[259] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Unique Signatures of Histograms for Local Surface Description". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 356–369. ISBN: 978-3-642-15558-1.

[260] Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review". In: *Sensors* 21 (2021), p. 2140.

[261] Pascal Vincent et al. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”. In: *J. Mach. Learn. Res.* 11 (Dec. 2010), pp. 3371–3408. ISSN: 1532-4435.

[262] Paul Viola and Michael J. Jones. “Robust Real-Time Face Detection”. In: *International Journal of Computer Vision* 57.2 (May 2004), pp. 137–154. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000013087.49260.fb. URL: https://doi.org/10.1023/B:VISI.0000013087.49260.fb.

[263] Andrey Voynov, Stanislav Morozov, and Artem Babenko. “Object Segmentation Without Labels with Large-Scale Generative Models”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR, July 2021, pp. 10596–10606.

[264] Bin Wang, Fan Zhong, and Xueying Qin. “Robust edge-based 3D object tracking with direction-based pose validation”. In: *Multimedia Tools and Applications* 78.9 (May 2019), pp. 12307–12331. ISSN: 1573-7721. DOI: 10.1007/s11042-018-6727-5. URL: https://doi.org/10.1007/s11042-018-6727-5.

[265] Chen Wang et al. “DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[266] Haibin Wang, Jian Zhang, and Yong Gao. “Direct Visual Servoing Using Image Moments Without Explicit Feature Extraction”. In: *Journal of Field Robotics* 37.1 (2020), pp. 115–130.

[267] Weiyue Wang and Ulrich Neumann. “Depth-aware CNN for RGB-D Segmentation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[268] Xinlong Wang et al. *FreeSOLO: Learning to Segment Objects without Annotations*. 2022. arXiv: 2202.12181 [cs.CV].

[269] Yangfan Wang et al. “Recent advances in 3D object detection based on RGB-D: A survey”. In: *Displays* 70 (2021), p. 102077. ISSN: 0141-9382. DOI: https://doi.org/10.

1016/j.displa.2021.102077. URL: https://www.sciencedirect.com/science/article/pii/S0141938221000846.

[270] Yue Wang and Justin M Solomon. "PRNet: Self-Supervised Learning for Partial-to-Partial Registration". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper%5C_files/paper/2019/file/ebad33b3c9fa1d10327bb55f9e79e2f3-Paper.pdf.

[271] Yue Wang and Justin M. Solomon. "Deep Closest Point: Learning Representations for Point Cloud Registration". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[272] Yue Wang et al. "Dynamic Graph CNN for Learning on Point Clouds". In: *ACM Trans. Graph.* 38.5 (Oct. 2019). ISSN: 0730-0301. DOI: 10.1145/3326362. URL: https://doi.org/10.1145/3326362.

[273] Ziyin Wang. "Unsupervised Visual Knowledge Discovery and Accumulation in Dynamic Environments". In: *Purdue University* (Nov. 2019). DOI: 10.25394/PGS.10298894.v1. URL: https://hammer.purdue.edu/articles/thesis/Unsupervised%5C_Visual%5C_Knowledge%5C_Discovery%5C_and%5C_Accumulation%5C_in%5C_Dynamic%5C_Environments/10298894.

[274] Liyana Wijayathunga, Alexander Rassau, and Douglas Chai. "Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review". In: *Applied Sciences* 13.17 (2023). ISSN: 2076-3417. DOI: 10.3390/app13179877. URL: https://www.mdpi.com/2076-3417/13/17/9877.

[275] W.J. Wilson, C.C. Williams Hulls, and G.S. Bell. "Relative end-effector control using Cartesian position based visual servoing". In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 684–696. DOI: 10.1109/70.538974.

[276] Jay M. Wong et al. *SegICP-DSR: Dense Semantic Scene Reconstruction and Registration*. 2017. arXiv: 1711.02216 [cs.RO]. URL: https://arxiv.org/abs/1711.02216.

[277] Yi Wu et al. “Deep Learning-Based Visual Servoing with Implicit Representation Learning for Dexterous Manipulation”. In: *Robotics and Autonomous Systems* 150 (2022), pp. 556–566.

[278] Zhirong Wu et al. “3D ShapeNets: A Deep Representation for Volumetric Shapes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[279] Peter R. Wurman, Raffaello D’Andrea, and Mick Mountz. “Coordinating hundreds of cooperative, autonomous vehicles in warehouses”. In: *AI Magazine* 29 (2008), pp. 9–20.

[280] Yu Xiang et al. “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes”. In: *arXiv preprint arXiv:1711.00199* (2017).

[281] Yu Xiang et al. “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes”. In: *Robotics: Science and Systems (RSS)*. 2018.

[282] Zhibin Xiao et al. “FETNet: Feature Exchange Transformer Network for RGB-D Object Detection”. In: *British Machine Vision Conference*. 2021. URL: https://api.semanticscholar.org/CorpusID:249892200.

