Object Recognition with Progressive Refinement for Collaborative Robots Task Allocation

Wenbo Wu

Thesis submitted in partial fulfillment of the requirements for

Master of Applied Science

in

Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

With the rapid development of deep learning techniques, the application of Convolutional Neural Network (CNN) has benefited the task of target object recognition. Several state-of-the-art object detectors have achieved excellent performance on the precision for object recognition. When it comes to applying the detection results for the real world application of collaborative robots, the reliability and robustness of the target object detection stage is essential to support efficient task allocation. In this work, collaborative robots task allocation is based on the assumption that each individual robotic agent possesses specialized capabilities to be matched with detected targets representing tasks to be performed in the surrounding environment which impose specific requirements. The goal is to reach a specialized labor distribution among the individual robots based on best matching their specialized capabilities with the corresponding requirements imposed by the tasks. In order to further improve task recognition with convolutional neural networks in the context of robotic task allocation, this thesis proposes an innovative approach for progressively refining the target detection process by taking advantage of the fact that additional images can be collected by mobile cameras installed on robotic vehicles. The proposed methodology combines a CNN-based object detection module with a refinement module. For the detection module, a two-stage object detector, Mask RCNN, for which some adaptations on region proposal generation are introduced, and a one-stage object detector, YOLO, are experimentally investigated in the context considered. The generated recognition scores serve as input for the refinement module. In the latter, the current detection result is considered as the a priori evidence to enhance the next detection for the same target with the goal to iteratively improve the target recognition scores. Both the Bayesian method and the Dempster-Shafer theory are experimentally investigated to achieve the data fusion process involved in the refinement process. The experimental validation is conducted on indoor search-and-rescue (SAR) scenarios and the results presented in this work demonstrate the feasibility and reliability of the proposed progressive refinement framework, especially when the combination of adapted Mask RCNN and D-S theory data fusion is exploited.

Acknowledgements

I would like to thank my academic supervisor, Professor Pierre Payeur. Thank him for supervising my research and providing me with inspiration, motivations and support throughout my study and research.

I would also like to thank all of my colleagues in the SMART research team for their help and support, especially, Victor Pereira Piesiecki for helping with the data collection and annotation, Omar Al-Buraiki, Matthew Ross, and Shengsong Yang for their valuable suggestions during the research and development of the related publications.

In addition, I would like to thank my friends Chaokun Yang, Nan Liu, Rui Sun, and Xiafei Yu for their company and support during my life in Ottawa.

Finally, I would like to express my sincerest gratitude to my parents. Thank them for their unconditional love and encouragement, I would never be who I am without their love and encouragement.

Table of Contents

List of Figures

List of Tables

Acronyms

3DVP	3D Voxel Pattern
AP	Average Precision
AV	Autonomous Vehicle
bbox	Bounding box
BPA	Basic Probability Assignment
CNN	Convolutional Neural Network
CPU	Central Processing Unit
D-S theory	Dempster-Shafer theory
DeconvNet	Deconvolutional Neural Network
DPM	Deformable Part-based Model
DRC	Dempster's Rule of Combination
ELU	Exponential Linear Unit
FCN	Fully Convolutional Network
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Networks
FPS	Frames Per Second
GB	Gigabyte
GCP	Google Cloud Platform
GPU	Graphics Processing Unit
HDD	Hard Disk Drives
HOG	Histogram of Oriented Gradients
ILSVRC	The ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
mAP	Mean Average Precision
MOT	Multi-object tracking
P	Precision
PCL	Point Cloud
PNG	Portable Network Graphics
PreLU	Parametric Rectified Linear Unit
R	Recall
RCNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
RFCN	Region-based Fully Convolutional Networks
RoI	Region of Interest
RoIAlign	Region of Interest Feature Alignment

RoIPool	Region of Interest Feature Pooling operation
RPN	Region Proposal Network
SAR	Search-and-rescue
SENet	Squeeze and Excitation Networks
SGD	Stochastic Gradient Descent
SORT	Simple and Online Tracking algorithm
SPPNet	Spatial Pyramid Pooling Networks
SS	Selective search
SSD	Single Shot Multibox detector
SSH	Secure Shell
SVM	Support Vector Machine
TP	True Positive
UGV	Unmanned Ground Vehicle
VM	Virtual Machine
VOC	Visual Object Classes
YOLO	You Only Look Once
YOLOv2	The second version of YOLO
YOLOv3	The third version of YOLO

Chapter 1 Introduction

1.1 Motivation and Research Problem

With the rapid development of deep learning techniques, especially the application of Convolutional Neural Networks (CNNs) to learn feature representations, a remarkable breakthrough and unprecedented performance in visual target object recognition was achieved in the recent years.

Object recognition consists of classifying objects present in an image into predefined categories, while object detection is defined as recognition augmented with localization, which generates a recognized class of an object instance associated with a bounding box or segmentation mask that defines the localization of the instance in the original image. As a cornerstone of image understanding and computer vision, target object recognition forms the foundations for solving complex or high level vision tasks, such as object segmentation, object tracking, activity recognition, etc. Besides, target object detection supports a wide range of applications, including robot vision, autonomous driving, intelligent video surveillance, collaborative robots task allocation, etc.

When a target object detector is capable of recognizing the class label of present objects with confidence scores while also determining the localization of present objects with bounding boxes or segmentation masks, it can achieve excellent performance on target object detection. Such target object detectors are generally grouped into two genres: two-stage and one-stage object detectors. In both cases, a target object detection pipeline consists of a backbone network for extracting image features and a detection subnetwork for classifying the detected targets and generating the corresponding bounding boxes. As the quality of feature representations is critical for object detection, researchers have made efforts to improve the performance on CNN-based backbone networks to advance the accuracy of recognition and detection. However, training deep CNNs requires a lot of Graphics Processing Units (GPUs) for parallel computational supports. Beyond researches on backbone network improvement, there are several works focusing on improving

target object detection accuracy by bounding box refinement for single object detectors and on detection results fusion for multiple object detectors.

As the performance of object detectors continuously improves, how can we utilize the output of these detectors to support other tasks, such as collaborative robots task allocation? In the context of this research, a specific task allocation process is considered for specialized unmanned robotic agents evolving as part of a collaborative multi-robot swarm. It is assumed that each individual robotic agent possesses specialized capabilities, and that target objects representing the tasks to be performed can be visually observed in the surrounding environment while each imposing specific task requirements. The definition of specialization for the robotic agents leverages the embedded hardware and software characteristics of each agent and attempt to best match those physical capabilities with the estimation of requirements imposed by specific target objects. As a result, an advanced form of specialized labor division emerges in the swarm, which distributes the labor among the individual agents based on best matching the tasks' specific requirements to each robot's specialized capabilities. Vision sensors are embedded on robotic agents and make it possible to detect and recognize predefined classes of target objects in the robotic swarm environment in order to feed the task allocator and direct specific agents toward specific targets. Therefore, reliable target object detection plays a critical role as an intelligent input layer to make efficient and responsive task allocation for specialized unmanned robotic systems possible.

Inspired by the concept of data fusion, this thesis proposes an iterative and progressive refinement method for target object detection. The objective is to iteratively utilize the confidence scores generated by target object detectors, and progressively refine and reinforce the confidence on the class of detected target objects through an iterative process that efficiently leverages previous detections and successively acquired visual data. The proposed framework serves as the task detection stage necessary to perform efficient task allocation in collaborative robotic systems. On the other hand, the bounding boxes (with or without mask segmentation) information produced by the detector can also support the navigation of robots closer to a target object where an embedded camera captures a new image, which then helps further refine the recognition and localization of a target. Considerations on robots navigation and collaborative robots task allocation processes are beyond the scope of this thesis, which focuses on the target object recognition and detection layers.

For the sake of development and experimental validation, the objects considered of interest are associated with indoor search-and-rescue (SAR) operations to be executed by one or some robotic agents working in collaboration under a robot-task allocation framework. Five classes of target objects that can realistically be imaged from a camera mounted over unmanned ground vehicles (UGVs) are considered, and represent the following task situations: a person to be rescued, a door to be opened, stairs to be climbed, posted signs or floor maps to be read to support robots navigation, and a fire to be extinguished. Besides, we only consider static scenes in this work. That is, target objects are not expected to move in between frames, and only the camera mounted on the robot can move toward the target. Therefore, we consider a sequence of images captured from a mobile camera on a static target to validate the performance of the proposed progressive refinement strategy in the context of collaborative robots coordination.

1.2 Objectives

The main objectives of this thesis are:

- Explore and evaluate the feasibility of CNN-based two-stage and one-stage target object detectors to recognize and localize target objects to support collaborative mobile robotic systems, and propose some adaptations to the classical frameworks to improve performance in the given context of application.
- Build an original dataset that is relevant to indoor search-and-rescue (SAR) scenarios for experimental validation, and conduct comparative experiments to evaluate the recognition and detection performance of state-of-the-art two-stage and one-stage target object detectors in the context of autonomous robotics.
- Develop an original method for progressively refining the confidence on detected target objects with the goal to reinforce response capability and reliability in collaborative robotic systems.
- Experimentally test and validate the recognition performance of the proposed method on real visual data.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 summarizes the literature review on CNN-based milestone target object detection methods in the deep learning era. It discusses publicly available datasets of immediate interest in the context considered in this thesis, as well as standardized evaluation metrics in machine learning. It also covers some real-world applications for object detection.
- Chapter 3 introduces two-stage and one-stage target object detectors, and conducts comparative experiments on the built dataset to validate their feasibility in the context of collaborative mobile robotic systems.
- Chapter 4 presents the proposed progressive refinement target object recognition framework and validates its performance.
- Chapter 5 concludes the thesis with a summary of the work, the contributions made, and proposes possible directions for future research.

Chapter 2 Literature Review

Deep Convolutional Neural Networks (CNNs) has shown exemplary performance in several applications related to computer and machine vision. Application areas of CNN include Image Classification and Segmentation, Object Detection, Active Recognition, Video Processing, Natural Language Processing, etc. The use of multiple feature extraction stages that can automatically learn representations from the data is the main reason for deep CNN to reach powerful learning ability. Availability of a large amount of data and advancements in the hardware technology have accelerated the research in CNNs. Recently, very efficient deep CNN architectures have been reported that benefit object recognition research as well.

In this chapter, we will review the basic CNN components and backbone architecture evolution in Section 2.1. In Section 2.2, CNN-based milestone object detectors will be summarized, and in Section 2.3 and Section 2.4, popular public datasets and evaluation metrics will be reviewed, respectively. Then, refinements in target object detection and real-world applications including relevant work on collaborative robots task allocation, which forms the context of application in this thesis, will be surveyed in Sections 2.5 and 2.6, respectively. Finally, Section 2.7 proposes a summary of the literature review.

2.1 Convolutional neural networks (CNN)

In recent years, deep CNNs have played a central role in many computer vision tasks. As the accuracy of a detector depends heavily on its feature extraction network, this section will review the basic CNN components and the representative CNN architectures that are popularly used as backbone networks in the object detection pipeline, e.g. the ResNet, VGG, etc.

2.1.1 Basic CNN components

The topology of a CNN is a combination of convolutional layers, non-linear processing units, and subsampling layers. CNNs are feedforward multilayered hierarchical networks, where each layer, using a bank of convolutional kernels, performs multiple transformations [1]. Convolution

operation helps in extracting useful features from locally correlated data. The output of the convolutional kernels is then assigned to a non-linear processing unit (activation function), which not only helps in learning abstractions but also embeds non-linearity in the feature space. This nonlinearity generates different patterns of activation for different responses and thus facilitates the learning of semantic differences in images. The output of the non-linear activation function is usually followed by subsampling, which helps in summarizing the results and also makes the input invariant to geometrical distortions [1][2]. A CNN, which provides automatic feature extraction ability, reduces the need for a separate feature extractor [3]. Thus, a CNN can learn good internal representations from raw pixels without exhaustive processing. Important attributes of a CNN are its hierarchical learning, automatic feature extraction, multi-tasking, and weight sharing [4].

A typical CNN architecture generally consists of alternate layers of convolution and pooling followed by one or more fully connected layers at the end. In some cases, a fully connected layer is replaced with a global average pooling layer. In addition to different mapping functions, different regulatory units such as batch normalization and dropout are also incorporated to optimize CNN performance. Figure 2.1 presents the LeNet-5 [5] architecture, which straightforwardly illustrates the basic CNN architecture.

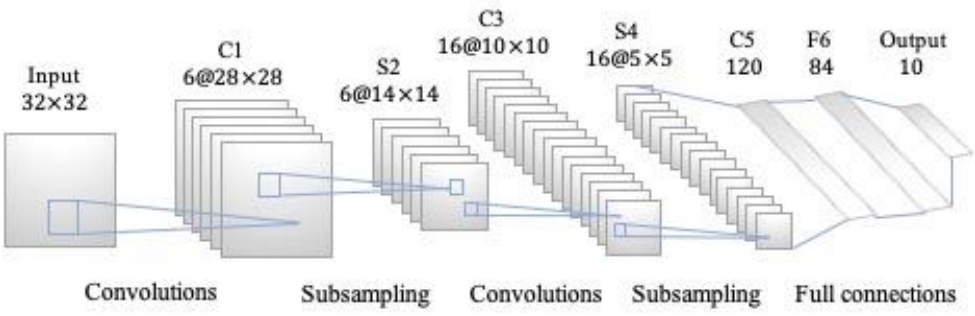


Figure 2.1: Basic CNN architecture (derived from [5]).

- **Convolutional Layer**

A convolutional layer is composed of a set of convolutional kernels, which work by dividing the image into small slices commonly known as receptive fields, through convolution operation. A kernel with a specific set of weights convolves with the images by multiplying its elements with the corresponding elements of the image receptive field, generating various feature maps. There

are three main advantages of the convolution operation: 1) the weight sharing mechanism in the same feature map reduces the number of parameters; 2) local connectivity learns correlations among neighboring pixels; and 3) invariance to the location of the object.

- **Pooling Layer**

A pooling layer generally follows a convolutional layer and can be used to reduce the dimensions of feature maps and network parameters. Once features are extracted, their exact location becomes less important since their approximate position relative to others is preserved. The use of pooling operation helps to extract a combination of features, which are invariant to translational shifts and small distortions, because its computation takes neighboring pixels into account. It sums up similar information in the neighborhood of the receptive field and outputs the dominant response within this local region. Reduction in the size of the feature map to an invariant feature set not only regulates the complexity of the network but also helps in increasing the generalization by reducing overfitting. Different types of pooling formulations such as max pooling and average pooling [6], spatial pyramid pooling [7], etc., are used in CNN.

- **Activation Function**

The activation function can be regarded as a decision function that helps in learning complex patterns. Selection of an appropriate activation function can accelerate the learning process. Different activation functions such as sigmoid, tanh, rectified linear unit (ReLU), and variants of ReLU such as leaky ReLU, exponential linear unit (ELU), and parametric rectified linear unit (PreLU) are used to inculcate non-linear combinations of features [8]. However, ReLU and its variants are preferred as they help in overcoming the vanishing gradient problem [9]. The vanishing gradient problem is an issue that sometimes arises when training machine learning algorithm through gradient descent. The key point is that the calculated partial derivatives are used to compute the gradient as the layer goes deeper into the network. Since the gradients control how much the network learns during training, if the gradients are very small or zero, then little to no training can take place, leading to poor predictive performance.

- **Batch Normalization**

Batch normalization serves for addressing the issues caused by internal covariance shift within feature maps. The internal covariance shift is a change in the distribution of hidden units' values, which slows down the convergence and requires careful initialization of parameters. The advantage of batch normalization is that it unifies the distribution of feature map values by setting

them to zero mean and unit variance. Besides, it smoothens the flow of gradient and acts as a regulating factor, which helps in improving generalization of the network [10].

- **Dropout**

Dropout introduces regularization within the network, which ultimately improves generalization by randomly skipping some units or connections with a certain probability. This random dropping of some connections or units produces several thinned network architectures, and finally one representative network is selected with small weights. This selected architecture is then considered as an approximation of all proposed networks [11].

- **Fully Connected Layer**

A fully connected layer is mostly used at the end of the network following the last pooling layer for classification purposes. Unlike convolution and pooling, it is a global operation. It takes input from feature extraction stages and globally analyses the output of all preceding layers. It contains about 90% of the parameters in a CNN. It forms a non-linear combination of the selected features, which are used for the classification of data.

2.1.2 Evolution of CNN backbone architectures

In the deep learning era, various improvements in CNN learning methodology and architecture were performed to make CNN scalable to large, heterogeneous, complex and multiclass problems. In this section, some of the important CNN backbone architectures will be introduced. Figure 2.2 shows the evolution of milestone CNN architectures. Besides, Table 2.1 provides a summary of these CNN architectures in terms of their contributions and parameters.

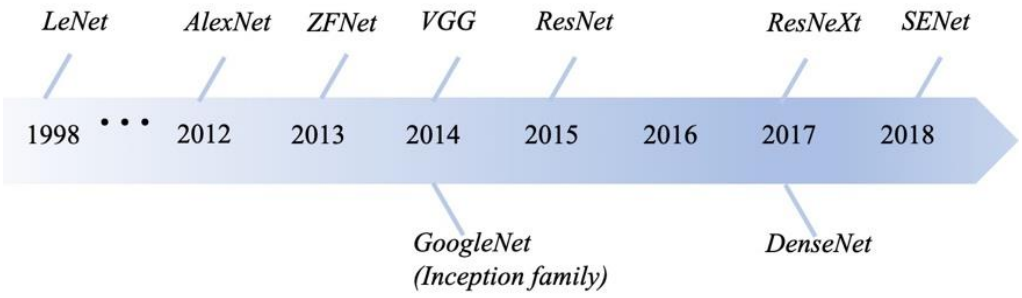


Figure 2.2: Evolution of CNN architectures.

- **LeNet**

The first CNN architecture that showed state-of-the-art performance on hand-written digit recognition task was LeNet [12], proposed by LeCun *et al*. LeNet is a feed-forward neural network that consists of five alternating convolutional and pooling layers, followed by two fully connected layers, so it is named as LeNet-5 [5]. LeNet-5 exploited the underlying basis of image representation, which involves that neighboring pixels are correlated to each other and are distributed across the entire image. Therefore, it gave an insight that convolution with learnable parameters is an effective way to extract similar features at multiple locations with few parameters, which changed the conventional view of training where each pixel was considered as a separate input feature from its neighborhood and ignored the correlation among them. The contribution of LeNet-5 is that it proved that a CNN not only reduced the number of parameters but was able to automatically learn features from raw pixels.

- **AlexNet**

Krizhevesky *et al*. proposed AlexNet [13], which was considered as the first CNN for image classification tasks. Though LeNet began the history of deep CNNs, it was limited to hand-written digit recognition and did not perform well on all classes of images. AlexNet represented a breakthrough for image classification and recognition tasks, mainly because its learning capacity was enhanced by making the CNN deeper and by applying a number of parameter optimization strategies. Compared with LeNet-5, the depth of AlexNet was extended from 5 (LeNet) to 8 layers to make CNN applicable for diverse categories of images, and it used large size filters (11 x 11 and 5 x 5) at the initial layers. Besides, a rectified linear unit (ReLU) was employed as a non-saturating activation function to improve the convergence rate by alleviating the problem of vanishing gradient to some extent [14][15]. Overlapping subsampling and local response normalization were also applied to improve the generalization by reducing overfitting. In addition to this, AlexNet was the first CNN that trained on graphical processing units (GPUs).

- **ZFNet**

In 2013, Zeiler and Fergus proposed a multilayer Deconvolutional Neural Network (DeconvNet), which got famous as ZFNet [16]. ZFNet was developed to quantitatively visualize network performance. DeconvNet works in the same way as the forward pass CNN but reverses the order of convolutional and pooling operation. This reverse mapping projects the output of a convolutional layer back to visually perceptible image patterns and consequently gives the neuron-

level interpretation of the internal feature representation learned at each layer [17]. Zeiler and Fergus maximized the learning of CNN by reducing both the filter size and stride to retain the maximum number of features in the first two convolutional layers. This readjustment in CNN topology resulted in performance improvement, which suggested that features visualization can be used for identification of design shortcomings and for timely adjustment of parameters.

- **VGG**

In 2014, Simonyan and Zisserman proposed an effective CNN architecture named VGG [18]. It was 19 layers deep compared to AlexNet and ZFNet, both of which were 8 layers deep, to simulate the relation of depth with the representational capacity of the network [13][16]. VGG replaced the 11x11 and 5x5 filters with a stack of 3x3 filters and experimentally demonstrated that concurrent placement of small size (3x3) filters can induce the effect of the large size filter (5x5 and 7x7). Use of the small size filters provides an additional benefit for low computational complexity by reducing the number of parameters for each small filter. VGG showed good results both for image classification and localization problems. VGG was in 2nd place in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition but got fame due to its simplicity, homogeneous topology, and increased depth. The main limitation associated with VGG was the use of 138 million parameters in total, which makes it computationally expensive and difficult to deploy on low resource systems.

- **GoogleNet**

GoogleNet, also known as Inception [19][20][21][10] is a broad family of CNN models proposed by Google Inc since 2014. It was proposed to achieve high accuracy with a reduced computational cost. GoogleNet introduced the inception block in CNNs. This block encapsulates filters of different sizes (1x1, 3x3, and 5x5) to extract spatial information at different scales including both fine and coarse grain levels. The exploitation of the idea of split, transform and merge by GoogleNet, helped in addressing the problem of learning diverse types of variations present in the same category of images but having different resolutions. GoogleNet regulates the computations by adding a bottleneck layer of 1x1 convolution filter, before employing large size kernels. Besides, the sparse connections (not all the output feature maps are connected to all the input feature maps) helped in overcoming the problem of redundant information and reduced cost by omitting feature maps that were not relevant. The main contribution of the Inception family is the introduction of factorizing convolution and batch normalization. These parameter tunings caused a significant

decrease in the number of parameters from 138 million, which VGG learns, to 4 million parameters. However, the main drawback of GoogleNet was its heterogeneous topology that needs to be customized from module to module. Besides, the representation bottleneck drastically reduces the feature space in the next layer and therefore sometimes may lead to the loss of useful information.

- **ResNet**

In 2015, He *et al*. proposed ResNet [22], which introduced residual learning in CNNs and an optimal methodology for the training of deep networks. The residual block of ResNet was 20 and 8 times deeper than AlexNet and VGG, respectively. However it showed less computational complexity than previously proposed networks [13][18]. ResNet with 152-layers won the 2015-ILSVRC competition. Such good performance of ResNet on image recognition and localization tasks showed that depth is of central importance for many visual recognition tasks. The drawback of ResNet was that it explicitly preserves information through additive identity transformations due to which many layers may contribute very little or no information.

- **DenseNet**

In 2017, DenseNet [23] was proposed by Huang *et al*. to solve the vanishing gradient problem. DenseNet introduced a densely connected block, which connected each layer to every other layer in a feed-forward fashion, thus feature maps of all preceding layers were used as inputs into all subsequent layers. Since DenseNet concatenates previous layers' features instead of adding them, the network gains the potential to explicitly differentiate between information that is added to the network and information that is preserved.

- **ResNeXt**

In the same year, Xie *et al*. proposed ResNeXt [24] also known as Aggregated Residual Transform Network, which is an improvement over the Inception Network. The Inception network has not only improved the learning capability of conventional CNNs but also made a network resources effective. However, due to the use of diverse spatial embeddings (such as the use of 3x3, 5x5 and 1x1 filters) in the transformation branch, each layer needs to be customized separately. ResNeXt utilized the deep homogeneous topology of VGG and simplified the GoogleNet architecture by fixing spatial resolution to 3x3 filters within the split, transform and merge block. Whereas, it used residual learning to improve the convergence of deep and wide networks [18][19][22]. The main contribution of ResNeXt is that it introduced cardinality [19] and it showed that increase in

cardinality significantly improves the performance. Besides, the complexity of ResNeXt was regulated by applying low embeddings (1x1 filters) before 3x3 convolution.

- **SENet**

In 2018, Hu *et al*. published their work on Squeeze and Excitation Networks (SENet) [25], which won 1st place in the ILSVRC 2017 classification competition. The main contribution is the integration of global pooling shuffling to learn channel-wise importance of the feature map.

Table 2.1: Summary of milestone CNNs.

CNN Architecture	**Main Contribution**	**Number of Layers**	**Number of Parameters ($\times 10^6$)**
LeNet [5]	First popular CNN architecture.	5	0.06
AlexNet [13]	Deeper and wider than the LeNet; Uses Relu, dropout and overlap Pooling; GPUs NVIDIA GTX 580; Visualization.	8	60
ZFNet [16]	Visualization of intermediate layers.	8	60
VGG [18]	Homogeneous topology; Uses small size kernels.	19	138
GoogleNet [19]	Introduces block concept; Split transform and merge idea.	22	4
ResNet [22]	Residual learning; Identity mapping based skip connections.	152 110	25.6 1.7
ResNeXt [24]	Cardinality; Homogeneous topology; Grouped convolution.	101	68.1
DenseNet [23]	Cross-layer information flow.	190 250	25.6 15.3
SENet [25]	Integration of global pooling and shuffling.	152	27.5

2.2 Milestone object detectors

Over the past two decades, it is widely accepted that the progress on object detection in computer vision, which benefits from the performance on object recognition, has generally gone through two historical periods: a "traditional object detection" period (before 2014), and a "deep learning based object detection" period (after 2014), as shown in Figure 2.3. The milestone object detectors that supported this evolution will be introduced in this section.

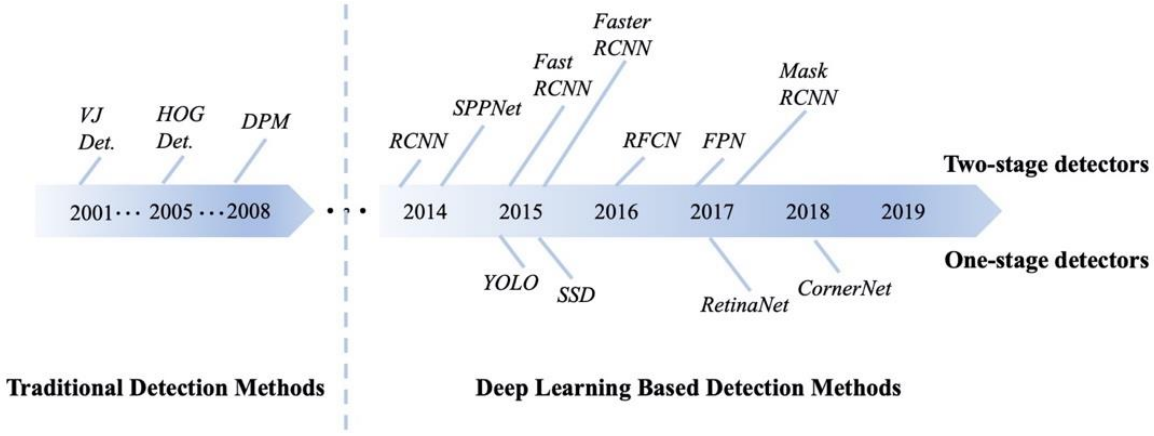


Figure 2.3: Milestone object detectors evolution roadmap.

2.2.1 Traditional object detectors

Most of the early object detection algorithms that leveraged images and computer vision were built based on handcrafted features. The most famous work in the traditional detectors era were the Viola-Jones (VJ) detector [26][27], the Histogram of Oriented Gradients (HOG) feature descriptor [28], and the Deformable Part-based Model (DPM) [29]. Due to the lack of effective image representation at that time, people had no choice but to design sophisticated feature representations, and a variety of speed up strategies to optimize the usage of limited computing resources.

In 2001, Viola and Jones achieved real-time detection of human faces for the first time without any constraints (e.g., skin color segmentation). The detection algorithm was referred to as the "Viola-Jones (VJ) detector" [26][27]. The VJ detector follows a most straightforward way of detection, i.e., a sliding window that is to go through all possible locations and scales in an image to see if any window contains a human face. Since the computation needed behind that process

was far beyond the power of computers at that time, the VJ detector introduced a cascade detection paradigm to address this problem.

Dalal and Triggs proposed the Histogram of Oriented Gradients (HOG) feature descriptor in 2005 [28]. HOG can be considered as an important improvement that provided a scale-invariant feature transform [30] and shape contexts [31] at this time. The HOG descriptor was designed to be computed on the dense grid of uniformly spaced cells and with overlapping local contrast normalization for improving accuracy, in order to balance the feature invariance (including translation, scale, illumination, etc.) and nonlinearity (on discriminating different object categories). For the detection of objects with different size, the HOG detector rescales the input image multiple times while keeping the size of the detecting window unchanged.

The Deformable Part-based Model (DPM) was the peak of the traditional object detection methods. The method was proposed by Felzenszwalb *et al*. [29] in 2008 as an extension of the HOG detector, and then a variety of improvements have been made by Girshick *et al*. [32]. The DPM follows the detection philosophy of "divide and conquer", where the training can be simply considered as the learning of a proper way of decomposing an object, and the inference can be considered as an ensemble of detections on different parts. This part of the work, also known as "star-model" was completed by Felzenszwalb *et al*. [29]. Later, in order to deal with objects in the real world under more significant variations, Girshick *et al*. further extended the star-model to the "mixture-models" [33][32]. A typical DPM detector consists of a root-filter and a number of part-filters. Instead of manually specifying the configurations of the part filters (e.g., size and location), a weakly supervised learning method is developed in DPM where all configurations of part filters can be learned automatically as latent variables. Although object detectors today have far surpassed DPM in terms of detection accuracy and detection speed, many of them are still deeply influenced by its valuable insights, e.g., mixture models, bounding box regression, etc.

In the following section, we mainly focus on the CNN-based object detectors that form the deep learning era. The datasets mentioned in the following section, such as ImageNet, PASCAL-VOC07, MS-COCO, and the performance metric, mean Average Precision (mAP), will be detailed in Section 2.3 and Section 2.4.

2.2.2 CNN-based object detectors

As a deep convolutional neural network is able to learn robust and high-level feature representations of an image, it is capable of being adapted to object detection tasks. Girshick *et al.* made a breakthrough in that direction in 2014 by proposing Regions with CNN features (RCNN) for object detection [34][35]. Since then, object detection started to evolve at an unprecedented speed. With the development of deep learning techniques, object detection can be grouped into two genres: “two-stage detection” and “one-stage detection”, where the former frames the detection as a “coarse-to-fine” process while the latter frames it as a “complete in one step” procedure. Figure 2.4 shows an overview of CNN-based milestone object detectors and Table 2.2 provides a summary of the characteristics for a number of approaches.

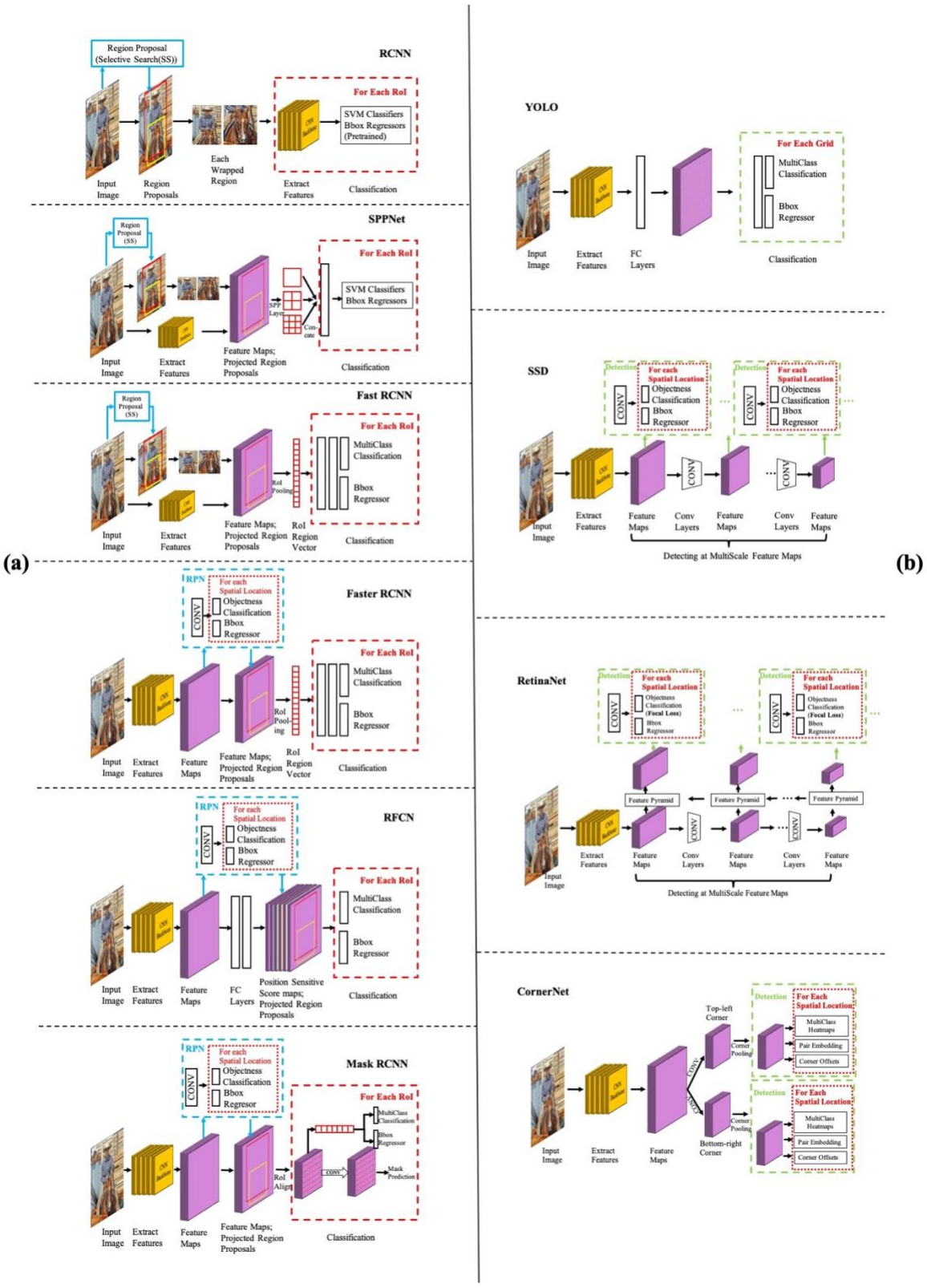


Figure 2.4: Overview of CNN-based milestone object detection architectures: (a) Two-stage detectors; (b) One-stage detectors.

2.2.2.1 CNN based two-stage object detectors

- **RCNN**

The original mechanism of Region-based Convolutional Neural Network (RCNN) [34][35], introduced in 2014, first extracts a set of object proposals via selective search (SS) [36]. Then each proposal is rescaled to a fixed size image and fed into a CNN model trained on ImageNet dataset (e.g., AlexNet [13]) to extract features. Finally, linear support vector machine (SVM) classifiers are used to predict the presence of an object within each region and to recognize object categories. RCNN yields a significant performance boost on the PASCAL-VOC07 dataset. Although RCNN has made great progress, it shows obvious drawbacks in redundant feature computation on a large number of overlapped proposals (over 2000 boxes from one image), which leads to an extremely slow detection speed (14s per image with GPU).

- **SPPNet**

Later in the same year, Spatial Pyramid Pooling Networks (SPPNet) [7] were proposed by He *et al*. to overcome this problem. The main contribution of SPPNet is the introduction of a Spatial Pyramid Pooling (SPP) layer on the top of the last convolutional layers. It removes the fixed-size constraint of the network, namely it enables a CNN to generate a fixed-length representation regardless of the size of image/region of interest. In comparison, the previous CNNs required a fixed-size input, e.g., a 224×224 image for AlexNet. When using SPPNet for object detection, the feature maps can be computed from the entire image only once, and then fixed-length representations of arbitrary regions can be generated for training the detectors, which avoids repeatedly computing the convolutional features. The detection speed of SPPNet is more than 20 times faster than RCNN without sacrificing any detection accuracy. Although SPPNet has efficiently improved the detection speed, there are still some drawbacks. First, the training is still multi-stage. Second, SPPNet only fine-tunes its fully connected layers while simply ignoring all previous layers.

- **Fast RCNN**

In 2015, Girshick proposed the Fast RCNN [37], which is a further improvement of RCNN and SPPNet. Fast RCNN made training a detector and a bounding box regressor simultaneously under the same network configuration come true. It increased detection speed up to over 200 times faster

than RCNN. Although Fast RCNN successfully integrated the advantages of RCNN and SPPNet, the detection speed is still limited by the proposal detection.

