DRONE BASED OBJECT TRACKING WITH CAMSHIFT ALGORITHM AND NEURAL NETWORK

by

Xin Zhang, MS

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Computer Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

MAY, 2021

DRONE BASED OBJECT TRACKING WITH CAMSHIFT ALGORITHM AND NEURAL NETWORK

by

Xin Zhang

APPROVED BY

_____

Jiang Lu, PhD, Chair

_____

Liwen Shih, PhD, Committee Member

_____

Xiaokun Yang, PhD, Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

_____

David Garrison, PhD, Associate Dean

_____

Miguel A. Gonzalez, PhD, Dean

Dedication

This thesis is wholeheartedly dedicated to our family, who have been our source of inspiration and gave us strength when I thought of giving up, who continually provide their moral, spiritual, emotional, and financial support.

To my wife, who shared her words of advice and encouragement to finish this study. And lastly, we dedicated this thesis to my daughters, they are treasure in my life.

Acknowledgments

Before anything else, I would first like to thank my committee chair, Dr. Jang Lu for his perseverant guidance, patience, and support throughout the course of the research. Without his great help, this research would not have been possible.

Also, thanks to Dr. Liwen Shih and Dr. Xiaokun Yang for their availability and patience as my committee members, and for their valuable and insightful advice on the research.

Thanks also to all friends, colleagues, the department faculty and staff for making my time at the University of Houston – Clear Lake an invaluable experience. Finally, thanks to my father, mother, and all my family members for their love and encouragement.

ABSTRACT

DRONE BASED OBJECT TRACKING WITH CAMSHIFT ALGORITHM AND NEURAL NETWORK

Xin Zhang
University of Houston-Clear Lake, 2021

Chair: Jiang Lu, PhD

Integration of tracking system and the drone has been a novel research topic in recent years, especially when drones are required to implement complex tasks which cannot be done easily with human control. Usually, the drone uses camera to gather full information about environments. The main processor calculates all necessary trajectories for drones. However, such tracking methodology does not apply to real-world problems mostly due to the complexity which surrounded the environment. For example, when a large number of persons gathered closely together, the tracking system is difficult to find people that you want to track down. In addition, the similar background color is also one of the main reasons the system can't track people. This thesis proposes an approach that combines camshift algorithm and neural network to track the object. This approach is more cost-efficient and environment-adaptive compare to the previous approaches which use traditional tracking algorithms. Our model altered the previous Yolo neural network, combined camshift algorithm, and neural network. Results show the new approach is faster than Yolo neural network. On the other hand, it solves the problems of occlusion, illumination, scale, and noise in camshift algorithm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:

INTRODUCTION

1.1 Background

In recent years, Drones have been one of the most popular technology products in the market. As of January 2021, there were 1,782,479 drones registered in the United State by the Federal Aviation Administration (FAA) [1]. According to market research from Gartner, the commercial drone is the significant revenue opportunity, which Gartner forecasts grew to $3.69 billion with 170,000 million units sold by 2017 [2]. Drones have been found in many different fields such as transportation, agriculture and security. Among these applications, tracking object is very important for surveillance and security in the city and suburbs. It can give security departments a new way to track suspects. Drone tracking system also can help to find people lost in the woods. When people are lost in forests, search and rescue experts use drone to fly over the area where they are most likely to found. The drone uses AI technology to detect people and track them [4].

Boulder surveillance is a serious problem in the United States. The government spends a lot of money for bolder security each year. Because the complex environment and 1,954 mi continental border between Mexico and USA. The boulder surveillance face enormous challenge. The drone can easily solve this problem due to its cheap price and flexible characteristics. It can easy to go where people can't arrive and track down suspicious people.

To track endangered species like Cheetah, African Wild Dog, Rhino or Leopard, various forms of tracking collars are used. These include radio, GPS and satellite collars. This equipment makes it possible for Wildlife ACT’s monitors to track these animals daily, which means that if they are injured, sick, trapped in a poacher’s snare, or have

escaped out of a reserve, help is not far away. The drone will become a better monitor device for wild animals. It can send real-time information to monitors the movements of the animals and their exact location. It also can result in poor knowledge of a population and its demographics and ultimately poor population management.

The drone is also a flexible and convenient tool for military assistants. Worldwide, military drone research and development is expected to increase from $3.2 billion in 2020 to $4 billion in 2029. Procurement funding is projected to reach $10.3 billion during the same timeframe. One of the most important reasons the military uses drones is for surveillance for gathering intelligence. Aircraft like the Global Hawk and the newer Ultra LEAP — which has clocked 18,000 combat flight hours — are critical to surveillance missions run by military branches.

Most of the applications require real time processing. Thus, to improve the speed and accuracy of the tracking system, we present the object tracking system that combines Camshift algorithm and neural network based on Raspberry Pi, Pi camera.

The key technique in the system is Camshift algorithm and CNN model. They are important technologies in computer vision for real-time object tracking. It has yielded many applications which include surveillance systems, research, and rescue system, etc.

In the field of target tracking, Camshift algorithm because of the small amount of calculation, ease to operation, high real-time performance, and the advantage of an adaptive change of the target window size and received extensive attention from the researchers.

The disadvantage of Camshift algorithm is that it is based on the target probability of a single color histogram, so when the target and the background color are the same or similar, the histogram cannot accurately reflect the characteristics of the target, pixel

probability value cannot be accurately calculated by express. Which can easily lead to track into local optimum.

The advantage of YOLOv3 is that predictions (object locations and classes) are made from one single network. It can be trained end-to-end to improve accuracy. YOLOv3 is more generalized. It outperforms other methods when generalizing from natural images to other domains like artwork. Region proposal methods limit the classifier to the specific region. YOLOv3 accesses the whole image in predicting boundaries. With the additional context, YOLOv3 demonstrates fewer false positives in background areas. YOLOv3 detects one object per grid cell. It enforces spatial diversity in making predictions. The disadvantage of YOLOv3 is that because it uses a deep neural network, it is difficult to run mobile devices.

The camshift algorithm with CNN model can be implemented in some other mobile devices, such as mobile phones. Because the simple Camshift algorithm and the proposed CNN model have fewer parameters, it is suitable for devices with limited memory.

1.2 Challenges

Challenges include the design of a deep neural network, a real-time tracking algorithm, and the interference protection measure. The number of layers and the choice of a dataset are very important in designing a deep neural network. The system needs less time to track the person.

How to solve the interference of the environment is also a challenge when we design the system. Some threshold values need a lot of time to select. The other challenges contain:

- improving the speed of the system;
- developing construction of deep neural network;
- Solving the problem of occlusion in Camshift algorithm

- Solving the problem of illumination in the picture

1.3 Contributions

We proposed an object tracking system with camshaft and neural networks. There are four contributions to the proposed system. Firstly, the proposed system improves camshift algorithm. Comparing the previous histogram and current histogram of the target, we can solve the background noise problem that camshift algorithm cannot fix. Second, a light CNN model is developed. It includes fewer parameters and layers than the deep CNN model. Third, the system solves the problem of target overlap and occlusion. When two targets overlap and the target is occluded by other obstacles, the system still can track the target. Lastly, the system can track multiple objects in real time at the same time.

The neural network, besides implemented in the system, can be used in some other mobile devices because it has a small number of parameters and it needs less time to extract object features.

1.4 Organization

The rest of the thesis is organized as follows: Chapter II is the related work. It contains steps of person recognition, the present person detection algorithms, the algorithm of camshift, the neural network of person tracking. Chapter III is tracking of targets in mobile robots based on Camshift Algorithm. It research about Camshift algorithm in tracking system in mobile robots. Chapter IV is high performance computer (HPC) on a deep neural network. It explores the performance of the deep neural network in HPC. Chapter V describes system configuration and setup processes. It includes hardware setup and main python libraries used in the system. Chapter VI presents the methods and results of designing the system. It contains dataset selection and collection, image preprocessing, CNN models of feature extraction, design of tracking algorithm,

person classification on still images and person tracking on real-time videos. Chapter VI presents simulation results. Chapter VIII summarizes the thesis.

CHAPTER II:

RELATED WORK

2.1 Object Detection

Object detection is one of the most famous and extensively researched topics in the field of Image Vision. Object detection determines the classes to which it belongs in the image. It can identify the classes such as person, car, and animals from the image. To recognize different objects, we need to extract visual features which can provide a semantic and robust representation. Traditional feature extraction methods include SIFT, HoG, and Harr-like feature descriptors.

Scale-Invariant Feature Transform (SIFT) algorithm extracts Scale-Invariant Keypoints to find distinctive image features. It identifies potential interest points that are invariant to scale and orientation by difference-of-Gaussian function (DoG). Therefore, keypoint locations can be selected from these interest points. Each keypoint location assigns one or more orientations based on local image gradient directions. SIFT can transform image data into orientation, scale, and location to local features [5]. The advantage of SIFT is that it extracts distinctive invariant features which can be correctly matched against a large database of features, providing a basis for object and scene recognition. The extracting features are invariant to image scale and rotation. Therefore, the extracting features can resist distortion and noise. The disadvantages of SIFT are that it is quite slow and is not effective for low powered devices.

Histograms of oriented gradient (HOG) algorithm is an excellent descriptor for human detection. It evaluating well-normalized local histograms of image gradient orientations in a dense grid. HOG uses the distribution of local intensity gradients to characterize local object shape. It divides the image into some small spatial regions. Each region accumulates a histogram of gradient directions or edge orientations through the

pixels of the cell. The larger spatial regions can be normalized by accumulating a measure of local histogram energy. Finally, the combined vectors are fed to a linear SVM for object classification [6]. The advantage of HoG is that it can capture edge or gradient structure that is very characteristic of local shape and it also can control local geometric and photometric transformations. The disadvantage is HoG is that it takes more time to extract features because the final vectors grow larger.

Harr-like feature algorithm is a boosted cascade of simple feature classifiers. Haar features are a sequence of rescaled square shape functions proposed by Alfred Haar in 1909. They are similar to convolution kernels in the Convolution Neural Network (CNN). They are series of features used to identify an object in an image. Using sliding windows and a number of haar features, finally leading to detect an object or not. Depending upon the sliding windows size and object location, the number of features, the object can be detected at a certain stage [7]. The advantage of Harr-like feature is that it has a relatively high level of precision and recall when detecting objects in images and requires only a few attempts to get a working model for object detection. The disadvantage of Harr-like feature is that it needs to take much longer to be processed than a convolutional neural network.

Thanks to the emergency of Deep Neural Networks (DNNs), they are changing the world we live. DNNs are similar to how the nerve system is structured where each neuron connected the other and passing information. It has a remarkable ability to derive meaning from complicated or imprecise data. A trained neural network can solve complex real-world problems, such as image detection and tracking.

Image detection is the most basic application of the convolutional neural network (CNN) technology [8]. The CNN allows the computer to operate in a self-learning mode to classify multiple objects of an image, without being explicitly programmed. A CNN

takes a picture that includes multiple objects, extracts features from the picture through different kinds of layers, and predicts the probability of the classes. A basic CNN architecture has an input layer, convolutional layers, Relu layers, pooling layers, and a fully connected layer. Figure 2.1 shows a basic CNN architecture.

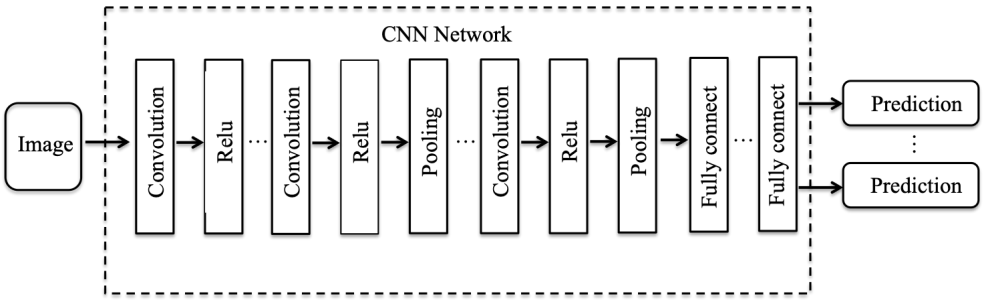

Figure 2.1: A basic CNN architecture

The input layer reads an array of pixels from an input image. For example, if the size of an image is 256×256×3. Where the first 256 is width, the second 256 is height, and 3 is RGB channel values. Therefore, this image has a total of 196,608 pixels. The value of each pixel has a number from 0 to 255 which is the intensity of each pixel.

The convolutional layer is the key technology of CNN. When the matrix with pixel values entered into the convolutional layer, the network began reading pixel values from the top left of the matrix. The network selected a small filter that moves along the matrix and operates these pixel values in the filter. The filter multiplies its value by the original pixel values, and then sums up all these multiplications. After passing the filter across all positions of the matrix, the network obtained a new matrix that is smaller than the input matrix. The Relu layer is added after the convolutional layer. The Relu layer has a Relu activation function that helps to decide if the neuron would work or not. ReLU function is the most widely used activation function in CNN. It converts all negative

inputs to zero and the neuron does not get activated. Therefore, the Relu function generates the variable to decide a class label.

The pooling layer follows the Relu layer. The target of the pooling layer is to reduce the number of parameters and computation in the network. Because some features have been identified in the previous convolutional layer, the pooling layer compressed these features for controlling overfitting.

The fully connected layer is the last layer in CNN. It takes the output information from convolutional networks, flattens them, and turns them into a single vector. It reaches a classification decision for the correct label.

VGG network is a very deep CNN for large-scale image recognition. VGG network used only 3×3 filter in the first layer network. The advantage of a small filter is to reduce the number of parameters and allows VGG to have a large number of weight layers. Reducing the number of parameters decreases the complexity of the network. So When VGG network has very deep layers, it still can handle overfitting. The convolutional layers in VGG are followed by a Relu unit. VGG has three fully-connected layers: the first two have 4096 channels each and the third has 1000 channels, 1 for each class. Based on the depth of network, the family of VGG networks includes VGG11, VGG13, VGG16, VGG19. Figure 2.2 shows a basic VGG network architecture [9].

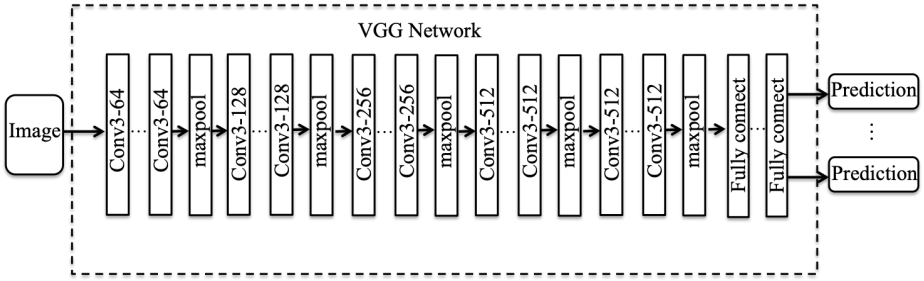

Figure 2.2: A basic VGG network architecture

ResNet network can build a very deep CNN that is over a hundred layers by learning the residual representation functions. ResNet network solves the vanishing gradient problems when the CNN architecture is going deeper and deeper. The key technology of ResNet network is called "identity shortcut connection" which skips one or more layers. The shortcut connection is added from the input value to the output value after few convolutional layers. Resnet network uses the zero padding and a linear projection can handle the problem that input and output have a different size at the shortcut connection. ResNet network includes many residual blocks. Each residual block has a shortcut connection and several convolutional layers. ResNet network consists of ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152. Figure 2.3 shows a basic ResNet network architecture [10].

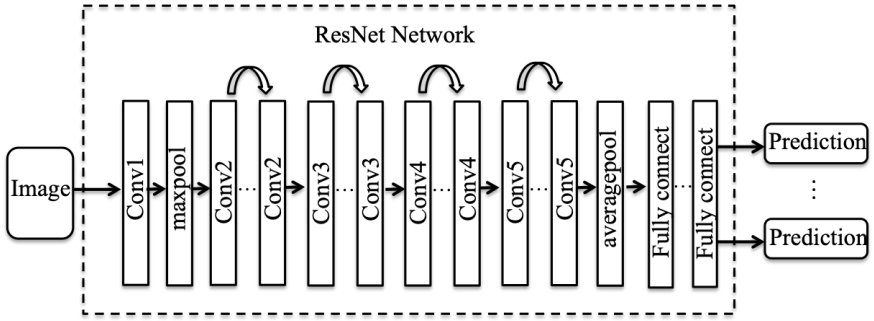

Figure 2.3: A basic ResNet network architecture

The inception network is a depth wise separable convolutional network by Google. The inception network is also a pretty deep network that is subject to the vanishing gradient problem. The inception network includes many inception modules. Each inception module uses a different size filter to capture different scale information on parallel convolutional layers. For example, a small size filter focus on detail like dense contours, and large size filter benefits for processing coarse outlines. Then all outputs from these parallel convolutional layers are concatenated together and sent to a fusion

layer. The concatenated features in the fusion layer are fed to the next inception module as the input. The inception network includes inception v1, inception v2, inception v3, and inception v4. Figure 2.4 shows a basic Inception network architecture [11].