[283] Junyuan Xie, Ross Girshick, and Ali Farhadi. “Unsupervised deep embedding for clustering analysis”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. Vol. 48. JMLR.org, 2016, pp. 478–487.

[284] Saining Xie and Zhuowen Tu. “Holistically-nested edge detection”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1395–1403.

[285] Xuejun Xing et al. “Efficient Single Correspondence Voting for Point Cloud Registration”. In: *IEEE Transactions on Image Processing* 33 (2024), pp. 2116–2130. DOI: 10.1109/TIP.2024.3374120.

[286] Tifan Xiong and Mingyuan Wu. “A Review of Sensing Technologies for Indoor Autonomous Mobile Robots”. In: *Sensors* 24 (2024), p. 1222.

[287] Zhiqiang Xu et al. “Hybrid Visual Servoing Control of a 6-DOF Manipulator Based on Improved Visual Features and Estimated Depth”. In: *IEEE Access* 11 (2023), pp. 69277–69288.

[288] H. Xue, S. Zhang, and D. Cai. “Depth Image Inpainting: Improving Low Rank Matrix Completion With Low Gradient Regularization”. In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4311–4320.

[289] Fan Yang, Chen Xue, and Wei Li. “Probabilistic Framework for Pose Estimation under Occlusion and Clutter”. In: *IEEE Transactions on Robotics* 38.3 (2022), pp. 1540–1550.

[290] Guang-Zhong Yang, Bradley J. Nelson, Robin R. Murphy, et al. “Medical robotics–Regulatory, ethical, and legal considerations for increasing levels of autonomy”. In: *Science Robotics* 2 (2017), pp. 1–12.

[291] Heng Yang, Jingnan Shi, and Luca Carlone. “TEASER: Fast and Certifiable Point Cloud Registration”. In: *IEEE Transactions on Robotics* 37.2 (2021), pp. 314–333. DOI: 10.1109/TRO.2020.3033695.

[292] Jiaolong Yang et al. “Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11 (2016), pp. 2241–2254. DOI: 10.1109/TPAMI.2015.2513405.

[293] Jun Yang, Jian Yao, and Steven L. Waslander. *Active Pose Refinement for Textureless Shiny Objects using the Structured Light Camera*. 2023. arXiv: 2308.14665 [cs.RO]. URL: https://arxiv.org/abs/2308.14665.

[294] X. Yang, J. Sun, and W. Diao. “Depth Image Inpainting for RGB-D Camera Based on Light Field EPI”. In: *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*. 2018, pp. 214–219.

[295] Xu Yang et al. “A Multi-stage 6D Object Pose Estimation Method of Texture-less Objects Based on Sparse Line Features”. In: *2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 2022, pp. 0221–0225. DOI: 10.1109/IEEM55944.2022.9989794.

[296] Zi Jian Yew and Gim Hee Lee. "3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[297] Zi Jian Yew and Gim Hee Lee. "RPM-Net: Robust Point Matching Using Learned Features". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[298] Bowen Yin et al. *DFormer: Rethinking RGBD Representation Learning for Semantic Segmentation*. 2024. arXiv: 2309.09668 [cs.CV]. URL: https://arxiv.org/abs/2309.09668.

[299] Chunyan Yu, Seth Hutchinson, and Alessandro De Luca. "Image-Based Visual Servoing for Coordinated Multirobot Systems". In: *IEEE Transactions on Robotics* 30.5 (2014), pp. 1177–1189.

[300] Wentao Yuan et al. "DeepGMR: Learning Latent Gaussian Mixture Models for Registration". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 733–750. ISBN: 978-3-030-58558-7.

[301] Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper%5C_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

[302] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. "DPOD: 6D pose object detector and refiner". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1941–1950.

[303] Andy Zeng et al. "3DMatch: Learning Local Geometric Descriptors From RGB-D Reconstructions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[304] Vicky Zeng et al. “Visual Identification of Articulated Object Parts”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 2443–2450. DOI: 10.1109/IROS51168.2021.9636054.

[305] Shiyi Zhang et al. “A Visual Servoing Method based on Point Cloud”. In: *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. 2020, pp. 369–374. DOI: 10.1109/RCAR49640.2020.9303277.

[306] Y. Zhang and T. Funkhouser. “Deep Depth Completion of a Single RGB-D Image”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 175–185.

[307] Yuqi Zhang et al. “EANet: Edge-Attention 6D Pose Estimation Network for Texture-Less Objects”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–13. DOI: 10.1109/TIM.2022.3150568.

[308] Heng Zhao et al. “Learning Symmetry-Aware Geometry Correspondences for 6D Object Pose Estimation”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 13999–14008. DOI: 10.1109/ICCV51070.2023.01291.