- **Faster RCNN**

In the same year, Ren *et al*. proposed the Faster RCNN detector [38], which is the first end-to-end, and the first near-real-time deep learning detector. The main contribution of Faster RCNN is the introduction of a Region Proposal Network (RPN) to generate object proposals with a CNN model. From RCNN to Faster RCNN, most individual blocks of an object detection system, e.g., proposal detection, feature extraction, bounding box regression, etc., have been gradually integrated into a unified, end-to-end learning framework. Although Faster RCNN breaks through the detection speed bottleneck of Fast RCNN, there is still computation redundancy at subsequent detection stages.

- **R-FCN**

In 2016, Dai *et al*. proposed the Region-based Fully Convolutional Networks (R-FCN) [39] to avoid running the costly region of interest (RoI)-wise subnetwork in Faster-RCNN hundreds of times, i.e. once per proposal. R-FCN proposed the idea of position sensitive feature maps. In this approach, each feature map is responsible for outputting scores for a specific part, like top left, center, bottom right, etc., of the target class. The parts are identified with RoI-Pooling cells which are distributed alongside each part of a specific feature map. Final scores are obtained by average voting on every part of the RoI from the respective filter. This implementation trick introduced some more translational variance to structures that were essentially translation invariant by construction. Translational variance in object detection can be beneficial for learning localization representations. Although this pipeline seems to be more precise, it is not always better, performance-wise, than its Faster RCNN counterpart. It improved the overall inference time speed of two-stage detectors but performed slightly worse.

- **FPN**

In 2017, Lin *et al*. proposed the Feature Pyramid Networks (FPN) [40] on the basis of Faster RCNN. Before FPN, most of the deep learning based object detectors run detection only on the top layer of a network. Although the features in deeper layers of a CNN are beneficial for category

recognition, it is not conductive to localizing objects. To this end, a top-down architecture with lateral connections is developed in FPN for building high-level semantics at all scales. Since a CNN naturally forms a feature pyramid through its forward propagation, the FPN shows great advances for detecting objects with a wide variety of scales. Using FPN in a basic Faster RCNN system, it achieves state-of-the-art single model detection results on the MS-COCO dataset. FPN has now become a basic building block of many latest detectors.

- **Mask RCNN**

Mask RCNN [41] was proposed by He *et al*. in 2017, as an improvement of Faster RCNN. Mask RCNN added a branch to Faster RCNN so it is capable of outputting a binary mask that says whether or not a given pixel is a part of an object. The extra branch is a Fully Convolutional Network (FCN) [42] on top of a CNN based feature map. Besides, Mask RCNN introduced RoI Align to address the problem of misalignment in RoI Pooling. RoI Align uses bilinear interpolation to calculate the value of four regularly sampled locations on each cell. It allows to fix to some extent the alignment between the computed features and the regions they are extracted from. Mask RCNN brought consistent improvements to all Faster RCNN baselines on the MS-COCO dataset.

2.2.2.2 CNN based one-stage object detectors

- **YOLO**

In 2015, Redmon and Farhadi proposed the first one-stage detector in the deep learning era, named You Only Look Once (YOLO) [43]. YOLO is extremely fast: a fast version of YOLO has achieved a detection speed of 155 fps on the PASCAL-VOC07 dataset with a mean Average Precision of (mAP)=52.7%, while its enhanced version has achieved 45 fps on PASCAL-VOC07 with mAP=63.4%. In the YOLO framework, the authors have completely abandoned the previous detection paradigm consisting of "proposal detection + verification". Instead, it follows a totally different philosophy: to apply a single neural network to the full image. This network divides the image into regions and simultaneously predicts bounding boxes and probabilities for each region to contain a given category of object. Later, the same authors made a series of improvements on the basis of YOLO and proposed its v2 and v3 editions [44][45], which further improve the detection accuracy while keeping a very high detection speed. In spite of its great improvement on

detection speed, YOLO suffers from a drop on the localization accuracy compared with two-stage detectors, especially for some small objects. YOLOv2, YOLOv3 and a proposed alternative detector, SSD [46], have paid attention to this problem.

- **SSD**

The Single Shot Multibox detector (SSD) [46] was proposed by Liu *et al*. in 2016, as the second one-stage detector introduced in the deep learning era. The main contribution of SSD is the introduction of multi-reference and multi-resolution detection techniques, which significantly improve the detection accuracy of one-stage detectors, especially for small objects. SSD improves both the detection speed and accuracy. The main difference between SSD and any previous detectors is that the former detects objects of different scales on different layers of the network, while the latter only run detection on their top layers.

- **RetinaNet**

In spite of their high speed and simplicity, one-stage detectors have trailed the accuracy of two-stage detectors for years. Lin *et al*. have discovered the reasons behind this and proposed RetinaNet in 2017 [47]. They claimed that the extreme foreground-background class imbalance encountered during training of dense detectors was the central cause. To this end, a new loss function named "focal loss" has been introduced in RetinaNet by reshaping the standard cross entropy loss so that the detector will put more focus on hard, misclassified examples during training. Focal loss enables one-stage detectors to achieve comparable accuracy to two-stage detectors while maintaining very high detection speed.

- **CornerNet**

In 2018, Law and Deng proposed CornerNet [48], which offered a new "anchor free" approach for object detection by predicting bounding boxes as paired top-left and bottom-right keypoints. It also showed that one can get rid of the prominent anchors step while gaining accuracy and precision. The fully convolutional networks were used to produce independent score heat maps for both corners for each class in addition to learning an embedding for each corner. The embedding similarities were then used for grouping them into multiple bounding boxes.

Table 2.2: Summary of Milestone Object Detectors.

Detector	Region Proposal	Backbone CNN	Input Image Size	Advantages	Disadvantages
Two-stage detectors					
RCNN [34]	SS	AlexNet	Fixed	First to integrate CNN with region proposal methods; Dramatic performance improvement over previous state-of-the-art.	Multistage pipeline of sequentially-trained; Training is expensive in space and time; Testing is slow.
SPPNet [7]	SS	ZFNet	Arbitrary	First to introduce spatial pyramid pooling into CNN architecture; Enable convolutional feature sharing; Accelerate RCNN evaluation by orders of magnitude without sacrificing performance.	Inherit disadvantages of RCNN; Does not result in much training speedup.
Fast RCNN [37]	SS	AlexNet; VGG16	Arbitrary	First to enable end-to-end detector training; Design a RoI pooling layer; Much faster and more accurate than SPPNet.	External region proposal computation is exposed as the new bottleneck; Still too slow for real-time application.
Faster RCNN [38]	RPN	ZFNet; VGG	Arbitrary	Propose RPN for generating nearly cost-free and high quality RPs instead of SS; Introduce translation invariant and multiscale anchor boxes as references in RPN; Unify RPN and Fast RCNN into a single network by sharing CONV layers.	Training is complex; Still falls short of real time.
RFCN [39]	RPN	ResNet101	Arbitrary	Fully convolutional detection framework; Design a set of position sensitive scope maps using a bank of specialized CONV layers; Faster than Faster RCNN without	Training is not a streamlined process.

				sacrificing the accuracy.	
Mask RCNN [41]	RPN	ResNet-FPN; ResNeXt-FPN	Arbitrary	A simple, flexible, and effective framework for object instance segmentation; Extends Faster RCNN by adding an extra branch for predicting an object mask in parallel with the existing branch for bbox prediction; FPN is utilized; Outstanding performance.	Falls short for real-time application.
One-stage detectors					
YOLO [43]	-	GoogleNet like	Fixed	First efficient unified detector; Drop region proposal process completely; Elegant and efficient detection framework; Significantly faster than previous detectors.	Accuracy falls far behind state-of-the-art detectors; Struggle to localize small objects.
YOLOv2 [44]	-	DarkNet	Fixed	Propose a faster DarkNet19; Use a number of existing strategies to improve both speed and accuracy; YOLO9000 can detect over 9000 object categories in real time.	Not good at detecting small objects.
SSD [46]	-	VGG16	Fixed	Efficiently combine ideas from RPN and YOLO to perform detection at multi-scale CONV layers.	Not good at detecting small objects.
RetinaNet [47]	-	ResNet-FPN	Arbitrary	Focal loss suppresses the gradients of easy negative samples instead of simply discarding them; Use feature pyramid networks to detect multi-scale objects at different levels of feature maps.	-Detection speed is not as high as other one-stage detectors.
CornerNet [48]	-	Hourglass Net [49]	Arbitrary	Anchor-free object, the goal is to predict key points of the bounding box, instead of trying to fit an object to an anchor.	Detection speed is lower than two-stage and one-stage detectors.

2.3 Datasets for object detection

2.3.1 General datasets

Public datasets play an important role as they not only allow to measure and compare the performance of object recognition research but also provide resources allowing to learn object models. This section discusses some of the main datasets that were released in the era of deep learning and that are used in recent literature on object detection. Among the most famous datasets widely used in the domain, one can consider: the PASCAL Visual Object Classes (VOC) [50][51], MS-COCO [52], ImageNet [53][54], and Open Images [55]. The four datasets form the backbone of their respective visual recognition challenges.

- **PASCAL VOC**

The PASCAL Visual Object Classes (VOC) [50][51] is the most iconic object detection dataset. It supported object recognition challenges that were held from 2005 to 2015. 20 classes of objects that are common in life emerge in the dataset (Person: person; Animal: bird, cat, cow, dog, horse, sheep; Vehicle: aeroplane, bicycle, boat, bus, car, motor-bike, train; Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor). There are two versions of PASCAL-VOC mostly used in object detection: PASCAL-VOC07 and PASCAL-VOC12, where the former consists of 5K training images and 12K annotated objects, and the latter consists of 11K training images and 27K annotated objects. It is now used as a test bed for most new algorithms. As it is quite small, some larger datasets, like ImageNet and MS-COCO, have been released.

- **MS-COCO**

MS-COCO [52] is the most challenging object detection dataset available today. Besides a bunch of annotated images, there are also 120K unlabeled images released that follow the same class distribution as the labeled images. They may be useful for semi-supervised learning on MS-COCO. Compared with ImageNet and PASCAL-VOC, it contains a lower number of object categories than ImageNet but over 4 times more than PASCAL-VOC07 and PASCAL-VOC12. Besides, the biggest progress of MS-COCO is that apart from the bounding box annotation, each object is further labeled using per-instance segmentation to aid in precise localization. In addition, MS-

COCO contains more small objects (for which the area is smaller than 1% of the image) and more densely located objects than PASCAL-VOC and ImageNet.

- **ImageNet**

ImageNet [53] contains 21841 classes of objects and more than 1 millions images with bounding box and category annotations. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [54] was organized from 2010 to 2017, and the corresponding dataset that contains a detection challenge is a subset of ImageNet [53]. The detection task in ILSVRC, in which each object instance has to be detected, involves 200 classes. In ILSVRC-17, a subset of ImageNet containing more than 470K images and 534K annotated objects, the number of images/object instances is two orders of magnitude larger than PASCAL-VOC07 and PASCAL-VOC12. It has been fixed to provide a standardized benchmark for the ILSVRC recognition challenge.

- **Open Images**

Open Images is the dataset from Open Images V4 Challenge [55], which offers the largest dataset to date for common object detection with up to 500 classes (including the familiar ones from PASCAL-VOC) on 1,743,042 images and more than 12,000,000 annotated bounding boxes with category labels.

The summary of commonly used object detection datasets and the statistics for the number of images and object instances for the corresponding recognition challenges are detailed in Table 2.3.

Table 2.3: General object detection datasets summary.

Dataset	Categories	Number of Images			Number of annotated objects		Train + Val			Image Size
		Train	Val	Test	Train	Val	Images	Boxes	Boxes / Image	
PASCAL-VOC07 [50]	20	2,501	2,510	4,952	6,301	6,307	5,011	12,608	2.5	470×380
PASCAL-VOC12 [51]	20	5,717	5,823	10,991	13,609	13,841	11,540	27,450	2.4	470×380
Highlights	Covers 20 categories that are common in everyday life; Large number of training images; Close to real-world applications; Significantly larger intra-class variations; Multiple objects in one image.									
MS-COCO-15 [52]	80	82,783	40,504	81,434	604,907	291,875	123,287	896,782	7.3	640×480
Highlights	Even closer to real world scenarios; Each image contains more instances of objects and richer object annotation information; Contains object segmentation data.									
ILSVRC-17 [54]	200	456,567	20,121	65,500	478,807	55,502	476,688	534,309	1.1	500×400
Highlights	Large number of object categories; More instances and more categories of objects per image.									
Open Images V4 [55]	500	1,643,042	100,000	99,999	11,498,734	696,410	1,743,042	12,195,144	7.0	Varied
Highlights	Annotated with image level labels, object bounding boxes and visual relationships.									

2.3.2 Datasets for specific detection tasks

Besides general object detection, there are many specific object detection datasets dedicated for applications in a variety of sub-fields. For example, face detection, pedestrian detection, traffic sign/light detection, and remote sensing target detection. Table 2.4 summarizes some of the most popular datasets for such specific detection tasks.

Table 2.4: Some popular datasets for specific detection tasks.

Dataset	Year	Description
Caltech [56][57]	2009	One of the most famous pedestrian datasets and benchmarks. Consists of ~190,000 pedestrians in the training set and ~160,000 in the testing set. The metric is PASCAL-VOC07 @0.5 IoU.
AFLW [58]	2011	One of the most famous datasets for face detection. Consists of ~26,000 faces and 22,000 images from Flickr with rich facial landmark annotations.
KITTI [59]	2012	One of the most famous datasets for traffic scene analysis. Consists of ~100,000 pedestrians.
WiderFace [60]	2016	One of the largest face detection datasets. Consists of ~ 32,000 images and 394,000 faces with rich annotations (i.e., scale, occlusion, pose, etc.).

2.4 Evaluation metrics

Generally, there are three main criteria considered for performance evaluation of object detectors: detection speed in Frames Per Second (FPS) or seconds per image, precision, and recall. In recent years, the most commonly used metric has been the "Average Precision (AP)", derived from precision and recall. AP is defined as the average detection precision under different recalls, and is usually evaluated in a category specific manner, i.e., computed for each object category separately. To compare performance over all object categories, the mean AP (mAP) averaged over all object classes is usually adopted [50][51][53][61].

The outputs of a target object detector applied to a testing image *I* are the predicted detections $\{(b_i, c_i, p_i)\}$, indexed by object i, of bounding box (bbox) b_i, predicted category c_i, and corresponding confidence score p_i. A predicted detection (b, c, p) is regarded as True Positive (TP) if the predicted category c equals the ground truth label c_{gt}, and the intersection over union (IoU) between the predicted bbox b and the ground truth bbox b^{gt} is not smaller than a predefined threshold ε, for which a typical value of ε is 0.5.

$$IoU(b, b^{gt}) = \frac{area(b \cap b^{gt})}{area(b \cup b^{gt})}, \tag{2.1}$$

where, ∩ and ∪ denote intersection and union, respectively. Otherwise, the prediction is considered as False Positive (FP). The confidence level p is usually compared with some threshold β to determine whether the predicted class label c is accepted.

AP is computed separately for each of the object classes, based on precision and recall. For a given object class c and a testing image I_j, let $\{b_{ji}, p_{ji}\}_{i=1}^{M}$ denote the detection results returned by a target object detector, ranked by confidence p_{ji} in decreasing order, and where M denotes the number of predictions considered. Each detection is either a TP or a FP, which can be calculated by the Algorithm 2.1. The precision $P(\beta)$ and recall $R(\beta)$, as defined in Table 2.5, can be computed as a function of the confidence threshold β based on the TP and FP, so by varying the confidence threshold different pairs (P, R) can be obtained and the precision can be regarded as a function of recall $P(R)$, from which the Average Precision (AP) can be found. Based on the PASCAL VOC challenge [50], AP is defined as the mean precision at a set of eleven equally spaced recall levels $\beta \epsilon [0, 0.1, \ldots, 1]$:

$$AP = \frac{1}{11} \sum_{\beta \epsilon \{0, 0.1, \ldots, 1\}} P(R)\,. \tag{2.2}$$

Algorithm 2.1 Algorithm for TPs and FPs calculation by greedily matching detection results to ground truth labels

Input:	$\{b_i, p_i\}_{i=1}^{M}$	M predictions for image I for given object category c, ranked by the confidence p_i in decreasing order;
	$B = \{b_k^{gt}\}_{k=1}^{K}$	ground truth bboxes on image I for given object category c. where K denotes the number of ground truth bboxes.
Output:	$\boldsymbol{a} \in \mathbb{R}^M$	a binary vector indicating each (b_i, p_i) to be a TP or a FP.

1: Initialize $\boldsymbol{a} = 0$
2: **for** $i = 1, \ldots, M$ **do**
3: Set $A = \emptyset$ and $t = 0$
4: **for** each unmatched object b_k^{gt} in B **do**
5: **if** $IoU(b_i, b_k^{gt}) \geq \varepsilon$ and $IoU(b_i, b_k^{gt}) > t$ **then**
6: $A = \{b_k^{gt}\}$
7: $t = IoU(b_i, b_k^{gt})$
8: **end**
9: **end**
10: **if** $A \neq \emptyset$ **then**
11: Set $\boldsymbol{a}(i) = 1$ since object prediction result (b_i, p_i) is a TP
12: Remove the matched ground truth box in A from B, $B = B - A$
13: **end**
14: **end**

Due to the popularity of the MS-COCO [52] dataset, the accuracy of the bounding box location, which defines where a detected object is placed in an image, received more attention. Instead of using a fixed IoU threshold, MS-COCO AP is averaged over multiple IoU thresholds between 0.5 (coarse localization) and 0.95 (perfect localization). This change of metric has encouraged more accurate object localization. Despite the recent changes, the VOC-based mAP is still the most frequently used evaluation metric for object detection [34][37][38][43][44][46]. Table 2.5 summarizes the main metrics used in object detection challenges related to PASCAL, ImageNet and MS-COCO datasets.

Table 2.5: Summary of evaluation metrics in object detection.

Parameter	Meaning	Definition and description	
TP	True positive	A true positive detection (correct classification)	
FP	False positive	A false positive detection (erroneous classification)	
β	Confidence threshold	A confidence threshold for computing $P(\beta)$ and $R(\beta)$	
ε	IoU threshold	PASCAL-VOC	Typically 0.5
		ILSVRC	$\min\left(0.5, \frac{w \times h}{(w+10)(h+10)}\right)$; $w \times h$ is the size of ground truth box
		MS-COCO	Ten IoU thresholds $\varepsilon \in \{0.5 : 0.05 : 0.95\}$
$P(\beta)$	Precision	The fraction of correct detections out of the total detections with confidence of at least β. $$P = \frac{TP}{TP + FP} \quad (2.3)$$	
$R(\beta)$	Recall	The fraction of all N objects detected by detector having a confidence of at least β. $$R = \frac{TP}{N} \quad (2.4)$$	
AP	Average Precision	Computed over the different levels of recall achieved by varying the confidence β, as defined in (2.2).	
mAP	mean Average Precision	PASCAL-VOC	AP at a single IoU and averaged over all classes
		ILSVRC	AP at a modified IoU and averaged over all classes
		MS-COCO	AP_{COCO} : mAP averaged over ten IoUs:$\{0.5. : 0.05 : 0.95\}$ $AP_{COCO}^{IoU=0.5}$: mAP at $IoU = 0.5$ (PASCAL-VOC metric) $AP_{COCO}^{IoU=0.75}$: mAP at $IoU = 0.75$ (strict metric)

2.5 Refinements on object detection

The excellent performance reported on object detection with the above mentioned state-of-the-art CNN architectures is traditionally demonstrated on a single image, which is captured by a fixed camera. In the context of swarm robotics, this situation corresponds to an image captured by a camera mounted on one of the collaborative robots as they begin to explore and understand the surrounding environment. However, when it comes to continuously detecting and recognizing target objects through successively captured images, the approach can offer additional benefits. The context considered in this research involves that a robot keeps moving toward a target as directed by a previous object detection result. Taking advantage of additional information that is made available via repetitive imaging of the target object, the recognition confidence score generated by the object detector can potentially increase as the object becomes clearer in the progressively captured sequence of images. In turn, this particularity of a navigating vision sensor can be leveraged to increase the reliability and robustness of target detection in order to support specialized robots task allocation.

There are research works in the literature that focus on further improving target object detectors' performance by post-processing the detection results. However, very limited work studied the problem from the perspective considered in this thesis. Two popular groups of methods are found in recent studies: 1) bounding box refinement with single object detector, and 2) detection results fusion with multiple object detectors.

2.5.1 Bounding box refinement

Bounding box refinement is the most intuitive way to improve the localization accuracy of a target object detector. Although the bounding box regression has been integrated into most of target object detectors, there are still some objects with unexpected scales that cannot be well captured by any of the predefined anchors. This will inevitably lead to an inaccurate prediction of their locations. For this reason, the authors of [62][63][64] have introduced the "iterative bounding box refinement" method to iteratively feed the detection results into a bounding box regressor until the prediction converges to a correct localization and size. The general strategy to improve the

localization accuracy and detection performance is to modify the detection pipeline by adding refining layers to iteratively refine object box proposals. However, the refinement is conducted only on a unique image captured by the static camera. It does not involve the support of navigating the camera toward the object and then acquiring images successively under guidance from the earlier detection.

2.5.2 Detection results fusion

Besides performing bounding box refinement on the detection of a single object, other research works explored the fusion of individual detection results from multiple object detectors to enhance the detection performance, but they applied the various detectors on a same image and not on a sequence of frames as proposed in this thesis. Currently, the dominating decision fusion approaches are the Bayesian fusion algorithm and the Dempster-Shafer fusion method [65].

- **Bayesian data fusion**

Information fusion based on the Bayesian inference provides a formalism for combining evidence according to the probability theory rules. Uncertainty is represented using the conditional probability terms that describe beliefs and takes on values in the interval $[0,1]$, with zero indicating a complete lack of belief and one indicating an absolute belief. The Bayesian inference is based on the Bayes rule as follows:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}, \tag{2.5}$$

where the posterior probability $P(Y|X)$ represents the belief in the hypothesis Y given the information X. This probability is obtained by multiplying the priori probability of the hypothesis $P(Y)$ by the probability of having X given that Y is true, $P(X|Y)$. The value $P(X)$ is used as a normalizing constant. The main disadvantage of the Bayesian inference is that the probabilities $P(X)$ and $P(X|Y)$ must be known. Liggins *et al.* [66] argued that the difficulty in establishing the value of prior probabilities and the difficulty in describing the uncertainty of the decisions are the problems associated with Bayesian inference.

- **Dempster-Shafer theory**

The Dempster-Shafer (D-S) theory proposed by Dempster [67] and Shafer [68] and the Dempster's Rule of Combination (DRC) provides an alternative strategy for data fusion, which forms the foundation of the proposed target object detection approach with progressive refinement that will be detailed in Chapter 4. The D-S theory is based on a belief function and plausible reasoning that obtains degrees of belief for a hypothesis from subjective probabilities of related hypotheses. The Dempster's Rule of Combination (DRC) seeks to combine probabilities when they are based on independent items of evidence.

The D-S theory begins by assuming a frame of discernment or a universe set Θ, which is a finite set of mutually exclusive hypotheses. It is the set of all states under consideration. The power set 2^{Θ} is the set of all possible subsets of Θ including the empty set $\emptyset$. For example, if:

$$\Theta = \{H_1, H_2, \dots, H_M\}, \tag{2.6}$$

then

$$2^{\Theta} = \{\emptyset, \{H_1\}, \{H_2\}, \dots, \Theta\}. \tag{2.7}$$

The theory of evidence assigns a mass value m between 0 and 1 to each subset of the power set, which defines the *mass function* or the *basic probability assignment* (BPA). The mathematical expression is:

$$m\colon 2^{\Theta} \rightarrow [0, 1]. \tag{2.8}$$

A BPA has two properties: First, the mass of the empty set is zero:

$$m(\emptyset) = 0, \tag{2.9}$$

and second, the masses of remaining members of the power set must sum to 1:

$$\sum_{A \subseteq 2^{\Theta}} m(A) = 1\,. \tag{2.10}$$

Given basic probability assignments for the power set, the upper and lower bounds of a probability interval can be determined since these are bounded by two measures that can be calculated from the mass, the *Belief function (Bel)* and the *Plausibility function (Pl).* The belief function of a hypothesis A, $Bel(A)$, defines the sum of all mass values of all the non-empty subsets of A:

$$Bel(A) = \sum_{B \subseteq A} m(B)\,. \tag{2.11}$$

The plausibility function of A, $Pl(A)$, sums the masses of all the sets that intersect A, which takes into account all the elements related to A:

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B)\,. \tag{2.12}$$

The belief function and plausibility function are related to each other as follows:

$$Pl(A) = 1 - Bel(\neg A). \tag{2.13}$$

It is always true that:

$$m(A) \leq Bel(A) \leq Pl(A). \tag{2.14}$$

For the subset A, $Pl(A)$ and $Bel(A)$ represent the upper and lower bounds of the probability interval respectively, and the interval $[Bel(A), Pl(A)]$ represents the probability range of uncertainty. The relationships between Bel value, Pl value and uncertainty are shown graphically in Figure 2.5.

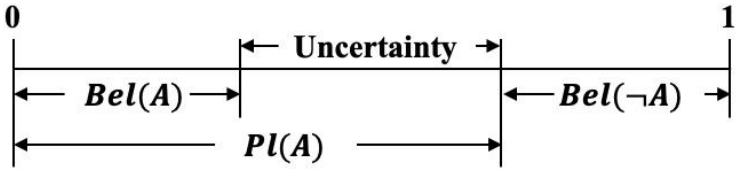


Figure 2.5: Relationships between Belief function $Bel(A)$, Plausibility function $Pl(A)$, and uncertainty $Pl(A) - Bel(A)$.

The Dempster's rule of combination is concerned with uniting two independent sets of BPAs on a frame of discernment Θ, for example m_1 and m_2. The joint BPA of m_1 and m_2 would be expressed as $m_{1,2}$ or m_f where:

$$m_f(A) = m_{1,2}(A) = (m_1 \oplus m_2)(A), \tag{2.15}$$

and

$$(m_1 \oplus m_2)(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_1(B) m_2(C)\,, \tag{2.16}$$

where K, a measure of the conflict between the two mass sets, is given by:

$$K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C)\,, K \neq 1. \tag{2.17}$$

Equation (2.16) emphasizes the agreement between multiple sources of information and ignores conflicting evidence by using a normalization factor, which is equal to $1 - K$. The normalization

factor attributes any mass associated with conflict to the null set. DRC can be extended for multiple pieces of evidence using the associative and commutative properties of BPAs, for example $m_1, \dots, m_M$:

$$m_f(A) = \frac{1}{1-K} \sum_{X_1 \cap X_2 \cap \dots \cap X_M} \prod_{i=1}^{M} m_i(X_i), \tag{2.18}$$

where $K = \sum_{X_1 \cap X_2 \cap \dots \cap X_M = \emptyset} \prod_{i=1}^{M} m_i(X_i)$.

In contrast to the Bayesian inference, prior probabilities are not required in the Dempster-Shafer inference, because they are assigned at the instant that the information is provided. The authors of [69][70] have compared the use of Bayesian inference and Dempster-Shafer theory as fusion methods for object detection. More recently, Lee *et al*. [71][72] proposed a fusion algorithm based on Dempster-Shafer theory to improve the detection performance by fusing decisions from multiple object detectors applied in parallel on a same image. However, they did not investigate refining the detection performance by applying a detector successively on different images.

2.6 Applications of target object detection

In this section, some important applications of target object detection are reviewed, including multi-object tracking (MOT), especially for pedestrians, and autonomous driving. These applications are considered here as they share common characteristics with the objectives of this research, which requires the detection of unique or several instances of predefined objects in a given environment through iterative image capture and application of object detectors.

2.6.1 Multi-object tracking (MOT)

The outcomes from target object detection are identified in the literature as a key factor influencing object tracking performance. In the context of multi-object tracking (MOT) the goal is to analyze videos in order to identify and track objects belonging to one or more categories, such as pedestrians or cars, without any prior knowledge about the appearance and number of targets. The standard approach employed in MOT is tracking-by-detection, where target objects are detected from video frames and represented as bounding boxes which are used to guide the tracking process,

usually by associating them together in order to assign the same ID to bounding boxes that contain the same target. That is, given the raw frames of a video, a target object detector is run to obtain the bounding boxes of the targets; then, for each detected object, the visual appearance and motion features are computed; after that, an affinity computation is conducted to calculate the similarity between pairs of detections; finally an association step assigns a numerical ID to each object.

The CNN-based target object detectors that we described in Section 2.2.2 have been used as the detection step in many MOT works. The Simple Online and Realtime Tracking (SORT) [73] has been one of the first MOT frameworks that utilize CNN for the detection and tracking of pedestrians, where Faster RCNN is used as target object detector. Then the object motion is predicted by Kalman filter [74] and the detections are associated together with the help of the Hungarian algorithm [75]. Besides, the work proposed by Yu *et al.* [76] also has taken advantage of the detection accuracy of Faster RCNN. In the work of [77], Mask RCNN has been used to detect and track pedestrians. Both two-stage and one-stage object detectors are used as the detection stage in pedestrian tracking. Kieritz *et al.* [78] applied the SSD detector in the detection step while Kim *et al*. [79] utilized YOLOv2 to perform pedestrians detection. However, detecting small pedestrian targets that are captured far from the camera in the scene and hard negative samples caused by the confusion of background instances still represent difficulties and challenges in pedestrian detection and tracking. Some recent solutions to improve small pedestrian detection are feature fusion [80] and introducing extra high-resolution handcrafted features [81]. Recent improvements for hard negative detection include the integration of boosted decision tree [80] and the introduction of "cross-modal learning" to enrich the feature of hard negatives by using RGB and infrared images [82]. Even though these works acquire successive image frames in the video format, they still apply the detector individually on every images to detect the target since the cameras are usually static.

2.6.2 Autonomous driving

Autonomous driving requires an accurate perception, which refers to the understanding of its surrounding environment, such as where obstacles are located, detection of traffic signs/lights and categorizing objects by their semantic labels such as pedestrians and vehicles. Target object

recognition and detection is a fundamental function of the perception system in an autonomous vehicle (AV), as failing to identify and recognize road agents might lead to safety-related incidents.

CNN-based target object detection works that we described in Section 2.2.2 are using 2D detection methods, however, the 2D methods do not provide depth information, which is required for autonomous driving tasks. Alternatively, 3D target object detection methods introduce a third dimension that reveals more detailed object's size and location information. Table 2.5 presents a comparison on 2D versus 3D target object detection.

Table 2.6: Comparison between 2D and 3D target object detection methods.

	2D target object detection	3D target object detection
Advantage	Well established detection framework and available datasets. RGB image input can achieve accurate results in image plane.	3D bounding box provides target object size and position in world coordinates. The above detailed information leads to better environment understanding.
Disadvantage	Limited information: lack of pose of target object and 3D position information.	Required depth estimation for precise localization. Extra dimension regression increases model complexity. Limited 3D labelled datasets.

Several works on autonomous driving are based only on RGB images. They first detect 2D target object candidates on the image plane and then extrapolate them to 3D through reprojection constraints or bounding box regression. Xiang *et al*. proposed the 3D Voxel Pattern (3DVP) [83] representation that models appearance through RGB intensities, and 3D shapes as a set of voxels. The dictionary of 3DVPs is obtained by clustering the patterns observed on the data and training a classifier for each specific pattern given a 2D image segment of a vehicle. They achieve 3D detection by minimizing the reprojection error between the projected 3D bounding box to the image plane and the 2D detection. Chabot *et al*. estimate 3D pose through 3D templates [84]. They first obtain 2D bounding regression and parts localization through a two-level refinement region-proposal network. Next, based on the inferred shape, 3D model matching is performed to obtain

the 3D pose. Mousavian *et al*. [85] first extend standard 2D target object detector with 3D orientation (yaw) and bounding box size regression.

Besides using RGB images for the perception stage, point clouds generated by a Lidar sensor as the input data for autonomous driving have been researched as well. Li *et al*. [86] use a cylindrical projection mapping and a Fully Convolutional Network (FCN) to predict 3D bounding boxes around vehicles. Simony *et al*. [87] propose a method using a YOLOv2 based architecture [44] with extensions to predict the extra dimension and yaw angle, named Complex-YOLO, which uses the bird-eye view projection to generate 3D proposals. Under the scenario of autonomous driving, sensors are mounted on vehicles, but the vehicle is not driven to aim toward the detected objects, instead it avoids getting close to the detected objects which are usually regarded as obstacles, such as pedestrians and other vehicles.

2.7 Summary of literature review

In this chapter, we reviewed the evolution of convolutional neural networks (CNNs). We highlighted their powerful learning ability for feature representations, which enables CNNs to be deployed as the backbone network in target object detection frameworks while reaching excellent performance on target object recognition. As such, we investigated state-of-the-art solutions for CNN-based target object detectors with the two main classes that emerged in the deep learning era: CNN-based two-stage detectors and CNN-based one-stage detectors. For two-stage detectors, the RCNN family of detectors (RCNN, SPPNet, Fast/Faster RCNN, RFCN and Mask RCNN), all involve a region proposal generation stage, which aims at coarsely finding candidate proposals, and a detection stage, which is used to finely recognize objects and improve the precision of candidate's offsets. One-stage detectors, especially the YOLO family of detectors (YOLO, YOLOv2 and YOLOv3), reframe the object detection as a single regression problem that estimates bounding box coordinates and category probabilities straight from image pixels. The latter sacrifices some of the detection precision compared with two-stage detectors. Besides, we reviewed the most widely used datasets and performance evaluation metrics for object recognition. Finally, an overview of real-world applications of target object recognition is presented to expose shared characteristics and differences with the specific application considered in this thesis.

The literature study shows that target object recognition is of great importance for visual perception in many real-world applications. In the context of task allocation for collaborative specialized robotic agents, which is investigated in this thesis, detecting and recognizing predefined classes of target objects that the agents may encounter while exploring a workspace plays a critical role. As a result, accurate detection of the target objects is crucial to assist in directing the specialized individual agents toward their corresponding targets with maximum efficiency, leading to a division of labor that emerges within a multiple agents system. At the end, accurate and reliable target object detection contributes to increase the net efficiency of the deployed robotic swarm. The work presented in the following chapters aims to exploit CNN-based target object recognition and detection methods in the specific context of collaborative robots task allocation, and to contribute an original progressive refinement strategy for the recognition process, while leveraging state-of-the-art pattern recognition and data fusion methods.

Chapter 3 Classical CNN-based Target Object Detection

Given the excellent performance offered by CNN architectures for target object recognition and detection, as reported in the literature works, we exploit their potential in the context of collaborative robots task allocation for search-and-rescue (SAR) operations in this chapter. The purpose is to experimentally explore CNN-based object detectors' abilities to efficiently deal with images containing multiple target objects as well as instances of images at different scales while a robot is moving toward those objects from far away and getting closer. We also aim to evaluate their potential for being embedded as a real-time processing stage on robots. In Section 3.1, we explore and propose some modifications on the two-stage frameworks of Faster RCNN and Mask RCNN to make them better suit the requirements imposed by the context of this thesis, while initially experimenting on public datasets. Then, considering the detection speed advantage that one-stage detectors provide, Section 3.2 introduces and explores the YOLO architecture. Finally, Section 3.3 presents an experimental comparison between two-stage and one-stage detectors on a custom dataset, which is built to better match the SAR operational context investigated in this thesis.

3.1 Two-stage target object detector

The overview of CNN based two-stage detector architectures was presented in Section 2.2.2.1. Detectors of the RCNN family have been attracting increasing attention because of their performance. Basically all two-stage RCNN family detectors are composed of two major components, which are the region proposal generator and the detector. The region proposal generation part serves to coarsely identify candidate proposals that correspond to regions of an image that are likely to contain instances of an object of interest. The following detection part refines the recognition of objects and improves the precision on candidate's coordinates. The differences in between different two-stage object detection CNN architectures reside in how region proposals are generated and how detection is conducted. For region proposal generation, RCNN, SPPNet and Fast RCNN obtain region proposals through a Selective Search (SS), which leads to huge computations. From Faster RCNN to Mask RCNN, the Region Proposal Network (RPN) is introduced to generate candidate proposals with reduced computational cost. In terms of

classification and detection, RCNN and SPPNet use Support Vector Machines (SVMs) as classifiers and perform bounding boxes regression to locate the target objects positions. Since the Fast RCNN was proposed, alternative detectors that were introduced achieved classification and bounding box regression directly followed by the CNN. The latter only serves as the individual part for feature extraction in RCNN and SPPNet, with a generated classification layer and a bounding box regression layer. As a result, from Faster RCNN to Mask RCNN, the region proposal generation and the target classification and detection have been gradually integrated into a unified, end-to-end learning framework, while they performed as individual and separately learned blocks in RCNN, SPPNet and Fast RCNN.

