Topological Representations for Visual Object Recognition in Unseen Indoor Environments

Ekta Umesh Samani

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:
Ashis G. Banerjee, Chair
Santosh Devasia
Krithika Manohar
Siddhartha Srinivasa

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Topological Representations for Visual Object Recognition in Unseen Indoor Environments

Ekta Umesh Samani

Chair of the Supervisory Committee:
Ashis G. Banerjee
Department of Industrial & Systems Engineering and Department of Mechanical Engineering

Object recognition is an essential component of visual perception tasks that help robots build a semantic-level understanding of their environment. Although deep learning methods achieve extraordinary recognition performance in previously seen environments, they are insufficient for deployment in complex and continually-changing environments due to their sensitivity to environmental variations. To realize the goal of long-term autonomy in robots, we need perception methods that go beyond statistical correlation [204]. Therefore, this dissertation focuses on developing robust object recognition methods using topological methods and human-like reasoning mechanisms.

We begin by using topologically persistent features, which capture the objects' 2D shape information for recognition in unseen environments. In particular, we present two kinds of representations, namely, sparse persistence image (PI) and amplitude, computed by applying persistent homology to multi-directional height function-based filtrations (nested sequences of cubical complexes) representing the objects' segmentation maps. Using a benchmark dataset, we demonstrate that sparse PI features show better recognition performance in unseen environments than the features from widely-used deep learning-based feature extractors. On a new dataset, the UW Indoor Scenes (UW-IS) dataset, designed to test object recognition performance in unseen environments, the performance of sparse PI features remains rel-

atively unchanged in an unseen test environment, unlike a state-of-the-art domain-adaptive object detection method.

Next, we propose a new descriptor, TOPS, to capture the 3D shape information of point clouds generated from depth images, and an accompanying recognition framework, THOR, inspired by human reasoning. The descriptor employs a novel slicing-based approach to compute topological features from filtrations of simplicial complexes using persistent homology, and facilitates reasoning-based recognition using object unity. Apart from a benchmark dataset, we report performance on a new dataset, the UW Indoor Scenes (UW-IS) Occluded dataset, curated using commodity hardware to reflect real-world scenarios with different environmental conditions and degrees of object occlusion. THOR outperforms state-of-the-art methods on both the datasets and achieves substantially higher recognition accuracy for all the scenarios of the UW-IS Occluded dataset.

Subsequently, we extend the TOPS descriptor to incorporate object color information via color embeddings and obtain the TOPS2 descriptor. The color embeddings are computed by leveraging the similarity and connectivity between colors in a color network generated using the Mapper algorithm. The accompanying THOR2 framework, trained entirely on synthetic RGB-D images of unoccluded objects, witnesses considerable performance improvements over the shape-based THOR framework on both the OCID and UW-IS Occluded datasets. THOR2 also achieves substantially higher accuracy than a state-of-the-art vision transformer adapted for RGB-D object recognition on the OCID and UW-IS Occluded dataset, regardless of the camera orientation and environmental conditions, respectively.

Therefore, the approaches presented in this work, which have also been successfully implemented on a low-cost robot, lay the foundation for achieving robust object recognition in unseen environments using computational topology tools.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

SIMPLEX: the convex hull of a collection of vertices; a κ-simplex has κ+1 vertices.

SIMPLICIAL COMPLEX: a finite union of simplices in $\mathbb{R}^n$ such that every face of a simplex from S is also in S, and the non-empty intersection of any two simplices in S is a face of both the simplices.

ELEMENTARY CUBE: a finite product of elementary intervals with the dimension given by the number of its non-degenerate components.

CUBICAL COMPLEX: a finite set of elementary cubes aligned on the grid $\mathbb{Z}^n$.

FILTRATION: (of a finite simplicial complex S) is a nested sequence of simplicial complexes $S_1, \ldots, S_r$ such that $S_1 \subseteq ... \subseteq S_r = S$.

PERSISTENCE DIAGRAM: (of m-th order) is a countable multiset of points in $\mathbb{R}^2$. Each point $(\mathtt{x}, \mathtt{y})$ represents an m-dimensional hole born at a time $\mathtt{x}$ and filled at a time $\mathtt{y}$. The diagonal of a persistence diagram is a multiset $D = \{(\mathtt{x}, \mathtt{x}) \in \mathbb{R}^2 | \mathtt{x} \in \mathbb{R}\}$, where every point in D has infinite multiplicity.

PERSISTENCE IMAGE: a stable and finite dimensional vector representation obtained from a persistence diagram. It is obtained by computing an equivalent diagram of birth-persistence points, which are regarded as a sum of Dirac delta functions and convolved with a Gaussian kernel over a rectangular grid of evenly sampled points.

TOPOLOGICAL FEATURES: are m-dimensional holes. Examples of topological features include connected components (0-dimension holes), loops (1-dimensional holes), and voids (2-dimensional holes).

COVER: is a collection of open sets $\{U_x\}_{x \in X}$ (X is the index set) such that every data point in a dataset is included in at least one U_x.

NERVE: of a cover $U = \{U_x\}_{x \in X}$ is an abstract simplicial complex whose vertices are U_x, and each $(\mathtt{k}+1)$-way intersection of U_x's represents a $(\mathtt{k})$-simplex.

OBJECT UNITY: a reasoning mechanism that allows a human to recognize the presence of occlusion and associate the visible part of an occluded object with the original unoccluded object.

OBJECT CONSTANCY: the ability to understand that objects remain the same irrespective of viewing conditions

VIEW NORMALIZATION (IN HUMANS): the process of rotating an object from the observer's viewpoint to a standard orientation, whose representation is stored in memory.

MACADAM ELLIPSE (IN HUMANS): region on the chromaticity diagram with all the colors indistinguishable (to a human eye) from the color at the ellipse's center. Therefore, the contour of the ellipse represents the just-noticeable differences in chromaticity.

ACKNOWLEDGMENTS

First and foremost, I would like to express heartfelt gratitude to Prof. Ashis G. Banerjee for providing me with this wonderful learning opportunity. I sincerely appreciate his invaluable guidance, unwavering support, and immense patience throughout this journey. I am extremely grateful to him for consistently inspiring me to surpass my limitations, which has undoubtedly prepared me for the future.

I would also like to thank Prof. Santosh Devasia, Prof. Sawyer Fuller, Prof. Krithika Manohar, and Prof. Siddhartha Srinivasa for agreeing to join my supervisory committee; their feedback and insights have been instrumental in guiding my research efforts.

Further, I would like to thank the Department of Mechanical Engineering at the University of Washington for facilitating this journey in every way possible. I am also thankful to my collaborators from Amazon Inc., The Boeing Company, and Volvo Car Technology USA for working with me on several research projects during this journey and helping me expand my research horizons beyond what has been presented in this dissertation. I also want to thank my lab mates, Ben, Chris, Jundi, Marina, Wei, and Xingjian; their comradery and friendship have been an important part of my journey. In addition, I also want to thank my friends and colleagues from the Boeing Advanced Research Center and the Precision Controls Lab for their help and encouragement.

This journey would not have been possible without my loving parents, Umesh and Rita, and my dearest brother and sister-in-law, Neel and Nirali. I cannot thank them enough for their unconditional love and tireless support, which have been my anchors throughout this journey. I am also incredibly grateful for my nephew, Nivaan, our miracle, who invariably makes me smile.

Last but not least, I want to thank Akshay, Annie, Chinmay, Gaurav, Gitika, Prasanna, Raj, Rashmi, Shashank, Vaishnavi, Zainab, and many others for helping me build a life in Seattle.

DEDICATION

to the one who inspired me to take this journey and those who supported me in it

Chapter 1
INTRODUCTION

Perception is one of the core capabilities required by autonomous mobile robots. Vision plays a crucial role in perceiving a robot's environment and developing a semantic-level understanding of it. Most vision tasks fundamentally rely on the ability to recognize objects, scenes, and categories. Therefore, object recognition and localization, often collectively referred to as detection, form an essential aspect of robot visual perception. Object recognition in humans is incredibly sophisticated. Humans can recognize a multitude of objects in their surroundings with little effort. Humans also recognize objects in images regardless of occlusion or variations in appearance, viewpoint, size, scale, or pose. Achieving such recognition performance is still a challenge for robot vision systems [204]. Therefore, we begin this endeavor toward achieving robust object recognition for robots operating in complex and continually changing real-world environments by reviewing how object recognition works in humans.

1.1 Object Recognition in Humans

Evidence from the field of cognitive neuropsychology states that there are four stages in the process of object recognition in humans [135, 191]. These stages also resemble the theory of vision proposed by Marr [119]. Fig. 1.1 shows the four stages and their component processes. The first stage involves processing the basic components of an object, such as color, depth, and form. Subsequently, these components are grouped to segregate surfaces into figure and ground (known as foreground segmentation in the computer vision parlance). The ability to understand that objects remain the same irrespective of viewing conditions, known as object

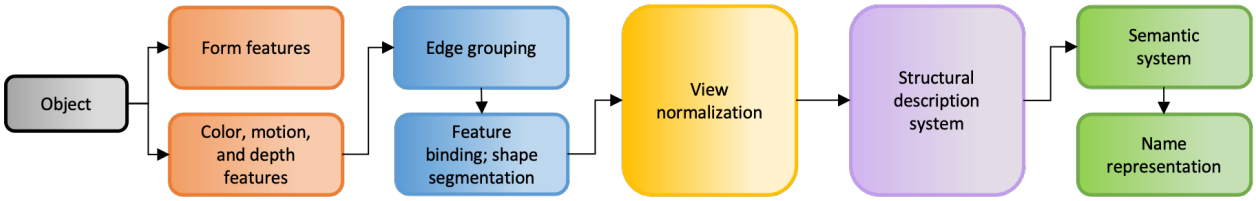


Figure 1.1: A hierarchical model of object recognition in humans. Recreated based on figures in [135, 191].

constancy, is not present in the object representations at this stage. Instead, objects are represented according to the viewpoint of the observer. This viewpoint-based description is then matched with the structural descriptions of objects from memory. However, representations in memory are often assumed to represent only selected viewpoints. Therefore, object constancy is achieved either by computing it as part of the matching process or by 'normalizing' the view by rotating the object to a standard orientation. Last, the visual representation is associated with semantic attributes to provide meaning and recognition. In this dissertation, we direct our efforts toward developing an object recognition framework for robots that aligns with this model of object recognition in humans. First, we briefly discuss how object recognition frameworks have evolved over the past two and half decades.

1.2 Visual Object Recognition in Robots

Numerous approaches have been proposed by the computer vision community toward achieving human-like performance in recognizing rigid objects. These approaches are potential candidates for use in robot vision systems as well. Therefore, we discuss how object recognition methods in computer vision have evolved from classical approaches to the more powerful deep learning-based approaches.

1.2.1 Classical Approaches

Object recognition approaches from the pre-deep learning era can be grouped into methods for identifying instances of a specific object and methods to categorize object instances at a more basic category level [71]. We briefly discuss these approaches in the following subsections.

Specific Object Recognition

Early research explored ordered and orderless global image representations for specific object recognition. Most of these approaches involve comparisons of entire images or image windows and therefore run into issues associated with partial occlusion and substantial changes in viewpoint. Alternative approaches that detect specific objects in a given test image involve using local feature representations. Typically, the recognition procedure using such representations has the following three steps.

1. Local features are extracted from both the training and test images.

2. Feature sets from training and test images are matched to find putative correspondences

3. Verification is performed to check if the matched features occur in a consistent geometric configuration.

The feature extraction step mainly involves using local feature detectors to detect localized points of interest (e.g., Harris and Hessian detectors), scale invariant regions (e.g., Laplacian of Gaussian and Difference of Gaussian detectors), affine covariant regions (e.g., Harris-Affine and Hessian-Affine detectors) or stable segmentation regions (e.g., Maximally Stable Extremal Regions detector). After the regions of interest are detected, their content is captured in a descriptor. The Scale-Invariant Feature Transform (SIFT) [112] and Speeded-Up Robust Features (SURF) [12] descriptors are the most popular choices of such local descriptors. The obtained features sets from training and test images are then matched

using tree-based algorithms or visual vocabularies. Tree-based algorithms perform exact (e.g., *kd*-tree [59]) or approximate (e.g., locality-sensitive hashing [34, 83]) nearest neighbor searches in the high-dimensional space of the descriptors, whereas approaches based on visual vocabularies quantize the feature space into discrete 'visual words' before matching. In the last step, the matches are verified to check if they occur in a consistent geometric configuration by estimating the underlying transformation between them using methods such as RANSAC [117] and Generalized Hough Transform (GHT) [11].

Generic Object Recognition

Unlike specific object recognition, approaches for generic object recognition aim to cope with variations in appearance and shape among instances of the same category. Such approaches largely follow the following three-step procedure:

1. A representation of objects and an accompanying model is chosen.

2. The given test image is searched for object candidates, and confidence scores are assigned to them.

3. Any redundant or conflicting detections are suppressed.

The choices for object representations can be grouped into two major types based on the type of the accompanying model. The first type is the window-based holistic appearance/texture descriptors such as Eigenfaces, bag-of-words (BoW) descriptor, and Histogram of Oriented Gradients (HOG) descriptor. The other type is representations used with part-based models such as the Constellation and Star models. The candidate detection stage employs sliding windows-based approaches for the window-based models and voting and fitting-based methods for the part-based models. Popular examples of generic object recognition approaches include the Viola-Jones face detector [185], bag-of-words discriminative classifiers [43], HOG person detector [44], the Implicit Shape Model detector [101], and the Deformable Part-based Model detector [57, 58].

1.2.2 Deep Learning-based Approaches

Apart from specific and generic object recognition methods, researchers in the computer vision community also worked on developing large benchmark datasets such as the PASCAL VOC [53] and ImageNet [45]. The tremendous increase in available image data and computation power (resulting from GPU usage) led to a series of breakthroughs in image classification and object detection using deep convolutional neural networks (CNNs) starting from AlexNet [93]. Models such as the VGG [159], GoogLeNet [167], ResNet [78], ResNeXt [198] and EfficientNet [169] quickly followed and achieved very high image classification accuracy. Several other models were proposed, a more detailed description of which can be found in [107, 136]. These models served not only as good image classifiers but also as models for extracting deep, high-dimensional feature representations of images. Subsequently, the Vision Transformer (ViT) was also proposed for achieving equally good performance on image classification tasks, while using fewer computational resources to train the model [49].

On the object detection side, Girschick et al. [69] showed that CNNs could lead to dramatically higher object detection performance than the classical approaches by proposing the Region-Convolutional Neural Network (R-CNN). The paper introduced the use of region proposals instead of looking for an object in every location. Fast R-CNN [68] introduced an improvement to R-CNN that reduced the computation time in R-CNN by using a CNN to generate a single feature map for an image and extracting region proposals from it. The Faster R-CNN [140] architecture replaced the selective search algorithm in Fast R-CNN with a Region Proposal Network (RPN) to further reduce computation time. On the other hand, You Only Look Once (YOLO) [137] proposed splitting the entire image into a grid and identifying objects from bounding boxes within each grid. Single Shot Detector (SSD) [109] framework differed from all its predecessors by performing the task of object localization and classification in a single forward pass of the network. The variety of different object detection frameworks described above and their variants combined with feature extraction using image classification models lead to tremendous success in recognizing and detecting

objects even in cluttered or crowded scenes.

1.3 Object Recognition for Long Term Autonomy

Just as object recognition, other aspects of robot technology have also improved considerably over the years. Consequently, autonomous robots have begun to operate in increasingly complex environments. However, when an autonomous robot operates autonomously over extended periods (i.e., weeks, months, or years) in complex and continually changing environments, the challenge of long-term autonomy arises. If the environment in which the robot is deployed is fully known and static, the challenge of long-term autonomy boils down to that of robustness [95]. In the absence of such simplifying assumptions, autonomous robots face multiple other difficulties. For instance, the environment may undergo short-term, medium-term, or long-term changes. Examples of these changes include objects moving in and out of the robot's view, furniture moving from one room to another, and the growth of plants, respectively. Moreover, new objects may appear in the environment, and the environment itself may not be fully known beforehand. Several studies show efforts to address such challenges in the context of robot perception [28, 95, 113].

We limit the scope of our discussion to a scenario where all the objects are known. However, deep learning-based object recognition and detection methods fail to achieve the desired robustness, even in this limited case. Therefore, a common practice is to use a model that is pre-trained on large databases with millions of images, such as ImageNet, and then fine-tune the model based on scene images from the environments where the robots are expected to operate. This practice becomes cumbersome and unsuitable when the models are to be deployed in continually changing environments. Moreover, it renders the models sensitive to variations in environmental conditions such as illumination and object texture [39, 174]. Efforts have been made to address this sensitivity and learn domain (i.e., environment) invariant features through adaptation or generalization. However, domain-invariance achieved through adaptation is limited to the specific source and target domains considered; performance drops are witnessed in robotics applications for unseen target domains [151]. Domain generalization

methods rely on large volumes of data and techniques like domain randomization to learn invariant features. However, the need for abundant real-world training data is often a barrier to implementing these methods on robotics systems with commodity hardware [8]. Therefore, domain-invariant approaches suitable for everyday use on low-cost robots are required. Consequently, this dissertation investigates the use of topological data analysis in designing such approaches.

1.4 Topological Data Analysis

Topological Data Analysis (TDA) is an applied mathematics approach, with a strong theoretical base from algebraic topology and computational geometry, for analyzing high-dimensional data. It provides tools to obtain key information regarding the structure of various types of input data, such as point clouds, images, and meshes. The basic assumption in TDA is that a dataset is sampled from an unknown topological space, which characterizes the "shape" of the data [215]. The central idea of TDA is to study this underlying topological space and infer robust information about the data from it. Consequently, it has tremendous scope in real-world applications where data is often corrupted with noise.

Within TDA, the Mapper algorithm is a popular tool used to recover the underlying space by obtaining a graphical representation of the data. Persistent homology is another widely-used tool to compute the topological and geometric features of the underlying topological space. It provides stability guarantees (with respect to the noise in data) based on theoretical results[1]. In particular, persistent homology has been used to extract topologically persistent features for machine learning tasks [127] and computer vision problems such as texture recognition [138] and classification [75], human posture recognition [75,138], and classification of hand-written digits [62]. Topological methods have also been used in robotics for planning and navigation [15,55], but literature on their application to robot perception [14] is limited.

[1]It is important to note that persistent homology's sensitivity to noise depends on the choice of filtration and persistence representations, especially when working with image data [176]. We discuss filtration and persistence representations in detail in Chapter 3

In this work, we use computational topology tools to obtain suitable representations of objects to achieve robust object recognition in unseen environments.

1.5 Dissertation Outline

In this chapter, we discuss how object recognition works in humans and how visual object recognition for robots has evolved over the past two and a half decades toward achieving human-like object recognition performance. We also note that achieving such performance in unseen environments presents robustness challenges, even in the limited case where all the objects are known. Recognition robustness becomes even more critical in the case of robotic systems equipped with commodity hardware. We briefly introduce topological data analysis (TDA) as one of the ways to address this challenge.

Chapter 2 covers the literature related to the approaches adopted in this dissertation toward achieving robust object recognition in unseen environments. Specifically, we discuss domain adaptation and domain generalization methods in the first section of the chapter. In the second section, we cover explicit handcrafted approaches and implicit learning-based approaches to obtain object representations for recognition. In particular, we cover approaches for obtaining exclusively shape-based object representations and combined shape and color-based representations. As this dissertation primarily focuses on topological methods for obtaining suitable object representations using TDA, we cover some mathematical preliminaries associated with TDA in Chapter 3. In Chapter 4, we initiate the application of TDA for object recognition. We present representations that capture the 2D shape of objects through topologically persistent features for recognition in unseen indoor environments. The resulting findings demonstrate the promise of topological representations and highlight the need for a 3D shape-based object recognition approach, which is presented in Chapter 5. Specifically, we propose a new 3D shape descriptor, TOPS, for point clouds generated from depth images and an accompanying framework, THOR, inspired by human reasoning. The TOPS descriptor employs a novel slicing-based approach to compute topological features and facilities object unity-based recognition in unseen cluttered environments.

Chapter 6 builds upon the work in the previous chapter to present a 3D shape and color-based descriptor, TOPS2, and an accompanying framework, THOR2. In particular, color embeddings computed obtained using the Mapper algorithm for topological soft clustering are interleaved with TOPS to obtain the TOPS2 descriptor. Last, Chapter 7 summarizes the core contributions of this dissertation work, the anticipated impact of this work on the research community, and potential directions for future work.

Chapter 2
RELATED LITERATURE

This dissertation focuses on object recognition in unseen environments. Section 2.1 reviews existing domain adaptation and domain generalization methods that have been proposed for achieving a similar goal. Domain adaptation methods consider a single source and target domain and use (labeled or unlabeled) data from the latter to adapt to it. Domain generalization methods consider data from multiple different source domains for training. Alternatively, this work is geared toward using a single source domain for obtaining representations that work across multiple target domains. To that end, computational topology tools are used to obtain shape and color-based representations from RGB-D images. Therefore, Section 2.2 covers relevant works that explicitly or implicitly compute shape and color-based representations of objects for recognition.

2.1 Domain Adaptation and Domain Generalization Methods

Typical deep learning-based object recognition (and detection) models reviewed in Section 1.2.2 assume that training and test data are drawn from the same distribution. However, a shift in the test data distribution resulting from environmental variations (i.e., covariate shift) leads to performance degradation [174]. Domain adaptation [102] and generalization-based [208] object detection methods have emerged as one of the ways to address this challenge by learning domain-invariant feature representations of objects.

2.1.1 Domain Adaptation Methods

Domain adaptation methods aim to develop a robust object detector using label-rich data from the source domain and label-agnostic data from the target domain. Such methods,

which do not use hand-labeled data from the target domain, are preferred over fine-tuning pre-trained models with hand-labeled data from the target domain because annotated data is not always available. These methods mainly differ in how they address the 'domain shift' between the source and the target domains. They can also be distinguished based on whether its a single step adaptation or a multi-step adaptation, based on whether they are supervised, semi-supervised, weakly-supervised, few-shot, or unsupervised with respect to the target domain, or based on the object detection framework (e.g., Faster R-CNN, YOLO, SSD) used in them. We organize this discussion based on methods employed to address the domain shift between the source and the target domain.

Adversarial Learning-based Adaptation:

Adversarial learning-based domain adaptation is one of the most popular approaches for adaptation. It involves using adversarial training to encourage confusion between the source domain and the target domain through an adversarial objective, using losses such as the domain-confusion loss in [60]. Domain Adaptive Faster R-CNN [38] is the first work to incorporate some of these ideas for object detection. Within the Faster R-CNN framework, they use adversarial learning for image and object instance level adaptation. Alternatively, region-level adaption is performed in [213] within the Faster R-CNN framework. He et al. [79] perform domain feature alignment and proposal feature alignment within the Faster R-CNN detection framework. Saito et al. [148] use adversarial learning for strong local and weak global alignment of features in an unsupervised setting, whereas Wang et al. [189] proposed a few-shot adaptive Faster R-CNN framework. Some other noteworthy examples of adversarial learning-based adaptation methods appear in [36, 156, 214].

Reconstruction-based Adaptation:

Reconstruction-based domain adaptation methods assume that reconstructing source or target domain samples would be beneficial in learning suitable features and improving object detection performance. Often, generative adversarial networks (GANs) are used for recon-

struction. For instance, Arruda et al. [9] use CycleGAN to perform unsupervised image-to-image translation to translate source domain images to the target domain. A cross-domain car detector is then trained using the translated images and the original images annotations. Methods proposed in [47, 103, 108] also employ image-to-image translation for vehicle detection, object detection in thermal images, and object detection on context-enhanced images, respectively. Guo et al. [74] also work with thermal infrared images and use image transformers to convert between color images and thermal images for pedestrian detection.

Other Methods for Adaptation:

Several methods employ a combination of the previously discussed strategies for adaptation. Notable examples in this category include the work of Hsu et al. [81], who use adversarial learning but perform progressive adaptation through an intermediate domain, which is synthesized from the source domain images to mimic the target domain. Inoue et al. [84] propose a two-step progressive adaptation technique with the SSD framework that uses image-level instance labels for the target domain in a weakly-supervised setting where the detector is fine-tuned on samples generated using CycleGAN and pseudo-labeling. Similarly, Kim et al. [91] use CycleGAN to perform domain diversification, followed by multidomain-invariant representation learning using the SSD framework. Unlike other methods, Yu et al. [205] explicitly reduce the domain content distribution gap through their semi-supervised learning framework.

Some domain adaptation methods do not fall into any previously described categories or their combinations. For instance, Cai et al. [30] use a mean teacher paradigm and object relations in the Faster R-CNN framework to address the domain shift. Alternatively, Xu et al. [201] use graph-induced prototype alignment, and Xu et al. [200] use categorical regularization to align the source and target domains. On the other hand, Khodabandeh et al. [89] use a simple but effective framework where they treat target domain predictions of a detector trained on the source domain as noisy labels and train a final detection model using them.

2.1.2 Domain Generalization

While some domain adpatation methods aim to learn domain-invariant features through the adaptation, such invariance is limited to the specific source and target domains considered during adaptation; performance drops are witnessed in robotics applications for unseen target domains [151]. Unlike domain adaptation methods, domain generalization methods aim to learn a fixed set of parameters that perform well in previously unseen environments. Such methods learn from multiple source domains to ensure that domain-specific information is suppressed. While training on enormous datasets such as the ImageNet [45] can achieve a similar goal, gathering such large annotated datasets for every perception task is impossible. Therefore, methods that are specifically designed to learn generalized features become essential. Blanchared et al. [17] and Muandet et al. [123] pioneered domain generalization for the classification problem. Similar to [123], fixed shallow features were also explored in [54] for image classification and in [64, 90, 203] for object recognition. However, very little work has been done on domain generalization for object detection [154, 208]. Single-shot YOLO detectors that use invariant risk minimization have been proposed in [104, 105]. Alternatively, Seemakurthy et al. [154] adopt a domain-generalized, entropy-based regularization approach for detection using a Faster R-CNN framework. We refer the reader to [211] for an in-depth discussion on some of these approaches.

2.2 Object Representations for Recognition

We note from Section 1.2 that computation of object representations for recognition has been of interest since the pre-deep learning era. This section reviews recent works that focus on obtaining shape-based representations from depth images (or point clouds) and representations combining shape and color information from RGB-D images for object recognition.

2.2.1 Shape-based Representations for Recognition

Several studies have been performed to obtain 2D representations of shapes, ranging from simple descriptor-based instance retrieval methods to machine learning approaches. A discussion of those studies is beyond the scope of this review, but a comprehensive summary of the studies can be found in [96]. While such representations, when used for object recognition, provide invariance to some environmental variations (e.g., lighting conditions), they run into issues when the 2D shape of the object itself changes due to changes in viewpoint or occlusion of objects [151]. Recognition using features that capture the 3D shape of the objects can alleviate the difficulty associated with viewpoint variations. In the context of object recognition in robots, one of the approaches to access the 3D shape of the objects is using depth images (or point clouds generated from them). Therefore, we review methods for obtaining 3D shape representations from point clouds in the following subsections.

Hand-crafted and Topological Representations:

In the pre-deep learning era, several feature representations that capture the local geometric structure of point clouds were proposed. These representations can be broadly divided into extrinsic and intrinsic descriptors. Extrinsic descriptors are usually derived from the shape's coordinates in 3D space. These include shape context, spin images, integral features, distance-based descriptors, point feature histograms, and normal histograms. Intrinsic descriptors consider the 3D shape as a manifold discretized as a mesh or graph. Examples of this class of methods include the global point signatures, heat and wave kernel signatures, and their variants. An overview of such hand-crafted methods can be found in [16,76]. In the context of robot perception, Lai et al. [98] used the spin images for performing object category and instance recognition. Rusu et al. [147] propose a global fast point feature histogram for scene interpretation in mobile manipulation scenarios. Shape descriptors for object recognition such as the viewpoint feature histogram [145] and its variants [6,7], global radius-base surface descriptor [120], and ensemble shape functions descriptor [194] also followed. In

contrast to these global descriptors, Beksi et al. [13] proposed a signature of topologically persistent points, which encodes the topological information of a 3D point cloud.

Learning-based Approaches:

Following the extraordinary results that deep learning models achieved on images, efforts also have been made to adapt such methods to geometric data like point clouds. As such data does not have a grid-like structure present in images, view-based [163, 192], volume-based [92, 121, 170, 196] and combination of view-based and volume-based representations [132] have been proposed to place geometric data on a grid. These representations thereby allow using convolution and pooling blocks from deep learning models, originally designed for images, on geometric data.

However, methods such as the ones using volumetric representations usually require excessive memory and introduce quantization artifacts, making it difficult to capture fine-grained features. Therefore, Qi et al. [131] proposed PointNet, pioneering the work on networks that are specially designed to manipulate raw point cloud data instead of an intermediate representation. It independently operates on each point in the point cloud and applies a symmetric function to accumulate features, thereby achieving permutation invariance of points. PointNet++ [133] and KCNet [155] extend this idea, and exploit local features by considering neighborhoods of points instead of acting on individual points. These techniques maintain permutation invariance by treating points at the local scale independently. However, such an approach results in a neglect of the geometric relationships among points. Certain graph-based models have also been proposed for learning representations [158, 190]. In [190], Wang et al. propose a graph-based model called the Dynamic Graph CNN (DGCNN). They propose an operation called the EdgeConv and fuse it with PointNet to obtain this model. EdgeConv captures local geometric structures while maintaining permutation invariance. It generates edge features that capture the relationship between points and their neighbors in a permutation invariant manner, constructs a local graph, and learns the embeddings for the edges. Some purely convolution-based models [110, 202] and transformer-based models [73, 122, 209]

have also been proposed. Goyal et al. [70] compare some of these methods under consistent augmentation and evaluation techniques and note that their approach, named SimpleView, performs at par or better than the models considered. As the name suggests, SimpleView involves a simple projection of points to depth maps along orthogonal views followed by a lightweight CNN to fuse the features. A comprehensive study of these methods' performance in the case of noisy and corrupted point clouds can be found in [139].

2.2.2 Combined Shape and Color-based Representations for Recognition

Object recognition using only depth data is a challenging task as it does not utilize other information, such as the color of objects. Recognition becomes even more challenging in cluttered environments where objects are occluded. Therefore, in the following section, we look at existing approaches for obtaining shape and color-based object representations from RGB-D images.

Hand-crafted Representations:

Most early works in RGB-D object recognition adopt the following framework. First, features capturing the visual appearance of objects are computed from the RGB image. Next, features capturing the object shape are computed from the corresponding depth image. Often, more than one type of color-based features and and shape-based features are computed and fused into a global descriptor through concatenation [129] or aggregate representations [25, 98]. These global descriptors are then used for recognition using a classifier. For instance, Lai et al. [98] compute multiple features such as SIFT, color histograms, and texton histograms to capture the visual appearance of objects. Spin images are then computed from the depth image to capture the shape of objects. Efficient match kernel (EMK) [20] features are then computed from the spin images, as well as the color features. Next, feature selection is performed to obtain low-dimensional shape and visual descriptors, respectively, using principal component analysis (PCA). These desciptors are combined for subsequent recognition using different classifiers such as linear support vector machine (LinSVM), gaussian kernel support

vector machine (kSVM) and random forest. Similar kernel-based descriptors for RGB-D object recognition have also been proposed in [19–21, 27]. Covariance matrices-based descriptors that capture visual and shape information have also been explored [56, 130, 178]. Further, approaches that used traditional feature learning techniques such as hierarchical matching pursuit (HMP) [22], PCA [165], and single-layer convolutional operators [18, 40] have also been proposed.

Learning-based Approaches:

The remarkable object recognition performance of deep learning-based methods in the case of RGB images has also inspired CNN [61] and transformer-based approaches [180] for RGB-D object recognition.

Among multimodal CNNs for RGB-D object recognition, a common practice is using off-the-shelf CNNs (one per modality) pre-trained on the ImageNet dataset. Typically, the RGB images are directly fed to the corresponding pre-trained network. Depth images, on the other hand, are first rendered as three-channel RGB images to emulate the distribution of the corresponding RGB images before feeding them to the corresponding pre-trained network. Such a rendering is often performed using hand-crafted or learned depth encodings. The most common depth encoding methods include surface normal-based encodings [2, 22], ColorJet [51, 153], HHA (stands for the horizontal disparity, the height above ground, and the angle the pixel's local surface normal makes with the inferred gravity direction) [77], and embedded depth encoding [206]. A learning-based encoding, inspired by methods of colorizing grayscale photographs, is also proposed in [33]. Some works also employ two or more depth encodings by using multiple depth streams [134] or ensemble learning [1].

In addition to the choice of depth encodings, such works also focus on appropriately fusing information from the RGB and depth modalities. The two most popular fusion schemes in literature are early and late fusion [152]. In the case of early fusion, raw data from both modalities (or their corresponding features) are fused to obtain a joint representation for further joint processing. The central idea of such approaches is to exploit correlations between

the two modalities through joint processing. Some works perform this fusion through direct concatenation of fully-connected layers [2,33,51,210]. In other cases, matrix transformations [187], canonical correlation analysis (CCA) [188], and reconstruction independent component analysis [85] have been used for fusion using fully-connected layers. In addition, fusion through a single convolutional layer [106] or multiple recurrent neural networks [29, 162] has also been explored. On the other hand, late fusion refers to the scenario where most of the feature learning of the two modalities happens independently, and fusion is performed before the last decision stage [10, 206].

More recently, approaches that rely on the flexibility of transformers networks to capture cross-modal interactions between multimodal inputs have also been proposed for scene and action recognition [66, 67], semantic segmentation [67, 207], and object recognition [180]. Specifically, Tziafas et al. [180] adapt the Vision Transformer (ViT) [49] to perform RGB-D object recognition. They investigate various strategies for depth encoding and fusing the encoded depth with RGB input, and show that late fusion of surface normal-based encodings with RGB input works better than the more popularly employed early fusion techniques in the case of RGB-D object recognition.

Chapter 3
MATHEMATICAL PRELIMINARIES

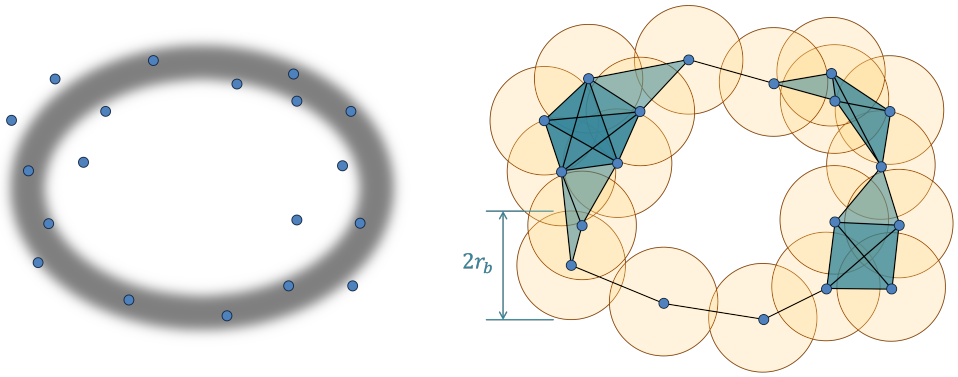
This chapter covers some of the mathematical preliminaries associated with topological data analysis (TDA). As mentioned in Section 1.4, the core idea of TDA is to recover the topological space underlying the data and study it to obtain robust information about the structure of the data. The following subsections describe two popular approaches for recovering the underlying space.