[309] Tianyu Zhao et al. “Point Cloud Segmentation Based on FPFH Features”. In: *Proceedings of 2016 Chinese Intelligent Systems Conference*. Ed. by Yingmin Jia et al. Singapore: Springer Singapore, 2016, pp. 427–436. ISBN: 978-981-10-2335-4.

[310] Xiao Zhao and Hua Li. “Robust real-time object detection and pose estimation for industrial applications”. In: *Neurocomputing* 493 (2022), pp. 433–446.

[311] Lei Zhou and Weiyufeng Wei. “DIC: Deep Image Clustering for Unsupervised Image Segmentation”. In: *IEEE Access* 8 (2020), pp. 34481–34491. DOI: 10.1109/ACCESS.2020.2974496.

[312] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Fast Global Registration”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 766–782. ISBN: 978-3-319-46475-6.

[313] Yanling Zhou, Runhua Wang, and Xuebo Zhang. "A Hand-Eye Calibration Method based on Nonlinear Joint Optimization for RGBD Cameras". In: *2022 34th Chinese Control and Decision Conference (CCDC)*. 2022, pp. 3523–3528. DOI: 10.1109/CCDC55256.2022.10034272.

[314] Yi Zhou et al. "On the Continuity of Rotation Representations in Neural Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5738–5746. DOI: 10.1109/CVPR.2019.00589.

[315] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

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

[317] Zhengxia Zou et al. "Object Detection in 20 Years: A Survey". In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276. DOI: 10.1109/JPROC.2023.3238524.

Appendix A

Evaluation matrices and computational complexity derivation for 3D edge detection method

Preliminary remarks

The research in this appendix was published in the following paper:

- **Aggarwal, A.**, Stolkin, R. & Marturi, N., Unsupervised learning-based approach for detecting 3D edges in depth maps. Sci Rep 14, 796 (2024). https://doi.org/10.1038/s41598-023-50899-3

Note: Some parts of this appendix, including kernel functions, evaluation metrics, and computational complexity derivation are adapted from [11].

A.1 Kernel functions

The four kernel functions employed in 3D edge detection to extract gradient-based edge features from depth maps include Sobel, Roberts, Prewitt, and Laplacian of Gaussian (LoG) [13]. These kernels are designed to identify abrupt changes in depth at specific points in the depth maps

by calculating derivatives in both horizontal and vertical directions. The magnitude of the gradients is computed using the root sum squared (RSS) method. A threshold is then applied to differentiate edges from non-edges based on the computed derivatives. Each of these kernels is detailed below.

Sobel kernel is an isotropic 3×3 gradient kernel used to approximate derivatives in a depth map through the convolution operation. The Sobel kernel in the X and Y directions is defined as follows:

$$\mathbf{G}_X^s = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}, \quad \mathbf{G}_Y^s = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

The Sobel kernel is more sensitive to detecting diagonal edges than to horizontal and vertical edges. It has been observed to perform well for most depth maps but is prone to high-value variations, which can lead to inaccuracies in edge detection.

Roberts kernel is a 2×2 gradient kernel designed to calculate discrete differentiation at each point of the depth map. It provides intensity differences in the diagonal direction. The Roberts kernel for the X and Y directions is given as follows:

$$\mathbf{G}_X^r = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \mathbf{G}_Y^r = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}.$$

Due to its small size and integer values, the Roberts kernel is highly sensitive to noise. This sensitivity is observed in its computations, making it most effective when applied to binary images, where noise levels are minimal.

Prewitt kernel is a 3×3 gradient kernel that operates similarly to the Sobel kernel but uses the maximal directional gradient to identify edges. The Prewitt kernel in the X and Y directions

is defined as follows:

$$\mathbf{G}_X^p = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix}, \quad \mathbf{G}_Y^p = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}.$$

While the Prewitt kernel is simpler than the Sobel kernel, it is more susceptible to noise, which can affect its accuracy in edge detection, particularly in depth maps with high noise levels.

Laplacian of Gaussian (LoG) kernel is a 3×3 kernel that computes second-order derivatives, providing a sharp response at points where depth values experience significant disruptions. Two different LoG kernels were utilised together, and their magnitude was calculated using the RSS method. These kernels are defined as follows:

$$\mathbf{G}_1^l = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad \mathbf{G}_2^l = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

The first LoG kernel is particularly effective in calculating gradients for horizontal and vertical edges, while the second kernel excels in identifying variations in all directions. Together, these kernels enhance the ability to detect complex edge patterns in depth maps, though they are also more prone to capturing noise as part of the detected edges.

A.2 Evaluation metrics

In this section, the two evaluation metrics, namely pose estimation and F-measure, which are utilised for the quantitative analysis of the proposed 3D edge detection method, are thoroughly discussed.