Particularly, Mask RCNN not only classifies and locates the target objects, but also precisely labels each target at the pixel level, resulting in a mask segmentation. The success of Mask RCNN showed that object detection can be improved by learning with semantic segmentation. The strategy to improve object detection by segmentation in Mask RCNN is to introduce an additional segmentation branch on top of the original Faster RCNN detection framework, and to train the model with multi-task loss functions, including segmentation loss and detection loss. The details on loss functions will be introduced in Section 3.1.4. Figure 3.1 shows the detailed architecture of Faster RCNN and Mask RCNN. The three main components in the two-stage target object detection framework are: the backbone architecture, the region proposal network (RPN), and the detection and segmentation, which serves for bounding box recognition (classification and regression) and mask prediction. These components are detailed individually in the following sections.

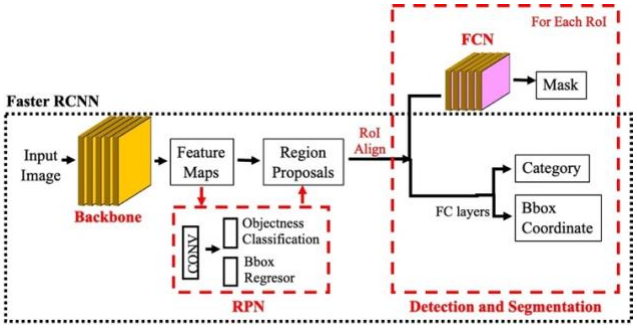


Figure 3.1: Detailed two-stage target object detection frameworks of Faster RCNN (inside dotted black line) and Mask RCNN with three main components: backbone network, region proposal network (RPN), and detection and segmentation network.

3.1.1 Backbone architecture

The quality of feature representations is of great importance for target object detection. The role of the backbone network involved in the detection framework is to extract features from the input image. Since great generalization performance was achieved with CNN-based object classification tasks, several implementations take advantage of state-of-the-art deep CNNs as the backbone networks and combine them with different loss functions in the target object detection pipeline. In the literature related to Mask RCNN [41], the combination of ResNet of depth 50 layer and FPN (ResNet-50-FPN), the combination of ResNet of depth 101 layers and FPN (ResNet-101-FPN), and the combination of ResNeXt of depth 101 layers and FPN (ResNeXt-101-FPN) were separately used as the backbone network for feature extraction. With the network layers increasing, the computation costs more. Therefore, considering the limitation that exists on computational resources available when dealing with mobile robotic agents, as considered in this research, we focused on the implementation of ResNet-50 and FPN (ResNet-50-FPN) with adaptations and modifications as a feature extraction backbone network in our initial work on investigating two-stage target object detectors.

- **ResNet**

In previous neural networks, the output of each layer feeds into the following layer directly with the expectation that a deeper network should reach better performance. However, with the network depth increasing, the gradient vanishing problem occurs, as the gradient is back-propagated to earlier layers, and repeated multiplication may make the gradient infinitely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly [88]. The major contribution of ResNet is that it introduced a deep residual learning framework with "identity shortcut connections" to solve the degradation problem. The introduced shortcut connections connect layers further behind the predicting layer to a given layer. This allows gradients to propagate faster to deep layers before they can be attenuated to small or zero values. ResNet consists of several building blocks (Figure 3.2(a)). In these blocks, the identity shortcut connection is applied, that is to skip one or more layers. The building block is defined as:

$$y = F(x, \{W_i\}) + x \tag{3.1}$$

Here x and y denote input and output vectors of the layers considered, W_i is the weight vector. $F(x, \{W_i\})$ represents the residual function. Figure 3.2(a) shows a two-layer building block, $F(x) = W_2\sigma(W_1 x)$, where σ represents ReLU [15]. Shortcut connection and element-wise addition is represented by $F(x) + x$, which does not introduce any extra parameters or computations. The identity shortcuts (Equation (3.1)) can be directly used when the dimension of input and output is the same, which represents the solid line shortcuts in Figure 3.3. Otherwise, when the dimensions increase, as shown as the dotted line shortcuts in Figure 3.3, a linear projection W_s is introduced in the shortcut connection to match the dimensions (done by 1×1 convolutions):

$$y = F(x, \{W_i\}) + W_s x \tag{3.2}$$

When the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

Two different types of building blocks are shown in Figure 3.2(b) and (c), the former is used in 34-layer ResNet, while the latter is used in 50-layer/101-layer/152-layer ResNet. The former 2-layer building block was modified to the latter 3-layer "bottleneck" building block. The stack of 3 layers are $1 \times 1, 3 \times 3$ and 1×1 convolutions, where the 1×1 layers are used for the reducing and increasing dimensions, which leaves the 3×3 layer a bottleneck with smaller input/output dimensions.

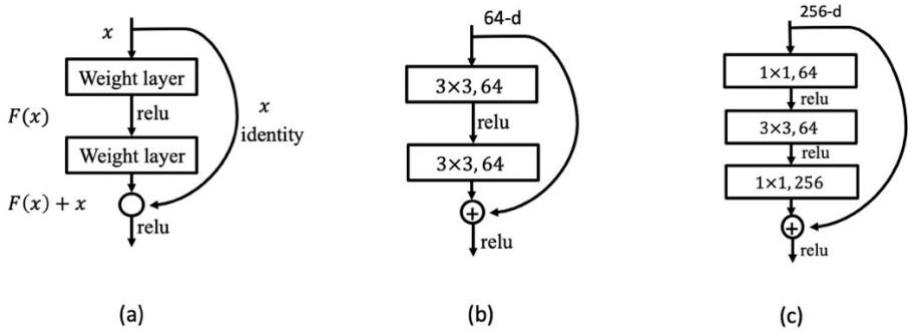


Figure 3.2: ResNet building blocks. (a) Identity shortcut connection, (b) building block for ResNet-34, and (c) "bottleneck" building block for ResNet-50/101/152.

Since the ResNet showed excellent performance on image recognition and localization in literature works, considering the depth and computational complexity, the architecture of 50-layer ResNet, which is used as the backbone network in our investigation work, is illustrated in Figure 3.3. The convolution stages are denoted as conv1, conv2_x, conv3_x, conv4_x and conv5_x, where the x represents the numbers of building blocks.

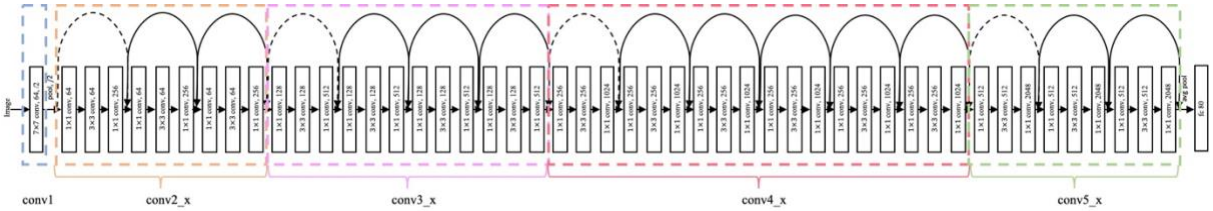


Figure 3.3: ResNet-50 architecture for MS-COCO dataset. The identity shortcuts are presented in two ways: solid line shortcuts indicate the input and output are of the same dimension; dotted line shortcuts indicate a dimension increase. Down sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

Compared with the literature work, in which the outputs at the last fully connected layer (fc) were 1000 dimensional vectors for 1000 classes of ImageNet 2012 dataset, this work modifies the final outputs at the last fully connected layer as 80 dimensional vectors to adapt to the MS-COCO dataset. This modification is motivated by the fact that the pretrained weights of target object detectors that we conducted the investigation with were determined on the MS-COCO dataset, which contains 80 category classes. These experiments use an implementation from [89], from which we loaded the pretrained weights as starting point to train our adapted model. Table 3.1 shows the details of the ResNet-50 architecture for the MS-COCO dataset. Besides, reducing the dimensions of output reduces the computation costs in the training stage and also accelerates the detection process, which makes it possible to deploy the detection pipeline on mobile robot platforms with small processors.

Table 3.1: ResNet-50 architecture for the MS-COCO dataset. Building blocks are shown in brackets (also see Figure 3.3), with the number of blocks stacked. Down sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

Layer name	Output size	ResNet-50	Number of layers
conv1	$\mathbf{112 \times 112}$	77, 64, stride 2	1
conv2_x	$\mathbf{56 \times 56}$	33 max pool, stride 2	9
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	
conv3_x	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	12
conv4_x	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	18
conv5_x	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	9
Fully connected (fc)	1×1	average pooling, 80-d fc, softmax	1

- **FPN**

Detecting objects at different scales is challenging in particular for small objects, which often appear in the context of collaborative robots task allocation when the target is initially imaged from far by a sensor mounted on the robots. To address the problem of object detection over multiple scales, which involves detecting target objects of different sizes in a recognition system, previous work [7] used a pyramid of the same image at different scales to detect objects. However, processing multiple scaled images is time consuming and the memory demand is too high for the network to be trained end-to-end simultaneously. Alternatively, a pyramid of features was created for object detection. The Feature Pyramid Network (FPN) [40] forming a feature extractor was designed to support such pyramidal concepts, with the accuracy and speed in mind, to generate multiple feature map layers (multi-scale feature maps) with better quality information than the regular feature pyramid for object detection. Figure 3.4 illustrates the construction of a FPN. It consists of a bottom-up pathway, top-down pathway, and lateral connection.

The bottom-up pathway is the usual convolutional network for feature extraction. As it goes up, the spatial resolution decreases. With more high-level structures and stronger features detected, the semantic value for each layer increases. The top-down pathway is designed to construct higher resolution layers from a semantic rich layer. While the reconstructed layers are semantically strong, the locations of objects are not precise after all the down-sampling and up-sampling. Therefore, lateral connections between reconstructed layers and the corresponding feature maps are added to help the detector predict the locations more accurately. It also acts as skip connections to make training easier (similar to what ResNet does).

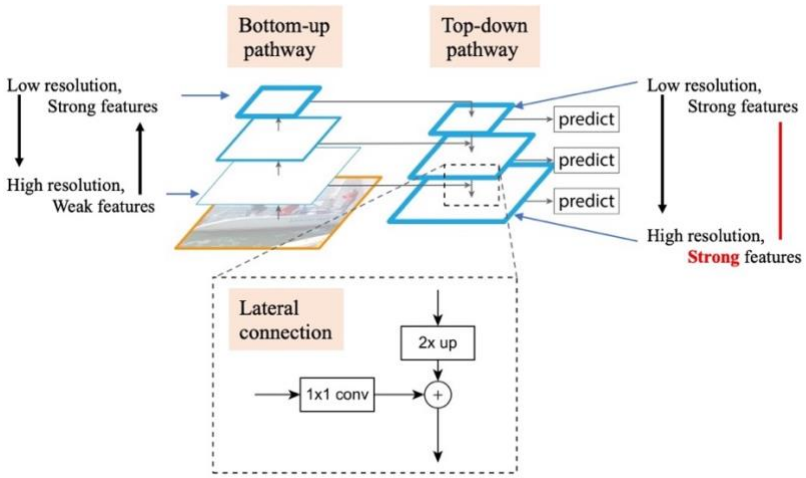


Figure 3.4: Feature Pyramid Network (FPN) building blocks, consisting of a bottom-up pathway, top-down pathway, and lateral connections.

The mechanism of the backbone network in our initial experimental implementation combines ResNet-50 and FPN with several adaptions and modifications to better fit with our development context in collaborative robots task allocation, as shown in Figure 3.5. The feature activation output from the last residual block of each stage in ResNet, shown in Figure 3.3, is used to construct the bottom-up pathway. $\{C_2, C_3, C_4, C_5\}$ denote the output of the last residual blocks for conv2, conv3, conv4 and conv5, and they will be later used in the top-down pathway. Besides, they have strides of $\{4, 8, 16, 32\}$ pixels with respect to the input image, which means the spatial dimension is reduced by $1/2$ as we move up. Then a 1×1 convolutional filter is applied to reduce

the channel dimension of C_5 to create the coarsest resolution map M_5. As we go down the top-down path, the previous layer is upsampled by 2 using nearest neighbors upsampling. Again, a 1×1 convolution is attached to the corresponding feature maps in the bottom-up pathway. Next, we merge them by element-wise addition. After that, a 3×3 convolution is applied to the merged layers and generate the feature map P_5. This filter reduces the aliasing effect when merged with the up-sampled layer. This process is iterated until the finest resolution map is generated.

Compared with the literature work which had four pyramid feature maps in total, a modification is proposed in this work by which we introduce P_6 in our implementation to cover a larger anchor scale, which is formed from a stride 2 subsampling of P_5. Therefore, the final set of pyramid feature maps in our implementation is called $\{P_2, P_3, P_4, P_5, P_6\}$, associated with 5 different scales. The addition of P_6 expands the scales of anchors, which benefits the scenario of interest in this work where the target gets larger in the collected image when the robot moves closer to it. The FPN takes an arbitrary size single-scale image as input, and outputs proportionally sized feature maps at multiscale levels, which will be later fed into the region proposal network (RPN), detailed in Section 3.1.2.

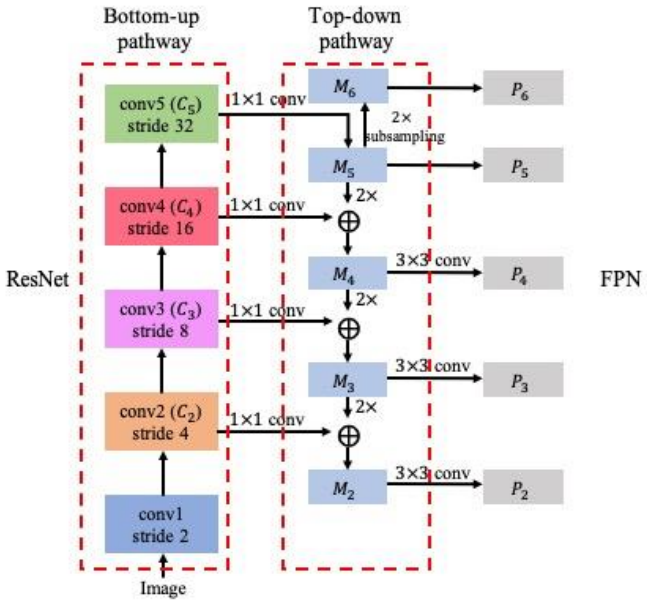


Figure 3.5: Backbone architecture of the combination of ResNet and FPN for feature extraction with the proposed additional pyramid feature map P_6.

3.1.2 Region proposal network (RPN)

The Region Proposal Network (RPN) concept was firstly introduced in the work on Faster RCNN [38]. It is a sliding-window class-agnostic object detector. Since convolutional feature maps can be used for generating region proposals, the RPN is constructed on the top of these convolutional features by adding two sets of additional convolution layers. One layer with a 3×3 convolutional filter encodes a convolution map position into a short feature vector, and another layer, at each convolution map position, is to output two sibling layers with 1×1 convolutions: a classifier with an objectness score (cls layer) and an object bounds regressor (reg layer) for k region proposals with various scales and aspect ratios at that location. Figure 3.6 illustrates the architecture of RPN.

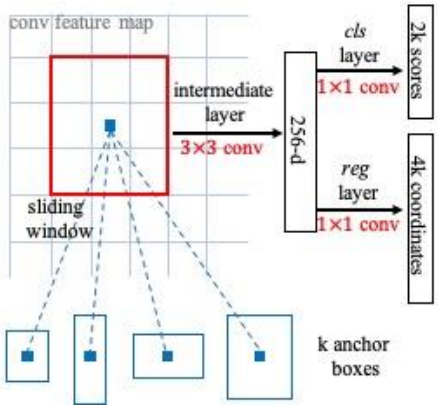


Figure 3.6: Architecture of Region Proposal Network (RPN).

Every point in the feature map is an anchor point, which can be considered as the central point of the sliding window. The k anchor boxes are generated for each anchor point, so the reg layer has $4k$ coordinates of k proposals and the cls layer outputs $2k$ scores that estimate the probability of binary classification for object/non-object for each proposal. Scales and aspect ratio are two important parameters in candidate boxes generation. In the original work on Faster RCNN [38], the authors used 3 scales and 3 aspect ratios. As a result, in total 9 proposals are possible for each pixel, and this is how the value of k is decided. For an image, the number of anchors is $W \times H \times k$, where W is the image width and H is the image height.

In the experimental implementation developed for this thesis, it is proposed to upgrade the framework with 3 aspect ratios and 5 scales based on the backbone output, with the objective to

produce more candidate proposals with different sizes. It is expected that the latter will be capable of matching target objects over a broader range of scales, which is essential to support the progressive refinement approach on target object detection that will be detailed in Chapter 4. As such, we increase the number of anchors for each anchor point, where $k = 15$. Figure 3.7 illustrates the combination of the extended backbone structure with the P_6 feature map (described in section 3.1.1) and the expanded region proposal network that supports 15 anchors in total, rather than the original 9, as used in our experiments. The features extracted from the backbone architecture (ResNet and FPN) are fed into the RPN for generating candidate proposals. In order for the two components to integrate properly and fully support object detection at multiple scales, the RPN is adapted by replacing the single-scale feature map with FPN. The head design (3×3 convolution and two sibling 1×1 convolutions) is attached to each level of the feature pyramid. The anchors of a single scale are assigned to each level. The three aspect ratios used for anchors are $\{1:2, 1:1, 2:1\}$ at each level. And the scales of anchor are designed for $\{8^2, 16^2, 32^2, 64^2, 128^2\}$ pixels on $\{P_2, P_3, P_4, P_5, P_6\}$, respectively. Compared with the scales of anchors used in literature works, we use smaller anchor scales in order to adapt to small objects in collaborative robots task allocation, where the target objects tend to initially appear in small size when captured from a large distance.

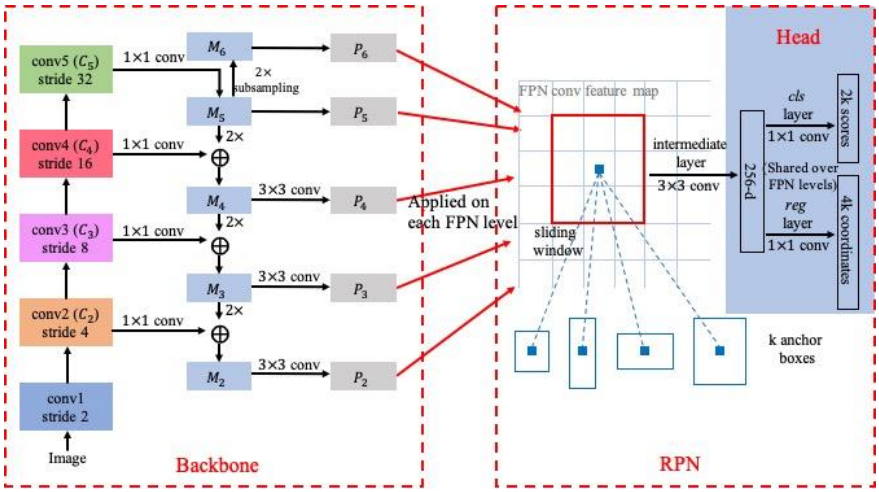


Figure 3.7: Region proposal network with the combination of extended ResNet and FPN with additional anchors as a feature extraction backbone architecture to detect target objects at different scales.

This architecture forms the first stage for generating candidate proposals, resulting in regions of interest (RoI) in the image. In the following section, the second processing stage for target object detection and segmentation on those RoIs will be introduced.

3.1.3 Detection and segmentation network

In terms of the Faster RCNN and Mask RCNN that we investigate in our implementations, the detection parts in both frameworks are performed by the same procedure, whereas the segmentation part only exists in Mask RCNN, and is performed by adding a parallel branch that outputs a binary mask for each RoI. Figure 3.8 shows the detection and segmentation part of the Faster RCNN and Mask RCNN. In Faster RCNN [38], for each object proposal a region of interest pooling layer extracts a fixed-length feature vector. Each feature vector is fed into a sequence of fully connected (fc) layers that finally generate two sibling output layers: one layer for classification producing a softmax probability, and another layer for bounding box regression outputting four real-value numbers representing a refined bounding box position. In Mask RCNN [41], for better adjusting the object segmentation functionality, the independent third branch for the object mask prediction, which is performed by using a FCN on each RoI and a RoIAlign operation, which is pixel-to-pixel alignment, were proposed in the literature to replace RoIPool that was applied in Fast/Faster RCNN.

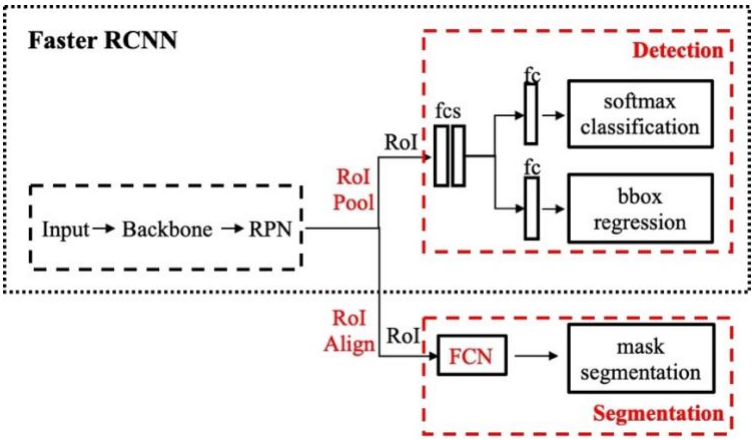


Figure 3.8: Detection and Segmentation Network in Faster RCNN (dotted black line) and Mask RCNN framework (overall architecture).

RoIPool in Faster RCNN is used for extracting a small feature map from each RoI. The quantization process performed in the RoIPool introduces misalignment between the RoI and the extracted features. This coarse spatial quantization for feature extraction may not impact classification, but it negatively affects the prediction of pixel-accurate segmentation masks. Therefore, the RoIAlign procedure, a quantization-free operation, is proposed in the literature to solve this misalignment problem. RoIAlign removes the harsh quantization of RoI boundaries or bins in RoIPool. It uses bilinear interpolation [91] to precisely compute the value of input features in each RoI bin. This pixel-to-pixel alignment performs the extraction of fine spatial layout of an object, which plays an important role in segmentation mask prediction. Therefore, we keep it in our implementation.

- **Fully convolutional networks (FCN)**

The object mask encodes the spatial layout of an input object. In the mask segmentation branch of Mask RCNN, a mask with $m \times m$ size is predicted from each RoI using a small Fully Convolutional Network (FCN) [42]. Figure 3.9 illustrates the architecture of FCN. The FCN network pipeline is an extension of the classical CNN. The main idea is to make the classical CNN take as input arbitrary-sized images. The restriction of CNNs to accept inputs only for specific sized inputs comes from the fully-connected layers (fcs) which are fixed by definition. FCN gets rid of fully-connected layers and only uses convolution and pooling layers. It first tries to encode input with a series of convolution layers. By replacing fully-connected layers with convolution layers, the classification output of fully-connected layers are replaced by heatmap of object classes. While it yields coarse output maps for input of any size, the output dimensions are typically reduced by subsampling from the size of the input by a factor equal to the pixel stride of the receptive fields of the output unit, which is $1/32$ of its original size. Since the network produces several feature maps with small sizes and dense representations, it needs up-sampling through deconvolution layers to resize the image to its original dimensions. Besides, it adds skip connections in the network to combine high-level feature map representations with more specific and dense ones at the top of the network. Similar to the literature, we use FCN as the mask segmentation branch for the experiments reported in this thesis.

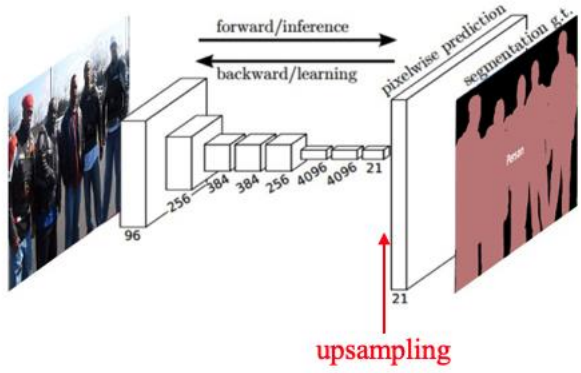


Figure 3.9: Fully Convolutional Network (FCN). FCN can learn to make dense predictions for per-pixel tasks like semantic segmentation.

The above illustration is the second stage of two-stage object detectors. Faster RCNN outputs the target objects classification and localization results, while Mask RCNN not only outputs the target detection results like Faster RCNN does, but also generates the precise mask segmentation in parallel by using a small FCN.

3.1.4 Experimental investigation of two-stage object detectors on public dataset

In this section, our initial experimental investigation of CNN-based two-stage target object detectors is reported to explore their feasibility and potential for recognizing and detecting objects with the goal to support task allocation in swarm robots under indoor Search-and-Rescue (SAR) scenarios [90]. The Faster RCNN and Mask RCNN architectures, involving the proposed modifications and adaptations described in sections 3.1.1 and 3.1.2, are experimentally validated. Five predefined classes of target objects that can realistically be imaged from a camera mounted over unmanned collaborative robotic systems are considered: person, door, stairs, sign, and fire (substituted by images of tvmonitor for obvious security reasons and testing constraints).

- **Data preparation**

Given the complexity and time required to create and manually annotate extensive data samples required for training the CNN-based target object detectors, this initial experiment aims to use published datasets that contain objects of relevance in indoor environments to perform a fast

validation of the classical architectures with the modifications described in sections 3.1.1 and 3.1.2 and in the given context. As a result, PASCAL-VOC07 dataset is selected for this experimentation, since the "person" and "tvmonitor" categories that it contains are of immediate interest. Moreover, it not only provides category and bounding box offset annotation in all images, with which the comparison between Faster RCNN and Mask RCNN on object class recognition and bounding box detection is conducted, but also provides segmentation masks annotation over 632 images, which supports the evaluation of Mask RCNN target object instance segmentation. From 632 images having mask labels, 400 images are randomly selected as a training set, and another 100 images are used for testing, while comparing with the ground truth provided by the predefined classes and segmentation masks.

- **Loss function**

The loss function defined in Mask RCNN is the multi-task loss on each sampled region of interest (RoI), which is the combination of the loss of classification (L_{cls}), bounding box regression (L_{box}) and segmentation mask prediction (L_{mask}):

$$L = L_{cls} + L_{box} + L_{mask}, \tag{3.3}$$

where L_{cls} and L_{box} are the same as in Faster RCNN. Table 3.2 lists the symbols that will be used in the description of loss function.

Table 3.2: Symbols and descriptions of parameters in the loss function.

Symbol	Definition and description
L_{cls}	The loss of classification.
L_{box}	The loss of bounding box regression.
L_{mask}	The loss of segmentation mask prediction.
u	Class label, $u \in 0,1,\dots,K$, where 0 denotes the background.
p	Discrete probability distribution (per RoI) over $K+1$ classes: $p = (p_0,\dots,p_k)$, computed by a softmax over the $K+1$ outputs of a fully connected layer.
v	Ground truth bounding box $v = (v_x, v_y, v_w, v_h)$(center coordinate, width, height).
t^u	Predicted bounding box correction, $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$.
y_{ij}	The label of cell (i,j) in the ground truth mask for the region of size $m \times m$.
$\hat{y}_{ij}^u$	The predicted value of the y_{ij} in the mask learned for the ground truth class u.

The classification branch outputs the softmax probability $p = (p_{0,\dots,}p_K)$, over $K+1$ classes, where $K+1$ represents K object classes plus 1 background class. As a result, the classification loss is the negative loss of the softmax probability of ground truth class u:

$$L_{cls} = -\log p_u\,. \tag{3.4}$$

The bounding box regression branch outputs the predicted bounding box offsets for each of the K object classes indexed by k, denoted as $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$, where t^kspecifies a scale-invariant translation (t_x^k, t_y^k) and log-space width or height shift (t_w^k, t_h^k) relative to an object proposal. The goal is to narrow down the gap between the predicted bounding box for class u, $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$, and the ground truth proposal for class u, $v = (v_x, v_y, v_w, v_h)$. The bounding box regression loss is defined as:

$$L_{box}(t^u, v) = \sum_{i\in(x,y,w,h)} {L_1}^{smooth}(t_i^u - v_i), \tag{3.5}$$

where, the smooth L_1 loss is defined as:

$${L_1}^{smooth}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases}. \tag{3.6}$$

The mask branch generates a mask, of dimension $m \times m$, for each RoI and each class, as illustrated in Section 3.1.3. For all K classes, the total output is the binary matrix of size of Km^2, where each pixel corresponds to a sigmoid probability indicating whether the pixel belongs to the mask of class u. Then the mask loss, L_{mask}, is defined as the average binary cross-entropy loss for each pixel.

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \log \hat{y}_{ij}^{u} + \left(1 - y_{ij}\right) \log\left(1 - \hat{y}_{ij}^{u}\right) \right], \tag{3.7}$$

where y_{ij} is the label of cell (i, j) in the ground true mask for the region of size $m \times m$, $\hat{y}_{ij}^{u}$ is the predicted value of the same cell in the mask learned for the ground truth class u.

- **Network training**

In terms of the Faster RCNN and Mask RCNN training, the feature extraction backbone network is ResNet-50 and FPN (ResNet-50-FPN) as we illustrated in Section 3.1.1. Given the similarity in between Faster RCNN and Mask RCNN, the Faster RCNN architecture is not trained from scratch individually. Instead, it is trained by using the Mask RCNN architecture with minor modifications: 1) deactivate the mask branch and train classification and bounding box branches only; 2) keep RoIAlign layer, which should have been RoIPool layer in Faster RCNN. The network is initialized with a pre-trained weights on the MS-COCO dataset that provides a large number of pre-labelled images suited for object detection, but no mask segmentation. The classification, bounding box and segmentation branches are further adjusted and trained on the PASCAL-VOC07 dataset that contains labelled mask areas. In details, the training of Mask RCNN is performed in 3 stages, as shown in Figure 3.10, that consist of: i) fixing all layers except the head part of classification, bounding box and mask segmentation branches, and train the head part, ii) unfreezing the layers in ResNet-50 stage 4 and up, train the region proposal part and head part, iii) unfreezing all layers and fine-tuning the whole model.

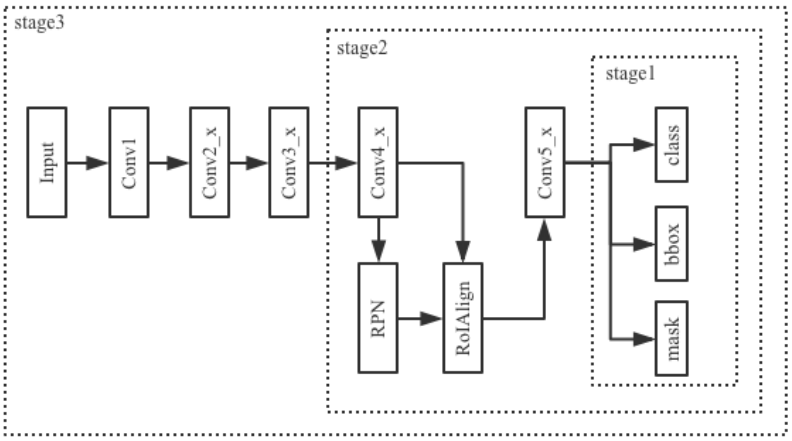


Figure 3.10：Three-stage training strategy of two-stage detectors (Faster/Mask RCNN).

A stochastic gradient descent (SGD) optimizer is used for network training, with starting learning rate of 0.001, weight decay of 0.0001, momentum of 0.9, gradient clip norm of 5.0 and a batch size of 1. Only in the last fine-tuning stage (stage 3) the learning rate is reduced to 0.0001. All training process are performed on an 8G memory NVIDIA Tesla P4 GPU configured in virtual machine supported by Google Compute Engine.

- **Experimental results**

The 100 images from the testing part of PASCAL-VOC07 described above are used for monitoring the performance of Faster RCNN and Mask RCNN at recognizing instances of objects among the 20 classes that it contains and are also used for evaluating instances segmentation of Mask RCNN.

For target object category recognition, Mask RCNN reaches an overall precision over 20 classes of 73.3%, while Faster RCNN scores 62.4% overall precision. In terms of the comparison on target bounding box detection performance, considering a bounding box mAP with a threshold at IoU=0.5 for true positive, as defined in Section 2.4, the results of Mask RCNN and Faster RCNN are shown in Table 3.3 For this experiment, we trained the full Mask RCNN framework, but only the classification and box outputs were used and the mask output was ignored for inference. Besides, for the training of Faster RCNN, we trained a version of Mask RCNN without the mask branch, denoted by “Faster RCNN (with RoIAlign)” in Table 3.3. The results demonstrate that Mask RCNN outperforms Faster RCNN on precision for object category recognition and bounding

box detection. While in terms of detection speed, Mask RCNN and Faster RCNN reaches on average 0.33 and 0.2 seconds per image respectively while testing on a GPU, making the Faster RCNN architecture slightly faster. We also tested the detection speed on a CPU, validating its potential for embedding recognition models on mobile robots. In that case, the average detection speed reaches 3.26s and 3.02s per image with Mask RCNN and Faster RCNN respectively.

Table 3.3: Object category recognition and detection results comparison between Faster RCNN and Mask RCNN over 100 images from PASCAL-VOC07 dataset.

	Backbone network	Category recognition P (%)	Bbox detection $mAP(\%)$	Average detection speed on GPU	Average detection speed on CPU
Faster RCNN (with RoIAlign)	ResNet-50-FPN	62.4	43.85	0.2s	3.02s
Mask RCNN	ResNet-50-FPN	73.3	59.36	0.33s	3.26s

Besides comparing the performance on target object classification and bounding box detection between Faster RCNN and Mask RCNN, the target object instances segmentation performance of Mask RCNN is evaluated on the 100 images from the testing set as well. The mask mean average precision (mAP) for instance segmentation scores 67.0% over the 100 test images. Figure 3.11 presents the qualitative results of Mask RCNN on the class, bounding box and segmentation mask prediction and their comparison with ground truth on various objects.

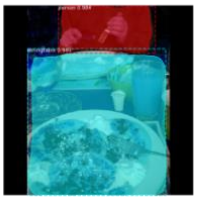
Ground Truth	Prediction Results	Ground Truth	Prediction Results
			
(a) car (yellow); person1(pink); person2 (blue); person3 (purple); person4 (green); person5 (yellow); person6 (red)	(b) car (purple, 97.3%); person1 (yellow, 99.2%); person2 (cyan, 98.3%); person3 (red, 99.8%); person4 (green, 98.9%); person5 (blue, 99.6%); person6 (pink, 97.7%)	(c) person (red); dining table (blue)	(d) person (red, 98.4%); dining table (blue, 99.1%)
			
(e) train (red); person1 (green); person2 (blue)	(f) train (red, 99.6%); person1 (blue, 99.6 %); person2 (green, 91.8%)	(g) person (yellow); potted plant1 (blue); potted plant2 (purple); potted plant3 (green); potted plant4 (red)	(h) person (blue, 99.6%); potted plant1 (yellow, 99.1 %); potted plant2 (purple, 98.9%); potted plant3 (red, 98.2%); potted plant4 (green, 99.4%)
			
(i) horse (red); person (green)	(j) horse (red, 97.7%); person (green, 99.6%)	(k) tvmonitor (red); chair1 (blue); chair2 (green)	(l) tvmonitor (purple, 99.0%); chair1 (green, 97.7%); chair2 (red, 95.6%); false positive dining table (cyan, 61.4%)

Figure 3.11: Mask RCNN prediction results versus ground truth for class recognition with corresponding confidence level, and segmentation results on test samples from the PASCAL-VOC07 dataset.

In Figure 3.11 (a)-(f) the bounding box (dotted lines) predictions closely correspond to the ground truth, while the segmentation mask predictions (colored areas) also match well, tend to be continuous, and form relatively accurate segmentation of the objects detected. Besides, the smaller objects are successfully recognized and detected, which demonstrates that the proposed modification detailed in Section 3.1.2 to increase the number of anchors for candidate proposal scales does work and can prove beneficial for the navigation of collaborative robotic systems. Color codes are not associated to specific classes, which explains the change of color in between matched instances in the ground truth and prediction results images. Confidence level in the class prediction, shown in percentage under each prediction result, also offers high reliability, reaching over 95% in these examples. No false positive detection takes place, and only one person who is not fully visible on the upper right edge of Figure 3.11(b) is missed.