3.1 Persistent Homology-based Approach

Persistent homology [50, 215] is a predominant tool in TDA for computing the topological features of data. In this approach, the first step is to construct a nested sequence of complexes, called a filtration. Second, persistent homology is applied to track the evolution of topological features in the filtration. Last, the resultant topological information is summarized in suitable persistence representations. The following subsections describe these steps in detail.

3.1.1 Constructing Complexes from Cata

Different types of complexes can be constructed to capture the connectivity of data points in any data. The choice of complex type is often based on the type of data. Specifically, in the case of point cloud data, the connectivity of the points is usually represented by a simplicial complex. A simplicial complex S is a set composed of 0-simplices (points), 1-simplices (line segments), 2-simplices (triangles), and their n-dimensional counterparts. It is a finite union of simplices in $\mathbb{R}^n$ such that every face of a simplex from S is also in S, and the non-empty intersection of any two simplices in S is a face of both the simplices. In the case of image



(a) (b)

Figure 3.1: (a) Sample point cloud with a gray elliptical annulus highlighting its topological feature, i.e., a loop. (b) A simplicial complex constructed from the point cloud using the union of balls approach. Source: modified from [195].

data, an image can be considered as a point cloud (by treating every pixel as a point in $\mathbb{R}^2$) and represented using a simplicial complex. However, since an image is made up of pixels, it has a natural grid-like structure to it. Therefore, a cubical complex is a more efficient representation [186]. A cubical complex C in $\mathbb{R}^n$ is a finite set of elementary cubes aligned on the grid $\mathbb{Z}^n$, where an elementary cube is a finite product of elementary intervals with dimension given by the number of its non-degenerate components [86].

In the case of point cloud data, a simplicial complex representing the connectivity of its points can be built by considering the union of balls with radius, r_b, centered on the points in the point cloud, and connecting pairs of points that are no further apart than $2r_b$. When r_b is sufficiently small, the simplicial complex consists of n_p 0-simplices, where n_p is the number of points in the point cloud. The simplicial complex would be a single $(n_p - 1)$-simplex when r_b is considerably large. Fig 3.1(a) shows a sample point cloud; the gray elliptical annulus explicitly shows the topological feature (a loop) that describes the point cloud. Fig. 3.1(b) shows the corresponding simplicial complex constructed using an arbitrary value of r_b.

In the case of image data, we consider an n-dimensional image to be a map $\mathcal{I} : I \subseteq \mathbb{Z}^n \longrightarrow \mathbb{R}$. The subset $I \subseteq \mathbb{Z}^n$ is the image, an element $v \in I$ is the voxel, and its value $\mathcal{I}(v)$ is the intensity. When $n = 2$, the voxel is known as a pixel, and the intensity is known as the grayscale value. There are various ways of constructing a cubical complex from an n-dimensional image [62, 142]. We follow the method used by Garin et al. [62], where a k-cube represents a voxel, and all the adjacent lower-dimensional cubes (faces of the k-cube) are included. The values of the voxels v are extended to all the cubes λ in the resulting cubical complex C as

$$\mathcal{I}'(\lambda) := \min_{\lambda \text{ is face of } v} \mathcal{I}(v). \tag{3.1}$$

3.1.2 Computing Topological Features

In Section 3.1.1, we note that different simplicial complexes can be constructed from a point cloud based on the value of r_b. Instead of identifying an optimal r_b, we examine simplicial complexes obtained using a range of r_b [50,65]. The corresponding nested sequence of simplicial complexes is called the filtration. Formally, a filtration of a finite simplicial (or cubical) complex S is a nested sequence of simplicial complexes $S_1, \ldots, S_r$ such that $S_1 \subseteq ... \subseteq S_r = S$. Commonly, such a filtration is generated by considering the sublevel sets $L_t = f_d^{-1}([-\infty, t])$ of a descriptor function $f_d : \mathbb{X} \longrightarrow \mathbb{R}$ on a topological space $\mathbb{X}$ indexed by a parameter $t \in \mathbb{R}$. Fig. 3.2 shows selected simplicial complexes in the filtration of the point cloud in Fig. 3.1(a).

Persistent homology studies the topological changes of these sublevel sets L_t as t increases from $(-\infty, \infty)$. During filtration, topological features appear and disappear at different scales, referred to as their birth and death times, respectively. The lifetime of each feature is termed the persistence of the feature. From a geometric perspective, these topological features are interpreted as m-dimensional holes, e.g., connected components as 0-dimension holes, loops as 1-dimensional holes, and voids as 2-dimensional holes. For instance, the point cloud in Fig. 3.1(a) can be described as having one connected component (0-dimensional

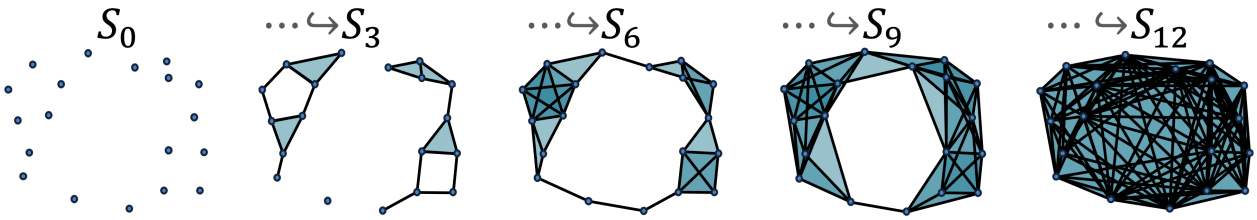


Figure 3.2: A sample filtration [195] of simplicial complexes, corresponding to the point cloud in Fig. 3.1(a), generated using the Euclidean distance between two points as the descriptor function.

hole) and one loop (1-dimensional hole). Multiple persistence representations have been proposed to summarize this topological information [4, 26, 50, 65]. In the next subsection, we cover two such representations, namely, the persistence diagram [50] and the persistence image [4].

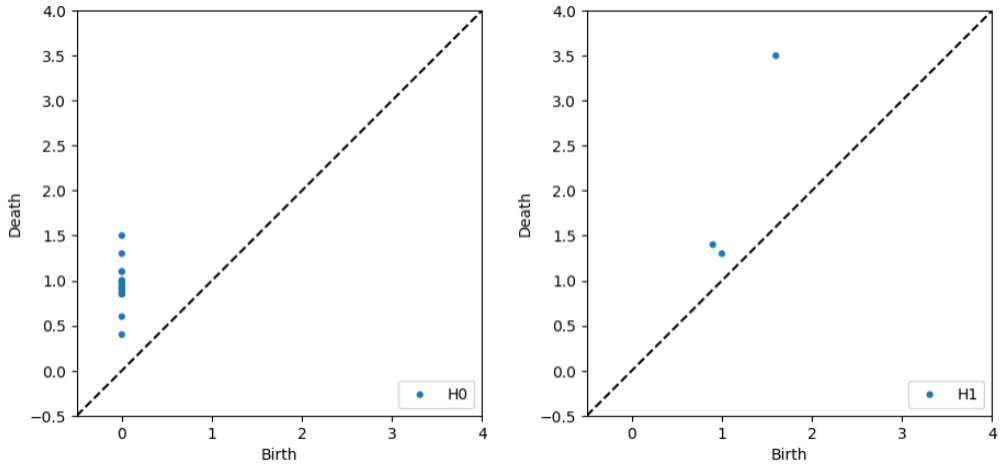
3.1.3 Persistence Representations

Persistence Diagrams:

The topological information captured by persistent homology can be summarized in an m-dimensional persistence diagram (PD). An m-dimensional PD is a countable multiset of points in $\mathbb{R}^2$. Each point (a, b) represents an m-dimensional hole born at a time a and filled at a time b. The diagonal of a PD is a multiset $\mathcal{D} = \{(a, a) \in \mathbb{R}^2 | a \in \mathbb{R}\}$, where every point in $\mathcal{D}$ has infinite multiplicity. Fig. 3.3 shows the PDs for the point cloud in Fig. 3.1(a).

Persistence Images:

Several stable representations of persistence can be obtained from a PD. One such representation is the Persistence Image (PI), a stable and finite dimensional vector representation of persistent homology [4]. To obtain a PI, an equivalent diagram of birth-persistence points, i.e., $(a, b-a)$, is computed. The birth-persistence points are then regarded as a sum of Dirac



(a) 0-dimensional persistence diagram (b) 1-dimensional persistence diagram

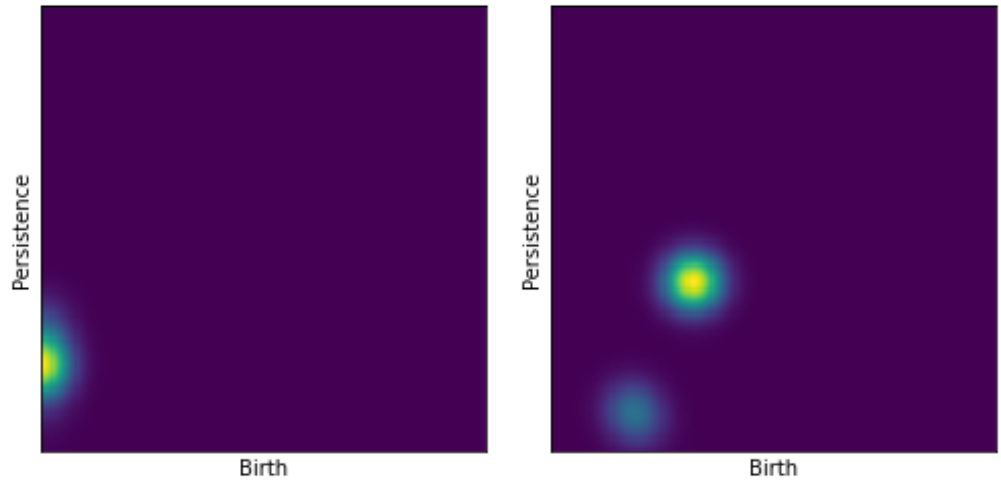
Figure 3.3: Persistence diagrams corresponding to the point cloud in Fig. 3.1(a) obtained by applying persistent homology to the filtration in Fig. 3.2.

delta functions, which are convolved with a Gaussian kernel over a rectangular grid of evenly sampled points to compute the PI. Fig. 3.4 shows the PIs for the point cloud in Fig. 3.1(a).

3.2 Mapper Algorithm-based Approach

The Mapper algorithm [161] is an exploratory data analysis tool used to summarize and visualize the structure of any given data. It is based on the idea of using the nerve of a data's cover to build a graph (or simplicial complex) representing the data. Specific structures (e.g., loops) can be identified in the graph for subsequent identification of interesting clusters or feature selection.

The Mapper algorithm assumes that a high dimensional dataset, say D, resides on a low dimensional manifold. Therefore, the first step in the algorithm is to apply a carefully chosen function, $f_l : D \longrightarrow \mathbb{R}$, to reduce the dataset to a low-dimensional space. The choice of the function f_l, called the lens function, depends on the features of the data we intend



(a) 0-dimensional persistence image (b) 1-dimensional persistence image

Figure 3.4: Persistence images corresponding to the point cloud in Fig. 3.1(a) obtained from the persistence diagrams shown in Fig. 3.3.

to highlight in the output graph. In the next step, a cover is built in the low-dimensional space. A cover, U, of a dataset is defined as a collection of open sets $\{U_x\}_{x\in X}$ (X is the index set) such that every data point is included in at least one U_x. Let $\mathcal{U}$ denote the overlapping cover of $f_l(D)$, where $\mathcal{U} = \{\mathcal{U}_y\}_{y\in Y}$. The Mapper algorithm then pulls the cover back to D via f_l^{-1} to obtain a collection of subsets $\left\{f_l^{-1}(\mathcal{U}_y)\right\}_{y\in Y}$ called the *pullback* cover of D. Next, a clustering algorithm is applied to each $f_l^{-1}(\mathcal{U}_y)$, where $y \in Y$. The collection of the subsequently obtained clusters[1] is called a *refinement* of $\left\{f_l^{-1}(\mathcal{U}_y)\right\}_{y\in Y}$. Let $\mathcal{R}$ denote the refined pullback cover of D. The output, G, of the Mapper algorithm is the nerve of $\mathcal{R}$ [35]. This nerve is constructed by collapsing each cluster $R \in \mathcal{R}$ into a vertex and creating a p-simplex to represent each $(p+1)$-way intersection of R's. Fig. 3.5 illustrates the Mapper algorithm applied to the sample point cloud in Fig. 3.1(a).

[1]In case the Mapper algorithm is defined in a topological space, the clustering step splits each $f_l^{-1}(\mathcal{U}_y)$, where $y \in Y$ into connected components, instead of clusters.

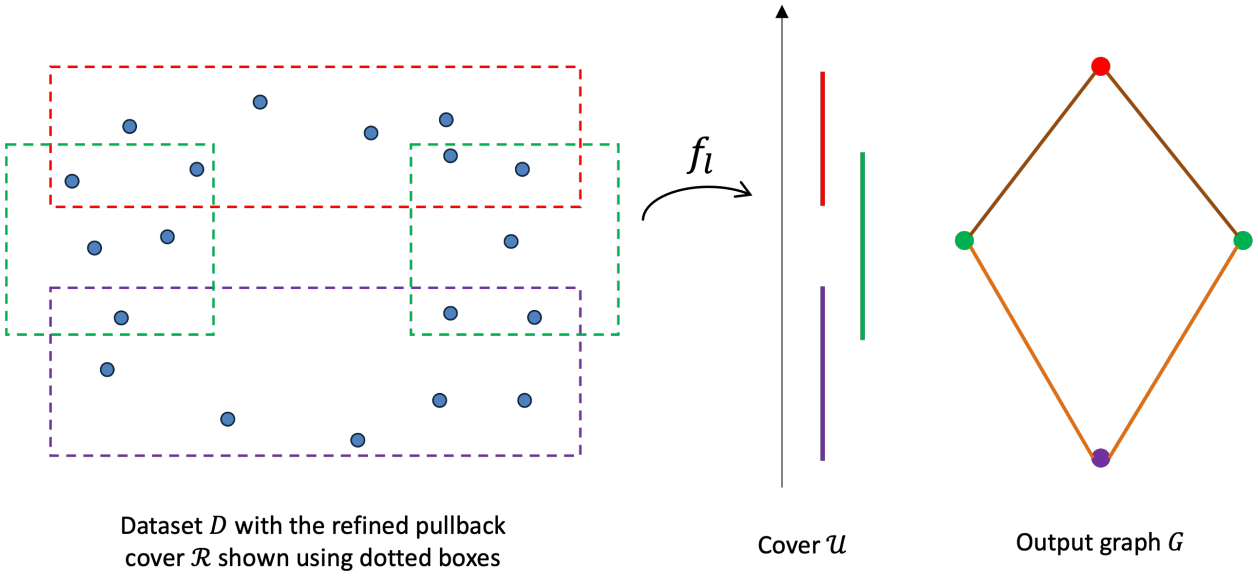


Figure 3.5: An illustration of the Mapper algorithm applied to the sample two-dimensional point cloud in Fig. 3.1(a). In this example, the height function is used as the lens function, f_l, to build the cover in a one-dimensional space. The dotted boxes indicate clusters in the corresponding refined pullback cover of the point cloud. The clusters are collapsed into vertices and intersections between them are represented by edges in the output graph. Source: modified from [35].

Chapter 4

2D SHAPE-BASED RECOGNITION IN UNSEEN ENVIRONMENTS

The work presented here appeared in the IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 7509-7516, Oct. 2021. In this work, we propose the use of topologically persistent features for object recognition in indoor environments. We acknowledge help from Xingjian Yang (Department of Mechanical Engineering, University of Washington) in performing the robot implementation experiments.

4.1 Motivation

Autonomous mobile robots are a major part of the increasingly common smart warehouses. Such robots are expected to operate autonomously in a continually changing environment for extended periods. Therefore, they share the same difficulties (concerning object recognition) associated with environmental variations as long-term autonomous robots, as discussed in Section 1.3. Domain adaptation and domain generalization methods, discussed in Section2.1.1, are one class of approaches aimed at addressing some of these challenges. However, domain adaptation methods require data from the target domain, and domain generalization methods require abundant real-world training data, which poses deployment barriers on robots with commodity hardware.

Alternatively, we propose using topologically persistent features, which capture the objects' shape information, for recognition. Specifically, we propose two kinds of topological representations of 2D shape: sparse persistence image (PI) and amplitude. We also present a new dataset, the UW Indoor Scenes dataset, to evaluate the robustness of object recognition methods in unseen environments. The dataset comprises scene images from two different

environments: a living room and a mock warehouse. We show that recognition using topological representations is more robust to changing environments than a state-of-the-art domain adaptation-based object detection model. We also demonstrate that sparse PI features have better recognition performance than features from deep learning-based recognition methods and lead to better performance than end-to-end object detection methods (in terms of accuracy and recall) in unseen environments. We also successfully implement the proposed framework on a real-world robot.

The rest of the chapter is organized as follows. Section 4.2 provides details of the proposed approach, and Section 4.3 describes the datasets used. Experimental details, results, and failure modes are summarized in Section 4.4, followed by concluding remarks in Section 4.5.

4.2 Methods

Given an RGB scene image, our goal is to recognize all the objects in the scene based on their shape information that is captured using topologically persistent features. We first generate segmentation maps for the objects using a deep neural network, as explained in Section 4.2.1. We then compute topological representations (i.e., sparse PI and amplitude) from the object segmentation maps, as described in 4.2.2. These representations are then fed to a fully connected network for recognition. Fig. 4.1 illustrates the proposed framework.

4.2.1 Object Segmentation Map Generation

To generate the object segmentation maps for an input scene image, we follow a foreground segmentation method that is similar to the one proposed in [199]. In particular, we use the state-of-the-art DeepLabv3+ architecture [37], which is pre-trained on a large number of classes and is hypothesized to have a strong representation of 'objectness'. Subsequently, we train the network using pixel-level foreground annotations for a limited number of images from our datasets.

A shape-based object recognition method relies on the objects' contours for distinguishing among multiple objects. However, the number of foreground pixels is very low as compared

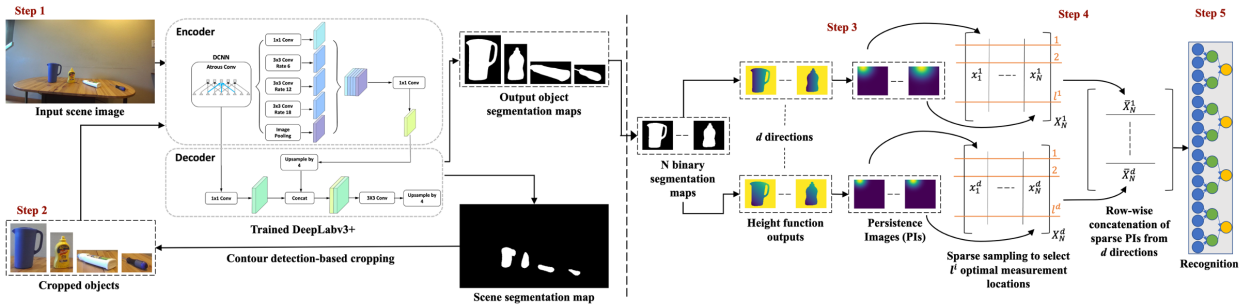


Figure 4.1: Proposed framework for 2D shape-based object recognition using topologically persistent features captured in the sparse PI representation.

to the background when segmentation is performed on images taken from distances as large as two meters. Hence, it is difficult for a segmentation model to capture the minor details in the objects' shapes in a single shot. Therefore, we employ a two-step segmentation framework to preserve the objects' contours details. In the first step, the segmentation model predicts a segmentation map for the input scene image. Contour detection is performed on this scene segmentation map to obtain the bounding boxes for the objects. These bounding boxes are used to divide the scene image into multiple sub-images, each of which contains a single object. In step two, these sub-images are fed to the same trained segmentation model for predicting the individual object segmentation maps.

4.2.2 Computation of Topological Representations

Segmentation maps are essentially binary images comprising only black and white voxels. A grayscale image, suitable for building a filtration of cubical complexes, can be generated from such binary images using various filtration functions. We select a commonly used filtration function, known as the height function, which computes a sufficient statistic to uniquely represent shapes in $\mathbb{R}^2$ and surfaces $\mathbb{R}^3$ in the form of the persistent homology transform [177].

For the cubical complexes in our case, we define the height function as stated in [62]. Consider a binary object segmentation map $\mathcal{B} : I \subseteq \mathbb{Z}^n \longrightarrow \{0, 1\}$, a grayscale image $\mathcal{H} : I \longrightarrow \mathbb{R}$ for the segmentation map I, and a direction $\vec{p} \in \mathbb{R}^n$ of unit norm. The values of all the voxels of $\mathcal{H}$ are then reassigned as

$$\mathcal{H}(v) := \begin{cases} \langle v, \vec{p} \rangle & \text{if } \mathcal{B}(v) = 1 \\ H_\infty & \text{if } \mathcal{B}(v) = 0. \end{cases} \tag{4.1}$$

Here, $\langle v, \vec{p} \rangle$ is the distance of voxel v from the hyperplane defined by $\vec{p}$ and H_∞ is the filtration value of the voxel that is farthest away from the hyperplane.

In step three, we obtain d such grayscale images for each object segmentation map by considering d directions that are evenly spaced on a unit 1-sphere $\mathbb{S}^1$. We construct cubical complexes from each grayscale image according to Eq. (3.1). The sublevel sets of the d complexes are then computed to obtain d filtrations. Persistent homology is applied to these filtrations to obtain d persistence diagrams (PDs) for every object segmentation map. We only consider 0^{th} order homology for generating the PDs. We investigate the performance of two types of topological representations from the generated PDs. The following subsections 4.2.2 and 4.2.2 describe their computation details.

Sparse persistence image representation:

Since the number of points in a PD varies from shape to shape, such a representation of persistence is not suitable for machine learning tasks. Instead, we use persistence images (PIs) to generate a suitable representation for training the recognition network. However, only a few key pixel locations of the PIs, which contain nonzero entries, sparsely encode topological information. Therefore, we adopt QR-pivoting based sparse sampling to obtain a Sparse PI, as proposed in [75]. For every object segmentation mask, d PIs are generated from their corresponding PDs. Fig. 4.2 shows sample PIs for two different objects using height functions in multiple directions, illustrating the (collective) presence of sufficient discriminative information. In step four, for every direction $\vec{p}_k, k \in \{1, \ldots, d\}$, the corresponding PIs for all

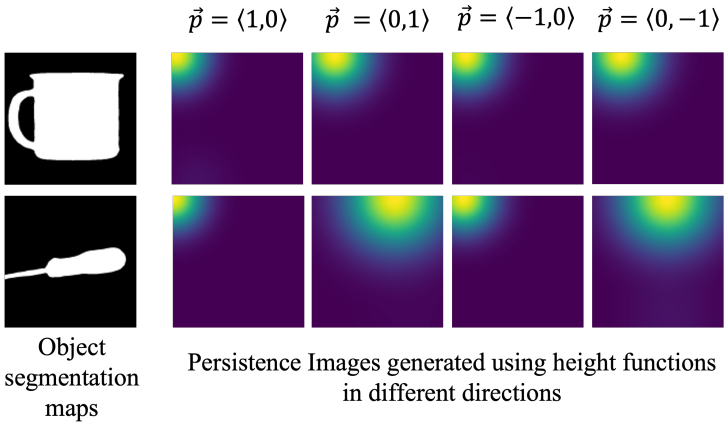


Figure 4.2: Persistence images (PIs) for two sample objects. The collection of PIs, computed using height functions in different directions on object segmentation maps, have enough information to distinguish between the two objects.

the N training (binary) object segmentation maps are vectorized and arranged as columns of a large matrix X_N^k.

The dominant PI patterns U_{l^k} are obtained by computing the truncated singular value decomposition of X_N^k as

$$X_N^k \approx U_{l^k}^k \Sigma_{l^k}^k (V_{l^k}^k)^T, \tag{4.2}$$

where l^k is the optimal singular value threshold [63] for the $\hat{p}_k$-th direction PIs. U_{l^k} is then discretely sampled using the pivoted QR factorization as

$$U_{l^k}^T (\Pi^k)^T = Q^k R^k. \tag{4.3}$$

The numerically well-conditioned row permutation matrix Π^k is then multiplied with X_N^k to give a matrix $\bar{X}_N^k$ of sparsely sampled PIs. We finally perform row wise concatenation of all the d sparsely sampled PI matrices to generate the final representation for recognition in step five.

Amplitude representation:

An alternative method of representing topologically persistent features is using the amplitude or the distance of a given PD from an empty diagram. For each of the d generated PDs corresponding to an object segmentation map, we compute the bottleneck amplitude [62], A_b^k, as

$$A_b^k = \frac{\sqrt{2}}{2} \sup_i (y_{ii}^k - x_{ii}^k), \tag{4.4}$$

where (x_{ii}^k, y_{ii}^k) are all the non-diagonal points in the $\hat{p}_k$-th direction PD. These amplitudes are stacked to form a d-dimensional topological representation. Such d-dimensional representations are generated for all the N object segmentation maps to train the recognition network.

4.3 Datasets

4.3.1 MPEG-7 Shape Silhouette Dataset

We first choose the widely used MPEG-7 Shape Silhouette dataset to solely characterize the shape recognition capability of the two persistent features-based networks for (almost) ideal object segmentation maps. In particular, we use a subset of this dataset, namely, the MPEG-7 CE Shape 1 Part B dataset, which is specifically designed to evaluate the performance of 2D shape descriptors for similarity-based image retrieval [99]. It includes the shapes of 70 different classes and 20 images for each class, for a total of 1,400 images. Fig. 4.3 shows sample images from the dataset.

4.3.2 RGB-D Scenes Dataset v1

The shapes in the MPEG-7 are detailed and fairly distinguishable from each other. However, common objects in indoor environments are often less detailed and, therefore, more challenging for topological methods. Most deep learning models work exceptionally well in recognizing such everyday objects in their training environments. However, they face challenges when used in new (previously unseen) environments without any retraining, even if

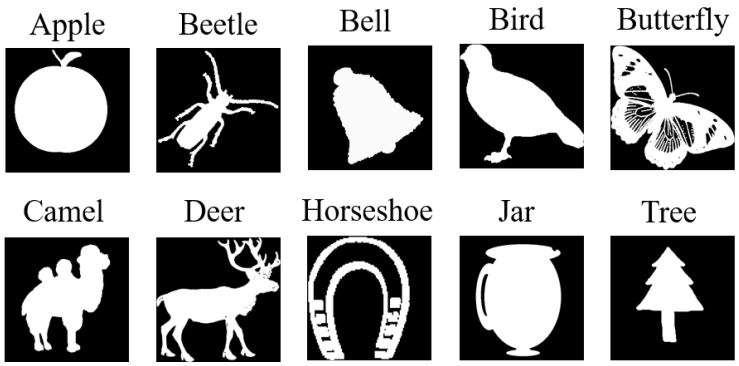


Figure 4.3: Sample images from the MPEG-7 Shape Silhoutte Dataset.

the objects remain the same. Therefore, we choose the widely used benchmark, the RGB-D Scene Dataset v1 [98], to evaluate the performance of our proposed methods. The dataset consists of eight scene setups with objects that belong to the RGB-D Object Dataset [98]. The scenes are shot in five different environments: *desk*, *kitchen_small*, *meeting_small*, *table*, and *table_small*. In particular, we select the *table_small* and *desk* environments for our evaluation. Among all the objects present in the scenes, we consider six object classes for recognition corresponding to the six objects types that appear in both the environments. For our current analysis, we only consider RGB images where the objects are not occluded.

4.3.3 UW Indoor Scenes Dataset

While the RGB-D Scenes Dataset consists of scenes from multiple environments, all of them are tabletop environments with limited variation in terms of lighting conditions and object types. On the other hand, the dataset in [141] provides images with several objects but only in a single environment. Although there are quite a few other indoor scene datasets in the literature, none of them includes a large enough set of object types, poses, and arrangements with varying backgrounds and lighting conditions. Therefore, we introduce a new RGB dataset, which we call the UW Indoor Scenes (UW-IS) dataset, for evaluating object recognition performance in multiple, typical indoor environments. For this purpose,

we select a fixed set of fourteen different objects from the benchmark Yale-CMU-Berkeley (YCB) object and model set [32].

The UW-IS dataset consists of indoor scenes taken in two completely different environments. The first environment is a living room scene where the objects are placed on a tabletop. The second environment is a mock warehouse setup where the objects are placed on a shelf. For the living room environment, we have a total of 347 scene images. The images are taken in four different illumination settings from three different camera perspectives and varying distances up to two meters. Sixteen out of the 347 scene images are with two different objects, 135 images are with three different objects, 156 images are with four different objects, and 40 images are with five different objects. For the mock warehouse environment, we have a total of 200 scene images taken from distances up to 1.6 meters. Sixty out of 200 images are images with three different objects, 68 images with four different objects, and 72 images are with five different objects. Fig. 4.4a shows some sample living room scene images, and Fig. 4.4b shows sample images from the mock warehouse environment. Fig. 4.4c shows all the fourteen objects used in our dataset. The dataset is publicly available at `https://data.mendeley.com/datasets/dxzf29ttyh/`.

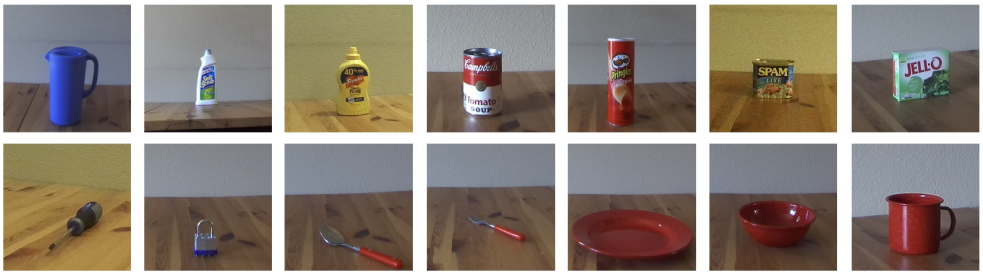
4.4 Experiments

4.4.1 Implementation Details

We perform five-fold training and testing on all images of the MPEG-7 Shape Dataset. For evaluation on the RGB-D Scenes dataset, we use the *table_small* environment for both training and testing, and the *desk* environment only for testing. For the UW-IS dataset, we use the living room images for training and testing, and the mock warehouse images only for testing. All the training and testing is performed using GeForce GTX 1080 and 1080 Ti GPUs on workstations running Windows 10 and Ubuntu 18.04 LTS, respectively. The code for the proposed methods is available at `https://github.com/smartslab/objectRecognitionTopologicalFeatures`.

(a) Living room environment

(b) Mock warehouse environment



(c) Objects

Figure 4.4: Representative images from the UW Indoor Scenes Dataset.

For the MPEG-7 Shape Dataset, we divide the 1,400 images into five sets of 280 images each, with four images of each class included in every set. We perform five-fold training and testing using these sets, such that each set is used once as a test set while the remaining four sets are used for training and validation. We use the giotto-tda [171] library to generate the PDs and the Persim package in Scikit-TDA Toolbox to generate the PIs. The PDs are generated using height functions along 8 evenly spaced directions on S^1. We choose a grid size equal to 50×50, a spread of 10, and a linear weighting function for generating the PIs. We use three-layered and five-layered fully connected networks for recognition using amplitude features and sparse PI features, respectively. We use the ReLU activation (last layer uses softmax activation), Adam optimizer and categorical cross-entropy loss function for training. The learning rate is set to 0.01 for the first 500 epochs. It is decreased by a factor of 10 after every 100 epochs for the next 200 epochs, and by a factor of 100 for the last 100 epochs.

For the RGB-D Scenes dataset and the UW-IS dataset, we use the DeepLabv3+ segmentation model with Xception-65 backbone following the implementation in [172] to obtain object segmentation maps. The network is initialized using a model pre-trained on the ImageNet and PASCAL VOC2012 datasets available from [172]. We use 147 images from the *table_small* environment to train the network for the RGB-D Scenes dataset and 200 living room images to train the network for the UW-IS dataset. Horizontally flipped counterparts of the images are also used for training the models. We generate the corresponding foreground annotations using LabelMe [193]. We train the network for 20,000 steps using the categorical cross-entropy loss with 10% hard example mining after 2,500 steps in the case of the RGB-D Scenes dataset. For the UW-IS dataset, we use 1% hard example mining[1].

We then perform five-fold training and testing of the proposed methods on both datasets by diving the scene images from the training environment into five folds. The resulting five

[1]Unlike the UW-IS dataset, the smaller training set for the RGB-D Scenes dataset leads to poor quality object segmentation maps that are unsuitable for extracting topological features. Therefore, we use a separate DeepLabv3+ model for Step 2, which is fine-tuned on images of individual objects obtained from the original scene images

models are also tested separately on all the images of the test environment. First, the trained DeepLabv3+ segmentation models are used to obtain object segmentation maps from all the images. To ensure the quality of segmentation maps remain consistent across training and test environments, we generate object segmentation maps for the test environments using segmentation models fine-tuned in these environments. We then appropriately pad all the segmentation maps with zeros and consistently resize them to obtain 125 $\times$ 125 binary images. We augment the training data by rotating every object segmentation map by 90^o, 180^o, and 270^o. The PDs and PIs are generated in the same manner as for the MPEG-7 dataset, except that the spread value is chosen to be 20. We also use the same fully connected network architectures and hyperparameters as for the MPEG-7 dataset.

Comparison with deep learning-based object recognition methods:

To solely characterize the recognition performance of the topologically persistent features-based representations, we compare their performance with features extracted using two other widely used object recognition methods, namely, ResNetV2-56 [78] and EfficientNet-B4 [169], on the RGB-D Scenes benchmark. We use the same sets of cropped objects obtained using DeepLabv3+ in Step 1 of our proposed method for five-fold training and testing of both the methods. The networks for both the methods are trained using the implementations available in Keras [88].