A.2.1 Pose estimation

As outlined in Section 3.5.3, poses are computed by registering the detected edges with the point cloud of the model using the Random Sample Consensus (RANSAC) algorithm. These initial estimates are subsequently refined using the Iterative Closest Point (ICP) algorithm. The outcome is a homogeneous transformation matrix $\mathbf{P}$ of size (4×4), which comprises a (3×3) rotation matrix $\mathbf{R}$ and a (3×1) translation vector $\mathbf{t}$. The pose matrix is represented as follows:

$$\mathbf{P} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}. \tag{A.1}$$

Let the predicted pose after registration be denoted as $\mathbf{P}^{pred}$. Several metrics are then employed to quantify the error between the predicted pose and the ground truth pose $\mathbf{P}^{true}$, as detailed below.

Frobenius norm of pose difference

This metric, denoted by $||A||_F$, provides the squared sum of the element-wise differences between the two matrices. The Frobenius norm is computed to quantify the difference between $\mathbf{P}^{true}$ and $\mathbf{P}^{pred}$ and is calculated as follows:

$$||A||_F = \sqrt{\sum_{i}^{4} \sum_{j}^{4} (\mathbf{P}_{i,j}^{true} - \mathbf{P}_{i,j}^{pred})^2}. \tag{A.2}$$

Rotational error

The rotational error, represented by $\mathbf{R}_{err}$, measures the difference in rotation between two pose matrices. This metric is computed by taking the *arccos* of the phase difference between the two rotational matrices along the diagonal components. The rotational error is calculated as:

$$\mathbf{R}_{err} = arccos((Tr(\mathbf{R}^{pred}(\mathbf{R}^{true})^{-1}) - 1)/2), \tag{A.3}$$

where $Tr(\cdot)$ represents the trace operator. The axis-angle representation is used by this metric to compute the angular error between the two rotations.

Translation error

The translation error, denoted by $\mathbf{T}_{err}$, quantifies the difference in translation between the ground truth and predicted translation vectors. This metric is computed by measuring the Euclidean distance [66] between the two vectors and is expressed as:

$$\mathbf{T}_{err} = \sqrt{\sum(\mathbf{t}^{true} - \mathbf{t}^{pred})^2}. \tag{A.4}$$

Average distance of model points

The average distance of model points, represented by $\boldsymbol{Z}_E$, is calculated as the average point-to-point Euclidean distance [66] between the two models of the objects transformed by $\mathbf{P}^{pred}$ and $\mathbf{P}^{true}$. This metric is particularly applicable when the model has an indistinguishable view along all directions. The metric is computed as:

$$\boldsymbol{Z}_E = \frac{1}{M}(\sqrt{\sum(\mathbf{P}^{true}\mathbf{x} - \mathbf{P}^{pred}\mathbf{x})^2}), \tag{A.5}$$

where M denotes the number of points in the model point cloud, and $\mathbf{x}$ is a homogeneous point belonging to the point cloud.

A.2.2 F-measure

The F-measure is used to assess the accuracy of the model with respect to its ground truth. For 3D edge detection, the F-measure quantifies the accuracy of the 3D edges predicted by the trained model compared to the ground truth edges. It is computed as the harmonic mean of precision and recall, calculated as:

$$F_{meas} = \frac{2PrecRec}{Prec + Rec}, \tag{A.6}$$

where $Prec$ denotes precision and Rec denotes recall. Each of these measures is computed using a distance threshold, which is considered to be one point in each direction. The details of these metrics are discussed below.

Precision

Precision is defined as the ratio of the number of correctly identified edges to the total number of identified edges. It serves as a measure of the detection accuracy of the trained model. In a distance threshold-based computation, the correctness of each predicted edge relative to the ground truth edge is assessed within a distance of one point in all directions. Precision is computed as:

$$Prec = \frac{1}{|\mathbf{E}^p|} \sum_{e \in \mathbf{E}^p} (dist(e, \mathbf{E}^t) < \theta), \tag{A.7}$$

where $\mathbf{E}^t$ represents the set of ground truth edge points, $\mathbf{E}^p$ represents the set of predicted edges, θ is the distance threshold for considering successful edge detection, and $dist(\cdot)$ denotes the distance calculation function.

Recall

Recall is defined as the ratio of the number of correctly detected edge matches to the total number of edge points in the ground truth. It represents the reproducibility of the trained model. The distance threshold for recall is computed in a manner similar to precision and is expressed as:

$$Rec = \frac{1}{|\mathbf{E}_t|} \sum_{e \in \mathbf{E}_t} (dist(e, \mathbf{E}^p) < \theta). \tag{A.8}$$

A.3 Computational complexity derivation

The computational complexity of an algorithm is determined by calculating the sum of the total number of basic arithmetic operations required for its computation. In this section, the total number of operations involved in the base components of the 3D edge detection network is first discussed. Subsequently, the overall computational complexity of the network is derived by

combining these individual component complexities.