When it comes to situations where the shape of objects is slender, as shown in Figure 3.11(g)-(h), or where an object is partially occluded, as shown in Figure 3.11(i)-(j), the trained Mask RCNN architecture still succeeds to recognize and locate these instances with a high confidence level, but degradation is observed on the precision of the segmentation mask. Besides, in Figure 3.11(k)-(l), the desk, which is not a formal class in the PASCAL-VOC07 dataset, was falsely recognized as a dining table, due to the similarity between the two types of objects. This result actually demonstrates that the Mask RCNN architecture for target object recognition offers a significant level of robustness and is not restricted to visually explicit appearance of instances. This is a particularly suitable characteristic for application in automated task allocation for unmanned systems, as scenarios in which a group of autonomous vehicles are typically involved are far from being perfectly constrained and predefined. The fact that a lower level of confidence (61.4%) is associated with the recognition of a desk as a dining table serves as an indicator to the task allocation scheme to act with care, perhaps even triggering validation by a human supervisor before dispatching a robotic agent.

Given the two convolutional neural networks architectures investigated for object recognition in the context of automated robotic task allocation, the Mask RCNN framework with the proposed modifications on the backbone architecture and region proposal network demonstrated its ability and reliability to recognize and locate instances of objects imaged at different scales and from a

diversity of classes, and with only limited training. Even though it sacrifices the detection speed compared with Faster RCNN, the effect of 0.2s delay can be neglected. The architecture proved its capability to associate accurate confidence level to the recognized classes, which provides an important trade-off between accuracy and flexibility, an important characteristic for the management of unmanned systems operating in less than perfect conditions and uncontrolled environments. Moreover, the modified Mask RCNN architecture reliably generates detailed segmentation mask information to separate instances of objects. This characteristic provides additional benefits over only category classification and determination of a rectangular bounding box surrounding each object, as predicted by the Faster R-CNN architecture. Precise segmentation of an instance of target object leads to efficient path planning for the most competent agent to navigate toward the identified target. While robot navigation research is out of the scope of this thesis and was conducted by a colleague, a precise segmentation of target objects can reduce critical response time in search-and-rescue operations. The segmented area also determines a very specific search region in an image to detect and recognize features at a higher level of resolution, which opens the door to multi-stage object recognition to allocate specialized agents with more specificity.

Although the target objects used in this experiment do not represent ideal match with the potential SAR tasks, this initial experimental investigation of CNN-based two-stage object detector as an entry point to an automated task-agent allocation process, validates the feasibility and reliability of the solution for category classification, bounding box estimation and mask extraction on object instances in scenarios of various complexity. In order to further confirm that observation, the following section will study the performance of a one-stage object detector as a potential speed improvement and on the same public dataset from PASCAL-VOC07.

3.2 One-stage target object detector

In order to expand our study and evaluate the potential of lighter architectures, we also investigated the performance of the state-of-the-art one-stage object detector, YOLO [43][44][45], in the context of target object detection for autonomous robots task allocation. The literature work about YOLO [43], and its improved versions of YOLOv2 [44] and YOLOv3 [45] have shown that

YOLOv3 outperforms YOLO and YOLOv2 both on detection accuracy and detection speed. As a result, we experimentally evaluated YOLOv3 in the considered indoor SAR scenarios application. In this section, we focus on illustrating the basic functioning of YOLOv3. For simplicity, we formulate YOLOv3 as YOLO in the following content.

3.2.1 Architecture description

The architecture of YOLO is presented in Figure 2.4(b). The system design of YOLO is elegant and simple, which reframes the object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Namely, the system takes an image as input, passes it through a neural network named Darknet-53, as detailed in Table 3.4, and generates a vector of bounding boxes and class predictions at the output. Figure 3.12 illustrates the YOLO detection mechanism.

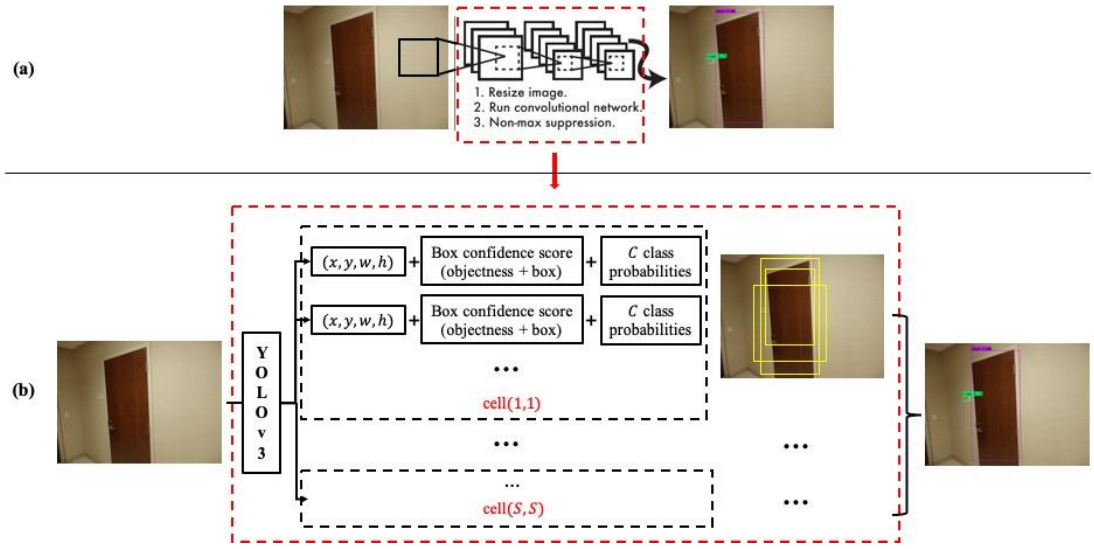


Figure 3.12: YOLO detection scheme. (a) The abstract of YOLO detection system; (b) The details of YOLO detection system.

Table 3.4: The architecture of Darknet-53. It has 53 convolutional layers, which consist of successive 3×3 and 1×1 convolutional layers and some shortcut connections.

Layer type	Output size	Darknet-53	Number of Conv layers
Conv	256×256	$[3 \times 3, 32]$	1
Conv	128×128	$[3 \times 3, 64]$	1
Conv Conv Residual	128×128	$\begin{bmatrix} 1 \times 1, & 32 \\ 3 \times 3 & 64 \end{bmatrix}$	2
Conv	64×64	$[3 \times 3, 128]$	1
Conv Conv Residual	64×64	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3 & 128 \end{bmatrix} \times 2$	4
Conv	32×32	$[3 \times 3, 256]$	1
Conv Conv Residual	32×32	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3 & 256 \end{bmatrix} \times 8$	16
Conv	16×16	$[3 \times 3, 512]$	1
Conv Conv Residual	16×16	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3 & 512 \end{bmatrix} \times 8$	16
Conv	8×8	$[3 \times 3, 1024]$	1
Conv Conv Residual	8×8	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3 & 1024 \end{bmatrix} \times 4$	8
Fully connected (fc)	1×1	average pooling, 80-d fc, softmax	1

YOLO divides the input image into an $S \times S$ grid. Each grid cell makes 3 different scale predictions, and it predicts only one object, of which the center falls inside the grid cell. Besides, it predicts a fixed number (B) of bounding boxes and each box has 4 parameters for the bounding box and 1 box confidence score (objectness), and C conditional class probabilities (one per class

for the likeness of the object class). Figure 3.12(b) details the prediction scheme of YOLO. The box confidence score reflects how likely the box contains an object (objectness) and how accurate is the boundary box. The conditional class probability is the probability that the detected object belongs to a particular class (one probability per category for each cell). Hence, the final prediction of YOLO at each scale has a shape of $(S, S, B \times 5 + C)$. For each prediction box, the class confidence score, which measures the confidence on both the object classification and its corresponding localization, is computed as:

$$\begin{aligned} &class\ confidence\ score(P_r(class_i) \cdot IoU) \\ &= box\ confidence\ score(P_r(object) \cdot IoU) \\ &\times conditional\ class\ probability\big(P_r(class_i|object)\big), \end{aligned} \tag{3.8}$$

where,

$P_r(class_i)$ is the probability of the object belonging to $class_i$;

IoU is the IoU (intersection over union) between the predicted box and the ground truth;

$P_r(object)$ is the probability of the box containing an object;

$P_r(class_i|object)$ is the probability of the object belonging to $class_i$ given that an object is present.

The number of bounding boxes for each grid cell is automatically determined by applying k-means clustering on the training set bounding boxes, where $k = 3$ at each scale is a good tradeoff between model complexity and high recall. Hence, there are 9 clusters in total: $(10 \times 13), (16 \times 30), (33 \times 23), (30 \times 61), (62 \times 45), (59 \times 119), (116 \times 90), (156 \times 198), (373 \times 326)$ [45].

3.2.2 Experimental investigation of one-stage object detectors on public dataset

- **Loss function**

The loss function defined in YOLO is the multi-part loss:

$$L = L_{loc} + L_{confidence} + L_{class}, \tag{3.9}$$

where, each part of loss is defined as:

$$L_{loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [(x_i - x_i)^2 - (y_i - \hat{y}_i)^2]$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 - \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]. \quad (3.10)$$

$$L_{confidence} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2. \quad (3.11)$$

$$L_{class} = \sum_{i=0}^{S^2} \mathbb{I}_{i}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2. \quad (3.12)$$

Table 3.5 defines the variables that are used in the formulation of the loss function.

Table 3.5: Symbols and descriptions of parameters in the loss function.

Symbol	Definition and description
L_{loc}	The loss on localization.
$L_{confidence}$	The loss associated with the confidence score for each bounding box.
L_{class}	The loss on classification probability of a grid cell.
B	The number of bounding box predictors.
S	The number of grid cells.
λ_{coord}	Scale parameter to control the degree to increase the loss from bounding box coordinate predictions.
λ_{noobj}	Scale parameter to control the degree to decrease the loss of confidence score predictions for boxes without objects.
$\mathbb{I}_{ij}^{obj}$	It indicates whether the j-th bounding box of cell i is responsible for the object prediction. If yes, it equals 1; otherwise, it is 0.
$\mathbb{I}_{i}^{obj}$	It indicates whether the cell i contains an object. If yes, it equals 1; otherwise, it is 0.
x, y	The predicted center coordinates of ground truth bounding box.
w, h	The predicted width and height of ground truth bounding box.
$\hat{x}, \hat{y}$	The center coordinates of ground truth bounding box.
$\hat{w}, \hat{h}$	The width and height of ground truth bounding box.
C_i	The predicted confidence score of cell i.
$\hat{C}_i$	The confidence score of cell i.
c	The set of all classes.
$p_i(c)$	The predicted class probability.
$\hat{p}_i(c)$	The probability of whether cell i contains an object of class c.

- **Network training**

For the training of YOLO, we use the same 400 images from the training set of PASCAL-VOC07 which are also used for training two-stage detectors Faster RCNN and Mask RCNN in Section 3.1.4. The feature extraction backbone network is Darknet-53 with pre-trained weights on MS-COCO dataset. The training is performed in 2 stages, as shown in Figure 3.13, that consists of: i) fixing all layers except the head part of classification and bounding box branches, and train the head part, ii) unfreezing all layers and fine-tuning the whole model. With start learning rate of

0.001, weight decay of 0.0005, momentum of 0.9 and a batch size of 3, all training processes are performed on a 8G memory NVIDIA Tesla P4 GPU configured in virtual machine supported by Google Compute Engine as well.

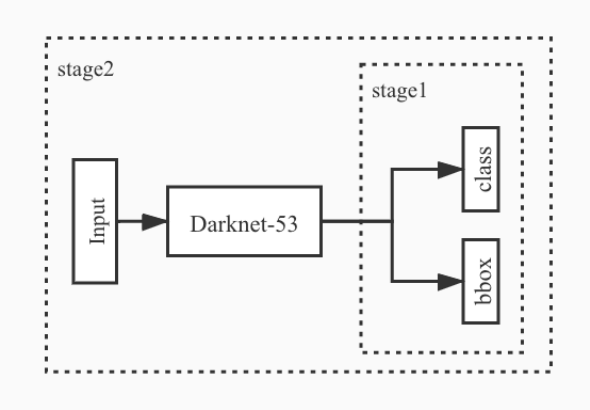


Figure 3.13: Two stages training strategy for one-stage detector (YOLO).

- **Experimental results**

For the performance evaluation of YOLO, we use the same 100 testing images from PASCAL-VOC dataset as for the evaluation of Faster RCNN and Mask RCNN. In terms of the object category recognition, YOLO reaches an overall precision over 20 classes of 66.2%. The bounding box mAP scores 46.52 at IoU =0.5. The average detection speed reaches 0.13s on GPU and 1.06s on CPU. Table 3.6 presents the performance comparison on YOLO versus Faster RCNN and Mask RCNN (Table 3.3).

Table 3.6: Object category recognition and detection results comparison between YOLO, Faster RCNN, and Mask RCNN.

	Backbone network	Category recognition P (%)	Bbox detection $mAP(\%)$	Average detection speed on GPU	Average detection speed on CPU
YOLO	Darknet-53	66.2	46.53	0.13s	1.06s
Faster RCNN (with RoIAlign)	ResNet-50-FPN	62.4	43.85	0.2s	3.02s
Mask RCNN	ResNet-50-FPN	73.3	59.36	0.33s	3.26s

Besides the quantitative evaluation, the qualitative results obtained with the YOLO detector on the same testing samples listed in Figure 3.11 are presented in Figure 3.14.

Input Image	Prediction Results	Input Image	Prediction Results
			
car; person1-6(from left to right)	(b) car (red, 98.5%); person1 (pink, 60.3%); person3 (pink, 93.5%); person4 (pink, 87.4%); person5 (pink, 86.2%); person6 (pink, 98.6%); person2: false negative	(c) person; dining table	(d) person (pink, 83.4%) dining table: false negative
			
(e) train; person1-2 (from left to right)	(f) train (red, 89.8%); person1 (pink, 99.0 %); person2 (pink, 98.9%)	(g) person; potted plant1-4 (from left to right)	(h) person (pink, 95.7%); potted plant1 (green, 75.0 %); potted plant2 (green, 77.3%); potted plant4 (green, 77.7%); potted plant3: false negative
			
(i) horse; person	(j) horse (blue, 99.9%); person (pink, 96.6%)	(k) tvmonitor; chair1-2(from left to right)	(l) tvmonitor (red, 77.3%); chair1 (purple, 93.8%); chair2 (purple, 67.1%)

Figure 3.14: YOLO prediction results for class recognition with corresponding confidence level on test samples from PASCAL-VOC07 dataset.

In Figure 3.14(a)-(b), most target objects in the image are successfully detected with precise bounding box around them and recognized with the confidence score from 60.3% to 98.6%. However, the small target “person2” failed to be recognized, compared with Figure 3.11(a)-(b). In Figure 3.14(c)-(d), recognition of the “dining table” failed with the YOLO detector. This might be caused by the limited training samples of the class “dining table”. All targets in Figure 3.14(e)-(f) are correctly recognized and detected. In Figure 3.14(g)-(h), except for missing the small target “potted plant3”, which is the false negative recognition, other targets are successfully detected with precise bounding box position and recognized with confidence scores all over 75%. In Figure 3.14(i)-(l), all target objects in the images are correctly recognized and detected.

Given this initial experimental comparison on the public PASCAL-VOC07 dataset with two-stage detectors Mask RCNN and Faster RCNN and one-stage detector YOLO, Mask RCNN shows the best capability for precisely recognizing and detecting target objects among the three detectors, while YOLO achieves highest target object detection speed. Both of these advantages have potential to support collaborative robots task allocation. Therefore, we will further conduct comparative experiments between the two-stage detector, Mask RCNN, and the one-stage detector, YOLO, in more realistic situations on our custom dataset with five predefined classes relevant to SAR scenarios.

3.3 Comparative experimental evaluation on custom dataset

In this section, comparative experiments between Mask RCNN and YOLO are conducted in more realistic situations on a custom dataset that we developed for this research which considers five predefined classes that are relevant for indoor SAR scenarios.

3.3.1 Experimental setup specifications

Table 3.7 lists the software and hardware specifications on which our experimental implementation is developed.

Table 3.7: Experimental setup specifications.

Software	
Google Cloud Platform (GCP)	Cloud computing service which provides the virtual machine (VM) with GPU supporting the training for this work
Ubuntu 16.04 LTS	Operating system of virtual machine on GCP for models' training
Mac OS 10.15.4	Operating system for local laptop testing, with 2.9 GHz Dual-Core Intel Core i5 and 8GB memory
Nvidia Tesla P4	Graphics processing unit (GPU) of virtual machine on GCP for accelerating models' training
CUDA 8.0	Toolkit for general computing on GPU
CUDA Deep Neural Network library (cuDNN) 6.0	GPU-accelerated library of primitives for deep neural networks
Python 3.5	Programming language for this implementation
TensorFlow-gpu 1.3	Open-source artificial intelligence library
Keras 2.1.4	Open-source neural-network library
LabelMe	Image polygonal annotation tool
Termius	SSH client that can work on mobile phone
Hardware	
Robot TurtleBot 3 Waffle Pi with Raspberry Pi camera	A little robot used for collecting image data

- **Google Cloud Platform (GCP)**

Google Cloud Platform (GCP) [92] is a suite of computing services offered by Google. It is capable of providing compute engine, which allows us to create a virtual machine comprised of virtual CPU cores. It supports several standard and customized operating systems with several storage options and also offers GPUs integration with stable performance and relatively low fees. Considering the stable performance and the GPU that GCP provides, we created a virtual machine (VM) instance on GCP and set up the development environments for our experimental implementation.

In terms of building a VM instance used for our deep learning related experimental implementation, we firstly configured the machine type by selecting the available options for the number of CPUs, the memory size, the storage options, the operating system, and the GPU type. The operating system of the VM instance created for our implementation is Ubuntu 16.04 LTS. We integrated two CPUs with 7.5 GB memory, a 256 GB standard hard disk drives (HDD), and an Nvidia Tesla P4 GPU. Then, we configured the development environment on this VM instance. The installed CUDA version is 8.0 and cuDNN version is 6.0, which support GPU accelerated computation. The programming language is Python 3.5. The implementation of models are based on the TensorFlow platform [93], which is an end-to-end open source deep learning platform provided by Google. Finally, the access to the VM instance from a local laptop is set through secure shell (SSH) protocol, which enables us to connect and work on the VM by inputting the command line through a terminal window on the local laptop.

- **LabelMe**

LabelMe [94] is a graphical image annotation tool to build image databases for computer vision research, which is capable of presenting many detailed information, such as object instances categories, bounding boxes, polygons, and segmentation masks of object instances. We installed the desktop software version to generate the image samples with ground truth labels that form our custom dataset for the detection models' training and testing.

Figure 3.15 shows a screenshot of the LabelMe annotation tool. The usage of LabelMe is easy. The area under the polygon, such as the rectangle enclosing the sign posted on the wall, is denoted as the colored mask which is the segmentation mask of that object instance. The bounding box (depicted in green) coordinates of that object instance are estimated from all drawing points that the user clicked to draw the closed polygon. The user is free to label as many objects depicted in the image as needed. For example, in Figure 3.15, there are two instances of objects of interest in the image, the sign posted on the wall and the elevator door. The annotation results on an image are category labels and all points of the contour, which are stored in the JSON file format that makes the annotations portable and easy to extend.

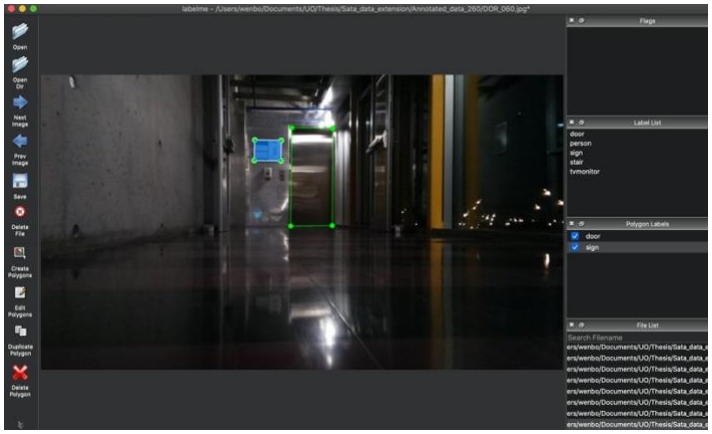


Figure 3.15: A screenshot of LabelMe annotation tool in use.

- **TurtleBot 3 Waffle Pi**

TurtleBot 3 Waffle Pi is a mobile robot with an embedded Raspberry Pi Camera Module V2 as one of its sensor, which serves for capturing images while the mobile robot moves toward a given target object. Figure 3.16 presents the main components of TurtleBot 3 Waffle Pi robot.

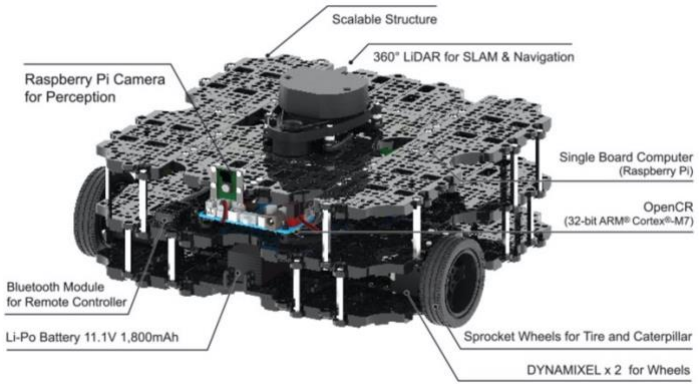


Figure 3.16: Main components of TurtleBot3 Waffle Pi robot [95].

3.3.2 Data preparation

Our initial experimental investigation on Faster RCNN, Mask RCNN and YOLO used the PASCAL-VOC07 dataset, which only contains two categories of interest, “person” and “tvmonitor” (denoted as “fire”), which are relevant for indoor SAR scenarios. In order to obtain additional

representative samples on a larger number of classes (5 considered here) and with more instances, the categories of “door”, “sign” and “stairs” are taken from the MCIndoor20000 dataset [96]. However, a major limitation of MCIndoor20000 dataset is that one image only contains one target object which limits the variety and complexity of training samples. Therefore, we extended the dataset by manually collecting and annotating additional images that include target objects of interest for the 5 classes considered in situations with increased complexity to better represent the indoor SAR scenario application. The latter also provides more realistic testing grounds to evaluate the target object category recognition and detection capabilities of two-stage and one-stage detectors.

- **Data collection**

With regard to collecting images including the predefined 5 classes (person, tvmonitor (fire), door, sign, and stairs) in indoor locations with the goal to expend the dataset, we first listed the required constraints for image sampling to increase the variety of scenes that will make up for the limitations observed on readily available images from PASCAL-VOC07 and MCIndoor20000. Table 3.8 shows the requirements established for data collection. Then, we collected image data by controlling the TurtleBot3 Waffle Pi robot, on which the Raspberry Pi Camera Module V2 serves to capture the images. Considering the flexibility of controlling the Waffle robot while it is moving and capturing images, we used a cell phone as the controller for the robot’s movements and for capturing images through a SSH protocol connection, which is implemented by a SSH client named Termius installed on the mobile phone. Since the Raspberry Pi Camera Module V2 takes at least 2 seconds for the sensor to sense the light levels, and considering that the robot is moving toward the target during the process, the time between successive images was set up to 5 seconds in order to make sure that each image is depicting differences in the scene. During the image collection process, the light conditions varied and included day/night light, sunny/dim conditions, to extend the diversity of ambient conditions and support a more exhaustive validation of the object detection framework under imperfect working conditions, as would be found in SAR scenarios. We also increased the number of classes per image over what was available from the PASCAL-VOC07 and MCIndoor20000 datasets and the complexity of the content in each image. For example, we did not only captured images that contain a single class of object but also took images including multiple classes in a same image, to monitor the robustness of the object detectors on

more complex scenes. Besides, we conducted image collection in several buildings on uOttawa campus to improve the variety of the visual appearance in scenes. The buildings visited include SITE, Colonel By, STEM Complex and Learning Crossroads.

Table 3.8: Required conditions for images to be collected.

Light condition	e.g. day/night, sunny/dim, etc.
Intra-class variation	e.g. person with different poses (standing, sitting, facing, back side, etc.)
Number of classes per image	e.g. one class, multi-class (possible combinations: signs on the door, person aside the tvmonitor (fire), etc.)
Possible indoor location	e.g. SITE, Colonel By, STEM Complex, Learning Crossroads, etc.
Number of images	e.g. at least 50 images per class

- **Data annotation**

After collecting the image data, which are saved in the JPEG format, we manually annotated the target objects in those images with a category label, bounding box and segmentation mask, which serves as ground truth for the models' training. The annotation process involved labelling the category, and selecting the points that correspond to the contour of the object as precisely as possible. The polygon representing the object was generated automatically, under which the area denotes the mask of that object. Figure 3.17 shows an example of a target object annotation page. The annotation results consist of two parts for each target object that occurred in the image: 1) the category label, and 2) the coordinates of all drawing points, which are saved in JSON file format, as shown in Figure 3.18.

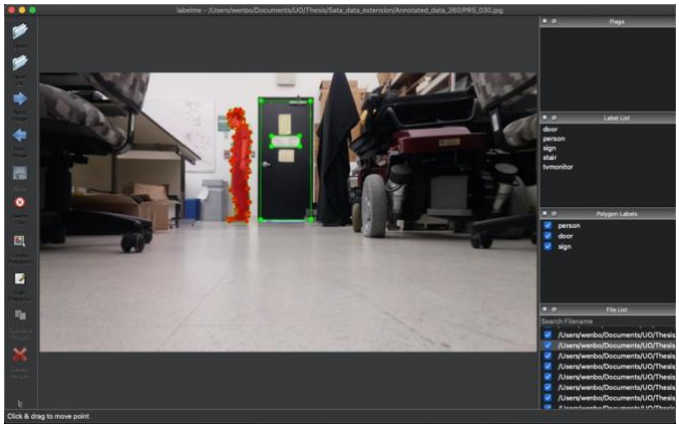


Figure 3.17: Annotation example with three labelled object instances: a person, a door and a posted sign.

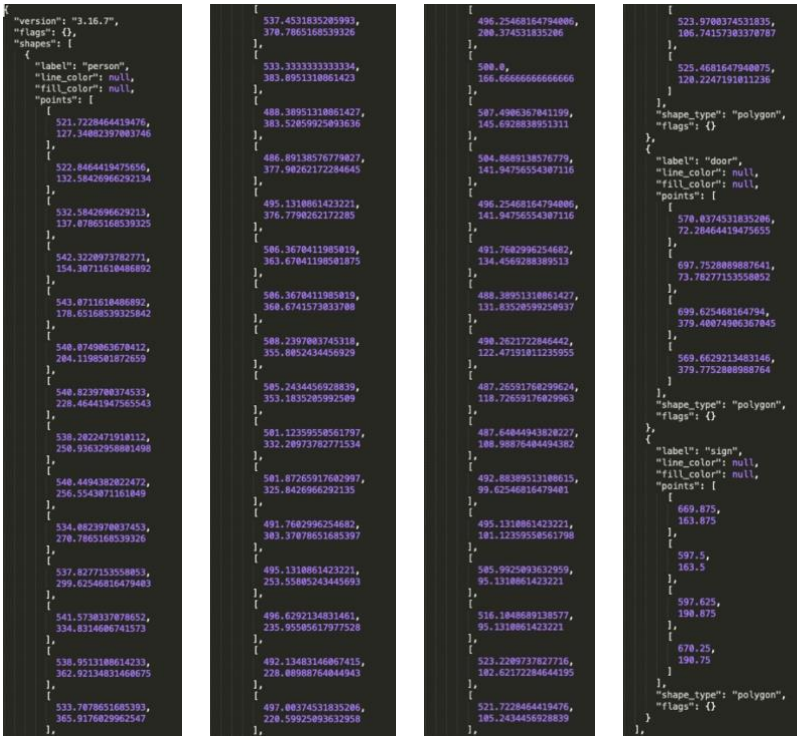


Figure 3.18: Example of annotated results of category label and bounding box in JSON file format.

Based on the original image in JPEG format and its corresponding annotation results in JSON file, we conducted the data processing in order to adjust the format of annotation information and generate the ground truth data (category label, bounding box and segmentation mask) for it to be utilized for training the object detectors. The category label of the target object instance is extracted

from the saved JSON file. The coordinates of the bounding box surrounding each target object are estimated by finding the minimum and maximum of the x and y values $(x_{min}, x_{max}, y_{min}, y_{max})$ from all coordinates of the contours points in the saved JSON file for that object. The rectangular shape generated by four vertices $(x_{min}, y_{min}), (x_{min}, y_{max}), (x_{max}, y_{max}), (x_{max}, y_{min})$ denotes the bounding box. Then the information about the category label and bounding box are rewritten to a new XML file. The mask information of that target is extracted by matching the area under all contour points with the corresponding JPEG image and then saved in the PNG image format. Figure 3.19 shows an example of saved mask information in PNG format. The complete annotation process is summarized in Figure 3.20.

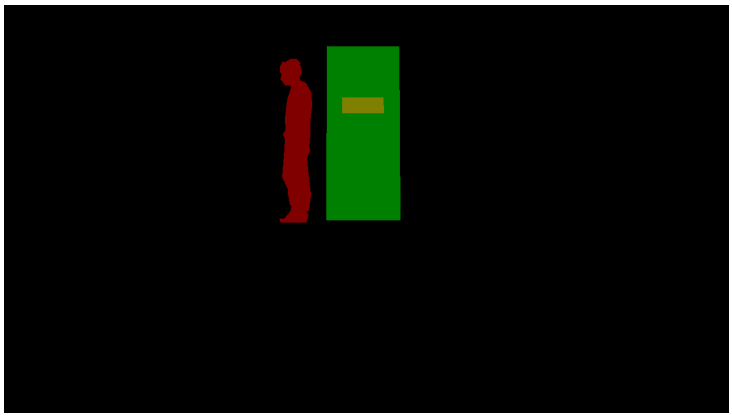


Figure 3.19: Example of annotated mask image in PNG format.

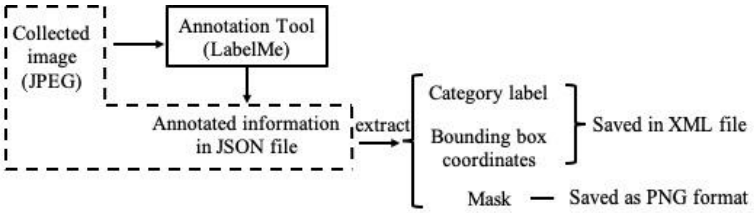


Figure 3.20: Complete annotation process.

- **Dataset description**

As a result, the final built dataset is composed of three parts. One part contains 300 sample images from the MCIndoor20000 dataset with the categories of “door”, “sign” and “stairs”. To that, the second part contains 195 sample images extracted from the PASCAL-VOC07 dataset

exemplifying “person” and “tvmonitor (fire)”. The third part is formed of 310 sample images, representing real indoor environments in which robots may circulate and relatively complex situations, as depicted in Figures 3.15 and 3.17, which were captured by our team. Table 3.9 summarizes the number of images from the different sources. All samples were manually annotated through LabelMe annotation tool, except for images from the PASCAL-VOC07 dataset. The dataset formation process led to a dataset size of 805 image samples representing each of the 5 classes considered. Images including more than one target objects only counts as one image and not as individual instances of each class. In order to support future research in the domain, the resulting dataset is also made available to other researchers [97]. The custom component composed of 310 manually captured images, as well as the annotation of 300 single instances from the MCIndoor20000 represent a contribution of this work.

Table 3.9: Dataset image samples formation.

	Sample source	Number of image
Pre-labelled	Pascal-VOC07	195
Manually labelled	MCIndoor20000	300
	Manually captured	310
	Total	805

When working with deep learning models, a dataset usually splits into three parts: training set, validation set and testing set. The training set is used for model’s learning. The validation set can be regarded as a part of training set since it is used for tuning the parameters during the training procedure and while monitoring its performance. Usually the validation set is considered as part of the training set and the process of proportionally separating validation from training is integrated in the implementation code. The testing set is used for evaluating the final performance of the trained model. As a result, the dataset used in this work was divided into a training set and a testing set with 4:1 ratio, which corresponds to 644 images in the training set and 161 in the testing set. Even though some sample images contain more than one instance of target objects, the number of objects considered for each class considered for training and testing follows a similar distribution in spite of the fact that the separation of the training and testing sets was performed at the image level rather than at the object level. However, the number of objects per class presents a certain

imbalance. Table 3.10 details the image samples distribution, per image and for each of the five classes. Figure 3.21 depicts the object class instances distribution in the training set and in the testing set. The number of objects in the “tvmonitor (fire)” and “stairs” classes is smaller than that of other three categories, which is due to the limited sample objects that we collected in the visited buildings and those available from the public datasets considered. Figure 3.22 shows some image samples from all three sources and for the five classes with their corresponding mask segmentation, where (a) and (b) are image samples from public datasets, and (c) is from our manually collected images. The latter were collected in less perfect conditions than what is found is public datasets with regards to the sizing and positioning of the target objects. Objects are often collected from far, which means they are small in the image. They are also not necessarily centered in the image. These factors contribute to form a custom dataset which is more realistic, and challenging, than what most deep learning solutions are usually trained with and tested on.

Table 3.10: Dataset image samples distribution.

Category	Dataset		Training set		Testing set	
	Number of images	Number of objects	Number of images	Number of objects	Number of images	Number of objects
Sign	145	519	115	394	30	125
Person	206	459	171	374	35	85
Door	195	337	158	276	37	61
Stairs	154	171	119	133	35	38
tvmonitor (fire)	105	195	81	151	24	44
Total	805	1681	644	1328	161	353

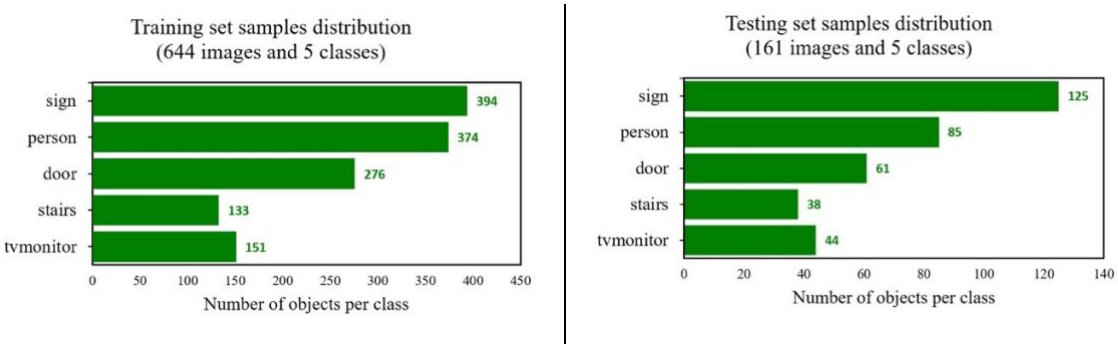
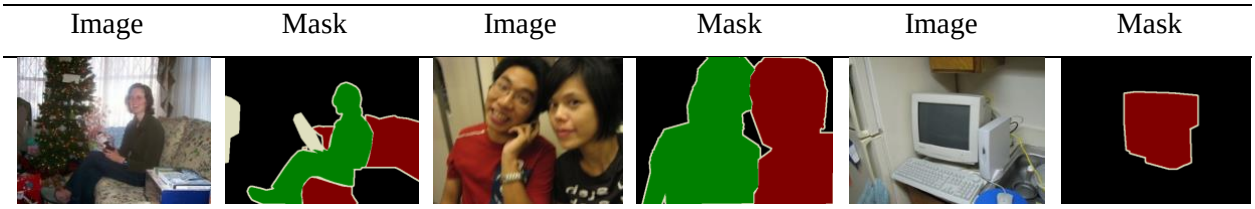


Figure 3.21: Training set and testing set object class instances distribution.



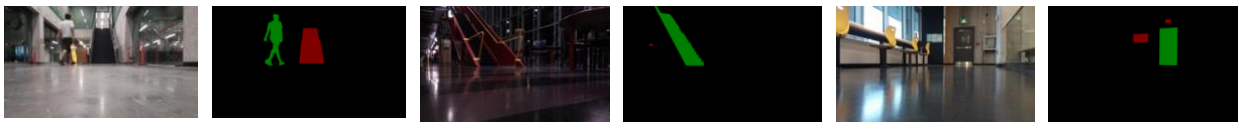
(a) samples of “person” and “tvmonitor” (denotes “fire”) classes from PASCAL-VOC07 dataset

(b) samples of “door”, “sign”, and “stairs” classes from MCIndoor20000 dataset

“tvmonitor (fire)”

combination of “sign”, “person” and “door”

combination of “person” and “tvmonitor (fire)”



combination of “person” and “stairs”

combination of “sign” and “stairs”

combination of “sign” and “door”

(c) samples with variety in content and complexity captured by our team in real environments

Figure 3.22: Sample images for 5 classes contained in the built custom dataset.