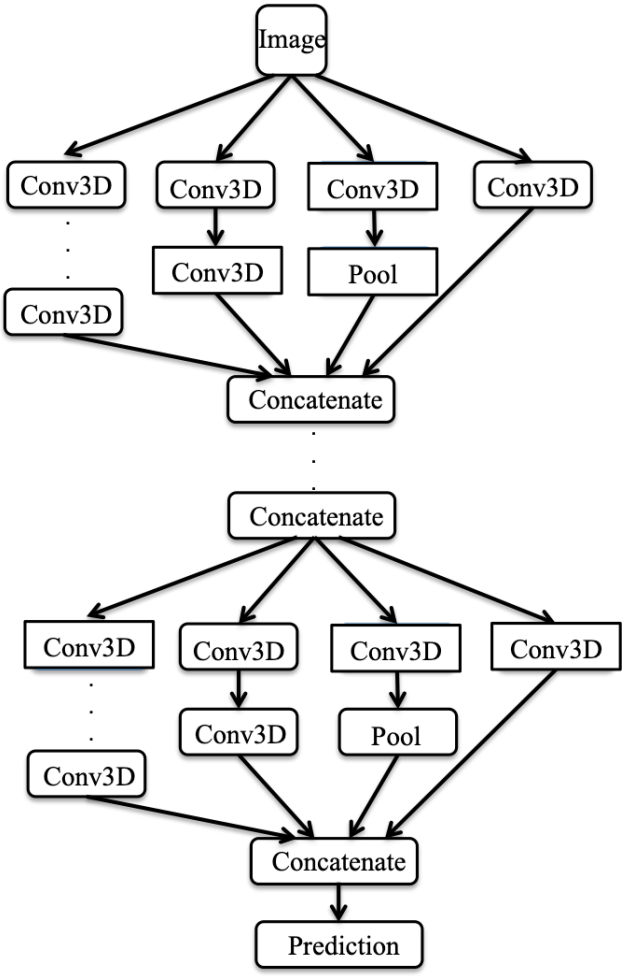

Figure 2.4: A basic Inception network architecture

Compared to previous traditional algorithms and convolutional neural network (CNN), the speed of CNN is relatively faster than the traditional algorithm, and it is suitable to use in a low power device. Its requirement for computing is not very high. In the thesis, a tracking system is designed based on Raspberry Pi, Pi camera, and screen.

2.2 Object Tracking

2.2.1 Traditional Algorithm

Point tacking includes the Kalman filter. Kalman filter, also known as linear quadratic estimation (LQE), is an optimal recursive data processing algorithm with a linear model and Gaussian probability distribution. Kalman filter is a typical point tracking method. It has been used for tracking in inter-active computer graphics. Kalman filter includes two groups of mathematical equations. One of the groups is time update equations and another one is measurement update equations. Time update equations is a prediction step that is responsible for projecting forward the current stateand error covariance estimates to obtain an estimate for the next time step. Kalman filter assumes that all of the variables are random and Gaussian distributed. Therefore, every variable has a mean value which is the center of the random distribution, and a variance which is the uncertainty. Measurement update equations is a correction step that takes the measurement update where the correction to the estimate of the current state is calculated. After each time and measurement update, the Kalman filter repeats this process with the previous estimates used to project or predict the new estimate. Kalman filter is a single point tracking object measurement at each instant time. If the Kalman filter solves the problem of tracking multiple objects, it needs to solve associated measurements for multi-object data [12] [13] [14] [15].

Kernel tacking includes particle filter, KLT features tracking, and meanshift or camshift algorithm. Particle filter is a non-linear or non-Gaussian method with Monte Carlo and recursive Bayesian estimation. Particle filter has improved performance in the non-linear or non-Gaussian environment over Kalman filter. It has been used for tracking multiple objects, robotics, and navigation. It shows density distribution using random sampling particles. The working mechanism of particle filter is that the filter partition the

state space into many parts and measure particles probability. If probability is high, particles are concentrated. The particle filter includes three major steps: selection, prediction, and measurement. The selection step picks up the particles with the highest probability among the previous particle set and then generates a new particle set. The prediction step uses a dynamic model to evaluate how the state of the object will change. The measurement step reevaluated the particle's weight by new data. A particle filter is an astochastic tracking algorithm. It uses multiple discrete particles to represent the location of an object. It combines particles at a position into a single particle and generates weight for this particle to reflect the number of particles that were combined to form it. Figure 2.5 shows the procedure of particle filter [16].

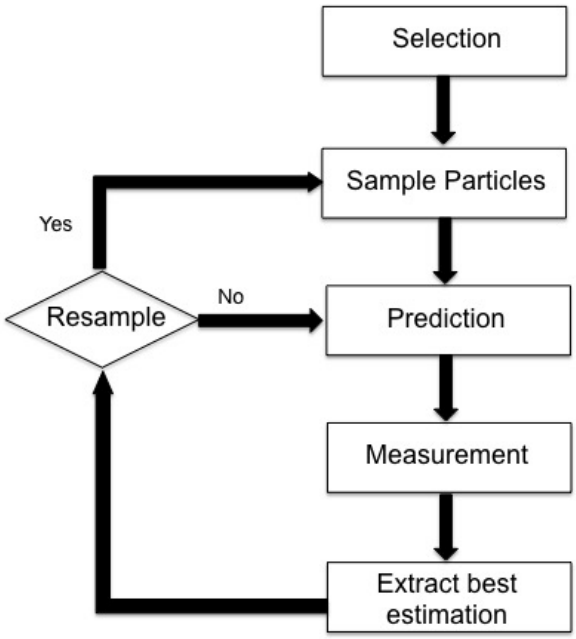

Figure 2.5: The procedure of particle filter

Kanade-Lucas-Tomasi (KLT) feature tracker is an approach to feature extraction. The main purpose of KLT feature tracker deals with the problem of traditional image registration techniques. KLT uses spatial intensity information to search the best match

position. It is faster than other traditional tracking techniques [17]. KLT minimizes a sum of squared differences (SSD) function between two image frames through a set of parameters to register or align the images. Because SSD can converge quickly, the gradient is easy to derive [18]. KLT tracking includes two steps. The first step is called feature detection that it locates the trackable features in the initial frame. In this step, KLT focuses on finding the best features to track in an image. The linear intensity of image texture is provided by image gradients. Texture includes trackable features. The features can be defined by the texture within a finite size window. The window size determines the number of features detected. The second step is called feature tracking that it tracks each one of the detected features in the rest of the frame by means of its displacement. In this step, the location of the object is determined in the scene by means of its detected features [19].

Meanshift is a non-parametric feature space analysis technique that locates extremum in the distribution of data probability density. It can estimate gradient function in data probability density. Meanshift shows an iterative process about a shift of the mean vector. First, it computes the shift of the current data means. And then, it uses the direction of shift mean to move current data. At last, it changes moved data into current data, and process this procedure iteratively. When meanshift algorithm converges these processes, it reaches target tracking. Camshiftis called Continuously Adaptive Meanshift. It is an improved method of meanshift. Camshift uses meanshift method to track the target, and measure the distance between the model and the current object based on the Bhattacharyya parameter. Camshift can adjust the window size of the tracking object automatically to fit the object area when the size of the tracking object is reflected by any variation in the distance between the camera and the object. Because the windowed

distribution gradient climbing of camshift can ignore outlines, it can ignore other nearby noise. Figure 2.6 shows the procedure of Meanshift [20].

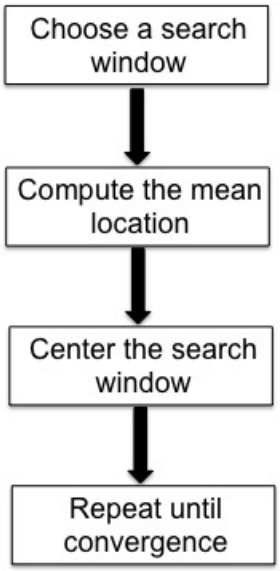

Figure 2.6: The procedure of Meanshift

Silhouettes tracking can match the shape or contour of the objects to track objects. It estimates the object region in each frame. The object region includes information that can be in the form of appearance density and shape models which are usually in the form of edge maps [21].

Table 2.1 shows the advantages and disadvantages of several tracking algorithms. Comparing these tracking algorithms, point tracking is not suitable for tracking in mobile robots because it can only track a point. Silhouette tracking also cannot be applied for tracking in mobile robots because it is not accurate for shape recognition. Therefore, kernel tracking is the best tracking type with mobile robots. And then, Camshift tracking technology is the best algorithm in all of the algorithms in the kernel tracking area. So this paper suggests Camshift algorithm as an optimal choice.

Table 2.1: Comparison of video tracking techniques

Type of Tracking	Algorithm	Advantages	Disadvantages
Point tracking	Kalman filter	Track points in noisy images	Distributed by Gaussian
Kernel Tracking	Particle filter	Dealing with non-Gaussian noise	Number of particles increases with increasing model dimension
Kernel Tracking	KLT	It is faster than traditional techniques	KLT is less reliable than Feature trackers
Kernel Tracking	Meanshift and Camshift	Suitable for real data analysis	The selection of a window size is not trivial
Silhouette tracking	State space models	Can handle complex models for rigid and non-rigid objects	Shape recognition is not so accurate

2.2.2 Method Based on Deep Convolutional Network

Image classification is the more advanced application of the convolutional neural network (CNN) technology than image classification. The image detection network is divided into two parts: the backbone network determines the classification of objects and the object detector network determines the location of objects. The backbone network is a deep neural network that extracts the basic features for object detection. The object detector network is a single deep neural network that predicts the bounding boxes and the class probabilities for all detecting objects. Generally, the object features are extracted from the input image using the backbone network at first. Then the output layers of the backbone network connect the object detector network for the classification of the extracted features. The output of the object detector network classifies N+1 predictions, where N is the number of classes, and 1 is for the background. The output also provides 4 coordinates predictions of each bounding box. The common image detection network architectures include SSD, R-CNN, and YOLO. The paper chose these three kinds of

common image detection architectures to test the performance of the effect of HPC. Figure 2.7 shows the architecture of a general object detection network.

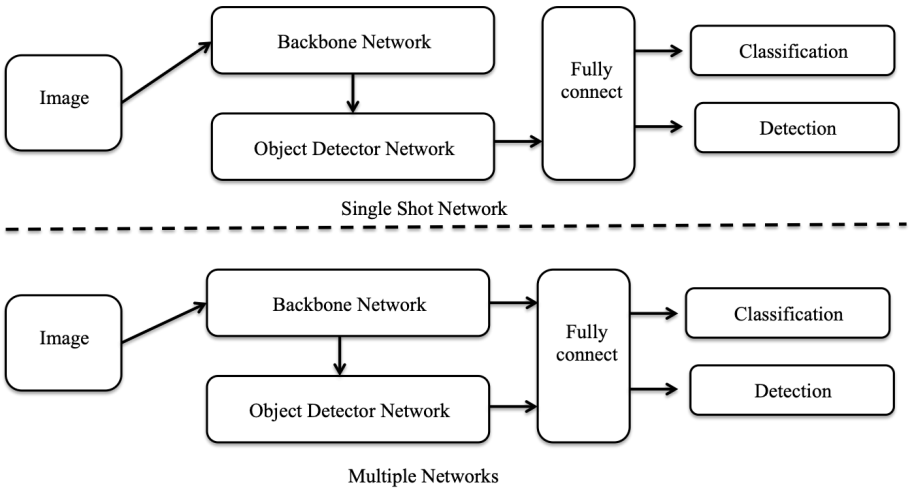

Figure 2.7: The architecture of a general object detection network

SSD network is a single shot mutibox detector for object detection in real-time. SSD network uses VGG16 network as the backbone network. VGG16 discards the fully connected layers and adds 6 auxiliary convolutional layers as the object detector network. Therefore, the SSD network can use multiple layers to detect objects independently. The auxiliary convolutional layers are for object detection. Multi-scale feature maps can improve accuracy significantly. SSD network associates a set of default bounding boxes with each auxiliary convolutional layer. The default bounding boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. Object detector network predicts 4 offsets relative to the original default box shape depend on feature map cells. Figure 2.8 shows SSD network architecture [22].

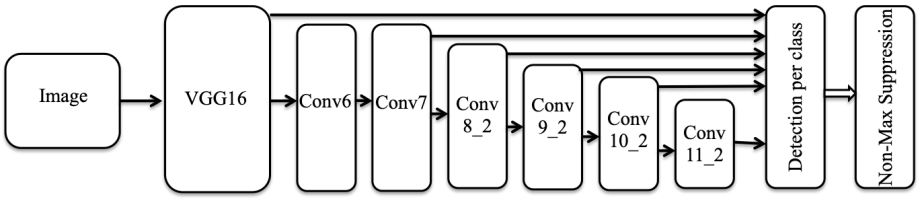

Figure 2.8: SSD network architecture

The fast R-CNN network is designed to tackle object detection problems. Fast R-CNN network includes backbone network, region proposal network and full connect network. The backbone network extracts the features for object classification. The region proposal network is similar to the backbone network. It combines the features and forms a fixed-length feature vector in the RoI pooling layer. Each of the feature vectors consists of a classification module and a localization module. The classification module classifies object classes. The localization modules output four locations for each object class. The full connect network connects RoI pooling layer for the classification and localization of objects. Figure 2.9 shows Fast R-CNN network architecture [23].

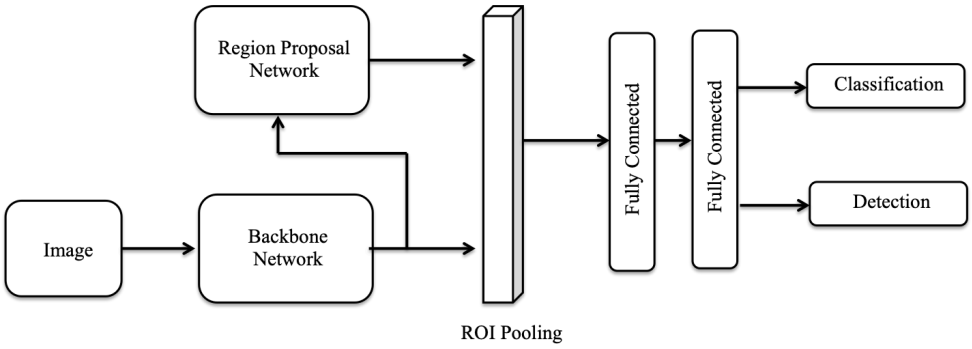

Figure 2.9: Fast R-CNN network architecture

YOLO network is one of the faster CNN networks for object detection. Although it is not the most accurate object detection network, it is a good choice for real-time detection without loss of too much accuracy. Like the SSD network, the YOLO network also uses a deep CNN network as the backbone network for feature extraction and an

FPN network for object detection. YOLO network algorithm splits an input image into m×m grid cells. Each grid cell predicts whether the center of the object falls into the grid cell. YOLO networks include YOLO v1, YOLO v2, YOLO v3, YOLO v4, and YOLO v5. Figure 2.10 shows YOLO network architecture [24].