A.3.1 Base operations

Convolution layer

In the network, a convolution layer applies a fixed-size convolution kernel over the entire input in a moving window format to generate the output. The computational complexity of a convolution layer is dependent on several parameters, including the input data size (height h, width w, and channels c), filter size n_f, and kernel size n_k. The total number of multiplications $N_{\text{conv}}^{\text{mult}}$ and additions $N_{\text{conv}}^{\text{add}}$ required for the convolution operation with a single filter are given by:

$$\begin{aligned} N_{\text{conv}}^{\text{mult}} &= h \times w \times (n_k \times c), \\ N_{\text{conv}}^{\text{add}} &= h \times w \times (n_k \times c - 1). \end{aligned} \tag{A.9}$$

In addition to these operations, a bias term is added for each convolution operation. The number of bias computations $N_{\text{conv}}^{\text{bias}}$ is:

$$N_{\text{conv}}^{\text{bias}} = h \times w \times 1. \tag{A.10}$$

Thus, the total number of operations N_{conv} required for a single filter in a convolution layer is calculated as:

$$\begin{aligned} N_{\text{conv}} &= N_{\text{conv}}^{\text{mult}} + N_{\text{conv}}^{\text{add}} + N_{\text{conv}}^{\text{bias}}, \\ N_{\text{conv}} &= h \times w \times (2 \times n_k \times c). \end{aligned} \tag{A.11}$$

When considering all the filters, the total number of operations becomes:

$$N_{\text{conv}} = h \times w \times (2 \times n_k \times c) \times n_f. \tag{A.12}$$

Activation functions

Activation functions are employed to introduce non-linearity into the learning process. In the 3D edge detection method, two activation functions are used: ReLu and student-T. The computational complexity of the ReLu activation function N_{relu} applied to an input of size

(h, w, n_f) is calculated as:

$$N_{\text{relu}} = h \times w \times n_f. \tag{A.13}$$

Similarly, the computational complexity of the student-T activation function N_{stuT}, with 2 clusters, is computed as:

$$\begin{aligned} N_{\text{stuT}}^{nume} &= h \times w \times (n_f + 4), \\ N_{\text{stuT}}^{deno} &= h \times w \times (n_f + 4), \\ N_{\text{stuT}} &= 2 \times (N_{\text{stuT}}^{nume} + N_{\text{stuT}}^{deno} + 1), \\ N_{\text{stuT}} &= 4 \times h \times w \times n_f + 16 \times h \times w + 2, \end{aligned} \tag{A.14}$$

where N_{stuT}^{nume} represents the number of computations from the numerator, and N_{stuT}^{deno} represents the number of computations from the denominator of the student-T equation.

Edge thinning layer

The edge thinning layer performs a series of minimum pool operations with a vectorised kernel size n_m. A minimum pool operation is akin to a max pool operation, where all values are first negated, followed by max pooling, and then the values are inverted again. For an input of size (h, w, c), the number of computations required for the minimum pool operation N_{minpool} is calculated as:

$$N_{\text{minpool}} = h \times w \times ((n_m - 1) \times c + 2). \tag{A.15}$$

A combination of min pool operations along the X, Y, and XY directions is employed to eliminate noise and narrow down the edge gradients. Additional operations are performed to merge the results of the three minpool operations. The total computational complexity of the edge thinning operation N_{edgeThin} is then calculated as:

$$\begin{aligned} N_{\text{minpool}}^{\text{XY}} &= 2 \times N_{\text{minpool}}, \\ N_{\text{edgeThin}}^{\text{merge}} &= 2 \times h \times w, \\ N_{\text{edgeThin}} &= N_{\text{minpool}}^{\text{X}} + N_{\text{minpool}}^{\text{Y}} + N_{\text{minpool}}^{\text{XY}} + N_{\text{edgeThin}}^{\text{merge}}, \end{aligned} \tag{A.16}$$

where $N^{\mathrm{X}}_{\mathrm{minpool}}$, $N^{\mathrm{Y}}_{\mathrm{minpool}}$, and $N^{\mathrm{XY}}_{\mathrm{minpool}}$ represent the number of operations for minpool in the X, Y, and XY directions, respectively. $N^{\mathrm{merge}}_{\mathrm{edgeThin}}$ represents the number of computations required to merge the outputs of the minpool operations in the edge thinning layer.

Normalisation layer

In the proposed approach, *min-max* normalisation is employed. The computational complexity of the normalisation layer N_{norm} for an input of size (h, w, c) is calculated as:

$$\begin{aligned} N^{\mathrm{min}}_{\mathrm{norm}} &= h \times w, \\ N^{\mathrm{max}}_{\mathrm{norm}} &= h \times w, \\ N_{\mathrm{norm}} &= \left(N^{\mathrm{min}}_{\mathrm{norm}} + N^{\mathrm{max}}_{\mathrm{norm}} + 1 + 2 \times (h \times w) \right) \times c, \end{aligned} \tag{A.17}$$

where $N^{\mathrm{min}}_{\mathrm{norm}}$ and $N^{\mathrm{max}}_{\mathrm{norm}}$ represent the computations required to calculate the minimum and maximum within the given input.