Besides the above built dataset, we also collected an additional testing set of 80 image samples which contains the 5 pre-defined classes of interest in the building of uOttawa by controlling the Waffle robot. The collected images in this additional testing set are not contained in our built dataset and are not annotated either with category label or bounding box coordinates. As such they

were not used for training but served for further testing the models' capability when dealing with new environments, as will be reported in the final part of Section 3.3.4 as well as in Chapter 4.

3.3.3 Training details

As concluded from our initial investigation on two-stage and one-stage object detection architectures in Section 3.1 and 3.2, this study retained the models of Mask RCNN and YOLO to monitor their performance on the custom dataset prepared to target SAR robotic scenarios. The same training strategies are used for Mask RCNN and YOLO, presented in Figure 3.10 and Figure 3.13 respectively, were adopted as well as the same training configuration described in the initial experiments using the PASCAL-VOC07 public dataset only. However, this time the custom dataset detailed in Section 3.3.2 was considered. After several training trials, the best performance of Mask RCNN was achieved after: training the first stage for 20 epochs, training the second stage for 40 epochs, and the third stage for 20 epochs. It took about 4.2 hours to train all 3 stages. Table 3.11 reports on the training time for each stage of Mask RCNN. For YOLO, the training of the first stage ran on 30 epochs and the second stage on 20 epochs, which took about 16.6 hours overall for all 50 epochs, as detailed in Table 3.12.

Table 3.11: Training time per stage of Mask RCNN.

	Epochs	Steps per epoch	Average training time per epoch per step
Stage 1	20	483	348ms
Stage 2	40	483	394ms
Stage 3	20	483	433ms

Table 3.12: Training time per stage of YOLO.

	Epochs	Steps per epoch	Average training time per epoch per step
Stage 1	30	193	3s
Stage 2	20	193	11s

3.3.4 Results and performance evaluation

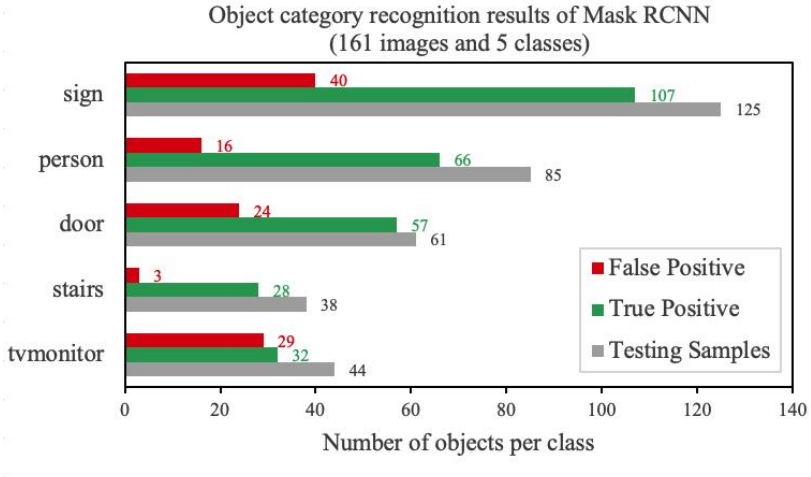
A comparative evaluation of performance for target objects category recognition and detection with Mask RCNN and YOLO was conducted on the testing samples from the custom dataset presented in Figure 3.21, which contains 161 images with 353 objects.

The evaluation is split into four parts: 1) the target object category recognition performance on Precision (P) and Recall (R) at IoU=0.5 for each class; 2) the target object category detection performance on bounding box mean Average Precision (mAP) at IoU=0.5, and the per-class Average Precision (AP); 3) the detection speed; and 4) the target object segmentation performance on mask mean Average Precision, which is only evaluated on Mask RCNN since YOLO does not provide a segmentation capability. The evaluation metrics, Precision (P), Recall (R), Average Precision (AP) and mean Average Precision (mAP) are defined in Section 2.4.

- **Target object category recognition performance**

The target object recognition results of Mask RCNN and YOLO on the testing set at IoU=0.5 are shown in Figure 3.23, which depicts the statistics of correct classification (True Positives, TP) and wrong classification (False Positives, FP). Table 3.13 summarizes the comparison of performance on recognition in terms of Precision (P) and Recall (R).

(a)



(b)

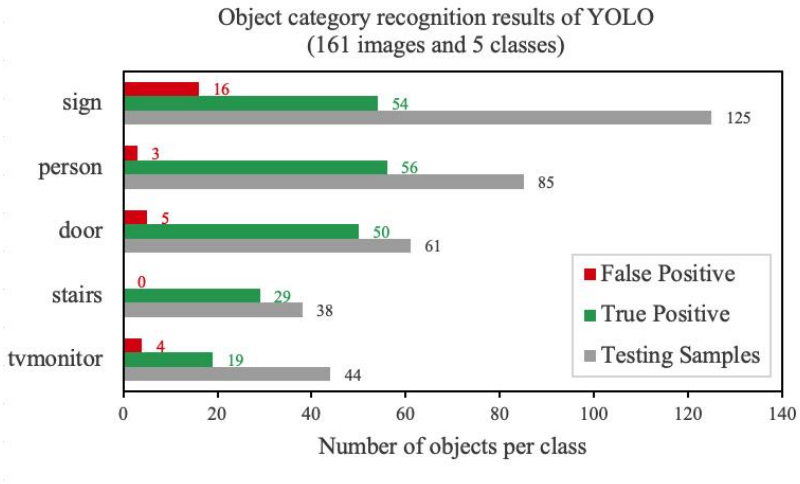


Figure 3.23: Target object recognition results on testing set. (a) Recognition results of Mask RCNN; (b) Recognition results of YOLO.

Table 3.13: Target object category recognition comparison between Mask RCNN and YOLO.

	Mask RCNN		YOLO	
	P (%)	R (%)	P (%)	R (%)
sign	72.79	85.60	77.14	43.20
person	80.49	77.65	94.92	65.88
door	70.37	93.44	90.91	81.97
stairs	90.32	73.68	100	76.32
tvmonitor (fire)	52.46	72.73	82.61	43.18
overall	72.14	82.16	88.14	58.92

For Mask RCNN, the target object category recognition overall precision computed over all 5 classes instances classes is 72.14%, which indicates that the trained recognition model can correctly recognize more than 70% of object instances in these images (TP), while the overall recall is 82.16%, indicating that less than 20% of the instances failed to be detected (FN), which is also illustrated by the differences between the number of testing samples in each class (gray bars) and the total of TP+FP samples (green and red bars). For YOLO, the target object recognition overall precision over all 5 classes is 88.14%, which indicates that the trained detection model can correctly recognize almost 90% of object instances in these images (TP), while the overall recall is 58.92%, indicating that almost 40% of the instances failed to be detected (FN).

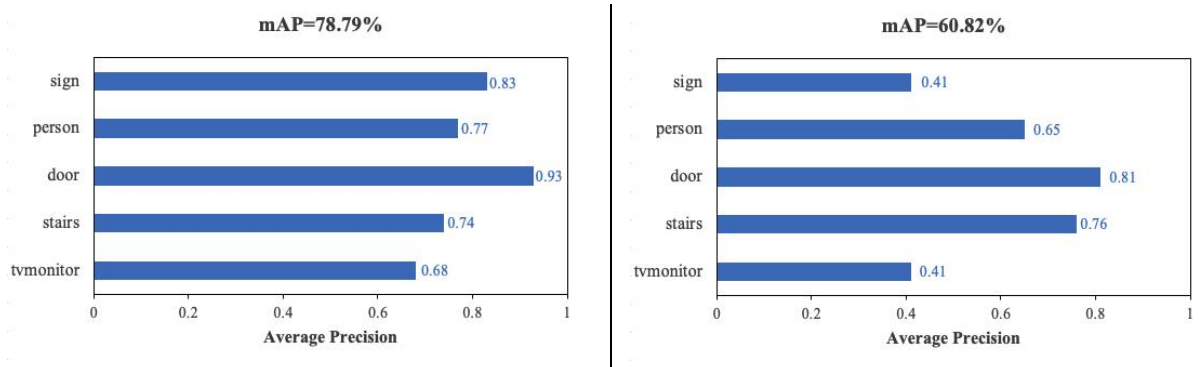
In terms of per-class performance, on the category of “stairs”, YOLO performs better than Mask RCNN with more TP classifications and fewer FP classifications, resulting in both higher Precision and higher Recall. The 100% precision on “stairs” of YOLO means that no other objects are wrongly recognized as “stairs”. However, since the evaluation of “stairs” is based on the smaller number of samples (38) among all categories in the testing set, we cannot conclude that YOLO is systematically performing better on the “stairs” category. On the rest of categories for “sign”, “person”, “door” and “tvmonitor (fire)”, both the TP and FP results recognized by Mask RCNN are higher than recognition results with YOLO, which leads to a higher Precision performance with YOLO but higher Recall performance with Mask RCNN. In particular, the Recall score on “sign” detected by YOLO is about half the Recall of Mask RCNN, which indicates that “sign” instances are often missed (False Negative) by YOLO. That is, in these tests YOLO missed 71 “sign” instances times, while Mask RCNN only missed 18 of them, as shown in Figure 3.23. The main reason to explain this is that YOLO is not a good detector on small targets, while most “sign” targets in the images from our custom dataset are small.

In essence, this evaluation reveals that the Mask RCNN model may be less confident in the recognized classes, but tends to miss fewer target objects among the expected classes, even when object instances only cover a small region of an image. This is a favorable argument for the Mask RCNN architecture, over YOLO, in the context of this thesis, which aims to develop a progressive refinement strategy that will help confirm the nature of a given target object, once it is initially detected. By directing additional image acquisitions and performing iterative rounds of object recognition, the strategy should contribute to improve the overall confidence level on the classification, while leveraging the segmentation capability of the network to drive the iterative acquisition process. As such, a lower precision can be tolerated in the initial discovery of a target object, while missing an object from the beginning may lead to more severe consequences in the robotic task allocation process.

- **Target object bounding box detection performance**

A comparison on target object bounding box detection performance at IoU=0.5 between the two-stage detector Mask RCNN and one-stage detector YOLO on the 161 testing images of our custom

dataset is shown in Figure 3.24 and Table 3.14. Both the bounding box AP for each class and the overall mAP are included.



(a) Mask RCNN

(b) YOLO

Figure 3.24: Target object bounding box detection results on testing set. (a) Results of Mask RCNN; (b) Results of YOLO.

Table 3.14: Results comparison on bounding box mAP between Mask RCNN and YOLO.

	mAP (%)	sign	person	door	stairs	tvmonitor (fire)
Mask RCNN	78.79	**82.75**	**76.52**	**93.07**	73.59	**68.03**
YOLO	60.82	40.84	64.60	81.31	**76.32**	41.49

The bounding box mAP achieved with Mask RCNN is 78.79%, while the YOLO scores 60.82%. On the categories of "sign", "person", "door" and "tvmonitor (fire)", Mask RCNN scores 10-40% higher than YOLO. However, on the "stairs" category, YOLO achieves a slightly higher mAP than Mask RCNN. In this case again, the exception happens on the category that offers the smallest number of testing samples (38 only) among all classes. As such, a strong conclusion on the fact that YOLO performs better than Mask RCNN cannot be derived, more especially that it is based on a difference of less than 3%.

This evaluation shows that Mask RCNN globally performs better on target object bounding box detection than YOLO. This indicates that Mask RCNN provides more accurate localization of

recognized targets within the field of view of a camera. A more accurate localization of the target object in the image will better support the robots with embedded vision sensors to navigate efficiently toward an object of interest, and therefore contribute to iteratively capture images of higher quality and better refine the confidence in target object recognition via the progressive refinement strategy that is developed in the context of this thesis and will be detailed in Chapter 4.

- **Target object detection speed performance**

To evaluate the relative detection speed of the two architectures, we still used the same 161 test images from the custom dataset. For this part, the evaluation was conducted on both the virtual machine with Tesla P4 GPU, on which the models were also trained, and the local CPU of a laptop, as detailed in Section 3.3.1. For the detection speed on GPU, Mask RCNN achieves an average detection speed of 0.45s for each image, while YOLO takes 0.23s for each image. While running on the CPU, the average detection time per image of Mask RCNN and YOLO is 3.05s and 1.59s, respectively. Table 3.15 summarizes the detection speed performance.

Table 3.15: Object detection speed comparison between Mask RCNN and YOLO.

	Backbone network	Detection speed on GPU (second per image)	Detection speed on CPU (second per image)
Mask RCNN	ResNet-50-FPN	0.45s	3.05s
YOLO	Darknet-53	0.23s	1.59s

It is worth noticing that the detection time is not significantly influenced by the number of object instances in a given image. The major factor that affects detection performance is whether the object detector architecture is implemented on massive computing resources (GPU) or on lighter processing resources (CPU). This is an important factor in the context considered in this research, as the goal is to ultimately implement the detection stage on autonomous robots. Current large-scale GPU resources are too bulky and consume too much energy to fit on battery-powered robotic platforms. Therefore, attention must be paid to detection speed on more accessible and efficient CPU processors. Table 3.15 reveals that the required computing time remains tractable even on a CPU, while YOLO clearly offers an advantage over Mask RCNN from this perspective on both GPU and CPU processors by reducing detection time in half.

- **Target object segmentation performance**

The performance of target object instance segmentation was evaluated on Mask RCNN only, since YOLO does not support the generation of a mask segmentation output for each individual instance. The mask mAP of Mask RCNN scores 66.8% on the testing set. Figure 3.25 presents samples of qualitative results of Mask RCNN with confidence level on the class, bounding box location in dotted lines and segmentation mask region prediction for the five categories considered. It also includes the corresponding prediction (confidence and bounding box) of YOLO on the same input images, but without mask segmentation.

Input Image	Prediction Results Mask RCNN	Prediction Results YOLO
		
(a) door	(a.1) door (red, 95.2%)	(a.2) door (purple, 99.3%)
		
(b) tvmonitor (fire)	(b.1) tvmonitor (fire) (red, 91.2%)	(b.2) tvmonitor (fire) (red, 83.3%)
		
(c) sign	(c.1) sign (red, 99.7%)	(c.2) sign (green, 100%)
		
(d) stairs	(d.1) stairs (red, 99.2%)	(d.2) stairs (yellow, 96.9%)

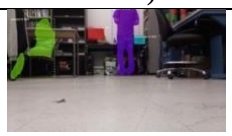
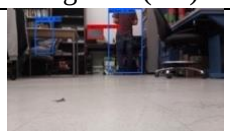
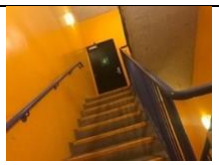
		
(e) person1-2 (from left to right)	(e.1) person1 (cyan, 99.9%) person2 (red, 96.6%)	(e.2) person1(blue, 99.8%) person2 (blue, 60.8%)
		
(f) person tvmonitor1-2(fire) (from left to right)	(f.1) person (cyan, 99.9%) tvmonitor1(fire) (red, 95.3%) tvmonitor2 (fire) (purple, 100%)	(f.2) person (blue, 91.6%) tvmonitor2 (fire) (red, 96.7%) tvmonitor1 (fire): false negative (FN)
		
(g) person1-2 (from left to right) tvmonitor1-2 (fire) (from left to right)	(g.1) person1 (yellow, 99.2%) person2 (purple, 99.6%) tvmonitor1 (fire) (red, 99.7%) tvmonitor2 (fire) (cyan, 99.2%)	(g.2) person1 (blue, 96.0%) person2 (blue, 98.6%) tvmonitor1 (fire) (red, 87.7%) tvmonitor2 (fire): FN
		
(h) stairs sign door	(h.1) stairs (red, 83.4%) sign: FN door: FN	(h.2) stairs (yellow, 97.7%) sign (green, 87.4%) door: FN
		
(i) door sign1-5 (from left to right)	(i.1) door (yellow, 87%) sign1 (purple, 99.6%) sign2 (red, 99.8%) sign4 (blue, 70.5%) sign5 (green, 100%) sign3: FN	(i.2) door (purple, 99.8%) sign1 (green, 92.9%) sign3 (green, 98.9%) sign4 (green, 74.8%) sign2: FN sign5: FN

Figure 3.25: Qualitative prediction results of Mask RCNN and YOLO on testing images from custom dataset.

In Figure 3.25(a)-(d), there is only one target in each image and both Mask RCNN and YOLO correctly detect and recognize the target. The prediction results of Mask RCNN (a.1, b.1, c.1, d.1) consist of a bounding box (dotted rectangle) with mask segmentation (color areas), and the classification score is shown in percentage under each bounding box. On the other hand, the prediction results of YOLO (a.2, b.2, c.2, d.2) present the bounding box as a solid rectangle without mask segmentation information, and show the class confidence level on the top of each bounding box. For clarity the respective confidence levels are also reported in the figure caption for each sample. The confidence level in the class prediction with Mask RCNN and YOLO offer high reliability on such simple images, reaching over 91% and 83%, respectively. The bounding box predictions closely cover the targets with both Mask RCNN and YOLO, and the mask predictions of Mask RCNN also match the target well.

Figure 3.25(e)-(i) show situations where multiple targets are contained in the image. For Figure 3.25(e), there are two targets (two persons) appearing in the image. In that case, both Mask RCNN and YOLO detect all targets with precise bounding boxes, while Mask RCNN shows better performance on the classification of the two targets than YOLO. The confidence level for two targets (person1 and person2) predicted by Mask RCNN is 95%, while YOLO classifies the relatively small target (person2) with a lower level of confidence (61%). For Figure 3.25(f) and (g), there are three targets and four targets in the images, respectively. Mask RCNN successfully detects all targets with precise bounding boxes and high classification scores, as shown in Figure 3.25(f.1) and (g.1), while YOLO fails to detect the relatively small targets of tvmoniter1 (fire) in Figure 3.25(f.2) and tvmonitor2 (fire) in Figure 3.25(g.2). These missing targets are False Negative (FN) detections, which are caused by the fact that the Recall (43.18%, described in Table 3.13) and AP (41.49%, described in Table 3.14) for the category “tvmonitor (fire)” perform relatively poorly with YOLO. For Figure 3.25(h) and (i), FN detections occur on both Mask RCNN and YOLO predictions. Only the “stairs” is successfully detected and the “door” and “sign” fail to be detected by Mask RCNN in Figure 3.25(h.1), while the “sign” is correctly detected by YOLO in Figure 3.25(h.2), and only the “door” is missed with the latter model. In Figure 3.25(i.1), only one target (sign3) fails to be detected by Mask RCNN among six targets in the image. In contrast, YOLO fails to detect two smaller targets (sign2 and sign5).

Besides the evaluation conducted on the testing set from the built dataset, we also performed a qualitative evaluation of Mask RCNN and YOLO performance on an additional testing set containing 80 images which were acquired in unvisited areas of building and not considered in the initial training and testing phase, as introduced in Section 3.1.1. The objective was to further study Mask RCNN and YOLO's object category recognition and detection capabilities when dealing with new environments. The same pre-trained Mask RCCN and YOLO models, as considered previously, were used. Figure 3.26 presents examples of the detailed prediction results.

In terms of the prediction results on additional testing images, both single target and multiple targets cases are successfully detected by Mask RCNN and YOLO. For Figure 3.26(a)-(b), all targets in the images are successfully detected and recognized by Mask RCNN and YOLO. For Figure 3.26(c)-(d), the confidence levels on small targets for the "sign" class in (c.1) and (d.1) detected by Mask RCNN are not as high as for other targets, like "door" and "stairs" in (c.1) and five "person" and "stairs" in (d.1) which are of a larger dimension in images. However, Mask RCNN successfully recognizes all targets that occurred in the image. On the other hand, YOLO more systematically fails to detect the smaller targets, including the "sign" in (c.2), and "person1" and "sign" in (d.2). Other than on the small targets that are anyhow missed by YOLO, Mask RCNN detects all other targets with confidence levels over 95%, with precise bounding boxes and accurate matched mask areas. YOLO shows great bounding boxes results as well, but the confidence levels on detected targets varies from 87% to 99%.

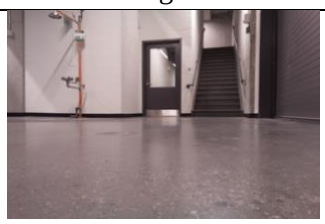
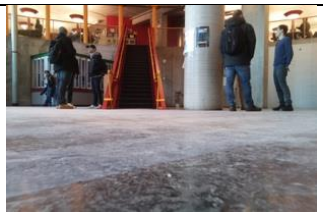
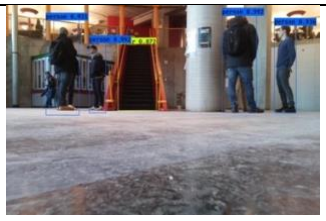
Image Input	Detection Results Mask RCNN	Detection Results YOLO
		
(a) stairs	(a.1) stairs (red, 99.7%)	(a.2) stairs (yellow, 70.3%)
		
(b) person sign	(b.1) person (cyan, 98.8%) sign (red, 67.2%)	(b.2) person (blue, 78.5%) sign (green, 89.6%)
		
(c) sign door stairs	(c.1) sign (green, 68.8%) door (blue, 99.9%) stairs (red, 99.9%)	(c.2) door (purple, 94.7%) stairs (yellow, 94.4%) sign: FN
		
(d) persons1-5(from left to right) stairs sign	(d.1) person1 (yellow, 96.8%) person2 (purple, 95.2%) person3 (light blue, 98.9%) person4 (red, 95%) person5 (green, 97.4%) stairs (blue, 99.4%) sign (light green, 80.6%)	(d.2) person2 (blue, 93.1%) person3 (blue, 99.2%) person4 (blue, 99.2%) person5 (blue, 93.6%) stairs (yellow, 87.2%) sign: FN person1: FN

Figure 3.26: Prediction results on additional testing images.

The qualitative results on the testing images from the built dataset and the additional testing set with 5 relevant pre-defined classes to SAR scenarios operation show that both Mask RCNN and YOLO can predict precise bounding boxes with high confidence level for each predefined class. However, Mask RCNN not only predicts the bounding boxes with category scores as YOLO does, but also predicts the mask areas. Besides, for small objects, Mask RCNN performs more robustly than YOLO, which is a favorable factor in the context of this thesis. That is, when a robot with an embedded camera is still far from a target, which leads to the object appearing small in the image, Mask RCNN is more likely to reliably detect the object and perform a relatively accurate recognition, which is essential. Moreover, compared with YOLO which detects objects without differentiating individual instances of same category, Mask RCNN is able to differentiate each detected instances with accurate segmentation masks even when they are of the same class. This can improve the efficiency on allocating the tasks among specialized robots when there are multiple targets being detected and recognized in a same image.

3.4 Summary

In this chapter, the classical two-stage target object detectors Faster RCNN and Mask RCNN, and the one-stage target object detector YOLO were examined in details. We first conducted an experimental investigation using samples from the public PASCAL-VOC07 dataset to evaluate the feasibility and potential of two-stage detectors, on which we propose valuable adaptations and modifications on backbone and region proposal network to better fit the architectures with the context of this thesis. The initial experimental results show that Mask RCNN generally performs better on category recognition for detected objects. However, it is slower than Faster RCNN. As such, we investigated the one-stage YOLO architecture as an alternative, simpler and potentially faster solution. This led to comparative experiments with Mask RCNN and YOLO on a custom dataset which represents realistic scenes with 5 predefined classes of target objects relevant for SAR robotic scenarios. The results show that both Mask RCNN and YOLO are capable of recognizing and detecting targets at multiple scales by generating confidence scores and bounding boxes which estimate the positions of targets in the image. However, when smaller targets are involved, which occurs often in the context considered in this thesis, Mask RCNN tends to perform

better and more reliably. On the other hand, the detection speed of YOLO performs twice faster as Mask RCNN both on GPU and on CPU. The latter implementation provides additional potential for embedding target object recognition on mobile robots when exploring the environments while iteratively guiding the robot to move toward the target with increased confidence, as will be detailed in Chapter 4.

Chapter 4 Object Recognition with Progressive Refinement

In the previous chapter, we initially validated the feasibility of the adapted Mask RCNN and YOLO architectures for detecting and recognizing individual object instances on images depicting objects at multiple scales and in environments with variable complexity. These conditions were defined for the object detection and recognition process to support collaborative robots task allocation in the context of indoor search-and-rescue (SAR) scenarios.

This chapter investigates the concept of progressively refining the confidence on detected target objects with the goal to further increase the reliability and robustness of mobile robot navigation when it moves toward a given target object to perform a task on it. The general idea, presented in Figure 4.1, consists of guiding the robot by iteratively detecting and recognizing the object, with increasing confidence on the recognition, over a sequence of images captured by a camera mounted on the robot. In spite of the performance reported in Chapter 3 with classical CNN object detectors, the considered detectors are not intrinsically designed to support such an iterative and progressive refinement process. Therefore, this chapter proposes an original design that leverages data fusion to create a formal progressive refinement strategy. The approach leverages previous and sequential detection results as a priori evidence in order to iteratively improve the current prediction and refine the estimation of the confidence scores on the detected target objects.

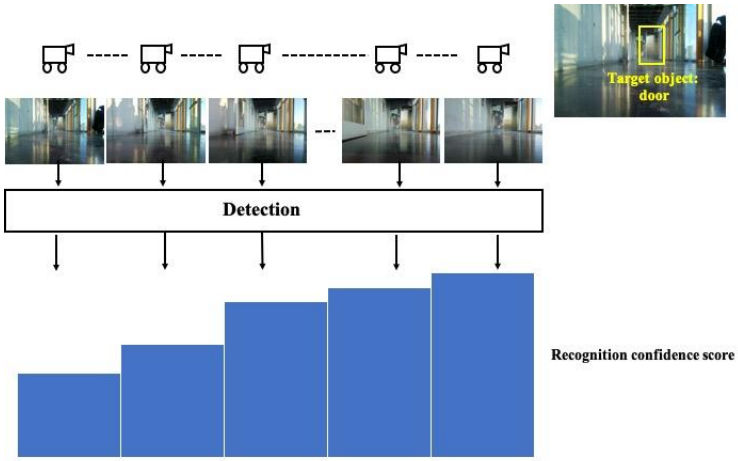


Figure 4.1: The general concept of progressive refinement on target recognition.

Given that it is difficult to estimate the 3D position of an object, which serves as a target for navigating a robot in the real word, by only relying on the detected 2D image coordinates from single camera mounted on the robot, we only consider the case of static scenes in this work. That is, target objects are not expected to move in between frames, while only the camera mounted on the robot moves iteratively toward the target object. As a result, we consider a sequence of successive images captured from a mobile viewpoint on a static target and deploy a progressive refinement recognition strategy. We also experimentally evaluate its performance in the context of collaborative robots coordination.

In Section 4.1, we first test the performance of target detectors described in Chapter 3 without applying the progressive refinement model but while processing successive images captured by a mobile robot equipped with an embedded camera. This serves as the comparative basis against which progressive refinement recognition will be compared. Section 4.2 presents the proposed progressive refinement strategy in details. Finally, we study the performance on the proposed method in Section 4.3.

4.1 Target object recognition on image sequences without progressive refinement

In order to test the performance of the Mask RCNN and YOLO object detectors on successively captured images, we selected five sets of successive images that satisfy the conditions of successively captured scenes in a given environment, while getting a closer view of a same set of object. Each set had 4 to 6 successive images with an estimated distance variation of 3 to 5 meters relative to the target from the beginning to the end of the sequence. These images were extracted from the additional testing set of 80 unlabelled images described at the end of Section 3.3.2.

The main objective of this experiment is to study the impact on the recognition stage of various sizes of target objects in the input test image, and to evaluate the relevance of adopting a progressive refinement strategy that would leverage a feedback control loop from the object recognition stage into the robot path planning and navigation controller to increase the confidence level on detected target objects via an opportunistic image collection approach. Figures 4.2 to 4.6

show the prediction results obtained with Mask RCNN and YOLO on the five sets of successively captured images for different target object recognition cases. In these examples, color codes are not associated to specific classes, which explains the change of color in between same instances in different predicted images.

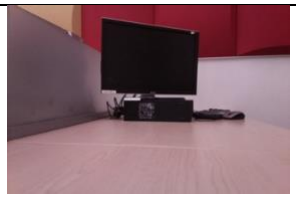
Image Input	Recognition Results Mask RCNN	Recognition Results YOLO
 (1-1) tvmonitor (fire)	 (1-1.a) tvmonitor (fire) (red, 99.8%)	 (1-1.b) tvmonitor (fire) (red, 94.9%)
 (1-2) tvmonitor (fire)	 (1-2.a) tvmonitor (fire) (red, 99.7%)	 (1-2.b) tvmonitor (fire) (red, 67.1%)
 (1-3) tvmonitor (fire)	 (1-3.a) tvmonitor (fire) (red, 97.5%)	 (1-3.b) tvmonitor (fire) (red, 77.4%)
 (1-4) tvmonitor (fire)	 (1-4.a) tvmonitor (fire) (red, 99.4%)	 (1-4.b) tvmonitor (fire) (red, 85.8%)

Figure 4.2: Prediction results of Mask RCNN and YOLO on successive images from the first sequence of testing images.

In Fig. 4.2(1-1)-(1-4), only one target of class “tvmonitor (fire)” appears in the captured images while it progressively grows in size as the camera gets closer to the object. Both Mask RCNN and YOLO correctly detect the target object with bounding boxes and provide classification scores in all images. Mask RCNN provides consistent confidence levels from 97% to 99% at all scales, while YOLO reaches results with more variability in the confidence levels, which range from 67% to 95% and increase as the object gets closer and appears larger. This behavior is coherent with our initial observations in Section 3.2 about the weaknesses of YOLO when dealing with small size objects in the input image.

In Fig. 4.3(2-1)-(2-5), Mask RCNN initially detects two targets of classes “door” and “stairs” respectively, and from far away in (2-1.a), (2-2.a) and (2-4.a). Then as the robot gets closer to the targets, the “door” gets truncated as it partially goes out of the field of view. At the end, Mask RCNN fails to detect the “door” in (2-5.a). The classification scores for the “stairs” detected by Mask RCNN reaches over 95% for all five images in the sequence, while the confidence levels of the detected “door” in the first three images in the sequence vary from 68% to 75%. On the other hand, YOLO only successfully detects the target “stairs” from the beginning in (2-1.b) and to the end in (2-5.b), with confidence scores varying from 64% to 97%. YOLO completely misses the door, potentially because it is made of glass and does not exhibit very strong features due to transparency.

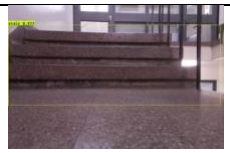
Image Input	Recognition Results Mask RCNN	Recognition Results YOLO
		
(2-1) stairs door	(2-1.a) stairs (cyan, 99.1%) door (red, 75.1%)	(2-1.b) stairs (yellow, 65.0%)
		
(2-2) stairs door	(2-2.a) stairs (red, 98.7%) door (cyan, 72.4%)	(2-2.b) stairs (yellow, 64.1%)
		
(2-3) stairs door	(2-3.a) stairs (red, 99.5%)	(2-3.b) Stairs (yellow, 63.8%)
		
(2-4) stairs door	(2-4.a) stairs (green, 98.1%) door (cyan, 68.1%)	(2-4.b) stairs (yellow, 93.3%)
		
(2-5) stairs	(2-5.a) stairs (red, 96.6%)	(2-5.b) stairs (yellow, 97.2%)

Figure 4.3: Prediction results of Mask RCNN and YOLO on successive images from the second sequence of testing images.

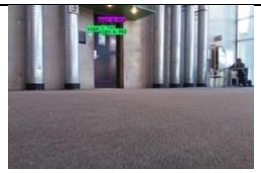
Image Input	Recognition Results Mask RCNN	Recognition Results YOLO
 (3-1) door sign1-2(from left to right) person	 (3-1.a) door (red, 99%) sign2 (green, 90.1%) person (blue, 79.8%) sign1: FN	 (3-1.b) door (purple, 98.6%) sign (green, 69.0%)
 (3-2) door sign1-2(from left to right) person	 (3-2.a) door (red, 99.5%) sign2 (blue, 97.6%) person (green, 99.5%) sign1: FN	 (3-2.b) door (purple, 98.5%) sign (green, 89.4%)
 (3-3) door sign1-2(from left to right) person	 (3-3.a) door (green, 99.9%) sign1(red, 99.4%) sign2 (cyan, 98.7%) person (purple, 96.9%)	 (3-3.b) door (purple, 95.1%) sign1 (green, 90.3%) sign2 (green, 70.5%)
 (3-4) door sign1-2(from left to right) person	 (3-4.a) door (red, 98.3%) sign1 (purple, 80.3%) sign2 (green, 99.5%) person (cyan, 95.2%)	 (3-4.b) door (purple, 90.5%)

Figure 4.4: Prediction results of Mask RCNN and YOLO on successive images from the third sequence of testing images.

In Fig. 4.4(3-1)-(3-4), the categories of “door”, “sign” and “person” are all included in the same scene. Mask RCNN successfully detects all three categories in (3-1.a), (3-2.a), (3-3.a) and (3-4.a), with the confidence level of the “sign2” target being reinforced from (3-1.a) to (3-4.a) as the camera comes closer to the target object. Figure 4.7(d) details the trend observed in the confidence level. Conversely, the small target “sign1” detected in the later stage shown in (3-3.a) and (3-4.a) was initially missed in (3-1.a) and (3-2.a). In this case, getting closer to the scene, while being attracted mainly by the detected door allows the system to detect finer target objects of interest and complement the understanding of the scene. As for the application of YOLO in this scene, the “person” category fails to be detected in all images ((3-1.b), (3-2.b), (3-3.b) and (3-4.b)) as it is further to the side of the image and somewhat dissimulated among other objects that do not represent targets of interest in the application considered. The “door” category is successfully detected in all images ((3-1.b), (3-2.b), (3-3.b) and (3-4.b)). Moreover, in (3-1.b) and (3-2.b), one of the “sign” targets posted on the door is correctly detected. Then, in (3-3.b) YOLO temporarily successfully detects the two “sign” targets, but drops both of them in (3-4.b) even though they are more visible than before.

In Fig. 4.5(4-1)-(4-4), Mask RCNN correctly detects two targets “sign” and “door” in four successive images, (4-1.a), (4-2.a), (4-3.a) and (4-4.a), with confidence levels varying from 90% to 99% and 82% to 99%, respectively. From the target objects detected by YOLO in (4-1.b) only the “door” category is recognized while it fails to detect the “sign” target. From slightly closer viewpoints, (4-2.b) and (4-3.b), YOLO correctly detects both targets. However, in (4-4.b) YOLO not only detects “sign” and “door” as expected, but also wrongly detects a Purell distributor as a “sign”, which represents a False Positive (FP) detection.