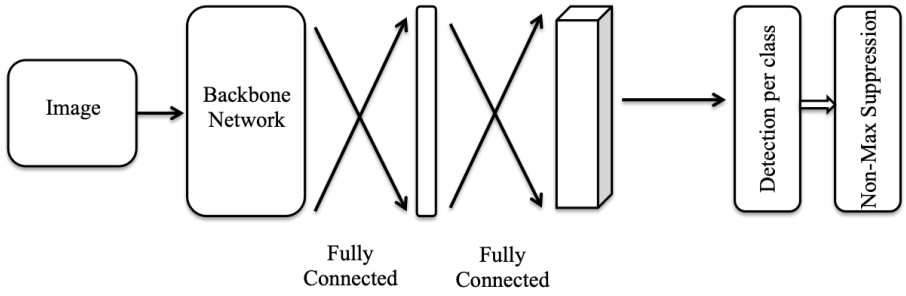

Figure 2.10: YOLO network architecture

CHAPTER III:
TRACKING OF TARGET IN MOBILE ROBOTS BASED ON CAMSHIFT ALGORITHM

3.1 Introduction

Vision is a very important sensing system for humans to obtain outside information. People achieve more than 80 percent of outside information by vision. Therefore, a lot of applications for computer vision are developed swiftly. These applications can make computers recognize, analyze, and understand outside information, just as the human vision system can do for humans. Visual tracking is a core application of computer vision. It can detect, extract, recognize and track the moving objects in an image sequence. Today, visual tracking technology has been applied in many areas such as radar guidance, video compression, medical diagnosis, Robots aerospace, education and entertainment. Intelligent video surveillance is an important visual tracking application. It can realize target detection, recognition, and tracking in video. Video compression is another important application in visual tracking. It applies target tracking technology to the encoding method because encoding computation spends on two aspects which are block matching and filtering. An intelligent traffic system is also a very popular visual tracking application because it needs target detection and visual tracking to track detected vehicles. In addition, tracking of pedestrians in cross-road can help the vehicle move safely on the crossroad. The visual tracking can be classified as a single scene and multi-scenes. Single scene tracks one designed target in one video. Multi-scenes track every target continuously in the network by fusing all monitors. Visual tracking single target tracking and multi-target tracking by a number of moving targets. Visual tracking is classified as tracking with a static camera and moving camera. The visual tracking method can be based on geometrical similarity, local features, contour

profile, and forecast. Tracking method is based on target geometrical similarity assumes that the target can be expressed by the simple geometrical form which is stored as a target template. The tracking method is based on local feature extracts local features of the target, and recognizes and tracks target in image sequence by fused local features. Tracking method is based on contour profile needs a general position of targets, and recursion by the differential equation is solved to converge contour profile to the local minimum value. The tracking method based on the forecast is to reduce and express the posterior probability of target mode accurately effectively by existing data. [29] [30]

One of the main goals of robot vision is to enable the robot to realize the basic function of human vision, such as motion perception of the objects. To finish this goal, the mobile robotic use visual tracking technology in robot vision. The pivotal content of visual tracking is the motion state of a target object in each frame of a sequence of images. A typical visual tracking system includes four components: object initialization, appearance modeling, motion estimation, and object localization. Firstly, object initialization can draw an object location with a bounding box by the people or object detectors. Secondly, appearance modeling is composed of visual representation and statistical modeling. Visual representation uses different types of visual features to construct object descriptors. Statistical modeling uses statistical learning techniques to build math models. Thirdly, motion estimation predicts a dynamic motion state by predictors. Lastly, object localization makes a decision based on motion estimation. Visual tracking application is a fundamental application in mobile robots. The object of target tracking control for a mobile robot is to keep constant distance and angle with a specified target when it moves in the environment. The position of the target should be obtained as the input of the algorithm in target tracking. The tracking objects include three major methods: point tracking, kernel tracking, and silhouette tracking. Point

tracking tracks the target as a point. Kernel tracking tracks the target as the kernel which represents the shape of the targets. Silhouette tracking tracks the target as contour or shape matching. [31] [32] [33] [34]

In this paper, several popular visual tracking algorithms will be presented. Some of them are based on point tracking. Some of them are based on kernel tracking. Comparing these algorithms, kernel tracking is suited to be applied to mobile robots. Because Camshift is the best algorithm of all kernel tracking algorithms, Camshift is chosen in the research.

3.2 System Architecture

3.2.1 Hardware Structure

The target tracking system in this paper includes mainly the following parts: video camera, signal processing model, monitor, and control platform. The video camera obtains the video signal from the external environment. The signal processing model process the video signal and sends the command to the control platform. The control platform controls the video camera to track the target. The monitor can observe the location of the tracking target. The structure chart of a typical target tracking system is shown in Figure 3.1. The target tracking system obtained the target image from a video camera, signal processing model obtains the target location from the target image and predict the target location in the next frame. The signal processing model sent the control signal to the control platform. The control platform controls the direction of the video camera depend on the control signal [35].

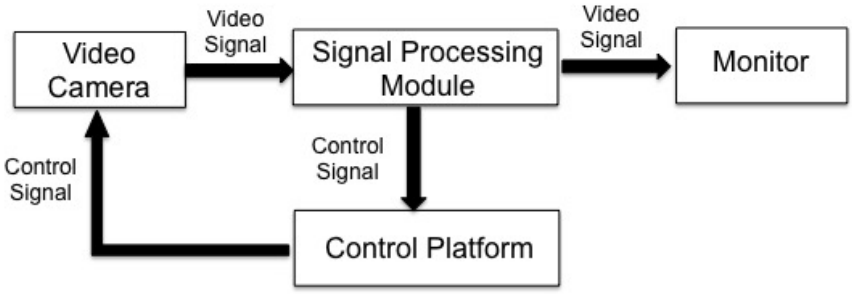

Figure 3.1: The target tracking system structure chart.

3.2.2 Software Structure

The structure of the software is shown in Figure 3.2. The software includes image tracking and image prediction. Image tracking identifies the location of the target in the current image. Image prediction predicts the location of the target in the next image through the sequence image.

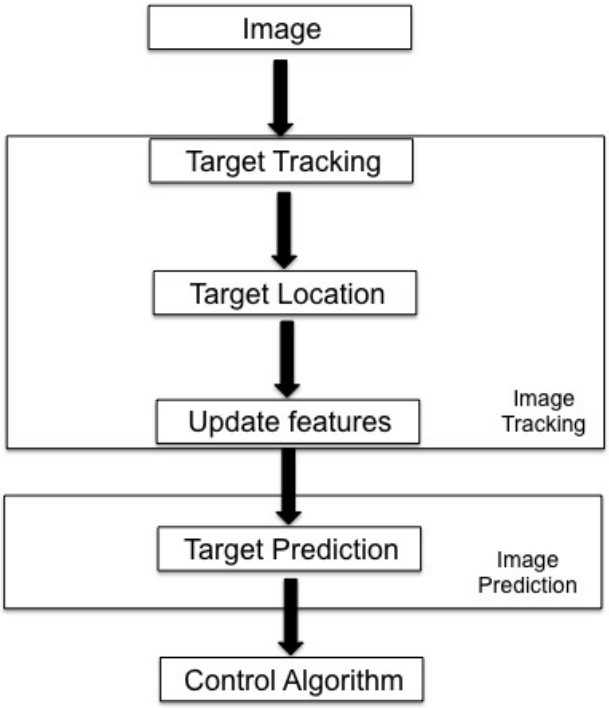

Figure 3.2: The structure of the target tracking system software

3.3 Problem Solution

3.3.1 Illumination Variations

There is a serious problem block the development of target tracking in mobile robots with camshift algorithm. When the targets are tracked, they have lower performance under intense illumination variation. Camshift algorithm uses color histogram as its target model. When the target and background are alike in color, the tracking result may not satisfied. In addition, camshift algorithm is sensitive to the illumination environment. An improved Camshift algorithm can solve this problem. HSV color model includes Hue value (H value), saturation value (Svalue) and brightness value (V value). Traditional camshift only computes H value. S and V values are set thresholds. When S and V values are below the threshold, the pixels are effective. The new algorithm adjusts the thresholds of S and V values adaptively against the environment changes. When the environment has a strong illumination, the algorithm ignores the pixels with high S and V values, and when the environment has weak illumination, the algorithm ignores the pixels with low S and V values. When the ratio of back projection value between the whole image and the search window reaches the minimum, the adjustment finishes [36].

3.3.2 Fast Moving Objects

Another serious problem is that camshift algorithm is difficult to track a high speed or speed-changing object. Unlike the Kalman filter object tracking, which estimates the velocity change of the object, the conventional camshift uses a fixed moving distance to search the neighbors of the current object center. In order to solve this problem, the concept of the central tendency is proposed. This concept is based on the relationship between the former center and the current center. The current center adds a

movement weight for determining the modified center. The movement weight depends on the distance between the former center and the current center.

3.4 Experiments and Results

The experiment uses the computer to simulate the tracking system in the mobile robots with different scenarios, such as object size changing, illumination variation, and fast-moving target. The tracking system is compiled by python language.

The program uses OpenCV library. Figure 3.3 shows the tracking of an object in the tracking system. Through these two images, the tracking window can change with the shape of the object. The object is colored before the object is tracked. Figure 3.4 shows the occlusion of the object. When the object of tracking is occluded by another object, the tracking window is not affected by occlusion in the camshift algorithm. From this experiment, the camshift algorithm makes the application of the object tracking system in mobile robots obtain the best effect. When the background is light and the object is occluded by another object, the object is still tacked by the tracking window. With the improvements, camshift algorithm can be easily implemented for tracking in mobile robots.

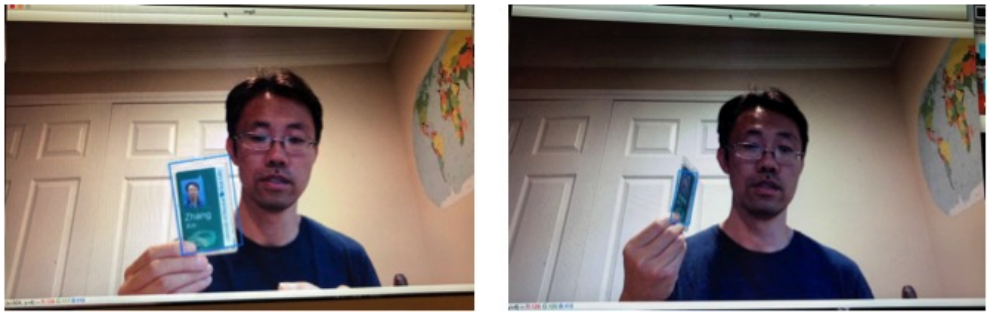

Figure 3.3: The target tracking result.

Figure 3.4: The Occlusion tracking result

3.5 Conclusion and Future Work

In this paper, robustness and real-time target tracking system of the mobile robot is presented. Hardware and software architectures of the target tracking system are designed. Comparing several popular tracking algorithms in all of target tracking applications, a novel flexible Camshift tracking algorithm has been proposed, which has good robustness to target variation, partial occlusion, and fast object tracking is designed and implemented. So far, the target tracking system are mainly applied to track the single target. In the future, the application of multi-targets tracking will be considered. It can use multiple cameras in the same background to recognize multiple tracking targets and use a different camera to detect and track the different targets. In addition, some new target tracking technologies will be applied to the target tracking system. For example, it will consider using an artificial neural network to predict the moving direction of the target.

CHAPTER IV:
HIGH PERFORMANCE COMPUTER ON DEEP NEURAL NETWORK

4.1 Introduction

Nowadays, CNNs have become the most important tool in image processing applications. Many very deep CNNs have already been designed. However, to implement CNNs for image processing applications, computing resource is an extreme challenge. In the past 20 years, the computational capability and speed of computers are two of the most important parameters in computer development. So far, many advanced technologies allow computers to run faster with stronger computing capabilities such as CPU with multiple cores, GPU, and computer clusters. They could change the game in such fields as computer vision, artificial intelligence (AI), cryptography, chemistry, biology, and physics. There are several popular computation resources for computers nowadays as follow:

CPU is called a central processing unit and it is often used to perform arithmetic and logic computations and acts as the brain of the computer. Modern CPUs offer multiple cores. Nowadays, CPUs can have around 2 to 18 cores. The CPU with multiple cores speeds up the system because the computer can do multiple things at once. There are some excellent multi-core processors from 2017 to 2019. For example, the Intel Core i9-7900X processor and AMD Ryzen Thread ripper processor are the two most advanced multi-core processors in their product lines, respectively [37]. The computers can process complex calculations depend on Multi-cores CPU. Besides, Multi-cores CPU supports many consumer software with multithreading. GPU is known as a graphical processing unit and it is used to operate digital images as a graphical device. GPUs start at a couple of hundred cores and can have up to several thousand. The memory size of GPUs is suitable for huge amounts of data computations in deep learning. GPUs have perfect

achievement in practical applications. For example, eBay's maxDNN has already shown that all of the programs can achieve excellent performance with very high GPU utilization [38]. Google Colab is a free cloud service that can provide convenient CPU and GPU resources for programmers. Google Colab is essentially a Jupyter Notebook within the web. The programmers can write programs and run them using a remote CPU or GPU in Google Colab. Google Colab has installed many deep learning libraries such as PyTorch, Keras, TensorFlow, and OpenCV. It's very convenient for image processing applications.

HPC is an indispensable tool. An HPC system is described by numerous processors, heaps of memory, fast systems administration, and expansive information stores. The HPC is intended to utilize parallel computing to apply more processor force for the solution of a problem. Therefore, An HPC cluster is comprised of numerous nodes. The nodes include computing nodes and master nodes. Most of the nodes are compute nodes. A compute node performs one or more tasks based on the scheduling system. The master nodes observe the status of individual nodes and issue administrative orders. The Extreme Science and Engineering Discovery Environment (XSEDE) can provide high-performance computing resources for scholars and researchers. XSEDE is the most advanced and powerful collection of data resources and services in the world. The data resources include supercomputers, software, networks, and data storage. XSEDE has many partner institutions including the Pittsburgh Supercomputing Center (PSC) at the Carnegie Mellon University and the University of Pittsburgh, Texas Advanced Computing Center (TACC) at the University of Texas, Austin, San Diego Supercomputer Center (SDSC) at the University of California and so on [39]. Google cloud is another good provider for HPC. It is applied in large data, artificial intelligence, and other fields, gradually shifting from scientific research to commercialization [40].

For example, PayPal serves more than 300 million customers and developing online, mobile, and in-store services by HPC on google cloud.

In this paper, the three basic image processing applications, image classification, image detection, image segmentation, based on different CNN architectures are tested utilizing several different computing systems. The first image processing application is image classification. Three popular deep neural networks, VGG16, ResNet50, and Inception v3 are chosen. The second image processing application is image object tracking. The three classical CNN architectures, SSD, Fast R-CNN, and Yolov3 are chosen. The third image processing application is image segmentation. The three popular network models are picked. They are U-net, Mask R-CNN, and PANet.

The rest of the paper is organized as follows. Section 2 briefly reviews related background. Section 3 details deep neural network structure methods and algorithms of the three image processing applications. The results are provided in Section 4. Finally, Section 5 concludes the paper.

4.2 Related Work

Image classification is the most common application of a convolutional neural network. There are many popular CNNs for image classification and detection. A. Krizhevsky et al. [41] proposed a large, deep convolutional neural network called AlexNet for image classification and detection. K. Simonyan et al. [42] proposed the Very Deep Convolutional Networks called VGG for in the large-scale image recognition setting. C. Szegedy et al. [43] proposed a deep convolutional neural network architecture is called GoogleNet, which was responsible for setting the new state-of-the-art for classification and detection. K. He et al. [44] proposed a residual learning framework for image recognition [45]. F. Chollet [46] proposed a novel deep convolutional neural network architecture is called Xception for a larger image classification dataset.

Image detection is another important application of a convolutional neural network. So far, more and more CNN architectures support systems to track objects from an image or video. J. Redmon et al. [47] proposed a new approach is called YOLO to spatially separated bounding boxes and associated class probabilities. W. Liu et al. [48] proposed a Single Shot MultiBox Detector method for detecting objects in images using a single deep neural network. R. Girshick et al. [49] proposed an R-CNN method where we use selective search to extract just 2000 regions from the image. S. Ren et al. [50] proposed a state-of-the-art object detection network is called Faster R-CNN depend on region proposal algorithms to hypothesize object locations. J. Redmon et al. [51] proposed an update method of YOLO is called YOLOv3 for image object tracking.

Image segmentation is applied to machine vision, medical imaging, and video surveillance, and so on. Until now, CNN is still one of the best technologies for image segmentation. J. Long et al. [52] proposed a Fully Convolutional Network (FCN) (containing only convolutional layers) trained end-to-end for image segmentation. G. Sharma et al. [53] proposed an end-to-end convolutional network which is called ParseNet predicting values for all the pixels at the same time for image segmentation. O. Ronneberger et al. [54] proposed a neural network called U-net composed in the contracting part and expanding part for biological microscopy image segmentation. T.-Y. Lin et al. [55] proposed a feature pyramid network (FPN) for object detection or image segmentation. H. Zhao et al. [56] proposed the Pyramid Scene Parsing Network (PSPNet) to better learn the global context representation of a scene. K. He et al. [57] proposed the Mask R-CNN model for object instance segmentation. L.-C. Chen et al. [58] proposed

the Deeplabv3+ framework using an encoder-decoder structure for image segmentation. S. Liu et al. [59] proposed the Path Aggregation Network (PANet) based on the Mask R-CNN and the FPN frameworks for image segmentation. H. Zhang et al. [60] proposed a Context Encoding Network (EncNet) capturing global information in an image to improve scene segmentation.

4.3 Related Work CNN Architectures for Image Processing Applications

4.3.1 CNN Network for Image Classification

Image classification is the most basic application of the convolutional neural network (CNN) technology [61]. The CNN allows the computer to operate in a self-learning mode to classify multiple objects of an image, without being explicitly programmed. A CNN takes a picture that includes multiple objects, extracts features from the picture through different kinds of layers, and predicts the probability of the classes. A basic CNN architecture has an input layer, convolutional layers, Relu layers, pooling layers, and a fully connected layer. Figure 4.1 shows a basic CNN architecture.

