Department of Computer Science

People Detection with Convolutional Neural Networks in RGB-D Data

Kaiyang Zhou

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering

September 2016 | CSMSC-16

Declaration:

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Kaiyang Zhou, September 2016

Abstract

Deep learning, especially referring to *Convolutional Neural Network* (CNN), has significantly changed the computer vision community. In the domains of image classification and generic object detection, the results obtained by deep learning-based methods outperform other non-deep learning methods by considerable margins. Deep learning-based methods have also been applied to RGB-D object recognition/detection tasks where the results show that adding depth information is helpful to improve the performance obtained by using merely colour images.

In this dissertation, we propose a deep learning-based people detection system that works in RGB-D data. This people detection system mainly consists of two modules, namely a *region-of-interest* (ROI) selector and a CNN-based people detector. The ROI selector is used to reduce the search space in images. It leverages the depth information in RGB-D data to remove the ground plane in images and produce a list of candidate proposals. The people detector is composed of two CNNs that work independently in RGB and depth images. It aims to classify the candidate proposals by fusing information aggregated from RGB and depth data. The two networks are identical in architecture, in which we embed the ROI-pooling layers that were proposed in *Fast R-CNN* (a state-of-the-art object detector). With the ROI-pooling layer, our networks are able to perform convolutions on an entire image, which promotes the sharing of computations on convolutions, significantly reducing the runtime. To overcome the scarcity of large RGB-D training data, we initialise the two networks with the same weights that were pre-trained in ImageNet and perform fine-tuning to enable them to adapt to target data. To make depth images compatible with the input of pre-trained model, we propose a depth-encoding method called *CECD-encoding*, which is used to transform depth images from one-channel to three-channel. By applying this transformation, the important features contained in the original depth images such as object shape can be well preserved. The encoded depth images are close to colour images in terms of information contained so that less efforts are required to adjust the pre-trained model. We evaluate the effectiveness of our people detection system in a public RGB-D people detection dataset, by which we verify that adding depth to the CNN detection is able to improve the detection accuracy. To the best of our knowledge, our work is the first one that applies CNNs in RGB-D data for people detection. The main achievements are summarised as follows:

1. We proposed an efficient ROI selection method, which significantly reduces the search space in images.

2. We proposed a CNN-based people detector, which fuses colour and depth information to perform people detection.

3. We proposed an effective depth-encoding method, which transforms depth images from single-channel to three-channel, while preserving the important features contained in original images.

4. We re-annotated the RGB-D people detection dataset to fill some missing annotations.

Acknowledgements

First, I would like to thank my supervisor, Dr. Adeline Paiement, for her continuous support and encouragement. I am very grateful for her help of providing the essential GPUs, without which I would not be able to complete my work. I very appreciate her valuable comments and patient guidance on my various reports, from which I have learnt a lot. I would like to thank my supervisor, Prof. Majid Mirmehdi, who inspired me to propose this project, encouraged me and provided me with useful feedbacks on my dissertation.

I would like to thank Dr. Sion Hannuna and Dr. Massimo Camplani, the SPHERE project researchers, for providing me with essential datasets. I would like to thank my assessors, Dr. Dima Damen and Dr. Anne Roudaut, for their constructive feedbacks in my poster presentation. I would like to thank my friend, Joseph Finnegan, for being the proofreader of my dissertation. I would also like to thank Callum Wright, a member of the ACRC team, for helping me install the essential software in Blue Crystal.

Finally, I would like to thank my parents, who have been unconditionally supporting me both in finance and emotion throughout my Master study.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In this chapter, we provide an introduction and motivation for this research project, explaining why we are interested in using convolutional neural networks for people detection in RGB-D images and what we expect to achieve. We commence by introducing the background and motivation. Then we demonstrate the aim and objectives. The structure of this thesis is outlined at the end of this chapter.

1.1 Background and Motivation

People detection is an important technology for many computer vision applications such as video surveillance, robotic navigation and vehicle-driving assistance. A key factor affecting the performance in people detection is feature extraction. The extracted features are preferred to be discriminative enough such that people can be distinguished from complex background scenes. As one of the most popular feature extraction methods, HOG descriptor [8] has been widely used in people detection. HOG descriptor captures the distribution of gradients in a given image patch, which has been proven to be useful in people detection. However, HOG descriptor does not encode high-level semantic information since it is computed mainly based on gradients (low-level). RGB images provide not only edge information, but also colour and texture information, which cannot be captured by HOG.

In recent years, *Convolutional Neural Networks* (CNNs) have pushed computer vision forward to a new level. For example, the accuracy achieved by CNNs in some large image classification datasets such as ImageNet[1] outperforms other non-deep learning methods by a large margin [9]. In generic object detection, the winners of some well-known benchmarks have been occupied by CNN-based object detectors such as Fast R-CNN [10]. Such huge progresses made by CNNs can be attributed to their distinctive feature-learning ability. Unlike hand-crafted feature extractors such as HOG, CNNs automatically learn feature representations from raw image pixels in a supervised way. The learned features are distributed from low-level to high-level, forming a hierarchical structure. Such hierarchical features satisfy the rule of a good visual representation, which is 'pixels $\rightarrow$ edglets $\rightarrow$ motifs $\rightarrow$ parts $\rightarrow$ objects' [11].

By taking advantage of this nice property, many computer vision researchers have applied CNNs to people detection and have achieved state-of-the-art performance in detecting pedestrians [12, 13]. Nevertheless, two limitations are posed in the current literature. (1) Most CNN-based people detectors are computationally slow. This is because CNNs are repeatedly applied to each candidate window in the image domain without sharing computations on convolutions [14]. In terms of proposal methods, the traditional sliding-window for detection is prohibitively expensive because windows of multiple scales are scanned at each image pixel resulting in tens of thousands of windows to be examined. (2) To the best of our knowledge, no one has applied CNNs to particularly detect people in RGB-D images. What are RGB-D images? RGB-D images contain both colour and depth images. Each pixel in a depth image encode the distance from camera. In the past, depth images were not widely used in research because devices producing depth data were either expensive (e.g. Time-of-Flight camera) or not straightforward to use (e.g. stereo camera). With the invention of Microsoft Kinect, high-quality depth images are now available at a much lower cost, and the way to obtain depth data is relatively easier than that of stereo cameras which requires hard efforts on calibration. But why do we need depth instead of using the pure RGB? In general, there are two reasons to support the use of depth. First, RGB images are known to be rich in texture and colour, however, the visual

[1] ImageNet Large Scale Visual Recognition Challenge: http://www.image-net.org/challenges/LSVRC/

quality is easily downgraded by poor illuminations. Fortunately, the depth sensor in the Kinect is robust to lighting changes, but it is limited to range. This complementary nature can be utilised to solve fundamental problems in computer vision [1]. Second, the boundary information, or shape, is more strongly activated in depth images, which provide opportunities for us to use CNNs to extract semantically meaningful features.

How do we intend to overcome these two limitations? For the first limitation, we intend to use the idea of ROI-pooling, which is proposed by Girshick in Fast R-CNN [10]. The CNN part in Fast R-CNN performs convolutions on the entire input image to get image feature maps and uses the ROI-pooling layer to pool a fixed-length feature vector in each candidate window that is projected from the image domain to the feature maps. Each feature vector is then processed by the following fully connected layers. Such a processing pipeline significantly reduces the runtime by sharing computations on convolutions. In addition, we can take advantage of depth images to do *region-of-interest* (ROI) selection i.e. to utilise depth to reduce the search space in images. With the help of depth, a 3D scene of an image can be reconstructed, from which we can remove the ground plane as well as pixels close to it. In doing so, irrelevant pixels are not required to be scanned by a sliding window. We can further constrain the sliding window to one scale for one pixel by using the depth. For the second limitation, we plan to apply CNNs to learn feature representations from both colour and depth images, and design an effective way to combine them. Although depth images only encode distance information in each pixel, they share some common high-level semantic features with RGB images such as object boundaries. In particular, the shapes of head and shoulders of people are well captured by depth images, which can serve as a strong cue for classification[2]. Fusing RGB and depth information is able to produce better performance which has been proven in some RGB-D object detection/recognition applications [15, 7, 16]. Therefore, by combining RGB and depth, we expect to improve the detection accuracy achieved by using RGB only.

1.2 Aim and Objectives

The goal of this project is to develop a real-time RGB-D people detection system based on CNNs. This system mainly detects the upper body parts of people in images and is used for indoor settings. We will apply CNNs to learning feature representations independently from colour and depth images, and design an effective way to combine them to do inference. In doing so, we expect to improve the detection accuracy achieved by using only colour images. We aim to reduce unnecessary computations for the people detector by leveraging the idea of ROI-pooling proposed in the recent state-of-the-art object detector, Fast R-CNN. With the ROI-pooling layer, the CNN can perform convolutions on the entire input image. In addition, we plan to design an ROI selection method based on depth, expecting to reduce the search space for the people detector and further reduce the runtime. The objectives of this project are specified as follows:

1. To design and develop an ROI selection method to reduce the search space for the people detector. The output of this method is a set of candidate windows that are going to be examined by the people detector.

2. To design and develop a CNN-based people detector that fuses colour and depth information to do inference.

3. To conduct experiments to prove whether adding depth in CNNs for detection can improve the performance achieved by using colour images only, and to evaluate the effectiveness of the ROI selection method.

4. To write the final dissertation to summarise this project and indicate the future work.

[2]Note that we aim to detect only the upper body of people.

1.3 Structure of Dissertation

The rest of this dissertation is organised as follows. In Chapter 2, we provide a comprehensive review on the literature, mainly in three domains: basic knowledge about the Kinect sensor, use of depth for ROI selection, and CNNs and applications in detection. In Chapter 3, we describe the proposed people detection system in detail. In Chapter 4, we design and conduct experiments and perform a thorough analysis on the experimental results. In Chapter 5, we conclude our project and indicate the future work.

Chapter 2

Literature Review

In this chapter, we provide a comprehensive review of the literature related to our project. The structure of this chapter is organised as follows. In section 2.1, we briefly review the Kinect sensor, particularly for its sensor mechanism and the data format. In section 2.2, we review the depth-based ROI selection methods. In section 2.3, we review CNNs and current state-of-the-art CNN-based detectors. In section 2.4 we conclude this chapter.

2.1 A Brief Review of the RGB-D Sensor — Kinect

The motivation behind this brief research in Kinect is that, in the end, we will perform our proposed people detector on RGB-D images produced by the Kinect sensor, so it is necessary to know what the output of the Kinect sensor is and whether we can directly use the output images. In general, the research in this subject mainly aims to answer two questions, namely *i*) what is the output of the Kinect, *ii*) How to use the data generated by the Kinect. For more comprehensive investigations into the mechanism of the Kinect sensor and its applications, the reader is referred to [1] and [17].

2.1.1 What is Kinect

Kinect is a RGB-D camera initially introduced in November 2010 by Microsoft. As an accessory of Xbox 360, it produces colour and depth images simultaneously, enabling the human motion to be well captured in a 3-D sense. The complementary nature of the depth and visual (RGB) information provided by the low-cost Kinect sensor has also been used in many computer vision applications, including human detection and tracking [18], hand gesture analysis [19], pose estimation [20], indoor 3-D reconstruction [21] and etc.

2.1.2 Sensor Mechanism

As shown in Fig.2.1, the Kinect sensor consists of an infrared (IR) projector, an IR camera and a RGB camera. The IR projector and the IR camera constitute the depth sensor. The colour image is obtained by the RGB camera while the depth map is constructed by the IR camera working together with the IR projector. The depth is measured by a structured light method. In particular, an IR speckle dot pattern is projected from the IR projector to the 3-D space, and the reflected IR speckles are received by the IR camera. The speckle is invisible to the RGB camera but can be captured by the IR camera. Since each of the projected dot pattern is unique, it is feasible to match the observed dot pattern with the known pattern by the IR camera [1].

Due to the Kinect using infrared light as the measure tool, the guaranteed detection range is limited between 0.8m and 5m. In fact, depth can be estimated further than 5m, but the accuracy decreases as the distance increases. In addition, the IR sensor is sensitive to sunlight. Therefore the Kinect is more suitable to be used in an indoor environment. For outdoor usage, other depth devices such as stereo camera [4] or Time-of-Flight camera [22] can be the alternative. Another issue should be noticed that some materials do not reflect well with the infrared light so there are some missing or inaccurate depth values in the depth image [17].

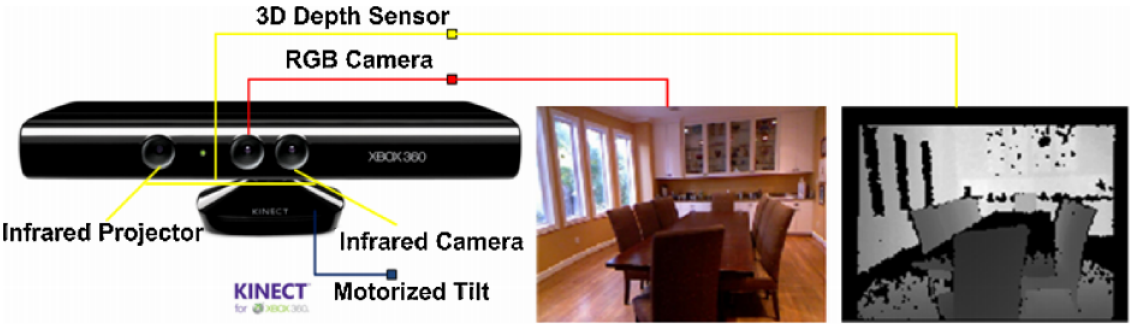


Figure 2.1: Hardware configuration of Kinect [1]. In the right-hand side, two example images corresponding to the colour image (*left*) and the depth image (*right*) are provided.

2.1.3 RGB-D Data Format

The colour images obtained by the RGB camera have the three traditional channels i.e. R, G and B, with 8 bits per channel and they have the size of 640 × 480. The depth images created by the depth sensor also have the same size of 640 × 480 but they are stored in 16 bits. There is only one channel in depth images, which encodes distance information i.e. the distance between the camera and an object by which the IR light is reflected.

Figure 2.2: *Left*: colour image. *Right*: depth image. Missing depth pixels in the depth image are represented by red [23].

The data generated by the Kinect sensor cannot be directly used because a pixel in the RGB image corresponds to a different pixel in the depth image [17]. This is caused by the difference of position of the RGB camera and the IR camera (2.5 cm), which can be seen in Fig.2.1. Therefore, it is necessary to spatially align the colour image and the depth image. In addition, the raw depth data are very noisy and due to multiple reflections, and transparent objects or scattering in certain surfaces (such as human tissue and hair), some pixels even have no depth values [1] (see Fig.2.2). To overcome these problems, technique such as fast max-filter [20] or more complex bilateral-based filter [23] can be performed to improve the readability and usability of the depth image (see Fig.2.3).

2.2 Use of Depth Information for ROI selection

In people detection tasks, it is observable that the detecting window does not have to exhaustively scan the entire image regions; only those where people are more likely to appear are required to be searched. These regions that are highly possible to contain people (or target objects) are so-called *region-of-interest* (ROI). The motivation behind the ROI selection is thus to speed up the detection,

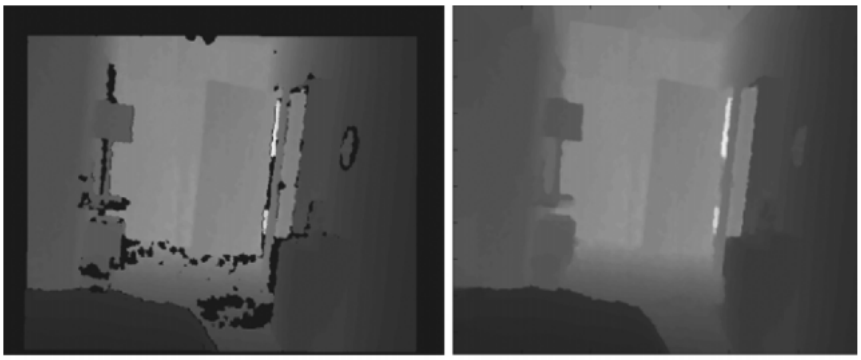


Figure 2.3: *Left*: raw depth image. *Right*: depth image after filtering. Missing depth pixels in the raw depth image are represented by dark black [1].

which matches our goal of achieving a real-time performance in people detection.

2D images only provide colour or grayscale information, which is insufficient to reflect the geometric sense of environments. However, by adding depth images, reconstructing the 3D world captured in an image becomes achievable, which is useful to ROI selection. For example, given the 3D information for each pixel, ground floors and ceilings in an image can be detected and then removed to extract ROIs for people detection. Moreover, depth information enriches the feature representations contained in a single pixel, which helps boost the classification of a pixel into foreground/background.

In this section, we provide a review of the use of depth information in RGB-D images for ROI selection. The structure of this section is organised as follows: section 2.2.1 discusses using geometric inference to extract ROIs; section 2.2.2 presents foreground segmentation methods based on depth background modelling; section 2.2.3 describes statistical modelling approaches using depth and colour for background subtraction; section 2.2.4 summarises this section.

2.2.1 Geometric Plane Removal

The evidence that supports the geometric plane removal for ROI selection is that, people are usually located in areas that are higher than the ground level while lower than ceilings. From a geometric point of view, a plane in the 3-dimensional Euclidean space can be easily determined using 3 non-collinear points. However, the 3-D coordinates for each pixel obtained from the colour and depth images are not absolutely accurate because there are noises with the depth data. In order to solve this problem for practical applications, many researchers employ the popular algorithm called random sample consensus (RANSAC) [24] to iteratively select an optimal plane that best fits the data.