Comparison with end-to-end object detection methods:

We also compare the overall performance of both the persistent features-based methods against end-to-end object detection methods on both the RGB-D Scenes dataset and the UW-IS dataset. Particularly, we compare performance against Faster R-CNN [140], a widely used object detection method, and its state-of-the-art variant for cross-domain object detection, Domain Adaptive Faster R-CNN [38]. For Faster R-CNN (referred to as FR-CNN), we use a pre-trained model with InceptionResNet-V2 feature extractor and hyperparameters available with the TensorFlow Object Detection API [82]. For Domain Adaptive Faster R-

CNN, we use a modified implementation from [94], where the VGG backbone is replaced with ResNet-50, and the RoI-pooling layer is replaced with the more popular RoIAlign. The modified implementation has been shown to outperform other state-of-the-art methods for cross-domain object detection [94]. We call this improved method DA-FR-CNN*. Similar to the proposed methods, we perform five-fold training and testing of these methods on both datasets. In the case of DA-FR-CNN* for the UW-IS dataset, all the training set images with artificial lighting (e.g., bottom row in Fig. 4.4a) are used as the source domain, while all the images with natural lighting (e.g., top row in Fig. 4.4a) are used as the target domain. Since such a split is not possible in the RGB-D Scenes dataset, we randomly divide the images into the source and target domains. The ground truth bounding boxes for both the datasets are generated using LabelImg [181].

4.4.2 Results

We first examine the recognition performance of both amplitude features and sparse PI features on the MPEG-7 dataset. We use the weighted F1 score, weighted precision, weighted recall, and accuracy for evaluating performance. Table 4.1 shows the test-time performance of the trained recognition networks. We observe that sparse PI-based recognition performance is quite impressive (more than 0.85) with respect to all the four metrics. It is also consistently better than amplitude-based recognition, indicating the usefulness of sparse sampling in selecting the key features. It is worth noting that 100% recognition performance using 2D shape knowledge is not possible for this dataset, since some of the classes contain shapes that are significantly different from the others in the same class but are similar to certain shapes in other classes [99].

We then examine the performance[2] of both the persistent features-based methods along with other object recognition and detection methods on the RGB-D Scenes benchmark. Ta-

[2]We do not account for the false negatives and false positives resulting from errors of the segmentation model to ensure a fair judgement of our methods' effectiveness. Equivalently, for object detection methods, false positives arising from incorrect region proposals are also not considered.

Table 4.1: Performance comparison of amplitude and sparse PI features (best in bold) on the MPEG Shape Silhouette dataset. (w) indicates weighted metric.

	Amplitude	Sparse PI
F1 score (w)	0.75±0.01	**0.87±0.01**
Precision (w)	0.77±0.01	**0.89±0.02**
Recall (w)	0.76±0.01	**0.87±0.01**
Accuracy	0.76±0.01	**0.87±0.01**

ble 4.2 reports the performance of all the methods (trained in the *table_small* environment) on the unseen *desk* environment. We observe that sparse PI-based recognition, which achieves an overall accuracy of 0.71±0.01, has the highest performance among all the methods in terms of recall and accuracy. Particularly, for the same set of object images obtained using the DeepLabv3+ model, recognition using sparse PI features computed from object segmentation maps is better than recognition using features learned by ResNetV2-56 and EfficientNet-B4 from RGB inputs with respect to accuracy, recall, and F1-score. Additionally, amplitude features-based recognition is comparable to both ResNetV2-56 and EfficientNet-B4. Additionally, the difference between the recognition performance using sparse PI features and amplitude features is, however, much lower than that for the MPEG-7 dataset. We also note that the sparse PI features-based method outperforms Faster R-CNN (referred to as FR-CNN) substantially and DA-FR-CNN* by a small margin in terms of accuracy and recall.

We then assess the performance of both the persistent features-based methods, FR-CNN, and DA-FR-CNN* on the living room scene images from the UW-IS dataset reported in Table 4.3. We observe that both the persistent features-based methods, which only use the information in object segmentation maps, perform reasonably well. Sparse PI-based recognition achieves an overall accuracy of 0.71 ± 0.01 and is marginally better than amplitude-based recognition, whose accuracy is 0.69 ± 0.01. Moreover, both FR-CNN and DA-FR-CNN*,

Table 4.2: Performance comparison of the proposed persistent features-based methods with ResNetV2-56, EfficientNet-B4, FR-CNN, and DA-FR-CNN* (best in bold) on the *desk* images from the RGB-D Scenes dataset.

Metric	Class	Amplitude	Sparse PI	ResNetV2-56	EfficientNet-B4	FR-CNN	DA-FR-CNN*
F1 score	Bowl	0.71±0.01	0.69±0.01	0.95±0.01	0.90±0.03	0.83±0.02	0.88±0.01
	Cap	0.36±0.02	0.63±0.03	0.46±0.06	0.58±0.04	0.46±0.03	0.99±0.01
	Cereal box	0.80±0.01	0.88±0.01	0.76±0.03	0.83±0.03	0.65±0.01	0.79±0.01
	Cup	0.24±0.02	0.50±0.03	0.11±0.08	0.06±0.04	0.87±0.02	0.26±0.12
	Soda can	0.76±0.01	0.66±0.01	0.30±0.05	0.29±0.04	0.62±0.01	0.51±0.03
	Stapler	0.70±0.01	0.71±0.01	0.88±0.03	0.84±0.05	0.69±0.03	0.78±0.01
F1 score (w)	-	0.70±0.00	0.71±0.01	0.65±0.01	0.65±0.01	0.70±0.01	**0.73±0.01**
Precision (w)	-	0.75±0.01	0.74±0.00	0.79±0.01	0.77±0.01	0.82±0.01	**0.87±0.01**
Recall (w)	-	0.68±0.00	**0.71±0.01**	0.68±0.01	0.69±0.01	0.66±0.01	0.69±0.01
Accuracy	-	0.68±0.00	**0.71±0.01**	0.68±0.01	0.69±0.01	0.66±0.01	0.69±0.01

which are trained on this environment and use RGB images as inputs, outperform these methods with respect to all the metrics including class-wise F1 scores.

We believe that the quality of the object segmentation maps has substantial impact on the performance of the persistent features-based methods. Notably, performance is considerably worse for the fork as compared to the other objects. To assess this impact, we compare the recognition performance of both these methods with that of a human given an identical set of segmentation maps. Table 4.4 summarizes the comparison results. For the persistent features based-methods, we only report the accuracy for those images where the human recognizes the object correctly. We observe that a human achieves an accuracy of 0.84 ± 0.01, which is lower that FR-CNN performance, and finds it difficult to recognize the objects based on the generated segmentation maps, especially for the spoon and fork classes. We refer the reader to Section 4.4.4 for further discussion on segmentation map quality.

Despite this challenge, we expect recognition performance to be relatively unaffected,

Table 4.3: Performance comparison of the proposed persistent features-based methods with FR-CNN and DA-FR-CNN* (best in bold) on the living room images from the UW-IS dataset.

Metric	Class	Amplitude	Sparse PI	FR-CNN	DA-FR-CNN*
F1 score	Spoon	0.62±0.04	0.57±0.01	0.84±0.03	0.68±0.02
	Fork	0.15±0.05	0.16±0.07	0.81±0.02	0.32±0.06
	Plate	0.83±0.02	0.89±0.01	0.98±0.01	0.95±0.02
	Bowl	0.95±0.01	0.94±0.02	0.98±0.01	0.97±0.01
	Cup	0.67±0.02	0.78±0.04	0.91±0.01	1.00±0.00
	Pitcher base	0.81±0.03	0.87±0.02	0.92±0.02	0.98±0.01
	Bleach cleanser	0.73±0.02	0.72±0.02	0.87±0.03	0.97±0.01
	Mustard bottle	0.63±0.02	0.68±0.03	0.84±0.03	0.98±0.01
	Soup can	0.68±0.04	0.69±0.02	0.84±0.04	0.86±0.02
	Chips can	0.72±0.02	0.75±0.02	0.92±0.03	0.90±0.03
	Meat can	0.55±0.05	0.56±0.04	0.82±0.03	0.88±0.01
	Gelatin box	0.55±0.02	0.64±0.04	0.91±0.02	0.91±0.01
	Screwdriver	0.58±0.04	0.72±0.04	0.91±0.02	0.94±0.02
	Padlock	0.74±0.03	0.72±0.03	0.97±0.02	0.93±0.03
F1 score (w)	-	0.68±0.01	0.71±0.01	**0.89±0.01**	0.88±0.01
Precision (w)	-	0.69±0.01	0.71±0.02	**0.91±0.01**	0.90±0.01
Recall (w)	-	0.69±0.01	0.71±0.01	**0.88±0.01**	**0.88±0.00**
Accuracy	-	0.69±0.01	0.71±0.01	**0.88±0.01**	**0.88±0.00**

Table 4.4: Comparison of recognition accuracy of the proposed persistent features-based methods with a human given the object segmentation maps for living room scene images from the UW-IS dataset.

Class	Human performance	Amplitude	Sparse PI
Spoon	0.37±0.07	0.44±0.05	0.47±0.09
Fork	0.40±0.11	0.16±0.08	0.12±0.07
Plate	0.96±0.02	0.90±0.03	0.90±0.04
Bowl	0.97±0.01	0.96±0.01	0.97±0.01
Cup	0.92±0.03	0.65±0.05	0.81±0.03
Pitcher base	0.97±0.01	0.87±0.04	0.92±0.02
Bleach cleanser	0.91±0.01	0.86±0.02	0.77±0.02
Mustard bottle	0.91±0.03	0.63±0.04	0.73±0.04
Soup can	0.76±0.05	0.83±0.06	0.81±0.06
Chips can	0.89±0.04	0.81±0.04	0.74±0.07
Meat can	0.86±0.03	0.55±0.07	0.61±0.03
Gelatin box	0.88±0.04	0.59±0.03	0.56±0.07
Screwdriver	0.83±0.03	0.63±0.06	0.74±0.06
Padlock	0.85±0.04	0.81±0.05	0.82±0.03
Accuracy	0.84±0.01	0.74±0.01	0.76±0.01

when the environments vary considerably but the objects are identical, provided the quality of the segmentation maps remains consistent. Accordingly, we test the performance of the persistent features-based methods on all the warehouse scene images of the UW-IS dataset without retraining the recognition networks. We also test the performance of FR-CNN and DA-FR-CNN* on all the warehouse images without any retraining. Table 4.5 summarizes the performances of all the four methods on the mock warehouse test environment. We observe that the performance of sparse PI-based recognition is unchanged from the living room scenario, and is substantially better than that of amplitude-based recognition, whose accuracy reduces by 4%. However, the performance of FR-CNN degrades a lot without fine-tuning (accuracy drops by 25%). The performance of DA-FR-CNN* also degrades considerably (accuracy drops by 18%). Similar to the RGB-D Scenes benchmark case, the sparse PI features-based method outperforms FR-CNN substantially and DA-FR-CNN* by a small margin with respect to accuracy and recall.

4.4.3 Robot Implementation

We also implement our proposed framework on a LoCoBot platform built on a Yujin Robot Kobuki Base (YMR-K01-W1) and powered by an Intel NUC NUC7i5BNH Mini PC. We mount a ZED2 camera with stereo vision on top of the LoCoBot and control the robot using the PyRobot interface [124]. The camera images are fed to the trained segmentation model and recognition networks, which are run on an on-board NVIDIA Jetson AGX Xavier processor, equipped with a 512-core Volta GPU with Tensor Cores and 8-core ARM v8.2 64-bit CPU. We use TensorRT [125] for optimizing the trained segmentation model. Fig. 4.5 shows a screenshot of the platform. The sparse PI features-based recognition runs at a speed of $1.1s$ per frame on this platform.

4.4.4 Discussion

We note that segmentation map quality has a considerable impact on the performance of the persistent features-based methods. For example, the performance is particularly bad for

Table 4.5: Performance comparisons of the proposed persistent features-based methods with FR-CNN and DA-FR-CNN* (best in bold) on mock warehouse scene images from the UW-IS dataset.

Metric	Class	Amplitude	Sparse PI	FR-CNN	DA-FR-CNN*
F1 score	Spoon	0.44±0.04	0.52±0.02	0.39±0.03	0.32±0.05
	Fork	0.28±0.04	0.20±0.04	0.24±0.03	0.00±0.00
	Plate	0.16±0.07	0.04±0.02	0.27±0.16	0.81±0.01
	Bowl	0.94±0.00	0.91±0.01	0.97±0.01	0.97±0.00
	Cup	0.51±0.02	0.90±0.01	0.94±0.01	0.92±0.00
	Pitcher base	0.86±0.01	0.94±0.00	0.93±0.03	1.00±0.00
	Bleach cleanser	0.79±0.02	0.78±0.01	0.84±0.04	0.96±0.01
	Mustard bottle	0.76±0.02	0.74±0.01	0.36±0.05	1.00±0.00
	Soup can	0.73±0.02	0.66±0.02	0.85±0.02	0.81±0.01
	Chips can	0.76±0.01	0.74±0.01	0.71±0.03	0.87±0.02
	Meat can	0.58±0.09	0.71±0.01	0.75±0.05	0.74±0.04
	Gelatin box	0.23±0.04	0.25±0.02	0.70±0.03	0.70±0.03
	Screwdriver	0.78±0.02	0.72±0.01	0.42±0.04	0.76±0.03
	Padlock	0.61±0.03	0.84±0.01	0.89±0.03	0.60±0.04
F1 score (w)	-	0.65±0.01	0.70±0.00	0.65±0.01	**0.75±0.01**
Precision (w)	-	0.67±0.01	0.70±0.01	0.77±0.02	**0.87±0.01**
Recall (w)	-	0.66±0.01	**0.71±0.01**	0.63±0.01	0.70±0.01
Accuracy	-	0.66±0.01	**0.71±0.01**	0.63±0.01	0.70±0.01

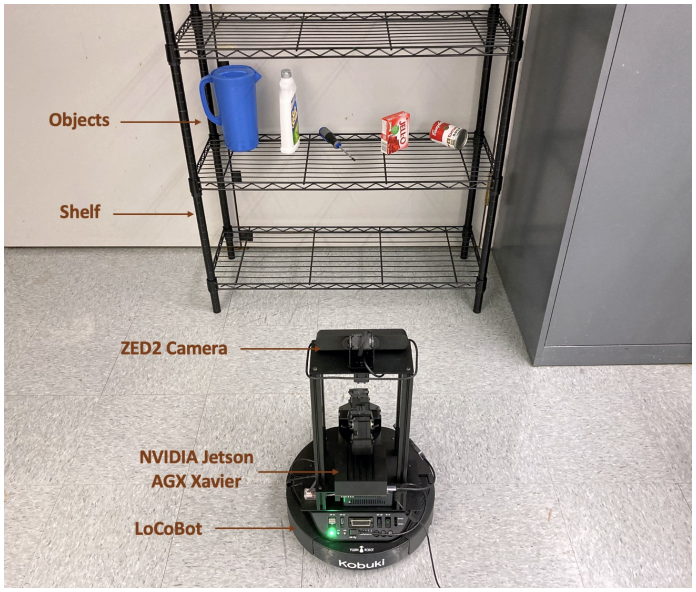


Figure 4.5: Screenshot of the LoCoBot operating in a mock warehouse setup.

the fork and spoon classes, as observed from Tables 4.3 and 4.5. The inherent similarity between forks and spoons, along with low segmentation quality, often makes it hard to distinguish between them, as shown in Fig. 4.6(a). Table 4.4 shows that even humans have difficulty distinguishing between them from the generated segmentation maps. Both Faster R-CNN (referred to as FR-CNN) and DA-FR-CNN* also have difficulties with these classes, especially in the unseen warehouse environment. On a related note, incomplete segmentation maps also affect performance of the proposed methods. For example, the shape in the incomplete segmentation map of a padlock shown in Fig. 4.6(b) is quite different from that of a padlock, making it difficult for a topology-based method to recognize the object. Moreover, we observe from Table 4.4 that our methods only achieve 76% and 74% of human performance using sparse PI and amplitude features, respectively. We believe this difference is primarily due to humans' cognitive capability to complete (partially visible) shapes.

Additionally, we observe relatively small variations in the class-wise performance of the proposed methods between the living room and mock warehouse environments except for

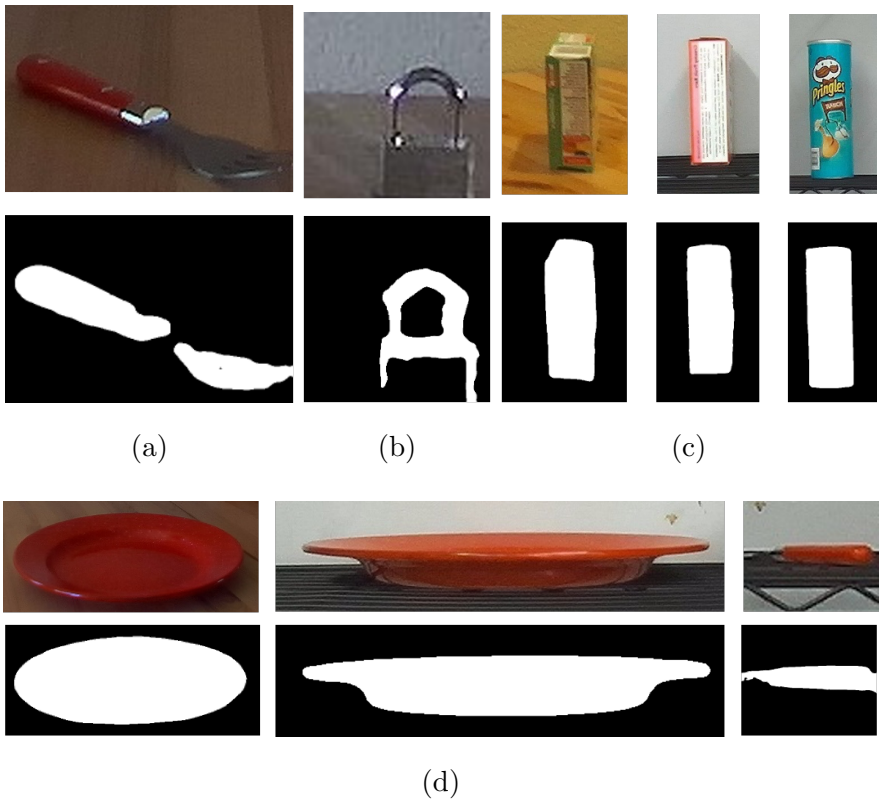


Figure 4.6: Sample failure cases. Fig. 4.6(a) shows a case where a poorly segmented fork is confused with a spoon. Fig. 4.6(b) shows a case where, unlike the proposed topological methods, a human succeeds by being able to complete the shape. Fig. 4.6(c) and 4.6(d) illustrate that camera pose changes cause some of the objects to appear similar to other objects.

the plate and gelatin box classes. This observation can be attributed to changes in camera locations and viewing angles that lead to unseen object poses and naturally occurring variations in object placements. Fig. 4.6(c) illustrates this problem for the gelatin box. The leftmost image is from the living room. The middle image shows the same object pose captured from a different camera viewing angle in the warehouse. The 2D shape in the middle image becomes very similar to that of the chips can in the rightmost image. Similarly, Fig. 4.6(d) shows how such a change results in a completely different 2D shape for the plate, which is similar to a half-visible spoon. The performance of FR-CNN is also affected by such variations in camera pose, as observed for the plate class in Table 4.5. On the other hand, the performance of DA-FR-CNN* drops due to changes in object appearance. For example, the cups in the *desk* and *table_small* environments look considerably different, leading to poor recognition, as reported in Table 4.2.

4.5 Summary

In this chapter, we propose the use of topologically persistent features for object recognition in indoor environments. We construct cubical complexes from binary segmentation maps of the objects. For every cubical complex, we obtain multiple filtrations using height functions in multiple directions. Persistent homology is applied to these filtrations to obtain topologically persistent features that capture the shape information of the objects. We present two different kinds of topological representations, namely, sparse PI and amplitude, to train a fully connected recognition network. Sparse PI features achieve better recognition performance in unseen environments than features learned from ResNetV2-56 and EfficientNet-B4. Unlike end-to-end object detection methods Faster R-CNN and its state-of-the-art variant DA-FR-CNN*, the overall performance of our methods remains relatively unaffected on a different test environment without retraining, provided the quality of the object segmentation maps is maintained. Moreover, the sparse PI features-based method slightly outperforms DA-FR-CNN* in terms of recall and accuracy, making it a promising first step in achieving robust object recognition.

Chapter 5

HUMAN-INSPIRED 3D SHAPE-BASED RECOGNITION IN UNSEEN CLUTTERED ENVIRONMENTS

The work presented here is currently under revision for potential publication in IEEE Transactions on Robotics [150]. We acknowledge the contributions of Xingjian Yang and Srivatsa Grama Satyanarayana (Department of Mechanical Engineering, University of Washington) in collecting the UW-IS Occluded dataset described in Section 5.3.2. A video associated with this work was presented at the IEEE International Conference on Robotics and Automation (ICRA), 2023, and a compact version [149] of this work was presented at the IEEE ICRA 2022 Workshop on Robotic Perception and Mapping: Emerging Techniques.

5.1 Motivation

The previous chapter illustrates the challenges domain adaptation-based methods face in unseen environments and showcases the potential of topological features in achieving robust object recognition. However, the proposed 2D shape-based framework has difficulties with varying camera poses, emphasizing the necessity for 3D shape-based features. In this work, we consider obtaining 3D shape information from point clouds generated using depth images. Several geometric, topological, and learned representations, discussed in Section 2.2.1, have been proposed to capture the 3D shape of object point clouds. However, incomplete and noisy point clouds, due to partial occlusion and imprecise depth imagery, respectively, present an additional challenge for such methods [139]. Consequently, robust approaches that account for occlusion-related changes in 3D shape are required for shape-based object recognition in unseen cluttered environments. This chapter presents a human-inspired approach to object recognition using topological representations to address this need.

Humans recognize the presence of occlusion using a reasoning mechanism known as object unity. Object unity enables association between the visible part of an occluded object with the original unoccluded object in memory. Specifically, this association is made between representations of the occluded and unoccluded objects. However, object representations in memory often represent only selected viewpoints. Therefore, humans 'normalize' the occluded object's view, i.e., rotate the object to a standard orientation. This ability to understand that objects remain the same irrespective of viewing conditions is called object constancy. We incorporate these ideas of object unity and constancy in our persistent homology-based approach to facilitate object recognition in unseen cluttered environments.

Specifically, we develop a novel descriptor function to obtain the Topological features Of Point cloud Slices (TOPS) descriptor using persistent homology. We also propose an accompanying object unity-based framework called TOPS for Human-inspired Object Recognition (THOR). In addition, the new UW Indoor Scenes (UW-IS) Occluded dataset recorded using commodity hardware for systematically evaluating object recognition methods in different environments with varying lighting conditions and degrees of clutter. We show that THOR, trained using synthetic data, outperforms state-of-the-art methods on a benchmark RGB-D dataset for object recognition in clutter and achieves markedly better recognition accuracy in all the environmental conditions of the UW-IS Occluded dataset. Notably, our descriptor function captures the detailed shape of objects while ensuring similarities in the descriptors of the occluded objects and the corresponding unoccluded objects. THOR uses this similarity, eliminating the need for extensive training data that comprehensively represents all possible occlusion scenarios.

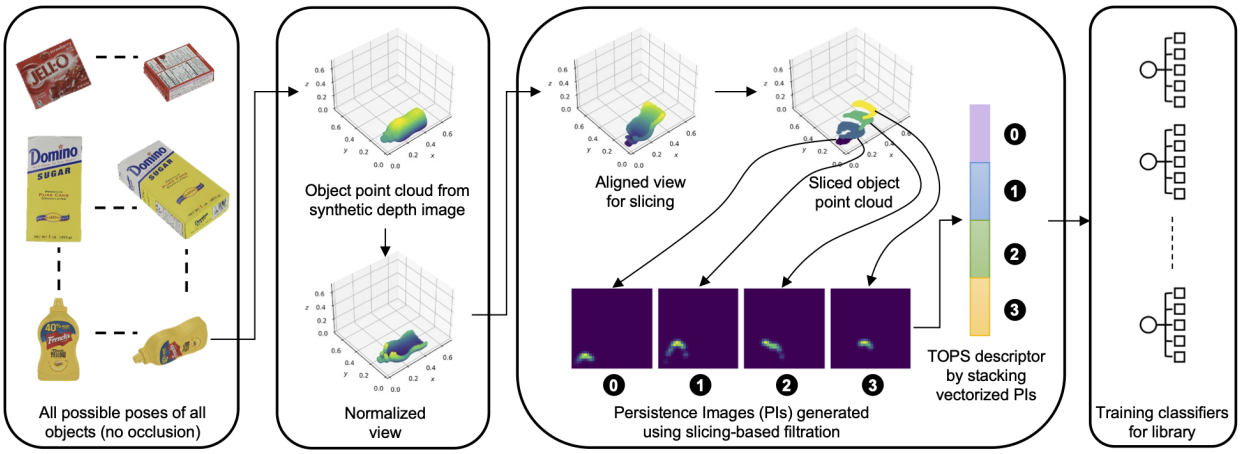
The rest of the chapter is organized as follows. Section 5.2 provides details of the THOR framework, and Section 5.3 describes the datasets used. Experimental details and results are reported in Section 5.4. Several aspects of the THOR's performance and failure modes are discussed in Section 5.5 followed by concluding remarks in Section 5.6.

5.2 Method

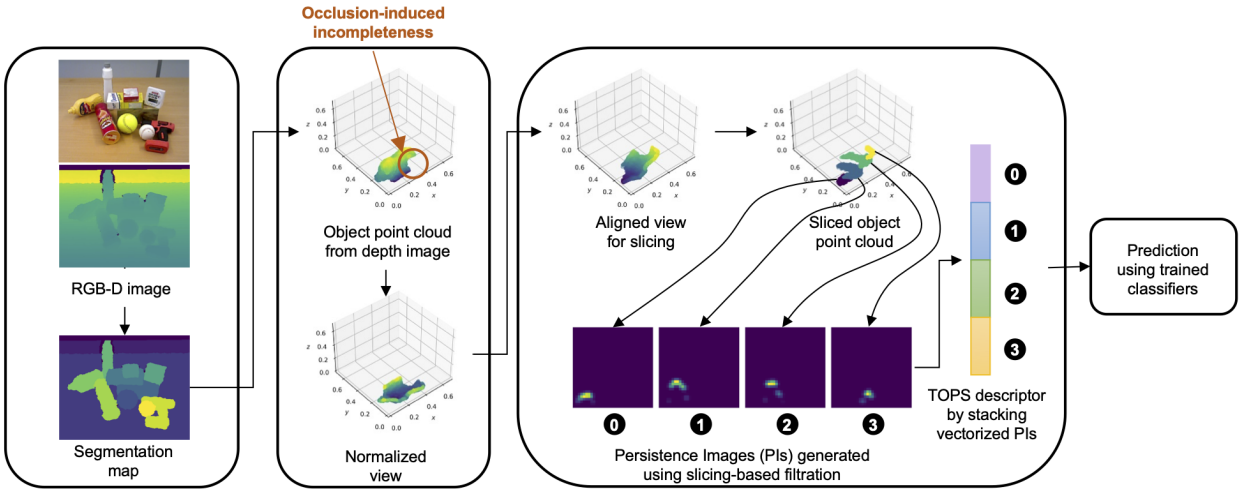
Given an RGB-D image of a cluttered scene, our goal is to recognize all the objects in the scene. As described in Section 1.1, visual object recognition in humans is a four-stage process. THOR follows similar stages, as shown in Fig. 5.1b. Since we focus on object recognition in this work, we assume that instance segmentation maps are available using methods such as those in [114, 197]. We use them to generate the individual point clouds of all the objects in the scene from the depth image. Next, we perform view normalization on every point cloud and compute its TOPS descriptor to capture the 3D shape information. We then perform recognition using models from a library of trained classifiers. To generate the library, we consider synthetic depth images corresponding to all the possible views of all the objects. Similar view normalization is performed on the point clouds generated from the depth images, and their descriptors are computed. The descriptors are then used to train classification models for the library. The following subsections describe these steps in further detail.

5.2.1 View Normalization.

Consider an object point cloud $\mathcal{P}$ in $\mathbb{R}^3$. To perform view normalization, we first compute the minimal volume bounding box of $\mathcal{P}$ using a principal components analysis (PCA)-based approximation of the O'Rourke's algorithm [126]. The bounding box is oriented such that the coordinate axes are ordered with respect to the principal components. We then rotate the point cloud such that the minimal-volume bounding box of the rotated point cloud is aligned with the coordinate axes. To fine-tune this alignment, we perform further rotations such that the 2D-bounding boxes of the point cloud's projection on the $x-y$ and $y-z$ planes are aligned with their respective coordinate axes. We then perform translation such that the resultant point cloud, $\tilde{\mathcal{P}}$, lies in the first octant.



(a) THOR training stage showing a visualization of the TOPS descriptor computed from the synthetic depth image corresponding to a sample object pose (of an unoccluded mustard bottle) chosen from the set of object poses considered during library training.



(b) THOR testing stage, aligned with the stages of human object recognition shown in Fig. 1.1, showing a visualization of the TOPS descriptor of an occluded object in a scene computed from the corresponding real depth image.

Figure 5.1: Proposed framework, THOR, for 3D shape-based recognition using object unity, facilitated by the similarity in the TOPS descriptors of unoccluded and occluded objects.

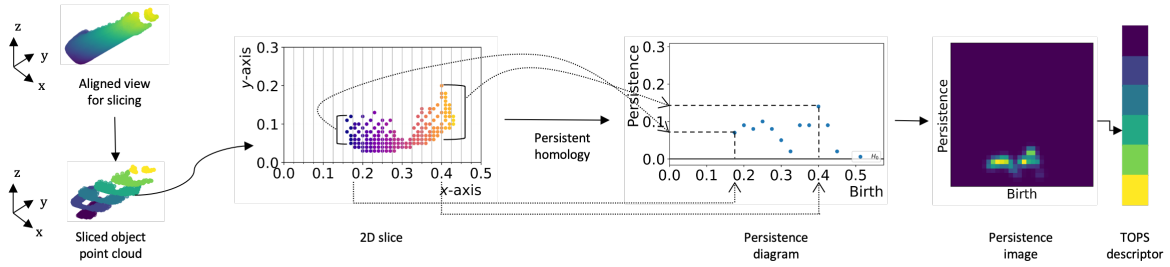


Figure 5.2: Visualization of the TOPS descriptor computation. Example of an aligned object point cloud, $\hat{\mathcal{P}}$, the slices $\mathcal{S}^0$ to $\mathcal{S}^6$ obtained from it, a visual mapping between one of its slices $\mathcal{S}^3$, the corresponding birth-persistence diagram showing how the filtration mimics further slicing of $\mathcal{S}^3$ across the x-axis, and the corresponding persistence image. Such persistence images of all the slices are vectorized and stacked to form the TOPS descriptor.

5.2.2 TOPS Descriptor Computation.

To obtain the TOPS descriptor of a view-normalized point cloud, $\tilde{\mathcal{P}}$, first, it is rotated by an angle α about the y-axis to obtain a suitably aligned point cloud $\hat{\mathcal{P}}$. Then, we slice $\hat{\mathcal{P}}$ along the z-axis to get slices $\mathcal{S}^i$, where $i \in \mathbb{Z} \cap [0, \frac{h}{\sigma_1}]$. Here, h is the dimension of the axis-aligned bounding box of $\hat{\mathcal{P}}$ along the z-axis, and σ_1 is the 'thickness' of the slices. Let $p = (p_x, p_y, p_z)$ be a point in $\hat{\mathcal{P}}$. The slices $\mathcal{S}^i$ are then obtained as follows.

$$\mathcal{S}^i := \left\{ p \in \hat{\mathcal{P}} \mid i\sigma_1 \leq p_z < (i+1)\sigma_1 \right\}. \tag{5.1}$$

Let $s = (s_x, s_y, s_z)$ represent a point in $\mathcal{S}^i$. For every slice $\mathcal{S}^i$, we modify the z-coordinates $\forall s \in \mathcal{S}^i$ to s'_z, where $s'_z = i\sigma_1$.

We then design a descriptor function to construct a filtration of simplicial complexes from every slice. It mimics further slicing of a slice along the x-axis through the filtration. Therefore, when persistent homology is applied to the constructed filtration, the shape of the slice is captured in the resulting PD and PI.

To construct the filtration, first, we compute a set of origin points, $\mathcal{O}^i$, and a set of termination points, $\mathcal{T}^i$, for every slice $\mathcal{S}^i$. Let $o = (o_x, o_y, o_z)$ and $t = (t_x, t_y, t_z)$ represent a

point in $\mathcal{O}^i$ and $\mathcal{T}^i$, respectively. The sets $\mathcal{O}^i$ and $\mathcal{T}^i$ are obtained as follows.

$$\begin{aligned}\mathcal{O}^i \;:=\; & \Big\{o, \forall j \in \mathbb{Z} \cap [0, \tfrac{w}{\sigma_2}] \mid o_x = (j+1)\sigma_2, \\ & o_y = \inf(\{s_y \forall s \in \mathcal{S}^i \mid j\sigma_2 \leq s_x < (j+1)\sigma_2\}), \\ & o_z = i\sigma_1 + \epsilon_1\Big\}, \end{aligned} \tag{5.2}$$

$$\begin{aligned}\mathcal{T}^i \;:=\; & \Big\{t, \forall j \in \mathbb{Z} \cap [0, \tfrac{w}{\sigma_2}] \mid t_x = (j+1)\sigma_2, \\ & t_y = \sup(\{s_y \forall s \in \mathcal{S}^i \mid j\sigma_2 \leq s_x < (j+1)\sigma_2\}), \\ & t_z = i\sigma_1 + \epsilon_2\Big\}. \end{aligned} \tag{5.3}$$

Here, w is the dimension of the axis-aligned bounding box of $\mathcal{S}^i$ along the x-axis, σ_2 represents the 'thickness' of a slice if further slicing of $\mathcal{S}^i$ is performed along the x-axis, and ϵ_1, ϵ_2 are arbitrarily small positive constants with $2\epsilon_1 < \epsilon_2$. For every slice $\mathcal{S}^i$, we then modify the x-coordinates $\forall s \in \mathcal{S}^i$ to s'_x such that if $j\sigma_2 \leq s_x < (j+1)\sigma_2$, then $s'_x = (j+1)\sigma_2$.