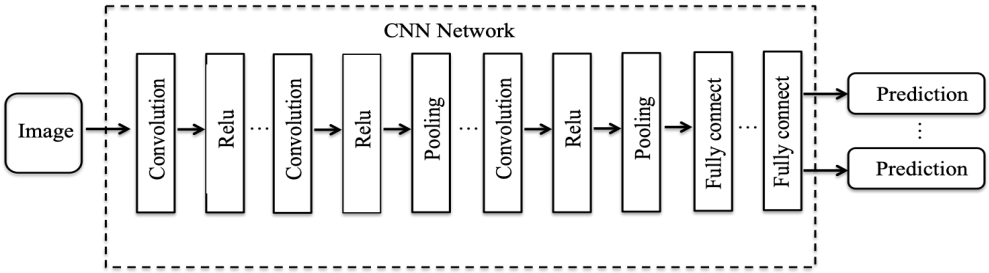

Figure 4.1: A basic CNN architecture

The input layer reads an array of pixels from an input image. For example, if the size of an image is 256×256×3. Where the first 256 is width, the second 256 is height,

and 3 is RGB channel values. Therefore, this image has a total of 196,608 pixels. The value of each pixel has a number from 0 to 255 which is the intensity of each pixel.

The convolutional layer is the key technology of CNN. When the matrix with pixel values entered into the convolutional layer, the network began reading pixel values from the top left of the matrix. The network selected a small filter that moves along the matrix and operates these pixel values in the filter. The filter multiplies its value by the original pixel values and then sums up all these multiplications. After passing the filter across all positions of the matrix, the network obtained a new matrix that is smaller than the input matrix.

The Relu layer is added after the convolutional layer. The Relu layer has a Relu activation function that helps to decide if the neuron would work or not. ReLU function is the most widely used activation function in CNN. It converts all negative inputs to zero and the neuron does not get activated. Therefore, the Relu function generates the variable to decide a class label.

The pooling layer follows the Relu layer. The target of the pooling layer is to reduce the number of parameters and computation in the network. Because some features have been identified in the previous convolutional layer, the pooling layer compressed these features for controlling overfitting.

The fully connected layer is the last layer in CNN. It takes the output information from convolutional networks, flattens them, and turns them into a single vector. It reaches a classification decision for the correct label.

This paper chose three kinds of common deep CNN architectures to test the performance of the effect of high performance computer. These architectures are VGG16, Resnet50, and Inception v3. Each architecture has its self-characteristics to be applied to different fields and hardware environments. For example, VGG is now one of the most

used image-recognition architectures. Resnet is one of the most popular architectures in various computer vision tasks. Inception is applied to mobile and embedded system environments.

VGG network is a very deep CNN for large-scale image recognition. VGG network used only a 3×3 filter in the first layer network. The advantage of a small filter is to reduce the number of parameters and allows VGG to have a large number of weight layers. Reducing the number of parameters decreases the complexity of the network. So When the VGG network has very deep layers, it still can handle overfitting. The convolutional layers in VGG are followed by a Relu unit. VGG has three fully-connected layers: the first two have 4096 channels each and the third has 1000 channels, 1 for each class. Based on the depth of the network, the family of VGG networks includes VGG11, VGG13, VGG16, VGG19. Figure 4.2 shows a basic VGG network architecture [62].

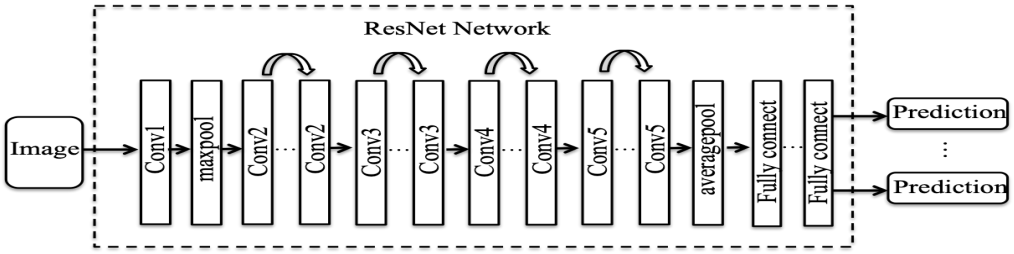

Figure 4.2: A basic VGG network architecture

ResNet network can build a very deep CNN that is over a hundred layers by learning the residual representation functions. ResNet network solves the vanishing gradient problems when the CNN architecture is going deeper and deeper. The key technology of ResNet network is called “identity shortcut connection” which skips one or more layers. The shortcut connection is added from the input value to the output value after few convolutional layers. Resnet network uses zero-padding and a linear projection

can handle the problem that input and output have a different size at the shortcut connection. ResNet network includes many residual blocks. Each residual block has a shortcut connection and several convolutional layers. ResNet network consists of ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152. Figure 4.3 shows a basic ResNet network architecture [63].

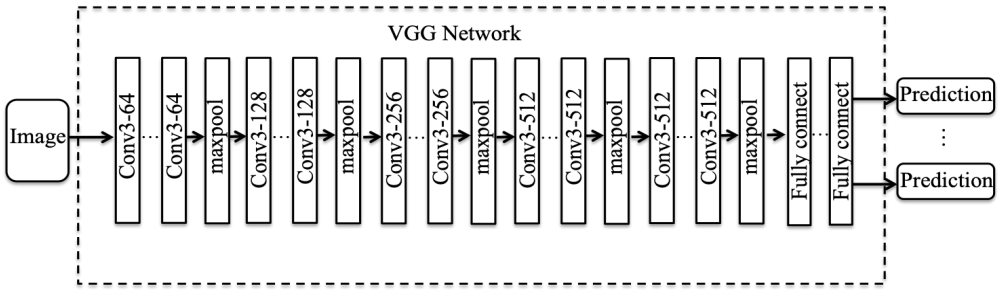

Figure 4.3: A basic ResNet network architecture

The inception network is a depthwise separable convolutional network by Google. The inception network is also a pretty deep network that is subject to the vanishing gradient problem. The inception network includes many inception modules. Each inception module uses a different size filter to capture different scale information on parallel convolutional layers. For example, a small-size filter focus on detail like dense contours, and large-size filter benefits for processing coarse outlines. Then all outputs from these parallel convolutional layers are concatenated together and sent to a fusion layer. The concatenated features in the fusion layer are fed to the next inception module as the input. The inception network includes inception v1, inception v2, inception v3, and inception v4. Figure 4.4 shows a basic Inception network architecture [64].

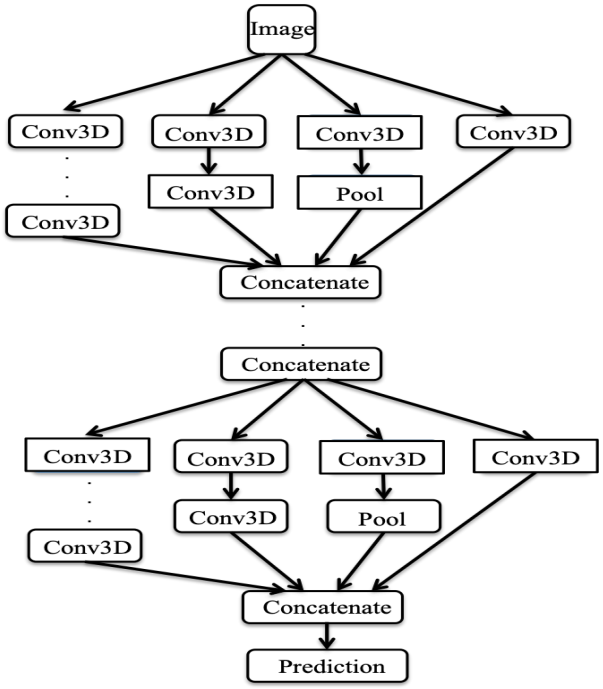

Figure 4.4: A basic Inception network architecture

4.3.2 CNN Network for Image Detection

Image classification is the more advanced application of the convolutional neural network (CNN) technology than image classification. The image detection network is divided into two parts: the backbone network determines the classification of objects and the object detector network determines the location of objects. The backbone network is a deep neural network that extracts the basic features for object detection. The object detector network is a single deep neural network that predicts the bounding boxes and the class probabilities for all detecting objects. Generally, the object features are extracted from the input image using the backbone network at first. Then the output layers of the backbone network connect the object detector network for the classification of the extracted features. The output of the object detector network classifies N + 1 predictions, where N is the number of classes, and 1 is for the background. The output also provides 4 coordinates predictions of each bounding box. The common image detection network

architectures include SSD, R-CNN, and YOLO. The paper chose these three kinds of common image detection architectures to test the performance of the effect of HPC. Figure 4.5 shows the architecture of a general object detection network.

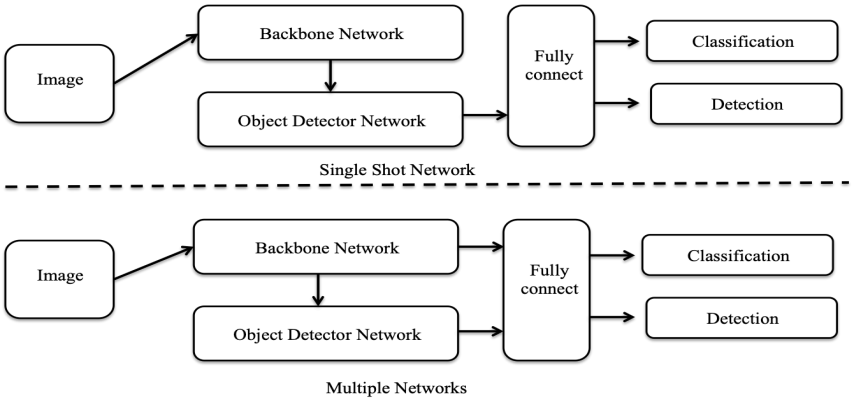

Figure 4.5: The architecture of a general object detection network

SSD network is a single shot mutibox detector for object detection in real-time. SSD network uses VGG16 network as the backbone network. VGG16 discards the fully connected layers and adds 6 auxiliary convolutional layers as the object detector network. Therefore, the SSD network can use multiple layers to detect objects independently. The auxiliary convolutional layers are for object detection. Multi-scale feature maps can improve accuracy significantly. SSD network associates a set of default bounding boxes with each auxiliary convolutional layer. The default bounding boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. Object detector network predicts 4 offsets relative to the original default box shape depend on feature map cells. Figure 4.6 shows SSD network architecture [65].

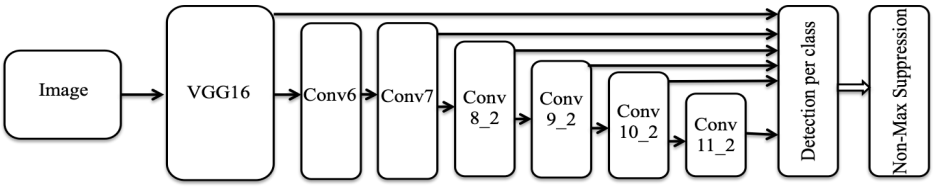

Figure 4.6: SSD network architecture

THE fast R-CNN network is designed to tackle object detection problems. Fast R-CNN network includes backbone network, region proposal network and full connect network. The backbone network extracts the features for object classification. The region proposal network is similar to the backbone network. It combines the features and forms a fixed-length feature vector in the RoI pooling layer. Each of the feature vectors consists of a classification module and a localization module. The classification module classifies object classes. The localization modules output four locations for each object class. The full connect network connects the RoI pooling layer for the classification and localization of objects. Figure 4.7 shows Fast R-CNN network architecture [66].

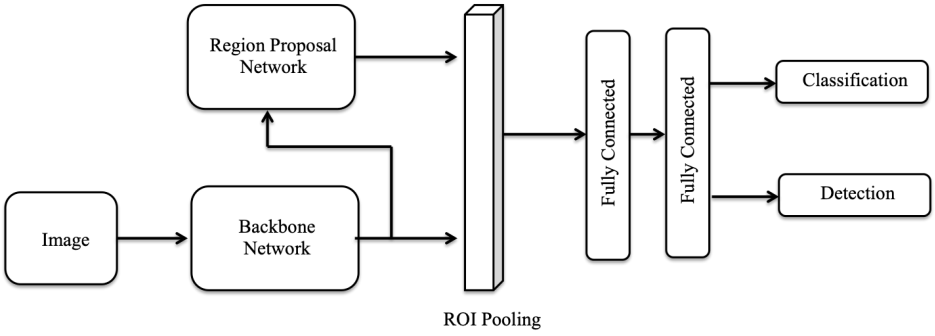

Figure 4.7: Fast R-CNN network architecture

YOLO network is one of the faster CNN networks for object detection. Although it is not the most accurate object detection network, it is a good choice for real-time detection without loss of too much accuracy. Like the SSD network, the YOLO network also uses a deep CNN network as the backbone network for feature extraction and an FPN network for object detection. YOLO network algorithm splits an input image into m

x m grid cells. Each grid cell predicts whether the center of the object falls into the grid cell. YOLO networks include YOLO v1, YOLO v2, YOLO v3, YOLO v4, and YOLO v5. Figure 4.8 shows YOLO network architecture [67].

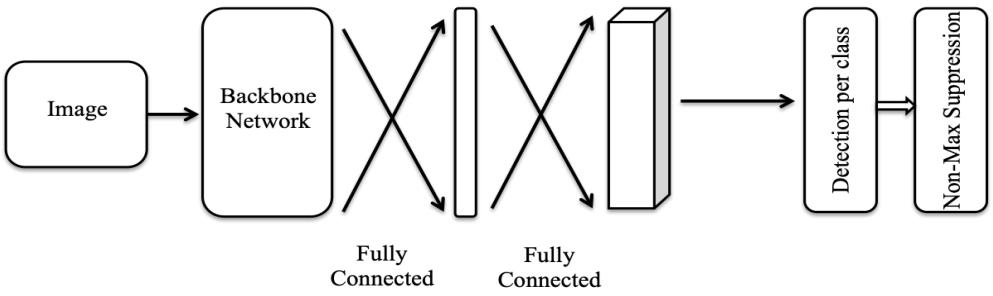

Figure 4.8: YOLO network architecture

4.3.3 CNN Network for Image Segmentation

Image segmentation is one of the most fast-growing applications of the CNN network because artificial intelligent machines need to analyze any object in a given image scenario today. It needs to combine image classification and detection technologies. Image segmentation is of two types: semantic segmentation and instance segmentation. Semantic segmentation links each pixel for each class label in an image. U-Net networks are for semantic segmentation. Instance segmentation masks each instance of an object contained in an image independently. Mask R-CNN network is for instance segmentation.

The panoptic segmentation combines semantic and instance segmentation such that all pixels are assigned a class label and all object instances are uniquely segmented. PANet network is for panoptic segmentation. The paper chose these three kinds of common image segmentation architectures to test the performance of the effect of HPC.

U-net network looks like a “U” which justifies its name. U-net is one of the famous Fully Convolutional Networks (FCN) for biomedical image segmentation. FCN network is an end-to-end deep CNN network for the prediction of image segmentation.

FCN network is different from traditional CNN. It gets rid of fully-connected layers and only uses convolution and pooling layers. U-net network consists of three parts: contraction network, bottleneck network, and expansion network. The contraction network is made of many convolution blocks. Each block has two convolution layers with 3×3 filters followed by a max-pooling layer with 2×2 filters. It tries to extract the features from the input image with a series of convolution layers. The bottleneck network connects the contraction network and the expansion network. It uses two convolution layers with 3×3 filters followed by an up convolution layer with 2×2 filters. The extension network is similar to the contraction network. It also has many convolution blocks. Each block has two convolution layers with 3×3 filters followed by a 2×2 upsampling layer with 2×2 filters. An extension network is used to reconstruct the features. Figure 4.9 shows U-net network architecture.

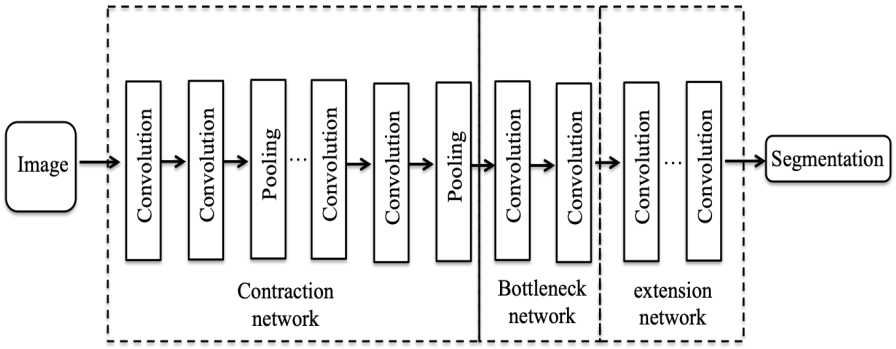

Figure 4.9: U-net network architecture

Mask R-CNN network is a deep neural network based on the Faster R-CNN network for instance segmentation. Fast R-CNN network classifies the objects and finds the bounding box of each object from an image. Extending the Faster R-CNN network, the Mask R-CNN network adds a binary mask classifier to predict a binary mask for each RoI. The binary mask classifier consists of CNN networks. This classifier uses various blocks of convolution and max pool layers to decompress an image. It then makes a class

prediction at this level of granularity. Finally, it uses up-sampling and deconvolution layers to resize the image to its original dimensions. So besides object classification and object localization, the Mask R-CNN network also predicts the pixels of each object detected. Figure 4.10 shows Mask R-CNN network architecture.