Zhang et al. [25] remove the ground plane and ceiling by using a RANSAC-based plane fitting algorithm, and also reject those points that are too close to the two planes. The plane parameters obtained in the previous frame become the initial input to the algorithm in the current frame. A density distribution along the depth dimension is calculated using the remaining points, and the nonparametric Parzen window algorithm [26] is applied to locate the local maxima, which are called depth-of-interests (DOIs) in [25]. They then group the 3-D points within these DOIs via connected component analysis to generate candidates. A cascade of detectors is performed on these candidates to detect the presence of humans. Another feature in their work that is worth to be studied is that, they take maximum advantage of geometric property of 3-D information to construct weak classifiers which are embedded in the cascade structure: a height-based detector used to reject those points that are out of a reasonable height range; a size-based detector used to reject those candidates that are greater than a max-size threshold; a surface-based detector used to reject extra planes such as walls and desk surfaces.

Munaro et al. [27] apply a RANSAC-based least square method to remove the ground plane and divide the remaining points into different clusters by labelling neighbouring 3D points based on their Euclidean distance. To detect the head of each individual, they further segment the clusters into sub-clusters according to the local maxima of a height map created along the image x axis. For each sub-cluster, they apply the *Histogram of Oriented Gradients* people detector [8] to compute its confidence of being a human, enabling people close to each other to be separately detected. In the other work [28], Munaro et al. estimate the ground plane with the Hough-based method proposed in [29] and further limit the search space with a constrained height range.

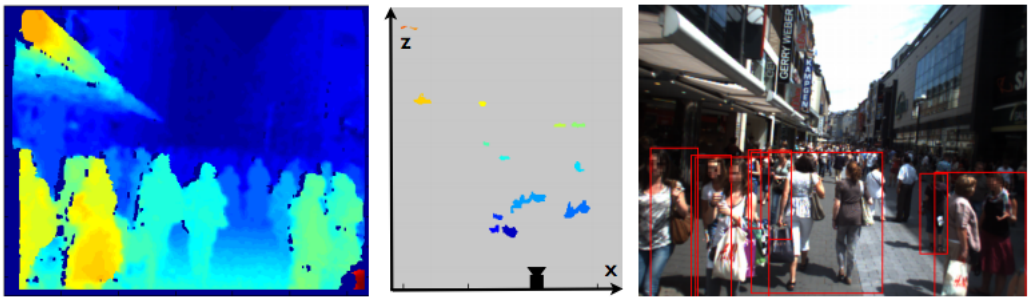


Figure 2.4: ROI extraction [2]. *Left*: input depth image. *Middle*: ROIs projected on a ground plane view. *Right*: ROIs back-projected to the image plane.

With a known camera height, Jafari et al. [18] first project the 3-D points lower than 2m height to a roughly estimated ground plane[1] forming a density map, and then they exclude points located in the bins that have higher density. The points remained are used to compute a more accurate ground plane by a RANSAC-based plane fitting similar to [25]. They then classify those points higher than the ground level into 2 categories i.e. objects (ROIs) and fixed structure (e.g. tall buildings and walls) based on the point density in each bin. The obtained ROIs are smoothed with a Gaussian filter and back-projected to the 2-D image with bounding boxes centred in each ROI (see Fig.2.4 for illustration). In order to avoid rescaling the bounding box for each person, they further segment the ROIs into individuals via the Quick-Shift algorithm [30], which group points around the local maxima in the density map.

Figure 2.5: Example image of ROI extraction [3]. The extracted ROIs are represented by blue.

Bansal et al. [3, 31] divide the ground plane into regular cells and for each cell, a histogram of height distribution is calculated for the 3-D points that fall inside. The height distribution in each cell is grouped into k bands along y axis, where the k needs to be specified. Each pixel with k features is classified into four classes i.e. ground, tall vertical structure, overhang and object candidates according to its likelihood, which is computed from likelihood densities learned by kernel density estimation. The

[1]The first ground plane is roughly estimated based on the camera height of the recording vehicle [18].

classification results are finally smoothed by the Markov Random Field, which penalises adjacent pixels with different labels.

Liu et al. [32, 33] back-project the 3-D point cloud onto the ground plane to generate a height map, where each cell is assigned the highest one of the 3-D points fallen inside. The local height maxima are detected and those with unreasonable distances from the ground are discarded. The ROIs are selected as a set of vertical cylinders with fixed size centred on these filtered local maxima. Instead of using the traditional Cartesian space, Bajracharya et al. [34] employ the polar coordinates as the map on which the depth data are projected. The areas with high density are selected as ROIs, which is based on the assumption that upright objects are more likely to be humans. This method, however, has a drawback that non-human objects (similar size to human) will be incorporated into the resulting ROIs, as addressed in the paper.

In summary, these geometry-based plane removal techniques are able to produce satisfactory ROIs, allowing the search space to be reduced and the false positive rate to be declined. In terms of the usage, they are suitable for both indoor and outdoor environments. Another advantage of these methods is that the backgrounds are not forced to be static, which means they are also applicable to mobile robots. Although the location of the detected ground plane might have slight changes in different frames which is caused by physical oscillations of a moving robot, this problem can be resolved by re-estimating the ground plane for each new frame [25]. The output ROIs are usually crude (see Fig.2.5 for example) in that some non-human objects having the similar 3D information with humans are incorporated in the ROIs. Thus, they need to be analysed by the rest of the pipeline.

2.2.2 Depth Background Modelling for Foreground Segmentation

Foreground segmentation techniques are commonly used for ROI extraction. Compared with the geometric plane removal, foreground segmentation has the advantage of excluding static non-human objects that are impossible to be ignored by the former method. Thus, the ROIs can be further refined. The principle of these methods is that the foreground objects are distinguished from a background model and then are passed to be processed for target detection. Traditional foreground segmentation approaches only use colour images as inputs [35, 36], resulting in suboptimal segmentation. This is because colour data is easily affected by moving shadows and sudden illumination changes. Fortunately, depth sensors have the advantage of being insensitive to these factors, providing more reliable inputs for foreground segmentation algorithms [37].

In most cases, a depth background model is constructed when there are no foreground objects (humans) moving in the scene. Using depth as the feature to build the background model has the advantage that moving objects make a bigger difference in depth images than that in colour images, enabling foreground objects to be easier to be recognised. Galanakis et al. [38] detect foreground pixels based on the depth difference from a background model but the detailed implementations of the foreground segmentation are not provided. Beymer and Konolige [39] first construct a background model by averaging the depth frames from an early stage where it is assumed to contain no people, and mark pixels as foreground if the difference between a pixel and the background model is larger than a threshold. The foreground pixels are further clustered into layers of dominant disparity[2] according to different depths. For each dominant depth layer, a predefined window with size proportional to the corresponding depth is used to detect humans, reducing the unnecessary computation of using different scales for people detection.

Almazan and Jones [40, 41] detect moving foreground pixels by thresholding the difference between the current depth frame and a depth background model. The depth background model is constructed from a certain amount of frames collected within a period of time assuming that there are no foreground

[2]In a stereo system, the obtained depth is called disparity, but it is analogous to depth. In paper [39], depth is measured by a stereo system.

objects in the scene (similar to [39]), and it is updated with a constant learning rate allowing foreground objects like mug to be absorbed by the background model. Similarly, Han et al. [42] classify a pixel as foreground if the absolute difference between the pixel of the current frame and a background model is larger than a manually defined threshold. For the purpose of simplicity, the background model of the current depth image is always defined as the previous depth image. The extracted foreground pixels are then further clustered into individual object by means of neighbouring depth-continuity analysis.

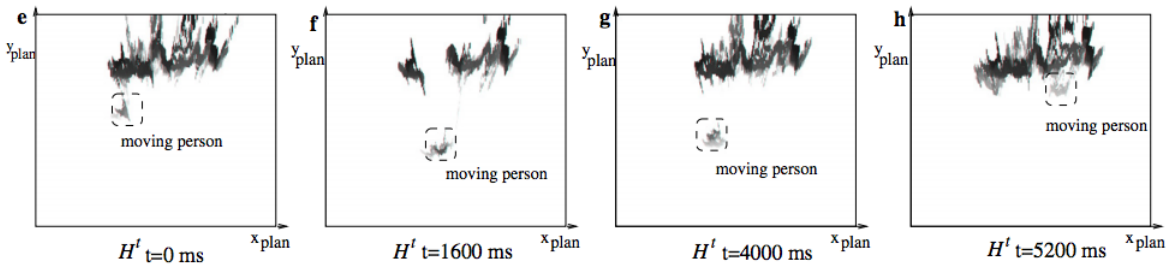


Figure 2.6: From left to right: a moving person is detected and tracked from a heigh-map [4]. The dark areas in the top of the four images represent the obtained background model, and H^t is timestamp.

Another way of using depth for background modelling is to utilise depth to reconstruct a 3-D representation for a scene and store this 3D world as the background in a purposely designed format. Munoz-Salinas et al. in [4] use this idea to reconstruct a 3-D world by transforming the local 3-D coordinates into global 3-D coordinates, and project these 3-D points onto a floor plane forming a plan-view map (called height-map). This map is then discretised into cells and each cell is filled with a height obtained by taking the median from a set of consecutive frames. The background model represented by the height-map is updated periodically in order to allow changes in the environment to be absorbed. In the segmentation stage, all pixels in an image are projected into the plan-view map and foreground points are isolated from the background model. A height threshold is applied to reject foreground points that are lower than a normal human height to refine the detected foregrounds. An example is illustrated in Fig.2.6 where the background model is represented by dark colour and the foregrounds (person) are represented by light colour circled by a dashed square.

In conclusion, these foreground segmentation techniques based on depth are simple to be implemented, and the obtained results can satisfy the demand of reducing the search space for real-time detection tasks. However, the constructed background models have poor generalisation to arbitrary circumstances because a new threshold needs to be experimentally determined when the algorithm is applied in a new environment. The noisy depth data also make the threshold hard to be decided. In addition, most of these methods have poor adaptability to new-coming background objects such as a chair being stationary in the current frame but moving in the past frames. Fortunately, these issues are possible to be alleviated by statistical modelling for each pixel, since the threshold is usually defined as 2.5 standard deviation and adaptive Gaussian distributions are capable of accepting new-coming background objects, which will be discussed in section 2.2.3.

2.2.3 Statistical Background Subtraction

In general, statistical background subtraction is somehow similar to depth background modelling (DBM) (section 2.2.2) in that they both construct a background model to represent the static scene from which the foreground is extracted. However, statistical background modelling differs from the DBM in the way that each pixel in images is now represented as a single Gaussian distribution [35] or a mixture of Gaussians [36] and the classification criterion is determined based on how close an observation (pixel) is to its expected distribution. The general procedure of Gaussian modelling for background subtraction is that, each pixel is modelled using a single Gaussian (or a mixture of

Gaussians) stored as means for each feature (e.g. R, G and B) and a covariance matrix. These parameters are updated each time with a learning rate which can be adaptively tuned in order to absorb new-coming background objects. Finally, a new observation is classified as background if it is close to the expected distribution, or foreground otherwise.

Many background modelling approaches [43, 44, 45] merely use RGB as their inputs. Although the resulting ROIs are adequate in boosting the real-time performance, there are still two common problems that are occasionally encountered: *a*) *shadow problem*: human shadows are incorporated in the ROIs because shadows change the luminance (see Fig.2.7); *b*) *colour camouflage*: human bodies are recognised as background when their clothes have the similar colour to the background. Fortunately, these problems can be solved by adding depth information. For example, shadows detected as intensity changes will make no difference in depth image; a moving object is easier to be distinguished in depth images than in grey images since in depth data it normally produces bigger differences.

(a) Input image (b) Output ROIs

Figure 2.7: Example images showing shadow problem [5].

The earliest works that combine depth and colour images to model a background are proposed in [46, 47] where the depths are obtained by stereo cameras. Gordon et al. in [46] use depth and normalised colour data to construct a mixture of Gaussians, and classify a pixel as foreground as long as either the depth-criterion or the colour-criterion is satisfied. This method is later applied by Salas and Tomasi [48] to segment the foreground objects, before which the points lower than the floor and higher than the ceiling are removed for efficiency. Eveland et al. in [47] use colour and depth data to learn a bimodal distribution composed of a normal distribution and a 'no information' token by taking into account the unreliable depth. The parameters of the normal distribution are not updated if a new observation exceeds a gate value.

Instead of using the traditional RGB, Harville et al. [49, 50] use the illuminance-invariant YUV colour and depth to learn a mixture of Gaussians. The activity level is calculated based on the luminance difference between two consecutive frames to modulate the learning rate, allowing people standing still in conversation to be not incorporated into the background model too quickly. This background modelling technique is also employed by Munoz Salinas in [51] to extract the foreground pixels. These pixels are projected to a plan-view and each group of them is surrounded by a rectangle. The people detection are performed on these rectangular regions whose height and density are within a certain range.

Taking into account the computational efficiency, Bahadori et al. [52] learn a single Gaussian with intensity, depth and edges computed using Sobel operator for an indoor usage. Similar to the work in [49], Bahadori et al. calculate the activity of each pixel based on the difference between the edges in the current frame and the background edge model, supported by the assumption that edge

variations can better reflect the presence of humans even if they are standing still (breathing and pose variations can cause changes in edges). The extracted foreground points are projected to a plan-view and segmented into connected components. These components with size and height inconsistent with people are removed to generate the final ROIs, which is similar to [51]. In the later work [53], Bahadori et al. extend the algorithm [52] by introducing a counter for the purpose of delaying the update of foreground pixels into the background. But this new method is more suitable in the environment where people are expected to be moving frequently.

The choice of using a single Gaussian or a mixture model is dependent on where the method is applied. For example, a single Gaussian is sufficient for an indoor environment where the influencing factors are mainly illumination changes, and a mixture model is competent to be used in more dynamic places (e.g. outdoor) [36]. By using this type of method, non-relevant points can be removed at the most extent, but it has a risk that motionless people are likely to be incorporated into the background model, leading to miss detections by people detectors.

2.2.4 Summary

We have reviewed the depth-based ROI selection methods. They are generally grouped into three categories according to their principles: *geometric plane removal* (GPR), *depth background modelling* (DBM) and *statistical background subtraction* (SBS). In each category, we have analysed the advantages and limitations of the methods.

To sum up, a description of these three types of ROI selection methods based on their characteristics is given in Table 2.1. In terms of usage, these three methods are all competent to be used in an indoor environment, and only the GPR and the SBS are capable of handling complex scenes in outdoor environments. The DBM and SBS both require a relatively static background because they need to learn a background model. Therefore, they cannot be used by mobile robots while the GPR does. Nevertheless, using a background model has the advantage that static non-human objects in the background are possible to be excluded, enabling the search space to be further narrowed down. The DBM requires a threshold, which makes it less general to different scenarios. A dynamic scene means parts of the background are variable. For example, a moving chair enters in the background and subsequently becomes stationary. The SBS is able to handle the dynamic scenes because the learning rate controlling the update speed can be adaptively tuned to allow a new object to be absorbed into the background, but it has a risk that motionless people may be regarded as background too. The GPR is also capable of handling dynamic scenes as long as ground plane is visible. The SBS is sensitive to illumination changes when RGB images are used along with the depth, but this problem does not appear in the other methods. However, in a different point of view, using colour images makes the SBS less dependent on depth, whereas the GPR needs relatively more accurate depth data to reconstruct a 3D world and the DBM relies only on depth to build up a background model.

2.3 Convolutional Neural Networks

This section is structured as follows: section 2.3.1 gives a brief review of what are convolutional neural networks; section 2.3.2 reviews current state-of-the-art object detectors built based on CNNs; section 2.3.3 discusses challenges of using CNNs in RGB-D data and their countermeasures.

2.3.1 Brief Introduction to CNNs

Convolutional Neural Network (CNN) is a type of artificial neural network inspired by biological processes [6]. It can be seen as a variant of multilayer perceptrons (MLP). In computer vision, a traditional MLP connects each hidden neuron with every pixel in the input image trying to find global patterns. Such a connectivity, however, is not efficient because pixels distant to each other

	GPR	DBM	SBS
Indoor usage	✓	✓	✓
Outdoor usage	✓		✓
Suitable for mobile robots	✓		
Real-time requirement	✓	✓	✓
Exclude non-human objects		✓	✓
Adaptive to dynamic scenes	✓		✓
Invariant to illumination changes	✓	✓	
Need a background model		✓	✓
Need a threshold		✓	
Highly dependent on depth	✓	✓	

Table 2.1: Description of the three types of ROI selection methods based on their characteristics.

are often less correlated. The found patterns are thus less discriminative to be fed to a classifier. In addition, due to this dense connectivity, the size of parameters grows largely as the size of an input image increases, resulting in substantial increases in both computational complexity and memory space usage. However, these problems can be alleviated in CNNs. A hidden neuron in CNNs only connects to a local patch in the input image. This type of sparse connectivity is more effective to discover local patterns and these local patterns learned from one part of an image are also applicable to other parts of the image.

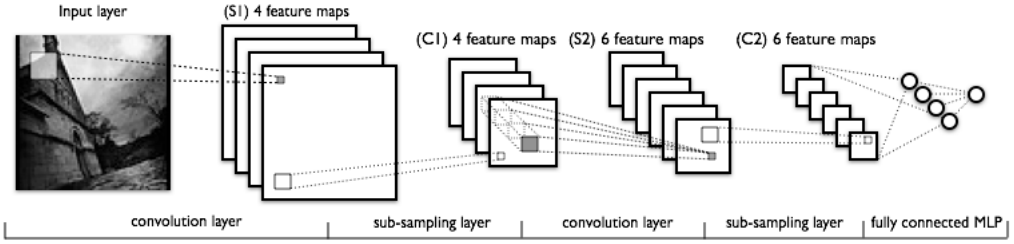


Figure 2.8: A typical architecture of CNN [6].

In general, the architecture of CNN can be decomposed into two stages, which are hierarchical feature extraction stage and classification stage. A typical architecture of CNN is depicted in Fig.2.8. An input image is convolved by a set of trainable filters (kernels) each with a nonlinear mapping (e.g. ReLU [54]) to produce so-called *feature maps*. Each feature map containing special features is then partitioned into equal-sized, non-overlapping regions and the maximum (or average) of each region is passed to the next layer (*sub-sampling* layer), resulting in resolution-reduced feature maps with depth unchanged. This operation allows small translation to the input image, thus more robust features that are invariant to translations are more likely to be found [55]. These two steps, *convolutions* and *subsampling*, are alternated for two iterations in the CNN in Fig.2.8 and the resulting feature maps are fully connected with a MLP to perform classification. In some applications, the final fully connected (FC) layer that performs classification is replaced with other classifiers e.g. SVM. For example, the state-of-the-art object detector R-CNN [56] extracts high-level features from the penult FC layer and feeds them to SVMs for classification, which will be discussed later in section 2.3.2.