The descriptor function, f, to construct a filtration from every $\mathcal{S}^i$ is then defined as

$$f(a,b) = \begin{cases} 0 & \text{if } a, b \in \mathcal{T}^i \\ g(a,b) & \text{otherwise,} \end{cases} \tag{5.4}$$

where $a = (a_x, a_y, a_z)$ and $b = (b_x, b_y, b_z)$ are any two points in $\mathcal{S}^i \cup \mathcal{O}^i \cup \mathcal{T}^i$. The function g is computed as follows.

$$g(a,b) = \begin{cases} \infty & \text{if } a_x \neq b_x \text{ or } |a_z - b_z| = \epsilon_2 \\ a_x + |a_y - b_y| & \text{otherwise.} \end{cases} \tag{5.5}$$

The descriptor function, f, creates a connected component (0-dimensional hole) at time 0 comprising all the termination points. It ensures (through the function g) that other connected components appear at times representing x-coordinates of the corresponding slices (along the x axis). The function g also ensures that the components only connect with other components corresponding to the same slice, or with the connected component of termination points. Applying persistent homology to the filtrations gives us a PD for every $\mathcal{S}^i$. We filter every PD such that for each unique value of birth, only the point with the highest persistence

(difference between death and birth times) is retained in the PD. Consequently, every point in the PD represents a slice, and its persistence represents the length of the slice. Last, we compute PIs from the PDs, vectorize and stack them into a single descriptor. Fig. 5.2 illustrates the PD and PI generation for one slice of a sample object.

5.2.3 Library Generation.

Object point clouds generated from depth images are partial, and the degree of missingness depends on the camera's pose relative to the object. Therefore, we consider synthetic depth images corresponding to all the possible views of all the objects for library generation. We divide them into three groups called the front, side, and top sets denoted by V_f, V_s, and V_t, respectively. Note that we do not consider the depth images of occluded objects in our training set. Instead, we incorporate the principle of object unity, which is used by humans to recognize the presence of occlusion and associate the visible part of an occluded object with the original unoccluded object in their memory. As an example, Fig. 5.1 shows that the PIs for a mustard bottle in the presence and absence of occlusion have similarities; the PIs for the slices unaffected by occlusion in Fig. 5.1b are similar to the PIs of the corresponding slices in Fig. 5.1a. We generate a library of trained classifiers in a manner that enables us to leverage such similarity for object unity-based recognition. The following subsections describe the division of the training set images into V_f, V_s, and V_t and the generation of trained classifiers from them. Fig. 5.3 illustrates the overall library generation procedure.

Division into V_f, V_s, and V_t:

First, we define some of the terminology used in subsequent text. For every unoccluded object, the orthographic views whose projections have the largest and smallest bounding box areas (among the three main views) are termed as its front view and top view, respectively. Accordingly, the view whose projection has neither the largest nor the smallest bounding box area is called the side view. For instance, the top left corner of Fig. 5.3 depicts the front, side, and top views of a mustard bottle. As stated previously, we consider all the possible

views of all the objects in our training set. We use the proximity of the viewpoints to the front, side, and top views to divide them into three groups V_f, V_s, and V_t. Viewpoints that are approximately equidistant from multiple views are considered in groups corresponding to all the respective views.

Generation of Trained Classifiers:

For V_f, V_s, and V_t, we separately generate the corresponding point clouds and scale them by a factor of σ_s. Next, we perform view normalization (using the minimal-volume bounding box computation from Open3D [212]) and augment the three sets by mirroring all the view-normalized point clouds across the coordinate axes (in place). We then compute the TOPS descriptor by only considering the first slice (after alignment) of every point cloud and train a classification model (e.g., SVM) using those descriptors. Next, we consider the first two slices of the aligned point cloud, compute the corresponding TOPS descriptor, and train another classification model. We continue this procedure until the last slice of the largest object ($N_s{}^{\text{th}}$ slice) has been considered. As the number of slices differs from object to object, we appropriately pad the descriptor before training the models to ensure that the input vectors are all of the same size. As a result, we obtain three sets of trained classification models from V_f, V_s, and V_t, denoted as M_f, M_s, and M_t, respectively.

5.2.4 Test-time Recognition.

Given a test image of a cluttered scene and the corresponding instance segmentation map, we first generate the individual point clouds of all the objects in the scene. Similar to the training stage, we scale the point clouds by a factor of σ_s. Consider an object point cloud $\mathcal{P}_t$ obtained from the test image (after scaling). To recognize $\mathcal{P}_t$, first, we perform view normalization to obtain $\tilde{\mathcal{P}}_t$. Next, we obtain the areas of the three orthogonal faces of $\tilde{\mathcal{P}}_t$'s axis-aligned bounding box and the *curvature flow* of the surfaces corresponding to those faces. We then use area and curvature flow-based heuristics to select suitable model

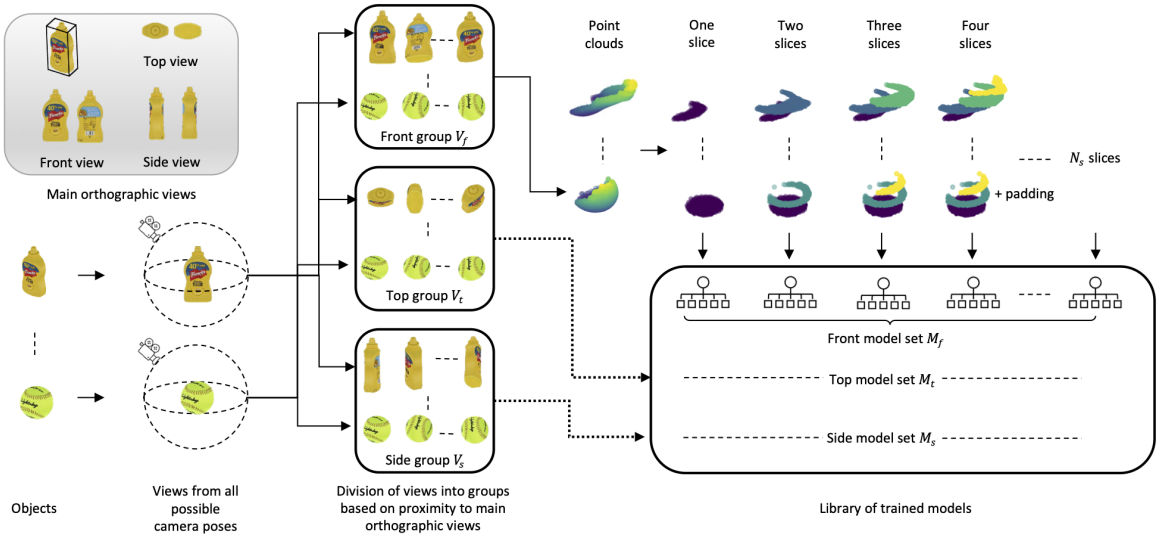


Figure 5.3: Visualization of the procedure for generating a library of training classifiers. Depth images corresponding to all possible relative camera poses for all the objects are obtained. Subsequently, all the images are divided into three sets based on their proximity to the corresponding main orthographic views. Multiple classifier models are trained for each set by incrementally considering object slices during descriptor computation.

set(s) from the library for recognition. The following subsections describe our definition of curvature flow, the selection of model sets, and prediction using selected model sets.

Curvature Flow:

We define curvature flow in a manner analogous to optical flow in computer vision. Curvature flow aims to calculate the change in curvature at every point in a surface with respect to a chosen normal direction. We obtain the individual surfaces corresponding to the faces of $\tilde{\mathcal{P}}_t$'s axis-aligned bounding box by clustering the points in the view-normalized point cloud $\tilde{\mathcal{P}}_t$. This clustering is performed using the angular distances of each point's estimated (outward) normal from the (outward) normals of the axis-aligned bounding box. We then compute the curvature flow of the individual surfaces as follows. First, we compute the curvature, $\mathcal{C}$, at every point in the point cloud from the eigenvalues of the covariance matrix obtained by considering its nearest neighbors (computed using the *kd*-tree algorithm). We then define a curvature constancy constraint, analogous to the brightness constancy constraint in optical flow computation, as follows.

$$\mathcal{C}(x, y, z) = \mathcal{C}(x + \Delta x, y + \Delta y, z + \Delta z), \tag{5.6}$$

Considering the Taylor series approximation and ignoring higher order terms, it follows that

$$\frac{\partial \mathcal{C}}{\partial x}\Delta x + \frac{\partial \mathcal{C}}{\partial y}\Delta y + \frac{\partial \mathcal{C}}{\partial z}\Delta z = 0. \tag{5.7}$$

For a surface whose corresponding normal of the axis-aligned bounding box is along the positive z-axis, the curvature flow at every point in the surface is then computed as

$$\frac{\partial \mathcal{C}}{\partial z} = -\left(\frac{\partial \mathcal{C}}{\partial x}\frac{\Delta x}{\Delta z} + \frac{\partial \mathcal{C}}{\partial y}\frac{\Delta y}{\Delta z}\right). \tag{5.8}$$

Similarly, the curvature flow at points in the other surfaces (with normals along the x and y axes) can be computed. We then use the interquartile range of the absolute values of curvature flow at all the surface points to represent the surface's overall curvature flow.

Model Sets Selection:

First, we identify the primary face (the face that is in direct view of the camera) among the three orthogonal faces of $\tilde{\mathcal{P}}_t$'s axis-aligned bounding box. If the primary face has the maximum area among the three faces, we only use the model sets M_f and M_s for recognition. We base this choice on the following heuristic. An object point cloud corresponding to any of the views used to obtain M_t (i.e., the views in the top set, V_t) cannot have a primary face with the maximum area among the three orthogonal faces of its axis-aligned bounding box. Similarly, if the primary face has the minimum area among the three faces, we select model sets M_s and M_t for recognition. The same heuristics can also be applied to select the model sets when two of the three faces have substantially similar areas and are greater (or smaller) than the area of the third face. When all the three faces have substantially similar areas, we consistently select the model set M_f for prediction.

If the primary face has neither the minimum nor the maximum area, we use a heuristic based on the curvature flow of the surfaces corresponding to the faces. If the surface corresponding to the primary face has the least overall curvature flow, we use the model sets M_s and M_t for recognition. Otherwise, we use the M_f and M_s model sets. We base this choice on a heuristic that for any object point cloud corresponding to the views in the top set (i.e., V_t), the surface corresponding to its primary face has the least overall curvature flow among the three surfaces.

Remark: We observe that the above heuristics largely hold when the objects are not heavily occluded. For the purpose of this work, we define heavy occlusion as follows. Consider an unoccluded object. Let a_1, a_2, and a_3 be the areas of the bounding boxes of the projections corresponding to its front, side, and top views, respectively. Let the primary faces corresponding to these views be termed as the front, side, and top face. Let $a_1^{'}, a_2^{'}$, and $a_3^{'}$ be the areas of the bounding boxes corresponding to the front, side, and top faces under occlusion. Consider any two face areas, say a_i and a_j ($i \neq j$ and $i, j \in \{1, 2, 3\}$), such that $a_i \; R_{ij} \; a_j$, where R_{ij} represents one of $<, >$, or $=$. We consider the object to be heavily

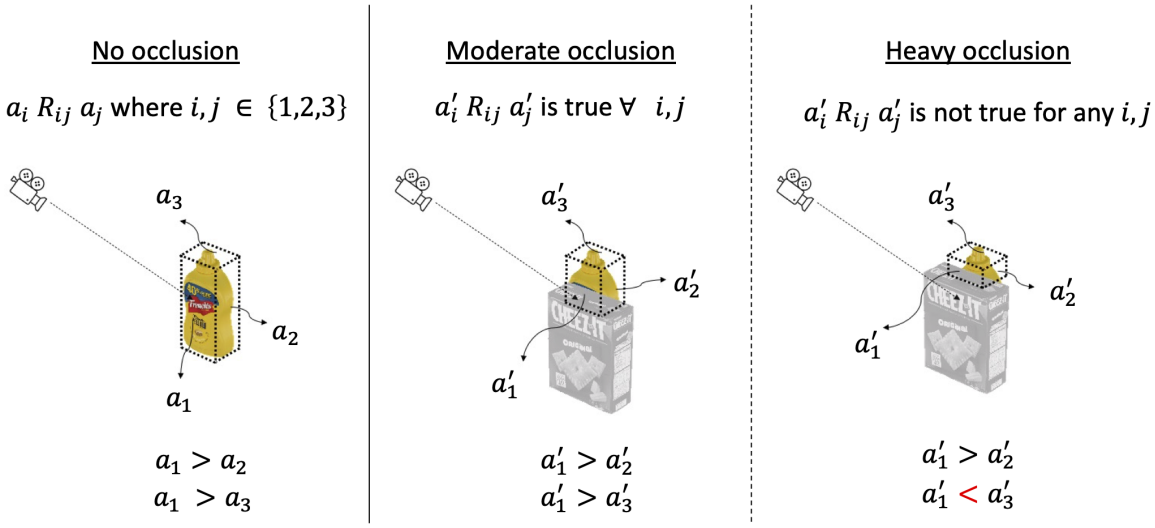


Figure 5.4: Illustration depicting instances of no occlusion, moderate occlusion, and heavy occlusion (as defined in this work) for a mustard bottle.

occluded if $a_i^{'}\ R_{ij}\ a_j^{'}$ is not true. If $a_i^{'}\ R_{ij}\ a_j^{'}$ is true, we consider the object to be moderately occluded. Fig. 5.4 shows instances of no occlusion, moderate occlusion, and heavy occlusion for a sample object.

Prediction using Selected Model Sets:

To recognize $\tilde{\mathcal{P}}_t$ using the models from the selected model set(s), first, we determine if the object corresponding to $\tilde{\mathcal{P}}_t$ is occluded. For this purpose, we obtain the object's contour (in the segmentation map) and use the neighboring pixels' segmentation labels and depth values. For every pixel in the contour, we assume it is part of an occlusion boundary if one or more of its neighboring pixels are labeled as object instances and have a depth value smaller than its own depth value.

If the object is occluded, we rotate $\tilde{\mathcal{P}}_t$ by π about the z-axis to ensure that the first slice on the occluded end of the object is not the first slice during subsequent TOPS descriptor computation. The corresponding $\hat{\mathcal{P}}_t$ generated during descriptor computation has one or

more missing slices. Therefore, we identify the number of slices (say n_s) in $\hat{\mathcal{P}}_t$. Then, from the selected model set(s), we perform recognition using the model(s) that considers only n_s slices. In doing so, we match the part of the object that is unaffected due to occlusion with the parts of unoccluded objects used for training the models in the library, thereby incorporating the idea of object unity. If the object corresponding to $\tilde{\mathcal{P}}_t$ is not occluded, we compute its TOPS descriptor and use the model(s) that considers the highest number of slices (with appropriate padding to the descriptor).

In the case where multiple models are used, the final prediction is obtained as follows. First, we eliminate any invalid predictions based on the known relation ($>, <$, or $=$) between the face areas of the minimum volume bounding box of the predicted object class. If none of the predictions are invalid, we consider the prediction with the highest probability to be the final prediction.

5.3 Datasets

5.3.1 OCID Dataset

First, we evaluate the performance of our framework on the YCB10 subset of the OCID dataset [164], a benchmark RGB-D dataset for object recognition in cluttered environments. It consists of sequences of increasingly cluttered scenes with up to ten objects. The sequences are divided into three types - cuboidal (all the objects have sharp edges), curved (all the objects have smooth curved surfaces), and mixed (both cuboidal and curved objects are present). Each sequence is recorded using two RGB-D cameras, the 'lower camera' and the 'upper camera,' positioned at different heights and angles to mimic the configurations of existing robotic systems. It also provides temporally smoothed, point-wise labeled point clouds for every frame in the sequence.

Table 5.1: Comparative summary of existing indoor scenes datasets and our UW-IS Occluded dataset.

	RGB-D Object [98]	RGB-D Scenes [97]	LM-O [24, 80]	HOPE Image [179]	Rutgers APC [141]	BigBird [160]	ARID [111]	OCID [164]	UW-IS [151]	UW-IS Occluded
Different environments/fixtures	✗	✓	✗	✓	✗	✗	✓	✓	✓	✓
Object occlusion	✗	✓	✓	✓	✓	✗	✓	✓	✗	✓
Occlusion categorization (e.g., low, high)	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
Lighting variation	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓
Lighting categorization	✗	✗	✗	✓	✗	✗	✓	✗	✓	✓
Depth	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
#object classes	51	5	8	28	24	125	51	89	14	20
Dataset size (#scene images)	250000	11427	1200	238	10368	75000	6000+	2346	347	8456
Object segmentation annotations	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓
6D object pose annotations	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓

5.3.2 UW-IS Occluded Dataset

Although several indoor scene datasets are available, there is a dearth of datasets with a large enough set of object types, poses, and arrangements with a systematic variation in environmental conditions and object occlusion. Toward building such a dataset, our previous work [151] presented the UW-IS dataset, an RGB dataset for evaluating object recognition performance in multiple indoor environments. However, the dataset was limited as it only included scenes that had plain backgrounds and no object occlusion. Therefore, in this work, we extend our contribution by presenting an RGB-D dataset, the UW-IS Occluded dataset. Table 5.1 presents a comparative summary of some of the existing benchmark datasets and our UW-IS Occluded dataset.

Similar to the UW-IS dataset, our dataset comprises two completely different indoor environments. The first environment is a lounge where the objects are placed on a tabletop. The second environment is a mock warehouse setup where the objects are placed on a shelf. For each of these environments, we have RGB-D images from 36 videos, comprising five to seven objects each, taken from distances up to approximately 2 m using an Intel

RealSense D435 camera. The videos cover two different lighting conditions, i.e., bright and dim, with eighteen videos each and three different levels of occlusion for three different object categories. Specifically, the dataset considers objects from the YCB [32] and BigBird [160] datasets belonging to the kitchen, food items, and tools/miscellaneous categories. As shown in Fig. 5.5, the first level of occlusion is where objects are placed such that there is no object occlusion. The second level of object occlusion is where some occlusion occurs, while the third level is where the objects are placed extremely close together to represent scenarios where the objects are occluded to a higher degree. Overall, the dataset considers 20 object classes and consists of 8,456 images, which have a total of 42,902 object instances. We provide instance segmentation masks for all the images generated semi-automatically using LabelFusion [118]. For potential use as a benchmark to evaluate other robot vision tasks such as pose estimation, we also provide 6D pose annotations for the dataset generated using LabelFusion. The dataset is publicly available at `https://doi.org/10.6084/m9.figshare.20506506`.

5.4 Experiments

We use synthetic training data and evaluate THOR on real-world OCID and UW-IS Occluded datasets. We report performance comparisons with two state-of-the-art point cloud classification methods, DGCNN [190] and SimpleView [70], on both the datasets. On the benchmark OCID dataset, we also perform ablation studies to draw further insights into the role of the different components of THOR. Specifically, the performance of the TOPS descriptor is compared with the widely-used Clustered Viewpoint Feature Histogram (CVFH) [7] and Ensemble of Shape Functions (ESF) [194] descriptors, and a topological descriptor called Signature of Topologically Persistent Points (STPP) [13]. In addition, we evaluate *Slice-ESF*, a version of THOR modified to use ESFs of point cloud slices instead of TOPS, to examine the role of slicing and the classifier library in THOR. We also examine the role of the area and curvature flow-based heuristics for model sets selection in THOR on the UW-IS Occluded dataset. Furthermore, we report the performance of 2D shape-based topological features, i.e., sparse PI features [151] on it to illustrate the importance of depth-based 3D

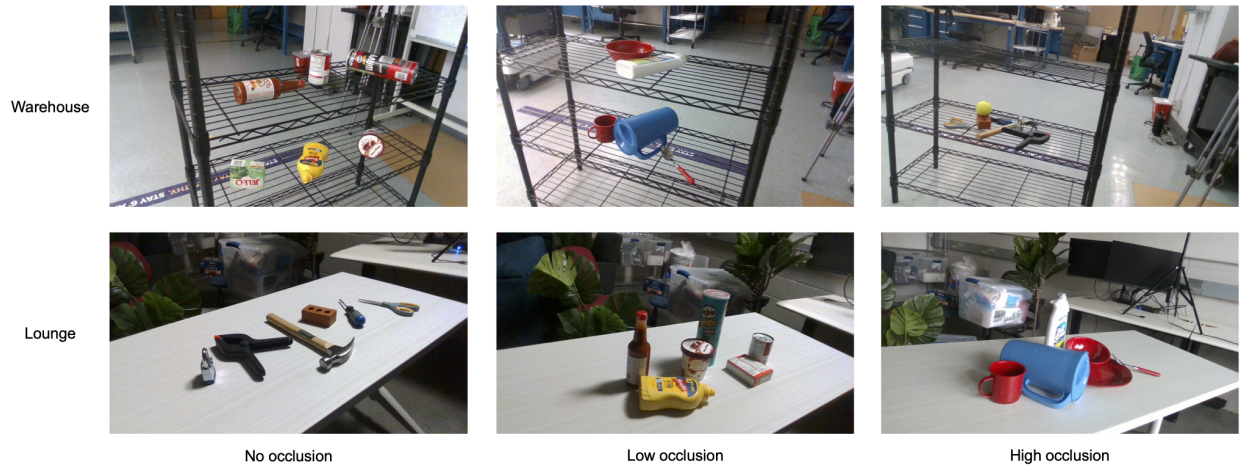


Figure 5.5: Representative images from the UW-IS Occluded dataset. The first row shows images from the warehouse environment, and the second shows images from the lounge. The first two columns show scenes with one kind of lighting condition and 'no occlusion' and 'low occlusion' scenarios, respectively. The third column shows scenes with another lighting condition and high occlusion between the objects.

shape features. A study on choice of parameters for TOPS computation is also performed. Lastly, we implement THOR on a real-world robot to demonstrate its usefulness.

5.4.1 Implementation Details

Synthetic Training Data Generation:

We obtain the synthetic depth images of objects for our training set using the Panda3D [128] framework and object meshes from [32]. As a side note, alternate rendering methods such as BlenderProc [46] and Kubric [72] can also be used. For our experiments, we only consider objects from the OCID and UW-IS Occluded datasets for which scanned meshes are available. The training set depth images are obtained as follows. We place the objects (one at a time) at the center of an imaginary sphere (as shown in Fig. 5.3). We then obtain depth images corresponding to the camera positions on the sphere considering polar angles ranging from $[0, \pi]$ in increments of $\frac{\pi}{36}$ and azimuthal angles ranging from $[0, 2\pi)$ in increments of $\frac{\pi}{36}$. The depth images are generated at a scale of 0.001m (i.e., a distance of 1mm in the real world corresponds to a value of 0.001 in the depth image). The code for the synthetic data generation and the subsequent training and testing of THOR is available at `https://github.com/smartslab/THOR.git`

TOPS Computation:

We set the scale factor $\sigma_s = 2.5$ and set $\sigma_1 = 0.1$, $\sigma_2 = 2.5 \times 10^{-2}$, and $\alpha = \frac{\pi}{4}$ to compute suitable TOPS descriptors. We refer the reader to Section 5.4.3 for a discussion on these choices. An approximate computation method is used to generate the PDs and the Persim package in Scikit-TDA Toolbox is used to generate the PIs for the descriptors. We generate PIs of size 32×32 by choosing a pixel size of 2.5×10^{-2}, and setting the birth and persistence ranges to $(0, 0.75)$. Also, the PIs are generated by considering a kernel spread of 2.5×10^{-4} and a linear weight function [75].

Library Generation for THOR:

We consider libraries with two types of classifiers, namely, SVM and Multi-layer Perceptron (MLP). Platt scaling is used when training an SVM classifier library to obtain prediction probabilities. For training an MLP classifier library, we train five-layer fully connected networks using the Adam optimizer to optimize the categorical cross-entropy loss over 100 epochs. The learning rate is set to 10^{-2} for the first 50 epochs and is decreased to 10^{-3} for the subsequent 50 epochs. We use workstations with GeForce GTX 1080 and 1080 Ti GPUs running Ubuntu 18.04 LTS for training (and testing) the models. Separate libraries of trained SVMs and MLPs are generated for every sequence using synthetic depth images in the case of the OCID dataset. We train one library each (considering seventeen object classes) with SVM and MLP as the classifiers for the UW-IS Occluded dataset.

THOR Testing:

In the case of the UW-IS Occluded dataset, we generate point clouds from depth images using the known camera intrinsic parameters and a depth scale of 0.001m (i.e., a dimension of 1mm in the real world corresponds to 0.001 in the point cloud coordinate system). We then consistently perform the following pre-processing steps for every object point cloud in all the environmental conditions. First, uniform voxel grid-based downsampling is performed using a voxel size of 0.03 using Open3D. Next, radius outlier removal is performed to remove the points with fewer than 220 neighboring points in a sphere of radius 5×10^{-2} around them. In the case of the OCID dataset, we do not perform outlier removal as the point clouds are obtained from temporally averaged depth images. Subsequently, TOPS descriptors are computed as described in Section 5.4.1, and predictions are obtained using the selected model(s) from the library. For model sets selection, we implement a tolerance threshold when comparing the areas of the faces of the bounding box because real-world depth data is noisy, and minimal-volume bounding box computation is approximate. Specifically, we consider a face area substantially greater than another if they differ by more than 20% (for

both the datasets). We report test results from five-fold cross-validation for THOR and all the comparison methods.

Comparison Methods:

We use the implementation provided by [70] for DGCNN and SimpleView. We use the same training set of point clouds generated from synthetic depth images for both the methods. As both methods require consistent alignment of point clouds, we use the view-normalized point clouds for training models. We consistently use the DGCNN protocol for point cloud data augmentation and the best test model selection scheme defined in [70] for both methods. We use the point cloud library implementations [146] to compute the CVFH and ESF descriptors, and the GUDHI library [173] to implement STPP (with both Betti 0 and Betti 1 features), as described in [13]. Since Sparse PI features are 2D shape features extracted from segmentation maps, we train the recognition module as described in [151] using the segmentation maps corresponding to our synthetic depth images training set. At test time, the Sparse PI features for the objects in the scene image are generated from the corresponding instance segmentation maps.

5.4.2 Results on the Benchmark OCID Dataset

Comparison with End-to-end Models:

We compare THOR's performance with two end-to-end deep learning-based models, DGCNN and SimpleView. Table 5.2 shows the recognition accuracies when the test sequences are recorded using the camera placed at a lower height, i.e., the lower camera. We note that THOR is better than DGCNN and SimpleView on all the test sequences with curved objects. THOR also has the best performance on all the cuboidal object sequences. In the case of mixed objects, THOR achieves the highest performance in all but two sequences. In particular, THOR is better at distinguishing objects with similar geometry, such as the tennis ball, golf ball, and baseball (see S-25 in Fig. 5.6). Additionally, we observe that the

overall performance of all the methods is better for the cuboidal objects than the curved objects, likely because the cuboidal objects have larger dimensional variations than the curved objects. In the case of mixed objects placed on the table, we observe that THOR performs even better than the sequences with curved and cuboidal objects. The performance of the other methods also improves, but not enough to outperform our method. As shown in Fig. 5.6, only THOR correctly identifies the relatively heavily occluded objects in S-31, i.e., the pitcher base and the tomato soup can.

We observe similar trends in Table 5.3 where THOR outperforms both DGCNN and SimpleView when the test sequences are recorded using the camera placed at a higher height, i.e., the upper camera. Fig. 5.7 shows sample results on the upper camera recordings of the test sequences. We note here that the sequences recorded with the upper camera are more likely to comprise objects views from the top view group, i.e., V_t. In general, recognizing objects from their top view is more challenging than recognizing them from the front or side view. Therefore, the overall performance of DGCNN, SimpleView, and THOR with the SVM library drops compared to the lower camera case. In contrast, the average performance of THOR with the MLP library remains unchanged across the two cameras. We refer the reader to Section 5.5 for further discussion on the performance.

Overall, these results show that THOR is better at recognizing occluded objects in cluttered environments than DGCNN and SimpleView.In addition, our method's notably higher recognition accuracy in the case of sequences with objects of similar geometry (i.e., curved and cuboidal) shows that the two-fold slicing in the TOPS descriptor (see equations (5.1), (5.4), and (5.5)) captures the local and global shape features more robustly than both DGCNN and SimpleView.

Ablation Studies:

First, we evaluate the TOPS descriptor outside the THOR framework and compare its performance with other point cloud descriptors CVFH, ESF, and STPP. Note that the TOPS descriptors are appropriately padded for this experiment to ensure all the descriptors are of

Table 5.2: Comparison of mean recognition accuracy (in %) of THOR with end-to-end models on the OCID dataset sequences recorded using the lower camera (best in bold).

Place	Scene type	Seq. ID	THOR		DGCNN	SimpleView
			SVM library	MLP library		
Table	Curved	S-25	**64.03±0.44**	**65.31±2.03**	41.84±1.82	35.99±3.64
		S-26	**61.85±0.74**	57.18±2.25	42.44±3.74	56.33±2.54
		S-35	**55.78±0.48**	49.83±2.07	24.81±2.96	19.15±2.63
		S-36	**71.79±0.00**	61.35±0.49	36.13±0.35	63.56±2.34
	Cuboid	S-23	**71.68±0.36**	**72.82±0.94**	48.10±1.96	61.08±0.66
		S-24	**70.22±0.26**	57.01±4.28	42.18±0.57	68.08±1.64
		S-33	58.17±0.25	**65.90±3.27**	43.38±1.25	39.94±1.08
		S-34	**79.46±0.22**	65.09±3.38	26.56±2.50	55.89±0.94
	Mixed	S-21	**74.78±0.32**	68.02±1.06	43.89±2.52	**75.40±1.65**
		S-22	**77.64±0.00**	62.11±3.77	57.44±3.86	61.66±4.12
		S-31	67.03±0.86	**74.59±1.34**	49.30±3.07	65.59±3.32
		S-32	**70.46±0.00**	67.16±1.76	47.13±1.71	60.31±4.12
Floor	Curved	S-05	59.11±0.00	**65.75±2.56**	29.84±1.32	43.1±5.08
		S-06	**78.18±0.46**	**79.01±2.72**	36.97±3.06	56.63±1.76
		S-11	49.47±0.91	**67.84±2.39**	39.03±5.37	47.47±4.40
		S-12	**77.47±0.34**	70.79±0.90	43.33±2.31	76.52±2.25
	Cuboid	S-03	**69.87±1.16**	63.16±1.07	51.76±1.93	60.72±2.60
		S-04	**64.81±1.11**	55.62±3.35	39.06±2.66	63.93±3.56
		S-09	65.31±0.00	**73.96±3.64**	49.80±4.28	**75.93±2.11**
		S-10	**81.11±1.11**	**84.32±3.11**	40.62±1.54	74.12±2.94
	Mixed	S-01	**93.09±0.56**	76.78±1.66	39.88±4.41	76.55±2.50
		S-02	70.13±0.20	73.04±1.26	73.03±2.70	**86.30±1.02**
		S-07	63.17±0.32	88.53±1.57	75.19±2.46	**95.31±1.26**
		S-08	37.96±1.85	**43.43±1.30**	37.07±0.94	**47.41±5.79**
All sequences			**68.88 ± 0.10**	66.67 ± 0.22	47.33 ± 0.31	62.54 ± 0.23

Table 5.3: Comparison of mean recognition accuracy (in %) of THOR with end-to-end models on the OCID dataset sequences recorded using the upper camera.

Place	Scene type	Seq. ID	THOR		DGCNN	SimpleView
			SVM library	MLP library		
Table	Curved	S-25	**60.34 ± 0.75**	49.82 ± 1.79	35.52 ± 3.58	41.83 ± 2.37
		S-26	**65.52 ± 0.22**	64.48 ± 2.49	34.32 ± 2.47	56.77 ± 4.49
		S-35	38.36 ± 0.30	**58.91 ± 2.32**	15.25 ± 3.58	21.72 ± 5.11
		S-36	50.56 ± 0.97	55.67 ± 1.15	32.77 ± 4.42	**68.93 ± 1.46**
	Cuboid	S-23	60.37 ± 0.37	**66.24 ± 1.03**	39.75 ± 3.26	50.31 ± 3.82
		S-24	**80.93 ± 0.45**	70.38 ± 3.39	36.71 ± 3.49	51.83 ± 2.00
		S-33	54.59 ± 0.92	**61.94 ± 1.42**	29.87 ± 3.74	37.75 ± 3.47
		S-34	**71.11 ± 0.00**	70.22 ± 1.64	33.96 ± 2.46	62.97 ± 2.43
	Mixed	S-21	**75.10 ± 0.22**	73.75 ± 2.01	33.49 ± 2.49	**74.69 ± 4.07**
		S-22	52.95 ± 0.00	**62.54 ± 1.46**	42.34 ± 4.57	**58.83 ± 3.33**
		S-31	68.64 ± 0.87	**75.87 ± 0.95**	40.37 ± 3.43	54.22 ± 2.17
		S-32	**89.20 ± 0.25**	72.47 ± 1.32	34.10 ± 4.52	64.33 ± 5.37
Floor	Curved	S-05	55.37 ± 0.33	**64.17 ± 2.94**	14.29 ± 1.84	38.16 ± 3.39
		S-06	**69.97 ± 0.00**	**69.78 ± 3.63**	47.20 ± 2.91	55.25 ± 2.56
		S-11	54.44 ± 0.56	**58.77 ± 2.74**	34.42 ± 0.58	45.80 ± 4.26
		S-12	61.31 ± 0.44	58.62 ± 1.74	53.56 ± 3.53	**87.19 ± 1.36**
	Cuboid	S-03	**73.13 ± 0.67**	59.78 ± 1.07	34.33 ± 2.07	64.39 ± 1.45
		S-04	**66.85 ± 0.34**	49.54 ± 1.34	30.52 ± 1.29	**68.63 ± 1.46**
		S-09	58.36 ± 1.02	49.59 ± 1.35	40.44 ± 4.53	**66.26 ± 1.95**
		S-10	66.94 ± 0.00	**77.04 ± 2.02**	27.72 ± 0.62	73.02 ± 0.89
	Mixed	S-01	**67.61 ± 1.24**	64.06 ± 1.16	19.06 ± 1.67	10.28 ± 2.47
		S-02	62.68 ± 0.39	76.61 ± 1.65	58.77 ± 1.91	**86.18 ± 0.55**
		S-07	69.07 ± 0.83	81.46 ± 1.78	64.17 ± 2.71	**85.06 ± 0.30**
		S-08	57.56 ± 0.00	**78.44 ± 2.38**	47.17 ± 6.64	59.61 ± 3.61
All sequences			65.43 ± 0.10	**66.50 ± 0.13**	41.45 ± 0.13	59.82 ± 0.06

the same size, regardless of the number of slices. A single MLP classifier is trained for each of the four descriptors using the synthetic training data and used for prediction at test time. Tables 5.4 and 5.5 show that the performance of the TOPS descriptor is substantially better than that of the widely-used features CVFH and ESF, irrespective of the camera view. TOPS also outperforms the topological descriptor STPP by a large margin.