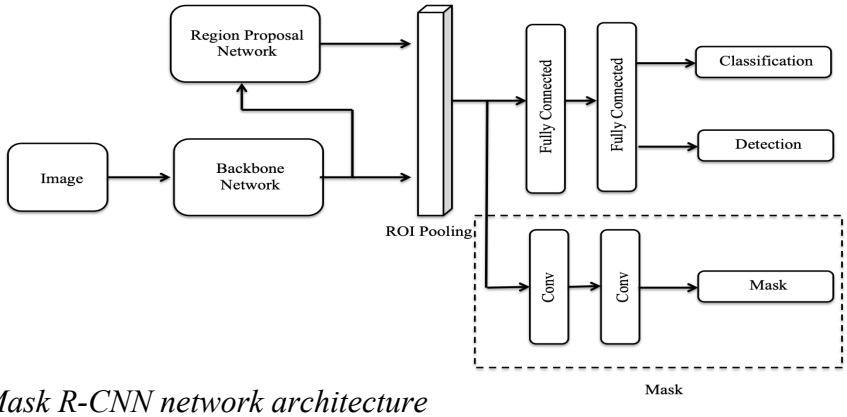

Figure 4.10: Mask R-CNN network architecture

PANet network is an extended Mask R-CNN for panoptic segmentation. Usually, Mask R-CNN provides good results on instance segmentation tasks. PANet uses FPN to provide information propagation paths. FPN used employs a top-down path to combine semantically rich features from high-level layers with accurate localization information residing in the higher resolution feature-maps of lower layers. It uses adaptive feature fooling to capture information from all levels. Figure 4.11 shows PANet network architecture.

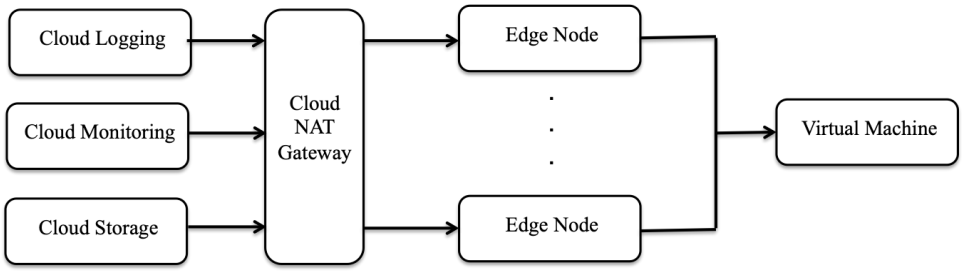

Figure 4.11: PANet network architecture

4.4 Experiment and Results

4.4.1 Hardware

Google Collaboratory (Colab) [68] is a compiler tool for machine learning developers. Google Colab provides CPU and GPU for the notebook that runs the programs. In Google Colab, CPU uses 2 cores of Intel(R) Xeon(R) CPU @ 2.30GHz. The users can get 34 GB of available RAM from Google Colab. The runtime duration can stay connected for up to 24 hours, and idle timeouts are relatively lenient. GPU uses Nvidia Tesla P100. Tesla P100 is a professional graphics card by NVIDIA. It has 3584 CUDA cores and 16 GB HBM2 memory at 732 GB/s. Single-precision performance can arrive at 9.3 TeraFLOPS. In our experiment, we use the CPU and GPU of Google Colab to test the performance of three image processing applications respectively.

Google cloud provides tightly coupled HPC workloads for the customers. Figure 4.12 shows HPC architecture on Google cloud. Google Cloud can create a virtual machine (VM) that includes an operating system, microprocessor, memory, and storage. The VM is a custom Slurm cluster on the Google Cloud Platform. Slurm is one of the leading workload managers for HPC clusters. Slurm provides an open-source, fault-tolerant, and highly-scalable workload management and job scheduling system. In the VM, the users can customize the number of CPU cores, memory, the number of GPUs, and the GPU type one would like their virtual machine to have. In our experiment, we use the AI platform notebook provided by the Google cloud platform (GCP). The advantage of the GCP notebook is that people could edit their machine size to fit their requirements. In our experiment, we use HPC with GPU of Google Cloud to test the performance of three image processing applications respectively. In Google Cloud, GPU uses Nvidia Tesla P100. We use 4 GPUs and 64 GB HBM2 memory.

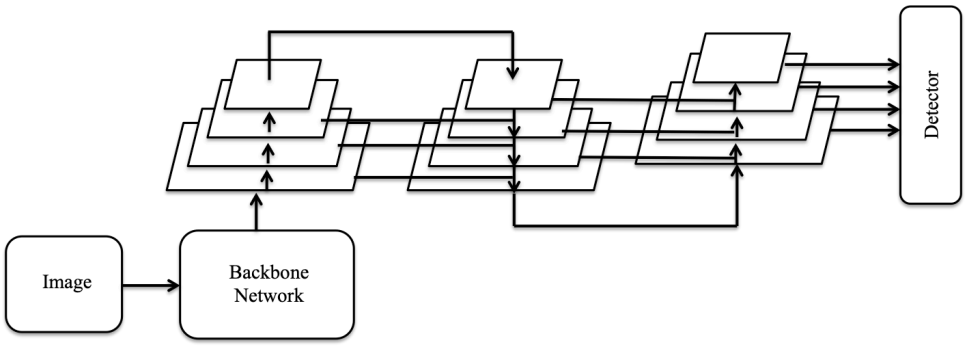

Figure 4.12: HPC architecture on Google cloud

XSEDE provides HPC resources for the programmers. We can access the PSC bridge-2 through XSEDE. PSC Bridges-2 is designed for converged HPC + AI + Data. Its custom topology is optimized for data-centric HPC, AI, and HPDA (High Performance Data Analytics). An extremely flexible software environment along with community data collections and BDaaS (Big Data as a Service) provide the tools necessary for modern pioneering research. The data management system, Ocean, consists of two-tiers, disk and tape, transparently managed as a single, highly usable namespace. PSC Bridges-2 has three types of compute nodes: "Regular Memory", "Extreme Memory", and GPU. The PSC Bridge-2 supercomputer comprises over 850 computational nodes. The paper was running in the Regular Shared Memory GPU (RSM-GPU) nodes which contained 48 nodes running either the Tesla K80 GPUs or P100 GPUS. The server being used is called the HPE Apollo 2000. Each Bridges' GPU node has two dual GPUs and two CPUs. The program can use anywhere from one GPU on one node to all GPUs on all GPU nodes.

4.4.2 Image Classification

PyTorch is an open-source machine learning library for Python. It gains widespread adoption because of its elegance, flexibility, speed, and simplicity. The PyTorch framework can be easy to build a simple neural network for an image

classification problem. PyTorch provides many functions for operating on tensors. The functions keep track of all the operations performed on tensors. Therefore, tensors can accelerate the numeric computations on GPU. Pytorch contains the model architectures for image classification. The model architectures include AlexNet, VGG, ResNet, Inception v3, GoogleNet, MobileNet, and so on. Our experiment uses VGG16, ResNet18, and Inception v3 model architectures on PyTorch for image classification.

In the image classification application, we use the ImageNet dataset organized according to the WordNet hierarchy. The ImageNet dataset is a large collection of human-annotated photographs for ILSVRC competition. There are more than 14 million images in the dataset and more than 21 thousand classes. We use about 1500 images for the training dataset, about 500 images for the testing dataset. In the training dataset, 1000 images are trained and 500 images are validated. We predict 10 object classes from the dataset. In the experiment, a batch size of 32 is chosen and the number of epochs is set to 100.

The experiment chooses three CNN networks for image classification: VGG16, ResNet18, and Inception v3. VGG16 has 13 convolutional layers, 5 max pool layers, and 3 fully connected layers. It includes a total of 138 million parameters. All of the convolution kernels are of size 3×3 and max pool kernels are of size 2×2 with a stride of 2. ResNet18 has 17 convolutional layers, 2 pooling layers, and 1 fully connected layer. It includes a total of 11 million parameters. It consists of convolutional layers with filters of size 3×3. Inception v3 has 81 convolutional layers, 15 pool layers, and 4 fully connected layers. It includes a total of 24 million parameters. It consists of convolutional layers with filters of size 1×1, 1×3, 3×3, 1×7, and 7×7. Table 4.1 shows the introduction of the pre-trained model.

Table 4.1: The introduction of the pre-trained model

Pretrained Model	Convolutional layers(levels)	Pooling layers(levels)	Fully Connected layers(levels)	Parameters(million)
VGG16	13	5	3	138
ResNet18	17	2	1	11
Inception V3	56	15	4	24

The experiment uses the three CNN networks to train the model on three computing platforms respectively. The input image is RGB formats and the size of the image is 224 × 224. In the first experiment, the accuracy of VGG16 is 67%, the accuracy of ResNet18 is 77%, and the accuracy of Inception v3 is 72%. The result of the experiment shows Inception v3 spends the shortest time for training the model. Although it has the most convolutional layers, the architecture of Inception v3 is parallel. Many convolutional layers can run on the computing platform simultaneously. VGG16 spends the longest time because it has the most parameters than other networks. Compare four computing platforms, the work efficiency of high performance computer with GPU is lower than Google Colaboratory (Colab) with CPU and GPU. Compare CPU and GPU, GPU decreases 93% than CPU in time. Figure 4.13 shows the result of the training model in three architectures.

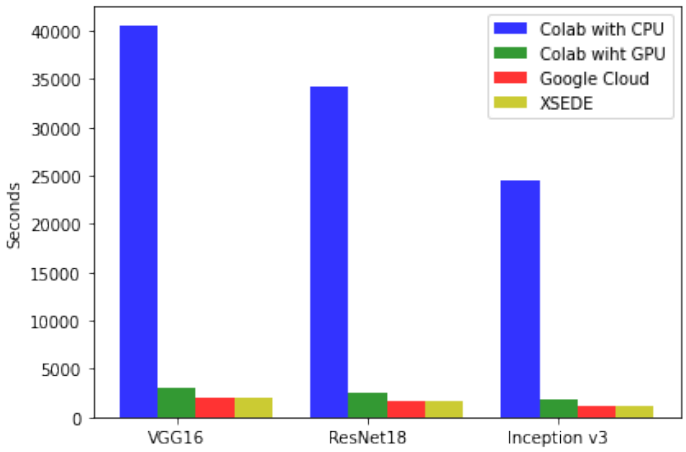

Figure 4.13: The result of the training model in three architectures

4.4.3 Image Detection

TensorFlow is a computational framework for image detection. TensorFlow was developed by Google and it's one of the most popular Machine Learning libraries on GitHub. The core data type in TensorFlow is the computational graph. The nodes of the same level in a computational graph can be executed in parallel. Tensorflow allows users to make use of parallel computing devices to perform multiple node operations. It can create multiple workers to schedule tasks on various computing devices. Therefore, TensorFlow can schedule the tasks on parallel computing devices (GPU). It also can schedule the operations on CPU and GPU simultaneously. Our experiment adopts three image detection architectures with Tensorflow.

In the image detection application, we use an open image dataset from the google website. It uses almost 9 million URLs for images. These images have been annotated with image-level labels bounding boxes spanning thousands of classes. The dataset contains a training dataset of 9 million images, a validation dataset of 41,260 images, and a test dataset of 125,436 images. We use about 1500 images for the training dataset, about 500 images for the testing dataset. In the training dataset, 1000 images are trained and 500 images are validated. We detect 5 object classes from the dataset. In the experiment, a batch size of 32 is chosen and the number of epochs is set to 100.

The experiment chooses three CNN network architectures for image detection: SSD, Fast R-CNN, and Yolo v3. SSD network architecture has a VGG16 network as the backbone network and 6 convolutional layers as the object detector network. It includes a total of 26.3 million parameters. It consists of 19 convolutional layers. Fast R-CNN network architecture has a VGG16 network as the backbone network and a region proposal network as the object detector network. It includes a total of 134.7 million parameters. It consists of 21 convolutional layers. Yolo v3 network architecture has a

darknet-53 network as the backbone network and 3 convolutional layers as the object detector network. It includes a total of 61 million parameters. It consists of 53 convolutional layers. Table 4.2 shows the introduction of the pre-trained model.

Table 4.2: The introduction of the pre-trained model

Petrained Model	Convolutional layers(levels)	Backbone Network	Parameters(million)
SSD	19	VGG16	26.3
Fast R-CNN	21	VGG16	134.7
Yolo v3	53	Darknet-53	61

The experiment uses the three CNN architectures to train the model on three computing platforms respectively. The input image is RGB formats and the size of the image is 224×224. In this experiment, we see that the accuracy of SSD is 72%, the accuracy of Fast R-CNN is 83%, and the accuracy of Yolo v3 is 79%. The result of the experiment shows SSD architecture spends the shortest time for training the model because it has minimum parameters and convolutional layers. Fast R-CNN spends the longest time because it has the most parameters than other networks and two layers structure of CNN networks. Compare four computing platforms, the work efficiency of high performance computer with GPU is lower than Google Colaboratory (Colab) with CPU and GPU. Compare GPU and Google cloud with HPC, HPC decreases 33% than GPU in time. Figure 4.14 shows the result of the training model in three architectures.

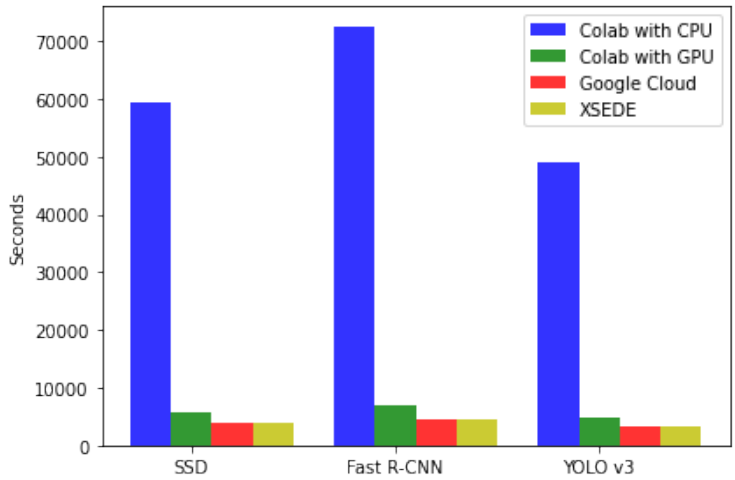

Figure 4.14: The result of the training model in three architectures

4.4.4 Image Segmentation

Keras is one of the most popular deep learning libraries. It provides a convenient way to train a deep learning model. Because Keras is a neural network API that runs on top of Tensorflow, Theano, and CNTK library, the developers can be easy to create a CNN with the functional API. Keras models accept three formats of input data: NumPy arrays, TensorFlow Dataset objects, and Python generators. These input data can be preprocessed asynchronously on the CPU while your GPU is busy. The data is buffered into a queue. When GPU finished the previous batch data, the data on memory is immediately available. So GPU can reach full utilization. Our experiment adopts three image Segmentation architectures with Keras.

In an image segmentation application, we use a COCO-Stuff dataset for image segmentation. COCO-Stuff dataset augments 164,000 images of the popular COCO dataset with pixel-level stuff annotations. The dataset contains a training dataset of 164,000 images, a validation dataset of 5,000 images, and a test dataset of 20,000 images for the image segmentation challenge. It covers 172 classes: 80 thing classes, 91 stuff classes, and 1 class ‘unlabeled’. We use about 1500 images for the training dataset, about 500 images for the testing dataset. In all of the training datasets, 1000 images are trained

and 500 images are validated. In the experiment, a batch size of 32 is chosen and the number of epochs is set to 100.

The experiment chooses three CNN network architectures for image segmentation: U-net, Mask R-CNN, and PANet. U-net network architecture has 23 convolutional layers and 4 pooling layers. It includes a total of 7,759,521 parameters. All of the convolution kernels are of size 3x3 and maxpool kernels are of size 2×2. Mask R-CNN network architecture has a VGG16 network as the backbone network and a region proposal network as the object detector network. It includes a total of 134.7 million parameters. It consists of 21 convolutional layers and 2 pooling layers. PANet network architecture has a VGG16 network as the backbone network and an FPN network as the object detector network. It includes a total of 14.7 million parameters. It consists of 31 convolutional layers and 5 pooling layers. Table 4.3 shows the introduction of the pre-trained model.

Table 4.3: The introduction of the pre-trained model

Petrained Model	Convolutional layers(levels)	Pooling layers(level)	Parameters
U-net	23	4	7,759,521
Mask R-CNN	22	5	134.7M
PANet	31	5	14.7M

The experiment uses the three CNN architectures to train the model on three computing platforms respectively. The input image is RGB formats and the size of the image is 224 × 224. From the experiment, the accuracy of U-net is 57%, the accuracy of Mask R-CNN is 65%, and the accuracy of PANet is 63%. These accuracies are lower because the number of the dataset and the size of images are small. The result of the experiment shows U-net architecture spends the shortest time training the model because

it has minimum parameters and convolutional layers. Mask R-CNN and PANet are both based on Fast R-CNN. Mask R-CNN spends the longest time because it has the most parameters than PANet networks. Besides, PANet includes an FPN network that is parallel. Therefore, PANet spends a shorter time than Mask R-CNN. Comparing four computing platforms, the efficiency of high performance computer with GPU is lower than Google Colaboratory (Colab) with CPU and GPU. Compare Google Cloud with HPC and XSEDE with HPC, they run with similar time because they have the same hardware resources. Figure 4.15 shows the result of the training model in three architectures.