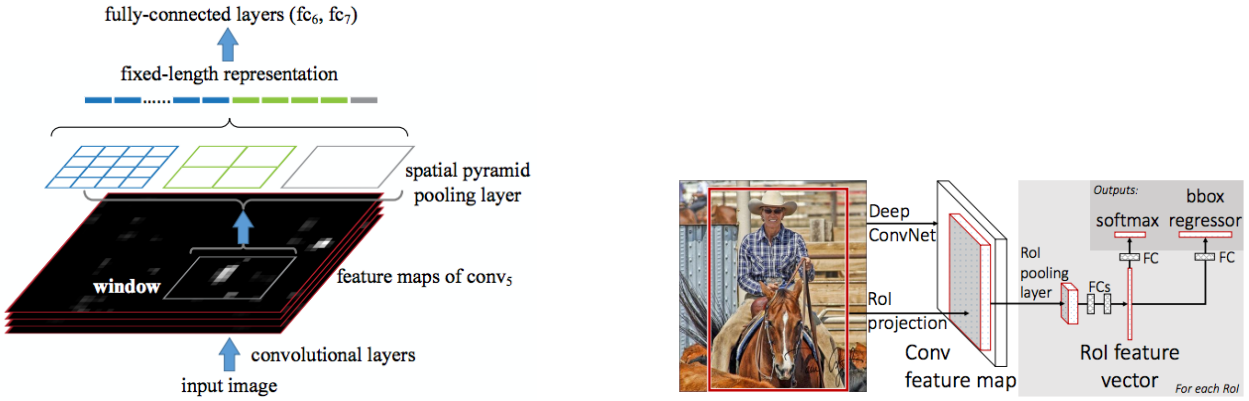
At the training stage, *stochastic gradient descent* algorithm is employed in a supervised way to update all coefficients simultaneously including convolutional filters and the weights in fully connected layers. To prevent overfitting, the *dropout* operation [57] is usually applied to FC layers. For each iteration in the training, each unit in the feature vectors of the FC layer is set to 0 with a probability

p (normally choose $p = 0.5$) and the selected units are put back in place at the end of the iteration. When a unit is dropped out, it will not contribute to the forward pass and back-propagation. Such an operation reduces complex co-adaptation of feature vectors during the training period, forcing more robust features to be learned [58]. At the inference time, the dropout is removed but the weights of those units that were dropped out during the training will be multiplied by p to ensure that the expected output of those units will remain the same.

2.3.2 State-of-the-Art CNN-Based Detectors

The success of deep learning, especially for CNNs, has pushed visual object detection forward to a new level. The detection accuracy, or more precisely mAP (mean Average Precision), achieved by CNN-based detectors, significantly outperforms other non-deep learning methods by a considerable margin [56] in some well-known datasets such as PASCAL VOC[3] and ImageNet. The framework used by most CNN-based detectors mainly consists of two components: a region proposal method and CNNs. Region proposal methods generate class-independent candidate proposals (windows) by detecting interest points in an image, which differs from the traditional dense sliding-window that exhaustively scans over image pixels at multiple scales. Here we do not discuss in detail the proposal methods because we mainly focus on the CNNs part[4]. Readers interested in this field are referred to [59], which provides a comprehensive evaluation on recent proposal methods.

Region-based Convolutional Neural Network (R-CNN) [56], as one of the ancestors of recent state-of-the-art CNN-based detectors, pointed out a new direction of designing efficient region-based detection framework using CNNs. In terms of the detection pipeline, R-CNN generates a set of category-independent proposals using selective search [60] in a given image and applies the CNN to extract fixed-length CNN features for each proposal directly from raw image pixels. The CNN features are then classified by category-specific linear SVMs. Although achieving high detection accuracy, R-CNN is too slow for real-time applications as the CNN is repeatedly applied to every candidate window in the input image.



(a) SPP-Net [61]

(b) Fast R-CNN [10]

Figure 2.9: Detection pipelines of SPP-Net and Fast R-CNN.

Based on the R-CNN detection framework, spatial pyramid pooling network (SPP-Net) [61] was developed. Similar to R-CNN, SPP-Net also uses selective search to produce candidate proposals and category-specific linear SVMs for classification. Unlike R-CNN that takes as input a cropped image patch from an input image, SPP-Net uses CNN to perform convolutions on the entire input image,

[3]The PASCAL Visual Object Classes classification: http://host.robots.ox.ac.uk/pascal/VOC/

[4]We aim to develop an ROI selection method to reduce areas scanned by the sliding-window, rather than to devise such a region proposal method.

generating convolutional feature maps at the last convolutional layer. Each candidate window in the image domain is projected to the aforementioned feature maps. A fixed-length feature vector for that window in the feature maps is obtained by performing max-pooling with a pyramid fashion (see Fig.2.9a). In particular, the original max-pooling layer positioned after the last convolutional layer is replaced with a spatial pyramid pooling (SPP) layer, which is depicted as the three small windows with different colours in the top of Fig.2.9a. The SPP layer pools pyramid features (3-level is used in Fig.2.9a) from a specified window (ROI window) in the convolutional feature maps and concatenates these features into a single feature vector which is passed to the following FC layers. The spatial bin numbers are fixed so that the pooled features have a fixed length while the sizes of bins are dependent on the ROI window. In this way, computations spent on convolutions are shared between proposals, which significantly reduces the runtime.

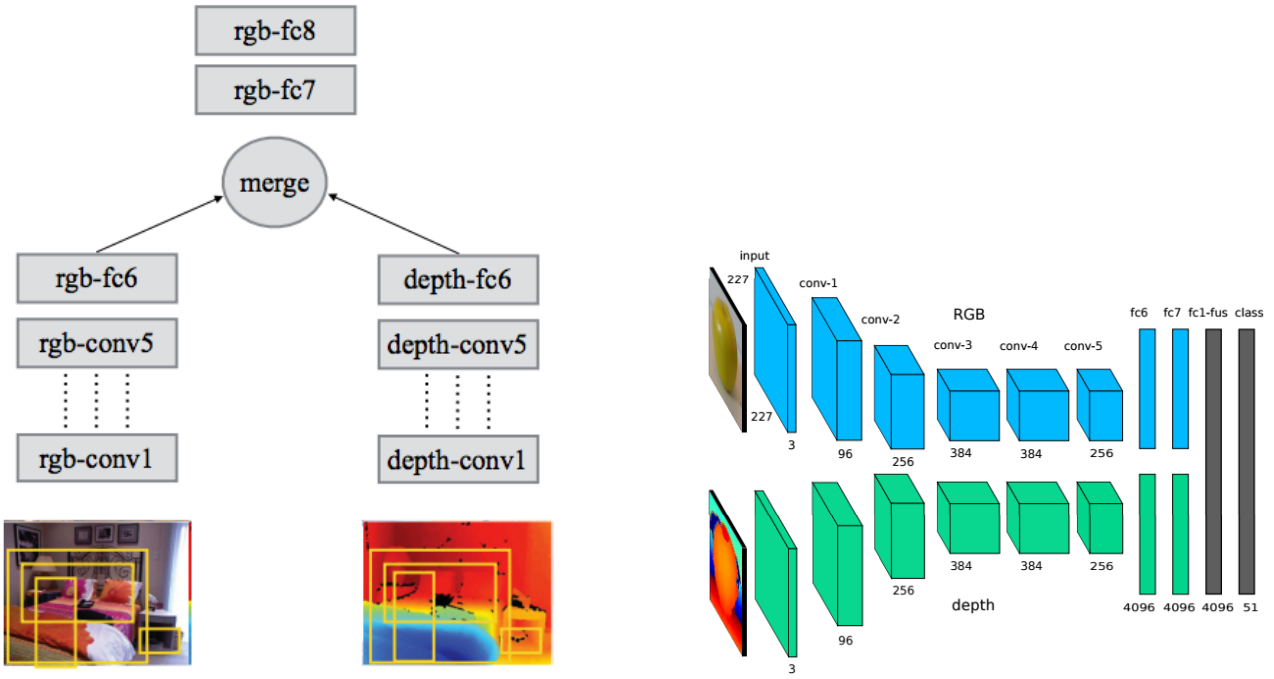
R-CNN and SPP-Net require separate training on their components (e.g. CNN and SVM), which is inefficient. Moreover, the convolutional layers prior to the SPP layer in SPP-Net cannot be fine-tuned, which limits the adaptation to new dataset when using deeper CNNs. Fast R-CNN [10], which adopts an efficient end-to-end training, was then proposed. The architecture of Fast R-CNN is shown in Fig.2.9b. It can be seen that the architecture of Fast R-CNN is similar to that of SPP-Net where the SPP layer is replaced with the ROI-pooling layer and linear SVMs are substituted by the softmax. Fast R-CNN also performs convolutions on the entire input image and produces convolutional feature maps at the last convolutional layer. The ROI-pooling layer, which is analogous to SPP layer, pools a fixed-length feature vector from each ROI window (max-pooling) and passes this feature vector to the subsequent classification pipeline. From another viewpoint, the ROI-pooling layer can be regarded as a special single-level SPP layer. The last classification layer i.e. softmax is kept as the author in [10] reported that using softmax not only reduced complexity in training, but also obtained slightly better accuracy. Moreover, the two FC layers in the architecture are restructured via *Singular Value Decomposition*, further accelerating the processing speed. This background knowledge is provided in Appendix B where we explain why and how the FC layers are restructured to accelerate the processing speed. We also use this technique in our CNNs to make computations faster.

2.3.3 Applications with CNNs in RGB-D Data

CNNs have been shown to have extraordinary achievements in standard RGB vision tasks such as image classification [9] and object detection [10]. With the RGB-D sensors being widely used, CNNs have also been applied in RGB-D images for visual recognition/detection. When using CNNs in RGB-D images, there are generally three challenges that might be encountered. First, there does not exist a large enough RGB-D dataset (currently) for training CNNs from scratch (with random initialisation). Fortunately, this problem can be eased by fine-tuning the weights pre-trained in a large RGB dataset like ImageNet. The transferability of CNN layers has been studied by Yosinski et al. in [62] where they showed that initialising a network with transferred weights can improve the generalisation performance of CNNs. Second, when one wants to fine-tune a CNN pre-trained in RGB data to adapt to depth data, the input data structure would be required to be consistent with the pre-trained data. Specifically, the depth data passed to the pre-trained CNN must have three channels. It is also expected that the transformed depth images are close to RGB images in terms of information contained e.g. edges and contrast. This will make the pre-trained model adapt faster to the transformed depth images because features pre-learnt for RGB images are reused[5]. Third, it is essential to design a model that combines RGB-CNN and Depth-CNN effectively.

Gupta et al. in [15] modified the R-CNN framework to have two streams of CNNs, one for RGB and one for depth. The CNN features extracted from RGB and depth images were concatenated

[5]Imagine that you taught a child to recognise coupe as vehicle because coupe has wheels, you do not need to re-teach the child to recognise truck as vehicle because truck and coupe both have wheels (they share common features).



(a) RGB-D detection pipeline in [16]

(b) Two-stream CNNs for RGB-D object recognition [7]

Figure 2.10: Example architectures of CNNs applied in RGB-D data.

and then passed to linear SVMs. To make depth images have similar structures with RGB images, they proposed to encode depth images in a pixel-wise way to produce three channels, which are horizontal disparity, height above ground and angle between surface normal and gravity direction (this was called HHA encoding in [15]). To overcome the scarcity of depth images, they artificially augmented the depth dataset by adding 3D synthetic CAD object models available from the Internet to the scenes of the training data. Hoffman et al. in [16] employed the framework of Fast R-CNN and replicated the CNN model by two, one for RGB and one for depth (see Fig.2.10a). The mid-level features from the mid-FC layers were merged and then processed by the following FC layers. They also used the HHA encoding to preprocess depth images. During training, the RGB-CNN (already pre-trained on ImageNet) was trained on a combination of an extra large RGB dataset and a small target RGB dataset, and the depth-CNN was initialised with the weights from the trained RGB-CNN and then trained on the limited target dataset. Eitel et al. in [7] designed a new encoding technique to deal with depth images. They first normalised the depth images such that pixel values were ranged between 0 and 255, which resembles the scale of RGB data. Then each pixel in the normalised images was mapped to three channels using reversed jet colormap (red means near and blue means far). The CNN architecture used by them is shown in Fig.2.10b, which is similar to the one used by Hoffman et al. [16]. For training the depth CNN, they augmented depth images by adding synthetic noise patterns (depth with no values) to enhance the robustness of classification in noisy real-world scenarios. Furthermore, they provided a qualitative comparison of different depth-encoding methods using their mixture-CNNs, which is shown in Fig.2.11. Their colorisation method (Depth-jet encoding) shows comparable accuracy with HHA encoding, while the latter method requires more complex computations. In our project, we will also compare different depth-encoding methods. It is interesting to see if our results agree with the one shown in Fig.2.11.

Depth Encoding	Accuracy
Depth-gray (single channel), from scratch	80.1 ± 2.6
Depth-gray	82.0 ± 2.8
Surface normals	$\mathbf{84.7 \pm 2.3}$
HHA	83.0 ± 2.7
Depth-jet encoding	83.8 ± 2.7

Figure 2.11: A comparison of different depth encoding methods on Washington RGB-D object recognition task from [7].

2.4 Conclusions

We have conducted the literature review in the domains of Microsoft Kinect, use of depth for ROI selection and applications with CNNs in object recognition/detection. We have analysed the advantages and deficiencies of reviewed methods. We summarise this chapter as follows.

In section 2.1, we briefly review the Kinect sensor, mainly for the sensor mechanism and data format. This is an essential step before carrying on with the subsequent reviews because the RGB-D images obtained by the Kinect sensor are different from normal RGB images. The key points that are necessary for us to notice are: *i*) the Kinect sensor is more reliable to be used in an indoor environment; *ii*) colour images and depth images are required to be spatially aligned in advance; *iii*) depth data are very noisy and some values may be missing, thus depth-recovering techniques such as fast max-filter [20] are useful if an entire depth image will be used.

In section 2.2, we review ROI selection methods based on depth. We generally divide these methods into three categories, which are *geometric plane removal* (GPR), *depth background modelling* (DBM) and *statistical background subtraction* (SBS), according to their principles. All of them can be used in an indoor environment and can suffice a real-time requirement. Since the depth data are usually noisy, the SBS is better than the other two that are more dependent on depth data. The SBS and DBM can exclude non-human pixels maximally, which seems to be more ideal than the GPR. However, the SBS and DBM are hazardous to be used because motionless people are likely to be incorporated into background models, which can lead to missing detections. Particularly in an indoor environment, people are more likely to stay in a motionless status as they may be sitting on sofas. Thus, it is safer to use GPR as an ROI selection method in our project.

In section 2.3, we give a brief introduction of CNNs and review current state-of-the-art object detectors based on CNNs. Fast R-CNN provides an excellent framework for people detection. The ROI-pooling layer proposed in Fast R-CNN promotes shared computations on convolutions. Also, it is less complex compared to the SPP layer in SPP-Net. We finally analyse the challenges of using CNNs in RGB-D images and discuss their counterparts by reviewing existing methods. HHA encoding [15] captures geocentric properties in depth data, but is less practical for real-time use because it requires more computations. In contrast, colorisation method [7] only requires simple manipulations in depth data while achieving comparable accuracy, which is a better encoding technique to be used for our project.

Chapter 3

People Detection in RGBD Data

In this chapter, we provide details on the design of our RGB-D people detection system. In section 3.1, we give an overview of the detection pipeline. The subsequent sections, which are 3.2, 3.3 and 3.4, expand on the specific modules proposed in the pipeline.

3.1 Detection Pipeline

Figure 3.1: People detection pipeline.

The people detection pipeline designed for our project is depicted in Fig.3.1. It consists of five modules, where the **preprocessing**, **ROI selection** and **people detection** are the cores in this pipeline. An overview of this detection pipeline is sketched as follows.

An RGB image and a raw depth image captured by the Kinect sensor constitute the inputs to this pipeline. The preprocessing module is mainly used to process the raw depth image as discussed in section 2.1 that it cannot be directly used. In particular, the spatial alignment and a hole-filling technique are carried out to process the raw depth, followed by a depth-encoding method that transforms the depth images from one-channel to three-channel. These are detailed in section 3.2. The ROI selection module takes as input the preprocessed depth image. The ground plane in the image is detected with the help of the depth and pixels belonging to this plane are removed (ground plane detection). A scan window is slid over the valid pixels (ROI) forming candidate proposals and the window size for each pixel is heuristically determined by the depth value (scale-informed search). In addition, the proposals that mainly contain invalid depth values are filtered out (candidate proposals filtering). The ROI selection module is demonstrated in section 3.3. The candidate proposals, along with the RGB and preprocessed depth images, are forwarded to the people detection module, which plays the most important role in this people detection system. The people detection module consists of two streams of CNN, one for RGB and one for depth. The two networks are identical in architecture. To promote sharing of computations on convolutions, we employ the ROI-pooling proposed in Fast R-CNN [10] to enable the networks to perform convolutions on the entire images. To fuse the information gathered from colour and depth images, we propose a novel way that combines the output probabilities of the two networks for the same candidate window infer a stronger probability, deciding whether the window contains a person or not. As a result, every candidate proposal is assigned a probability. Proposals with high probabilities are selected and then passed to the Non-Maximum Suppression, thus reducing highly overlapping windows. These are described in section 3.4.

3.2 Depth Image Preprocessing

As discussed in the review of the Kinect sensor in section 2.1, raw depth images cannot be directly used because they are not spatially aligned with the RGB images. Also, there are many pixels containing invalid values in depth images of testing data, so they need to be filled before being used by the CNN in our people detector. The main reason for why images with holes are not suitable is because the

interface between holes and data would form artifacts (e.g. false edges) that have no physical meaning and that would confuse the classifier. Theoretically, it is plausible to enable CNNs to perform robust classification in depth images with holes present, but this requires more efforts and will become our future work.