Further, we observe that the TOPS descriptor (using a single MLP classifier) performs slightly better than THOR with the MLP library. However, unlike THOR, a slight drop in performance is witnessed when the TOPS descriptor is tested on the upper camera recordings as compared to its performance on the lower camera recordings. Therefore, we believe THOR with an MLP classifier library is better equipped to perform recognition even in challenging scenarios where the objects are viewed from the top. This observation is further supported by our experiments investigating the role of slicing and the classifier library in THOR. Specifically, we obtain the performance of Slice-ESF, a version of THOR modified to use ESFs of point cloud slices instead of TOPS, on the lower and upper camera recordings. Slice-ESF considers slices of the object point clouds as in the case of TOPS, and ESF descriptors of all the slices are stacked to obtain the final descriptor. Test-time recognition, in this case, is performed using an MLP library, just as in the case of THOR. Tables 5.4 and 5.5 show that Slice-ESF performs better than ESF for both the lower and upper cameras. Moreover, the improvement in performance in the case of the upper camera is larger than that of the lower camera, indicating that recognition using a classifier library is beneficial in the case of more challenging object views.

5.4.3 Results on the UW-IS Occluded Dataset

Comparison with End-to-end Models:

We systematically examine the performance of THOR in two different environments under varying environmental conditions. Table 5.6 shows the performance in a warehouse and a lounge while considering variations in the object types. For both the environments, varying

Table 5.4: Comparison of mean recognition accuracy (in %) of TOPS with 3D shape descriptors on the OCID dataset sequences recorded using the lower camera.

Seq. ID	CVFH	ESF	STPP	Slice ESF with MLP library	TOPS
S-25	38.33 ± 0.56	38.10 ± 1.94	26.67 ± 2.42	40.26 ± 1.07	**50.88 ± 2.62**
S-26	26.67 ± 0.74	46.49 ± 0.47	17.51 ± 1.04	44.56 ± 2.62	**68.34 ± 1.57**
S-35	19.91 ± 0.85	26.67 ± 2.72	16.11 ± 2.55	37.38 ± 2.12	**59.32 ± 3.62**
S-36	16.67 ± 0.00	39.20 ± 1.20	39.16 ± 2.53	41.53 ± 1.28	**56.09 ± 3.03**
S-23	29.26 ± 2.00	45.17 ± 1.35	30.09 ± 1.01	37.98 ± 1.60	**72.70 ± 2.57**
S-24	23.52 ± 1.48	46.36 ± 1.20	27.25 ± 1.80	41.06 ± 2.90	**68.83 ± 1.98**
S-33	22.71 ± 0.99	40.98 ± 3.66	17.02 ± 0.74	42.39 ± 3.21	**70.77 ± 2.36**
S-34	14.76 ± 0.73	36.52 ± 3.06	42.42 ± 2.86	39.95 ± 2.76	**80.77 ± 0.53**
S-21	44.79 ± 0.81	38.83 ± 1.54	44.34 ± 0.86	52.06 ± 0.81	**71.46 ± 1.8**3
S-22	37.50 ± 0.00	56.05 ± 1.43	28.84 ± 2.48	53.56 ± 0.91	**62.67 ± 2.05**
S-31	34.90 ± 0.96	52.74 ± 1.51	48.26 ± 1.00	52.65 ± 2.66	**76.67 ± 1.35**
S-32	19.24 ± 1.51	61.28 ± 2.03	54.20 ± 1.16	55.43 ± 1.21	**72.12 ± 2.02**
S-05	22.22 ± 0.00	23.81 ± 0.00	45.53 ± 1.27	48.34 ± 0.50	**67.37 ± 2.40**
S-06	28.15 ± 0.92	23.07 ± 1.49	32.64 ± 3.15	44.49 ± 2.77	**91.67 ± 0.34**
S-11	21.39 ± 2.87	33.61 ± 1.26	29.48 ± 0.73	46.26 ± 1.59	**65.50 ± 1.31**
S-12	16.30 ± 1.48	42.72 ± 0.92	46.15 ± 0.77	37.35 ± 1.03	**70.67 ± 1.76**
S-03	17.41 ± 1.26	39.89 ± 1.24	34.35 ± 1.17	31.00 ± 2.57	**69.35 ± 1.00**
S-04	20.98 ± 3.51	58.43 ± 2.30	39.44 ± 2.22	43.02 ± 2.08	**72.24 ± 1.61**
S-09	14.44 ± 1.50	20.72 ± 2.67	39.18 ± 1.27	31.12 ± 1.66	**75.83 ± 3.60**
S-10	32.33 ± 0.96	47.70 ± 1.08	25.22 ± 1.92	43.48 ± 3.90	**90.12 ± 2.19**
S-01	32.90 ± 0.76	73.20 ± 3.61	31.22 ± 0.85	42.06 ± 3.01	**83.19 ± 1.53**
S-02	35.00 ± 0.00	**62.29 ± 0.97**	40.55 ± 0.62	**60.53 ± 2.44**	**63.91 ± 1.59**
S-07	12.00 ± 0.54	69.01 ± 0.79	45.32 ± 3.99	43.02 ± 2.58	**82.31 ± 0.71**
S-08	29.62 ± 0.35	32.84 ± 0.49	19.51 ± 0.66	42.44 ± 4.31	**53.04 ± 1.31**
All seq.	29.84 ± 0.19	41.54 ± 0.17	39.72 ± 0.05	43.63 ± 0.15	**69.38 ± 0.07**

Table 5.5: Comparison of mean recognition accuracy (in %) of TOPS with 3D shape descriptors on the OCID dataset sequences recorded using the upper camera.

Seq. ID	CVFH	ESF	STPP	Slice ESF with MLP library	TOPS
S-25	44.44 ± 2.41	46.30 ± 0.37	9.89 ± 4.28	36.28 ± 0.52	**59.96 ± 2.35**
S-26	43.26 ± 0.89	50.02 ± 2.18	30.03 ± 1.66	42.45 ± 4.24	**72.00 ± 3.29**
S-35	40.72 ± 1.03	22.96 ± 2.42	12.78 ± 0.68	46.15 ± 1.60	**57.89 ± 2.84**
S-36	45.11 ± 2.54	42.35 ± 0.00	24.67 ± 5.05	45.33 ± 0.42	**55.28 ± 1.47**
S-23	11.94 ± 0.34	47.02 ± 2.86	41.03 ± 1.79	34.92 ± 3.08	**66.99 ± 0.72**
S-24	15.46 ± 0.94	40.00 ± 3.40	28.10 ± 1.13	54.10 ± 2.60	**65.96 ± 0.93**
S-33	12.50 ± 0.00	18.44 ± 1.10	33.81 ± 3.59	34.97 ± 2.44	**63.87 ± 3.48**
S-34	22.19 ± 1.26	44.73 ± 4.18	41.67 ± 3.64	59.82 ± 4.53	**73.78 ± 1.23**
S-21	29.09 ± 0.97	26.88 ± 2.03	31.91 ± 1.57	56.39 ± 0.34	**71.11 ± 2.38**
S-22	30.75 ± 0.50	54.54 ± 1.88	17.29 ± 1.28	60.19 ± 1.46	**65.07 ± 2.04**
S-31	20.88 ± 0.55	47.17 ± 2.02	24.23 ± 1.21	50.70 ± 1.37	**72.74 ± 1.07**
S-32	12.31 ± 0.93	60.10 ± 0.46	40.48 ± 2.31	61.75 ± 0.54	**72.50 ± 0.73**
S-05	51.08 ± 1.06	25.20 ± 0.88	23.89 ± 2.46	40.76 ± 1.76	**65.15 ± 1.35**
S-06	24.81 ± 1.18	23.15 ± 1.30	26.66 ± 1.22	42.58 ± 3.42	**73.89 ± 1.29**
S-11	38.70 ± 2.80	34.72 ± 0.00	11.11 ± 0.00	43.37 ± 1.43	**62.86 ± 3.05**
S-12	18.59 ± 1.15	47.00 ± 0.78	45.44 ± 1.19	45.10 ± 2.58	**68.56 ± 1.54**
S-03	24.20 ± 0.84	51.78 ± 3.14	33.34 ± 2.55	42.11 ± 1.94	**58.98 ± 1.55**
S-04	13.06 ± 1.36	58.43 ± 2.03	41.11 ± 2.74	34.31 ± 1.32	**69.52 ± 1.94**
S-09	13.61 ± 0.52	24.92 ± 0.45	21.49 ± 2.78	48.26 ± 3.72	**57.81 ± 0.77**
S-10	17.78 ± 2.72	44.37 ± 1.06	23.75 ± 1.50	51.77 ± 3.19	**76.89 ± 4.20**
S-01	26.19 ± 0.91	51.00 ± 1.60	24.85 ± 2.23	47.57 ± 1.44	**68.13 ± 0.83**
S-02	26.86 ± 1.57	57.21 ± 0.64	52.45 ± 1.73	57.14 ± 0.61	**74.40 ± 1.45**
S-07	21.80 ± 0.81	56.54 ± 2.64	47.30 ± 2.05	49.19 ± 1.30	**79.84 ± 1.76**
S-08	33.00 ± 1.07	33.33 ± 0.00	21.36 ± 1.48	35.70 ± 4.51	**76.22 ± 1.14**
All seq.	26.79 ± 0.10	40.54 ± 0.27	35.36 ± 0.12	47.80 ± 0.21	**68.62 ± 0.06**

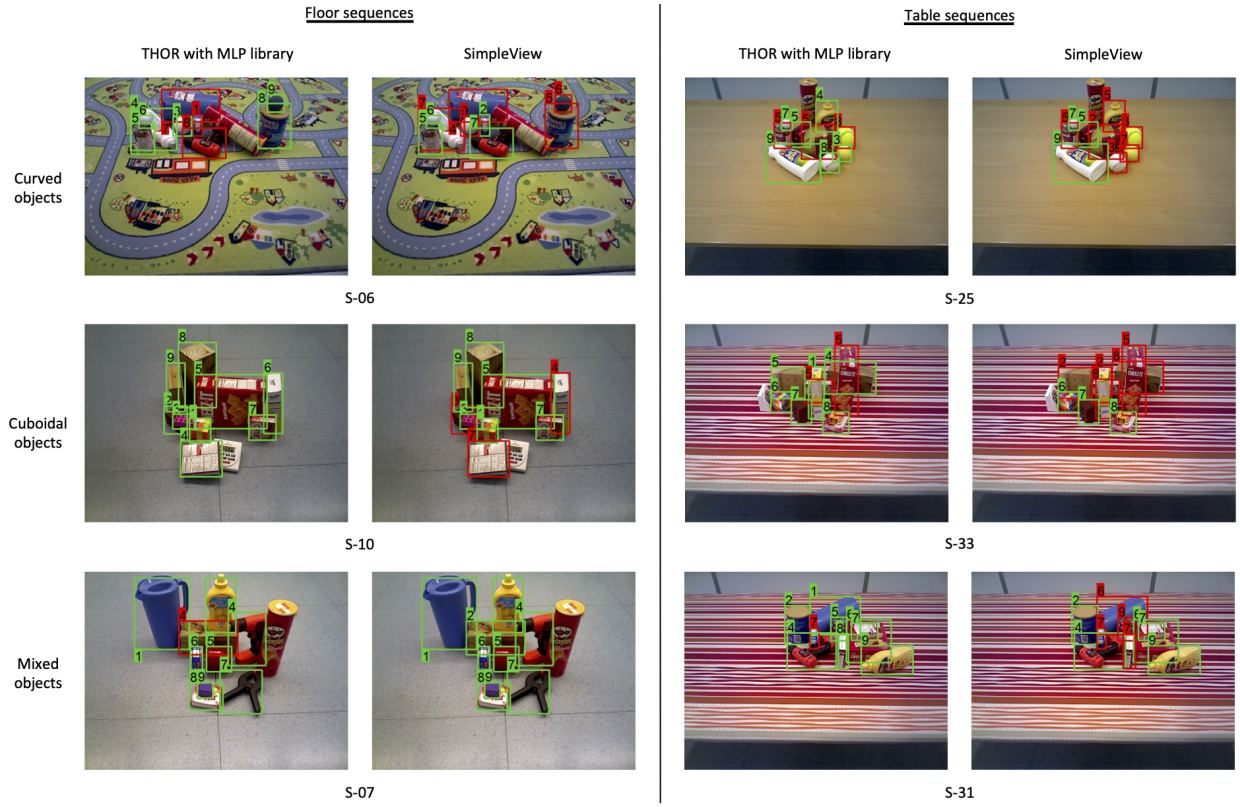


Figure 5.6: Sample results from the OCID dataset sequences recorded using the lower camera (green and red boxes represent correct and incorrect recognition, respectively). The first two columns show results (obtained using THOR and SimpleView, respectively) from sequences where objects are placed on the floor. Similarly, the last two columns show results from sequences where objects are placed on a table. The first, second, and third rows show results from sequences with curved, cuboidal, and mixed objects.

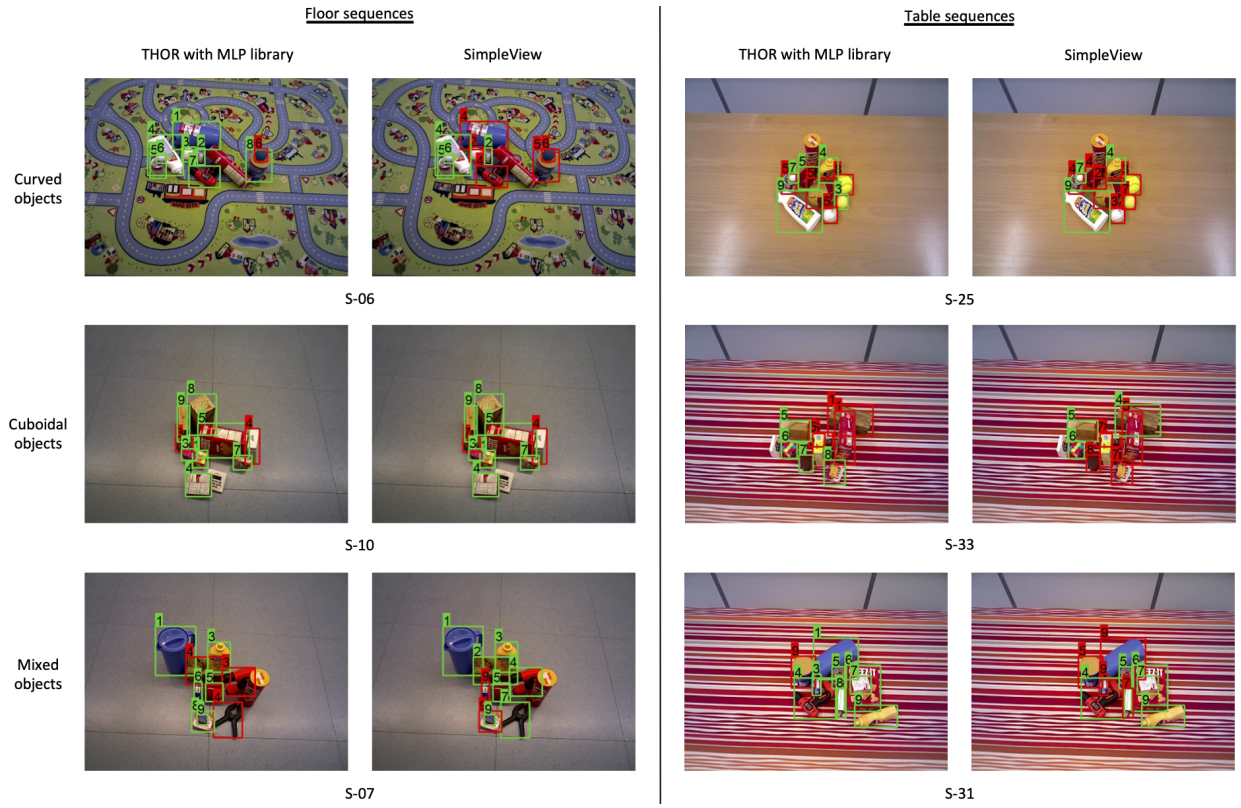


Figure 5.7: Sample results from the OCID dataset sequences recording using the upper camera. The first two columns show results (obtained using THOR and SimpleView, respectively) from sequences where objects are placed on the floor. Similarly, the last two columns show results from sequences where objects are placed on a table. The first, second, and third rows show results from sequences with curved, cuboidal, and mixed objects.

Table 5.6: Comparison of mean recognition accuracy over object classes belonging to different types (in %) on the UW-IS Occluded dataset. THOR's performance is compared with 2D shape-based representation, Sparse PI, and end-to-end models, DGCNN and SimpleView.

Environment	Object type	Sparse PI	THOR				DGCNN	SimpleView
			SVM lib.	SVM lib. w/o heuristics	MLP lib.	MLP lib. w/o heuristics		
Warehouse	Kitchen	10.07 ± 1.99	49.26 ± 0.06	48.19 ± 0.09	**52.33 ± 0.59**	**51.32 ± 0.51**	26.89 ± 1.21	28.52 ± 1.09
	Tools	1.53 ± 0.32	40.71 ± 0.09	38.22 ± 0.05	**46.89 ± 0.48**	**47.00 ± 0.52**	8.83 ± 0.71	26.04 ± 0.42
	Food	21.24 ± 1.22	41.11 ± 0.09	38.80 ± 0.19	**43.30 ± 1.48**	38.94 ± 1.08	0.53 ± 0.08	11.49 ± 1.00
Lounge	Kitchen	5.37 ± 0.88	65.63 ± 0.11	62.13 ± 0.06	**69.98 ± 0.43**	65.45 ± 0.38	30.89 ± 1.26	36.72 ± 1.66
	Tools	2.95 ± 0.28	41.38 ± 0.05	39.89 ± 0.05	**45.96 ± 0.62**	**44.43 ± 0.82**	10.32 ± 1.00	29.60 ± 0.93
	Food	12.40 ± 1.36	44.28 ± 0.06	40.80 ± 0.27	**45.93 ± 1.13**	41.49 ± 0.93	1.29 ± 0.32	14.43 ± 1.24
All environments & objectsn		7.02 ± 0.25	48.18 ± 0.04	45.63 ± 0.06	**52.22 ± 0.33**	49.76 ± 0.35	14.58 ± 0.49	26.43 ± 0.81

degrees of occlusion and different lighting conditions are considered in Tables 5.7 and 5.8, respectively. Overall, THOR outperforms all the other methods by a considerable margin under all the environmental conditions. When area and curvature flow-based heuristics for model sets selection are not used, i.e., when all the three sets M_f, M_s, and M_t are used at the prediction stage, THOR witness a small drop in performance but continues to outperform all the other methods by a large margin.

In Table 5.6, we observe that the performance of THOR, which is trained entirely on point clouds from synthetic depth images, is slightly better in the lounge environment as compared to the warehouse, most probably as the depth data is more accurate for the lounge. The kitchen objects are found to be easiest to recognize in both the environments for all the methods. We believe this is because the kitchen objects (e.g., plate, pitcher, and bleach cleaner) are much larger in size than the associated depth-sensing inaccuracies.

Further, Table 5.7 shows that the performance of THOR is best when none of the objects are occluded. However, the performance in the cases where objects are placed such that some occlusion occurs and when the objects are clustered together resulting in higher occlusion is only slightly lower than that for the no occlusion case. This observation demonstrates the

Table 5.7: Comparison of mean recognition accuracy over all the object classes (in %) on the UW-IS Occluded dataset under varying degrees of occlusion. THOR's performance is compared with 2D shape-based representation, Sparse PI, and end-to-end models, DGCNN and SimpleView.

Environment	Occlusion	Sparse PI	THOR				DGCNN	SimpleView
			SVM lib.	SVM lib. w/o heuristics	MLP lib.	MLP lib. w/o heuristics		
Warehouse	None	9.50 ± 0.50	46.88 ± 0.05	44.21 ± 0.14	**51.62 ± 0.53**	**50.76 ± 0.58**	13.38 ± 0.43	26.35 ± 0.80
	Low	9.31 ± 0.93	44.70 ± 0.08	42.55 ± 0.04	**48.07 ± 0.28**	45.67 ± 0.42	12.47 ± 0.47	21.95 ± 0.80
	High	9.10 ± 0.20	40.30 ± 0.03	39.56 ± 0.06	**44.26 ± 0.25**	43.49 ± 0.30	13.62 ± 0.51	21.89 ± 0.50
Lounge	None	8.56 ± 0.24	52.19 ± 0.09	48.38 ± 0.12	**56.72 ± 0.60**	53.51 ± 0.55	16.91 ± 0.59	31.43 ± 1.06
	Low	6.45 ± 0.55	53.05 ± 0.09	51.54 ± 0.08	**54.45 ± 0.24**	51.90 ± 0.38	15.49 ± 0.77	28.86 ± 1.09
	High	2.63 ± 0.30	45.85 ± 0.05	43.04 ± 0.04	**51.88 ± 0.46**	47.79 ± 0.37	14.38 ± 0.44	25.13 ± 0.85

Table 5.8: Comparison of mean recognition accuracy over all the object classes (in %) on the UW-IS Occluded dataset under varying lighting conditions. THOR's performance is compared with 2D shape-based representation, Sparse PI, and end-to-end models, DGCNN and SimpleView.

Environment	Lighting	Sparse PI	THOR				DGCNN	SimpleView
			SVM lib.	SVM lib. w/o heuristics	MLP lib.	MLP lib. w/o heuristics		
Warehouse	Bright	10.51 ± 0.81	44.20 ± 0.06	42.37 ± 0.09	**47.52 ± 0.36**	46.59 ± 0.49	12.87 ± 0.36	23.86 ± 0.60
	Dim	7.72 ± 0.36	43.29 ± 0.05	41.25 ± 0.07	**48.11 ± 0.39**	46.43 ± 0.40	13.54 ± 0.55	22.62 ± 0.69
Lounge	Bright	5.08 ± 0.35	46.48 ± 0.06	45.26 ± 0.09	**50.76 ± 0.48**	47.08 ± 0.48	13.60 ± 0.56	27.51 ± 0.93
	Dim	7.04 ± 0.29	54.34 ± 0.05	50.60 ± 0.06	**57.64 ± 0.39**	54.79 ± 0.41	17.18 ± 0.57	29.23 ± 1.06

robustness of THOR to the partial occlusion of objects. Table 5.8 shows performance on the different lighting conditions considered in the UW-IS Occluded dataset. As the quality of the depth sensing depends on the lighting conditions, the performance of all the methods shows some dependence on it. However, the effect of the lighting conditions on the performance is relatively minor, and THOR outperforms all the other methods substantially, irrespective of the lighting.

On the other hand, we observe that Sparse PI features have the least overall performance. This poor performance can be attributed to Sparse PI features being 2D shape features designed to recognize unoccluded objects under limited variations in object poses [151]. Even though DGCNN and SimpleView capture 3D shape information, THOR outperforms them in recognizing both the occluded and unoccluded objects. The poor performance of DGCNN and SimpleView shows that they are not robust to point cloud corruptions resulting from occlusion and sensor-related noise, as reported in [139]. Fig. 5.8 shows sample results for our method (with the MLP library) and SimpleView in both the environments for all the three levels of separation among the objects.

Overall, the analysis on the UW-IS Occluded dataset shows that THOR is better suited as an object recognition method for low-cost robots that would encounter different environmental conditions and degrees of occlusion during everyday use.

Study on Parameter Choices for TOPS:

As mentioned in Section 5.4.1, we set the scale factor $\sigma_s = 2.5$ and set $\sigma_1 = 0.1$, $\sigma_2 = 2.5 \times 10^{-2}$, and $\alpha = \frac{\pi}{4}$ to compute suitable TOPS descriptors. The scale factor is chosen based on the depth scale of the RGB-D camera and does not directly impact the computation of the TOPS descriptor. However, the values for σ_1 and σ_2 depend on the scale of the point cloud as they represent the thickness of the slices (along the z-axis and x-axis, respectively) obtained during descriptor computation. The parameter α determines the orientation of the point cloud before slicing, affecting the number of slices obtained during descriptor computation.

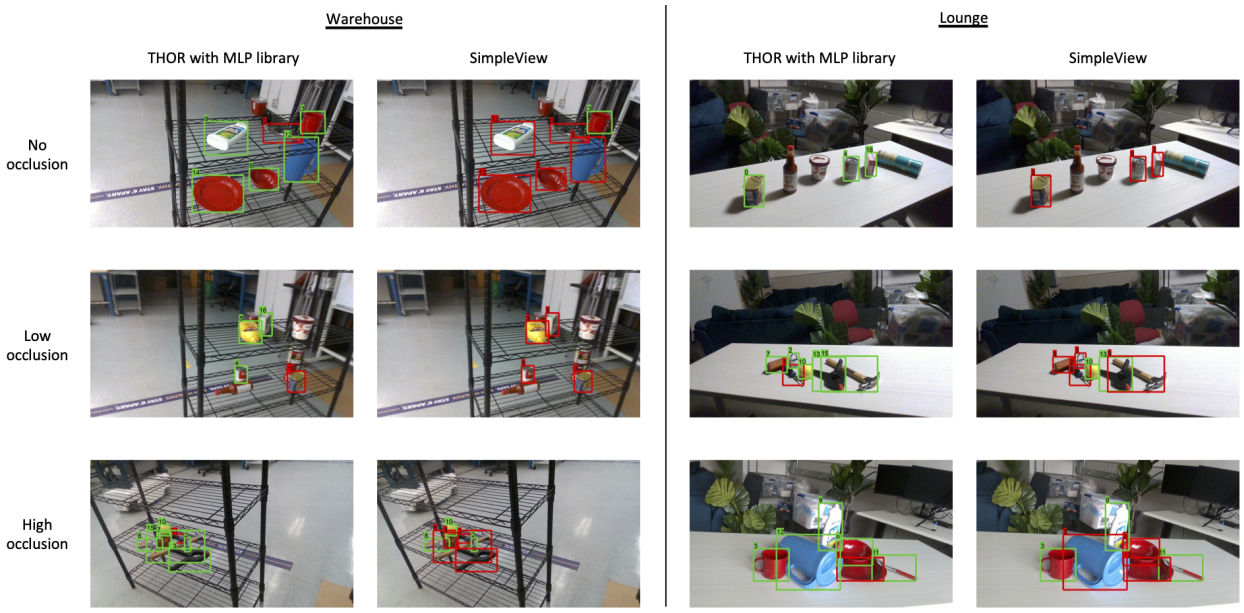


Figure 5.8: Sample results from the UW-IS Occluded dataset. The first two columns show the warehouse environment results (obtained using THOR and SimpleView, respectively). Similarly, the last two columns show results from the lounge. The first, second, and third rows show results from scenes with three different levels of separations between objects. Note that THOR outperforms SimpleView in all the scenarios.

Table 5.9: Comparison of mean recognition accuracy over all the object classes (in %) on the entire UW-IS Occluded dataset under varying parameter choices for TOPS descriptor computation.

	$\sigma_1 = 0.05$ $\sigma_2 = 1.25 \times 10^{-2}$	$\sigma_1 = 0.1$ $\sigma_2 = 2.50 \times 10^{-2}$	$\sigma_1 = 0.2$ $\sigma_2 = 5.00 \times 10^{-2}$
$\alpha = 0$	41.74 ± 0.31	46.75 ± 0.24	47.96 ± 0.60
$\alpha = \pi/4$	39.73 ± 0.19	**52.22 ± 0.33**	46.75 ± 0.90
$\alpha = \pi/2$	33.84 ± 0.09	41.03 ± 0.73	39.72 ± 0.97

Once σ_s is chosen, values for σ_1, σ_2, and α are empirically determined such that every object has a reasonable number of slices and the PDs corresponding to the slices have sufficient shape information. Intuitively, a larger value of α leads to more slices when σ_1 and σ_2 are unchanged, but the corresponding PDs capture little shape information. On the other hand, a smaller value of α leads to more informative PDs but very few slices. For a fixed value of α, a higher value of σ_1 and σ_2 enables capturing shape information at a higher granularity while increasing the dimensionality of the TOPS descriptor and the size of the classifier library. On the other hand, a smaller value of σ_1 and σ_2 leads to a relatively low dimensional descriptor (and smaller classifier library), which captures shape information at a lower granularity.

We report the performance of THOR (with the MLP library) on the UW-IS Occluded dataset for different values of σ_1, σ_2, and α in Table 5.9. Note that we vary σ_1 and σ_2 by the same factor to ensure the change in granularity (resulting from changing in σ_1 and σ_2) is the same along both z and x axes. We observe that $\sigma_1 = 0.1$, $\sigma_2 = 2.5 \times 10^{-2}$, and $\alpha = \frac{\pi}{4}$ give the best results when $\sigma_s = 2.5$.

5.4.4 Robot Implementation

We also implement THOR on a LoCoBot platform built on a Yujin Robot Kobuki Base (YMR-K01-W1) powered by an Intel NUC NUC7i5BNH Mini PC. We mount an Intel RealSense D435 camera on top of the LoCoBot and control the robot using the PyRobot interface [124]. THOR is run on an on-board NVIDIA Jetson AGX Xavier processor, equipped with a 512-core Volta GPU with Tensor Cores and an 8-core ARM v8.2 64-bit CPU. Fig. 5.9 shows a screenshot of the platform and sample recognition results.

THOR (with the SVM library) runs at an average rate of $0.82s$ per frame in a scene with six objects on this platform. This runtime includes the time for instance segmentation that is performed using a TensorRT-optimized [125] depth seeding network (along with the initial mask processor module) from [197]. Recognition prediction for every object point cloud in the scene is then obtained simultaneously using multiprocessing in Python. Note that in this case, we do not use the area and curvature flow-based heuristics for the model sets.

5.5 Discussion

The following subsections discuss several aspects of THOR's performance.

5.5.1 Depth Information: Quality and Significance

We note that the point clouds in the OCID dataset are obtained from temporally averaged depth images. Therefore, such point clouds have fewer inaccuracies that might arise from depth sensing-related issues. On the other hand, the new UW-IS Occluded dataset consists of depth images directly obtained from commodity hardware used in robotic systems. As a result, the dataset more closely represents a real-world setting likely to be encountered by low-cost indoor mobile robots. Consequently, the overall performance of all the methods is lower on this more challenging dataset. Nevertheless, THOR outperforms both the methods by a considerable margin under all environmental conditions and degrees of clutter, demonstrating promising robustness to imprecise depth images.

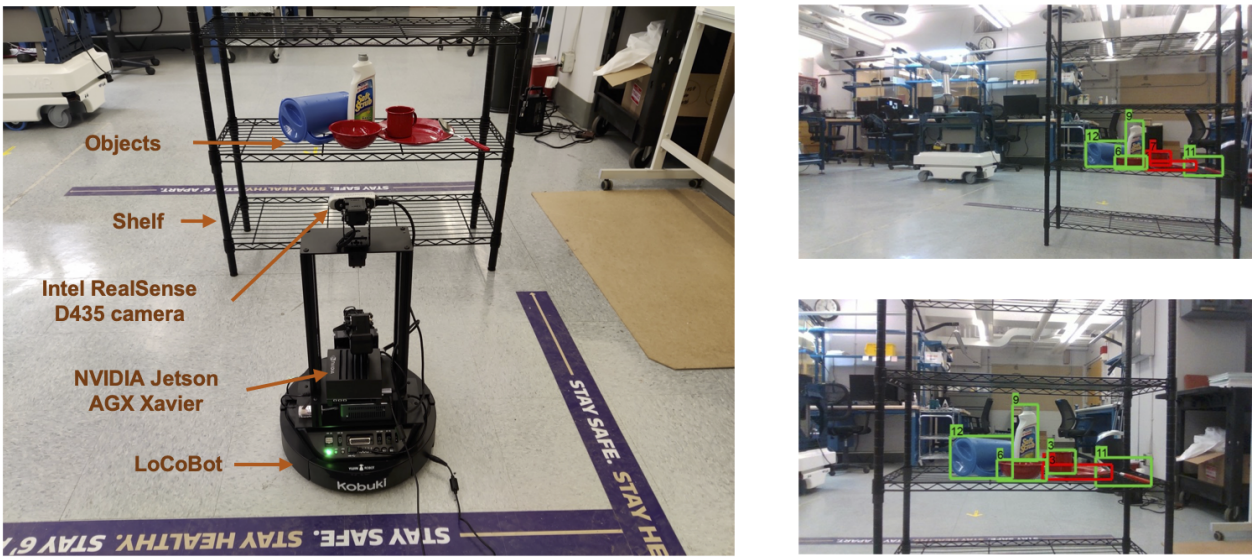


Figure 5.9: Screenshot of the LoCoBot operating in a mock warehouse setup (left) and the sample recognition results (right) obtained using THOR (with the SVM library) as the LoCoBot moves around the warehouse setup. The results demonstrate THOR's ability to recognize objects in unknown environments despite inaccuracies in depth sensing.

Additionally, we observe from Tables 5.6, 5.7, and 5.8 that the 2D shape-based Sparse PI features have the least overall performance on the UW-IS Occluded dataset. In general, Sparse PI features perform better in the absence of occlusion. This trend is expected because the Sparse PI features are designed to recognize unoccluded objects. However, the performance is still poor in the absence of occlusion, which can be attributed to the fact that they are 2D features designed for operation under limited variations in object poses. In contrast, the training set, in this case, considers all the possible object poses. This inferior performance of 2D shape features highlights the need for 3D shape features and motivates the use of depth images, despite the associated sensing inaccuracies.

5.5.2 Sim2Real Gap

In this work, we train THOR and all the other methods using synthetic depth images of unoccluded objects and test them on real-world depth images of cluttered scenes. In Table 5.7, we observe that THOR outperforms DGCNN and SimpleView in the case of occluded and unoccluded objects. This observation indicates that THOR is better at accounting for the sim2real gap in addition to being more robust to partial occlusions. We attribute this observation to the fact that the PIs in the TOPS descriptor are stable with respect to minor perturbations in the filtration [4], providing some robustness to the noise in the depth images. On the other hand, end-to-end models like DGCNN and SimpleView are known to be susceptible to point cloud corruptions resulting from imprecision depth images, self-occlusion, and partial occlusions [139]. Therefore, the use of topological features is a promising way to achieve object recognition in unknown environments without extensive real-world training data. On a related note, designing topology-aware adaptation modules for end-to-end models (like DGCNN and SimpleView) to address the sim2real gap could also be an interesting future work direction.

5.5.3 THOR: Failure Modes

Instance Segmentation Errors:

In this work, we focus on object recognition and assume that the instance segmentation map of a scene is available. However, it is important to note the impact of segmentation errors on THOR. Common instance segmentation errors include under-segmentation, over-segmentation, misaligned object boundaries, false positive segmentations, and the scenarios where the mask of an object is split due to an occluding object [197]. We believe THOR is relatively robust to over-segmentation errors or errors where the object masks are split into segments due to occlusion, since the idea of object unity is built into it. THOR is more likely to recognize the segments correctly as compared to the other methods. However, a post-processing step similar to non-maximum suppression would be required to identify if the segments belong to a single object. Further, the test-time outlier removal from point clouds provides a certain degree of robustness to errors arising from misaligned object boundaries. However, under-segmentation and false positive segmentations are more challenging. Modifying THOR to use appearance information and provide a 'none-of-the-above' decision is a potential way to address such challenges.