A.3.2 Network layers

Encoder layers

The encoder layers consist of three convolution operations with ReLu activation applied to depth images of respective sizes. Using equations (A.12) and (A.13), the total number of operations performed in the encoder layers is calculated. For the 3D edge detection network, $n_k = (9, 9, 9)$, $n_f = (16, 16, 16)$, $c = (1, 16, 16)$, $h = (H, H/2, H/4)$, and $w = (W, W/2, W/4)$ for each of the encoder layers, respectively. The total computations N_{enc} for the encoder block are given by:

$$\begin{aligned} N^{1}_{\mathrm{enc}} &= H \times W \times (2 \times 9 \times 1) \times 16 + H \times W \times 16, \\ N^{2}_{\mathrm{enc}} &= \frac{H}{2} \times \frac{W}{2} \times (2 \times 9 \times 16) \times 16 + \frac{H}{2} \times \frac{W}{2} \times 16, \\ N^{3}_{\mathrm{enc}} &= \frac{H}{4} \times \frac{W}{4} \times (2 \times 9 \times 16) \times 16 + \frac{H}{4} \times \frac{W}{4} \times 16, \\ N_{\mathrm{enc}} &= N^{1}_{\mathrm{enc}} + N^{2}_{\mathrm{enc}} + N^{3}_{\mathrm{enc}}, \\ &= 1749 \times H \times W. \end{aligned} \tag{A.18}$$

Here, N^1_{enc}, N^2_{enc}, and N^3_{enc} represent the computations from each layer. Considering the asymptotic bound on the encoder's performance, the computational complexity in terms of Big-O notation is expressed as $O(HW)$.

Multi-size split layers

These layers function similarly to the encoder, involving convolution operations. Therefore, the computations are equivalent to N_{enc}. In this case, $n_k = (1, 1, 1)$, $n_f = (1, 1, 1)$, $c = (16, 16, 16)$, $h = (H, H/2, H/4)$, and $w = (W, W/2, W/4)$ for each of the side outputs, respectively. The total number of computations N_{mult} for the multi-size split block is:

$$N_{\text{mult}} = 33 \times \left(H \times W + \frac{1}{4} \times H \times W + \frac{1}{16} \times H \times W \right). \tag{A.19}$$

Taking the asymptotic bound, the complexity of the multi-size split block is $O(HW)$.

Edge feature extractor layers

These layers consist of three steps: gradient filtering, edge thinning, and normalisation. The computations for each step are calculated and summed to obtain the final number of operations. These calculations are as follows:

Gradient filter The gradient filter is applied as a convolution operation with a kernel size $n_k = 9$ in the edge detection method. The total number of gradient computations N_{grad} for the three different sizes is:

$$N_{\text{grad}} = 18 \times \left(H \times W + \frac{1}{4} \times H \times W + \frac{1}{16} \times H \times W \right). \tag{A.20}$$

Edge thinning layer The edge thinning layer performs a series of operations as described in Section 3.4.4. For the edge detection method, the computational complexity is calculated using equation (A.16) with $n_m = (2, 2, 2)$ and $c = (1, 1, 1)$. The total edge thinning computations

$N_{\text{edge-thin}}$ are:

$$N_{\text{edge-thin}} = 8 \times \left(H \times W + \frac{1}{4} \times H \times W + \frac{1}{16} \times H \times W \right). \tag{A.21}$$

Normalisation operation Normalisation is utilised to control the range of data for improved learning. The normalisation computational complexity N_{n} can be directly derived from equation (A.17) with $c = (1, 1, 1)$ for the proposed method. The total computations are:

$$N_{\text{n}} = 4 \times \left(H \times W + \frac{1}{4} \times H \times W + \frac{1}{16} \times H \times W \right) + 3. \tag{A.22}$$

Taking these computations into account, the total number of calculations for the edge feature extractor N_{feat} is calculated as:

$$\begin{aligned} N_{\text{feat}} &= N_{\text{grad}} + N_{\text{edge-thin}} + N_{\text{n}}, \\ &= 30 \times \left(H \times W + \frac{1}{4} \times H \times W + \frac{1}{16} \times H \times W \right) + 3. \end{aligned} \tag{A.23}$$

In terms of Big-O notation, the computational complexity of the edge feature extractor is $O(HW)$.