3.2.1 Spatial Alignment

We assume the calibration has been conducted in the Kinect sensor to obtain the camera parameters that are necessary for aligning depth images. The camera parameters are explained in Table 3.1. The diagram showing how to find the location of a depth pixel in the RGB image is drawn in Fig.3.2.

We initialise an empty image $\hat{D}$ (the **new depth image** in Fig.3.2) with a size identical to the raw depth image. For each pixel (x_d, y_d) in the raw depth image D, the goal is to find its corresponding location (x, y) in the RGB image. The aligning steps are visualised in Fig.3.2.

First, we project a depth pixel (x_d, y_d) from the raw depth image to the depth camera 3D space (P_x, P_y, P_z) by using the depth camera intrinsics:

$$P_x = \frac{(x_d - c_{x\_d}) \times D(x_d, y_d)}{f_{x\_d}} \tag{3.1}$$

$$P_y = \frac{(y_d - c_{y\_d}) \times D(x_d, y_d)}{f_{y\_d}} \tag{3.2}$$

$$P_z = D(x_d, y_d) \tag{3.3}$$

where $D(x_d, y_d)$ represents the depth value at position (x_d, y_d) in the raw depth image.

Second, we use the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ to transform the pixel (P_x, P_y, P_z) to the RGB camera 3D system (Q_x, Q_y, Q_z):

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \end{bmatrix} = \mathbf{R} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \mathbf{t} \tag{3.4}$$

Description	Notation
Focal length for RGB and depth cameras	$f_{x\_rgb}, f_{y\_rgb}, f_{x\_d}, f_{y\_d}$
Optical center for RGB and depth cameras	$c_{x\_rgb}, c_{y\_rgb}, c_{x\_d}, c_{y\_d}$
Rotation and translation between the two cameras	$\mathbf{R}, \mathbf{t}$

Table 3.1: Kinect sensor parameters.

Third, with the RGB camera intrinsics, we map the pixel (Q_x, Q_y, Q_z) to the RGB image plane to obtain the position coordinates (x, y):

$$x = \frac{Q_x - f_{x\_rgb}}{Q_z} + c_{x\_rgb} \tag{3.5}$$

$$y = \frac{Q_y - f_{y\_rgb}}{Q_z} + c_{y\_rgb} \tag{3.6}$$

Finally, we copy the depth value $D(x_d, y_d)$ to the new position in the new depth image $\hat{D}$:

$$\hat{D}(x, y) = D(x_d, y_d) \tag{3.7}$$

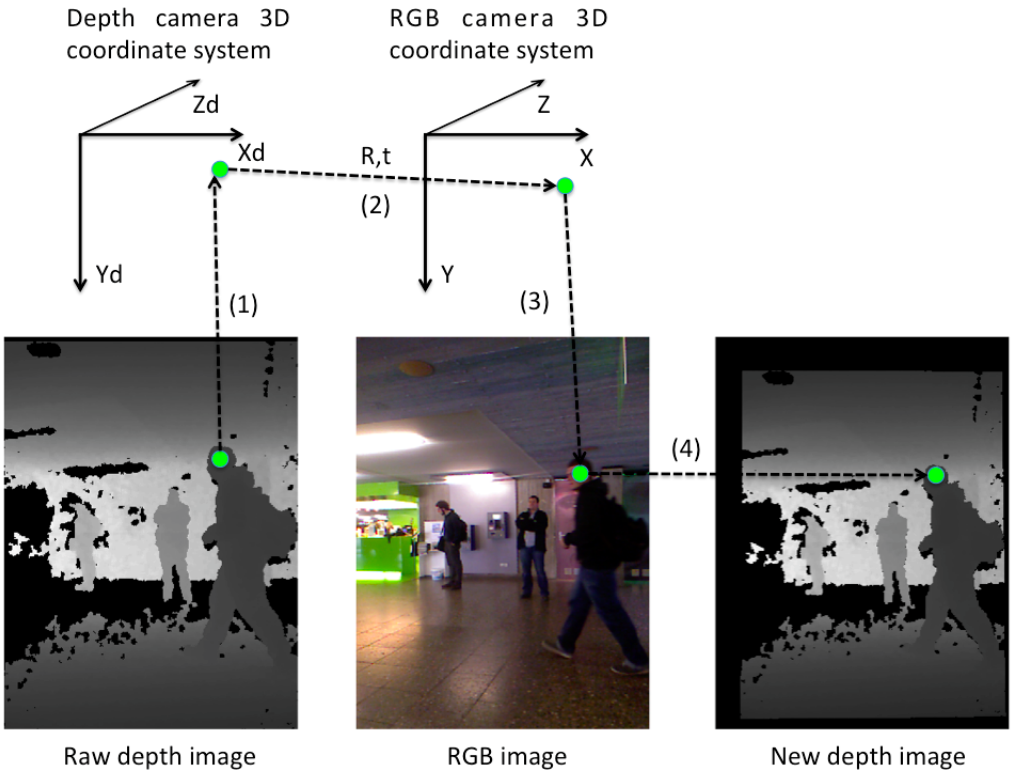


Figure 3.2: Spatial alignment between depth and RGB. Aligning steps are sorted in order. (1) project a depth pixel into metric 3D space via depth camera intrinsics; (2) map this pixel to the RGB camera 3D space via rotation $\mathbf{R}$ and translation $\mathbf{t}$; (3) compute the location of this pixel in the RGB image plane via RGB camera intrinsics; (4) reconstruct the depth image by copying this pixel's depth value to the computed location.

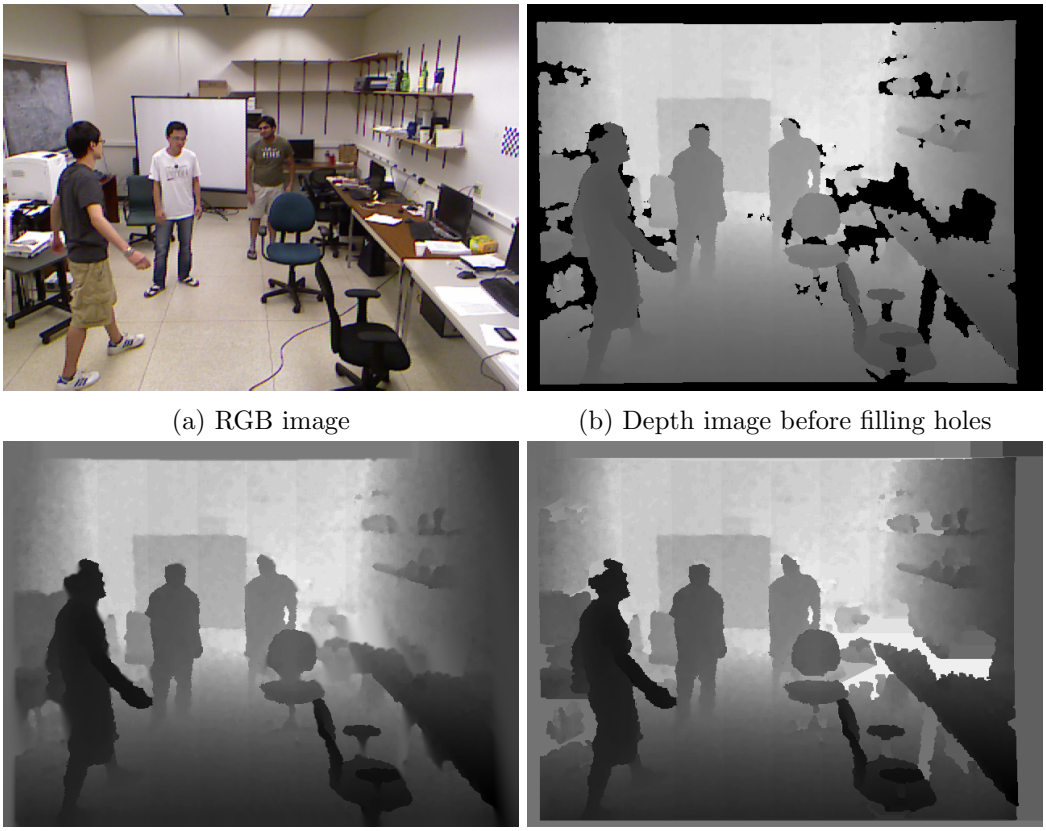
3.2.2 Hole-Filling

Taking into account the computational efficiency, we propose to use the mean-filter to fill the holes in depth images. The holes in depth images are stored as zeros. The mean-filter iteratively fills holes in a depth image with the mean of a pixel's non-zero neighbours (e.g. 5×5 window), which is fast and effective. In this way, the shape information in depth images can be well preserved. The example images with and without hole-filling are shown in Fig.3.3.

In fact, we also tried using the max-filter, which iteratively fills holes with the maximum among a pixel's neighbours, but the detection performance was worse than that of using the mean-filter. In practice, using the max-filter is more likely to distort the image shape than using the mean-filter. The depth image shown in Fig.3.3d was recovered with the max-filter. It can be seen from the image that there are some losses of shape information on, for example, the left desk next to the person and the right desk surface. However, these regions are better recovered in Fig.3.3c in which the mean-filter was applied.

3.2.3 Depth-Encoding

The goal of encoding depth images is to make them compatible with pre-trained CNN models. In the existing literature, there are some ways that have been proven to be effective for encoding depth such as HHA encoding [15] and colorisation method [7]. In our project, we mainly focus on two metrics when selecting appropriate depth-encoding methods. First, the encoding methods are expected to preserve as much information contained in the original depth as possible such as shape. Second, the



(a) RGB image

(b) Depth image before filling holes

(c) Depth image after filling holes with the mean-filter

(d) Depth image after filling holes with the max-filter

Figure 3.3: Example images of with and without hole-filling. Image resource from [63].

encoding methods are preferred to be computationally cheap.

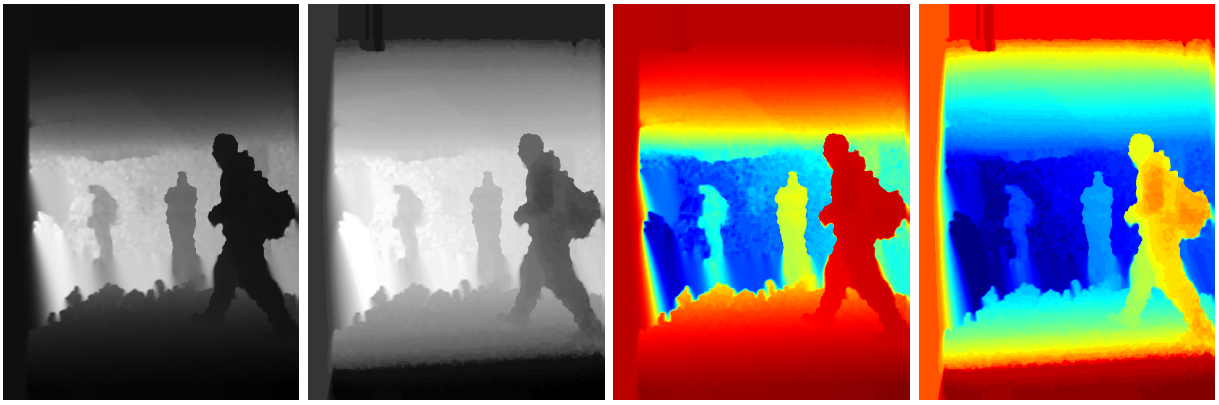
Based on these metrics, we select two from the existing methods, depth-gray encoding and colorisation method, which are used by Eitel et al. in [7]. In addition to these two methods, we propose to use the other two encoding methods, which are the variants of the two methods. The one is called contrast-enhanced depth-gray which was used by Ben et al. in [20] and the other one is called contrast-enhanced colour-depth. Even though the depth-gray encoding has been proven by Eitel et al. to be inferior to the colorisation method on RGB-D object recognition (see Fig.2.11 in section 2.3.3), we still intend to use it as a reference to our contrast-enhanced depth-gray. The detailed implementation of these four methods are described as follows[1] and the example images are shown in Fig.3.4 (the corresponding RGB image is in Fig.3.2).

- **Depth-gray**: The depth pixels are normalised to have values ranging from 0 to 255. In particular, for each depth pixel x, the normalised value x' is calculated by $x' = \frac{x - x_{min}}{x_{max} - x_{min}} \times 255$, where x_{min} and x_{max} correspond to the minimum and maximum in a depth image, respectively. Then the normalised 2D depth map is replicated to three channels.

- **Contrast-enhanced depth-gray**: A depth image is first normalised using the aforementioned way and then histogram equalisation is applied to the normalised image so as to enhance the contrast. Finally, the obtained 2D image is replicated to three channels.

- **Colour-depth**: In a normalised depth image, each pixel is mapped to three channels via a

[1]For consistency, we rename the colorisation method as *colour-depth*.



(a) depth-gray (b) contrast-enhanced depth-gray (c) colour-depth (d) contrast-enhanced colour-depth

Figure 3.4: Four depth-encoding methods.

reversed jet colormap, where the resulting colour values are ranged from red (near) over green to blue (far) [7].

- **Contrast-enhanced colour-depth**: The first two steps are identical to those used in **Contrast-enhanced depth-gray**, producing a contrast-enhanced 2D image. This single-channel image is then mapped to a three-channel image via the reversed jet colormap.

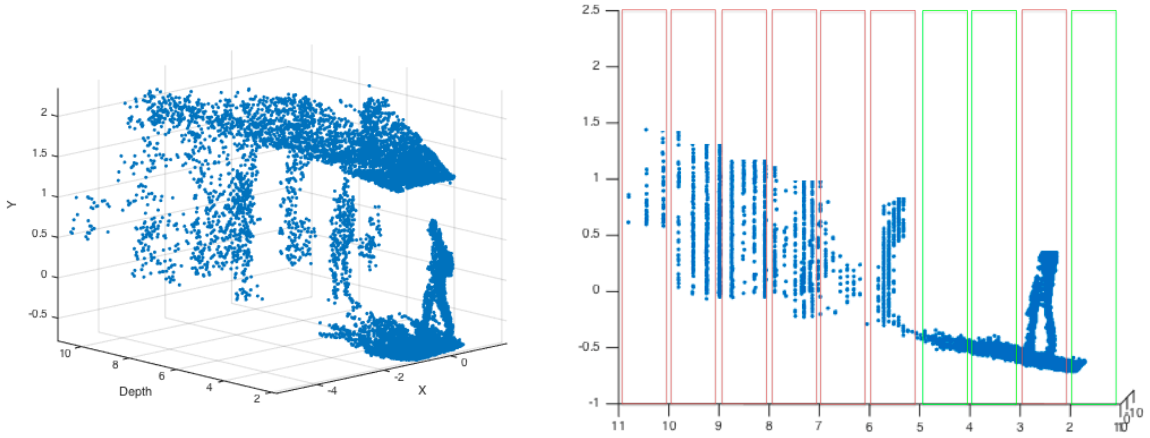
3.3 ROI selection

In this module, we aim to utilise depth in RGB-D images to reduce search space for our people detector. In section 2.2 we have identified the pros and cons of the three types of ROI selection methods and finally concluded that Ground Plane Removal (GPR) methods are safer than the other two to use. For example, using GPR is not likely to exclude pixels belonging up human, especially for the upper body. In fact, we implemented all the three ROI selection methods and we found that background subtraction methods were too hazardous to use in practice that people staying motionless for a long time were completely absorbed into the background. Because we do not want to sacrifice recall to gain speed, we choose to use the idea of GPR to achieve ROI selection. In addition to removing the ground plane, we take advantage of depth to heuristically compute the window size for each pixel scanned by the sliding window, which reduces the time wasted on multiple scale detection. In the end, we discard those windows that mainly contain invalid depth pixels, further reducing the number of windows. We provide detailed description for the three sub-modules in section 3.3.1, 3.3.2 and 3.3.3, respectively.

3.3.1 Ground Plane Detection

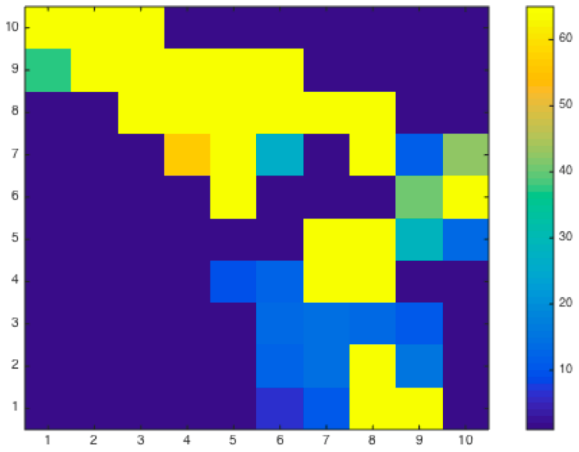
We propose a robust ground plane detection algorithm applied in depth data. With a set of 3D datapoints, we first remove as many outliers[2] as possible via an empirical criterion. The majority of the remaining datapoints, which are more likely to belong to the real ground plane, are then passed to a RANSAC-based plane detection algorithm, finally estimating the plane coefficients. With these plane coefficients, pixels close to the plane are set to non-ROIs so that they are not scanned by the sliding window. The output of this module is a binary mask, where 1 means ROI and 0 means non-ROI.

[2]Here outliers are referred to those 3D points that do not belong to the ground plane.



(a) 3D scene reconstructed from depth. The original 3D points are denser, but we downsampled them for the purpose of clear look.

(b) Sideview of the reduced 3D scene where green rectangular grids are retained and red rectangular grids are discarded.



(c) Projection of the reduced 3D points on X-Depth plane. Each grid contains a set of points fallen inside. Grids with brighter colour contain points that are more spread out in vertical direction.

Figure 3.5: Ground plane detection steps.