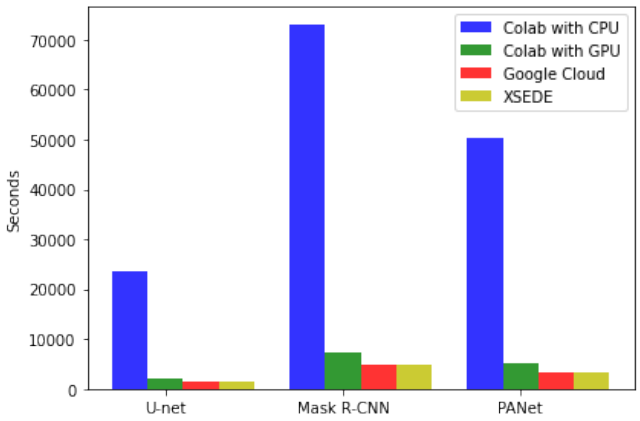

Figure 4.15: The result of the training model in three architectures

4.5 Conclusion and Future Work

Image classification, image detection, and image segmentation are the most basic applications in image processing. Robots and self-driving cars often use image processing to identify the presence, location, and type of one or more objects. So far, CNN network is a widely used technology to solve image processing problems. However, if the CNN network is applied to computer vision, the computing platform is a challenging problem. This paper tested and compared the performance of different computing platforms with several popular CNN network architectures in different image processing applications.

Each CNN network architecture represents different trends that are developing in applications. The results of the experiment show we were able to successfully train our model in a high-performance environment. Because HPC owns more flexible hardware resources than Google Colab, HPC is more suitable for CNN network applications. In the future, we will use other HPCs on the cloud to test the performance of CNN network architectures.

CHAPTER V:

PROPOSED SYSTEM SETUP AND INTEGRATION

The system contains hardware and software. The hardware consists of Raspberry Pi, camera module, wireless adapter, touch screen. Raspberry Pi provides computing, like a mini-computer. The camera module collects images and videos. The wireless adapter helps the Raspberry Pi to connect Internet without a physical connection. The touch screen makes the system more portable. Software is installed in Raspberry Pi through the interface of the command line. The software consists of TensorFlow, Keras, and scikit-learn, OpenCV, Dlib, NumPy, and imutils.

5.1 Hardware

Raspberry Pi 3 Model B+, distance sensor, motion sensor, camera module, Wireless adapter, touch screen, HDMI cable, Ethernet cable, MicroSD card, and power adapter are used in the project. Raspberry Pi 3 Model B+ uses a Linux operating system and provides computing to the project. The camera module offers to capture real-time video. The wireless adapter is used to connect Raspberry Pi to the wireless network.

5.1.1 Raspberry Pi 3 Model B+

Figure 5.1: A Raspberry Pi 3 Model B+

Raspberry Pi 3 Model B+ shown in Figure 5.1 has 1.4GHz 64-bit quad-core ARM Cortex processor, CSI camera port for connecting a Raspberry Pi camera, and Micro SD port for loading the operating system and storing data. A Raspberry Pi is shown in Fig.16. It supports dual-band 802.11ac wireless LAN, Bluetooth 4.2/BLE, 3×faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT). It also provides remote access [47].

Specification of Raspberry Pi 3 Model B+ is as follows:

- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- Gigabit Ethernet over USB 2.0 (maximum throughput of 300 Mbps)
- Full-size HDMI
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)
- Micro SD port for loading your operating system and storing data
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port

- CSI camera port for connecting a Raspberry Pi camera
- 4 USB 2.0 ports
- 5V/2.5A DC power input
- The extended 40-pin GPIO header

5.1.2 Camera Module and Wireless Adapter

Camera module supports Raspberry Pi Model A or B, B+, Raspberry Pi 3, and Raspberry Pi 3 Model B+. The angle of View is 54×41 degrees. Max frame rate is 30 frames per second. The maximum video resolution is 1080p and the still picture resolution is 2592×1944. It has an Omnivision OV5647 sensor in a fixed-focus lens and an infrared cut-off filter. The Pi camera is connected with Raspberry Pi by a CSI port. A cameral model is shown in Figure 5.2.

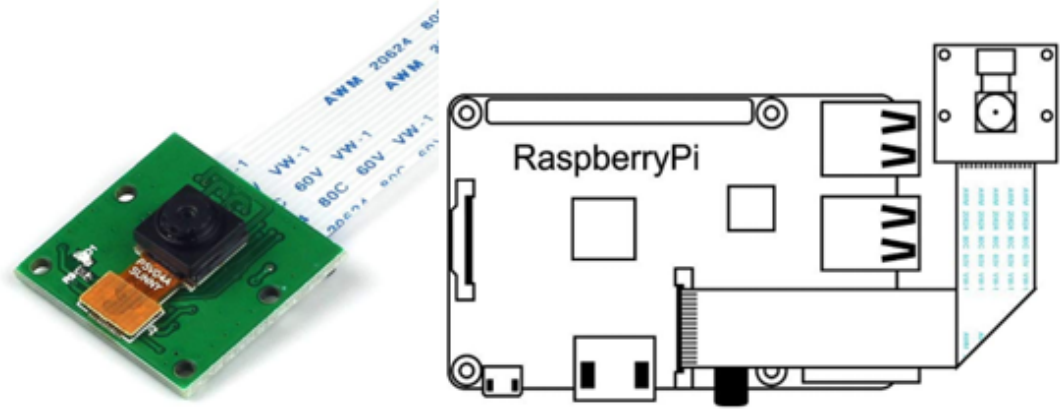

Figure 5.2: Camera module

A wireless adapter is a hardware device that is attached to a computer or other device to make it connect to a wireless network. It is shown in Figure 5.3. In the project, the wireless adapter is plugged into one of the USB ports on a Raspberry Pi to access the wireless network.

Figure 5.3: Wireless adapter

5.2 Hardware Setup

As a single-board computer, Raspberry Pi needs to be installed in an operating system and libraries. Raspbian is installed in the system. It is an official supported operating system for all models of the Raspberry Pi. We use Python 3.5.6 as our programming language. The steps of setup are as follows,

Step 1. Insert an SD card into the Raspberry Pi, then install Raspbian. Downloaded Raspbian from the website, https://www.raspberrypi.org/downloads/.

Step 2. Create a Python virtual environment and install NumPy, imutilsTensorFlow, and Keras.

Step 3. Download OpenCV and opencv_contrib libraries, install these libraries in the Raspberry Pi.

Step 4. Install the library dlib. A facial landmark detector is used in the system.

Step 5. Open the Raspberry Pi configuration and make that the camera is enabled. Test the cameral module by the command line *raspistill -o Desktop/image.jpg*.

Step 6. Install RPi.GPIO library to read and write pins on the GPIO header.

5.3 Software

5.3.1 Libraries for Deep Learning and Machine Learning

The libraries of TensorFlow, Keras, and scikit-learn are used to construct neural networks and classification models.

TensorFlow is a powerful framework for machine learning and deep learning. It was released by Google Brain on November 9, 2015, with the Apache 2.0 open source license. People can free use, change and distribute modifications. The framework supports C++, Python, and et al. In the system, we use Python. TensorFlow supports GPUs and CPUs. It, as the name indicates, manipulates tensors. A tensor is a vector and matrix which has n-dimensions. TensorFlow codes contain two parts. The first part is building a graph that represents the data flow of computations. The other part is executing a session that runs the graph. TensorFlow is a low-level mathematical framework. But Keras is a high-level framework with encapsulation. The simplicity of Keras comes from many intuitive functions. It is built on top of TensorFlow, in other words, TensorFlow offers the backend for it. Compared TensorFlow, it is a high-level library. When people install Keras, TensorFlow is needed to be installed at first. In the thesis, we use Keras to construct neural networks because of friendly implementation and fast prototyping, easy extensibility, and work with Python.

Scikit-learn is a simple and efficient tool for machine learning. It is built on NumPy, SciPy, and matplotlib. Additionally, it is an open-source Python library that contains many algorithms implemented in classification, clustering, regression, dimensionality reduction, model selection, and preprocessing. The library was publicly released in February 2010.

5.3.2 OpenCV and Dlib

OpenCV (Open Source Computer Vision Library) is an open-source library that provides more than 2500 algorithms in the field of computer vision. It owns Python, C++, MatLab and Java interfaces. We use a pre-trained Haar-Cascade algorithm from OpenCV for face detection and a VideoCaputre object for reading frames of videos.

Dlib is a C++ toolkit that contains many machine learning algorithms to solve problems in a lot of domains containing robotics, mobile phones, embedded devices, etc. It is also free of charge because of its open source license. In the thesis, a pre-trained landmark's facial detector model is implemented for face alignment.

5.3.3 NumPy and Imutils

NumPy is a fundamental package in Python. Functions of it include computation of multi-dimensional array and matrices, complex linear algebra, Fourier transformation and so on. In the thesis, it is used to represent images and process frames. For example, a gray image is represented by a 2-dimensional array.

Imutils is a library containing a set of efficient functions. These functions are related to image processing, for instance, rotation, affine transformation, resizing, translation, edge detection, and so on. We synthesize those functions to achieve face alignment.

CHAPTER VI:

PROPOSED METHODS

In order to meet the function of people tracking of the system, we need a modified meanshift algorithm for people tracking and a CNN model for people detection. The work contains dataset selection and collection, image preprocessing, meashift algorithm design, CNN design, and CNN model training. The workflow of the tracking system is shown in Figure 6.1. First, we capture the frame as the input image from camera. The detection system extracts the features in pretrained CNN model. Therefore, we can detect all of the human objects in the input image. Second, we choose the tracking target from all of the person and calculate the histogram for the tracking target. Third, we capture the next frame from camera. We highlight the tracking target through the pre-process frame Camshift algorithm can find the tracking target in the picture. Finally, we calculate the histogram for the new tracking target. Comparing the new histogram and previous histogram. If they are similar, we continue to use camshift algorithm to tracking the next frame. But if they are great differences, we use the CNN model to detect the position of the tracking target.

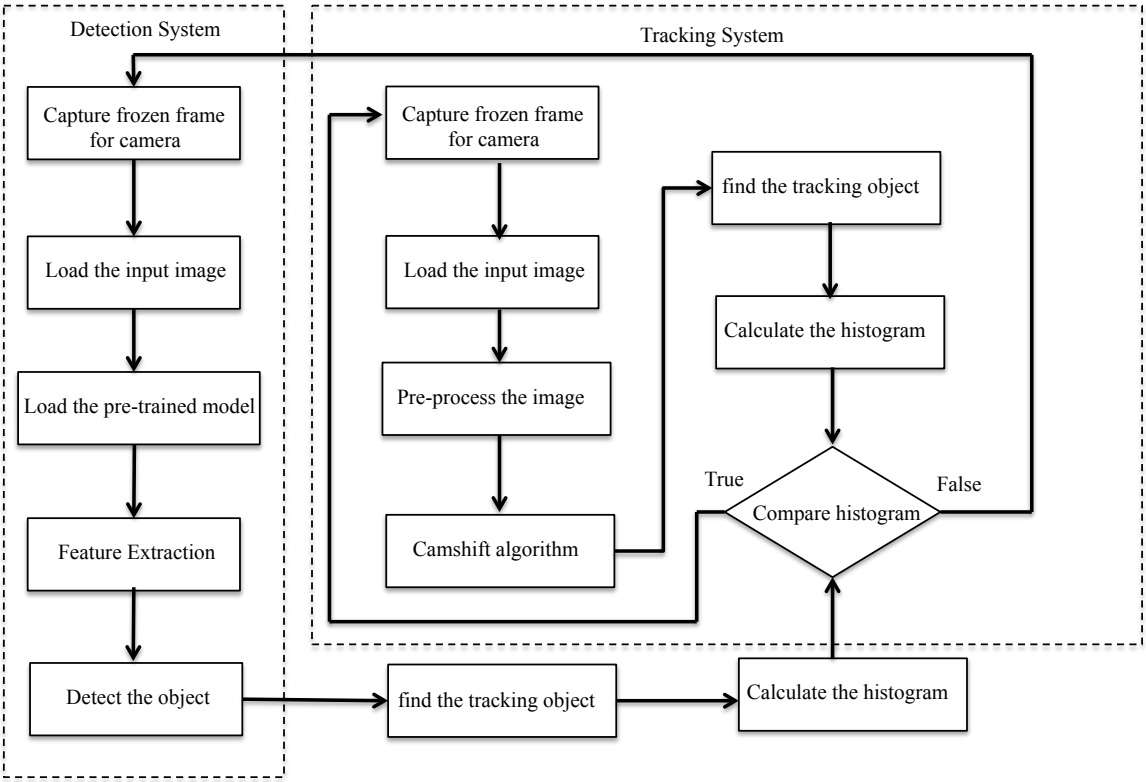

Figure 6.1: The workflow of person tracking

6.1 Camshift Algorithm Design

6.1.1 Image Preprocessing

In the thesis, image preprocessing contains color to grayscale conversion, face cropping, and affine transformation.

Color to grayscale conversion is implemented by a function in OpenCV. The function is gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY). Images or videos captured by cameras are color images. Each pixel in an image is described by three parameters. The three parameters represent the intensity of red, green, and blue, respectively. Each parameter is an integer which is between 0 and 255. However, each pixel in a gray image is defined by a single parameter that describes the intensity of luminance or an amount of light. The parameter ranges from 0 to 255. 0 represents black

at the weakest intensity and 255 represents white at the strongest intensity. Convert color images to gray images because the following reasons:

(a) Compared to color images, each pixel of a gray image just has one parameter, so a gray image has fewer dimensions. Gray images need less amount of computing and reduce the complexity of a model [21].

(b) Color information does not help to detect edges [20]. The gradient of the intensity of luminance is significant to edge or pattern detection which is effective information to distinguish objects. Converting color images to gray images reserves a gradient of the intensity of luminance and avoids color information.

6.1.2 Average Histogram

When the initially selected region contains some pixels from outside the object (background pixels), our 2D probability distribution image will be influenced by their frequency in the histogram back-projection. In order to assign a higher weighting to pixels nearer to the regional center, a weighted histogram may be used to compute the target histogram. We store histograms of five previous successive targets and calculate the average histogram of these five targets. Every time, when we obtain the histogram of the tracking object, comparing it and the average histogram. If they are similar, the tracking object will be confirmed. The person histogram is shown in Figure 6.2.

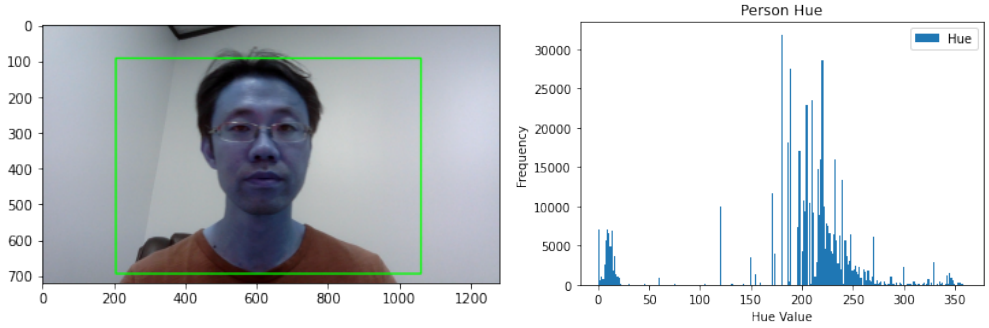

Figure 6.2: The person histogram

6.1.3 Illumination Changing of The Object

The RGB color space has weakness in representing shading effects or rapid illumination changes. We use Hue Saturation Value to describe the object. Only H channel in HSV model can express the color information, so we extract it to make a histogram. We calculate the HUE histogram of the object. For the tracking object of each frame, we calculate the threshold range of the histogram. We can normalize the value range of the histogram corresponding to every chrominance grade to the interval of [0, 255] so that through back projection the value of each pixel in the image is related to the corresponding to the value of each chrominance grade in the histogram. When the value of a pixel is not in the value range of the histogram, we change the value of the pixel to 0. Otherwise, we change the value of the pixel to 255. Thus the probability distribution graph of object color is obtained. The result of the probability distribution is shown in Figure 6.3.