Upscaling and merge layer

This layer upscales the smaller-sized edge features to the original input depth image size and then merges them. The number of computations N_u required to perform this task is calculated as:

$$N_u = 2(H + W) + 4HW. \tag{A.24}$$

Clustering layer

This layer is implemented as a k-means clustering operation, and the total number of computations N_c is:

$$N_c = 1 + 13HW. \tag{A.25}$$

Finally, the total number of operations N required for the 3D edge detection process is

calculated as:

$$\begin{aligned} N &= N_{\text{enc}} + N_{\text{mult}} + N_{\text{feat}} + N_u + N_c, \\ &= 1849 \times H \times W + 2 \times H + 2 \times W + 4. \end{aligned} \tag{A.26}$$

Upon finalising the network parameters for kernel size and filters, the overall computational complexity of the proposed 3D edge detection network is determined to be $O(HW + H + W)$.

Appendix B

Depth camera simulation and pose evaluation matrices for EOPEN

B.1 Simulating depth cameras

Depth cameras, such as those used in RGB-D sensors, operate based on specific intrinsic parameters that define the mapping from 3D space to 2D image coordinates. Understanding these parameters is crucial for accurately converting depth images into point clouds. The intrinsic parameters of a depth camera are encapsulated in a camera matrix, typically denoted as **K**. This matrix is a fundamental component in the projection equation that relates a 3D point in the camera coordinate system to its 2D projection on the image plane.

B.1.1 Camera intrinsic matrix

The camera intrinsic matrix **K** is defined as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

where f_x and f_y are the focal lengths in pixels along the x and y axes, respectively. c_x and c_y

are the coordinates of the principal point, which is usually the center of the image. The matrix assumes that there is no skew between the x and y axes, which is valid for most cameras.

B.1.2 Projection of 3D points to 2D image plane

The relationship between a 3D point $\mathbf{P}_c = [X_c, Y_c, Z_c]^T$ in the camera coordinate system and its corresponding 2D point $\mathbf{p}_i = [u, v]^T$ on the image plane is given by:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \\ 1 \end{bmatrix} \tag{B.2}$$

B.1.3 Conversion from depth image to point cloud

To convert a depth image to a point cloud, the inverse of the above projection process is applied. Given the pixel coordinates (u, v) and the depth value d (which corresponds to Z_c), the 3D coordinates (X_c, Y_c, Z_c) in the camera coordinate system can be computed as:

$$X_c = \frac{(u - c_x) \cdot d}{f_x} \tag{B.3}$$

$$Y_c = \frac{(v - c_y) \cdot d}{f_y} \tag{B.4}$$

$$Z_c = d \tag{B.5}$$

Here, the depth d is directly obtained from the depth image, where each pixel value represents the distance from the camera to the corresponding point in the scene.

The intrinsic parameters of a depth camera play a vital role in the conversion of depth images into point clouds. By accurately modelling the projection and inverse projection processes using the intrinsic matrix $\mathbf{K}$, 3D points can be reconstructed with high fidelity, enabling a realistic simulation of camera for data generation.

B.2 Evaluation methodologies for object pose estimation

The evaluation methodologies employed in EOPEN to assess the performance of object pose estimation are outlined as follows:

B.2.1 Visible Surface Discrepancy (VSD)

The Visible Surface Discrepancy (VSD) metric measures the average distance between the estimated visible surface points and the ground truth. It is computed as:

$$\text{VSD} = \frac{1}{|V|} \sum_{\mathbf{x} \in V} \min \left(\|\mathbf{R}\mathbf{x} + \mathbf{t} - \mathbf{R}_{gt}\mathbf{x} - \mathbf{t}_{gt}\|, \tau \right) \tag{B.6}$$

where $\mathbf{R}, \mathbf{t}$ are the estimated rotation and translation, $\mathbf{R}_{gt}, \mathbf{t}_{gt}$ are the ground truth, V is the set of visible points, and τ is a threshold for outlier influence.

B.2.2 Maximum Symmetry-Aware Surface Distance (MSSD)

The Maximum Symmetry-Aware Surface Distance (MSSD) measures the maximum distance between corresponding points on the object surface, considering symmetries:

$$\text{MSSD} = \max_{\mathbf{x} \in \mathcal{M}} \min_{\mathbf{S} \in \mathcal{S}} \|\mathbf{R}\mathbf{x} + \mathbf{t} - \mathbf{R}_{gt}\mathbf{S}\mathbf{x} - \mathbf{t}_{gt}\| \tag{B.7}$$

where $\mathcal{M}$ is the set of model points, $\mathcal{S}$ represents symmetry transformations, and $\mathbf{S}$ is a symmetry transformation applied to the points.