Reducing the outliers

The depth pixels (in meters) are transformed to a 3D space using the known depth camera intrinsics. Fig.3.5a shows the 3D scene of Fig.3.2. With the observation that ground plane is usually located in the lower part of an image, in Fig.3.5a we select the points that are located in the lower part of the image, resulting in a reduced 3D scene. The sideview of the reduced 3D scene is shown in Fig.3.5b. When there are too many outliers in data, the performance of RANSAC algorithm will be significantly degraded. For example, the estimated plane is likely to pass through a person (a person in the 3D scene is nearly a surface). In fact, without reducing the outliers, we frequently observed incorrect planes estimated by the RANSAC in experiments. Thus, managing to remove as many outliers as possible is necessary for an accurate plane estimation.

We remove the outliers based on the observation that clusterings of points with large vertical distribution usually correspond to non-ground plane objects. For instance, the red rectangular grids in Fig.3.5b obviously contain many outliers i.e. non-plane points. To this end, we project the reduced

3D points onto the X-Depth plane forming an occupancy map and discretise the map into 10 by 10 grids. For each grid, we compute the vertical standard deviation (VSTD) of the points fallen inside. The grids of points with VSTD larger than an experimentally determined threshold are removed and the rest of the points are fed to the RANSAC-based plane detection algorithm. Our outliers reduction method is similar to the one used by Jafari et al. in [18] (section 2.2.1), but they removes grids of points with high density, which is not robust because grids with high density may contain points that all belong to the ground plane when the plane points are compact enough. The visualisation of the aforementioned occupancy map is displayed in Fig.3.5c where grids with brighter colour have larger VSTDs.

Estimating the plane

With the refined 3D points, the basic RANSAC is capable of producing an accurate estimate of the ground plane. In general, there are four steps in the loop of our RANSAC-based plane detection algorithm: (1) Randomly select three non-collinear points; (2) Estimate the plane coefficients based on these points; (3) Measure the quality of estimated plane; (4) Compare the estimated plane with the previous best one and keep the winner.

The key in using the RANSAC to estimate a more accurate plane is '*how to measure the quality of estimated plane*' such that a really better estimate will be kept. Yang and Förstner in [64] defines that a good plane should be the one that fits more datapoints or that the fitted datapoints have small deviations relative to the plane. To achieve a more robust estimation, we define in the fourth step that a better plane should satisfy the two conditions at the same time. The pseudocode of our RANSAC-based plane detection algorithm is detailed in Algorithm 1. The detailed computation of the plane coefficients is provided in Appendix A.

3.3.2 Scale-Informed Search

The traditional sliding-window approach generates windows of multiple scales at a pixel, which exponentially increases the time for people detection. Fortunately, with the help of depth, the scale of window for a pixel can be effectively approximated by the simple triangulation. When sliding a window on pixels of ROIs, the window width win is determined by

$$win = \frac{fW}{Z} \tag{3.8}$$

where f is the focal length of depth sensor, W is the rough width of a normal human in real life, and Z is the depth value. We only detect the upper body of humans, so each window is a square bounding box.

3.3.3 Candidate Proposals Filtering

Depth images contain holes, which is caused by multiple reflections, transparent objects or materials with poor reflective properties (discussed in section 2.1.3). Nonetheless, the face and upper body of a human is usually visible in depth images, excluding some special cases where, for example, strong lights are shining on the human body.

Based on this observation, we propose to refine candidate proposals by filtering out those mainly containing invalid depth pixels (or zero pixels). To achieve this goal, we need to calculate how many valid depth pixels are present in a proposal. We employ the idea of integral image, which was proposed by Viola and Jones in [65] for the Haar-like features computation, to efficiently calculate the number of valid depth pixels in a specified window. For the detail of integral image, the reader is referred to [65]. After obtaining the number, we divide it by the square of window width to get the ratio of valid depth pixels, π. A proposal is only accepted if its π is larger than a threshold e.g. 0.3, otherwise is

Algorithm 1 RANSAC-Based Plane Detection

Input: X, θ_{t-1} {X is a list of points, θ_{t-1} is previous plane coefficients}
Output: θ_t

if θ_{t-1} == NULL **then**
  {this is the first plane}
  $p = 0.99, w = 0.5, m = 3$; {parameters for determining maximum number of iterations}
  $I_{max} = \frac{\log(1-p)}{\log(1-w^m)}$; {result equal to 35}
  bestSTD = ∞, bestSupport = 0;
else
  $I_{max} = 8$;
  $\theta' = \theta_{t-1}$; {current best plane}
  dis = distance2plane(X, θ'); {compute distance between points and plane}
  s = find$(dis < \tau)$; {τ is a threshold}
  bestSTD = calcSTD(s); {calculate standard deviation of shortlisted distances}
  bestSupport = length(s);
end if
$i = 1$;
while $i < I_{max}$ **do**
  $3pts$ = select3pts(X);
  θ = pts2plane$(3pts)$;
  dis = distance2plane(X, θ);
  s = find$(dis < \tau)$;
  STD = calcSTD(s);
  Support = length(s);
  if (Support >= bestSupport) $\wedge$ (STD <= bestSTD) **then**
    $\theta' = \theta$;
    bestSTD = STD;
    bestSupport = Support;
  end if
  $i = i + 1$;
end while
$\theta_t = \theta'$;

discarded. An example image showing valid and invalid proposals is displayed in Fig.3.6. The output of this module is a set of candidate proposals, defined by an $N \times 4$ matrix $\mathbf{B}$ containing the geometric positions of proposals in the image domain:

$$\mathbf{B} = \begin{bmatrix} x_1^1 & y_1^1 & x_2^1 & y_2^1 \\ x_1^2 & y_1^2 & x_2^2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & y_1^N & x_2^N & y_2^N \end{bmatrix}$$

where N is the number of proposals, (x_1^i, y_1^i) is the top-left corner of a proposal, (x_2^i, y_2^i) is the bottom-right corner of the proposal and i is the index of the proposal.

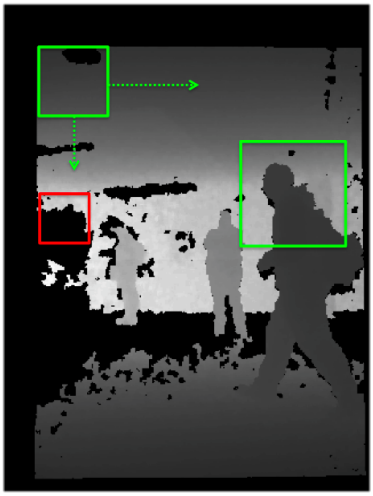


Figure 3.6: Green proposal is remained and red proposal is discarded. Holes correspond to black regions.

3.4 People Detection with CNNs

The main goal of this module is to perform people detection in RGB-D images. The input to this module includes the RGB and preprocessed depth images, together with a set of candidate proposals (windows) generated by the ROI selection module. Our people detector scores each window with a probability, or confidence level, indicating how likely a window contains a person (upper body). The windows with probabilities lower than a threshold (e.g. 0.5) are discarded. The remaining windows are fed to the Non-Maximum Suppression for post-processing. As a result, only one window for each instance is kept.

3.4.1 Proposed Model

Fig.3.7 illustrates the architecture of our people detector. There are generally two streams of CNNs in the detector. The blue stream network (RGB-CNN) processes RGB images while the green stream network (Depth-CNN) processes depth images. We use the CaffeNet[3] as the baseline CNN. It consists of five convolutional layers and three fully connected layers (the last one is a softmax classification layer). Max-pooling layers are inserted after the first and second convolutional layers. We replace the

[3]CaffeNet is essentially a variant of the AlexNet [9]. The main change made to the AlexNet is that the order of pooling and normalisation layers is switched. This model also comes with pre-trained weights in the Caffe implementation [66].

max-pooling layer after the fifth convolutional layer with the ROI-pooling layer. ReLU [54] is used in all layers except for the last classification layer.

The ROI-pooling layer is proposed by Girshick in Fast R-CNN [10]. Given an ROI window of size $h \times w$ in the conv5 feature maps, the ROI-pooling layer uses max-pooling to produce resolution-reduced feature maps of fixed size $H \times W$ (in our case 6×6). Here we simply recall how the ROI-pooling actually works. The ROI-pooling is essentially a sliding pooling-window, where the window (bin) size is $\lceil \frac{h}{H} \rceil \times \lceil \frac{w}{W} \rceil$ and the strides are $\lfloor \frac{h}{H} \rfloor$ and $\lfloor \frac{w}{W} \rfloor$ along each dimension[4]. For each feature map channel, the pooling-window slides across the ROI region and pools the maximum response in each bin, producing a smaller feature map containing the strongest activations. The resulting feature maps are concatenated into a single feature vector, which is fully connected with the subsequent layer (i.e. FC6 in Fig.3.7). Furthermore, we also follow the Fast R-CNN to restructure the FC6 and FC7 layers via *Truncated SVD*, resulting in size-reduced weight parameters which is useful to accelerate the processing speed. The detailed background for this technique is provided in Appendix B.

When executing people detection, each network works independently to score each proposal. In terms of one proposal, the output $P(y = 1|X_c)$ from the RGB-CNN and the output $P(y = 1|X_d)$ from the Depth-CNN are fused to deduce the final probability $P(y = 1|X_c, X_d)$, which is more convincing as an evidence to classify the proposal. In the two probabilities, $y = 1$ means a window containing a person (upper body), and X_c and X_d represent colour and depth images respectively.

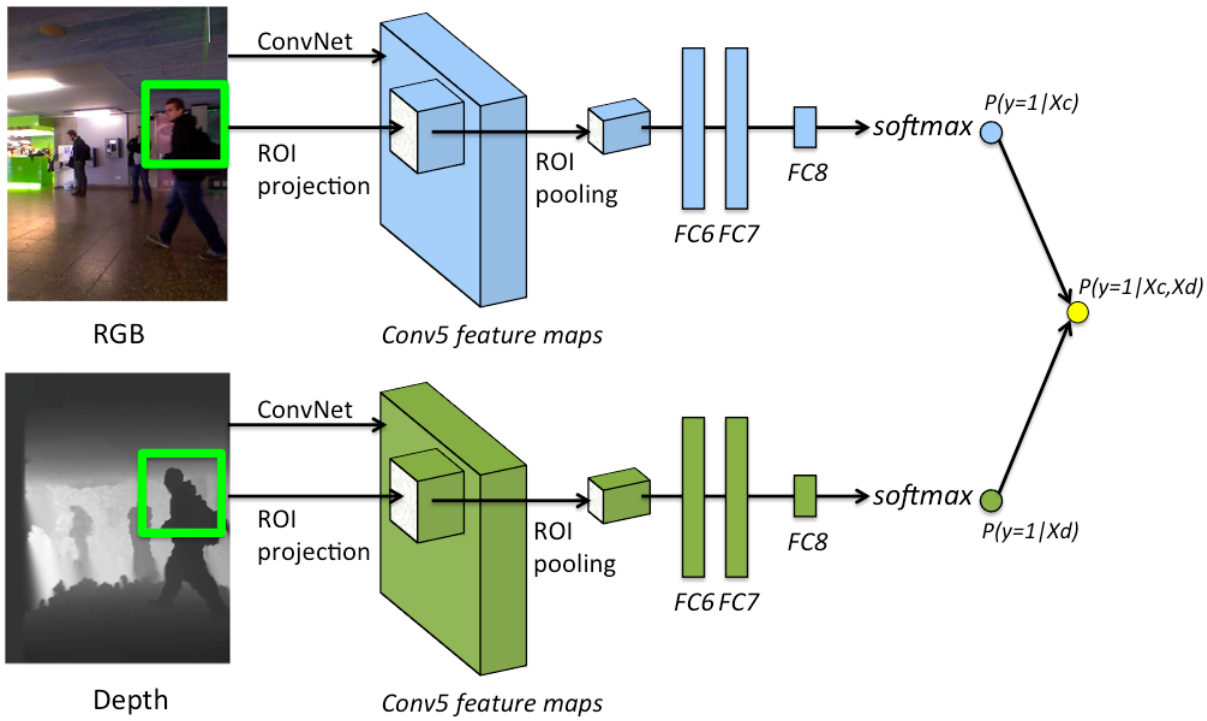


Figure 3.7: Architecture of the RGB-D people detector. The top stream (blue) processes RGB images and the bottom stream (green) processes depth images. Each network takes as input a full-size image and the same set of candidate proposals (ROIs). Each ROI is projected to the conv5 feature maps from which a fixed-length feature vector is pooled by the ROI-pooling layer. The feature vector is then passed through the subsequent FC layers, which finally output a probability. The two output probabilities from each network are fused to infer the final strong probability, deciding whether the ROI window contains a person. The two networks are trained independently using single-size training.

[4]The operators $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ correspond to ceil and floor operations respectively

3.4.2 Fusing the RGB and Depth

In the previous section, we merely mentioned that the two probabilities produced from the two networks are fused, but we did not explain how. In this section, we explain in detail how the two probabilities are fused and provide a justification in the end.

For a candidate proposal, $P(y = 1|X_c)$ represents its probability of containing a person in the RGB image, and $P(y = 1|X_d)$ represents its probability of containing a person in the depth image. We propose to calculate the final strong probability, $P(y = 1|X_c, X_d)$, by:

$$P(y = 1|X_c, X_d) \propto exp((1 - \omega)L(y = 1|X_c) + \omega L(y = 1|X_d)) \tag{3.9}$$

where $L(\cdot|\cdot)$ is log likelihood and ω is a weight dependent on the depth of the proposal. In particular, ω is computed by:

$$\omega = \begin{cases} 1 & d <= 1 \\ -\frac{1}{5}(d-1) + 1 & 1 < d < 6 \\ 0 & d >= 6 \end{cases} \tag{3.10}$$

where d is the depth in meters. A straightforward plot of this function is shown in Fig.3.8.

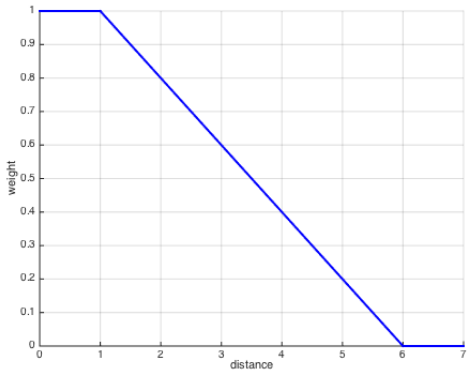


Figure 3.8: The visualisation of the weight of inference based on depth as a function of depth value.

The semantic meaning of Eq.3.9 can be understood as: if the proposal is close to the camera, we put more or all of the weight on the probability computed from depth; if the proposal is far away from the camera, we put more or all of the weight on the probability computed from RGB. This idea is derived from the fact that the performance of the Kinect depth sensor descends as the distance ascends. Fig.3.9 provides a more straightforward way to support this idea. From the three depth images given in Fig.3.9 we can conclude that people closer to the camera have more clear shapes than people who are far away from the camera. The worst case is shown in Fig.3.9d where the two persons standing at the distant place nearly disappear (in fact there is no depth on their body).

Why do we use the combination of RGB and depth images instead of using merely RGB images? Normally, RGB images are rich in, for example, texture and colour, but they are easily affected by poor lighting. In contrast, the depth sensor exhibits a strong ability to work in the environment where illuminations are unstable or varying. This nice property is attributed to the sensor mechanism (section 2.1.2). Taking advantage of this complementary nature of colour and depth images, we are able to obtain better performance in classification, which has been proven in [15, 16, 7]. We will also prove this in chapter 4.

Figure 3.9: Three pairs of RGB and depth images. The top row shows the RGB images and the bottom row shows their corresponding depth images. It can be seen that people closer to the camera have more clear shapes than people who are far away from the camera. The worst case is shown in (d) where the two distant people nearly disappear.

3.4.3 Training the CNNs

Training the networks is a critical job in this project. We do not have a large RGB-D people dataset, thus we cannot afford to train CNNs from scratch. To overcome this problem, we fine-tune our networks with initialised weights pre-trained in ImageNet. In particular, we copy the weights trained on ImageNet to all the layers in the two networks, regardless of the specific modality. Transferring weights from all the layers of a pre-trained network exhibits better generalisation performance, which has been studied in [62]. This rule also applies to the Depth-CNN since there are some common properties as well as high-level semantic information between RGB and encoded depth images such as object boundary [16].

We fine-tune the two networks independently and adopt different fine-tuning strategies. For the RGB-CNN, we lock the five convolutional layers and fine-tune only the last three fully connected layers. On one hand, this prevents overfitting. On the other hand, the convolutional filters are already good at extracting meaningful features in colour images so they do not need to be changed. For the Depth-CNN, we use two fine-tuning approaches. The first approach is identical to the one used in fine-tuning the RGB-CNN. In the second approach, we lock the first two convolutional layers and fine-tune the rest of the network. The motivation for the second approach is to expect the Depth-CNN to generalise better in processing depth images. The reason for keeping the first two layers unchanged is that the features in these two layers are general. The reason for fine-tuning the three convolutional

layers together with the fully connected layers is that these layers contain co-adapted features that are better to be fine-tuned together [62] and that (the layers) should adapt to the image structures specific to depth images. It is also interesting to see which fine-tuning strategy fits better with which depth-encoding method.

Chapter 4

Experiments

In this chapter, we provide detailed implementation of this project as well as the experimental results. The structure of this chapter is organised as follows. In section 4.1, we provide details on the collection of training data and testing data. In section 4.2, we explain how we implement this project, especially on the network training. In section 4.3, we describe the evaluation methodology that is used to evaluate our people detection system. In section 4.4, we analyse the experimental results particularly on the detection accuracy and detection speed. In section 4.5, we evaluate our people detection system.

4.1 Dataset

4.1.1 Training Data

Positive

To collect positive training data, we use video sequences provided by SPHERE[1]. The video was captured by a Kinect sensor in an office where different people walks in and out individually. There are totally 3271 valid frames where people are present. We use the binary masks obtained from OpenNI to extract the upper body parts from these frames. For RGB images, we directly crop the upper body parts with square bounding boxes. For depth images, we first perform the preprocessing steps described in section 3.2 to fill holes and encode them to three channels. Then we crop the upper body parts to generate positive depth data. We end up with 3271 positive images[2]. However, the poses of people in the positive images do not have diverse variations, thus the actually number of positive images with distinct pose types should be less than 327. Example images are shown in the top row of Fig.4.1.