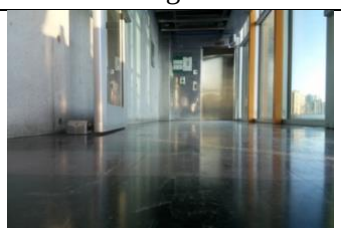
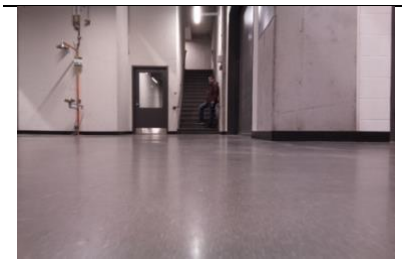
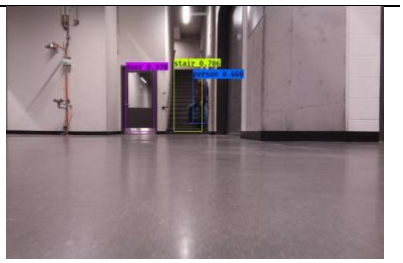
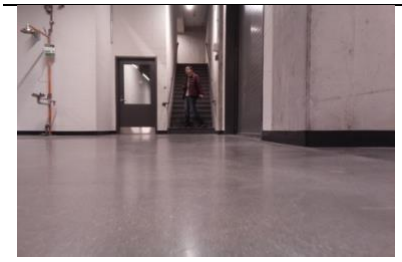
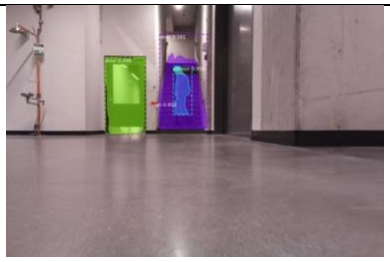
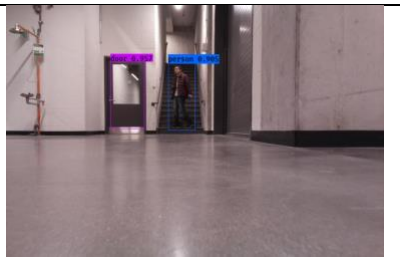
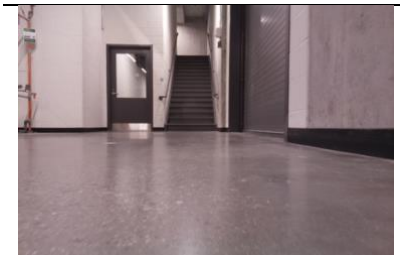
Image Input	Recognition Results Mask RCNN	Recognition Results YOLO
		
(4-1) door sign	(4-1.a) door (red, 98.7%) sign (cyan, 95%)	(4-1.b) door (purple, 93.8%)
		
(4-2) door sign	(4-2.a) door (cyan, 99.6%) sign (red, 99.9%)	(4-2.b) door (purple, 69.7%) sign (green, 90.8%)
		
(4-3) door sign	(4-3.a) door (red, 91.6%) sign (cyan, 90.2%)	(4-3.b) door (purple, 93.8%) sign (green, 99.0%)
		
(4-4) door sign	(4-4.a) door (red, 82.0%) sign (cyan, 95.9%)	(4-4.b) door (purple, 99.5%) sign1 (green, 98.3%) sign2 (green, 67.3%): FP

Figure 4.5: Prediction results of Mask RCNN and YOLO on successive images from the fourth sequence of testing images.

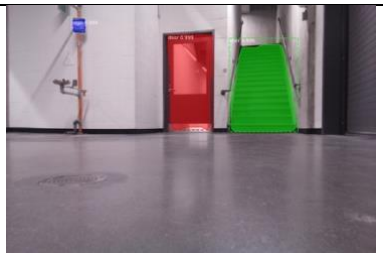
Image Input	Recognition Results Mask RCNN	Recognition Results YOLO
		
(5-1) sign door stairs person	(5-1.a) sign: FN door (red, 99.9%) stairs (green, 69.1%) person (cyan, 61.8%)	(5-1.b) sign: FN door (purple, 97.0%) stairs (yellow, 70.6%) person (blue, 66.0%)
		
(5-2) sign door stairs person	(5-2.a) sign: FN door (green, 99.6%) **stairs (purple, wrongly detected as door with 78.1%): FP** person (blue, 95.8%)	(5-2.b) sign: FN door (purple, 95.7%) stairs: FN person (blue, 90.5%)
		
(5-3) sign door stairs	(5-3.a) sign: FN door (red, 100%) stairs (cyan, 98.5%)	(5-3.b) sign: FN door (purple, 96.8%) stairs (yellow, 79.8%)

(5-4)
sign
door
stairs

(5-4.a)
sign (green, 68.8%)
door (blue, 99.9%)
stairs (red, 99.9%)

(5-4.b)
sign: FN
door (purple, 94.7%)
stairs (yellow, 94.4%)

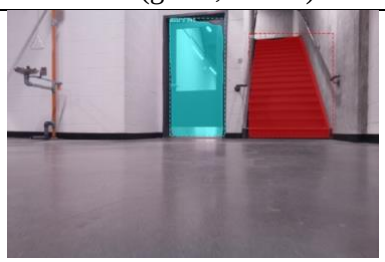
(5-5)
sign
door
stairs



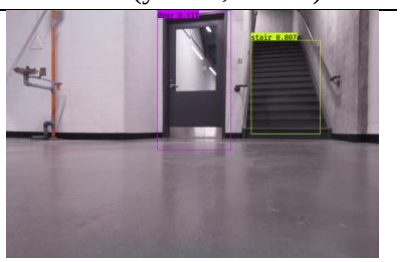
(5-5.a)
sign (blue, 98.9%)
door (red, 99.9%)
stairs (green, 99.6%)

(5-5.b)
sign: FN
door (purple, 98.9%)
stairs (yellow, 76.0%)

(5-6)
door
stairs



(5-6.a)
door (cyan, 99.9%)
stairs (red, 99.7%)



(5-6.b)
sign: FN
door (purple, 91.1%)
stairs (yellow, 80.7%)

Figure 4.6: Prediction results of Mask RCNN and YOLO on successive images from the fifth sequence of testing images.

Finally, in Fig. 4.6(5-1)-(5-6), the object from the “door” category is correctly detected in six successive images, (5-1.a) to (5-6.a), by Mask RCNN with confidence levels all over 99%. Only minor decrease in the confidence occurs from (5-1.a) to (5-2.a) when the robot moves closer to the targets. However, the confidence levels of “stairs” vary significantly in the first two successive images, (5-1.b) and (5-2.b), since it is wrongly detected as a “door” in (5-2.b), which is a FP

detection. Figure 4.7(h) details the trend observed in the confidence level. A major reason for this FP detection might be the occlusion between “person” and “stairs”, where the “person” stands in the center of the “stairs” and most areas are overlapped. As the viewpoints for these images gets closer to the objects, the “stairs” are redetected and recognized in the following four successive images, from (5-3.b) to (5-6.b), with the confidence score over 98%. The category of “sign” occurred in the first five images with small size, but it is only detected in the last two images, (5-4.a) and (5-5.a), where the positions of “sign” are more centered and with larger size in the images than in the previous three images. The category of “person” occurs only in the first two successive images, (5-1.a) and (5-2.a), where they are correctly detected. As for the application of YOLO in this case, the small “sign” fails to be detected in all first five successive images. However, YOLO correctly detects all “door” instances from (5-1.b) to (5-6.b) with confidence levels over 91%. The “stairs” is missed in (5-2.b) as a False Negative (FN) detection, while from (5-3.b) to (5-6.b) it correctly redetect the “stairs” with varying confidence levels. Same as for Mask RCNN, the “person” is correctly detected in the first two images.

Overall, when considering successively captured images, depicted in Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5 and Figure 4.6 respectively, as a robot moves toward pre-identified targets from far away, both approaches, Mask RCNN and YOLO, exhibit some variability in their recognition performance. In the collaborative robotic context considered for this research, this behavior of the target object detection stage contributes to increase the ambiguity for the robots to be optimally assigned to tackle specific tasks. Figure 4.7 shows the evolution in the confidence level for each object detector and for each class of target objects when they are imaged successively from a shortening distance as the camera is progressively guided toward to region where an object is initially discovered. It is observed that the confidence level for each target object recognized from the series of images does not systematically and smoothly increase as the camera comes closer to the target, as would be expected. However, Mask RCNN exhibits a more stable progression of its confidence level, likely explained by the fact that it begins with a higher performance even when the target object is observed at small scale from a large distance. Surprisingly, the confidence score of Mask RCNN may even slightly decrease as the object becomes more visible in the image. On the other hand, with YOLO a general positive trend in the confidence level is observed as the early

detection results on visually small target objects are underperforming but rapidly improve as the object occupies more space in the input image.

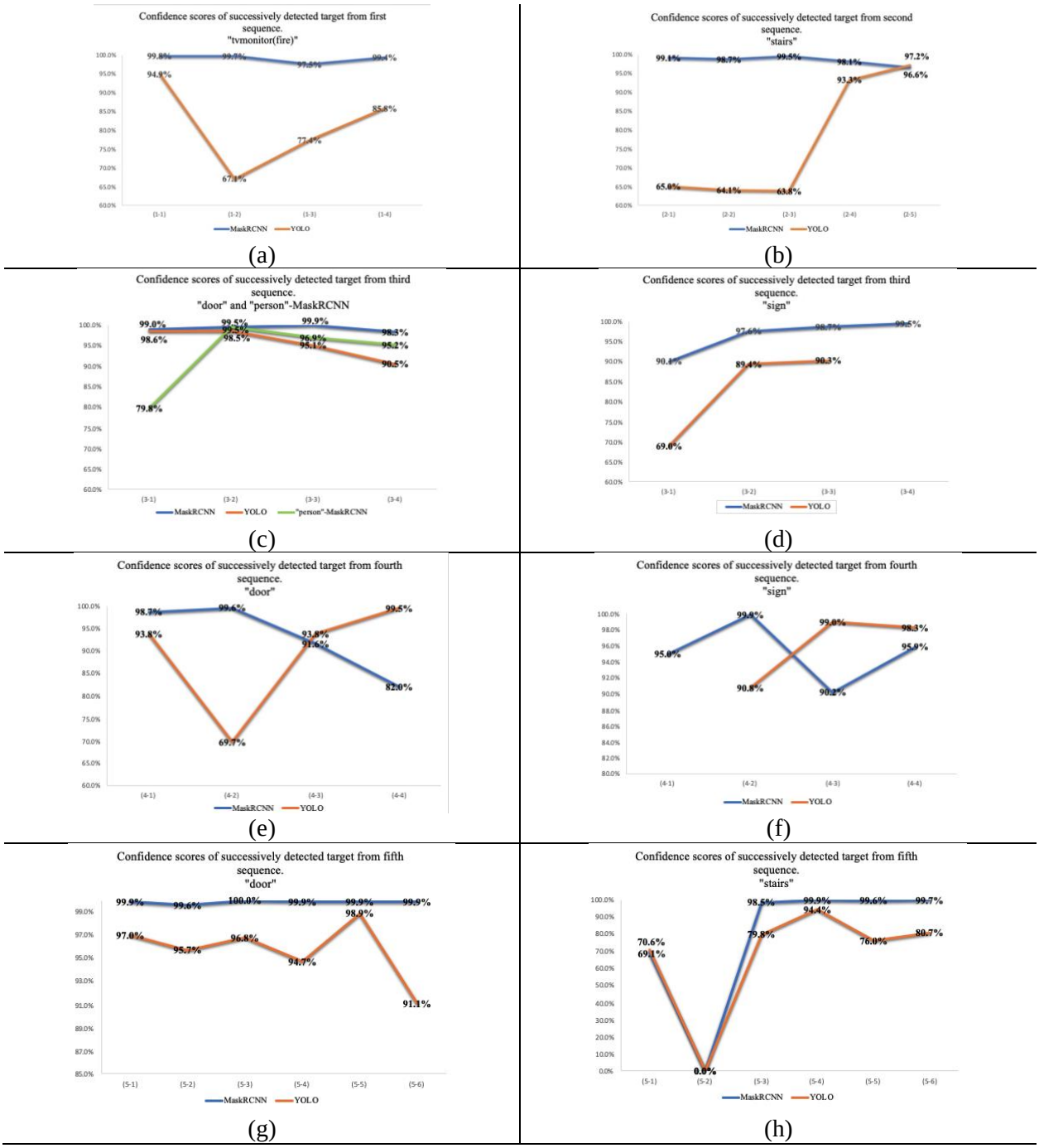


Figure 4.7: Confidence level evolution with Mask RCNN and YOLO applied on successive images containing target objects of interest.

It is important to note here that the confidence level extracted from Mask RCNN and YOLO represent stepwise individual confidence scores generated by the respective network at each instant of observation, that is, individually for each image. No data fusion scheme was involved to leverage prior observation and related confidence scores in the determination of updated confidence levels as the camera approaches from any target object. However, what we expected for the recognition results performance on successive images is that the confidence score increases as the robot gets closer to target objects and their projection on images becomes larger, therefore exhibiting more distinctive features. This experiment and the related observations open the door to different strategies to develop a framework that will favor such a behavior in object detection. Therefore, considerations on how to progressively refine the target object recognition stage based on the detection results of an object detector (Mask RCNN/YOLO) as a robot carrying a camera moves toward a target object from far away will be investigated in the next section. As such, a progressive refinement method for object detection and recognition will be proposed.

4.2 Progressive refinement recognition mechanism

Unlike other works from the literature [62][63][64], which add refining layers in the detection network to iteratively refine object box proposals, and [71][72], which enhance the detection performance by fusing detection decisions on a same image from multiple object detectors, this thesis proposes a progressive refinement method to iteratively improve the estimation of the confidence level on iteratively detected target objects. The proposed method avoids training the new detection network with additional refining layers, and there is no need to integrate recognition results originating from multiple detectors. Rather it utilizes the detection results generated by a single detection network and refines the confidence level progressively. The proposed progressive refinement method can be considered as a form of post-processing of recognition results at the inference stage for a given target object detector. The goal of the proposed approach is to achieve the highest possible confidence level on the certainty of target objects being detected among several successive images inputs, or from real-time continuous video inputs fed to the detection stage. Increasing confidence of the detected target objects is essential for providing robust and accurate inputs to a collaborative robots task allocation stage. Based on the outputs of the detection module (described and experimentally implemented in Chapter 3) which generates bounding boxes

(bboxs) around every detected object, along with corresponding category and confidence level, the proposed method iteratively fuses each recognition result from successive images input, resulting in a progressively refined detection strategy. Figure 4.8 presents the framework of the proposed progressive refinement method.

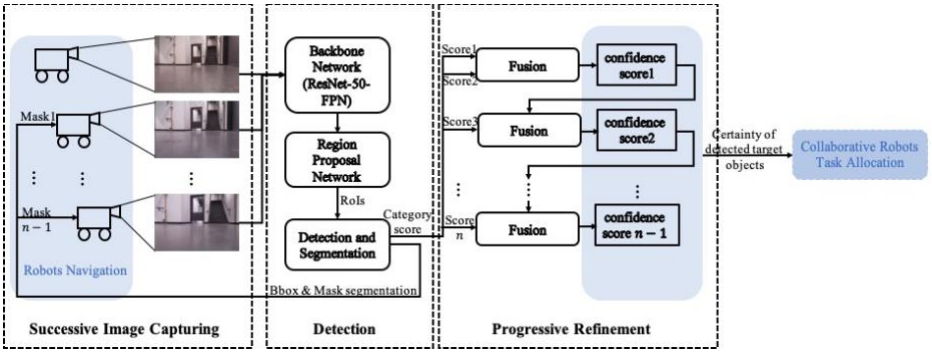


Figure 4.8: Mechanism of proposed progressive refinement method.

After a mobile robot with an embedded camera captures a first image containing target objects of interest, the captured image is fed into the detection module, which performs the detection process and generates an output identifying the class of objects and a confidence score for each detected target object, as well as the corresponding bounding box, and with (Mask RCNN) or without (YOLO) segmented mask areas. The detected bounding box (or mask segmentation) information can be used to guide the robot moving in the direction of the target of interest and then capture new images along its path, while it gets closer to the target. The robot navigation aspect remains beyond the scope of this thesis. Following an initial pair of detections, the detected class confidence scores become the input to the progressive refinement module, which iteratively performs data fusion between pairs of confidence scores achieved on a same target which was detected over two successively captured images. The data fusion process can be achieved through Bayesian or Dempster-Shafer theory based methods. This forms an iterative process, leveraging the previous recognition results as evidence to improving the current prediction, updating the confidence level and potentially increasing the certainty on detected target objects. In this research, both the Bayesian and Dempster-Shafer theory based fusion processes are computed using our own implementation. In the context considered for this research, the goal is to leverage the refined confidence level to benefit the process of collaborative robots task allocation. For the data fusion

stage, given that five classes are considered for target objects recognition, we reassign the recognition results from the detection module as a two-class object recognition result for each individual detected object, that is the target belongs to the specific category with a given confidence level, or not. The fusion process then goes on for each class independently.

4.2.1 Bayesian based data fusion method

For a two-class object detection problem based on the Bayesian method, there are two hypotheses: target (T) and non-target ($\neg T$). The summation of $P(T)$ and $P(\neg T)$ is 1, where $P(T)$ is the initial degree of belief in T, also known as the prior probability, and $P(\neg T)$ is the corresponding initial degree of belief in $\neg T$. For each detection, the result of category score p generated by the object detection module, which represents the possibility that the detected object belongs to the specific predefined category, is assigned as the conditional probability $P(D/T)$. The fusion of two individual detections from successive images is estimated by calculating the posterior probability $P(T/D)$, described in Equation (2.5), where the prior probability used in each detection is updated based on the previous detection.

Figure 4.9 shows the Bayesian based fusion method adapted for progressive refinement. Since the mobile robot moves toward an object of interest from far away and keeps capturing images simultaneously, we can consider the successive images captured by a sensor mounted on a robot as independent but successive inputs to the target object detection. The outputs of each detection (D_i) by Mask RCNN or YOLO, as described in Chapter 3, are encoded by bounding boxes (d_i^j) and the corresponding detected categories with confidence scores $\left(s_{ij}^c\right)$ for each instance of a target object of interest. This information provides the confidence estimated by the detection module on the detected target object belonging to a specific category of objects of interest. d_i^j denotes the j_{th} bbox of i_{th} input detection, and s_{ij}^c denotes the score of j_{th} bbox with corresponding detected category c at i_{th} input detection. In the considered indoor SAR scenario application, $c = 5$, from 1 to 5, where each number represents one category $(1: door; 2: person; 3: sign; 4: stairs; 5: tvmonitor(fire)$. d_i^j and s_{ij}^c will be the inputs to the progressive refinement module. The conditional probabilities of a specific detected target object

category, and non-target, are assigned as $P_{ij}(D_i|T_c)$ and $P_{ij}(D_i|\neg T_c)$, respectively, where T_c is the detected target object of category c and $P_{ij}(D_i|T_c)$ is the basic probability assignment (BPA) for the j_{th} bbox with corresponding category c at i_{th} detection, which is assigned by s_{ij}^c. The non-target probability $P_{ij}(D_i|\neg T_c)$, is assigned as $1 - s_{ij}^c$. The prior probability denotes as $P_{ij}(T_c)$. However, the prior probability for the first detection is unknown. Since it is usual to consider an unknown state corresponding to a probability of 0.5 in a probabilistic reasoning framework, we initially assign 0.5 as $P_{1j}(T_c)$. In the fusion process, the prior probability is updated based on the previous detection result, that is the probability of $P_{ij}(T_c|D_i)$ will be assigned as the prior probability of $P_{i+1,j}(T_c)$. The updated confidence score after each data fusion denotes as $S_{ij}(c)$. All related symbols are described in Table 4.1

Table 4.1: Symbols and descriptions used in the Bayesian based fusion method.

Symbol	Definition and description
c	Predefined target object category. In total 5 categories are considered in indoor SAR scenarios: 1: $door$; 2: $person$; 3: $sign$; 4: $stairs$; 5: $tvmonitor(fire)$
D_i	Detection results for i_{th} input image.
d_i^j	Detected bounding box. i denotes the i_{th} input image. j denotes the j_{th} bounding box detected in the image.
s_{ij}^c	Detected score of j_{th} bbox with corresponding category c for the i_{th} input detection.
T_c	Hypothesis of the specific detected target object c.
$\neg T_c$	Hypothesis of non-target.
$P_{ij}(T_c)$	Prior probability of j_{th} bbox with corresponding category c for the i_{th} detection.
$P_{ij}(D_i\|T_c)$	Conditional probability for j_{th} bbox with corresponding category c at i_{th} detection, which is assigned by s_{ij}^c.
$P_{ij}(D_i\|\neg T_c)$	Conditional probability for j_{th} bbox with non-target at i_{th} detection, which is assigned by $1 - s_{ij}^c$.
$P_{ij}(T_c\|D_i)$	Confidence score for j_{th} bbox with corresponding category c at i_{th} detection.
$S_{ij}(c)$	Updated confidence score of detected category c for j_{th} bbox after each fusion.

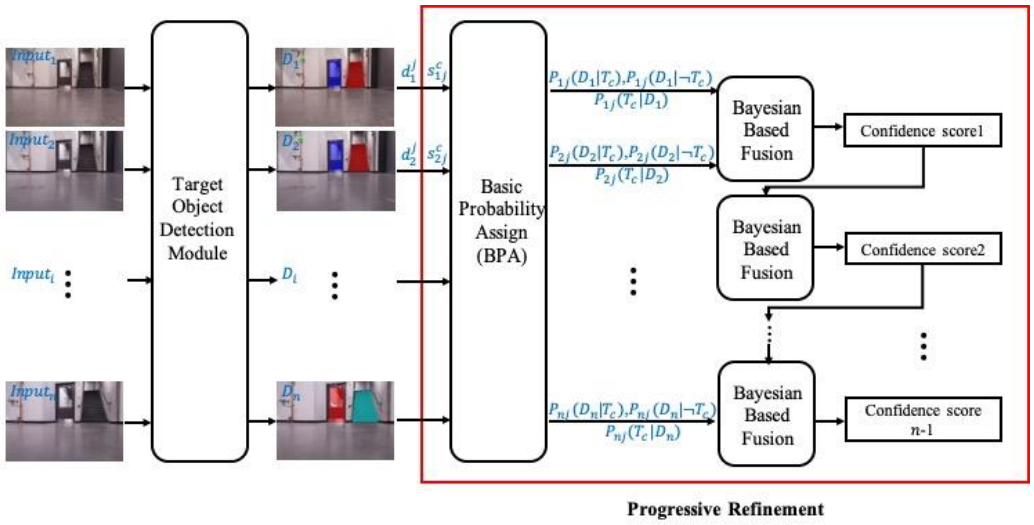


Figure 4.9: Bayesian based fusion method for progressive refinement. (Color codes are not associated to specific classes, which explains the change of color in between instances of a same object in different predicted images.)

4.2.2 Dempster-Shafer theory based data fusion method

For a two-class object detection problem based on the Dempster-Shafer (D-S) theory, let's consider the universe set $\Theta = \{T, \neg T\}$ and its power set $2^{\Theta} = \{\phi, T, \neg T, \{T, \neg T\}\}$, where T and $\neg T$ are target and non-target hypotheses, respectively. $\{T, \neg T\}$ represents detection ambiguity, which indicates that the subject observation could be either target or non-target. For each detection, the detection results of a category score p, which represents the possibility that the detected object belongs to a specific predefined category, is assigned as the basic probability of target hypothesis $\{T\}$, then $1 - p$ needs to be split to account for two basic probabilities of non-target $\{\neg T\}$ and target or non-target $\{T, \neg T\}$. We can hardly assign 0 as BPA for either $\{\neg T\}$ or $\{T, \neg T\}$, as it will lead to the same values of $Bel(T)$ and $Pl(T)$, which affects the certainty measurement. Therefore, we assumedly assign $\frac{1-p}{2}$ equally as the basic probability for the other two hypotheses $\{\neg T\}$ and $\{T, \neg T\}$. The fusion of the two individual detections from successive image inputs is achieved by computing joint BPAs of$\{T\}$, $m_f(T)$, by Dempster's rule of combination (DRC) in Equation (2.18).

Then the final probability interval of a detected target object is $[Bel(T), Pl(T)]$, which refines the confidence certainty of successively detected objects.

Figure 4.10 shows the D-S theory based fusion method adapted for progressive refinement. The inputs to the D-S theory based progressive refinement module are d_i^j and s_{ij}^c, which are the same as those to the Bayesian based progressive refinement module discussed in Section 4.2.1. The basic probabilities $m_i(T_c)$ will be assigned to three hypotheses for each detection: hypothesis of the detected specific target object c, hypothesis of non-target $\{\neg T_c\}$, and the hypothesis of target or non-target $\{T_c, \neg T_c\}$, where T_c is the detected target object of category c, and $m_i(T_c)$ is the basic probability assignment (BPA) for the detected specific target object at i_{th} detection, which is assigned by s_{ij}^c. $m_i(\neg T_c)$ and $m_i(\{T_c, \neg T_c\})$ are assumedly assigned as $\frac{1-s_{ij}^c}{2}$ respectively. Since we consider two successive detection results for data fusion, we need to get all BPAs for each detection, then the Dempster's rule of combination will be applied to calculate the joint BPAs $m_f(T_c)$, and the upper and lower bounds of the probability interval $[Bel(T_c), Pl(T_c)]$, which indicates the certainty of detected target objects, will be determined. The above fusion process will be conducted iteratively for all input images, resulting in a progressively refined confidence level and increased certainty for detected targets. All related symbols are described in Table 4.2.

Table 4.2: Symbols and descriptions used in the D-S theory based fusion method.

Symbol	Definition and description
c	Predefined target object category. In total 5 categories are considered in indoor SAR scenarios: $1: door; 2: person; 3: sign; 4: stairs; 5: tvmonitor (fire)$
D_i	Detection results for i_{th} input image.
d_i^j	Detected bounding box. i denotes the i_{th} input image j denotes the j_{th} bounding box detected in the image.
s_{ij}^c	Detected score of j_{th} bbox with corresponding category c for the i_{th} input detection.
T_c	Hypothesis of the specific detected target object c.
$\neg T_c$	Hypothesis of non-target.
$m_i(T_c)$	Basic probability assignment (BPA) for the detected specific target object c at i_{th} detection, which is assigned by s_{ij}^c.
$m_i(\neg T_c)$	BPA for non-target, which is assumedly assigned by $\frac{1-s_{ij}^c}{2}$.
$m_i(\{T_c, \neg T_c\})$	BPA for target or non-target, which is assumedly assigned by $\frac{1-s_{ij}^c}{2}$.
$m_f(T_c)$	Joint BPAs of two successive detections for the detected specific target object c.
$[Bel(T_c), Pl(T_c)]$	Upper and lower bounds of the confidence interval. 1-$[Pl(T_c) - Bel(T_c)]$ denotes the certainty of the detected target of category c.
$S_{ij}(c)$	Updated confidence score of detected category c j_{th} bbox after each fusion.

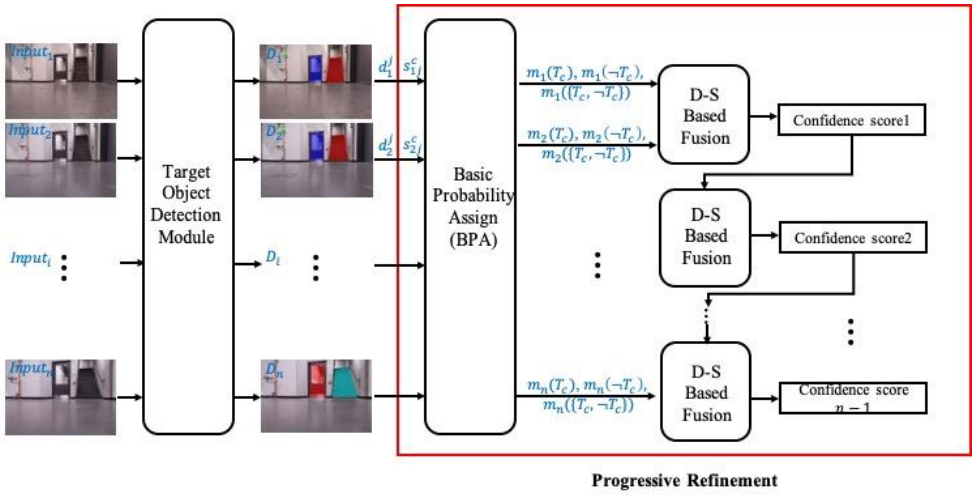


Figure 4.10: D-S theory based fusion method for progressive refinement. (Color codes are not associated to specific classes, which explains the change of color in between instances of a same object in different predicted images.)

Situations may happen where a false positive detection occurs at one iteration among the successive images, that is a given target is wrongly detected as another target class in one image before being correctly recognized again in the following images, as shown in Figure 4.6(5-2.a) for the “stairs” that are wrongly detected as “door” by Mask RCNN. In such cases, the score used for the fusion step takes “1 - score of the false positive detection”, which corresponds to the recognition score of a non-target detection. Alternatively, if the false negative detection happens over successive images, that is the given target failed to be detected correctly over a number of images and then gets correctly detected again after a few frames, as shown in Figure 4.6(5-2.b), where the “stairs” fail to be detected by YOLO, then, the score used for the fusion step assigns 0.5. The detailed application on these cases will be shown in the next section.

4.3 Target object recognition on image sequences with progressive refinement

In this section, details on the experimental validation conducted with the proposed progressive refinement method are reported, respectively with the Bayesian and Dempster-Shafer data fusion schemes. Both detectors selected in Chapter 3, that is Mask RCNN and YOLO, will be considered. Experiments are conducted on the 5 sequences of images introduced in Section 4.1 and presented in Figures 4.2 to 4.6. The initial evaluation of target object detection conducted in Section 4.1

without any refinement procedure demonstrated the variability of confidence scores when individual recognition confidence scores are considered independently. These confidence scores will be used here as the input to the refinement module. The influence of progressive refinement which is made possible with the Bayesian and D-S theory based data fusion methods will be evaluated and compared.

4.3.1 Progressive refinement method based on Mask RCNN detector with Bayesian fusion

Figure 4.11 presents the results of the Bayesian based progressive refinement strategy with application on the Mask RCNN detector for recognizing the target object "tvmonitor (fire)" over 4 successive images from the first sequence in Figure 4.2. Figure 4.12 shows the updated confidence score trend in comparison with the object recognition performance achieved with Mask RCNN individually on every frame.

The first sequence of successive images, Fig. 4.11(1-1), (1-2), (1-3) and (1-4), presents a case where Mask RCNN correctly recognizes the same target in each one of the successive images. For this case, the refined confidence score progressively increases in the beginning of the fusion process and quickly converges to a high confidence level of 99.9% after refining the recognition of the same detected target. In comparison, even for a single target "tvmonitor (fire)" in the image, Figure 4.12 shows that the individual detections (blue line) generate varying scores, starting strong at 99.8%, then keep going down to 97.5%, to end up with 99.4%, which is below the initial score of 99.8%. In spite of that, the Bayesian based data fusion strategy (orange line) keeps improving the confidence over the iterations and then converges to a high confidence of 99.9%, because the system continues to recognize a "tvmonitor (fire)" at all steps, which further confirms the initial finding. Compared with the original variable recognition score obtained by Mask RCNN (blue line) independently over the set of successive images captured by a mobile robot as it moves closer to the target, the Bayesian based refinement method (orange line) exhibits the expected performance on progressively improving and stabilizing the confidence score on the iteratively detected target.

	Image input	Recognition results (from detector)		Bayesian based refinement	
			Score (%)	Iterative fusion	Confidence score (%)
(1-1)	 $Input_1$	 D_1	tvmonitor (fire): $d_1^1, s_{11}^5 = 99.8$	Fusion1 (D_1, Prior)	tvmonitor (fire): $S_{11}(5) = 99.8$
(1-2)	 $Input_2$	 D_2	tvmonitor (fire): $d_2^1, s_{21}^5 = 99.7$	Fusion2 (S_1, D_2)	tvmonitor (fire): $S_{21}(5) = 99.9$
(1-3)	 $Input_3$	 D_3	tvmonitor (fire): $d_3^1, s_{31}^5 = 97.5$	Fusion3 (S_2, D_3)	tvmonitor (fire): $S_{31}(5) = 99.9$
(1-4)	 $Input_4$	 D_4	tvmonitor (fire): $d_4^1, s_{41}^5 = 99.4$	Fusion4 (S_3, D_4)	tvmonitor (fire): $S_{41}(5) = 99.9$

Figure 4.11: Target object detection results with Bayesian based progressive refinement strategy with application of the Mask RCNN detector on the first sequence.

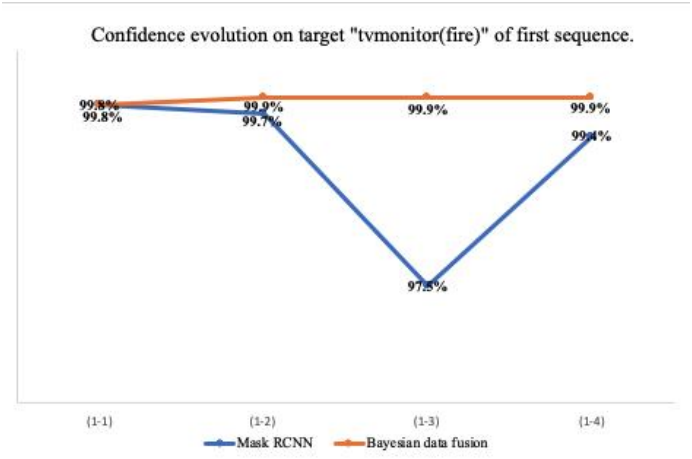


Figure 4.12: Evolution of confidence score on recognized object in the first sequence without and with Bayesian based progressive refinement (Mask RCNN detector).

Figures 4.13 and 4.14 illustrate the results of Bayesian based progressive refinement strategy with application of the Mask RCNN detector for recognizing the target object "stairs" over 5 successive images from the second sequence in Figure 4.3, and its updated confidence evolution, respectively.

	Image input	**Recognition results (from detector)**		**Bayesian based refinement**	
			Score (%)	Iterative fusion	Confidence score (%)
(2-1)	 $Input_1$	 D_1	stairs: $d_1^1, s_{11}^4 = 99.1$	Fusion1 (D_1, Prior)	stairs: $S_{11}(4) = 99.1$
(2-2)	 $Input_2$	 D_2	stairs: $d_2^1, s_{21}^4 = 98.7$	Fusion2 (S_1, D_2)	stairs: $S_{21}(4) = 99.9$
(2-3)	 $Input_3$	 D_3	stairs: $d_3^1, s_{31}^4 = 99.5$	Fusion3 (S_2, D_3)	stairs: $S_{31}(4) = 99.9$
(2-4)	 $Input_4$	 D_4	stairs: $d_4^1, s_{41}^4 = 98.1$	Fusion4 (S_3, D_4)	stairs: $S_{41}(4) = 99.9$
(2-5)	 $Input_5$	 D_5	stairs: $d_5^1, s_{51}^4 = 96.6$	Fusion5 (S_4, D_5)	stairs: $S_{51}(4) = 99.9$

Figure 4.13: Target object detection results with Bayesian based progressive refinement strategy with application of the Mask RCNN detector on the second sequence.

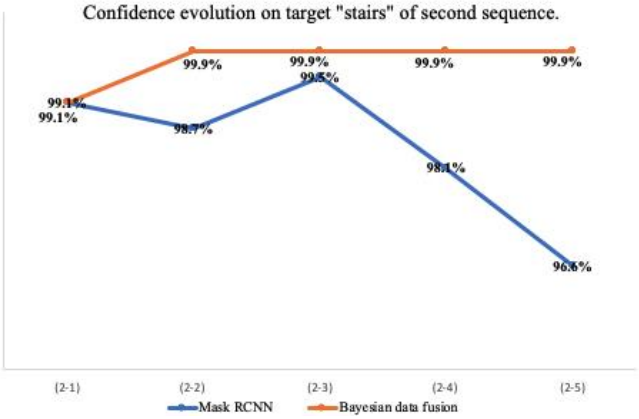


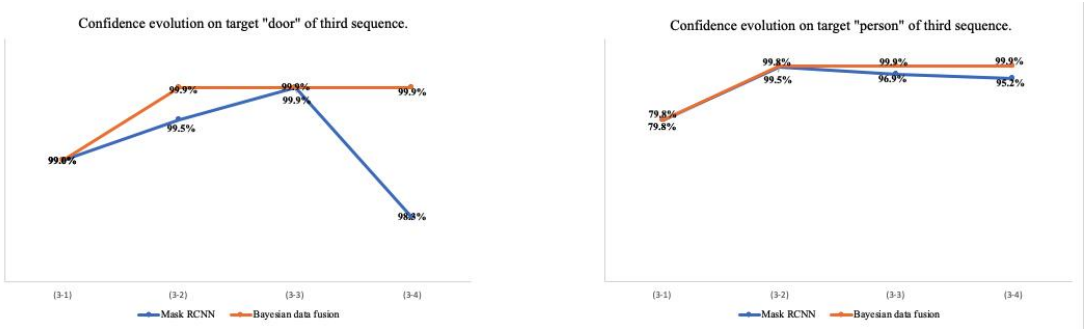
Figure 4.14: Evolution of confidence score on recognized object in the second sequence without and with Bayesian based progressive refinement (Mask RCNN detector).

In the second sequence, the confidence score of "stairs" independently generated by the detector starts at 99.1%, then goes down to 98.7%, and back to 99.5%, before finally going down to 96.6%, which is below the initial score of 99.1%. Conversely, when applying the Bayesian based data fusion, the confidence score keeps increasing and converges to a high confidence of 99.9%. The evolution of the recognition score becomes much less sensitive to inherent variations in the recognition performance of the Mask RCNN model on individual samples which may be influenced by the training of the network but also from the inherent variability in between sample images collected in realistic and not perfectly controlled environments, which is a factor that robotic systems must deal with.