Slicing-related Errors:

Tables 5.2 and 5.3 show that THOR outperforms SimpleView in most of the sequences. However, in certain sequences, SimpleView's performance is slightly higher than our method. We believe this discrepancy can be attributed to distortions in the point clouds of certain objects that affect our slicing-based method more than SimpleView. For instance, the wooden block in S-07 from Fig. 5.6 is occluded such that the slices of the resulting point cloud closely resemble those of a power drill.

Specific Occlusion Scenarios:

We specify in Section 5.2.4 that our heuristic-based approach for model sets selection largely holds when the object is not heavily occluded (see Fig. 5.4). In cases of heavy occlusion, the area and curvature flow-based heuristics may lead to incorrect model sets selection. At the same time, it is important to note that when heuristics are not used, the drop in THOR's performance is relatively small and continues to outperform the other methods (see Section 5.4.3). Additionally, we mention in Section 5.2.4 that if an object is occluded, the aligned point cloud is reoriented to ensure that the first slice on the occluded end of the object (i.e., the end where one or more slices may be missing) is not the first slice during subsequent TOPS descriptor computation. Therefore, scenarios where an object is occluded such that there are missing slices on both the ends pose difficulties. Such occlusion scenarios are relatively infrequent, and we believe additional information, such as the appearance of the object or depth image from a different viewpoint, would be necessary to perform recognition.

5.5.4 THOR: Classifier Choice

In general, THOR works with libraries generated using any classifier of choice. In this article, we report and analyze the performance of THOR using two different classifiers, SVM and MLP. Results reported in Sections 5.4.2 and 5.4.3 show that, overall, THOR outperforms the other methods regardless of the choice of the classifier. This observation, in addition to our ablation studies, indicates that the discriminative power of the TOPS descriptor is the primary reason for THOR's markedly better performance than DGCNN and SimpleView. However, we note that THOR with the MLP library maintains overall performance in case of challenging object views, as opposed to THOR with the SVM library, which undergoes a slight drop in performance. Furthermore, THOR with the MLP library performs better than THOR with an SVM library on the more challenging UW-IS Occluded dataset, where point clouds are generated from raw depth images (as opposed to the temporally smoothed point clouds of the OCID dataset).

5.5.5 Real Time Performance

With a prediction speed of 0.82s per frame, THOR operates in real time. Note that the current work is a feedforward approach, i.e., a single image frame is used to make predictions. THOR can be made more efficient in scenarios when the object locations do not change considerably between consecutive frames by avoiding repeated computation. Such modifications incorporating temporal information are also beneficial in environments where the objects change location due to external intervention(s).

5.6 Summary

This work presents a new topological descriptor, TOPS, and an accompanying human-inspired recognition framework, THOR, for recognizing occluded objects in unseen indoor environments. We construct slicing-style filtrations of simplicial complexes from the object's point cloud to obtain the descriptor using persistent homology. Our approach ensures similarities between the descriptors of the occluded and the corresponding unoccluded objects. We use this similarity in THOR to recognize the occluded objects using a human reasoning mechanism, object unity, thereby eliminating the need to collect large amounts of representative training data.

We report performance comparisons on two datasets: OCID and UW-IS Occluded. On the benchmark OCID dataset, comparisons with two state-of-the-art point cloud classification methods, DGCNN and SimpleView, show that THOR achieves the best overall performance when increasingly cluttered scenes are viewed from different camera positions. Further, our ablation investigations show that the TOPS descriptor achieves considerably higher recognition accuracy than the widely used descriptors CVFH and ESF. On the new UW-IS Occluded dataset, which reflects real-world scenarios with different environmental conditions and degrees of object occlusion, THOR achieves substantially higher recognition accuracy than the state-of-the-art methods. Therefore, THOR is a promising step toward robust recognition in low-cost robots, meant for everyday use in indoor settings.

Chapter 6

HUMAN-INSPIRED 3D SHAPE AND COLOR-BASED RECOGNITION IN UNSEEN CLUTTERED ENVIRONMENTS

A compact version of this work will be presented as a Late-Breaking Results poster at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023.

6.1 Motivation

The previous chapter presents a human-inspired framework THOR for 3D shape-based object recognition in unseen and cluttered real-world environments. Unlike recognition using widely used handcrafted and learning-based shape representations, THOR shows promising robustness to point cloud corruptions resulting from imprecise depth images, self-occlusion, and partial occlusions. However, recognition using shape information alone is challenging. Additional information cues, such as the color of an object, help achieve improved recognition [87,98], and are particularly crucial to distinguish between objects with similar geometry (e.g., a softball and a baseball). As discussed in Section 2.2.2, several approaches have been proposed for object recognition using RGB-D images, ranging from handcrafted descriptor-based methods to learning-based methods using multimodal convolutional neural networks and transformers. Due to the lack of large RGB-D datasets like the RGB-only ImageNet, several learning-based approaches employ depth encoding and transfer learning techniques to models trained on RGB images for RGB-D recognition [180]. Alternatively, we investigate using synthetic RGB-D images for training and extending THOR to use both 3D shape and color information for object recognition.

One of the key challenges associated with recognizing objects based on color is that the observed chromaticity of objects varies with the lighting conditions [41]. The image signal

processors on current RGB-D cameras run automatic white balancing algorithms to remove undesirable color casts caused by environmental lighting. However, such algorithms require further work to completely tune out the effect of illumination conditions in typical real-world scenes [5]. The problem of variation in observed chromaticity is further exacerbated in our case due to the sim2real gap between the training and test data. Therefore, we take inspiration from the human perception of color to account for this variation.

Specifically, regions of the color space (known as the MacAdam ellipses) have been identified where each region contains colors indistinguishable to the human eye [116]. Analogously, we use the Mapper algorithm to identify color regions (i.e., clusters of similar colors) in the sRGB color space and capture their connectivity in a color network. We use the color network to compute color embeddings for object point cloud slices and interleave them with the TOPS descriptor to obtain the TOPS2 descriptor. Similar to its predecessor, the TOPS2 descriptor embodies the principle of object unity and ensures similarities between the descriptors of the occluded and the corresponding unoccluded objects. We also propose the THOR2 framework, which selectively uses the TOPS and TOPS2 descriptors for recognition. We show that THOR2, trained using synthetic data, outperforms a state-of-the-art transformer network adapted for RGB-D object recognition in clutter and achieves substantially higher recognition accuracy in all the environmental conditions of the UW-IS Occluded dataset.

The rest of the chapter is organized as follows. Section 6.2 provides details of the color network generation and the THOR2 framework. Section 6.3 describes the experimental details and the results achieved. Several aspects of the THOR2's performance and failure modes are discussed in Section 6.4 followed by concluding remarks in Section 6.5.

6.2 Method: THOR2

Given an RGB-D image of a cluttered scene, we aim to recognize all the objects in the scene. We continue to follow the human object recognition framework discussed in Section 1.1 to achieve it. Similar to our approach in Chapter 5, we assume that instance segmentation maps are available using methods such as those in [114,197]. Subsequently, we obtain colored

point clouds for every object in the scene and perform view normalization, as described in Section 5.2.1. We then capture their shape and color information by computing the TOPS and TOPS2 descriptors. The TOPS2 descriptor incorporates color information through color embeddings based on the similarity and connectivity among different colors in a color network, which is obtained using the Mapper algorithm. Lastly, we perform recognition using two classifiers (one for each descriptor) trained on synthetic RGB-D images. Fig 6.1 depicts the overall framework, and the following subsections describe the associated steps in detail.

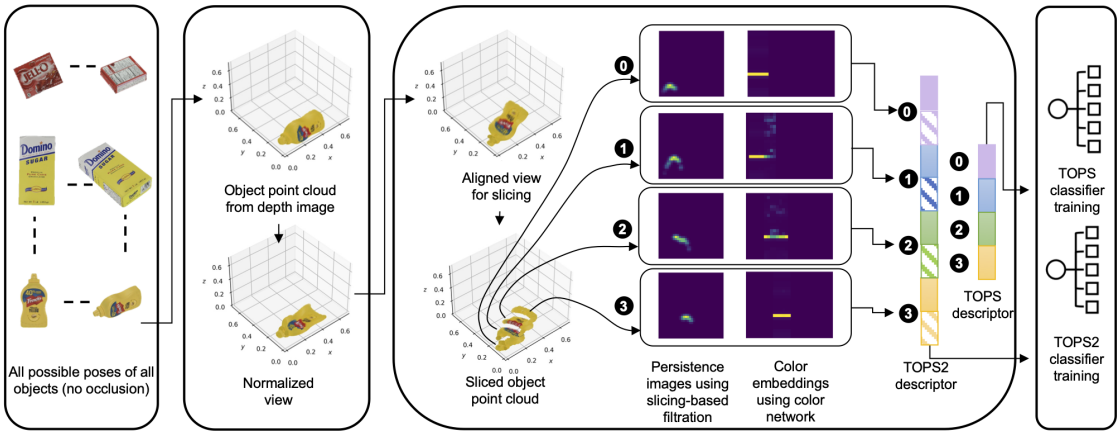
6.2.1 Color Network Generation

We consider the colors represented by the standard RGB (sRGB) color space, where values for each of the three color channels range from 0 to 255. Let X_{rgb} denote the set of these colors. The sRGB color space is not perceptually uniform; the Euclidean distance between two colors represented using this color space is not proportional to the difference perceived by humans. Therefore, we convert all the elements in X_{rgb} to the CIELAB color space (also known as the $L^*a^*b^*$ color space) to obtain the set X_{lab}. We note the CIELAB color space is also not truly perceptually uniform. However, it was purposefully designed such that the Euclidean distance between points is proportional to the perceived color difference. Therefore, it is the preferred color space for algorithmically distinguishing between objects by their color [41].

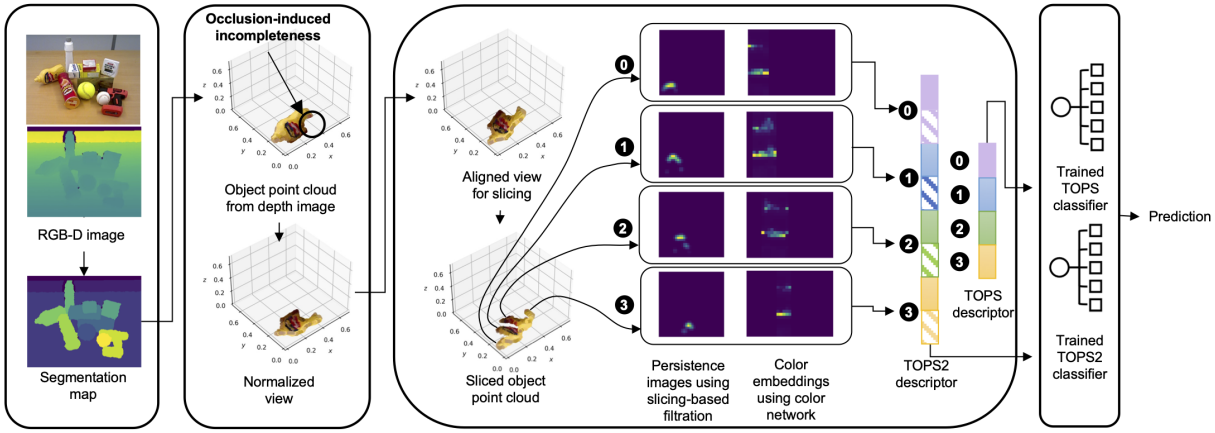
We then apply the Mapper algorithm to X_{lab} to obtain a color network. Specifically, let k_{L^*}, k_{a^*}, and k_{b^*} represent the L^*, a^*, and b^* components of a color k in X_{lab}. We use a chroma and hue-based lens function, f_l, to transform the three-dimensional data in X_{lab} to a two-dimensional space. We define f_l as follows.

$$f_l(k) = \left(\sqrt{k_{a^*}^2 + k_{b^*}^2}, \xi + \arctan\left(\frac{k_{b^*}}{k_{a^*}}\right) \right), \tag{6.1}$$

where ξ is a constant offset selected based on the cover. We adopt the standard choice [35]



(a) THOR2 training stage showing a visualization of the TOPS and TOPS2 descriptors computed from the synthetic RGB-D image corresponding to a sample object pose (of an unoccluded mustard bottle) chosen from the set of object poses considered during library training.



(b) THOR2 testing stage, aligned with the stages of human object recognition shown in Fig. 1.1, showing a visualization of the TOPS and TOPS2 descriptors of an occluded object in a scene computed from the corresponding real RGB-D image.

Figure 6.1: Proposed framework, THOR2, for 3D shape and color-based recognition using object unity, facilitated by the similarity in the TOPS and TOPS2 descriptors of unoccluded and occluded objects.

of building a cubical cover $\mathcal{U}$ by considering a set of regularly spaced intervals of equal length covering the set $f_l(X_{lab})$. Let r_1 and r_2 (where $r_1, r_2 > 0$) denote the lengths of the intervals (also known as the *resolution* of the cover) along the two dimensions of $f_l(X_{lab})$, respectively. Let g_1 and g_2 denote the respective percentages of overlap (also known as the *gain* of the cover) between two consecutive intervals. Subsequently, the Mapper algorithm applies a clustering algorithm to $f_l^{-1}(U)$ for each $U \in \mathcal{U}$ to obtain the refined pullback cover $\mathcal{R}$ of X_{lab}. For clustering, we use the HyAB distance metric [3] for computing the distance between two colors. Unlike other color difference formulae (e.g., CIEDE2000 [115]), the HyAB distance is applicable to a large range of color differences required for practical applications [3]. The HyAB distance between two colors m and n in the CIELAB space is defined as

$$\text{HyAB}(m, n) = |m_{L^*} - n_{L^*}| + \sqrt{(m_{a^*} - n_{a^*})^2 + (m_{b^*} - n_{b^*})^2}. \tag{6.2}$$

Next, the nerve of the refined pullback cover is obtained, as described in Section 3.2, to obtain the output of the Mapper algorithm. Note that the cyclic nature of the chroma-related dimension of $f_l(X_{lab})$ cannot be captured through the cubical cover described above. Therefore, we augment the output of the Mapper algorithm to add edges connecting the nodes corresponding to the first and last intervals along the dimension. Subsequently, we eliminate redundant nodes, as described in 6.3.2, to obtain the final color network shown in Fig. 6.2. In this network, the nodes represent useful color regions identified by the Mapper algorithm, and the edges represent overlap between the corresponding color regions.

Let $G = (V, E)$ be the color network, representing a non-empty set of vertices or nodes V and a set of edges E. Let n_c represent the numbers of vertices in G, i.e., the number of color regions. We define a similarity matrix, Δ, of size $n_c \times n_c$ to capture the similarity and connectivity between the different color regions in G. We define Δ as follows

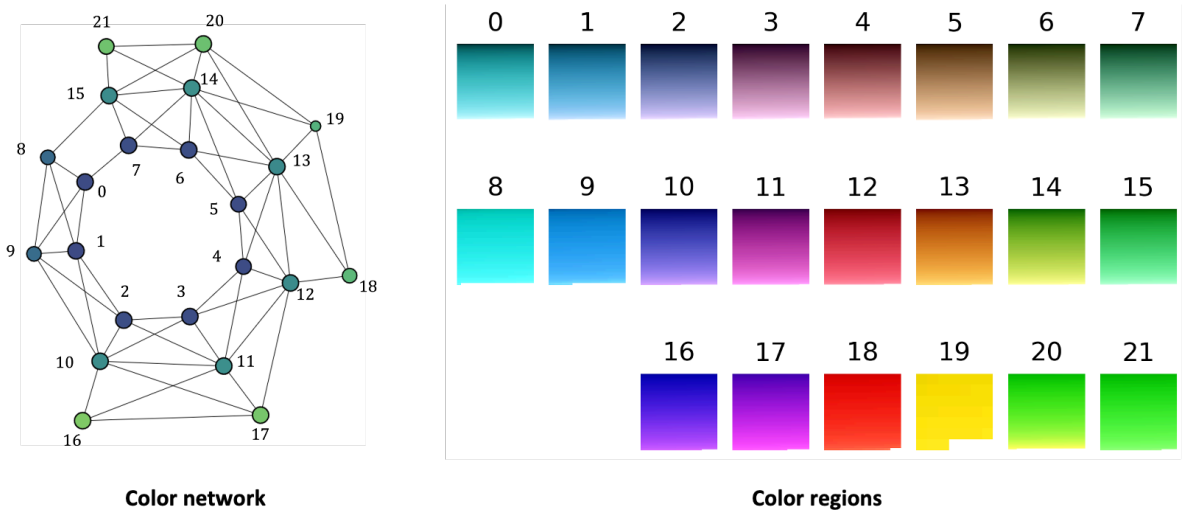


Figure 6.2: The color network that captures the connectivity between the different color regions identified using the Mapper algorithm.

$$\Delta = \begin{bmatrix} \delta_{11} & \delta_{12} & \dots & \delta_{1n_c} \\ \delta_{21} & \delta_{22} & \dots & \delta_{2n_c} \\ \vdots & \vdots & \vdots & \vdots \\ \delta_{n_c1} & \delta_{n_c2} & \dots & \delta_{n_cn_c} \end{bmatrix} \tag{6.3}$$

where $\delta_{i'j'}$ represents the similarity between the i'-th and j'-th nodes. Since every edge in E does not represent the same perceptual difference between the color regions, first, we assign a weight to each edge. Let $\eta_{i'j'}$ be an edge in E that connects the i'-th and j'-th nodes. To assign the weight, first, we compute the mean color[1] of the i'-th and j'-th colors nodes. The edge $\eta_{i'j'}$ is then assigned a weight equal to the HyAB distance between the mean colors of the i'-th and j'-th nodes. We then set $\delta_{i'j'} = \frac{1}{1+l_{i'j'}}$, where $l_{i'j'}$ is the weight of minimum weight path connecting the i'-th and j'-th nodes. This similarity matrix is pre-computed

[1]To compute the mean, we convert all the colors in that node to the sRGB space, calculate the arithmetic mean of the intensities in each channel [87], and convert the resulting mean color back to the CIELAB space

and used for TOPS2 descriptor computation, as described in the following subsection.

6.2.2 TOPS2 Descriptor Computation

Consider a colored object point cloud $\mathcal{P}$ in $\mathbb{R}^3$. First, we perform view normalization, as described in Section 5.2.1, to obtain the view-normalized point cloud, $\tilde{\mathcal{P}}$. Similar to the computation of the TOPS descriptor (refer to Section 5.2.2), we rotate $\tilde{\mathcal{P}}$ by an angle α about the y-axis to obtain a suitably aligned point cloud $\hat{\mathcal{P}}$. Then, we slice $\hat{\mathcal{P}}$ along the z-axis to get slices $\mathcal{S}^i$, where $i \in \mathbb{Z} \cap [0, \frac{h}{\sigma_1}]$. Here, h is the dimension of the axis-aligned bounding box of $\hat{\mathcal{P}}$ along the z-axis, and σ_1 is the thickness of the slices. Let $p = (p_x, p_y, p_z)$ be a point in $\hat{\mathcal{P}}$. The slices $\mathcal{S}^i$ are then obtained as follows.

$$\mathcal{S}^i := \left\{ p \in \hat{\mathcal{P}} \mid i\sigma_1 \leq p_z < (i+1)\sigma_1 \right\}. \tag{6.4}$$

Let $s = (s_x, s_y, s_z)$ represent a point in $\mathcal{S}^i$. For every slice $\mathcal{S}^i$, we modify the z-coordinates $\forall s \in \mathcal{S}^i$ to s'_z, where $s'_z = i\sigma_1$. Color embeddings for every slice are then obtained as follows

Consider a slice $\mathcal{S}^i$. We perform further slicing of $\mathcal{S}^i$ along the x-axis to obtain *strips* Ω^j, where $j \in \mathbb{Z} \cap [0, \frac{w}{\sigma_2}]$. Here, w is the dimension of the axis-aligned bounding box of the slice along the x-axis and σ_2 represents the 'thickness' of a strip. For every strip Ω^j, we obtain corresponding color vectors $\Phi^j = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_{n_c} \end{bmatrix}^T$ as follows.

$$\phi_\lambda = \sum_{\omega \in \Omega^j} \frac{\mathbb{1}_{X^\lambda_{rgb}}(\omega)}{\sum_{\lambda=1}^{n_c} \mathbb{1}_{X^\lambda_{rgb}}(\omega)}, \tag{6.5}$$

where $\lambda \in \{1, \ldots, n_c\}$ represents the λ-th color region, ω represents the color of a point in Ω^j (in the sRGB color space), $\mathbb{1}$ denotes the indicator function of a set, and X^λ_{rgb} represents the set of colors (in the sRGB color space) belonging to the λ-th color region. Consequently, the color vectors Φ^j approximately represent the color constitution (in terms of the color regions) of the strips Ω^j.

We then stack the color vectors (with appropriate zero padding) to obtain an $n_s^{max} \times n_c$ dimensional color matrix $\mathcal{C}^i$. Let $\mathcal{C}^i = \begin{bmatrix} \mathbf{o} & \cdots & \Phi_1 & \Phi_2 & \cdots & \Phi_{n_s} & \cdots & \mathbf{o} \end{bmatrix}^T$, where n_s is the number of

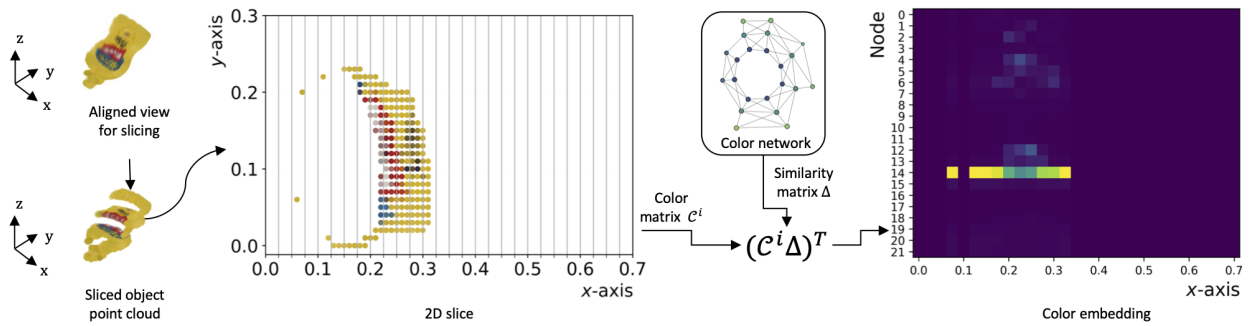


Figure 6.3: Visualization of the computation of the color embedding for obtaining the TOPS2 descriptor. Example of an aligned object point cloud, $\hat{\mathcal{P}}$, the slices $\mathcal{S}^0$ to $\mathcal{S}^4$ obtained from it, and the color embedding for one of its slices, i.e., $\mathcal{S}^1$.

strips in the corresponding slice $\mathcal{S}^i$, n_s^{max} is the maximum number of strips in any given slice, and $\mathbf{O}$ represents a $n_c \times 1$ dimensional zero matrix. Consequently, the color color matrix $\mathcal{C}^i$ approximately represents the color constitution (in terms of the color regions) of the slice $\mathcal{S}^i$ in a spatially-aware manner. Last, we obtain an embedding $\mathcal{E}^i$ corresponding to the color matrix $\mathcal{C}^i$ as follows.

$$\mathcal{E}^i = (\mathcal{C}^i\Delta)^T \tag{6.6}$$

Fig. 6.3 illustrates this embedding generation procedure for a slice of a sample object.

In addition to the color embedding $\mathcal{E}^i$, we also obtain a persistence image (say $\mathcal{I}^i$) for $\mathcal{S}^i$ using the slicing-based descriptor function, as described in Section 5.2.2. We then vectorize and concatenate both $\mathcal{I}^i$ and $\mathcal{E}^i$ to obtain a combined shape and color-based representation for $\mathcal{S}^i$. Such combined representations of all the slices are stacked to obtain the final TOPS2 descriptor.

6.2.3 THOR2: Training and Testing

As in the case of THOR, we consider a training set comprising synthetic RGB-D images corresponding to all the possible views of all the objects. We do not consider any object occlusion scenarios in our training set, owing to the slicing-based design of the TOPS2 descriptor, which captures the principle of object unity. We begin by generating colored object point clouds from the RGB-D images and scale them by a factor of σ_s. Next, we perform view normalization and simultaneously compute the TOPS and TOPS2 descriptors for the point clouds. Unlike THOR, which uses a library of multiple classifiers, THOR2 only uses two classifiers. We train one classifier (say M_1) using the TOPS descriptor and another (say M_2) using the TOPS2 descriptor.

When testing on a real RGB-D image of a cluttered scene, first, we generate the individual colored point clouds of all the objects in the scene using the instance segmentation maps. Similar to the training stage, we scale the point clouds by a factor of σ_s. Consider a scaled object point cloud $\mathcal{P}_t$ obtained from the test image. To recognize $\mathcal{P}_t$, first, we perform view normalization to obtain $\tilde{\mathcal{P}}_t$. Similar to THOR, we then determine if the object corresponding to $\tilde{\mathcal{P}}_t$ is occluded. For this purpose, we obtain the object's contour (in the segmentation map) and use the neighboring pixels' segmentation labels and depth values. For every pixel in the contour, we assume it is part of an occlusion boundary if one or more of its neighboring pixels are labeled as object instances and have a depth value smaller than its own depth value. If the object is occluded, we rotate $\tilde{\mathcal{P}}_t$ by π about the z-axis to ensure that the first slice on the occluded end of the object is not the first slice during subsequent TOPS and TOPS2 descriptor computation. We then compute the TOPS and TOPS2 descriptors corresponding to $\tilde{\mathcal{P}}_t$, and use the corresponding classifier models, M_1 and M_2, respectively, to obtain two predictions. We choose the prediction with the highest probability as our final prediction.

6.3 Experiments

We use synthetic training data and evaluate THOR2 on the real-world OCID and UW-IS Occluded datasets. We report performance comparisons with THOR [150] and the widely-used Vision Transformer (ViT) adapted for RGB-D Object Recognition [180] on both the datasets. In addition, we report comparisons with the case when RGB-D ViT is trained using both synthetic as well as real-world data from the YCB dataset. We also perform ablation experiments to study the role of the different topological representations used in THOR2.

6.3.1 Datasets

Training:

We followed the procedure in Section 5.4.1 for generating synthetic training data corresponding to textured object meshes from [32] using Panda3D [128]. In certain experiments (i.e., Section 6.6), we also use additional real-world data from the YCB dataset [32] for training the RGB-D ViT model. The YCB dataset consists of RGB-D images obtained using a scanning rig [160] that comprises five RGB cameras and five depth sensors placed at different locations. Each object is placed on a turntable that is rotated in three-degree increments, generating 600 RGB-D images per object.

Testing:

Similar to Chapter 5, we use the YCB10 subset of the OCID dataset [164] and the UW-IS Occluded dataset for evaluation purposes[2]. In the case of the OCID dataset, we consider all the three types of sequences of increasingly cluttered scenes: sequences with curved objects, cuboidal objects, and mixed objects. We test recordings of these sequences obtained using

[2]The OCID dataset uses some objects (e.g., soccer ball, power drill) that have a different color from the textured meshes of the YCB dataset objects used for synthetic training data generation. Despite this difference, we consistently test all the methods on all the images of the OCID dataset.

the lower and the upper RGB-D camera. In the case of the UW-IS Occluded dataset, we perform evaluations under all the different environmental conditions captured in the dataset. Specifically, we evaluate the performance of all the methods in the lounge and the warehouse environments of the dataset. For both environments, we consider all three degrees of occlusion and two types of lighting conditions.

6.3.2 Implementation Details

Color Network Generation:

We use the Kepler Mapper library [182, 183] for applying the Mapper algorithm to X_{rgb}. X_{rgb} consists of colors obtained by uniformly sampling each color channel with a step size of five to obtain a total of 52^3 color values. We use the CIE standard illuminant D65 (and its color space chromaticity coordinates corresponding to the standard 2° observer) to convert colors from the sRGB space to the CIELAB color space. Since the Mapper algorithm is an exploratory data analysis tool, a commonly used strategy is to explore a range of associated parameters; the parameters that provide the most informative output (with respect to the user perspective) are selected [35]. We choose $\xi = \frac{\pi}{8}$, $g_1 = 10\%$, and $g_2 = 25\%$. We set r_1 and r_2 to divide the corresponding dimensions into three and eight equally-spaced intervals, respectively. We use the DBSCAN algorithm [52] with HyAB as the distance metric for clustering. During clustering, a point is considered a core point if it has at least six data points in its neighborhood (points that are at most seven units apart are considered neighbors). Once the Mapper output is obtained, we perform post-processing to eliminate redundant nodes. First, we identify node pairs that have more than 95% members in common. For each pair, we then compute the mean colors of the nodes. If the mean colors are neighbors (as defined above), we appropriately merge the nodes to form a single node.

TOPS2 computation:

Following the findings reported in Section 5.4.3, we set the scale factor $\sigma_s = 2.5$ and set $\sigma_1 = 0.1$, $\sigma_2 = 2.5 \times 10^{-2}$, and $\alpha = \frac{\pi}{4}$ to compute suitable TOPS2 descriptors. Consequently, n_s^{max} is 29 and 31 for the OCID and UW-IS Occluded datasets, respectively. We use an approximate computation to generate the PDs and the Persim package in Scikit-TDA Toolbox to generate the corresponding persistence images, as described in Section 5.4.1.

THOR2 Training and Testing:

We use workstations with GeForce GTX 1080 and 1080 Ti GPUs running Ubuntu 18.04 LTS for training (and testing) the classifier models. Separate TOPS-based and TOPS2-based classifiers are trained for every sequence using synthetic RGB-D images in the case of the OCID dataset. In the case of the UW-IS Occluded dataset, we train a single TOPS-based classifier and a single TOPS2-based classifier. For training, first, we augment the training set by mirroring all the view-normalized point clouds across the x and y coordinate axes (in place). We train five-layer fully connected networks, i.e., Multi-layer Perceptrons (MLPs) for THOR2. We use the Adam optimizer to optimize the categorical cross-entropy loss over 100 epochs. The learning rate is set to 10^{-2} for the first 50 epochs and is decreased to 10^{-3} for the subsequent 50 epochs.

For testing on the OCID dataset, we use the temporally smoothed, point-wise labeled point clouds provided for every frame in the sequence. In the case of the UW-IS Occluded dataset, we generate point clouds from depth images using the known camera intrinsic parameters and a depth scale of 0.001m. Since the dataset is recorded using commodity hardware and consists of noisy depth images, we consistently perform pre-processing for every object point cloud, as described in Chapter 5.4.1. Specifically, uniform voxel grid-based downsampling is performed using a voxel size of 0.03 using Open3D [212]. Next, radius outlier removal is performed to remove the points with fewer than 220 neighboring points in a sphere of radius 5×10^{-2} around them. We report test results from five-fold cross-validation for THOR2

(and all the comparison methods).

Comparison methods:

We refer the reader to Section 5.4.1 for details regarding the implementation of THOR. In the case of RGB-D ViT, we use the same synthetic training data used for THOR and THOR2. We obtain surface normal representations for the depth images as in [29]. Following [180], we use the base configuration for ViT [49] initialized using weights from pretraining on ImageNet (provided by PyTorch). We use the mean and standard deviation corresponding to the ImageNet dataset for standardizing the normalized RGB images and surface normal representations of the synthetic dataset. For testing, first, depth interpolation is performed [29], and object crops are obtained using the instance segmentation maps. Subsequently, we replace the background (i.e., background pixels of the RGB image and corresponding values of the surface normal representation of the depth image) to match that of the synthetic images used for training. The object crops are then resized and padded appropriately to achieve the required input size (i.e., 224×224). Note that we perform the same procedure for background replacement, resizing, and padding RGB-D images from the YCB dataset in experiments where they are used for training the RGB-D ViT model.

6.3.3 Results

Comparison with End-to-end Methods:

We compare the performance of THOR2 with THOR (with the MLP library), which, unlike THOR2, only uses 3D shape information. In addition, we also compare THOR2's performance with ViT adapted for RGB-D Object Recognition [180].

Table 6.1 reports the performance of all the three methods on the test sequences of the OCID dataset recorded using the lower camera, and Fig. 6.4 shows a few sample results. THOR2 achieves approximately 8.5% higher recognition accuracy than THOR and approximately 18.9% higher accuracy than RGB-D ViT trained on synthetic data. We note that

incorporating color information in THOR2 is particularly beneficial in the case of curved and cuboidal object sequences where object geometries are similar and less easily distinguishable. For instance, the color information in THOR2 helps distinguish between a softball and a baseball in S-25 (see the first row in Fig. 6.4). In another instance, color helps distinguish between the sugar box and the pudding box in S-23 (see the second row in Fig. 6.4). THOR2 achieves substantially better performance than THOR in such sequences. Performance improvements are also observed for all but two mixed object sequences where object geometries vary considerably.

We observe similar trends in Table 6.2 when the test sequences are recorded using the upper camera. Although THOR2 undergoes a slight drop in performance as compared to the lower camera case, it continues to outperform THOR and RGB-D ViT by 4.94% and 16.42%, respectively. Fig. 6.5 shows sample images recorded using the upper camera for the sequences that are considered in Fig. 6.4. The slight drop in THOR2's performance can be attributed to the fact that recording using the upper camera is more likely to result in challenging object poses. We refer the reader to Section 6.4 for further discussion on this and other factors that affect THOR2's performance.

We also evaluate the performances of THOR, THOR2, and RGB-D ViT under different environmental conditions of the UW-IS Occluded dataset. Tables 6.3, 6.4, and 6.5 show that THOR2 outperforms both the methods regardless of the environmental condition. Fig. 6.6 and Fig. 6.7 show sample results for different types of objects under different degrees of occlusion and lighting conditions in the lounge and warehouse environments, respectively.

In particular, Table 6.3 shows the performance in the warehouse and the lounge while considering variations in the object types. Overall, THOR2 outperforms THOR achieving approximately 10% higher accuracy. THOR2 also achieves approximately 20% higher accuracy than RGB-D ViT trained on synthetic data. The largest performance improvement in THOR2 (over THOR) is observed for the food category objects, some of which are similar in size and geometry but considerably distinct in color. Fig. 6.6 shows one such example where, unlike THOR, THOR2 successfully distinguishes between the potted meat can and

Table 6.1: Comparison of mean recognition accuracies (in %) of THOR, THOR2, and RGB-D ViT on the OCID dataset sequences recorded using the lower camera (best in bold).