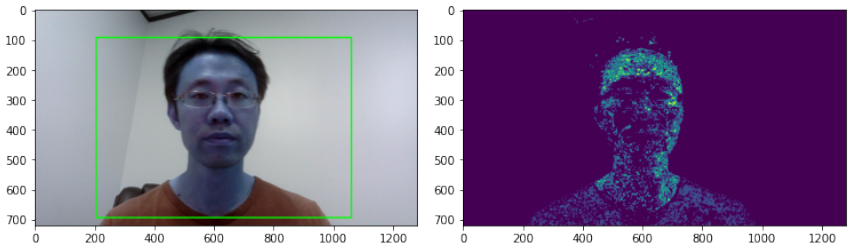

Figure 6.3: The result of the probability distribution

6.1.4 Camshift Algorithm

The CAMShift (**Continuously Adaptive Mean Shift**) algorithm is a color-based object tracking method. The CAMShift algorithm is derived from the mean shift algorithm, which is responsible for finding the center of the probability distribution of the object to track. The main difference is that CAMShift adjusts itself to the search window size.

To calculate the new location of a target, the meanshift algorithm is used. Meanshift takes a probability distribution image and an initial search window computes the window's center of mass. This movement will change what is under the window, and so the recentering process is repeated until the movement vector converges to zero. The last calculated center of mass will be the new location of the target [25].

The zeroth and first moments are calculated as:

$$M_{00} = \sum_X \sum_Y I(x,y) \quad (1)$$

$$M_{10} = \sum_X \sum_Y xI(x,y) \quad (2)$$

$$M_{01} = \sum_X \sum_Y yI(x,y) \quad (3)$$

Where I(x,y) is the intensity value of the point (x,y) in the probability distribution image.

The following equations are used to calculate the search window's center of mass (y_c, x_c)

$$x_c = \frac{M_{10}}{M_{00}} \quad (4)$$

$$y_c = \frac{M_{01}}{M_{00}} \quad (5)$$

The search window update:

$$S = 2\sqrt{M_{00}/256} \quad (6)$$

The camshift algorithm is calculated using the following steps:

Step 1: Choose the initial location of the search window

Step 2: Execute the mean shift (one or many iterations):

Step 3: Compute the mean location in the search window

Step 4: Centre the search window at the mean location computed in the previous step

Step 5: Repeat steps 3 and 4 until you obtain convergence (or until the mean location moves to lower than the preset threshold)

Step 6: Set the search window size equal to a function of the zeroth moment found in Step 2.

One of the major drawbacks of camshift algorithm is that it cannot track the desired target when the background is of the same color. The mean shift moves the search window to the area of the maximum pixel density of the probability distribution, which is created with the histogram of the target to track. In this case, it would be necessary to encode structural image features, which is not possible with color-based tracking.

6.2 CNN Model Design

6.2.1 Feature Extractor Network Design Process

Darknet-53 is the original backbone network of YOLOv3. Darknet-53 includes 52 fully convolution layers, in which 46 layers are divided into 23 residual units with 5 different sizes [25]. The residual units are designed to avoid the vanishing-gradient problem inspired by the Resnet [26]. The Darknet-53 is a complex network, and its 40549216 parameters provide a guarantee for detection accuracy. However, our tracking system the object detection with a single category such as ship, the excessively huge parameters would bring overfitting risk and slow down the detection speed.

We design a new backbone network that is a simplified version of YOLOv3. It includes 12 convolution layers and 6 average pooling layers. The convolution layers are divided into 6 residual units with 3 different sizes. So, the new backbone network has a faster detection speed than YOLOv3 because of its shallow and simple network structure; however, its detection accuracy is lower obviously than YOLOv3. The visualization of the backbone network architecture is shown in Figure 6.4.

Therefore, for person detection, it is important to preserve the depth of network for capturing enough features to ensure detection accuracy while reducing network parameters to speed up. In addition, the tracking system is installed on a tiny and

affordable computer. The system speed can be affected by hardware limits. So, how to utilize the shallow network as much as possible to improve the detection speed becomes the key issue that should be solved. This paper proposes a small backbone network to achieve this goal.

	Type	Filters	Size	Output
	Convolutional	64	3×3	128×128
1×	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	3×3	64×64
1×	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	3×3	32×32
1×	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	3×3	16×16
1×	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	3×3	8×8
1×	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 6.4: The visualization of the backbone network architecture

6.2.2 Detection Network Design Process

We use a feature pyramid network (FPN) as the detection network in CNN model. FPN is a feature extractor designed for such a pyramid concept with accuracy and speed in mind. Feature pyramids are a basic component in recognition systems for detecting objects at different scales [27]. the FPN topology allows the detection network to learn objects at three different sizes: The 13×13 detection block has a broader context and a poorer resolution compared with the other detection blocks, so it specializes in detecting large objects, the 26×26 detection block specializes in detecting medium objects, the 52×52 detection block has a narrow context richer resolution compared with the other detection blocks, so it specializes in detecting small objects. Each of the detection heads has a separate set of anchor scales. The structure of the CNN is shown in Figure 6.5.

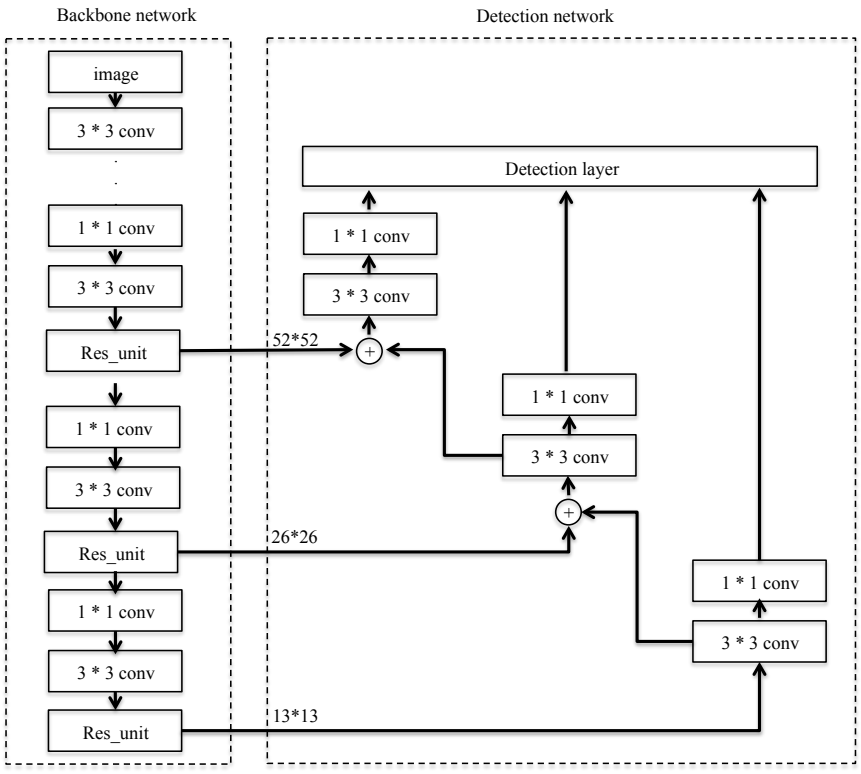

Figure 6.5: The structure of the CNN

6.2.3 Loss Function

The detection network predicts multiple bounding boxes per grid cell. To compute the loss for the true positive, there is one of the bounding boxes to be responsible for the object. For this reason, we select the one with the highest IoU with the ground truth. Each prediction can get better sizes and aspect ratios. The loss function composes of the classification loss, the localization loss, and the confidence loss.

The classification loss at each cell is the squared error of the class conditional probabilities for each class. If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class:

$$\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \quad (7)$$

where

$1_i^{obj} = 1$ if an object appears in cell i, otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

The localization loss measures the errors in the predicted boundary box locations and sizes.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2] \quad (8)$$

where

$1_{ij}^{obj} = 1$ if the jth boundary box in the cell I is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

w_i is the width of the box in cell i.

h_i is the height of the box in cell i.

The confidence loss measures the objectness of the box.

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 \quad (9)$$

where

$\hat{C}_i$ is the box confidence score of the box j in cell i.

$1_i^{obj} = 1$ if the j the boundary box in cell I is responsible for detecting the object, otherwise 0.

6.2.4 Non-maximal Suppression

Non-maximum Suppression (NMS) is a technique that filters the proposals based on some criteria. NMS selects the proposal with the highest confidence score, removes it from a list of original proposal boxes, and adds it to the final proposal list. IOU (Intersection over Union) calculation is used to measure the overlap between two proposal boxes. The IOU calculation is shown in Fig 6.6

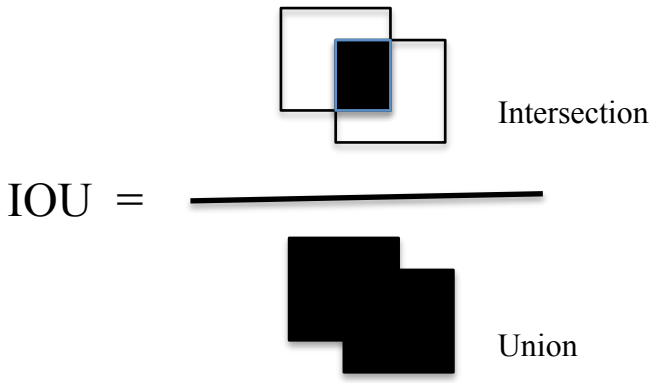

Figure 6.6: The IOU calculation

The overlap threshold is 0.5. Comparing the IOU (Intersection over Union) with every other proposal. If the IOU is greater than the threshold, remove that proposal from a list of original proposal boxes. NMS takes the proposal with the highest confidence from the remaining proposals in the original proposal list and removes it from the original proposal list and adds it to the final proposal list. NMS calculates the IOU of this proposal with all the proposals in the original proposal list and eliminates the boxes

which have high IOU than the threshold. This process is repeated until there are no more proposals left in the original proposal list.

6.2.5 Dataset Selection and Collection

For the CNN model, a customized Dataset using Google's Open Images [28] was created. Open Image is a collection of datasets that is customizable of approximately 9 million images annotated with labels, object bounding boxes, object segmentation masks, visual relationships, and localized narratives. The boxes have been already been manually drawn by professional annotators to ensure precision and regularity. The set of photos are incredibly diverse and include very complex scenes with several objects besides the selected classes.

To create the dataset, this project used darknet, an open-source platform of the Python and the neural network, a repository that can download any desired classes, create bounding boxes and format the files so it becomes compatible with the CNN model. The system successfully created a custom dataset of 2000 high-resolution images of our desired class, Person.

6.2.6 CNN Model Training Process

In the thesis, the split ratio is 0.1 in the customized dataset. For instance, the number of training images of a people is 9 and the number of testing is 1, assuming the number of all images of the people is 10. The dataset used to train the CNN model has a total of 2000 images belonging to the human class. The training dataset contains 1800 images and the testing dataset has 200 images. The epoch is set as 1000 which means the training dataset is passed forward and backward through the CNN model 100 times. A total of 1000 iterations to improve itself and reduce the losses it makes. The batch size is 128. Batch size is the number of training images in a batch. The training dataset is divided into batches because the entire dataset cannot be passed to the CNN at one with

the limitation of CPU or GPU memory. We choose other models: YOLOv3 and YOLOv3-tiny. We use three models to train the dataset. Compared with these two models and our CNN model on the customized dataset. The accuracy results are shown in Table 6.1.

Table 6.1: Comparison of model training results

	Parameters	Precision
YOLOv3	61,472,682	0.92
YOLOv3-tiny	8, 660, 345	0.85
Proposed model	68432	0.76

Comparing the three results, YOLOv3 has the highest accuracy, but it has too many parameters due to its deep layers network architecture. Therefore, YOLOv3 is not suitable for mobile devices. Although our proposed model doesn't have high accuracy, it has the lowest parameters than other models and highest frame per second. Therefore, it is easier to run on devices with limited memory. The accuracy is satisfied with the requirement of the system.

6.3 The System Design

The system captures the frame as the input image from camera. Then it highlights the tracking target through the pre-process frame Camshift algorithm can find the tracking target in the picture. The camshift algorithm tracks the target. The system calculates the histogram for the tracking target. Comparing the new histogram and previous histogram. If they are similar, we continue to use camshift algorithm to tracking the next frame. But if they are great differences, we use the CNN model to detect the position of the tracking target. The diagram of the system is shown in figure 6.7.

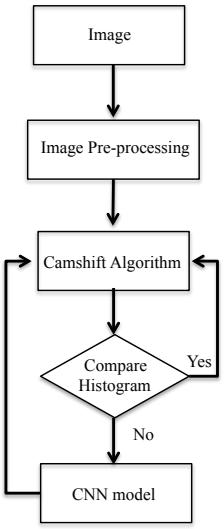

Figure 6.7: The diagram of the system

We choose two platform to test the performance of the proposed system. The first platform is Raspberry pi 3 that consists of ARMv8 CPU with 1.4GHz. The second platform is Apple mini consists of 6-core Intel Core i5 with 3.0 GHz. The performance of the proposed system is shown in Table 6.2.

Table 6.2: The performance of the proposed system

Platform	Image Size	Frame Rate
Raspberry Pi 3	416x416	25fps
Apple Mini	416x416	48fps

CHAPTER VII:

SIMULATION RESULTS

The tracking strategy proposed in this paper is used to perform the target tracking in the complex circumstances of the target occlusion, the pose variation, and multiple targets. The feasibility and effectiveness of the proposed method are shown by the comparative analysis with Camshift algorithm and CNN model.

7.1 Tracking Experiment for Single Target

In the first experiment that the system will track the body movement of a person. The body of a person is chosen as the initial frame. The position of the target change during video sampling. We use camshift algorithm and the proposed method to test the results respectively. Comparison tracking results of different positions with Camshift algorithm in first experiment are shown in Fig 7.1.

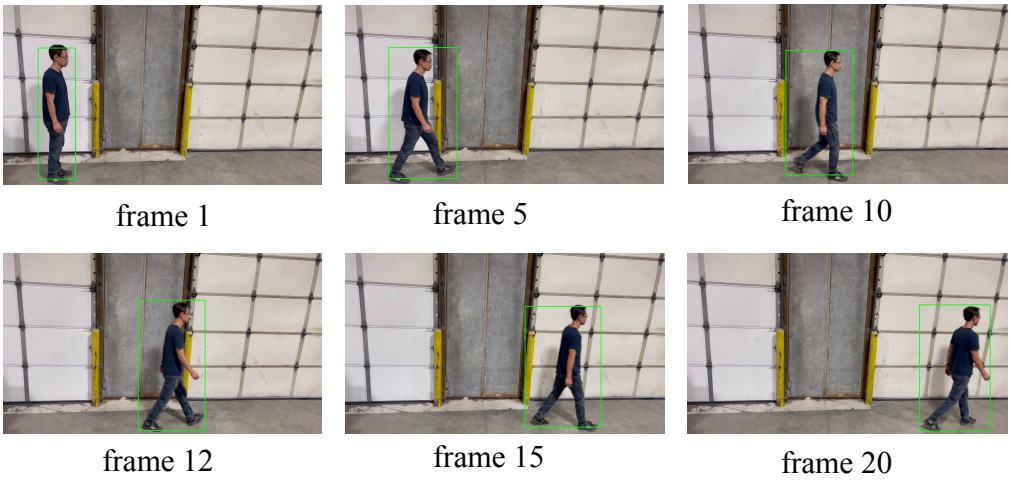

Figure 7.1: The results of different position with Camshift algorithm in first experiment

Comparison tracking results of different positions with the proposed method in first experiment are shown in Fig 7.2.

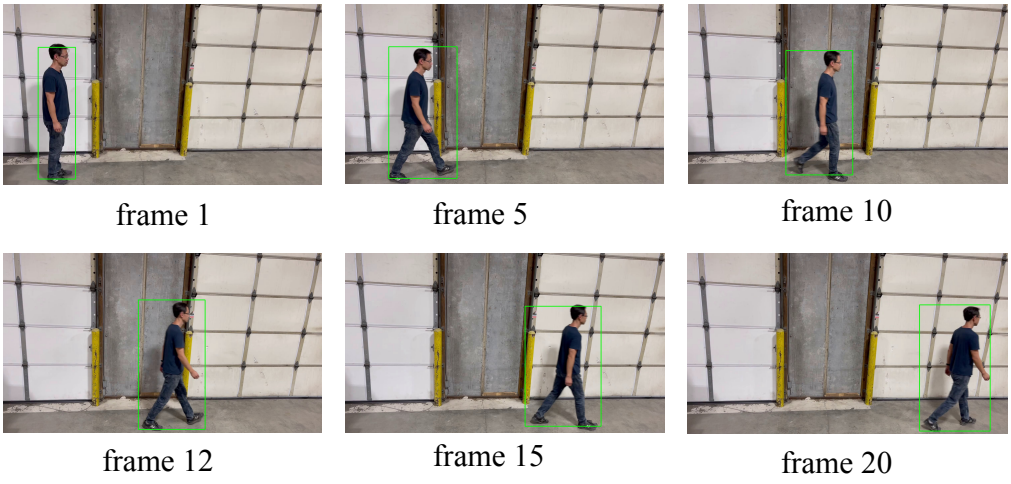

Figure 7.2: The results of different position with the proposed method in first experiment

Comparing two results, they obtain the same results. Because Camshift algorithm can complete the tracking successfully without the CNN model. In the proposed method, the tracking results are obtained by the basic Camshift algorithm. It is clear that the results located by the Camshift algorithm are accurate. There is less background information, which doesn't have interference with the target location, is included in the target area. The histogram of each frame are similar. We don't use CNN model to adjust the position of the tracking box.