Figures 4.15 and 4.16 show the refinement results in the third sequence, which includes target objects "door" and "person". Similarly, Figures 4.17 and 4.18 describe the refinements achieved on the fourth sequence with instances of the "door" and "sign" classes. In both these cases, the detector correctly recognizes the same target in each of the successive images and the progressively improved performance on confidence score is observed after applying the Bayesian based refinement strategy.

	Image input	Recognition results (from detector)		Bayesian based refinement	
			Score (%)	Iterative fusion	Confidence score (%)
(3-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 99.0$ person: $d_1^2, s_{12}^2 = 79.8$	Fusion1 (D_1, Prior)	door: $S_{11}(1) = 99.0$ person: $S_{12}(2) = 79.8$
(3-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 99.5$ person: $d_2^2, s_{22}^2 = 99.5$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.9$ person: $S_{22}(2) = 99.8$
(3-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 99.9$ person: $d_3^2, s_{32}^2 = 96.9$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.9$ person: $S_{32}(2) = 99.9$
(3-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 98.3$ person: $d_4^2, s_{42}^2 = 95.2$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.9$ person: $S_{42}(2) = 99.9$

Figure 4.15: Target objects detection results with Bayesian based progressive refinement strategy with application of the Mask RCNN detector on the third sequence.

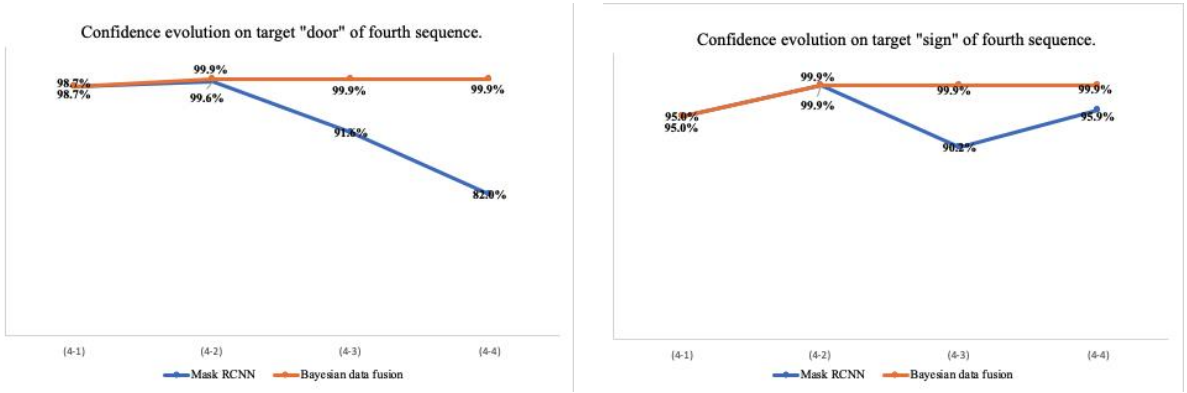


(a) (b)

Figure 4.16: Evolution of confidence score on recognized objects in the third sequence without and with Bayesian based progressive refinement (Mask RCNN detector).

	Image input	Recognition results (from detector)		Bayesian based refinement	
			Score (%)	Iterative fusion	Confidence score (%)
(4-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 98.7$ sign: $d_1^2, s_{12}^3 = 95.0$	Fusion1 (D_1, Prior)	door: $S_{11}(1) = 98.7$ sign: $S_{12}(3) = 95.0$
(4-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 99.6$ sign: $d_2^2, s_{22}^3 = 99.9$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.9$ sign: $S_{22}(3) = 99.9$
(4-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 91.6$ sign: $d_3^2, s_{32}^3 = 90.2$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.9$ sign: $S_{32}(3) = 99.9$
(4-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 82.0$ sign: $d_4^2, s_{42}^3 = 95.9$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.9$ sign: $S_{42}(3) = 99.9$

Figure 4.17: Target objects detection results with Bayesian based progressive refinement strategy with application of the Mask RCNN detector on the fourth sequence.



(a) (b)

Figure 4.18: Evolution of confidence score on recognized objects in the fourth sequence without and with Bayesian based progressive refinement (Mask RCNN detector).

Finally, Figure 4.19 and 4.20 reveal the refinement results and confidence evolution obtained on the fifth sequence that involve false negatives.

In the latter case, the Mask RCNN detector leads to erroneous (false positive) classification on a given target after a number of positive identifications and then correctly detects the target again. After the first correct detection of “stairs” in Fig. 4.19(5-1), the category of “stairs” is wrongly detected as “door” in Fig. 4.19(5-2), and later on it turns back to be detected as “stairs” in the subsequent detections from Fig. 4.19(5-3)-(5-6). For this case, the refined confidence score goes down in the beginning, when the false positive recognition happens, but it quickly recovers and goes up as the detector correctly reclassifies the target again. However, for that erroneous classification of the “stairs” as a “door” at the second iteration, the detector by default assigns 0 confidence to “stairs” which indicates a total loss of confidence in the target object of this category being detected. But from a robot navigation perspective, this can have a dramatic impact as the robot may be redirected toward a different target object, resulting in failure of the mission to tackle the task on the initially detected object. To prevent such drastic effect from happening, the proposed fusion scheme rather allocates the difference between 1 and the score erroneously assigned to the “door”, that is 78.1%, to mitigate the drop in the “stairs” confidence level. As a result, the Bayesian based refinement strategy is applied with a differential confidence score of 1-0.781 = 0.219 at that iteration, which leads to the confidence score on “stairs” to only exhibit a decrease to 38.5% since the detector relies on the previous detection results at 69.1% as an evidence to reinforce the current prediction. The robot may then keep exploring this region of space and acquire additional images, which lead to a full recovery of the “stairs” object instance, with the following iteration confidence score estimation reaching to 98.5% and the progressively refined confidence score raising back to 97.6%.

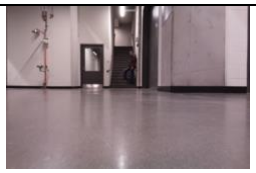
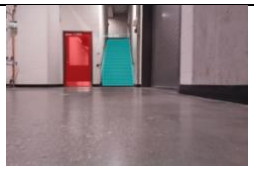
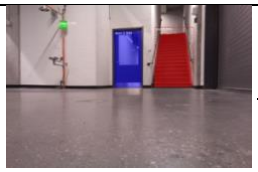
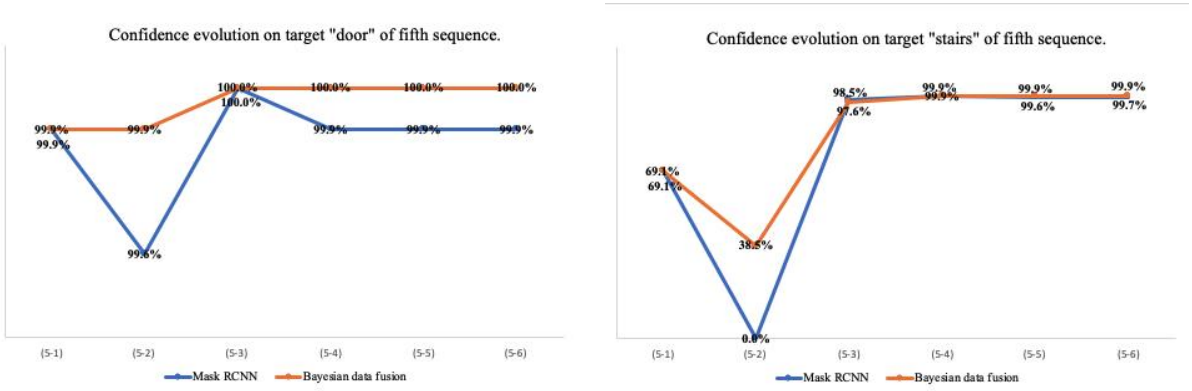
	Image input	Recognition results (from detector)		Bayesian based refinement	
			Score (%)	Iterative fusion	Confidence score (%)
(5-1)			door: $d_1^1, s_{11}^1 = 99.9$	Fusion1 (D_1, Prior)	door: $S_{11}(1) = 99.9$
	$Input_1$	D_1	stairs: d_1^2, s_{12}^4 =69.1		stairs: $S_{12}(4) = 69.1$
(5-2)			door: $d_2^1, s_{21}^1 = 99.6$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.9$
	$Input_2$	D_2	**door(purple): $\boldsymbol{d_2^2, s_{22}^1 = 78.1}$ stairs: $\mathbf{d_2^2, s_{22}^4 = 21.9}$ (for fusion)**		stairs: $S_{22}(4) = 38.5$
(5-3)			door: $d_3^1, s_{31}^1 = 100$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 100$
	$Input_3$	D_3	stairs: $d_3^2, s_{32}^4 = 98.5$		stairs: $S_{32}(4) = 97.6$
(5-4)			door: $d_4^1, s_{41}^1 = 99.9$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 100$
	$Input_4$	D_4	stairs: $d_4^2, s_{42}^4 = 99.9$		stairs: $S_{42}(4) = 99.9$
(5-5)			door: $d_5^1, s_{51}^1 = 99.9$	Fusion5 (S_4, D_5)	door: $S_{51}(1) = 100$
	$Input_5$	D_5	stairs: $d_5^2, s_{52}^4 = 99.6$		stairs: $S_{52}(4) = 99.9$
(5-6)			door: $d_6^1, s_{61}^1 = 99.9$	Fusion6 (S_5, D_6)	door: $S_{61}(1) = 100$
	$Input_6$	D_6	stairs: $d_6^2, s_{62}^4 = 99.7$		stairs: $S_{62}(4) = 99.9$

Figure 4.19: Target objects detection results with Bayesian based progressive refinement strategy with application of the Mask RCNN detector on the fifth sequence.



(a) (b)

Figure 4.20: Evolution of confidence score on recognized objects in the fifth sequence without and with Bayesian based progressive refinement (Mask RCNN detector).

4.3.2 Progressive refinement method based on Mask RCNN detector with D-S theory fusion

Figure 4.21 presents similar results when the Bayesian data fusion scheme is substituted with the D-S theory based progressive refinement strategy, but still considering the application of the Mask RCNN detector for recognizing the target object “tvmonitor (fire)” from the first sequence of four successive images. Accordingly, Figure 4.22 describes the updated confidence score evolution after each progressive refinement step.

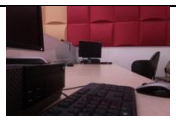
	Image input	Recognition results (from detector)		D-S theory based refinement		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(1-1)	 $Input_1$	 D_1	tvmonitor (fire): $d_1^1, s_{11}^5 = 99.8$	Fusion1 (D_1)	tvmonitor (fire): $S_{11}(5) = 99.80$	-
(1-2)	 $Input_2$	 D_2	tvmonitor (fire): $d_2^1, s_{21}^5 = 99.7$	Fusion2 (S_1, D_2)	tvmonitor (fire): $S_{21}(5) = 99.99$	tvmonitor (fire): $[99.99, 99.99]$
(1-3)	 $Input_3$	 D_3	tvmonitor (fire): $d_3^1, s_{31}^5 = 97.5$	Fusion3 (S_2, D_3)	tvmonitor (fire): $S_{31}(5) = 99.99$	tvmonitor (fire): $[99.99, 99.99]$
(1-4)	 $Input_4$	 D_4	tvmonitor (fire): $d_4^1, s_{41}^5 = 99.4$	Fusion4 (S_3, D_4)	tvmonitor (fire): $S_{41}(5) = 99.99$	tvmonitor (fire): $[99.99, 99.99]$

Figure 4.21: Target object detection results with D-S theory based progressive refinement strategy with application of the Mask RCNN detector on the first sequence.

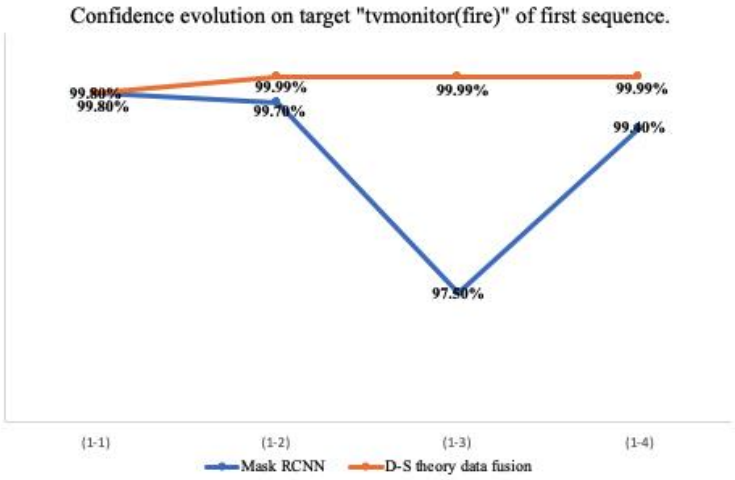


Figure 4.22: Evolution of confidence score on recognized object in the first sequence without and with D-S theory based progressive refinement (Mask RCNN detector).

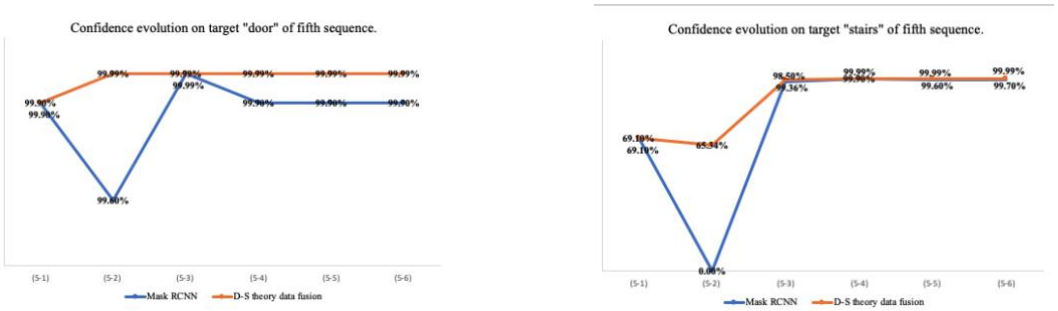
For the recognition scores obtained with Mask RCNN, the D-S theory based refinement also generates cumulatively updated confidence scores after every fusion step, as shown in Fig. 4.21(1-1)-(1-4). Besides the updated and increasing confidence score, D-S theory based refinement also provides additional information on the certainty of the detected target, outputting the confidence interval after each fusion step. Since the initial recognition score detected by Mask RCNN is generally with high confidence, the updated confidence interval after each fusion step shows minor or no uncertainty, which in turn explains the reinforced confidence after each fusion step. For example, in the first sequence, the confidence interval after the last three fusion steps generated by D-S theory based fusion are all [99.99%, 99.99%]. The difference between the upper and lower bounds is 0, which indicates no uncertainty on the class of the detected target as "tvmonitor (fire)". In other words, the certainty on the detected target reaches 1 immediately after the second fusion step. The trend of the confidence score after D-S theory based progressive refinement, shown in Figure 4.22, is similar to what is achieved with Bayesian data fusion in Figure 4.12. Both fusion methods result in a progressive increase of the confidence score while relying on previous evidence and eventually converge to a high confidence in the recognition.

On the second to the fourth sequences, a similar trend on the recognition confidence evolution happens. The detailed results are presented in Appendix A.1.

Finally, the refinement results and the recognition confidence evolution on the fifth sequence are presented in Figure 4.23 and 4.24, respectively.

	Image input	Recognition results (from detector)		D-S theory based refinement		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(5-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 99.9$ stairs: d_1^2, s_{12}^4 =69.1	Fusion1 (D_1)	door: $S_{11}(1) = 99.90$ stairs: $S_{12}(4) = 69.10$	-
(5-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 99.6$ **door(purple): $\boldsymbol{d_2^2, s_{22}^1 = 78.1}$ stairs: $\mathbf{d_2^2, s_{22}^4 = 21.9}$ (for fusion)**	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.99$ stairs: $S_{22}(4) = 65.34$	door: [99.99, 99.99] stairs: [65.34, 91.33]
(5-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 100$ stairs: $d_3^2, s_{32}^4 = 98.5$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.99$ stairs: $S_{32}(4) = 99.36$	door: [99.99, 99.99] stairs: [99.36, 99.84]
(5-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 99.9$ stairs: $d_4^2, s_{42}^4 = 99.9$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.99$ stairs: $S_{42}(4) = 99.99$	door: [99.99, 99.99] stairs: [99.99, 99.99]
(5-5)	 $Input_5$	 D_5	door: $d_5^1, s_{51}^1 = 99.9$ stairs: $d_5^2, s_{52}^4 = 99.6$	Fusion5 (S_4, D_5)	door: $S_{41}(1) = 99.99$ stairs: $S_{42}(4) = 99.99$	door: [99.99, 99.99] stairs: [99.99, 99.99]
(5-6)	 $Input_6$	 D_6	door: $d_6^1, s_{61}^1 = 99.9$ stairs: $d_6^2, s_{62}^4 = 99.7$	Fusion6 (S_5, D_6)	door: $S_{41}(1) = 99.99$ stairs: $S_{42}(4) = 99.99$	door: [99.99, 99.99] stairs: [99.99, 99.99]

Figure 4.23: Target objects detection results with D-S theory based progressive refinement strategy with application of the Mask RCNN detector on the fifth sequence.



(a) (b)

Figure 4.24: Evolution of confidence score on recognized objects in the fifth sequence without and with D-S theory based progressive refinement (Mask RCNN detector).

In this case, some target objects are wrongly detected as belonging to a different category over the sequence. For the category of “stairs”, which is wrongly detected as a “door” in the second independent detection by Mask RCNN, the confidence interval becomes [65.34%, 91.33%], meaning there is 25.99% uncertainly on the detected target actually being “stairs”. However, the confidence interval is updated to [99.36%, 99.84%] immediately after the third fusion step when the Mask RCNN properly detects “stairs” again, and the uncertainty decreases to just 0.48%. The interval rapidly reaches a range of [99.99%, 99.99%] at the next iteration with complete certainty that the detected target is from the “stairs” category. Overall, the confidence level is updated after each fusion step. Consequently, the certainty on the detected target category is also progressively improved after each iterative fusion with the D-S theory based refinement, as we consistently observe the difference between plausibility and belief to be reduced after every iteration.

In Figure 4.24, the updated confidence trend reveals that the D-S theory based fusion method results in a progressive increase of the confidence score while relying on previous evidence and eventually converges to a high confidence in the recognition, which is also shown when using the Bayesian fusion method. However, the D-S theory data fusion scheme brings particularities, as illustrated in Fig. 4.24(b) where an increased confidence is achieved when handling temporary false positive detections. In this case, the detrimental effect of a false positive detection, which brought the confidence down to 38.5% with the Bayesian scheme, as shown in Fig. 4.20(b), has only decreased the confidence to 65.34% with D-S theory fusion. As a result, it is demonstrated that contradictory conclusions from the object detector tend to have less drastic influence on the

outcomes with the D-S theory scheme, which brings a desirable level of stability to support robust robotic task allocation in challenging environments.

4.3.3 Progressive refinement method based on YOLO detector with Bayesian fusion

Figure 4.25 presents the successive input images from the first sequence iteratively processed with the YOLO detector from which the confidence scores are fed into the Bayesian based progressive refinement framework. Accordingly, Figure 4.26 depicts the updated confidence score trend in comparison with the object recognition performance achieved individually with YOLO on every frame.

	Image input	**Recognition results (from detector)**		**Bayesian based refinement**	
			Score (%)	Iterative fusion	Confidence score (%)
(1-1)	 $Input_1$	 D_1	tvmonitor (fire): $d_1^1, s_{11}^5 = 94.9$	Fusion1 (D_1, Prior)	tvmonitor (fire): $S_{11}(5) = 94.9$
(1-2)	 $Input_2$	 D_2	tvmonitor (fire): $d_2^1, s_{21}^5 = 67.1$	Fusion2 (S_1, D_2)	tvmonitor (fire): $S_{21}(5) = 97.4$
(1-3)	 $Input_3$	 D_3	tvmonitor (fire): $d_3^1, s_{31}^5 = 77.4$	Fusion3 (S_2, D_3)	tvmonitor (fire): $S_{31}(5) = 99.2$
(1-4)	 $Input_4$	 D_4	tvmonitor (fire): $d_4^1, s_{41}^5 = 85.8$	Fusion4 (S_3, D_4)	tvmonitor (fire): $S_{41}(5) = 99.8$

Figure 4.25: Target object detection results with Bayesian based progressive refinement strategy with application of the YOLO detector on the first sequence.

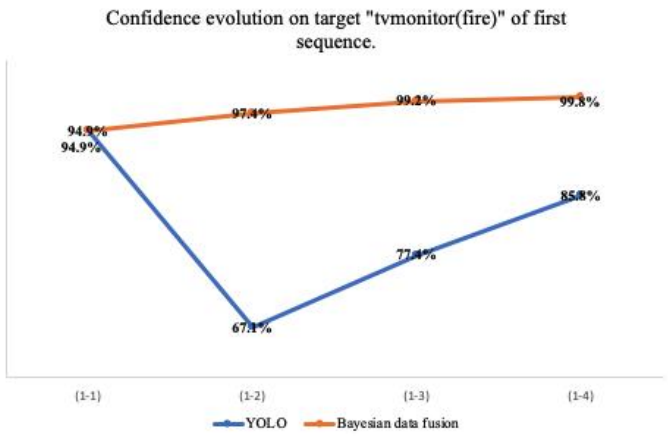


Figure 4.26: Evolution of confidence score on recognized object in the first sequence without and with Bayesian based progressive refinement (YOLO detector).

Figures 4.25(1-1), (1-2), (1-3) and (1-4) show the recognition scores on object detections with YOLO, and the Bayesian based refinement that generates cumulatively updated confidence scores after each fusion step. It exhibits that the confidence score on successively detected targets increases progressively. In Figure 4.26, for a single target of class “tvmonitor (fire)” in the image, the detector alone (blue line) generates varying scores, starting at 94.9%, then going down to 67.1%, and then keeps increasing to end up with 85.8%, which is below the initial score of 94.9%. In contrast, the Bayesian based data fusion strategy keeps improving the confidence at each iterations and then end up with 99.8%, thanks to the combination of evidence collected all over the robot’s path that provided a sequence of four coherent images.

Similar cases from the second to the fourth sequence (Figures 4.3 to 4.5), where the YOLO detector correctly recognizes the same target in each of the successive images, all show a progressively improving performance on the confidence score when applying the Bayesian based refinement strategy. The detailed results are shown in Appendix A.2.

Finally, Figures 4.27 and 4.28 illustrate the refinement results obtained on the fifth sequence that involves erroneous detections.

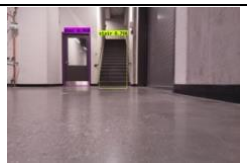
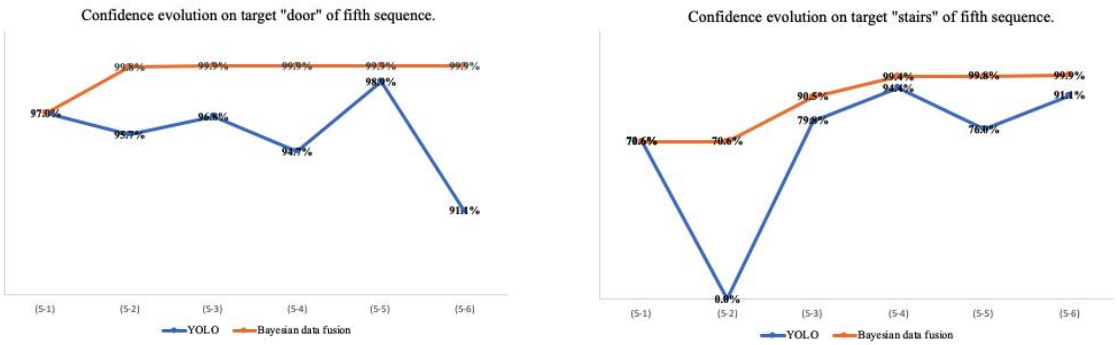
	Image input	Recognition results (from detector)		Bayesian based refinement	
			Score (%)	Iterative fusion	Confidence score (%)
(5-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 97.0$ stairs: $d_1^2, s_{12}^4 = 70.6$	Fusion1 (D_1, Prior)	door: $S_{11}(1) = 97.0$ stairs: $S_{12}(4) = 70.6$
(5-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 95.7$ **stairs: $\mathbf{d_2^2, s_{22}^4 = 0.5}$ (for fusion)**	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.8$ stairs: $S_{22}(4) = 70.6$
(5-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 96.8$ stairs: $d_3^2, s_{32}^4 = 79.8$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.9$ stairs: $S_{32}(4) = 90.5$
(5-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 94.7$ stairs: $d_4^2, s_{42}^4 = 94.4$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.9$ stairs: $S_{42}(4) = 99.4$
(5-5)	 $Input_5$	 D_5	door: $d_5^1, s_{51}^1 = 98.9$ stairs: $d_5^2, s_{52}^4 = 76.0$	Fusion5 (S_4, D_5)	door: $S_{51}(1) = 99.9$ stairs: $S_{52}(4) = 99.8$
(5-6)	 $Input_6$	 D_6	door: $d_6^1, s_{61}^1 = 91.1$ stairs: $d_6^2, s_{62}^4 = 80.7$	Fusion6 (S_5, D_6)	door: $S_{61}(1) = 99.9$ stairs: $S_{62}(4) = 99.9$

Figure 4.27: Target objects detection results with Bayesian based progressive refinement strategy with application of the YOLO detector on the fifth sequence.



(a) (b)

Figure 4.28: Evolution of confidence score on recognized objects in the fifth sequence without and with Bayesian based progressive refinement (YOLO detector).

In Figure 4.27(5-1) to (5-6), for the fifth set of images, differentiating from the case in Figure 4.19(5-1) to (5-6), where Mask RCNN leads to an erroneous (false positive) classification on a given target after a number of positive identifications before correctly detecting the target again, YOLO leads to a missed (false negative) detection. After the first correct detection of "stairs" in Fig. 4.27(5-1), the "stairs" is no longer discovered in Fig. 4.27(5-2), and later on it turns back to be detected as "stairs" in the following iterations from Fig. 4.27(5-3) to (5-6). For this case, the refined confidence score, shown in Fig. 4.28(b), preserves the same level as the previous result when the false negative recognition happens if the refinement scheme is applied. It then further increases its confidence as the detector correctly picks the target again. Besides, for that false negative classification at an individual detection step, the detector alone assigns 0 confidence, which could drastically impact the robotic task allocation process with no candidate target being discovered. However, the confidence score preserves it level at 70.6% after applying the Bayesian based refinement strategy, then relying on the previous detection results as an evidence to reinforce the current prediction. As a result of the proposed progressive refinement strategy, the stability of the target object detection is significantly improved.

4.3.4 Progressive refinement method based on YOLO detector with D-S theory fusion

Finally, Figure 4.29 presents the results obtained with the last combination investigated in this research, that is applying the D-S theory based progressive refinement strategy on the output of the YOLO detector for recognizing the "tvmonitor (fire)" from the first sequence of images.

Accordingly, Figure 4.30 describes the updated confidence score trend after each progressive refinement step.

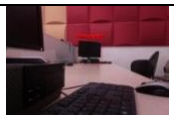
	Image input	**Recognition results (from detector)**	**D-S theory based refinement**			
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(1-1)	 $Input_1$	 D_1	tvmonitor (fire): $d_1^1, s_{11}^5 = 94.9$	Fusion1 (D_1)	tvmonitor (fire): $S_{11}(5) = 94.90$	-
(1-2)	 $Input_2$	 D_2	tvmonitor (fire): $d_2^1, s_{21}^5 = 67.1$	Fusion2 (S_1, D_2)	tvmonitor (fire): $S_{21}(5) = 97.97$	tvmonitor (fire): $[97.97, 99.49]$
(1-3)	 $Input_3$	 D_3	tvmonitor (fire): $d_3^1, s_{31}^5 = 77.4$	Fusion3 (S_2, D_3)	tvmonitor (fire): $S_{31}(5) = 99.48$	tvmonitor (fire): $[99.48, 99.87]$
(1-4)	 $Input_4$	 D_4	tvmonitor (fire): $d_4^1, s_{41}^5 = 85.8$	Fusion4 (S_3, D_4)	tvmonitor (fire): $S_{41}(5) = 99.92$	tvmonitor (fire): $[99.92, 99.98]$

Figure 4.29: Target object detection results with D-S theory based progressive refinement strategy with application of the YOLO detector on the first sequence.

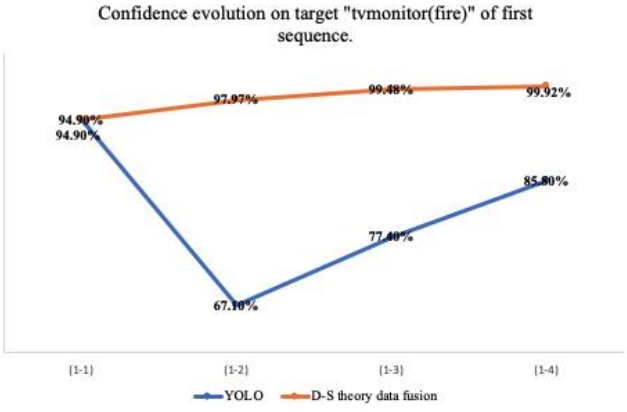


Figure 4.30: Evolution of confidence score on recognized object in the first sequence without and with D-S theory based progressive refinement (YOLO detector).

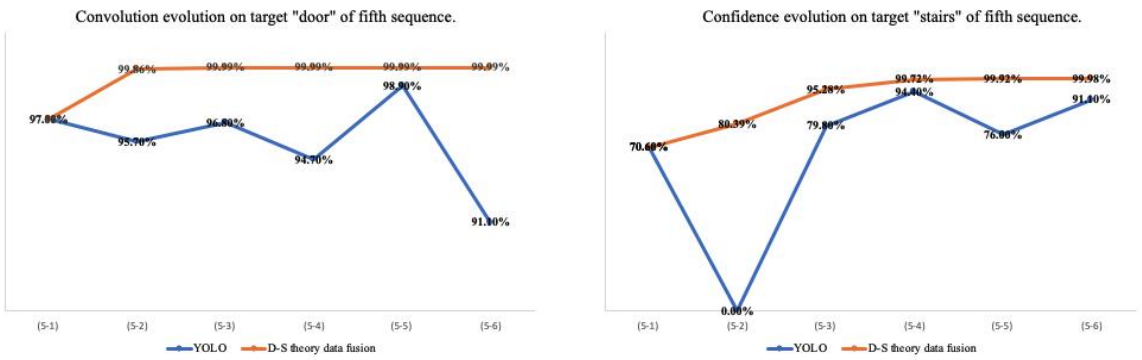
For the recognition scores detected by YOLO, the D-S theory based refinement also generates cumulatively updated confidence scores after every fusion step, and provides the certainty of the detected target. The initial recognition score generated by YOLO is lower than that of Mask RCNN, therefore the improvement of the updated confidence interval after each fusion step is easier to visualize as the progression is slower. For the YOLO detector applied in Figure 4.29(1-1)-(1-4), the updated confidence intervals are [97.97%, 99.49%], [99.48%, 99.87%] and [99.92%, 99.98%] after each fusion step, respectively. That is, the uncertainty of detected "tvmonitor (fire)" is decreasing from 1.52% to 0.39%, which also indicates that the certainty on the detected target class is increasing and end up with 99.61%. In contrast, in Figure 4.21(1-1)-(1-4), where Mask RCNN was applied, the updated confidence interval reached in the range [99.99%, 99.99%] immediately after the second fusion step.

Figures 4.31 and 4.32 show the results obtained on the fifth sequence with YOLO detector. Even though a false negative detection happens in Fig. 4.31(5-2) on the "stairs", the confidence score progressively increases when the D-S theory fusion scheme is applied, as shown in Fig. 4.32(b). The same trend happens in other successive images sequences, for which detailed results are presented in Appendix A.3.

Overall, the confidence level is updated after each fusion step. Consequently, the certainty on the detected target category is also progressively improved after each iterative with the D-S theory based refinement, as we consistently observe the difference between plausibility and belief to be reduced at every iteration. The trend on the confidence score after D-S theory based progressive refinement is similar with what was presented with the Bayesian based progressive refinement, that is exhibiting the expected performance on progressively improving the confidence score on successive images.

	Image input	Recognition results (from detector)		D-S theory based refinement		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(5-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 97.0$ stairs: $d_1^2, s_{12}^4 = 70.6$	Fusion1 (D_1)	door: $S_{11}(1) = 97.00$ stairs: $S_{12}(4) = 70.60$	- -
(5-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 95.7$ **stairs: $\mathbf{d_2^2, s_{22}^4 = 0.5}$ (for fusion)**	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.86$ stairs: $S_{22}(4) = 80.39$	door: [99.86,99.96] stairs: [80.39,95.10]
(5-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 96.8$ stairs: $d_3^2, s_{32}^4 = 79.8$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.99$ stairs: $S_{32}(4) = 95.28$	door: [99.99,99.99] stairs: [95.28,98.82]
(5-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 94.7$ stairs: $d_4^2, s_{42}^4 = 94.4$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.99$ stairs: $S_{42}(4) = 99.72$	door: [99.99,99.99] stairs: [99.72,99.93]
(5-5)	 $Input_5$	 D_5	door: $d_5^1, s_{51}^1 = 98.9$ stairs: $d_5^2, s_{52}^4 = 76.0$	Fusion5 (S_4, D_5)	door: $S_{51}(1) = 99.99$ stairs: $S_{52}(4) = 99.92$	door: [99.99,99.99] stairs: [99.92,99.98]
(5-6)	 $Input_6$	 D_6	door: $d_6^1, s_{61}^1 = 91.1$ stairs: $d_6^2, s_{62}^4 = 80.7$	Fusion6 (S_5, D_6)	door: $S_{61}(1) = 99.99$ stairs: $S_{62}(4) = 99.98$	door: [99.99,99.99] stairs: [99.98,99.99]

Figure 4.31: Target objects detection results with D-S theory based progressive refinement strategy with application of the YOLO detector on the fifth sequence.



(a) (b)

Figure 4.32: Evolution of confidence score on recognized objects in the fifth sequence without and with D-S theory based progressive refinement (YOLO detector).

Comparing the experimental results with progressive refinement combined with either Mask RCNN or YOLO, as relatively higher confidence scores are initially generated by Mask RCNN for the same image, the iterative fusion over successive high scores tends to be converging toward 100% faster, often after the first or second iteration. This can represent an advantage offered by Mask RCNN to improve the efficiency of a robot response. With a higher confidence in the class of a target object obtained quicker, the robotic task allocator is then able to determine the nature of the task to be performed, and can accordingly and promptly assign a specialized robotic agent to handle the detected task. In SAR scenarios, as considered in the context of this research, a timely response is of essence.

In terms of the data fusion methods, both Bayesian and D-S theory based fusion are able to gradually increase the recognition confidence on detected targets over successive images as expected. They both show excellent performance on refining the target recognition capability whenever a target object can be continuously captured over successive frames, and independently from the fact that the scene contains a single target or multiple targets which are visible in the images. The D-S theory based fusion refinement shows better performance in cases where there are erroneous detections of targets or missed targets in some images over the sequence as the confidence level is less drastically impacted by false positives or false negatives. In the context of this research, this may benefit the robots coordination layer by reducing the ambiguity in associating specific target objects with robotic agents, directly resulting in better responsiveness

for the task allocation. Besides the iteratively updated confidence score, D-S theory based refinement also provides additional information on the certainty of the detected target, outputting the confidence interval for each fusion, which also provides auxiliary information to robots to reinforce their interaction with detected objects.

4.4 Summary

In this chapter, we first tested the performance of Mask RCNN and YOLO without applying progressive refinement on successive images captured by a mobile robot equipped with an embedded camera. The study shows that when considered independently the detected confidence scores over successive images containing the same target objects tend to vary significantly. Then, the proposed progressive refinement strategy was presented to overcome the variation on confidence scores. Finally, we experimentally validated and demonstrated the performance on the proposed progressive refinement approach on a variety of image sequences with different levels of complexity.