Negative

The SPHERE data were collected in the situation where the Kinect sensor was fixed. Hence the background scene stayed the same throughout the entire frames and we were unable to get diverse negative training data from these sequences. To deal with this issue, we use NUYD2 dataset [67] as an alternative to collect negative training data. NUYD2 dataset is an RGB-D dataset, containing video sequences recorded by a Kinect sensor from a variety of indoor scenes. The diversity of indoor scenes in this dataset makes it suitable for us to sample negative training data. There are various scenarios in this dataset, from which we select eight[3] to generate negative data without people present. In each selected frame, we sample at most five instances with the maximum overlapping ratio of 20%. Each instance is cropped by a square bounding box with a window size dependent on the depth, which is consistent with the way used in the **Scale-Informed Search** (section 3.3.2). As a result, there are 7574 negative images. Example images are shown in the bottom row of Fig.4.1.

[1] http://www.irc-sphere.ac.uk/

[2] Here we mean 3271 images for each modality, including RGB and the four types of depth. The same to the negative images.

[3] The blah scenarios include: basements, classrooms, home offices, libraries, offices, reception rooms, studies and study rooms.

Training and Validation

To train the networks, we split the collected data into training set and validation set, where the training set consists of 2616 positive images and 5464 negative images and the validation set consists of 655 positive images and 2110 negative images.

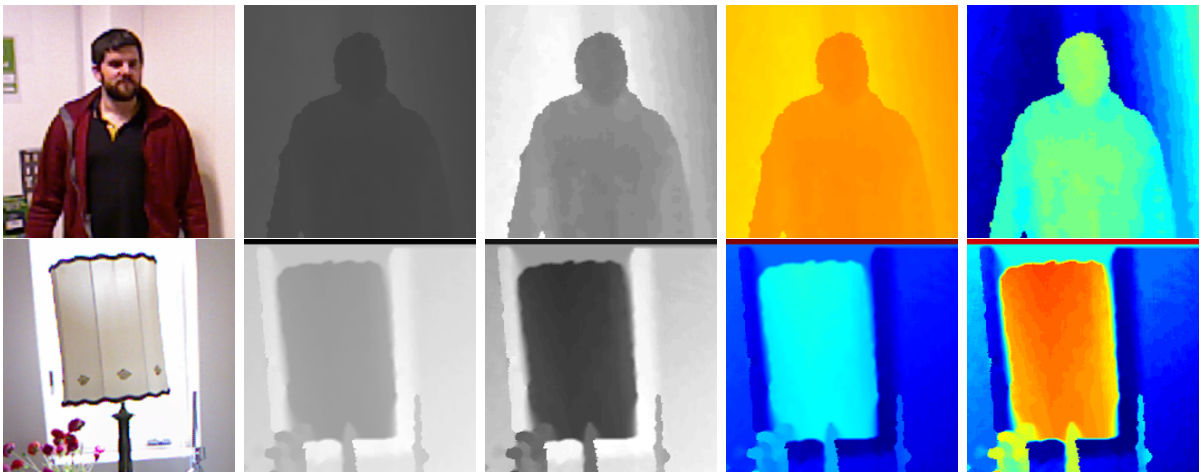


Figure 4.1: Example images of our training data. From column 1 to column 5: RGB, depth-gray, contrast-enhanced depth-gray, colour-depth, contrast-enhanced colour-depth.

4.1.2 Testing Data

We evaluate our people detection system in a publicly available RGB-D dataset where annotations on people are provided. The dataset is called *RGBD-people-dataset* and it is published by Spinello and Arras in [68]. This dataset was collected in the lobby of a university canteen at lunch time. It contains three videos taken by three vertically mounted Kinect sensor (fixed). Each video includes 1000+ pairs of RGB and raw depth images. Only the first 1087 RGB-D frames per video were annotated so we only use 1087 RGB-D frames per video. However, the annotations are not satisfactory because there are some missing annotations. There are two factors which make this dataset challenging for RGB-D people detection. The first is that the collected depth data are very noisy and contain many holes. The second is that some people present in these videos suffer from severe occlusions. Example images can be found in Fig.3.9.

4.2 Implementation Details

We implement this project mainly in MATLAB. We use the open source Caffe to train and test our CNNs in a NVIDIA GeForce GTX with 12GB memory. Stochastic Gradient Descent (SGD) algorithm is used to train the networks.

4.2.1 Single-Size Training

In the Caffe implementation, training images are preferred to have a fixed-size. Therefore, we set the input size of the networks to 115×115. This is smaller than the regular size of 227×227 because people tend to be small in resolution in videos. The cropped training images are resized to 125×125. We apply the common data augmentation technique as implemented in Caffe: random crop[4] + horizontal flip.

[4]Random crop works by randomly cropping a 115×115 patch from the 125×125 image.

4.2.2 SGD Hyperparameters

As mentioned previously that we use CaffeNet as the baseline network. We replace the original 1000-way classification layer with a 2-way classification layer for our purpose (non-person and person). The weights are initialised from zero-mean Gaussian distribution with standard deviation 0.01 and the biases are initialised to zero. The initial learning rate is set to 0.001. For fine-tuning the networks, the initial learning rate is preferred to be small. We actually tried using 0.01 as the initial learning rate and found that the model collapsed after few iterations (the training loss jumped to very high and never decrease). The momentum is 0.9 and the weight decay is 0.0005.

4.2.3 Multi-Scale Detection

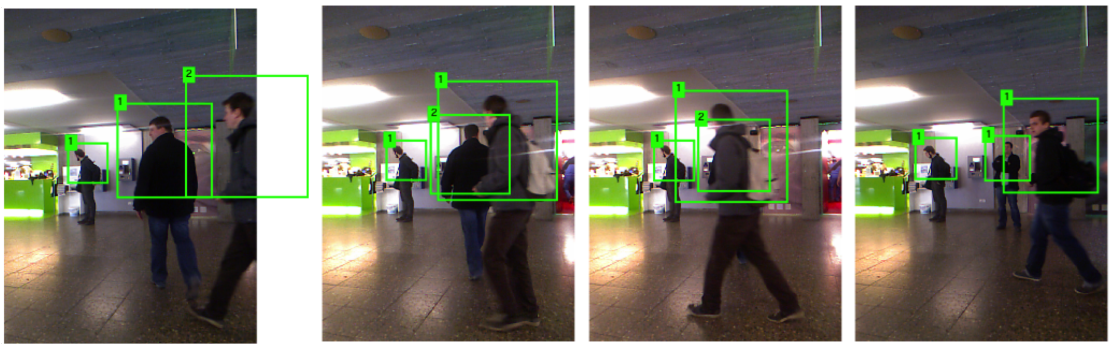
We employ the multi-scale detection [61] to improve the accuracy. Given an RGB (depth) image, we resize it such that $min(w, h) = s \in S = \{288, 360, 480, 600\}$, where w and h represent width and height of the image. For each candidate proposal, the RGB-CNN (or Depth-CNN) finds the single scale $s \in S$ such that the scaled proposal is the closest to 115×115, which is consistent with the training data size. Recall that we use depth to infer the window width, in practice we map the inferred window width to four constants by:

$$win = \begin{cases} 92, & w < 103 \\ 115, & 103 <= w < 134 \\ 154, & 134 <= w < 173 \\ 192, & w >= 173 \end{cases} \tag{4.1}$$

This trick is to make sure the ROI window in the convolutional feature maps has a size of at least 6×6 such that the ROI-pooling layer do not have to simply replicate features to output a 6×6 feature map.

4.3 Evaluation Methodology

We evaluate the effectiveness of our people detection system on the *RGBD-people-dataset*. In the annotations, the label for a person has three possible values: 0 means 'hidden', 1 means 'fully visible' and 2 means 'partially visible'. People with '0' labels are ignored.



(a) (b) (c) (d)

Figure 4.2: Ground truth annotations. 1 means fully visible. 2 means partially visible.

The authors in [68] count a detection as TP[5] if it overlaps with a ground truth by more than 40%. They adopt a rule called 'no-reward-no-penalty' to compute the detection results. In particular,

[5]True Positive: TP, False Positive: FP, False Negative: FN.

a detection matching a partially visible ground truth is not rewarded as TP or penalised as FP. In another case where a partially occluded person is missed, it is not counted as FN. For example, in Fig.4.2a (or Fig.4.2b), if a detection matches the bounding box labeled as '2', it is not counted as TP or FP. Although using this rule reduces the actual number of people that need to be detected, we have to follow this rule in our evaluation because, for example, in Fig.4.2c the person fully occluded is given an annotation '2'[6], without following the 'no-reward-no-penalty' rule this person will be regarded as FN which is incorrect.

There is another deficiency exposed in the annotations. In some frames, there are some people that lack annotations. For instance, in Fig.4.2d the middle person was not annotated in the provided annotation file. To get a fair evaluation, we loop through each frame in this dataset and re-annotate people that were supposed to be annotated (we respect the rule where 1 for 'fully visible' and '2' for 'partially visible').

In our evaluation, we also adopt the 'no-reward-no-penalty' rule, but we count a detection as TP if it overlaps with a ground truth by more than 50%. One ground truth is allowed to match at most one detection. Multiple detections corresponding to the same person will be penalised, i.e., only one is counted as TP and the rest are regarded as FP. We use *Average Precision* (AP) against *Average Recall* (AR) to evaluate the detection performance. In particular, for a frame, our people detection system generates a list of detections that are post-processed (NMS). By comparing with the ground truths in this frame we can get the TP, FP and FN. We accumulate these three metrics over all frames (3261 frames). In the end, the AP is obtained by:

$$AP = \frac{TP}{TP + FP} \tag{4.2}$$

and the AR is obtained by:

$$AR = \frac{TP}{TP + FN} \tag{4.3}$$

Note that each term in the two equations is divided by the total number of frames (3261). We use Average Precision-Average Recall instead of Precision-Recall to show the *per-image* performance.

4.4 Results and Analysis

We provide the experimental results and analyses on the detection accuracy and detection speed in section 4.4.1 and 4.4.2 respectively. Note that the detection accuracy here refers to the overall performance of the people detection system, rather than the statistical term which is computed by $(TP + TN)/(TP + FP + TN + FN)$.

4.4.1 Detection Accuracy

We adopt a single fine-tuning strategy for the RGB-CNN but different fine-tuning strategies for the Depth-CNN. We fine-tune the RGB-CNN for three epochs where the global learning rate is lower by 1/10 after two epochs. We fine-tune the Depth-CNN using three different settings: (1) fine-tune only the three fully connected layers for one epoch (fc678_epoch1); (2) fine-tune the last three convolutional layers and three fully connected layers together for one epoch (conv345fc678_epoch1); (3) fine-tune the last three convolutional layers and three fully connected layers for three epochs (conv345fc678_epoch3), where the global learning rate is lowered by 1/10 after two epochs. The reason for the use of these three settings is to see whether fine-tuning more layers or for more epochs can make the Depth-CNNs adapt better to the encoded depth data.

[6]This dataset is also used for tracking so some people that are severely occluded or hidden are labelled as 'partially visible'.

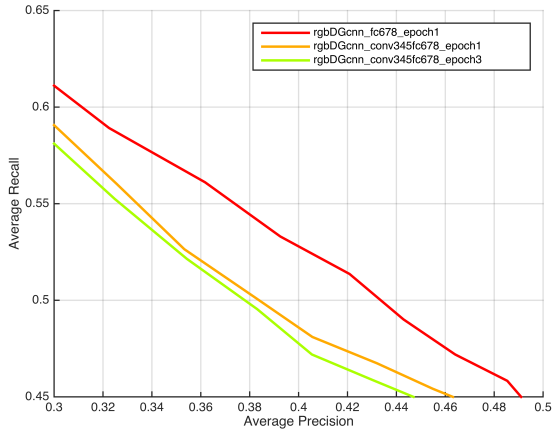
The experimental results of different settings for different combinations of RGB and depth are reported in Fig.4.3a to 4.3d. Recall that we propose four depth-encoding methods so there are four types of models that are going to be evaluated. For simplicity, we denote the four combinations of RGB and depth by rgbDGcnn, rgbCEcnn, rgbCDcnn and rgbCECDcnn, where DG means 'depth-gray', CE means 'contrast-enhanced-depth-gray', CD means 'colour-depth' and CECD means 'contrast-enhanced-colour-depth'. The results of different settings for each model are summarised as follows.

rgbDGcnn: Fig.4.3a shows that fine-tuning only fc678 for one epoch outperforms the other two settings by noticeable margins. Fine-tuning more layers or for more epochs seems to cause overfitting. This is probably due to that the DG-encoding simply replicates the information contained in one channel to three channels, thus the convolutional layers in the pre-trained model are able to extract good features on DG images without extra changes.

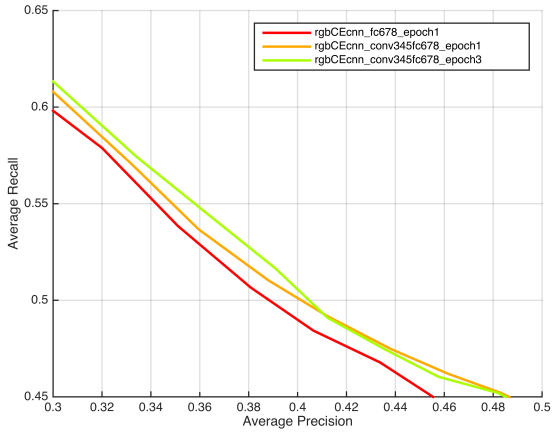
rgbCEcnn: Fig.4.3b shows that the more the layers (conv345fc678) being fine-tuned, the better the performance achieved. However, fine-tuning for more epochs only exhibits a slightly better performance. The results obtained by the CE-encoding are rather different from the DG-encoding, although they both replicate one-channel to three-channel. The interpretation for this phenomenon might be that applying histogram-equalisation has changed or enriched the information contained in a 2D depth image, thus fine-tuning more layers and more epochs is helpful to make the Depth-CNN adapt better to the data.

rgbCDcnn: Fig.4.3c exhibits similar results with the CE-encoding where fine-tuning the convolutional layers and fully connected layers jointly (conv345fc678) for 3 epochs gives the best performance.

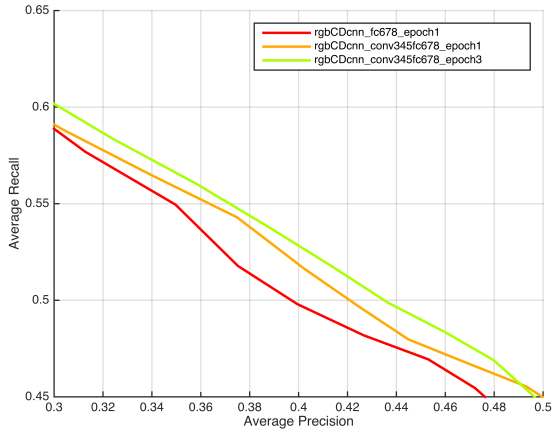
rgbCECDcnn: Fig4.3d shows that the three settings produce very similar results. This might be interpreted as that the CECD-encoding makes depth images closer to colour images in terms of information contained, therefore less efforts are required to adjust the pre-trained model.



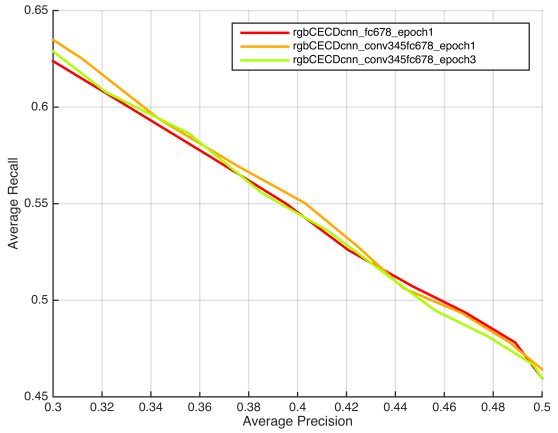
(a) rgb + depth-gray



(b) rgb + contrast-enhanced-depth-gray



(c) rgb + colour-depth



(d) rgb + contrast-enhanced-colour-depth

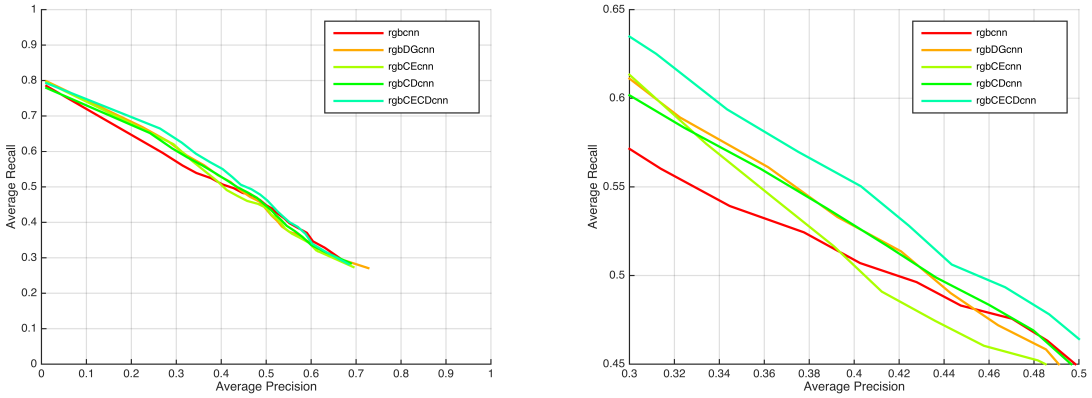
Figure 4.3: Performances of RGB-Depth-CNNs with different training strategies. The different training strategies are applied only to Depth-CNNs. Note that the figures are zoomed in to focus on the interesting sections. They have the same axes.