Place	Scene type	Seq. ID	THOR	THOR2	RGBD ViT
Table	Curved	S-25	65.31 ± 2.03	**74.90 ± 4.51**	49.10 ± 2.64
		S-26	57.18 ± 2.25	**75.70 ± 2.71**	64.06 ± 2.82
		S-35	49.83 ± 2.07	**64.38 ± 3.83**	34.59 ± 2.95
		S-36	61.35 ± 0.49	63.33 ± 0.98	**67.91 ± 1.95**
	Cuboid	S-23	72.82 ± 0.94	**83.75 ± 2.35**	62.88 ± 5.29
		S-24	57.01 ± 4.28	**70.53 ± 1.60**	58.25 ± 2.49
		S-33	65.90 ± 3.27	**77.54 ± 3.47**	55.38 ± 2.56
		S-34	65.09 ± 3.38	**79.86 ± 1.28**	53.98 ± 3.63
	Mixed	S-21	**68.02 ± 1.06**	**71.49 ± 3.85**	48.27 ± 2.06
		S-22	62.11 ± 3.77	**73.68 ± 2.43**	**80.22 ± 4.20**
		S-31	**74.59 ± 1.34**	**79.95 ± 4.58**	**81.07 ± 3.76**
		S-32	67.16 ± 1.76	**74.25 ± 1.87**	58.93 ± 3.80
Floor	Curved	S-05	**65.75 ± 2.56**	**66.44 ± 1.77**	42.65 ± 4.67
		S-06	79.01 ± 2.72	**91.50 ± 3.01**	32.35 ± 2.75
		S-11	**67.84 ± 2.39**	**69.98 ± 2.36**	54.30 ± 4.89
		S-12	70.79 ± 0.90	**81.30 ± 0.64**	61.57 ± 2.74
	Cuboid	S-03	63.16 ± 1.07	**83.75 ± 2.87**	60.59 ± 3.78
		S-04	55.62 ± 3.35	**71.78 ± 4.48**	62.28 ± 3.23
		S-09	**73.96 ± 3.64**	**70.12 ± 5.69**	58.03 ± 2.87
		S-10	84.32 ± 3.11	**94.32 ± 3.75**	58.58 ± 5.35
	Mixed	S-01	76.78 ± 1.66	**91.83 ± 2.15**	61.88 ± 6.28
		S-02	**73.04 ± 1.26**	64.36 ± 4.56	44.76 ± 2.34
		S-07	**88.53 ± 1.57**	67.76 ± 1.29	71.94 ± 3.17
		S-08	43.43 ± 1.30	**53.26 ± 1.95**	40.38 ± 2.48
All sequences			66.67 ± 0.22	**75.07 ± 0.33**	56.17 ± 0.42

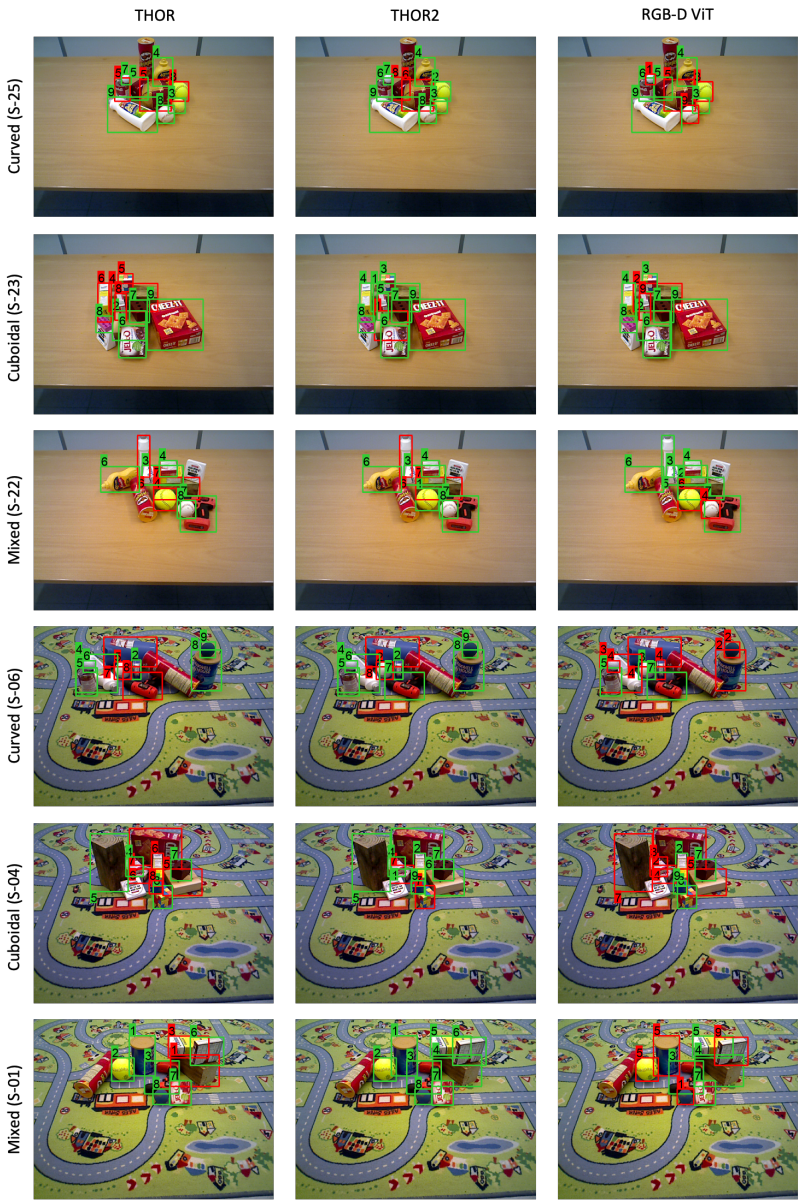


Figure 6.4: Sample results from the OCID dataset sequences recorded using the lower camera (green and red boxes represent correct and incorrect recognition, respectively). The top half shows results (obtained using THOR, THOR2, and RGB-D ViT) from sequences where objects are placed on a table and the bottom half shows results from sequences where objects are placed on the floor. The three rows in each half show results on sequences with curved, cuboidal, and mixed objects.

Table 6.2: Comparison of mean recognition accuracies (in %) of THOR, THOR2, and RGB-D ViT on the OCID dataset sequences recorded using the upper camera.

Place	Scene type	Seq. ID	THOR	THOR2	RGB-D ViT
Table	Curved	S-25	49.82 ± 1.79	**68.19 ± 3.57**	56.75 ± 5.59
		S-26	64.48 ± 2.49	**76.88 ± 3.22**	61.16 ± 4.43
		S-35	**58.91 ± 2.32**	**59.46 ± 2.96**	42.30 ± 3.64
		S-36	55.67 ± 1.15	53.23 ± 1.56	**63.31 ± 1.93**
	Cuboid	S-23	66.24 ± 1.03	**75.51 ± 3.48**	62.48 ± 4.07
		S-24	70.38 ± 3.39	**78.43 ± 4.18**	59.88 ± 0.64
		S-33	61.94 ± 1.42	**79.54 ± 4.47**	41.93 ± 5.75
		S-34	70.22 ± 1.64	**78.73 ± 1.67**	42.52 ± 4.63
	Mixed	S-21	**73.75 ± 2.01**	**78.32 ± 5.87**	41.85 ± 3.43
		S-22	62.54 ± 1.46	**74.69 ± 2.06**	**76.22 ± 6.53**
		S-31	**75.87 ± 0.95**	**79.96 ± 4.01**	**76.24 ± 1.58**
		S-32	**72.47 ± 1.32**	63.41 ± 4.67	**68.74 ± 3.92**
Floor	Curved	S-05	**64.17 ± 2.94**	**68.25 ± 2.13**	34.31 ± 4.94
		S-06	69.78 ± 3.63	**76.71 ± 2.66**	43.56 ± 4.90
		S-11	**58.77 ± 2.74**	**63.38 ± 3.51**	51.45 ± 2.79
		S-12	58.62 ± 1.74	**72.07 ± 1.98**	58.80 ± 3.49
	Cuboid	S-03	59.78 ± 1.07	**71.61 ± 2.93**	**66.07 ± 5.43**
		S-04	49.54 ± 1.34	**77.22 ± 3.30**	51.38 ± 3.09
		S-09	49.59 ± 1.35	**60.07 ± 4.01**	36.21 ± 4.30
		S-10	**77.04 ± 2.02**	**71.17 ± 4.14**	45.69 ± 8.00
	Mixed	S-01	64.06 ± 1.16	**85.43 ± 5.11**	65.02 ± 2.95
		S-02	**76.61 ± 1.65**	71.56 ± 3.06	51.47 ± 0.96
		S-07	**81.46 ± 1.78**	56.98 ± 2.54	64.15 ± 3.38
		S-08	**78.44 ± 2.38**	52.89 ± 4.70	30.59 ± 3.59
All sequences			66.50 ± 0.13	**71.44 ± 0.15**	55.02 ± 0.23

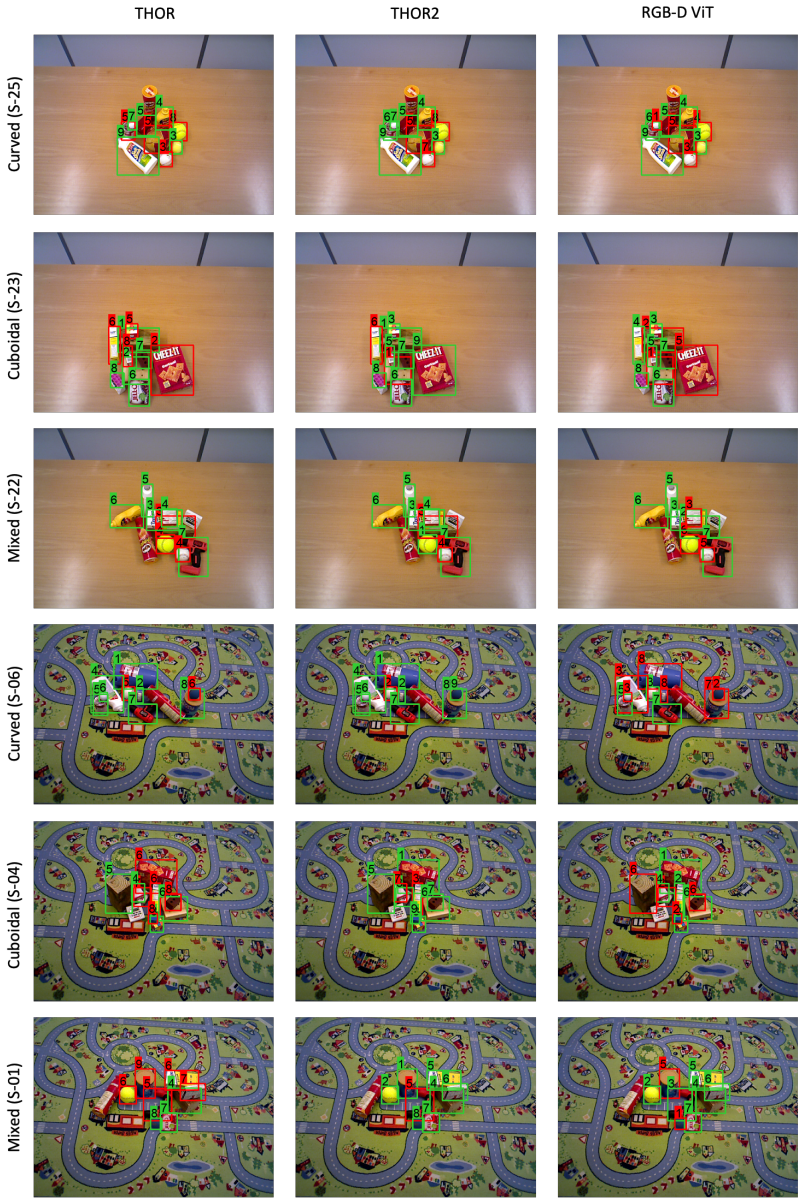


Figure 6.5: Sample results from the OCID dataset sequences recorded using the upper camera. The top half shows results (obtained using THOR, THOR2, and RGB-D ViT) from sequences where objects are placed on a table and the bottom half shows results from sequences where objects are placed on the floor. The three rows in each half show results on sequences with curved, cuboidal, and mixed objects.

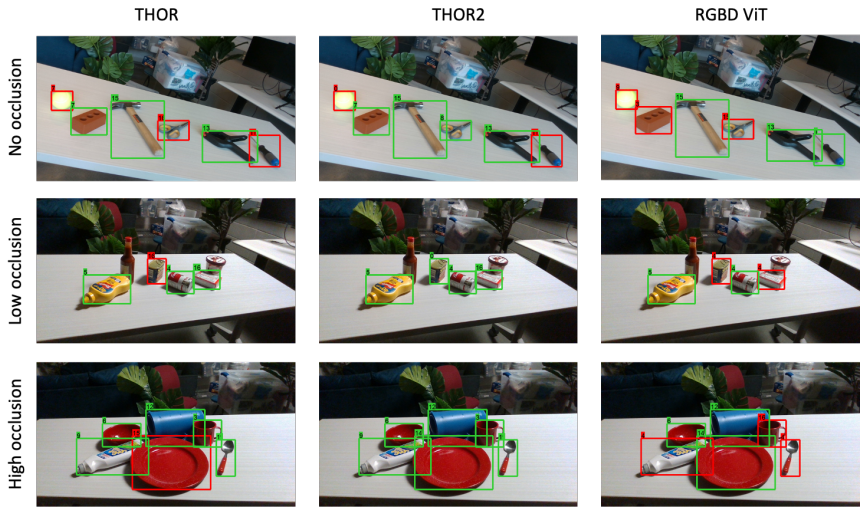


Figure 6.6: Sample results from the lounge environment of UW-IS Occluded dataset obtained using THOR, THOR2, and RGB-D ViT. The first, second, and third rows show results from scenes with three different levels of separations between objects.

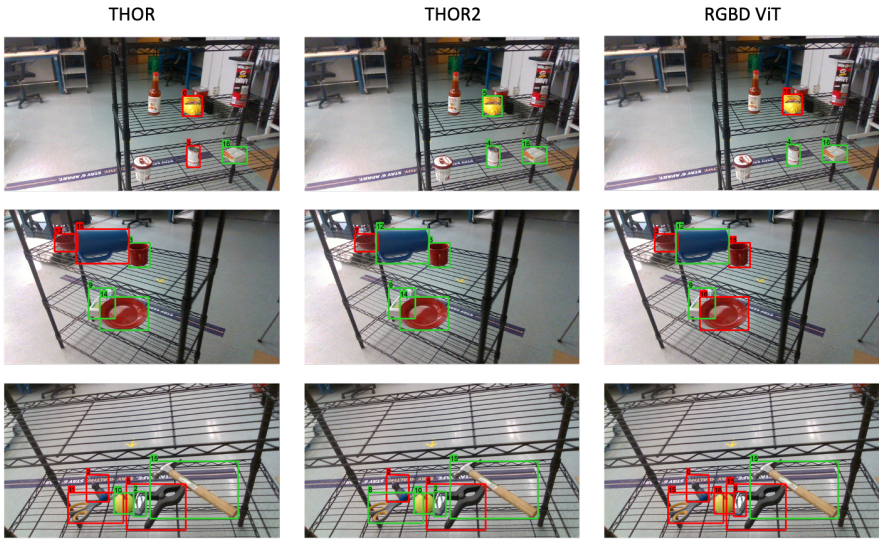


Figure 6.7: Sample results from the warehouse environment of UW-IS Occluded dataset obtained using THOR, THOR2, and RGB-D ViT. The first, second, and third rows show results from scenes with three different levels of separations between objects.

Table 6.3: Comparison of mean recognition accuracy over object classes belonging to different types (in %) on the UW-IS Occluded dataset. THOR2's performance is compared with THOR and RGB-D ViT.

Environment	Object type	THOR	THOR2	RGB-D ViT
Warehouse	Kitchen	52.33 ± 0.59	**62.92 ± 1.34**	45.02 ± 1.47
	Tools	46.89 ± 0.48	**53.55 ± 0.97**	35.42 ± 2.69
	Food	43.30 ± 1.48	**65.49 ± 2.98**	58.31 ± 2.00
Lounge	Kitchen	69.98 ± 0.43	**78.92 ± 1.32**	41.71 ± 1.34
	Tools	45.96 ± 0.62	**50.34 ± 0.96**	36.98 ± 1.47
	Food	45.93 ± 1.13	**63.50 ± 3.47**	52.32 ± 3.74
All environments and objects		52.22 ± 0.33	**62.58 ± 0.36**	42.96 ± 0.87

the gelatin box. Relatively smaller improvements are observed for objects belonging to the tools category. We believe this observation results from the fact that several objects in this category are small, have intricate shapes, and have shiny surfaces. In such cases, the captured color information is noisy due to comparatively more inaccurate depth sensing and instance segmentation. We refer the reader to Section 6.4.3 for further discussion on the impact of instance segmentation quality on THOR2's performance.

For both the environments, varying degrees of occlusion are considered in Table 6.4. THOR2 outperforms RGB-D ViT while gaining considerable performance improvements over THOR in the case of both unoccluded and occluded objects. Since the training data for all the three methods comprise images of unoccluded objects, it is particularly interesting to note that THOR2 outperforms RGB-D ViT even in the case of unoccluded objects. We investigate this trend further, in light of the sim2real gap between training and test data, by conducting further experiments where real-world RGB-D images are used for training RGB-D ViT. We refer the reader to the next subsection for details on those experiments. We also note from Table 6.4 that the impact of object occlusion is minimal on the performance of

Table 6.4: Comparison of mean recognition accuracy over all the object classes (in %) on the UW-IS Occluded dataset under varying degrees of occlusion. THOR2's performance is compared with THOR and RGB-D ViT.

Environment	Occlusion	THOR	THOR2	RGB-D ViT
Warehouse	None	51.62 ± 0.53	**61.40 ± 0.37**	43.65 ± 0.70
	Low	48.07 ± 0.28	**58.00 ± 0.49**	45.11 ± 1.37
	High	44.26 ± 0.25	**59.38 ± 0.35**	42.52 ± 1.07
Lounge	None	56.72 ± 0.60	**64.29 ± 0.34**	39.14 ± 2.07
	Low	54.45 ± 0.24	**65.87 ± 0.64**	43.06 ± 0.60
	High	51.88 ± 0.46	**59.95 ± 0.52**	43.50 ± 1.07

Table 6.5: Comparison of mean recognition accuracy over all the object classes (in %) on the UW-IS Occluded dataset under varying lighting conditions. THOR2's performance is compared with THOR and RGB-D ViT.

Environment	Lighting	THOR	THOR2	RGB-D ViT
Warehouse	Bright	47.52 ± 0.36	**61.14 ± 0.58**	42.15 ± 0.91
	Dim	48.11 ± 0.39	**58.28 ± 0.33**	47.48 ± 0.84
Lounge	Bright	50.76 ± 0.48	**62.68 ± 0.47**	40.07 ± 0.79
	Dim	57.64 ± 0.39	**63.95 ± 0.51**	44.80 ± 1.70

all three methods, thereby demonstrating the benefit of partial representations (slice-based in the case of THOR and THOR2 and patch-based in the case of RGB-D ViT).

Further, Table 6.5 shows that THOR2 outperforms RGB-D ViT regardless of the lighting conditions. It is particularly interesting to note that for each environment, the performance of THOR2 is comparable for both the lighting conditions. On the other hand, the lighting conditions impact RGB-D ViT's performance to a greater extent. This observation highlights the benefit of using the color network, which represents similar colors using a single node, to obtain the color embeddings for the TOPS2 descriptor.

Overall, we note that THOR2 achieves sizeable performance improvements over THOR with the help of additional color information while outperforming RGB-D ViT on both the OCID and UW-IS Occluded datasets.

Experiments with Real-world Training Data:

The previous section reports the recognition performance on real-world data when all the methods are trained using synthetic RGB-D images. The TOPS and TOPS2 descriptors used in THOR2, by virtue of their topological nature, are relatively robust to imprecise depth imagery or illumination-related changes in the real-world data. Since RGB-D ViT is not explicitly designed to achieve robustness to such issues, we perform additional experiments where real-world RGB-D images are used (in addition to the synthetic images) for training the RGB-D ViT. Table 6.6 reports the performance of these models on the real-world images of the UW-IS Occluded dataset.

Specifically, we use RGB-D images available from the YCB dataset for these experiments. In the first experiment, 20% of the RGB-D images available for each object are used for training and validation. These images are from one of the five RGB and depth sensor pairs used to record the YCB dataset. We observe that the corresponding RGB-D ViT model achieves approximately 4% higher accuracy on the UW-IS Occluded dataset. In subsequent experiments, we use 60% of the images (i.e., RGB-D images from three RGB and depth sensor pairs) and 100% of the images (i.e., RGB-D images from all five RGB and depth

Table 6.6: Comparison of mean recognition accuracy over all the object classes (in %) on the UW-IS Occluded dataset under varying degrees of occlusion. THOR2's performance is compared with RGB-D ViT trained using varying amounts of real-world data in addition to synthetic data.

Environment	Occlusion	THOR2	RGB-D ViT			
		Syn only	Syn only	Syn + 20% YCB	Syn + 60% YCB	Syn + 100% YCB
Warehouse	None	**61.40 ± 0.37**	43.65 ± 0.70	48.14 ± 1.72	50.17 ± 1.12	49.39 ± 2.57
	Low	**58.00 ± 0.49**	45.11 ± 1.37	47.56 ± 1.62	47.38 ± 1.78	47.25 ± 3.01
	High	**59.38 ± 0.35**	42.52 ± 1.07	48.38 ± 2.37	47.84 ± 1.91	46.45 ± 2.87
Lounge	None	**64.29 ± 0.34**	39.14 ± 2.07	44.72 ± 2.16	46.75 ± 1.64	46.84 ± 2.66
	Low	**65.87 ± 0.64**	43.06 ± 0.60	47.41 ± 0.76	47.63 ± 1.54	47.51 ± 2.26
	High	**59.95 ± 0.52**	43.50 ± 1.07	46.68 ± 1.75	48.31 ± 1.78	47.11 ± 2.90
All		**62.58 ± 0.36**	42.96 ± 0.87	47.04 ± 1.44	47.97 ± 1.47	47.41 ± 2.70

Note: Syn + x% YCB indicates that x% real images from the YCB dataset are used along with the entire synthetic dataset for training and validation.

sensor pairs). RGB-D ViT models from both the experiments achieve performance that is comparable to the model that is trained using only 20% of the YCB data.

Overall, these results indicate that RGB-D ViT's performance improves when real-world images are included during training. However, THOR2 continues to outperform RGB-D ViT without using any real-world training data, making it particularly suitable when gathering representative real-world training data is infeasible. We refer the reader to Section 6.4.2 for further discussion of the performance, in context of the sim2real gap between the training and test data.

Ablation Studies:

THOR2 uses two separate classifiers trained on TOPS and TOPS2 descriptors, respectively, for prediction. In this subsection, we perform ablation experiments using the UW-IS Occluded dataset to study this choice.

First, we report the recognition accuracy if only one of the descriptors is used for prediction. The fourth and fifth columns of Table 6.7 report the performance when TOPS and TOPS2 are used, respectively. Interestingly, the additional color embeddings in the TOPS2 descriptor lead to minor improvements in the warehouse environment and some instances of the lounge environment. However, the overall performance of TOPS and TOPS2 descriptors on the UW-IS Occluded dataset is comparable. At the same time, the color embeddings[3] (i.e., CE), themselves achieve a recognition accuracy approximately 8% higher than RGB-D ViT as shown in column three of Table 6.7. We hypothesize that the fusion of color (i.e., the color embeddings) and shape information (i.e., the persistence images) in the TOPS2 descriptor provides limited flexibility in selectively using these cues during testing; more selection flexibility could lead to performance improvements. Therefore, we perform additional experiments considering different ways of introducing such flexibility.

In particular, we obtain the recognition performance when a classifier trained on color

[3]The color embeddings are vectorized and stacked to obtain a single descriptor.

embeddings alone is used with another classifier trained on persistence images alone (i.e., the TOPS descriptor). The results in the sixth column of Table 6.7 show that incorporating higher flexibility by using separate classifiers for the color embeddings and the TOPS descriptor leads to higher overall performance than a single classifier trained using TOPS2.

Further analysis shows that even though incorporating selection flexibility is beneficial, the relative importance of the shape and color information in the prediction substantially impacts the performance.

For instance, using a color embeddings-based classifier and a TOPS2-based classifier (column seven of Table 6.7) implicitly assigns higher importance to color information than shape information. Similarly, equal importance is assigned to the shape and color information in the sixth and ninth columns in Table 6.7. The only scenario where the shape information, which is a more reliable cue for recognition, is assigned higher importance than color is in the case of THOR2. As a result, THOR2 achieves higher recognition performance than all the other scenarios.

These analyses indicate that selectively using the shape and color information for recognition, with relatively higher importance to the shape cues, results in the best overall recognition performance.

Robot Implementation

We also implement THOR2 on a LoCoBot platform built on a Yujin Robot Kobuki Base (YMR-K01-W1) powered by an Intel NUC NUC7i5BNH Mini PC. We mount an Intel RealSense D435 camera on top of the LoCoBot and control the robot using the PyRobot interface [124]. The LoCoBot is equipped with NVIDIA Jetson AGX Xavier processor that has a 512-core Volta GPU with Tensor Cores and an 8-core ARM v8.2 64-bit CPU. Fig. 6.8 shows a screenshot of the platform and sample recognition results.

THOR2 runs at an average rate of $0.7s$ per frame in a scene with six objects on this platform. This runtime includes the time for instance segmentation that is performed using

Table 6.7: Comparison of mean recognition accuracy over all the object classes (in %) achieved by using different combinations of topological representations for object recognition on the UW-IS Occluded dataset under varying degrees of occlusion.

Environment	Occlusion	CE	TOPS	TOPS2	CE + TOPS	CE + TOPS2	THOR2 (TOPS + TOPS2)	CE + TOPS + TOPS2
Warehouse	None	48.39 ± 0.29	54.39 ± 0.21	55.08 ± 0.85	54.46 ± 0.45	51.97 ± 0.52	**61.40 ± 0.37**	56.09 ± 0.44
	Low	46.49 ± 0.44	51.19 ± 0.30	52.07 ± 0.83	52.01 ± 0.48	49.60 ± 0.53	**58.00 ± 0.49**	53.23 ± 0.51
	High	48.93 ± 0.51	47.81 ± 0.24	55.31 ± 0.66	53.71 ± 0.40	53.35 ± 0.57	**59.38 ± 0.35**	56.31 ± 0.37
Lounge	None	52.43 ± 0.96	60.81 ± 0.09	55.44 ± 1.06	61.21 ± 0.40	55.43 ± 0.27	**64.29 ± 0.34**	61.52 ± 0.47
	Low	55.39 ± 0.44	56.87 ± 0.29	59.85 ± 0.78	61.19 ± 0.36	59.32 ± 0.26	**65.87 ± 0.64**	63.43 ± 0.54
	High	49.68 ± 0.57	54.77 ± 0.09	51.07 ± 0.93	57.13 ± 0.29	52.54 ± 0.42	**59.95 ± 0.52**	57.95 ± 0.36
All		51.01 ± 0.45	55.19 ± 0.12	55.44 ± 0.73	57.68 ± 0.25	54.58 ± 0.20	**62.58 ± 0.36**	59.20 ± 0.23

Note: CE represents the case when only the vectorized color embeddings of all the slices are used as the descriptor; `nameA` + . . . + `nameZ` indicates the case where classifiers corresponding to `nameA`, . . . , `nameZ` are all considered during testing, and the prediction with the highest probability is selected.

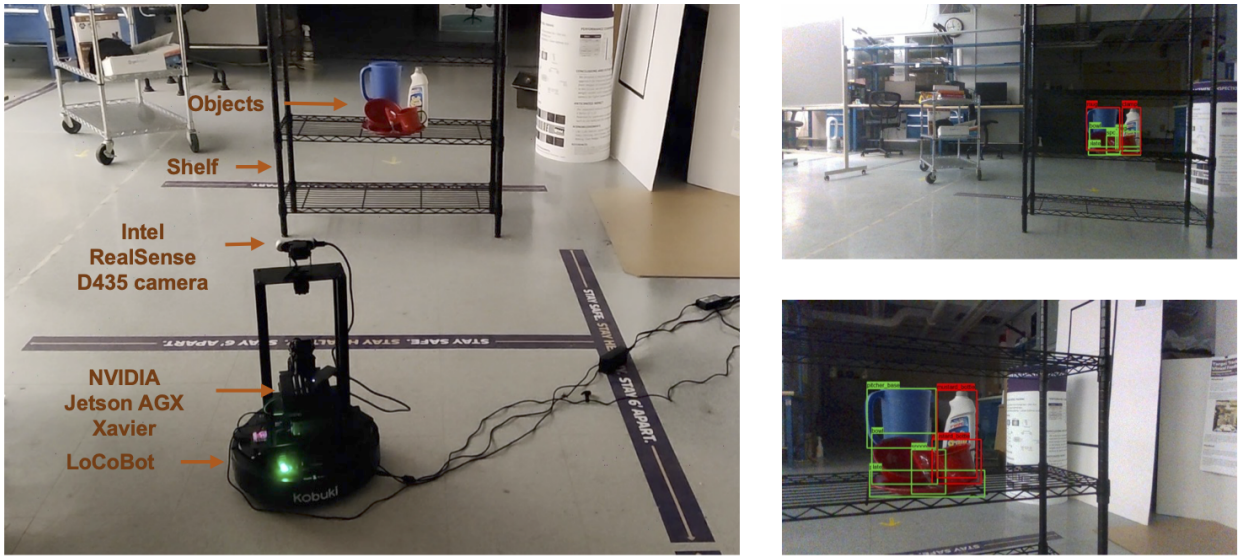


Figure 6.8: Screenshot of the LoCoBot operating in a mock warehouse setup (left) and the sample recognition results (right) obtained using THOR2 as the LoCoBot moves around the warehouse setup.

a TensorRT-optimized [125] depth seeding network (along with the initial mask processor module) from [197]. Recognition prediction for every object point cloud in the scene is then obtained simultaneously using multiprocessing in Python.

6.4 Discussion

The following subsections discuss several aspects of THOR2's performance.

6.4.1 THOR and THOR2 Comparison

THOR and THOR2 differ in two key aspects. First, THOR uses 3D shape information captured in the TOPS descriptor for recognition, whereas THOR2 uses 3D shape and color information captured in the TOPS and TOPS2 descriptors. Second, THOR uses a library of trained classifiers segregated based on the number of slices and view types (see Section 5.2), whereas THOR2 only uses two classifiers, one for each descriptor.

The results reported in Section 6.3.3 illustrate that incorporating color information helps THOR2 achieve higher overall performance than THOR on both the OCID and the UW-IS Occluded datasets. However, unlike THOR, THOR2 undergoes a slight drop in performance when the OCID dataset sequences are recorded using the upper camera (see Tables 6.1 and 6.2). This observation is consistent with our findings from the previous chapter (see Section 5.4.2), which show that the library of classifiers used in THOR equips it for recognition in such scenarios. Despite the drop in performance, THOR2 achieves higher accuracy than THOR. Moreover, the fewer classifiers in THOR2 make it more easily extendable to an open-world scenario where new object classes may appear. For instance, when new classes appear, a class-adaptive extension of THOR2 (with two classifiers) would be more efficiently updated than THOR, which requires updating all the classifiers in the library.

6.4.2 Sim2Real Gap

We train THOR and THOR2 using synthetic images of unoccluded objects and test them on real-world images of cluttered scenes. Table 6.4 shows that THOR2 outperforms RGB-D ViT

for unoccluded objects when the latter is also trained using synthetic images of unoccluded objects. Further, Table 6.6 shows that when synthetic and real-world RGB-D images are used to train RGB-D ViT, THOR2 continues to achieve high recognition accuracy for unoccluded objects using synthetic data alone. This observation indicates that THOR2 is better at accounting for the sim2real gap, particularly when a large volume of real-world RGB-D data is unavailable for training the RGB-D ViT. We attribute this observation to the stability of the PIs in the TOPS and TOPS2 descriptors with respect to minor perturbations in the filtration [4], providing some robustness to the noise in the depth images. In addition, the color network-based embeddings also provide a certain degree of robustness to the variation in object colors.

6.4.3 THOR2 Failure Modes

Instance Segmentation Quality:

The TOPS and TOPS2 descriptors embody the idea of object unity, making THOR2 relatively robust to over-segmentation errors or errors where the object masks are split into segments due to occlusion. However, under-segmentation and false positive segmentations are more challenging. Further, a certain degree of robustness to errors arising from misaligned object boundaries is also achieved through the test-time outlier removal from point clouds. On a related note, the temporally smoothed depth images (and point clouds) provided in the OCID dataset are relatively less noisy and do not require outlier removal. Such temporal smoothing leads to a misalignment between the RGB and depth image corresponding to a given timestep in the sequence. In sequences where the misalignment is more pronounced (e.g., sequences S-02 and S-07), the object point clouds are incorrectly colored on the boundaries resulting in a somewhat lower performance using the shape and color-based THOR2 compared to exclusively shaped-based THOR.

Specific Occlusion Scenarios:

Similar to THOR, recognizing heavily occluded objects (see Fig. 5.4), such as the bleach cleanser in S-36, is challenging for THOR2. Further, Section 6.2.3 mentions that if an object is occluded, the aligned point cloud is reoriented to ensure that the first slice on the occluded end of the object (i.e., the end where one or more slices may be missing) is not the first slice during subsequent descriptor computation. Therefore, scenarios where an object is occluded such that there are missing slices on both the ends pose difficulties. Such occlusion scenarios are relatively infrequent, and we believe additional information, such as an RGB-D image from a different viewpoint, would be necessary to perform recognition. For instance, the bleach cleanser in S-22 appears occluded on both ends in the lower camera view (see the third row in Fig. 6.4), leading to incorrect recognition in the case of THOR and THOR2. However, the third row of Fig. 6.5 shows that the same bleach cleanser is correctly recognized in the upper camera view.

6.5 Summary

This work builds upon the TOPS descriptor and the THOR framework to obtain a new topological descriptor, TOPS2, and an accompanying human-inspired recognition framework, THOR2, for recognizing occluded objects in unseen indoor environments. We use the Mapper algorithm to identify useful color regions and capture their connectivity to obtain a color network. Color embeddings for object point clouds generated using the color network are interleaved with the TOPS descriptor to obtain the TOPS2 descriptor. Our slicing-based approach ensures similarities between the descriptors of the occluded and the corresponding unoccluded objects. We use this similarity in THOR2 to perform object recognition using 3D shape and color information.