The experimental validation demonstrates that the proposed progressive refinement approach, with either Bayesian or D-S theory based data fusion, is able to gradually increase the recognition confidence on detected target objects, as expected. This represents a significant advantage when target objects are researched in an unknown environment with vision sensors mounted on mobile robots. In such conditions, initial images capturing specific classes of target objects would typically be imaged initially from far away and in imperfect conditions (small scale, weak lighting, combination of visible targets, etc.). The classical CNN object detectors considered in Chapter 3 have not been designed, nor extensively used under such conditions. The proposed progressive refinement approach proposed in this chapter provides a structured and coherent framework to evolve standalone object recognition platforms by leveraging previously acquired detection results as evidence to reinforce the current prediction in an iterative manner. The proposed framework contributes to improve the robustness of target recognition solutions for field work robotic applications.

Chapter 5 Conclusion

In this Chapter, Section 5.1 summarizes the research in this thesis. The contributions and future work are discussed in Section 5.2 and Section 5.3, respectively.

5.1 Summary

The objective of this thesis is to propose an innovative method for progressively refining target object recognition performance and robustness to support collaborative robots task allocation in multi-agents systems. As presented in literature works, improving the performance of CNN-based backbone networks, which extract image features, can contribute to increase the accuracy for object detection as required in several applications. However, this approach requires extensive computational support. Alternatively, improving the localization accuracy of bounding box proposals by adding additional refining layers in the detector framework also provides refinement in target object detection, but it involves to train the refined detection network from scratch. The thesis proposes an alternative progressive refinement strategy, which leverages the detection results provided by state-of-the-art object detection frameworks in an iterative manner, without retraining, and combines with data fusion methods to progressively refine the confidence on the recognized objects in scenes of varying complexity, and under imperfect imaging conditions.

Both the two-stage object detector Mask RCNN and the one-stage object detector YOLO are closely examined. Some modifications on the backbone and region proposal network are applied on the classical Mask RCNN architecture with the purpose to adapt to multiple scales in target images and better suit the application scenario considered. Both architectures are experimentally evaluated on a newly built dataset to study their feasibility and performance as an entry layer target object detector in the proposed framework. As such, one contribution of the work consists of an expanded image dataset with five predefined classes, which are relevant for collaborative robots task allocation in the context of indoor search-and-rescue scenarios. The dataset contains 805 images representing 1681 instances of objects from five classes in total, along with the annotation of category label, bounding box and mask segmentations.

In terms of object category recognition performance when tested on the dataset, the overall recall of Mask RCNN on all 5 classes reveals higher than that of YOLO, while YOLO achieves higher overall precision over all 5 classes. The high recall performance ensures that targets in the captured image are detected as thoroughly as possible, which is essential for the critical task detection stage involved in collaborative robotic systems. On the other hand, achieving a relatively lower precision performance is less critical in the proposed framework, since all detection results are then fed into a progressive refinement module to improve the confidence score over successively detected targets, which in turn increases the reliability and robustness of target object detection that support task allocation for robots. Overall, it is demonstrated that Mask RCNN outperforms YOLO on bounding box location accuracy, which may be beneficial to plan the path and navigate autonomous mobile robots toward detected target objects. On the other hand, the detection speed of YOLO is almost twice faster than that of Mask RCNN when running on either a GPU or a CPU. In spite of its longer processing time, the Mask RCNN topology reveals fast enough to be integrated on a CPU which could more easily be embedded on a mobile robot, with the only constraint that the framerate of the camera that capture successive images is adequate to support the average target object detection time at every iteration.

The progressive refinement framework was designed and implemented. It leverages the detection results generated by classical object detectors, but further processes the confidence scores and applies data fusion to iteratively refine the confidence in the recognition of specific classes of objects. The Bayesian and D-S theory based fusion methods are extensively evaluated to form the core element of the progressive refinement stage. Both approaches demonstrate the capability to efficiently support progressive refinement of the confidence score on object detection while images are captured successively by a robot as it moves toward initially detected target objects. However, the D-S theory based fusion refinement shows higher robustness on cases where targets are not correctly detected systematically. Moreover the D-S theory framework provides additional information on the certainty of the detected targets with a confidence interval. These advantages may benefit the robot coordination stage by decreasing the inherent ambiguity that remains on detected targets, leading to more responsive task allocation for the robotic agents. As a result, the Mask RCNN detector combined with the D-S theory based fusion method makes the proposed

progressive refinement framework reliable and robust for collaborative robots task allocation applications.

5.2 Contributions

This thesis makes the following contributions to the field of computer vison with a focus on the application of CNN-based target object detection for collaborative multi-robot task allocation:

- The design, implementation and experimental evaluation of a novel progressive refinement method to iteratively improve the confidence on detected target objects, which reinforces response capability and reliability in collaborative robotic systems.
- Adaptations of the backbone and region proposal generation for CNN-based two-stage target object detector Mask RCNN architecture to reinforce the detection capability of small scale target objects imaged over a large distance.
- An expanded annotated image dataset [97] with five predefined classes, which are relevant for collaborative robots task allocation in the context of indoor search-and-rescue scenarios.

This research led to the publication of [90] and [98].

5.3 Future work

In Chapter 4, the validation on the proposed approach was conducted based on static scenes where images are successively captured by the camera mounted on a mobile robot but objects are not moving. This imaging approach provided a series of coherent images with increasing visual details about the target objects present in the image. However, since it is difficult to accurately estimate the position of the target based only on 2D image coordinates provided by a single camera, the bounding box information and mask segmentation which could be used to navigate the robot toward the target was not fully utilized in the current work. Future work will investigate 3D target object detection, by leveraging depth sensors embedded on the robot. This will better support the localization process and help progressively and more accurately navigate the robot toward the target. The additional depth information will also require further adaptation of the combined CNN and progressive refinement object detection frameworks to fully leverage the data that will be made available.

Another extension of this work will be to integrate the object detection module and progressive refinement stage as a uniform framework, which will make it more efficient for embedded processing with limited CPU computational power on the robot. As such, image understanding and target recognition will perform faster and further improve the response capability of autonomous swarm robotic systems via efficient task allocation and optimal navigation under live visual guidance.

References

[1] Y. LeCun, K. Kavukcuoglu and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE international symposium on circuits and systems,* pp. 253-256, May 2010.

[2] D. Scherer, A. Müller and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *International conference on artificial neural networks,* Sep. 2010.

[3] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data,* vol. 2, no. 1, p. 1, 2015.

[4] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing,* vol. 187, pp. 27-48, 2016.

[5] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[6] Y. Boureau, J. Ponce and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10),* pp. 111-118, 2010.

[7] K. He, X. Zhang, S. Ren and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence,* vol. 37, no. 9, pp. 1904-1916, 2015.

[8] B. Xu, N. Wang, T. Chen and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*.

[9] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018, *arXiv:1811.03378*.

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[11] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[12] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective,* pp. 261-276, 1995.

[13] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[14] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107-116, 1998.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010.

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European conference on computer vision*, pp. 818-833, Sep. 2014.

[17] K. Simonyan, A. Vedaldi and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv:1312.6034*.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9, 2015.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818-2826, 2016.

[21] C. Szegedy, S. Ioffe, V. Vanhoucke and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *Thirty-first AAAI conference on artificial intelligence*, Feb. 2017.

[22] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016.

[23] G. Huang, Z. Liu, L. Van Der Maaten and K. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700-4708, 2017.

[24] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492-1500, 2017.

[25] J. Hu, L. Shen and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132-7141, 2018.

[26] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, pp. I-I, 2001.

[27] P. Viola and M.J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137-154, 2004.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886-893, Jun. 2005.

[29] P. Felzenszwalb, D. McAllester and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," *2008 IEEE conference on computer vision and pattern recognition*, pp. 1-8, Jun. 2008.

[30] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91-110, 2004.

[31] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509-522, 2002.

[32] P. F. Felzenszwalb, R. B. Girshick and D. McAllester, "Cascade object detection with deformable part models," *2010 IEEE Computer society conference on computer vision and pattern recognition*, pp. 2241-2248, Jun. 2010.

[33] P. F. Felzenszwalb, R. Girshick, D. McAllester and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627-1645, 2009.

[34] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587, 2014.

[35] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142-158, 2015.

[36] K. Van de Sande, J. Uijlings, T. Gevers and A. Smeulders, "Segmentation as selective search for object recognition," *2011 International Conference on Computer Vision*, pp. 1879-1886, Nov. 2011.

[37] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440-1448, 2015.

[38] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, pp. 91-99, 2015.

[39] J. Dai, Y. Li, K. He and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, pp. 379-387, 2016.

[40] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117-2125, 2017.

[41] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961-2969, 2017.

[42] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431-3440, 2015.

[43] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788, 2016.

[44] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271, 2017.

[45] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, *arXiv:1804.02767*.

[46] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. Berg, "Ssd: Single shot multibox detector," *European conference on computer vision*, pp. 21-37, Oct. 2016.

[47] T. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980-2988, 2017.

[48] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734-750, 2018.

[49] A. Newell, K. Yang and J. Deng, "Stacked hourglass networks for human pose estimation," *European conference on computer vision*, pp. 483-499, 2016.

[50] M. Everingham, L. Van Gool, C. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303-338, 2010.

[51] M. Everingham, S. Eslami, L. Van Gool, C. Williams, J. Winn and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98-136, 2015.

[52] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. Zitnick, "Microsoft coco: Common objects in context," *European conference on computer vision*, pp. 740-755, Sep. 2014.

[53] J. Deng, W. Dong, R. Socher, L. Li, K. Li and F. Li, "Imagenet: A large-scale hierarchical image database," *2009 IEEE conference on computer vision and pattern recognition*, pp. 248-255, 2009.

[54] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein and A. Berg, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211-252, 2015.

[55] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," 2018, *arXiv:1811.00982*.

[56] P. Dollár, C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: A benchmark," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 304-311, Jun. 2009.

[57] P. Dollár, C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, pp. 743-761, 2011.

[58] M. Koestinger, P. Wohlhart, P. Roth and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," *IEEE international conference on computer vision workshops (ICCV workshops)*, pp. 2144-2151, 2011.

[59] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354-3361, Jun. 2012.

[60] S. Yang, P. Luo, C. Loy and X. Tang, "Wider face: A face detection benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5525-5533, 2016.

[61] D. Hoiem, Y. Chodpathumwan and Q. Dai, "Diagnosing error in object detectors," *European conference on computer vision*, pp. 340-353, Oct. 2012.

[62] R. Rajaram, E. Ohn-Bar and M. Trivedi, "RefineNet: Iterative refinement for accurate object localization," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1528-1533, Nov. 2016.

[63] M. C. Roh and J. Lee, "Refining faster-RCNN for accurate object detection," *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, Nagoya, pp. 514-517, 2017

[64] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154-6162, 2018.

[65] S. Challa and D. Koks, "Bayesian and dempster-shafer fusion," *Sadhana*, vol. 29, no. 2, pp. 145-174, 2004.

[66] M. Liggins II, D. Hall and J. Llinas, Handbook of multisensor data fusion: theory and practice, CRC press, 2017.

[67] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Classic works of the Dempster-Shafer theory of belief functions*, pp. 57-72, 2008.

[68] G. Shafer, A mathematical theory of evidence, Princeton: Princeton university press, 1976.

[69] K. Khoshelham, S. Nedkov and C. Nardinocchi, "A comparison of Bayesian and evidence-based fusion methods for automated building detection in aerial data," *International Society for Photogrammetry and Remote Sensing*, Beijing, China, 2008.

[70] U. Knauer and U. Seiffert, "A comparison of late fusion methods for object detection," *IEEE International Conference on Image Processing*, pp. 3297-3301, Sep. 2013.

[71] H. Lee, H. Kwon, R. Robinson, W. Nothwang and A. Marathe, "Dynamic belief fusion for object detection," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1-9, Mar. 2016.

[72] Y. Cao, H. Lee and H. Kwon, "Enhanced object detection via fusion with prior beliefs from image classification," *IEEE International Conference on Image Processing (ICIP)*, pp. 920-924, Sep. 2017.

[73] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking," *IEEE International Conference on Image Processing (ICIP)*, pp. 3464-3468, Sep. 2016.

[74] R. E. Kalman, "A new approach to linear filtering and prediction problems," pp. 34-45, 1960.

[75] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83-97, 1955.

[76] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi and J. Yan, "Poi: Multiple object tracking with high performance detection and appearance feature," *European Conference on Computer Vision*, pp. 36-42, Oct. 2016.

[77] Z. Zhou, J. Xing, M. Zhang and W. Hu, "Online multi-target tracking with tensor-based high-order graph matching," *24th International Conference on Pattern Recognition (ICPR)*, pp. 1809-1814, Aug. 2018.

[78] H. Kieritz, W. Hubner and M. Arens, "Joint detection and online multi-object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1459-1467, 2018.

[79] S. J. Kim, J. Y. Nam and B. C. Ko, "Online tracker optimization for multi-pedestrian tracking using a moving vehicle camera," *IEEE Access,* vol. 6, pp. 48675-48687, 2018, doi: 10.1109/ACCESS.2018.2867621.

[80] L. Zhang, L. Lin, X. Liang and K. He, "Is faster r-cnn doing well for pedestrian detection?," *European conference on computer vision,* pp. 443-457, Oct. 2016.

[81] J. Mao, T. Xiao, Y. Jiang and Z. Cao, "What can help pedestrian detection?," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 3127-3136, 2017.

[82] D. Xu, W. Ouyang, E. Ricci, X. Wang and N. Sebe, "Learning cross-modal deep representations for robust pedestrian detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 5363-5371, 2017.

[83] Y. Xiang, W. Choi, Y. Lin and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1903-1911, 2015.

[84] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 2040-2049, 2017.

[85] A. Mousavian, D. Anguelov, J. Flynn and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 7074-7082, 2017.

[86] B. Li, T. Zhang and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," 2016, *arXiv:1608.07916*.

[87] M. Simony, S. Milzy, K. Amendey and H.M. Gross, "Complex-yolo: An Euler-region-proposal for real-time 3d object detection on point clouds," in *Proceedings of the European Conference on Computer Vision (ECCV),* 2018.

[88] K. He and J. Sun, "Convolutional neural networks at constrained time cost," In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353-5360, 2015.

[89] W. Adbulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow" 2007. [Online]. Available: https://github.com/matterport/Mask_RCNN. (Accessed: Jan. 10, 2019).

[90] W. Wu, P. Payeur, O. Al-Buraiki and M. Ross, "Vision-Based Target Objects Recognition and Segmentation for Unmanned Systems Task Allocation," In *International Conference on Image Analysis and Recognition*, pp. 252-263, Aug. 2019.

[91] M. Jaderberg, K. Simonyan and A. Zisserman, "Spatial transformer networks," *Advances in neural information processing systems*, pp. 2017-2025, 2015.

[92] "Google cloud platform(GCP)," [Online]. Available: https://cloud.google.com/. (Accessed: Mar. 01, 2019).

[93] M. Abadi, P. C. J. Barham, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard and M. Kudlur, "Tensorflow: A system for large-scale machine learning," *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265-283, 2016.

[94] B. Russell, A. Torralba, K. Murphy and W. Freeman, "LabelMe: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157-173, 2008.

[95] ROBOTICS, "ROBOTIS e-Manual: TurtleBot3, " 2020. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. (Assessed: Mar. 01, 2019).

[96] F.S. Bashiri, E. LaRose, P. Peissig and A.P. Tafti, "MCIndoor20000: a fully-labeled image dataset to advance indoor objects detection," *Data in brief*, vol. 17, pp. 71-75, 2018.

[97] W. Wu, V. Piesiecki and P. Payeur, University of Ottawa, Annotated image dataset for indoor search-and-rescue applications. https://github.com/wenbo-uo/Dataset-5-class

[98] O. Al-Buraiki, W. Wu and P. Payeur, "Probabilistic Allocation of Specialized Robots on Targets Detected Using Deep Learning Networks," *Robotics*, vol. 9, no. 3, art. 54, 2020.

Appendix A

A.1 Progressive refinement with Mask RCNN and Dempster-Shafer theory fusion

This section presents detailed experimental results in relation with the analysis in Section 4.3.2.

	Image input	**Recognition results (from detector)**		**D-S theory based refinement**		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(2-1)	 $Input_1$	 D_1	stairs: $d_1^1, s_{11}^4 = 99.1$	Fusion1 (D_1)	stairs: $S_{11}(4) = 99.10$	-
(2-2)	 $Input_2$	 D_2	stairs: $d_2^1, s_{21}^4 = 98.7$	Fusion2 (S_1, D_2)	stairs: $S_{21}(4) = 99.98$	[99.99, 99.99]
(2-3)	 $Input_3$	 D_3	stairs: $d_3^1, s_{31}^4 = 99.5$	Fusion3 (S_2, D_3)	stairs: $S_{31}(4) = 99.99$	[99.99, 99.99]
(2-4)	 $Input_4$	 D_4	stairs: $d_4^1, s_{41}^4 = 98.1$	Fusion4 (S_3, D_4)	stairs: $S_{41}(4)$ =99.99	[99.99, 99.99]
(2-5)	 $Input_5$	 D_5	stairs: $d_5^1, s_{51}^4 = 96.6$	Fusion5 (S_4, D_5)	stairs: $S_{51}(4) = 99.99$	[99.99, 99.99]

Figure A.1: Target object detection results with D-S theory based progressive refinement strategy with application of the Mask RCNN detector on the second sequence.

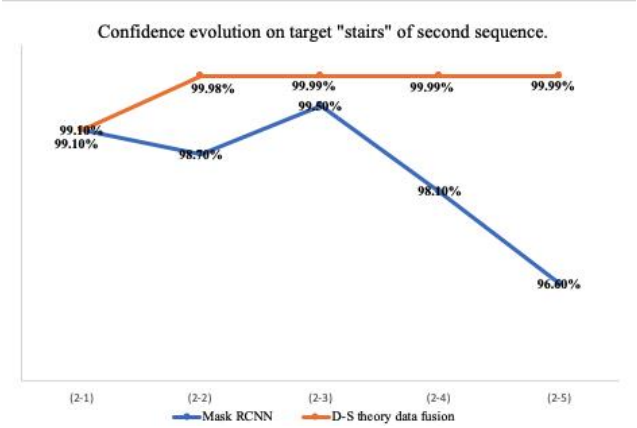
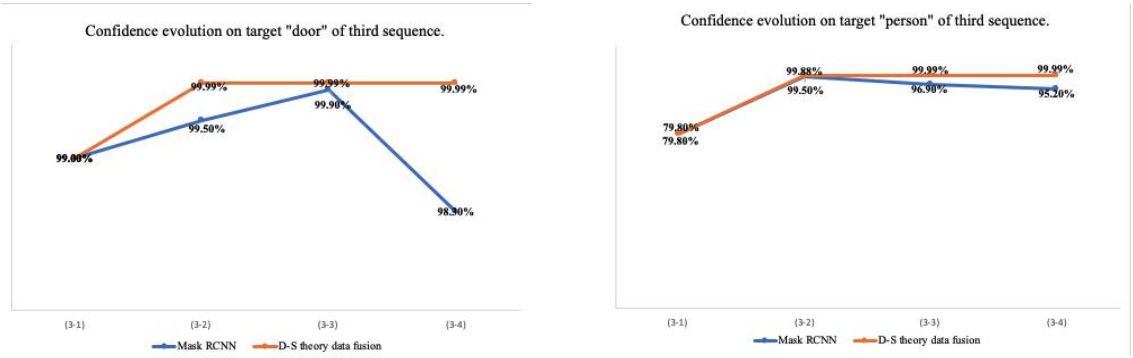


Figure A.2: Evolution of confidence score on recognized object in the second sequence without and with D-S theory based progressive refinement (Mask RCNN detector).

	Image input	**Recognition results (from detector)**		**D-S theory based refinement**		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(3-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 99.0$ person: $d_1^2, s_{12}^2 = 79.8$	Fusion1 (D_1)	door: $S_{11}(1) = 99.00$ person: $S_{12}(2) = 79.80$	-
(3-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 99.5$ person: $d_2^2, s_{22}^2 = 99.5$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.99$ person: $S_{22}(2) = 99.88$	door: $[99.99, 99.99]$ person: $[99.88, 99.97]$
(3-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 99.9$ person: $d_3^2, s_{32}^2 = 96.9$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.99$ person: $S_{32}(2) = 99.99$	door: $[99.99, 99.99]$ person: $[99.99, 99.99]$
(3-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 98.3$ person: $d_4^2, s_{42}^2 = 95.2$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.99$ person: $S_{42}(2) = 99.99$	door: $[99.99, 99.99]$ person: $[99.99, 99.99]$

Figure A.3: Target objects detection results with D-S theory based progressive refinement strategy with application of the Mask RCNN detector on the third sequence.

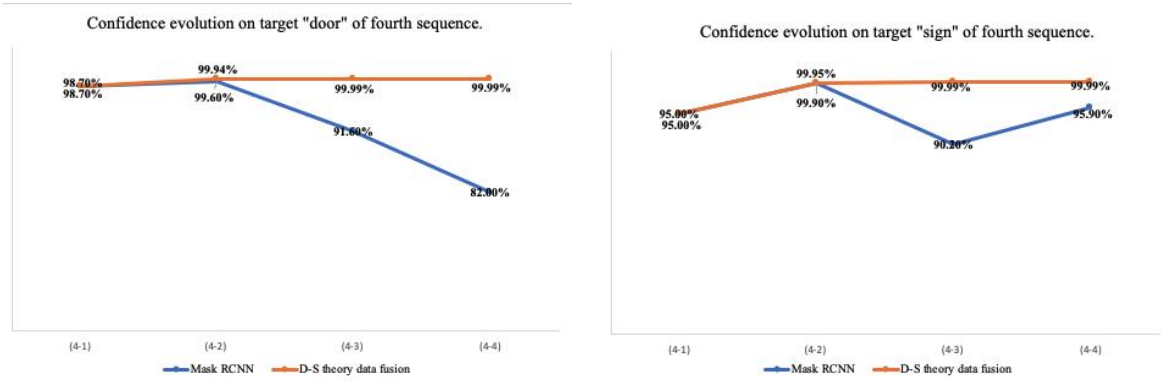


(a) (b)

Figure A.4: Evolution of confidence score on recognized objects in the third sequence without and with D-S theory based progressive refinement (Mask RCNN detector).

	Image input	**Recognition results (from detector)**		**D-S theory based refinement**		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(4-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 98.7$ sign: $d_1^2, s_{12}^3 = 95.0$	Fusion1 (D_1)	door: $S_{11}(1) = 98.70$ sign: $S_{12}(3) = 95.00$	-
(4-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 99.6$ sign: $d_2^2, s_{22}^3 = 99.9$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.94$ sign: $S_{22}(3) = 99.95$	door: [99.99, 99.99] sign: [99.99, 99.99]
(4-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 91.6$ sign: $d_3^2, s_{32}^3 = 90.2$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.99$ sign: $S_{32}(3) = 99.99$	door: [99.99, 99.99] sign: [99.99, 99.99]
(4-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 82.0$ sign: $d_4^2, s_{42}^3 = 95.9$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.99$ sign: $S_{42}(3) = 99.99$	door: [99.99, 99.99] sign: [99.99, 99.99]

Figure A.5: Target objects detection results with D-S theory based progressive refinement strategy with application of the Mask RCNN detector on the fourth sequence.



(a) (b)

Figure A.6: Evolution of confidence score on recognized objects in the fourth sequence without and with D-S theory based progressive refinement (Mask RCNN detector).

A.2 Progressive refinement with YOLO and Bayesian fusion

This section presents detailed experimental results in relation with the analysis in Section 4.3.3.

	Image input	**Recognition results (from detector)**		**Bayesian based refinement**	
			Score (%)	Iterative fusion	Confidence score (%)
(2-1)	 $Input_1$	 D_1	stairs: $d_1^1, s_{11}^4 = 65.0$	Fusion1 (D_1, Prior)	stairs: $S_{11}(4) = 65.0$
(2-2)	 $Input_2$	 D_2	stairs: $d_2^1, s_{21}^4 = 64.1$	Fusion2 (S_1, D_2)	stairs: $S_{21}(4) = 76.8$
(2-3)	 $Input_3$	 D_3	stairs: $d_3^1, s_{31}^4 = 63.8$	Fusion3 (S_2, D_3)	stairs: $S_{31}(4) = 85.4$
(2-4)	 $Input_4$	 D_4	stairs: $d_4^1, s_{41}^4 = 93.3$	Fusion4 (S_3, D_4)	stairs: $S_{41}(4) = 98.8$
(2-5)	 $Input_5$	 D_5	stairs: $d_5^1, s_{51}^4 = 97.2$	Fusion5 (S_4, D_5)	stairs: $S_{51}(4) = 99.9$

Figure A.7: Target object detection results with Bayesian based progressive refinement strategy with application of the YOLO detector on the second sequence.

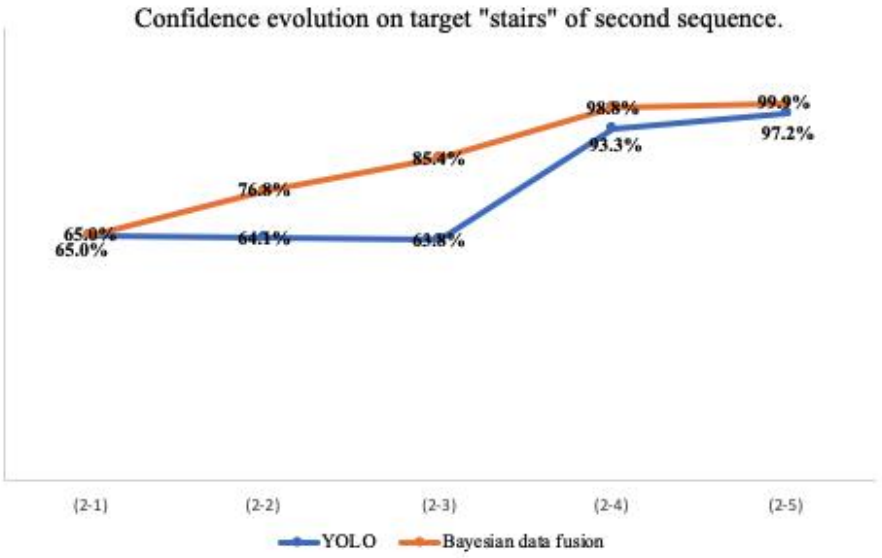


Figure A.8: Evolution of confidence score on recognized object in the second sequence without and with Bayesian based progressive refinement (YOLO detector).

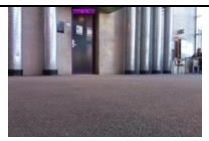
	Image input	Recognition results (from detector)		Bayesian based refinement	
			Score (%)	Iterative fusion	Confidence score (%)
(3-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 98.6$	Fusion1 (D_1, Prior)	door: $S_{11}(1) = 98.6$
(3-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 98.5$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.9$
(3-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 95.1$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.9$
(3-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 90.5$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.9$

Figure A.9: Target object detection results with Bayesian based progressive refinement strategy with application of the YOLO detector on the third sequence.

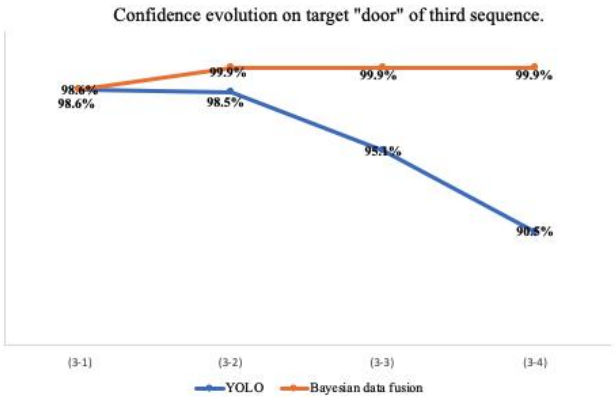
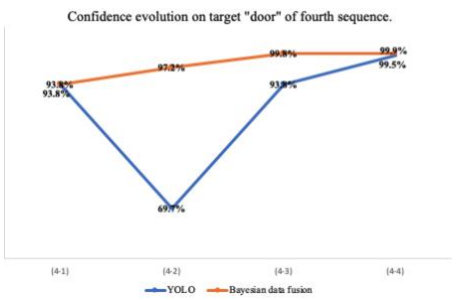


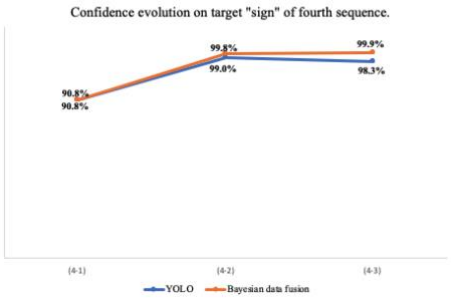
Figure A.10: Evolution of confidence score on recognized object in the third sequence without and with Bayesian based progressive refinement (YOLO detector).

	Image input	Recognition results (from detector)		Bayesian based refinement	
			Score (%)	Iterative fusion	Confidence score (%)
(4-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 93.8$	Fusion1 (D_1, Prior)	door: $S_{11}(1) = 93.8$
(4-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 69.7$ sign: $d_2^2, s_{22}^3 = 90.8$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 97.2$ sign: $S_{22}(3) = 90.8$
(4-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 93.8$ sign: $d_3^2, s_{32}^3 = 99.0$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.8$ sign: $S_{32}(3) = 99.8$
(4-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 99.5$ sign: $d_4^2, s_{42}^3 = 98.3$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.9$ sign: $S_{42}(3) = 99.9$

Figure A.11: Target objects detection results with Bayesian based progressive refinement strategy with application of the YOLO detector on the fourth sequence.



(a)



(b)

Figure A.12: Evolution of confidence score on recognized objects in the fourth sequence without and with Bayesian based progressive refinement (YOLO detector).

A.3 Progressive refinement with YOLO and Dempster-Shafer theory fusion

This section presents detailed experimental results in relation with the analysis in Section 4.3.4.

	Image input	**Recognition results (from detector)**		**D-S theory based refinement**		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(2-1)	 $Input_1$	 D_1	stairs: $d_1^1, s_{11}^4 = 65.0$	Fusion1 (D_1)	stairs: $S_{11}(4) = 65.00$	-
(2-2)	 $Input_2$	 D_2	stairs: $d_2^1, s_{21}^4 = 64.1$	Fusion2 (S_1, D_2)	stairs: $S_{21}(4) = 83.71$	stairs: $[83.71, 95.93]$
(2-3)	 $Input_3$	 D_3	stairs: $d_3^1, s_{31}^4 = 63.8$	Fusion3 (S_2, D_3)	stairs: $S_{31}(4) = 92.59$	stairs: $[92.59, 98.15]$
(2-4)	 $Input_4$	 D_4	stairs: $d_4^1, s_{41}^4 = 93.3$	Fusion4 (S_3, D_4)	stairs: $S_{41}(4) = 99.47$	stairs: $[99.47, 99.87]$
(2-5)	 $Input_5$	 D_5	stairs: $d_5^1, s_{51}^4 = 97.2$	Fusion5 (S_4, D_5)	stairs: $S_{51}(4) = 99.98$	stairs: $[99.98, 99.99]$

Figure A.13: Target object detection results with D-S theory based progressive refinement strategy with application of the YOLO detector on the second sequence.

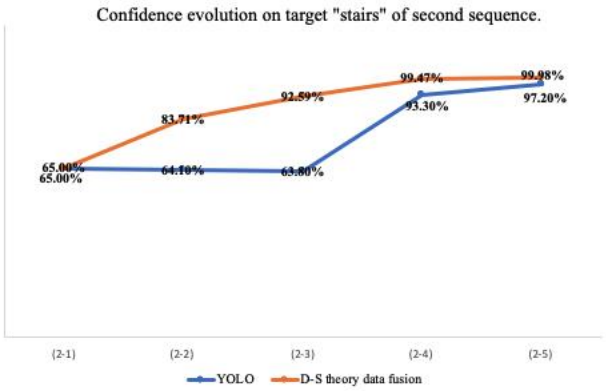


Figure A.14: Evolution of confidence score on recognized object in the second sequence without and with D-S theory based progressive refinement (YOLO detector).

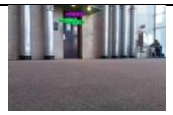
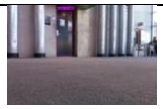
	Image input	**Recognition results (from detector)**		**D-S theory based refinement**		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(3-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 98.6$	Fusion1 (D_1)	door: $S_{11}(1) = 98.60$	-
(3-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 98.5$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 99.97$	$[99.97, 99.99]$
(3-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 95.1$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.99$	$[99.99, 99.99]$
(3-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 90.5$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.99$	$[99.99, 99.99]$

Figure A.15: Target object detection results with D-S theory based progressive refinement strategy with application of the YOLO detector on the third sequence.

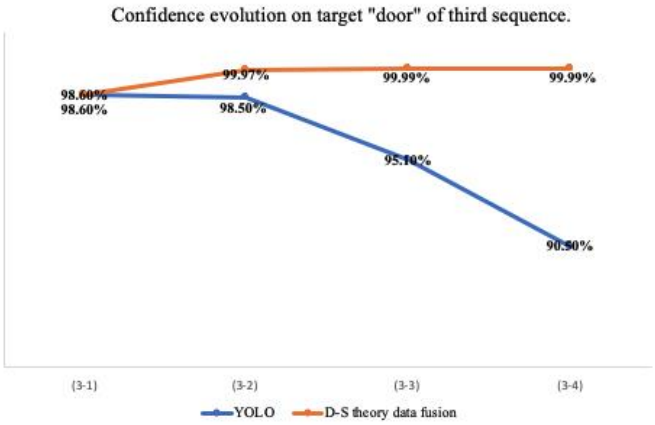
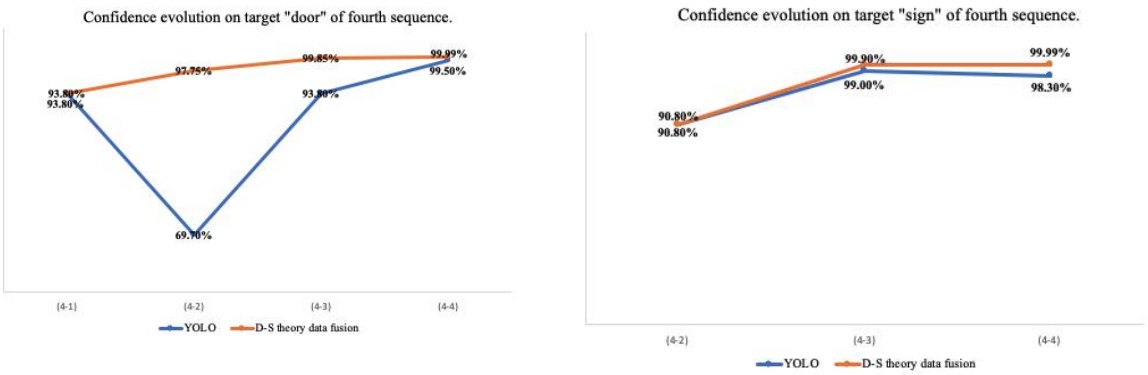


Figure A.16: Evolution of confidence score on recognized object in the third sequence without and with D-S theory based progressive refinement (YOLO detector).

	Image input	**Recognition results (from detector)**		**D-S theory based refinement**		
			Score (%)	Iterative fusion	Confidence score (%)	Confidence interval (%)
(4-1)	 $Input_1$	 D_1	door: $d_1^1, s_{11}^1 = 93.8$	Fusion1 (D_1)	door: $S_{11}(1) = 93.80$	-
(4-2)	 $Input_2$	 D_2	door: $d_2^1, s_{21}^1 = 69.7$ sign: $d_2^2, s_{22}^3 = 90.8$	Fusion2 (S_1, D_2)	door: $S_{21}(1) = 97.75$ sign: $S_{22}(3) = 90.80$	door: $[97.75, 99.44]$ -
(4-3)	 $Input_3$	 D_3	door: $d_3^1, s_{31}^1 = 93.8$ sign: $d_3^2, s_{32}^3 = 99.0$	Fusion3 (S_2, D_3)	door: $S_{31}(1) = 99.85$ sign: $S_{32}(3) = 99.90$	door: $[99.85, 99.96]$ sign: $[99.90, 99.97]$
(4-4)	 $Input_4$	 D_4	door: $d_4^1, s_{41}^1 = 99.5$ sign: $d_4^2, s_{42}^3 = 98.3$	Fusion4 (S_3, D_4)	door: $S_{41}(1) = 99.99$ sign: $S_{42}(3) = 99.99$	door: $[99.99, 99.99]$ sign: $[99.99, 99.99]$

Figure A.17: Target objects detection results with D-S theory based progressive refinement strategy with application of the YOLO detector on the fourth sequence.



(a) (b)

Figure A.18: Evolution of confidence score on recognized objects in the fourth sequence without and with D-S theory based progressive refinement (YOLO detector).