In the two contrast enhanced cases, fine-tuning the convolutional layers is not that useful. One possible explanation for that is that the contrast enhancement gives to the depth image similar properties than a colour image (notably in terms of contrast and edge strength), so the features are still reasonably valid for the depth image.

We choose the best performing model in each combination to compare with the baseline model, rgbcnn, which uses a single stream CNN processing RGB images. The experimental results are reported in Fig.4.4b. Overall, the performances of these models can be sorted in the decreasing order as: rgbCECDcnn > rgbCDcnn > rgbDGcnn > rgbcnn > rgbCEcnn (we use the AP at 0.44 as a reference point to compare results). Some example detection results using the rgbCECDcnn are provided in Fig.4.7.

The rgbCECDcnn produces the best performance among the five models including the baseline. It indicates that with the same pre-trained model, the CECD-encoding is better than the other encoding methods in producing new depth images that are closer to colour images. In other words, by using the CECD-encoding, the Depth-CNN requires less efforts on adjusting the pre-trained model. The rgbCDcnn and rgbDGcnn are very similar in performance. They both outperform the rgbcnn by noticeable margins, but they are inferior to the rgbCECDcnn. The rgbCEcnn is able to beat the

rgbcnn if the threshold is carefully chosen. It degrades rapidly as the threshold increases[7], which might indicate that some detected people do not obtain high confidence scores. However, with more training data and more epochs in fine-tuning, the CE-CNN is likely to converge better and the rgbCEcnn is then able to beat the baseline.



(a) Full detection result without zooming in

(b) Zoomed version of (a) in the most interesting area

Figure 4.4: Comparison of RGB-CNN (baseline) and the combinations of RGB and Depth CNNs. The best performing model is rgbCECDcnn.

4.4.2 Detection Speed

The ROI selection module in our people detection system consists of three sub-modules, which are ground plane detection (GPD), scale-informed search (SIS) and candidate proposal filtering (CPR). To test the effectiveness, we use different configurations of these sub-modules to generate proposals in images. We use three different settings: (1) only SIS is used; (2) SIS and GPD are used; (3) SIS, GPD and CPF are used, which is equivalent to the entire ROI selection module. We compare the averaged number of proposals generated in an image of size 640×480 by using these three settings. The traditional sliding-window[8] is used as a baseline. The statistical results are summarised in Table 4.1. Without any ROI selection, the raw number of proposals generated by the traditional sliding-window is approximately 49152, which will take a lot of time for a detection system to process unless a fast cascade is used instead of our CNNs. With the SIS (one scale for one pixel), the number of proposals is decreased to 7212, which exhibits a significant improvement. By removing the ground plane (GPD), the number of proposals is decreased to 5651. By discarding those proposals that mainly contain invalid depth pixels (CPF), the number of proposals is further reduced to 5023, which is much less than the crude 49152.

Table 4.2 provides a breakdown of the runtime of different parts in our system. The ROI selection takes about 14.5 milliseconds (ms) to generate a list of refined proposals. A single stream CNN, e.g. RGB-CNN or Depth-CNN, takes roughly 131 ms to process one image. The combination of RGB-CNN and Depth-CNN, e.g. rgbCECDcnn, takes about 284 ms to process one RGB-D image. In total, the people detection system, where two streams of CNN are used, runs at around 3.35 fps[9].

[7] The performance curve is generated by varying the threshold.
[8] We set the number of scales to be scanned at each pixel to four.
[9] $1/(0.0145 + 0.284) \approx 3.35$

	Num. of Proposals
Sliding-window	49152
SIS	7212
SIS + GPD	5651
SIS + GPD + CPF	5023

Table 4.1: Number of proposals generated by different methods. SIS represents 'scale-informed search' (section 3.3.2), GPD represents 'ground plane detection' (section 3.3.1) and CPF represents 'candidate proposal filtering' (section 3.3.3). Stride equal to 5 is used in all cases.

	Time (ms)
ROI selection	14.5
Single stream CNN	131
Double streams CNN	284

Table 4.2: Breakdown of runtime for each part of the detection system. Single stream CNN corresponds to RGB-CNN. Double streams CNN corresponds to the combination of RGB-CNN and Depth-CNN.

4.5 Discussion

How does adding depth to the CNN detection improve the detection accuracy

In general, adding depth to the CNN detection can improve the detection accuracy as illustrated in the previous section. Here we mainly investigate how adding Depth-CNN addresses the shortcomings of using only RGB-CNN by looking at specific detection results. To help demonstrate our findings, we give some representative detection results obtained by the rgbcnn (baseline) and rgbCECDcnn (best model) in Fig.4.5. In Fig.4.5, the first row shows the detection results obtained by the rgbcnn (we draw the bounding boxes in colour images); the second and third row show the results obtained by the rgbCECDcnn (in the third row we draw the bounding boxes in encoded depth images). We summarise the findings into three bullet points:

- **Depth-CNN is more tolerant to pose variations:** the front person in Fig.4.5a is missed by RGB-CNN, which might be caused by the small variation in pose. However, by adding Depth-CNN, the confidence level for this person is increased to 0.657, which supports that Depth-CNN is more tolerant to pose variations than RGB-CNN.
- **Shape information somewhere serves as a stronger clue than colour information:** in Fig.4.5b, the front person cannot be detected, which is potentially caused by the blurring effect and weak contrast. Fortunately, the negative impact of these two shortcomings can be alleviated by using depth images. For example, it can be seen in Fig.4.5h that the shape of the person still has a high discriminability. This indicates that depth images are more robust to capture the shape information (or object boundaries)[10], which is attributed to the depth sensor mechanism.
- **Confidence level can be strengthened by adding Depth-CNN:** it is worth noting that confidence levels (CL) corresponding to correct detections can be strengthened by adding Depth-CNN. For instance, by adding Depth-CNN, the CL of the right person in Fig.4.5a is increased from 0.932 to 0.959; the CL of the left person in Fig.4.5b is increased from 0.946 to 0.961; the CL of the right-most person in Fig.4.5c is increased from 0.609 to 0.852. This is also the evidence that the way we fuse colour and depth information to get stronger predictions is effective.

[10]This only applies to close distance.

Does the ROI selection hurt the detection accuracy

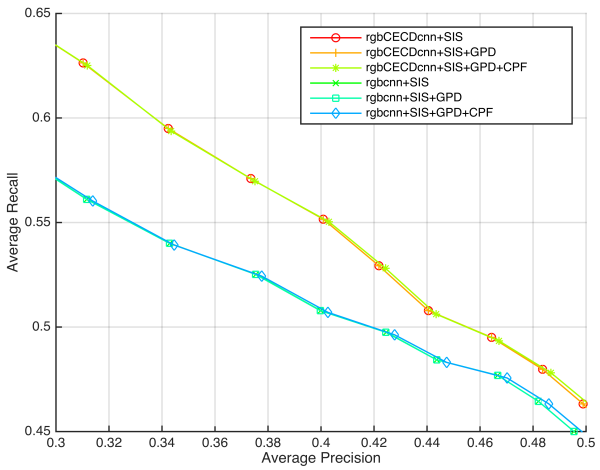


Figure 4.6: Comparison of models with and without the ROI selection. Models with and without the ROI selection nearly overlap, which means the ROI selection does not harm the detection accuracy.

Although using the ROI selection method is useful to reduce the runtime (Table 4.1), we have to verify whether it kills any true positive windows in advance or not. For example, we might be worried about whether using **Candidate Proposal Filtering** is likely to omit some positive windows. To this end, we design an experiment where we use the rgbCECDcnn (the best model) and rgbcnn to perform people detection in the *RGBD-people-dataset* but with different configurations of the ROI selection sub-modules. In particular, we use the SIS as the basic ROI selection method and add the GPR and CPF incrementally so that we can compare the resulting curves to see if any sub-modules lead to reduction in performance. The experimental results are reported in Fig.4.6. From the results we can see that the three different settings of each model nearly overlap with each other. This evidence strongly suggests that the ROI selection does not hurt the detection accuracy. On the other hand, the ROI selection does not improve it significantly either. It might be expected that the ROI selection is helpful to remove in advance some FPs, but this is not the case and the FPs tend to be in ROIs, which makes sense given the type of ROI selection.

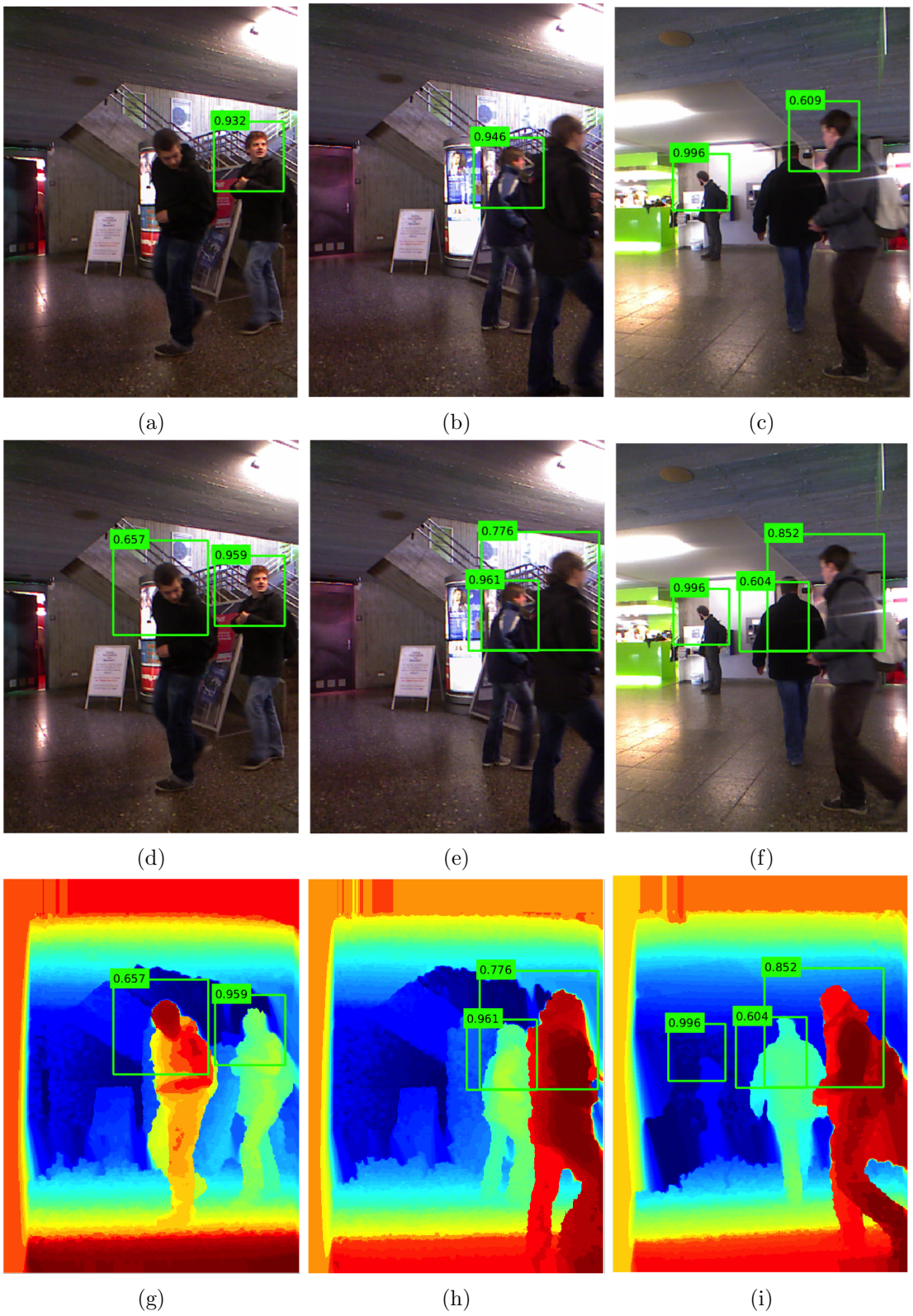
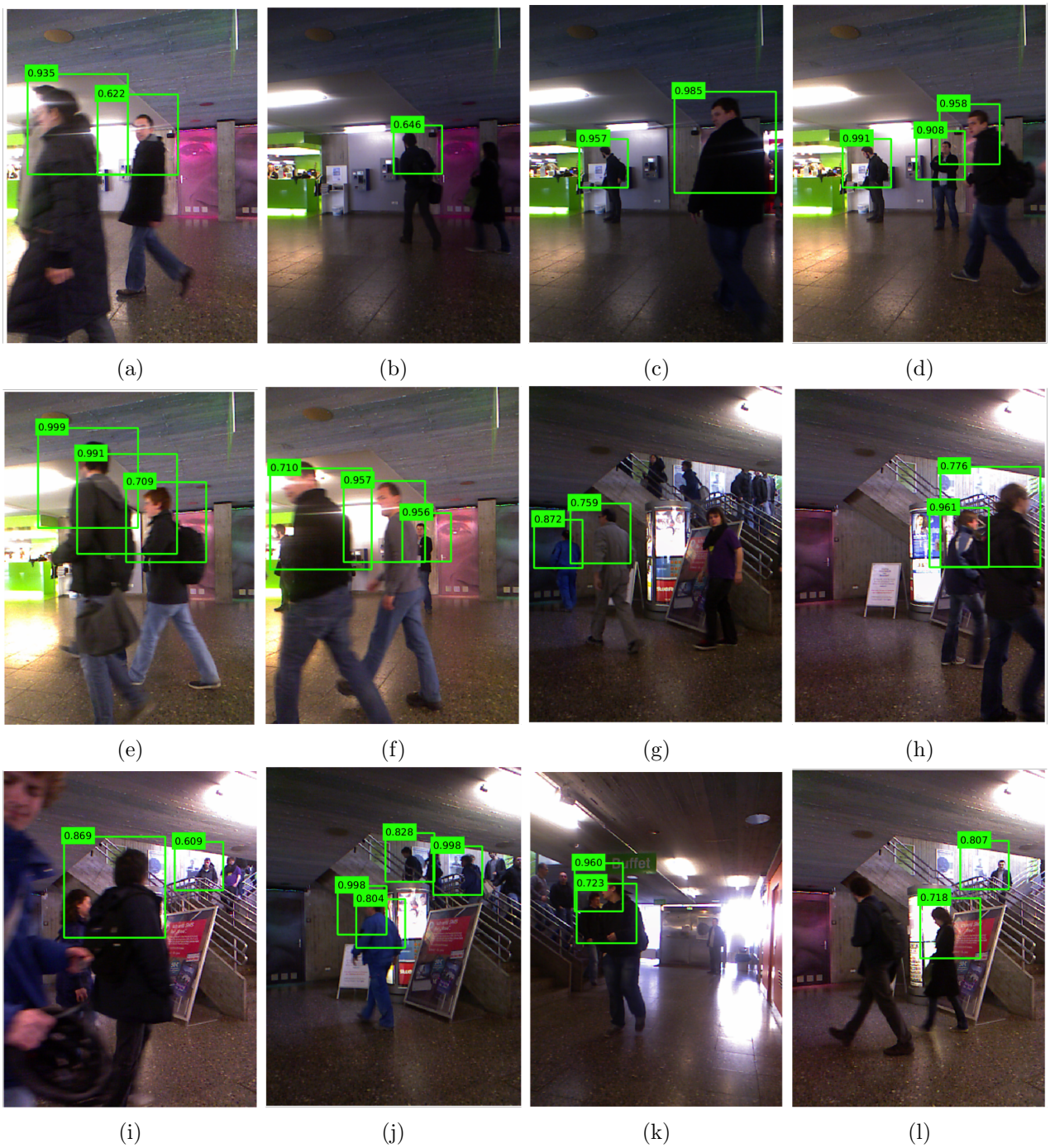


Figure 4.5: Example detection results obtained by rgbcnn and rgbCECDcnn. Each column correspond to the same frame. The 1st row results are obtained by the rgbcnn, and the 2nd and 3rd row results are obtained by the rgbCECDcnn. We also give CECD-encoded depth images (3rd row) in order to highlight the advantages of depth images that are not present in colour images.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 4.7: Examples of detection results in the *RGBD-people-dataset* using rgbCECDcnn.

Chapter 5

Conclusion and Future Work

In this chapter, we first summarise this project on what we have done and what have been achieved and give a summary of contributions. Then we address the limitations exposed in the experiments, followed by some suggestions of future work to improve this project.

5.1 Conclusion

We have designed and developed a CNN-based people detection system that works in RGB-D data. The system consists of an ROI selection module that exploits depth information to produce a list of candidate proposals, and a CNN-based people detector that produces a prediction for each proposal by incorporating colour and depth information. To the best of our knowledge, this is the first work that applies CNNs in RGB-D images for people detection.

In the ROI selection module, we present a robust ground plane detection algorithm that is able to accurately detect the ground plane in depth images regardless of the presence of substantial outliers. We use the depth information to heuristically infer the scale of sliding-window at each ROI pixel, which avoids redundant computations on windows of multiple scales. We further reduce the candidate proposal number by discarding proposal windows that mainly contain invalid depth pixels, which acts like a simple classifier in cascade structures. With this ROI selection module, the number of proposals generated for a single frame is reduced by a considerable amount. We have conducted experiments to prove that the ROI selection module is not harmful to the detection accuracy.

The people detector consists of two CNNs working independently in colour and depth images. We employ the idea of the ROI-pooling from Fast R-CNN [10] in our network design to promote the sharing of computations on convolutions. We present a novel way to fuse colour and depth information based on the depth value. In doing so, more convincing detection results can be obtained, which has been proven in section 4.5. To make the depth images compatible with the pre-trained model, we transform them from one-channel to three-channel by trying four different depth-encoding methods. Under the condition where the same pre-trained model is used, the rgbCECDcnn model produces the best detection result, indicating that adding depth information can boost the detection accuracy. This also suggests that when depth data are insufficient in quantity, the CECD-encoding can make depth images closer to colour images so that less efforts are required to adjust the model pre-trained on colour images in order to adapt to depth images. We do not have time to deploy other methods to compare with ours, which can be added to the future work.

5.2 Contributions

The contributions of this project are summarised as follows.