We report performance comparisons on two datasets: OCID and UW-IS Occluded. On the OCID dataset, comparisons with THOR and RGB-D ViT show that THOR2 benefits from incorporating color information and achieves the best overall performance when in-

creasingly cluttered scenes are viewed from different camera positions. THOR2 achieves substantially higher recognition accuracy on the UW-IS Occluded dataset than RGB-D ViT, regardless of the environmental condition. Moreover, THOR2 continues to outperform RGB-D ViT even when real-world RGB-D images from the YCB dataset are used along with synthetic images to train the latter. Therefore, THOR2 is a promising 3D shape and color-based approach toward robust recognition in low-cost robots, meant for everyday use in indoor settings.

Chapter 7
CONCLUSIONS

This chapter summarizes the core contributions of this dissertation, the anticipated impact of this work on the research community, and the potential directions for future work.

7.1 Core Contributions

This dissertation presents a substantial body of work geared toward achieving robust object recognition in unseen environments, particularly emphasizing the use of tools from computational topology. The core contributions of this work are as follows.

1. *Suitability of topological representations for object recognition in unseen environments*: Chapter 4 of this dissertation demonstrates the suitability of topological representations for object recognition in unseen environments. Specifically, 2D shape representations, i.e., the Sparse PI features, are obtained by capturing the topologically persistent features in object segmentation maps. The Sparse PI features achieve better recognition performance in unseen indoor environments than the features learned from widely-used deep learning networks such as ResNetV2-56 and EfficientNet-B4. The benefits of using the Sparse PI representations are clearly observed when they perform recognition in an unseen test environment. The overall recognition performance (in terms of recall and accuracy) using the Sparse PI features remains unaffected, as opposed to the widely-used Faster R-CNN and its domain-adaptive counterpart, DA-FR-CNN*****, whose performances drop considerably. These findings show that using topological representations is a promising way to achieve robust object recognition.

2. *Object unity-based topological representation for 3D shape-based object recognition in*

unseen cluttered environments: Building upon the promise of topological representations, Chapter 5 presents a first-of-its-kind human reasoning-inspired framework, termed THOR, for object recognition in unseen cluttered environments. Persistent homology is used in a novel slicing-based manner to embody object unity and obtain structural descriptors, TOPS, of object point clouds generated from depth images. The combination of persistent homology and object unity in the TOPS descriptors leads to relatively higher robustness to imprecise depth imagery-related noise and object occlusion than widely used classical 3D shape descriptors (CVFH and ESF). The reasoning-inspired framework, THOR, uses a library of classifiers (i.e., SVM and MLP) trained on synthetic depth images of unoccluded objects. It achieves substantially higher recognition accuracy than state-of-the-art learning-based approaches (DGCNN and SimpleView) in cluttered real-world environments regardless of the lighting conditions or the degree of occlusion. These results position THOR as a promising way to achieve robust object recognition for everyday-use robots equipped with commodity hardware operating in unseen environments.

3. *Topological soft clustering-based representation of color for 3D shape and color-based object recognition in unseen cluttered environments:* Chapter 6 extends the work from Chapter 5 and presents the TOPS2 descriptor that captures the shape and the color of objects while embodying the principle of object unity. In a one-of-its-kind approach, the Mapper algorithm for topological soft clustering is applied to obtain color regions (i.e., clusters of similar colors) and capture the connectivity between them. The similarity and connectivity between the colors captured in the resulting color network are leveraged to seek color embeddings that are relatively independent of the lighting conditions for building the TOPS2 descriptor. The accompanying THOR2 framework (trained entirely on synthetic RGB-D images of unoccluded objects) witnesses sizeable improvements over the shape-based THOR framework while achieving substantially higher recognition accuracy than RGB-D ViT (trained on synthetic and real-world

data) in two different real-world environments with varying environmental conditions and degree of clutter. With these results, THOR2 presents itself as a shape and color-based object recognition approach for everyday-use robots operating in unseen environments.

4. *Datasets for systematical evaluation of object recognition methods in unseen environments:* This dissertation contributes two datasets curated for systematically evaluating object recognition methods in unseen environments. Chapter 4 presents the UW Indoor Scenes (UW-IS) Dataset, which covers fourteen object types and comprises scene images from two different environments: a living room and a mock warehouse. The images are taken in four different illumination settings from three different camera perspectives and varying distances up to two meters. Bounding box annotations and semantic segmentation masks are provided for all the images. Chapter 5 presents the UW-IS Occluded dataset curated using commodity hardware to reflect real-world scenarios with different environmental conditions and degrees of object occlusion. It comprises two different environments: a lounge and a mock warehouse. The dataset has RGB-D images from 36 videos for each of these environments, comprising five to seven objects each, taken from distances up to approximately 2 m. The videos cover two different lighting conditions and three different levels of occlusion for three object categories (a total of 20 object classes). Semi-automatically generated instance segmentation masks are provided for all the images. For potential use as a benchmark to evaluate other robot vision tasks, such as pose estimation, 6D pose annotations are also provided.

7.2 Anticipated Impact

Topological methods have been used for planning and navigation. However, their applications to perception have been limited. This dissertation advocates using computational topology tools for achieving robust object recognition in everyday-use low-cost robots. Such

an approach can have impacts on robot perception that go beyond object recognition.

Specifically, we note that analogous to a human, a robot's visual behavior is guided by its need to move in and interact with its environment [23]. Therefore, object representations may be intertwined with high-level representations that support actions and spatial behavior. For instance, geometry or topology-based object representations (such as the TOPS descriptor) can support the computation of geometric affordance representations for action-based behaviors (e.g., pick-and-place). Contemporary methods rely on object classification, symbolic representations, or human-inspired learning stages to learn the affordance relations for a limited set of objects. However, treating the affordances as relations among multiple interacting entities and using the geometric information associated with the interacting surfaces can be used to compute the interaction representations [144] and the corresponding geometric affordances. On a related note, the computation of derived representations of the topological relationships among objects to support navigational behaviors in vision-based SLAM approaches can also be investigated.

Note that most of the approaches proposed in this dissertation consider a synthetically generated dataset for training and show better transferability to the real world than the learning-based methods. On the other hand, generating high-quality simulated data continues to become more manageable with diffusion models [42, 100]. Therefore, computational topology-based methods (such as those proposed in this dissertation) combined with high-quality simulations generated from diffusion models present themselves as a promising way to address the gap between the simulation and the real world.

7.3 Future Work

Although this dissertation provides the foundations for object recognition in unstructured, i.e., cluttered, uncertain, and dynamic environments using topological representations, further work needs to be done in the following areas to achieve it.

7.3.1 Active Object Recognition

Human perception is a process of active exploration to obtain additional information about the environment. However, all the object recognition methods discussed in this work are passive, *feedforward* approaches, i.e., a single scene image is used to recognize objects. As noted in Section 6.4.3, heavily occluded objects can be recognized by perceiving them from a different viewpoint. Furthermore, exploring different viewpoints can also help recognize objects with ambiguous placements. Therefore, a promising direction for future work is to extend the frameworks proposed in this work to perform active object recognition. Specifically, a viewpoint suitability criterion can be designed and optimized to actively seek a better viewpoint for improved recognition [31]. Another promising direction of future work could be an extension of the active exploration setup where the robotic system purposefully manipulates the scene to aid its perception of the occluded objects.

Active exploration also plays a vital role in perceiving objects heavily occluded during in-hand manipulation. In such cases, information about the contact surface's shape from the data obtained using vision-based tactile sensors can provide important information for subsequent recognition (and pose estimation) via shape completion. Existing works [48, 166] use learning-based methods to obtain the shape information from such data. Since the central idea of TDA is to recover the topological space underlying any given data and infer robust information from it, a natural avenue for future work is to study the suitability of the computational topology tools used in this dissertation for recovering the local shape information from tactile data and obtaining a unified representation of visual and tactile data.

7.3.2 Object Recognition in an Open World

As mentioned in Chapter 1, all the approaches in this dissertation consider a closed-world scenario, i.e., all the object classes are known. Therefore, the problem of object recognition in unstructured environments boils down to addressing the covariate shifts between the

training and test data distributions. However, the real world is an *open-world* scenario, where new object classes may appear in the environment. Such semantic shifts in the test data distribution pose additional challenges for recognition in an unstructured environment. Vaze et al., [184] show that the accuracy of model-based methods on closed-set classes is correlated with its ability to make the 'none-of-above' decision reliably. In light of their findings, the topological constitution of the TOPS and TOPS2 descriptors provide a promising foundation for developing certifiable frameworks [157,168] for recognition. Such an approach would then lend itself to developing self-supervised class-adaptive frameworks for recognition in an open world.

7.3.3 Object Recognition in Dynamic Environments

Real-world environments are dynamic, where both spatial and temporal occlusions occur. Therefore, accounting for objects that temporarily disappear from a robot's field of view requires approaches that track objects while incorporating the idea of *object permanence* [175]. Object permanence refers to the understanding that an object exists even if it cannot be seen or touched. Alternatively, a unified representation for actionable spatial perception, such as the 3D dynamic scene graphs [143], can be maintained to keep track of all the objects in a scene for subsequent planning. Topological representations, such as those presented in this work, can support fast, incremental updates to the spatial concepts and spatiotemporal relations in such graphs.

BIBLIOGRAPHY

[1] Andreas Aakerberg, Kamal Nasrollahi, and Thomas Heder. Improving a deep learning based RGB-D object recognition model by ensemble learning. In *IEEE International Conference on Image Processing Theory, Tools and Applications*, pages 1–6, 2017.

[2] Andreas Aakerberg, Kamal Nasrollahi, Christoffer Bøgelund Rasmussen, and Thomas B Moeslund. Depth value pre-processing for accurate transfer learning based RGB-D object recognition. In *International Joint Conference on Computational Intelligence*, pages 121–128, 2017.

[3] Saeedeh Abasi, Mohammad Amani Tehran, and Mark D Fairchild. Distance metrics for very large color differences. *Color Research & Application*, 45(2):208–223, 2020.

[4] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(1):218–252, 2017.

[5] Mahmoud Afifi, Marcus A Brubaker, and Michael S Brown. Auto white-balance correction for mixed-illuminant scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1210–1219, 2022.

[6] Aitor Aldoma, Federico Tombari, Radu Bogdan Rusu, and Markus Vincze. OUR-CVFH–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 113–122. Springer, 2012.

[7] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. CAD-model recognition and 6DOF pose estimation using 3D cues. In *IEEE International Conference on Computer Vision Workshops*, pages 585–592, 2011.

[8] Piotr Antonik, Nicolas Marsal, Daniel Brunner, and Damien Rontani. Human action recognition with a large-scale brain-inspired photonic computer. *Nature Machine Intelligence*, 1(11):530–537, 2019.

[9] Vinicius F Arruda, Thiago M Paixão, Rodrigo F Berriel, Alberto F De Souza, Claudine Badue, Nicu Sebe, and Thiago Oliveira-Santos. Cross-domain car detection using unsupervised image-to-image translation: From day to night. In *International Joint Conference on Neural Networks*, pages 1–8, 2019.

[10] Umar Asif, Mohammed Bennamoun, and Ferdous A Sohel. A multi-modal, discriminative and spatially invariant CNN for RGB-D object labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2051–2065, 2017.

[11] Dana H Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[13] William J Beksi and Nikolaos Papanikolopoulos. Signature of topologically persistent points for 3D point cloud description. In *IEEE International Conference on Robotics and Automation*, pages 3229–3234, 2018.

[14] William Joseph Beksi. *Topological Methods for 3D Point Cloud Processing*. PhD thesis, University of Minnesota, 2018.

[15] Subhrajit Bhattacharya, Florian T Pokorny, and Howie Choset. Guest editorial: Special issue on "topological methods in robotics". *Autonomous Robots*, 45(5):611–612, 2021.

[16] Silvia Biasotti, Andrea Cerri, Alex Bronstein, and Michael Bronstein. Recent trends, applications, and perspectives in 3D shape similarity assessment. In *Computer Graphics Forum*, volume 35, pages 87–119. Wiley Online Library, 2016.

[17] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in Neural Information Processing Systems*, 24, 2011.

[18] Manuel Blum, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller. A learned feature descriptor for object recognition in RGB-D data. In *IEEE International Conference on Robotics and Automation*, pages 1298–1303, 2012.

[19] Liefeng Bo, Kevin Lai, Xiaofeng Ren, and Dieter Fox. Object recognition with hierarchical kernel descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1729–1736, 2011.

[20] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. *Advances in Neural Information Processing systems*, 23, 2010.

[21] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Depth kernel descriptors for object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 821–826, 2011.

[22] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, pages 387–402, 2013.

[23] Stefania Bracci and Hans P Op de Beeck. Understanding human object vision: A picture is worth a thousand representations. *Annual Review of Psychology*, 74:113–135, 2023.

[24] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *European Conference on Computer Vision*, pages 536–551, 2014.

[25] Björn Browatzki, Jan Fischer, Birgit Graf, Heinrich H Bülthoff, and Christian Wallraven. Going into depth: Evaluating 2D and 3D cues for object classification on a new, large-scale object dataset. In *IEEE International Conference on Computer Vision Workshops*, pages 1189–1195, 2011.

[26] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16(1):77–102, 2015.

[27] Serhat S Bucak, Rong Jin, and Anil K Jain. Multiple kernel learning for visual object recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1354–1369, 2013.

[28] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[29] Ali Caglayan, Nevrez Imamoglu, Ahmet Burak Can, and Ryosuke Nakamura. When CNNs meet random RNNs: Towards multi-level analysis for RGB-D object and scene recognition. *Computer Vision and Image Understanding*, 217:103373, 2022.

[30] Qi Cai, Yingwei Pan, Chong-Wah Ngo, Xinmei Tian, Lingyu Duan, and Ting Yao. Exploring object relation in mean teacher for cross-domain detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11457–11466, 2019.

[31] Berk Calli, Wouter Caarls, Martijn Wisse, and Pieter P Jonker. Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation. *IEEE Transactions on Automation Science and Engineering*, 15(4):1810–1822, 2018.

[32] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: Using the yale-CMU-Berkeley object and model set. *IEEE Robotics and Automation Magazine*, 22(3):36–52, 2015.

[33] Fabio Maria Carlucci, Paolo Russo, and Barbara Caputo. $(DE)^2CO$: Deep depth colorization. *IEEE Robotics and Automation Letters*, 3(3):2386–2393, 2018.

[34] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *ACM symposium on Theory of computing*, pages 380–388, 2002.

[35] Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, 4:667963, 2021.

[36] Chaoqi Chen, Zebiao Zheng, Xinghao Ding, Yue Huang, and Qi Dou. Harmonizing transferability and discriminability for adapting object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8869–8878, 2020.

[37] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with atrous separable convolution for semantic image segmentation. In *European Conference On Computer Vision*, 2018.

[38] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain Adaptive Faster R-CNN for object detection in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3339–3348, 2018.

[39] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Revisiting R-CNN: On awakening the classification power of Faster R-CNN. In *European Conference On Computer Vision*, pages 453–468, 2018.

[40] Yanhua Cheng, Rui Cai, Xin Zhao, and Kaiqi Huang. Convolutional fisher kernels for RGB-D object recognition. In *International Conference on 3D Vision*, pages 135–143, 2015.

[41] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in Python*, volume 146. Springer Nature, 2023.

[42] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[43] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[44] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[45] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[46] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. BlenderProc. *arXiv preprint arXiv:1911.01911*, 2019.

[47] Chaitanya Devaguptapu, Ninad Akolekar, Manuj M Sharma, and Vineeth N Balasubramanian. Borrow from anywhere: Pseudo multi-modal object detection in thermal imagery. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[48] Snehal Dikhale, Karankumar Patel, Daksh Dhingra, Itoshi Naramura, Akinobu Hayashi, Soshi Iba, and Nawid Jamali. VisuoTactile 6D pose estimation of an in-hand object using vision and tactile sensor data. *IEEE Robotics and Automation Letters*, 7(2):2148–2155, 2022.

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[50] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002.

[51] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust RGB-D object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 681–687, 2015.

[52] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[53] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *Internation Journal of Computer Vision*, 88(2):303–338, 2010.

[54] Chen Fang, Ye Xu, and Daniel N Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *IEEE International Conference on Computer Vision*, pages 1657–1664, 2013.

[55] Michael Farber. Topology of robot motion planning. In *Morse Theoretic Methods in Nonlinear Analysis and in Symplectic Topology*, pages 185–230, 2006.

[56] Duc Fehr, William J Beksi, Dimitris Zermas, and Nikolaos Papanikolopoulos. Covariance based point cloud descriptors for object detection and recognition. *Computer Vision and Image Understanding*, 142:80–93, 2016.

[57] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[58] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009.

[59] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[60] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[61] Mingliang Gao, Jun Jiang, Guofeng Zou, Vijay John, and Zheng Liu. RGB-D-based object recognition using multimodal convolutional neural networks: a survey. *IEEE Access*, 7:43110–43136, 2019.

[62] Adélie Garin and Guillaume Tauzin. A topological "reading" lesson: Classification of MNIST using TDA. In *IEEE International Conference on Machine Learning Applications*, pages 1551–1556, 2019.

[63] Matan Gavish and David L Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.

[64] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *IEEE International Conference on Computer Vision*, pages 2551–2559, 2015.

[65] Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

[66] Rohit Girdhar, Alaaeldin El-Nouby, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. OmniMAE: Single model masked pretraining on images and videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10406–10417, 2023.

[67] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. OMNIVORE: A single model for many visual modalities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 16102–16112, 2022.

[68] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[69] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[70] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *International Conference on Machine Learning*, pages 3809–3820, 2021.

[71] Kristen Grauman and Bastian Leibe. Visual object recognition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(2):1–181, 2011.

[72] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022.

[73] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

[74] Tiantong Guo, Cong Phuoc Huynh, and Mashhour Solh. Domain-adaptive pedestrian detection in thermal images. In *IEEE International Conference on Image Processing (ICIP)*, pages 1660–1664, 2019.

[75] Wei Guo, Krithika Manohar, Steven L Brunton, and Ashis G Banerjee. Sparse-TDA: Sparse realization of topological data analysis for multi-way classification. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1403–1408, 2018.

[76] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3D object recognition in cluttered scenes with local surface features: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.

[77] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360, 2014.

[78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference On Computer Vision*, pages 630–645, 2016.

[79] Zhenwei He and Lei Zhang. Multi-Adversarial Faster-RCNN for unrestricted object detection. In *IEEE International Conference on Computer Vision*, pages 6668–6677, 2019.

[80] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, pages 548–562, 2013.

[81] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive domain adaptation for object detection. In *IEEE Winter Conference on Applications of Computer Vision*, pages 749–757, 2020.

[82] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7310–7311, 2017.

[83] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[84] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5001–5009, 2018.

[85] Lu Jin, Zechao Li, Xiangbo Shu, Shenghua Gao, and Jinhui Tang. Partially common-semantic pursuit for RGB-D object recognition. In *ACM International Conference on Multimedia*, pages 959–962, 2015.

[86] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational homology*, volume 157. Springer Science and Business Media, 2006.

[87] S Hamidreza Kasaei, Maryam Ghorbani, Jits Schilperoort, and Wessel van der Rest. Investigating the importance of shape features, color constancy, color spaces, and similarity measures in open-ended 3D object recognition. *Intelligent Service Robotics*, 14(3):329–344, 2021.

[88] Keras.

[89] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G Macready. A robust learning approach to domain adaptive object detection. In *IEEE International Conference on Computer Vision*, pages 480–490, 2019.

[90] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012.

[91] Taekyung Kim, Minki Jeong, Seunghyeon Kim, Seokeon Choi, and Changick Kim. Diversify and match: A domain adaptive representation learning paradigm for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12456–12465, 2019.

[92] Roman Klokov and Victor Lempitsky. Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In *IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[94] Krumo. Domain Adaptive Faster R-CNN.

[95] Lars Kunze, Nick Hawes, Tom Duckett, Marc Hanheide, and Tomáš Krajník. Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters*, 3(4):4023–4030, 2018.

[96] Laksono Kurnianggoro, Kang-Hyun Jo, et al. A survey of 2D shape representation: Methods, evaluations, and future research directions. *Neurocomputing*, 300:1–16, 2018.

[97] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3D scene labeling. In *IEEE International Conference on Robotics and Automation*, pages 3050–3057, 2014.

[98] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE Internation Conference on Robotics and Automation*, pages 1817–1824, 2011.

[99] Longin Jan Latecki, Rolf Lakamper, and T Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 424–429, 2000.

[100] Jiabao Lei, Jiapeng Tang, and Kui Jia. RGBD2: Generative scene synthesis via incremental view inpainting using RGBD diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8422–8434, 2023.

[101] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 878–885, 2005.

[102] Wanyi Li, Fuyu Li, Yongkang Luo, and Peng Wang. Deep domain adaptive object detection: A survey. *arXiv preprint arXiv:2002.06797*, 2020.

[103] Che-Tsung Lin. Cross domain adaptation for on-road object detection using multimodal structure-consistent image-to-image translation. In *IEEE International Conference on Image Processing*, pages 3029–3030, 2019.

[104] Hong Liu, Pinhao Song, and Runwei Ding. Towards domain generalization in underwater object detection. In *IEEE International Conference on Image Processing*, pages 1971–1975, 2020.

[105] Hong Liu, Pinhao Song, and Runwei Ding. WQT and DG-YOLO: Towards domain generalization in underwater object detection. *arXiv preprint arXiv:2004.06333*, 2020.

[106] Huaping Liu, Fengxue Li, Xinying Xu, and Fuchun Sun. Multi-modal local receptive field extreme learning machine for object recognition. *Neurocomputing*, 277:4–11, 2018.

[107] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318, 2020.

[108] Shuo Liu, Vijay John, Erik Blasch, Zheng Liu, and Ying Huang. IR2VI: enhanced night environmental perception by unsupervised thermal image translation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1153–1160, 2018.

[109] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot multibox Detector. In *European Conference On Computer Vision*, pages 21–37, 2016.

[110] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-Shape convolutional neural network for point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[111] Mohammad Reza Loghmani, Barbara Caputo, and Markus Vincze. Recognizing objects in-the-wild: Where do we stand? In *IEEE International Conference on Robotics and Automation*, pages 2170–2177, 2018.

[112] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[113] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2015.

[114] Yangxiao Lu, Ninad Khargonkar, Zesheng Xu, Charles Averill, Kamalesh Palanisamy, Kaiyu Hang, Yunhui Guo, Nicholas Ruozzi, and Yu Xiang. Self-supervised unseen object instance segmentation via long-term robot interaction. *arXiv preprint arXiv:2302.03793*, 2023.

[115] M Ronnier Luo, Guihua Cui, and Bryan Rigg. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 26(5):340–350, 2001.

[116] David L MacAdam. Visual sensitivities to color differences in daylight. *Josa*, 32(5):247–274, 1942.

[117] C Mach et al. Random Sample Consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Readings in Computer Vision*, pages 726–740, 1981.

[118] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *IEEE International Conference on Robotics and Automation*, pages 3235–3242, 2018.

[119] David Marr. Early processing of visual information. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 275(942):483–519, 1976.

[120] Zoltan-Csaba Marton, Dejan Pangercic, Radu Bogdan Rusu, Andreas Holzbach, and Michael Beetz. Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *IEEE International Conference on Humanoid Robots*, pages 365–370, 2010.

[121] Daniel Maturana and Sebastian Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 922–928, 2015.

[122] Kirill Mazur and Victor Lempitsky. Cloud transformers: A universal approach to point cloud processing tasks. In *IEEE International Conference on Computer Vision*, pages 10715–10724, 2021.

[123] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013.

[124] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. PyRobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.

[125] Nvidia. TensorRT.

[126] Joseph O'Rourke. Finding minimal enclosing boxes. *International journal of computer & information sciences*, 14(3):183–199, 1985.

[127] Deepti Pachauri, Chris Hinrichs, Moo K Chung, Sterling C Johnson, and Vikas Singh. Topology-based kernels with application to inference problems in Alzheimer's disease. *IEEE Transactions on Medical Imaging*, 30(10):1760–1770, 2011.

[128] Panda3D: Open source framework for 3D rendering & Games, May 2018.

[129] David Paulk, Vangelis Metsis, Christopher McMurrough, and Fillia Makedon. A supervised learning approach for fast object recognition from RGB-D data. In *International Conference on Pervasive Technologies Related to Assistive Environments*, pages 1–8, 2014.

[130] Fatih Porikli, Oncel Tuzel, and Peter Meer. Covariance tracking using model update based on lie algebra. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 728–735, 2006.

[131] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[132] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[133] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[134] Mohammad Muntasir Rahman, Yanhao Tan, Jian Xue, and Ke Lu. RGB-D object recognition with multimodal deep convolutional neural networks. In *IEEE International Conference on Multimedia and Expo*, pages 991–996, 2017.

[135] Brenda Rapp. *Handbook of cognitive neuropsychology: What deficits reveal about the human mind*. Psychology Press, 2015.

[136] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[137] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[138] Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4741–4748, 2015.

[139] Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. *arXiv preprint arXiv:2202.03377*, 2022.

[140] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[141] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[142] Vanessa Robins, Peter John Wood, and Adrian P Sheppard. Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1646–1658, 2011.

[143] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *arXiv preprint arXiv:2002.06289*, 2020.

[144] Eduardo Ruiz and Walterio Mayol-Cuevas. Geometric affordance perception: Leveraging deep 3D saliency with the interaction tensor. *Frontiers in Neurorobotics*, 14:45, 2020.

[145] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D recognition and pose using the viewpoint feature histogram. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162, 2010.

[146] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[147] Radu Bogdan Rusu, Andreas Holzbach, Michael Beetz, and Gary Bradski. Detecting and segmenting objects for mobile manipulation. In *IEEE International Conference on Computer Vision Workshops*, pages 47–54, 2009.

[148] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6956–6965, 2019.

[149] Ekta U Samani and Ashis G Banerjee. Topologically persistent features-based object recognition in cluttered indoor environments. *arXiv preprint arXiv:2205.07479*, 2022.

[150] Ekta U Samani and Ashis G Banerjee. Persistent homology meets object unity: Object recognition in clutter. *arXiv preprint arXiv:2305.03815*, 2023.

[151] Ekta U Samani, Xingjian Yang, and Ashis G Banerjee. Visual object recognition in indoor environments using topologically persistent features. *IEEE Robotics and Automation Letters*, 6(4):7509–7516, 2021.

[152] Jordi Sanchez-Riera, Kai-Lung Hua, Yuan-Sheng Hsiao, Tekoing Lim, Shintami C Hidayati, and Wen-Huang Cheng. A comparative study of data fusion for RGB-D based visual recognition. *Pattern Recognition Letters*, 73:1–6, 2016.

[153] Max Schwarz, Hannes Schulz, and Sven Behnke. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *IEEE International Conference on Robotics and Automation*, pages 1329–1335, 2015.

[154] Karthik Seemakurthy, Charles Fox, Erchan Aptoula, and Petra Bosilj. Domain generalisation for object detection. *arXiv preprint arXiv:2203.05294*, 2022.

[155] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Neighbors do help: Deeply exploiting local structures of point clouds. *arXiv preprint arXiv:1712.06760*, 1(2), 2017.

[156] Zhiqiang Shen, Harsh Maheshwari, Weichen Yao, and Marios Savvides. SCL: Towards accurate domain adaptive object detection via gradient detach based stacked complementary losses. *arXiv preprint arXiv:1911.02559*, 2019.

[157] Jingnan Shi, Rajat Talak, Dominic Maggio, and Luca Carlone. A correct-and-certify approach to self-supervise object pose estimators via ensemble self-training. *arXiv preprint arXiv:2302.06019*, 2023.

[158] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702, 2017.

[159] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[160] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. BigBird: A large-scale 3D database of object instances. In *IEEE International Conference on Robotics and Automation*, pages 509–516, 2014.

[161] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. *PBG@ Eurographics*, 2:091–100, 2007.

[162] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Ng. Convolutional-recursive deep learning for 3D object classification. *Advances in Neural Information Processing Systems*, 25, 2012.

[163] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *IEEE International Conference on Computer Vision*, pages 945–953, 2015.

[164] Markus Suchi, Timothy Patten, David Fischinger, and Markus Vincze. EasyLabel: A semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets. In *IEEE International Conference on Robotics and Automation*, pages 6678–6684, 2019.

[165] Shiying Sun, Ning An, Xiaoguang Zhao, and Min Tan. A PCA–CCA network for RGB-D object recognition. *International Journal of Advanced Robotic Systems*, 15(1):1729881417752820, 2018.

[166] Sudharshan Suresh, Zilin Si, Joshua G Mangelson, Wenzhen Yuan, and Michael Kaess. ShapeMap 3-D: Efficient shape mapping through dense touch and vision. In *IEEE International Conference on Robotics and Automation*, pages 7073–7080, 2022.

[167] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[168] Rajat Talak, Lisa R Peng, and Luca Carlone. Certifiable object pose estimation: Foundations, learning models, and self-training. *IEEE Transactions on Robotics*, 2023.

[169] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[170] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In *IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[171] Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Anibal Medina-Mardones, Alberto Dassatti, and Kathryn Hess. giotto-tda:

A topological data analysis toolkit for machine learning and data exploration. *arXiv preprint arXiv:2004.02551*, 2020.

[172] Tensorflow. Tensorflow models.

[173] The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2015.

[174] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pages 0–0, 2019.

[175] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. In *IEEE International Conference on Computer Vision*, pages 10860–10869, 2021.

[176] Renata Turkeš, Jannes Nys, Tim Verdonck, and Steven Latré. Noise robustness of persistent homology on greyscale images, across filtrations and signatures. *PloS one*, 16(9):e0257215, 2021.

[177] Katharine Turner, Sayan Mukherjee, and Doug M Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, 2014.

[178] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *European Conference on Computer Vision*, pages 589–600, 2006.

[179] Stephen Tyree, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Jeffrey Smith, and Stan Birchfield. 6-DoF pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 13081–13088, 2022.

[180] Georgios Tziafas and Hamidreza Kasaei. Early or late fusion matters: Efficient RGB-D fusion in vision transformers for 3D object recognition. *arXiv preprint arXiv:2210.00843*, 2023.

[181] Tzutalin. LabelImg, 2015.

[182] Hendrik Jacob van Veen, Nathaniel Saul, David Eargle, and Sam W. Mangham. Kepler mapper: A flexible python implementation of the mapper algorithm. *Journal of Open Source Software*, 4(42):1315, 2019.

[183] Hendrik Jacob van Veen, Nathaniel Saul, David Eargle, and Sam W. Mangham. Kepler Mapper: A flexible Python implementation of the Mapper algorithm, October 2020.

[184] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Open-set recognition: A good closed-set classifier is all you need. *arXiv preprint arXiv:2110.06207*, 2021.

[185] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I, 2001.

[186] Hubert Wagner, Chao Chen, and Erald Vuçini. Efficient computation of persistent homology for cubical data. In *Topological Methods in Data Analysis and Visualization II*, pages 91–106. Springer, 2012.

[187] Anran Wang, Jianfei Cai, Jiwen Lu, and Tat-Jen Cham. MMSS: Multi-modal sharable and specific feature learning for RGB-D object recognition. In *IEEE International Conference on Computer Vision*, pages 1125–1133, 2015.

[188] Anran Wang, Jiwen Lu, Jianfei Cai, Tat-Jen Cham, and Gang Wang. Large-margin multi-modal deep learning for RGB-D object recognition. *IEEE Transactions on Multimedia*, 17(11):1887–1898, 2015.

[189] Tao Wang, Xiaopeng Zhang, Li Yuan, and Jiashi Feng. Few-shot adaptive Faster R-CNN. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7173–7182, 2019.

[190] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019.

[191] Jamie Ward. *The student's guide to cognitive neuroscience*. Psychology Press, 2015.

[192] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1544–1553, 2016.

[193] Wkentaro. Github - wkentaro/labelme, 2019.

[194] Walter Wohlkinger and Markus Vincze. Ensemble of shape functions for 3D object classification. In *IEEE International Conference on Robotics and Biomimetics*, pages 2987–2992, 2011.

[195] Matthew Wright. Visualizing multi-dimensional persistent homology. Industrial and Applied Mathematics Seminar, University of Oxford, Oxford, UK, 2014.

[196] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[197] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics*, 37(5):1343–1359, 2021.

[198] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[199] Bo Xiong, Suyog Dutt Jain, and Kristen Grauman. Pixel objectness: learning to segment generic objects automatically in images and videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(11):2677–2692, 2018.

[200] Chang-Dong Xu, Xing-Ran Zhao, Xin Jin, and Xiu-Shen Wei. Exploring categorical regularization for domain adaptive object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11724–11733, 2020.

[201] Minghao Xu, Hang Wang, Bingbing Ni, Qi Tian, and Wenjun Zhang. Cross-domain detection via graph-induced prototype alignment. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12355–12364, 2020.

[202] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. PAConv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021.

[203] Zheng Xu, Wen Li, Li Niu, and Dong Xu. Exploiting low-rank structure from latent domains for domain generalization. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014.

[204] Guang-Zhong Yang, Jim Bellingham, Pierre E Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of Science Robotics. *Science Robotics*, 3(14):eaar7650, 2018.

[205] Fuxun Yu, Di Wang, Yinpeng Chen, Nikolaos Karianakis, Tong Shen, Pei Yu, Dimitrios Lymberopoulos, Sidi Lu, Weisong Shi, and Xiang Chen. Unsupervised domain adaptation for object detection via cross-domain semi-supervised learning. *arXiv preprint arXiv:1911.07158*, 2019.

[206] Hasan FM Zaki, Faisal Shafait, and Ajmal Mian. Convolutional hypercube pyramid for accurate RGB-D object category and instance recognition. In *IEEE International Conference on Robotics and Automation*, pages 1685–1692, 2016.

[207] Jiaming Zhang, Huayao Liu, Kailun Yang, Xinxin Hu, Ruiping Liu, and Rainer Stiefelhagen. CMX: Cross-modal fusion for RGB-X semantic segmentation with transformers. *arXiv preprint arXiv:2203.04838*, 2022.

[208] Xingxuan Zhang, Zekai Xu, Renzhe Xu, Jiashuo Liu, Peng Cui, Weitao Wan, Chong Sun, and Chen Li. Towards domain generalization in object detection. *arXiv preprint arXiv:2203.14387*, 2022.

[209] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *IEEE International Conference on Computer Vision*, pages 16259–16268, 2021.

[210] Feng Zhou, Yong Hu, and Xukun Shen. MSANet: multimodal self-augmentation and adversarial network for RGB-D object recognition. *The Visual Computer*, 35:1583–1594, 2019.

[211] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[212] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[213] Xinge Zhu, Jiangmiao Pang, Ceyuan Yang, Jianping Shi, and Dahua Lin. Adapting object detectors via selective cross-domain alignment. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 687–696, 2019.

[214] Chenfan Zhuang, Xintong Han, Weilin Huang, and Matthew Scott. iFAN: Image-instance full alignment networks for adaptive object detection. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 13122–13129, 2020.

[215] Afra Zomorodian. Topological data analysis. *Advances in Applied and Computational Topology*, 70:1–39, 2012.