B.2.3 Maximum Symmetry-Aware Projection Distance (MSPD)

The Maximum Symmetry-Aware Projection Distance (MSPD) evaluates the maximum distance between the 2D projections of the object model under the estimated and ground truth poses,

considering symmetries:

$$\text{MSPD} = \max_{\mathbf{x}\in\mathcal{M}} \min_{\mathbf{S}\in\mathcal{S}} \|\pi(\mathbf{R}\mathbf{x}+\mathbf{t}) - \pi(\mathbf{R}_{gt}\mathbf{S}\mathbf{x}+\mathbf{t}_{gt})\| \tag{B.8}$$

where $\pi(\cdot)$ denotes the 2D projection function, and $\mathcal{M}, \mathcal{S}, \mathbf{S}$ retain the same definitions as in MSSD.

B.2.4 Average Recall (AR)

The Average Recall (AR) aggregates performance across different error thresholds, measuring the fraction of correctly estimated poses. It is calculated as:

$$\text{AR} = \frac{1}{|\mathcal{O}|}\sum_{i=1}^{|\mathcal{O}|}\frac{1}{|\mathcal{T}|}\sum_{j=1}^{|\mathcal{T}|}\text{recall}(o_i,\tau_j) \tag{B.9}$$

where $\mathcal{O}$ is the set of objects, $\mathcal{T}$ is the set of error thresholds, and $\text{recall}(o_i,\tau_j)$ represents the recall for object o_i at threshold τ_j.

In accordance with BOP challenge metrics [107], AR for each of VSD, MSSD, and MSPD is calculated as follows:

AR_{VSD} The tolerance for misalignment (τ) is varied from 5% to 50% of the object's diameter, in increments of 5%. The threshold for correctness (θ_{VSD}) is varied from 0.05 to 0.5, with steps of 0.05. AR_{VSD} represents the average recall across all combinations of τ and θ_{VSD}.

AR_{MSSD} This metric is the average recall calculated for different levels of the MSSD threshold (θ_{MSSD}), varying from 5% to 50% of the object's diameter, with a step size of 5%.

AR_{MSPD} The average recall rate is computed for different levels of the MSPD threshold (θ_{MSPD}), expressed as a percentage of the scaled factor $r = \frac{w}{640}$, where w is the image width in pixels. The threshold is varied from 5r to 50r, in increments of 5r.

AR is then calculated as the average of AR_{VSD}, AR_{MSSD}, and AR_{MSPD}.

B.2.5 Average Distance of Model Points (ADD-S and ADD)

The ADD metric calculates the average distance between corresponding points on the model surfaces after applying the estimated and ground truth poses:

$$\text{ADD} = \frac{1}{|\mathcal{M}|} \sum_{\mathbf{x} \in \mathcal{M}} \|\mathbf{R}\mathbf{x} + \mathbf{t} - \mathbf{R}_{gt}\mathbf{x} - \mathbf{t}_{gt}\| \tag{B.10}$$

For symmetric objects, ADD-S considers the closest point distance:

$$\text{ADD-S} = \frac{1}{|\mathcal{M}|} \sum_{\mathbf{x} \in \mathcal{M}} \min_{\mathbf{x}' \in \mathcal{M}} \|\mathbf{R}\mathbf{x} + \mathbf{t} - \mathbf{R}_{gt}\mathbf{x}' - \mathbf{t}_{gt}\| \tag{B.11}$$

B.2.6 Rotation error

The rotation error measures the difference between the estimated rotation and the ground truth rotation, calculated as:

$$e_{\text{rot}} = \frac{1}{2} \arccos\left(\frac{\text{tr}(\mathbf{R}^\top \mathbf{R}_{gt}) - 1}{2}\right) \tag{B.12}$$

B.2.7 Translation error

The translation error measures the Euclidean distance between the estimated and ground truth translation vectors:

$$e_{\text{trans}} = \|\mathbf{t} - \mathbf{t}_{gt}\| \tag{B.13}$$

B.2.8 Mean Average Precision (mAP)

The mean Average Precision (mAP) evaluates the accuracy of object detection and pose estimation, calculated as:

$$\text{mAP} = \frac{1}{|O|} \sum_{i=1}^{|O|} \text{AP}(o_i) \tag{B.14}$$

Rotation mAP ($\mathbf{mAP_{rot}}$) The rotation mAP ($\mathrm{mAP_{rot}}$) measures the accuracy of the estimated rotation by evaluating recall at various rotation error thresholds:

$$\mathrm{mAP_{rot}} = \frac{1}{|O|} \sum_{i=1}^{|O|} \mathrm{AP_{rot}}(o_i) \tag{B.15}$$

Translation mAP ($\mathbf{mAP_{trans}}$) The translation mAP ($\mathrm{mAP_{trans}}$) assesses the accuracy of the estimated translation by evaluating recall at various translation error thresholds:

$$\mathrm{mAP_{trans}} = \frac{1}{|O|} \sum_{i=1}^{|O|} \mathrm{AP_{trans}}(o_i) \tag{B.16}$$

These metrics provide comprehensive methods for evaluating object pose estimation, considering visible surfaces, object symmetries, model point distances, and overall precision in both rotation and translation.