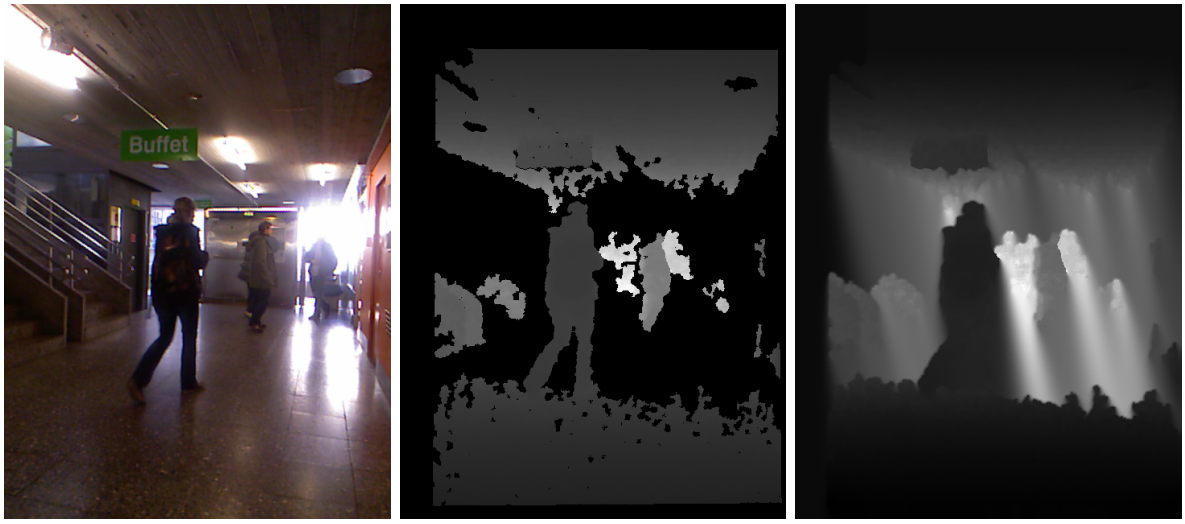
1. We proposed an efficient ROI selection method, which is useful to reduce search space for the people detector.
2. We proposed a CNN-based people detector that works in RGB-D data.
3. We proposed a novel way of fusing colour and depth information based on the depth value.
4. We proposed the CECD-encoding that can transform a single-channel depth image to three-channel while preserving the important features contained in the original image.

5. We re-annotated the *RGBD-people-dataset* to fill some missing annotations, which can be made available publicly.

5.3 Limitations

There exist two major weaknesses in our people detection system. (1) Our people detection system runs at roughly 3.35 fps when two CNNs work at the same time. Although it is faster than those methods that do not share computations on convolutions (note that they only process colour images while we process colour and depth images), it is still far away from the standard real-time requirement (30 fps). (2) The people detector, especially for the Depth-CNN, does not work well in depth images where massive holes are present. For example, Fig.5.1b shows the raw depth image where there are lots of holes. Fig.5.1c shows the depth image that has been applied the hole-filling technique. However, the boundaries of the near person are somewhat damaged. Therefore, making Depth-CNNs work in noisy depth images without hole-filling is a promising direction, which will be more flexible in real-world applications.

Due to time constraints, we only tested our method in one dataset i.e. one scenario, which is not enough. In the future, we can test our method in more scenarios (datasets). Also, we can compare our method with other methods such as the VJ detector [65] and HOG detector [8].



(a) RGB (b) Raw depth (c) Filled depth

Figure 5.1: There exist massive holes (black regions) in the raw depth image. After applying hole-filling, the shapes of people are somewhat distorted, especially for the distant one. In this frame, our people detector outputs nothing.

5.4 Future Work

Accelerate the people detector

We can consider to work in two directions to accelerate the people detector. First, we can insert (a) small CNN(s) between the ROI selection and the people detector [69]. The small CNN(s) is (are) expected to work fast to reject some easy windows such that fewer windows are passed to the people detector, in other words, the workload for the people detector is reduced. However, we need to collect a larger dataset so we can afford to train the small CNN(s) from scratch. Second, we can improve the

detection mechanism for the people detector, i.e. to change the multi-scale detection to single-scale-detection. The precondition of achieving this goal is to have training images where people have both large and small resolutions. Thus, we can force the CNNs to learn scale-invariance.

Make Depth-CNNs work in noisy depth data

It would be more robust to enable Depth-CNNs to work in noisy depth data without hole-filling. To achieve this, we can make the depth training data noisy such that Depth-CNNs are forced to perform more robust classification in noisy data [7]. Another way that might achieve this goal is to use the spatially-sparse CNN (SS-CNN) proposed by Graham in [70]. The main feature in SS-CNN is that neurons linked to irrelevant pixels (e.g. zero depth values) are deactivated, which might be useful in performing classification in noisy depth images.

Jointly train the RGB-CNN and Depth-CNN

Recall that we used a fusion function to combine the probabilities produced separately from the RGB-CNN and Depth-CNN, where the weight parameters were heuristically determined. In the future, it would be more interesting to learn this fusion function [71]. But we also need to collect training images where resolutions of people are diverse.

Appendix A

Plane Coefficients Computation

Determination of Three Non-Collinear Points

Given the three randomly selected 3D points, p_1, p_2 and p_3, we construct two vectors that are $\overrightarrow{p_1p_2}$ and $\overrightarrow{p_1p_3}$. The easiest way to determine whether these three points are collinear is to compute the cosine value of the angle between the two vectors:

$$\cos\theta = \frac{\overrightarrow{p_1p_2} \cdot \overrightarrow{p_1p_3}}{||\overrightarrow{p_1p_2}|| \cdot ||\overrightarrow{p_1p_3}||} \tag{A.1}$$

where $||\cdot||$ denotes the norm of a vector. As a result, the three points are non-collinear if the absolute of $\cos\theta$ does not equal to one, or collinear otherwise.

Determination of Plane Coefficients

In a 3D world, a plane is expressed as a four-element vector, $\phi = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$, satisfying the equation:

$$ax + by + cz + d = 0 \tag{A.2}$$

where $\begin{bmatrix} x & y & z \end{bmatrix}$ corresponds to the coordinates of a 3D point. We transform Eq.A.2 to:

$$ax + by + d = z \tag{A.3}$$

thus c is fixed to -1. All the three points should satisfy Eq.A.3. Denote the known points by $p_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}$ where $i \in \{1, 2, 3\}$, fitting the three points in Eq.A.3 can be expressed in the matrix form:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \\ d \end{bmatrix}}_{\mathbf{v}} = \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}}_{\mathbf{Z}} \tag{A.4}$$

therefore, $\mathbf{v}$ can be obtained by:

$$\mathbf{v} = \mathbf{A}^{\dagger}\mathbf{Z} \tag{A.5}$$

where $\mathbf{A}^{\dagger}$ is the pseudo inverse of $\mathbf{A}$.

Appendix B

Restructuring Neural Networks by Singular Value Decomposition

With the invention of the ROI-pooling layer [10], an input image to the CNN is only required to be convolved once. In doing so, the proportion of time spent processing the fully connected layers is increased because the number of ROIs is large. In fact, the operations between fully connected layers are simply matrix multiplications. Singular Value Decomposition (SVD) can be used to efficiently compress the weight matrices of those fully connected layers (this technique is called truncated SVD), which is helpful to accelerate the processing speed.

The principle of truncated SVD for compressing the weight matrix is described as follows. The weight matrix for a fully connected layer, $W_{m\times n}$, can be factorised as:

$$W_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n} \tag{B.1}$$

where U and V contain the left-singular vectors and right-singular vectors of W, respectively, and Σ is a diagonal matrix where the singular values of W are stored in the diagonal. By taking the t largest singular values in Σ, the weight matrix is approximated as:

$$W_{m\times n} \approx U_{m\times t}\Sigma_{t\times t}V^T_{t\times n} \tag{B.2}$$

therefore, this fully connected layer can be decomposed into two fully connected layers, which are $U_{m\times t}$ and $\Sigma_{t\times t}V^T_{t\times n}$. They are visualised in Fig.B.1.

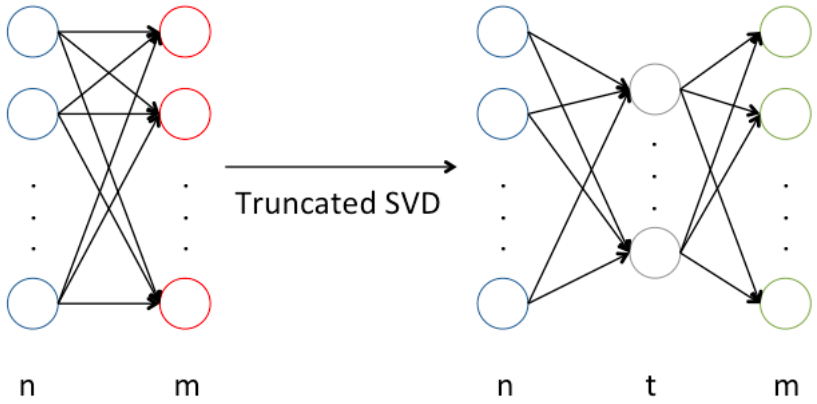


Figure B.1: Truncated SVD. The red layer in the left network is decomposed into the gray layer and green layer in the right network. In the right network, the non-linearity is only applied to the green layer.

Bibliography

[1] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

[2] D. Mitzel and B. Leibe, "Close-range human detection for head-mounted cameras," in *British Machine Vision Conference*, pp. 1–11, 2012.

[3] M. Bansal, S.-H. Jung, B. Matei, J. Eledath, and H. S. Sawhney, "A real-time pedestrian detection system based on structure and appearance classification.," in *ICRA*, vol. 10, pp. 903–909, 2010.

[4] R. Muñoz-Salinas, E. Aguirre, and M. García-Silvente, "People detection and tracking using stereo vision and color," *Image and Vision Computing*, vol. 25, no. 6, pp. 995–1007, 2007.

[5] M. Camplani and L. Salgado, "Adaptive background modeling in multicamera system for real-time object detection," *Optical Engineering*, vol. 50, no. 12, pp. 127206–127206, 2011.

[6] Y. LeCun and M. Ranzato, "Deep learning tutorial," in *Tutorials in International Conference on Machine Learning (ICML'13)*, Citeseer, 2013.

[7] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 681–687, IEEE, 2015.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

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

[10] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[11] Y. LeCun, K. Kavukcuoglu, C. Farabet, *et al.*, "Convolutional networks and applications in vision.," in *ISCAS*, pp. 253–256, 2010.

[12] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2056–2063, 2013.

[13] P. Luo, Y. Tian, X. Wang, and X. Tang, "Switchable deep network for pedestrian detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 899–906, 2014.

[14] J. Hosang, M. Omran, R. Benenson, and B. Schiele, "Taking a deeper look at pedestrians," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4073–4082, 2015.

[15] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision*, pp. 345–360, Springer, 2014.

[16] J. Hoffman, S. Gupta, J. Leong, S. Guadarrama, and T. Darrell, "Cross-modal adaptation for rgb-d detection," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5032–5039, IEEE, 2016.

[17] L. Cruz, D. Lucio, and L. Velho, "Kinect and rgbd images: Challenges and applications," in *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on*, pp. 36–49, IEEE, 2012.

[18] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5636–5643, IEEE, 2014.

[19] K. K. Biswas and S. K. Basu, "Gesture recognition using microsoft kinect®," in *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pp. 100–103, IEEE, 2011.

[20] B. Crabbe, A. Paiement, S. Hannuna, and M. Mirmehdi, "Skeleton-free body pose estimation from depth images for movement analysis," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 70–78, 2015.

[21] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[22] S. B. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor-system description, issues and solutions," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pp. 35–35, IEEE, 2004.

[23] M. Camplani and L. Salgado, "Efficient spatio-temporal hole filling strategy for kinect depth maps," in *IS&T/SPIE Electronic Imaging*, pp. 82900E–82900E, International Society for Optics and Photonics, 2012.

[24] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[25] H. Zhang, C. Reardon, and L. E. Parker, "Real-time multiple human perception with color-depth cameras on a mobile robot," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1429–1441, 2013.

[26] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[27] M. Munaro, F. Basso, and E. Menegatti, "Tracking people within groups with rgb-d data," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2101–2107, IEEE, 2012.

[28] M. Munaro, C. Lewis, D. Chambers, P. Hvass, and E. Menegatti, "Rgb-d human detection and tracking for industrial environments," in *Intelligent Autonomous Systems 13*, pp. 1655–1668, Springer, 2016.

[29] D. R. Chambers, C. Flannigan, and B. Wheeler, "High-accuracy real-time pedestrian detection system using 2d and 3d features," in *SPIE Defense, Security, and Sensing*, pp. 83840G–83840G, International Society for Optics and Photonics, 2012.

[30] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *European Conference on Computer Vision*, pp. 705–718, Springer, 2008.

[31] M. Bansal, B. Matei, H. Sawhney, S.-H. Jung, and J. Eledath, "Pedestrian detection with depth-guided structure labeling," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 31–38, IEEE, 2009.

[32] J. Liu, Y. Liu, G. Zhang, P. Zhu, and Y. Q. Chen, "Detecting and tracking people in real time with rgb-d camera," *Pattern Recognition Letters*, vol. 53, pp. 16–23, 2015.

[33] J. Liu, Y. Liu, Y. Cui, and Y. Q. Chen, "Real-time human detection and tracking in complex environments using single rgbd camera," in *2013 IEEE International Conference on Image Processing*, pp. 3088–3092, IEEE, 2013.

[34] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan, and L. H. Matthies, "A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1466–1485, 2009.

[35] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[36] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, IEEE, 1999.

[37] C. R. del Blanco, T. Mantecón, M. Camplani, F. Jaureguizar, L. Salgado, and N. García, "Foreground segmentation in depth imagery using depth and spatial dynamic models for video surveillance applications," *Sensors*, vol. 14, no. 2, pp. 1961–1987, 2014.

[38] G. Galanakis, X. Zabulis, P. Koutlemanis, S. Paparoulis, and V. Kouroumalis, "Tracking persons using a network of rgbd cameras," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, p. 63, ACM, 2014.

[39] D. Beymer and K. Konolige, "Real-time tracking of multiple people using stereo," in *Proc. of IEEE frame rate workshop*, 1999.

[40] E. Almazan and G. Jones, "Tracking people across multiple non-overlapping rgb-d sensors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 831–837, 2013.

[41] E. J. Almazán and G. A. Jones, "A depth-based polar coordinate system for people segmentation and tracking with multiple rgb-d sensors," in *ISMAR 2014 Workshop on Tracking Methods & Applications*, 2014.

[42] J. Han, E. J. Pauwels, P. M. de Zeeuw, and P. H. de With, "Employing a rgb-d sensor for real-time tracking of humans across multiple re-entries in a smart environment," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 255–263, 2012.

[43] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, pp. 28–31, IEEE, 2004.

[44] J. Cheng, J. Yang, Y. Zhou, and Y. Cui, "Flexible background mixture models for foreground segmentation," *Image and Vision Computing*, vol. 24, no. 5, pp. 473–482, 2006.

[45] S.-Y. Yang and C.-T. Hsu, "Background modeling from gmm likelihood combined with spatial and color coherency," in *2006 International Conference on Image Processing*, pp. 2801–2804, IEEE, 2006.

[46] G. Gordon, T. Darrell, M. Harville, and J. Woodfill, "Background estimation and removal based on range and color," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, IEEE, 1999.

[47] C. Eveland, K. Konolige, and R. C. Bolles, "Background modeling for segmentation of video-rate stereo sequences," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pp. 266–271, IEEE, 1998.

[48] J. Salas and C. Tomasi, "People detection using color and depth images," in *Mexican Conference on Pattern Recognition*, pp. 127–135, Springer, 2011.

[49] M. Harville, G. Gordon, and J. Woodfill, "Foreground segmentation using adaptive mixture models in color and depth," in *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pp. 3–11, IEEE, 2001.

[50] M. Harville, "Stereo person tracking with adaptive plan-view templates of height and occupancy statistics," *Image and Vision Computing*, vol. 22, no. 2, pp. 127–142, 2004.

[51] R. Muñoz-Salinas, "A bayesian plan-view map based approach for multiple-person detection and tracking," *Pattern Recognition*, vol. 41, no. 12, pp. 3665–3676, 2008.

[52] S. Bahadori, L. Iocchi, G. Leone, D. Nardi, and L. Scozzafava, "Real-time people localization and tracking through fixed stereo vision," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp. 44–54, Springer, 2005.

[53] S. Bahadori, L. Iocchi, G. Leone, D. Nardi, and L. Scozzafava, "Real-time people localization and tracking through fixed stereo vision," *Applied Intelligence*, vol. 26, no. 2, pp. 83–97, 2007.

[54] 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)*, pp. 807–814, 2010.

[55] I. G. Y. Bengio and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016.

[56] 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.

[57] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[58] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[59] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 814–830, 2016.

[60] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision*, pp. 346–361, Springer, 2014.

[62] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[63] W. Choi, C. Pantofaru, and S. Savarese, "Detecting and tracking people using an rgb-d camera via multiple detector fusion," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 1076–1083, IEEE, 2011.

[64] M. Y. Yang and W. Förstner, "Plane detection in point cloud data," in *Proceedings of the 2nd int conf on machine control guidance, Bonn*, vol. 1, pp. 95–104, 2010.

[65] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511, IEEE, 2001.

[66] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, ACM, 2014.

[67] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision*, pp. 746–760, Springer, 2012.

[68] L. Spinello and K. O. Arras, "People detection in rgb-d data," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3838–3843, IEEE, 2011.

[69] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson, "Real-time pedestrian detection with deep network cascades," 2015.

[70] B. Graham, "Spatially-sparse convolutional neural networks," *arXiv preprint arXiv:1409.6070*, 2014.

[71] J. Li, X. Liang, S. Shen, T. Xu, and S. Yan, "Scale-aware fast r-cnn for pedestrian detection," *arXiv preprint arXiv:1510.08160*, 2015.