7.2 Tracking Experiment for Person Occluded by Object

The second experiment is that the system will track a moving person. The person is detected by CNN model. So, we obtain the size of the tracking box in the initial frame. A person walks in the video. When the person walked behind an obstacle, the system waits and detects the target on the route of the walk. When the target appears in the picture again, the system will continue to track the target. We use camshift algorithm and the proposed method to test the results respectively. Comparison tracking results of different positions with Camshift algorithm in second experiment are shown in Fig 7.3.

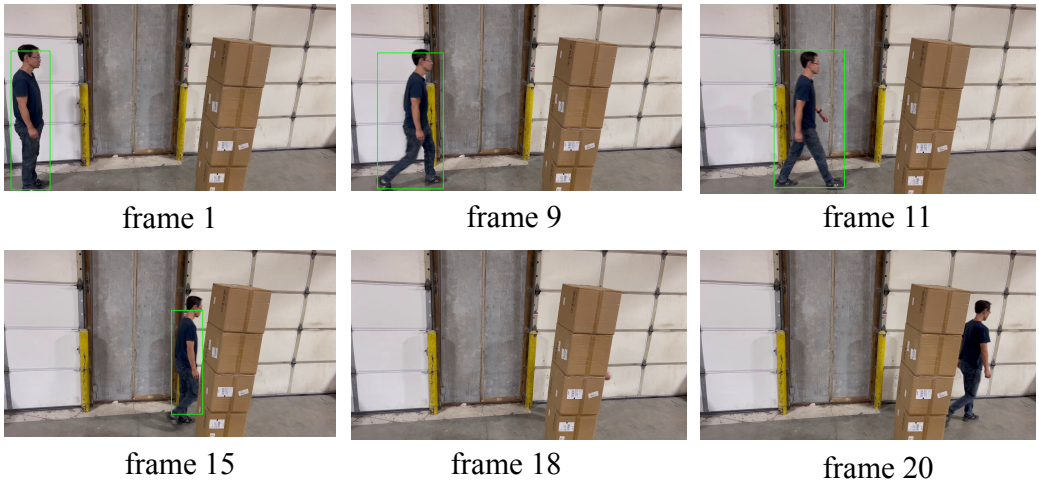

Figure 7.3: The results of different position with Camshift algorithm in second experiment

Comparison tracking results of different positions with the proposed method in the second experiment are shown in Fig 7.4.

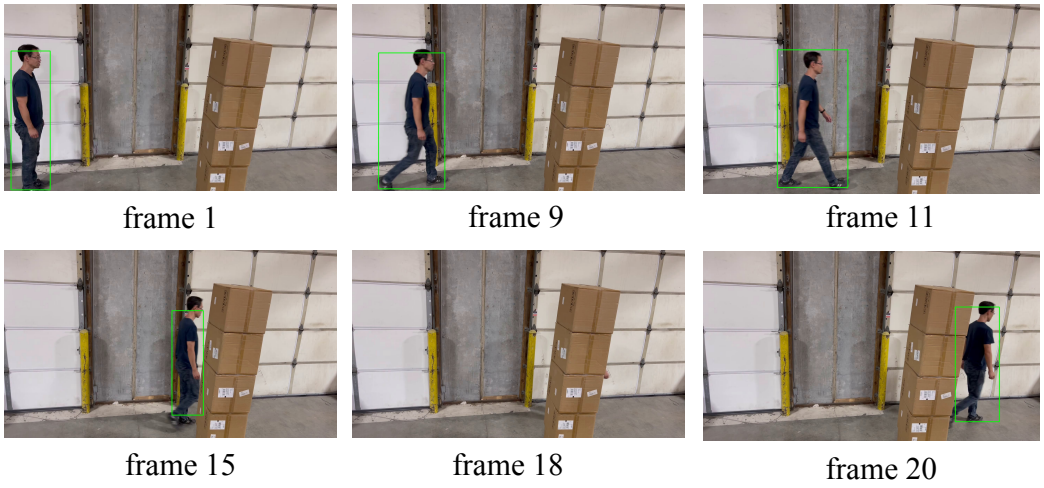

Figure 7.4: The results of different positions with the proposed method in second experiment

In Camshift algorithm, when the target is occluded by the box, the system stop to work due to the tracking target disappear. In the proposed method, the tracking results are obtained by the Camshift algorithm and CNN model. When the target didn't meet an

obstacle, the results located by the Camshift algorithm are accurate. However, when the target is occluded by a substance, the camshift algorithm doesn't work. The CNN model also cannot detect the target. Therefore, the system stored the information of the current tracking target such as the size of the tracking box, the direction of walking, and the histogram of the target. Then the system tries to detect the target in the direction of the walking based on the histogram of the target. As soon as the target appears in the picture, the system will track the target again.

The third experiment is similar to the second experiment. The difference is that the obstacle is another person. When the person walked behind another person, the system waits and detects the target on the route of the walk. When the target appears in the picture again, the system will continue to track the target. We use camshift algorithm and the proposed method to test the results respectively. Comparison tracking results of different positions with Camshift algorithm in third experiment are shown in Fig 7.5.

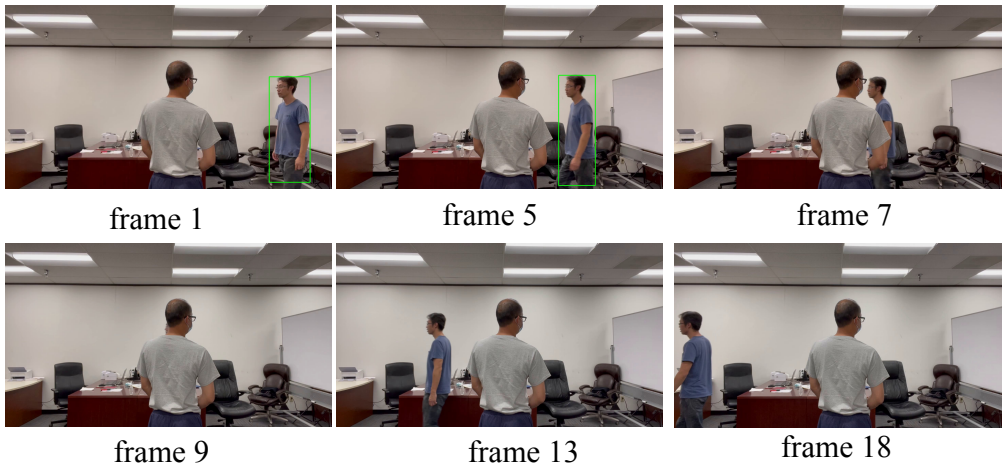

Figure 7.5: The results of different position with Camshift algorithm in third experiment

Comparison tracking results of different positions with the proposed method in third experiment are shown in Fig 7.6.

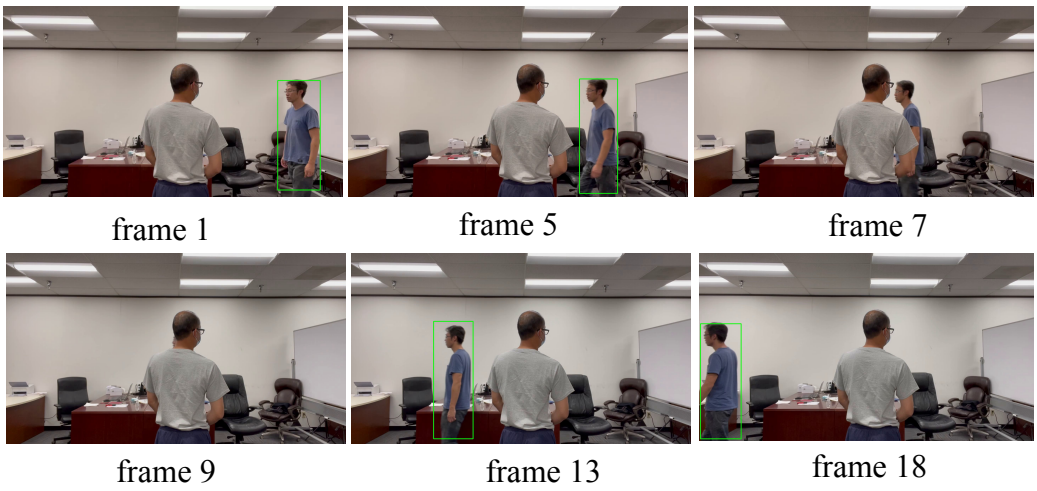

Figure 7.6: The results of different position with the proposed method in third experiment

In Camshift algorithm, when the target is occluded by another people, the system stop to work due to the tracking target disappear. In the proposed method, the tracking results are still obtained by the Camshift algorithm and CNN model. When the target didn't meet another person, the results located by the Camshift algorithm are accurate. However, when the target is occluded by the person, the camshift algorithm doesn't work. The CNN model also cannot detect tracking targets. The system will store the information of the current tracking target. Meanwhile, the system needs to record the information of another person. Then the system continues to explore the target in the direction of the walking based on the histogram of the target. As soon as the target appears in the picture, the system will track the target again.

7.3 Tracking Experiment for Two Targets

The fourth experiment is that the system will track two people at one time. In the video sample, two people walked across from each other. The system track two people at the same time. When a person is occluded by another person, the system detects the people who are occluded until he appears again. We use camshift algorithm and the

proposed method to test the results respectively. Comparison tracking results of different positions with Camshift algorithm in fourth experiment are shown in Fig 7.7.

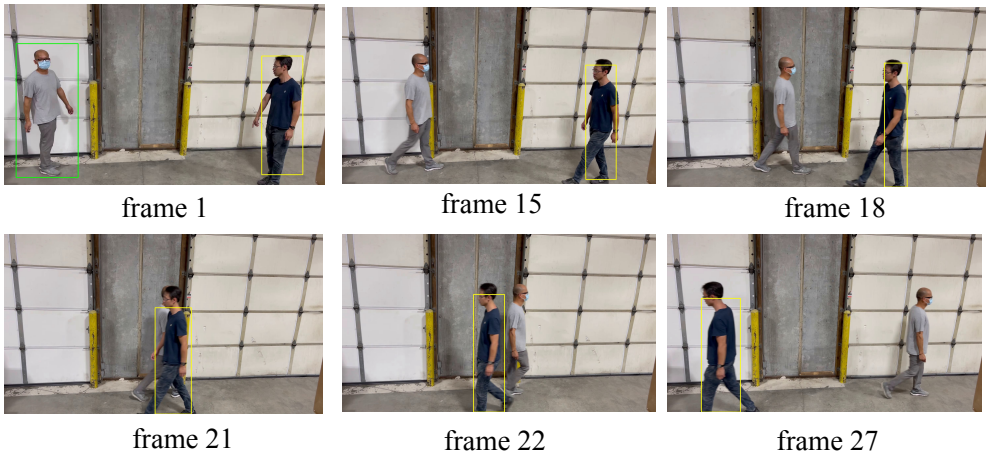

Figure 7.7: The results of different position with Camshift algorithm in fourth experiment

Comparison tracking results of different positions with the proposed method in fourth experiment are shown in Fig 7.8.

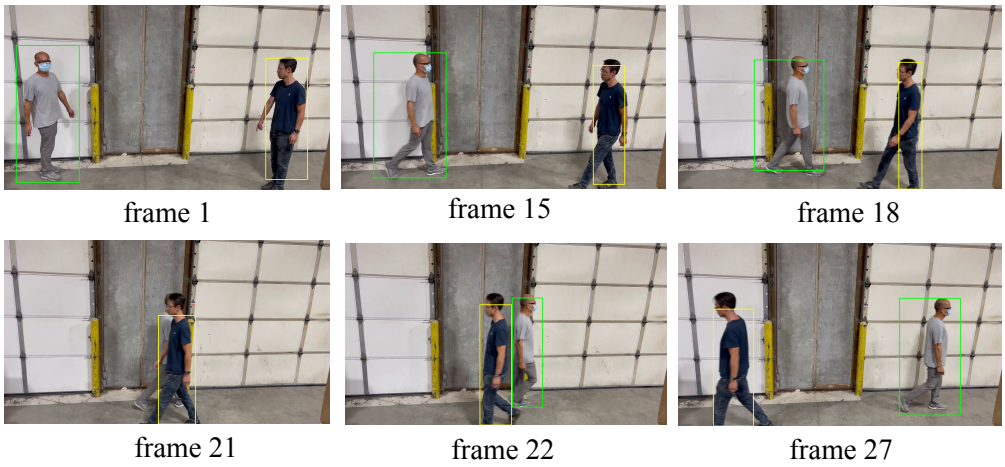

Figure 7.8: The results of different position with the proposed method in fourth experiment

In Camshift algorithm, when the left person is interrupted by the background, the system stop to work due to the tracking target has similar color with the background. In

the proposed method, the tracking results are obtained by the Camshift algorithm and CNN model. We use different labels to recognize different targets. When a target is occluded by another target. The system will wait and detect disappear target until it appears again. Meanwhile, another target is tracked by the system the entire time.

CHAPTER VIII:

CONCLUSION AND FUTURE WORK

We proposed a tracking system with Camshift algorithm and a CNN model. Camshift will do tracking most of the time. When Camshift algorithm is affected by environmental noise, the CNN model will correct the position of the tracking target. The CNN model also can help the system to detect the object and improve tracking accuracy. We design a CNN model with a relatively small number of parameters (68432) to detect a person. Our CNN model has 5 blocks. Each block is composed of 2 convolution layers. It is suitable for implementation in mobile devices, like Raspberry Pi or mobile phone. Results show camshift will lose track when the background color is close to the target. We use the average histogram to filter out the background included in camshift search. When the tracking target is blocked or two tracking targets cross-walk, the camshift algorithm cannot work. Then, our system can use the CNN model to help to track.

The system needs optimization in the future. Several tests and experiments have been left because of a lack of time. Future work concerns the design of neural networks for person feature extraction, new methods of person classification. This thesis has been mainly focused on the algorithm of Camshift, the classification model of CNN, real-time processing videos, leaving the study of thresholds of detecting person and so on outside the range of the thesis. The following ideas could be made in the future:

- The CNN model needs to be optimized because overfitting exists;
- The speed of image pre-processing needs to be improved;
- The accuracy of person detection should be improved.
- FPGA implementation

REFERENCES

[1] M. Satell, “Ultimate List of Drone Stats for 2021 | Philly By Air,” Jan. 15, 2021. https://www.phillybyair.com/blog/drone-stats/ (accessed Mar. 20, 2021).

[2] “Gartner Says Almost 3 Million Personal and Commercial Drones Will Be Shipped in 2017,” Gartner. https://www.gartner.com/en/newsroom/press-releases/2017-02-09-gartner-says-almost-3-million-personal-and-commercial-drones-will-be-shipped-in-2017 (accessed Mar. 20, 2021).

[3] A. Holmes, “How police are using technology like drones and facial recognition to monitor protests and track people across the US,” Business Insider. https://www.businessinsider.com/how-police-use-tech-facial-recognition-ai-drones-2019-10 (accessed Mar. 21, 2021).

[4] B. Yirka and T. Xplore, “Using artificial intelligence to help drones find people lost in the woods.” https://techxplore.com/news/2020-11-artificial-intelligence-drones-people-lost.html (accessed Mar. 22, 2021).

[5] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” Int. J. of Comput. Vision, vol. 60, no. 2, pp. 91–110, 2004.

[6] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in CVPR, 2005.

[7] R. Lienhart and J. Maydt, “An extended set of haar-like features for rapid object detection,” in ICIP, 2002.

[8] L. Nwosu, H. Wang, J. Lu, I. Unwala, X. Yang, and T. Zhang, “Deep Convolutional Neural Network for Facial Expression Recognition Using Facial Parts,”in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Comput-ing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Confon Big Data Intelligence and Computing and Cyber Science and TechnologyCongress

(DASC/PiCom/DataCom/CyberSciTech), Nov. 2017, pp. 1318–1321,doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.213.

[9] S. Liu and W. Deng, “Very deep convolutional neural network based im-age classification using small training sample size,” in 2015 3rd IAPR AsianConference on Pattern Recognition (ACPR), Nov. 2015, pp. 730–734, doi:10.1109/ACPR.2015.7486599.

[10] “ResNet-18.” https://kaggle.com/pytorch/resnet18 (accessed Aug. 03, 2020)

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the In-ception Architecture for Computer Vision,” arXiv:1512.00567 [cs], Dec. 2015,Accessed: Aug. 03, 2020. [Online]. Available: http://arxiv.org/abs/1512.00567.

[12] Brown, R. G. and P. Y. C. Hwang, Introduction to Random Signalsand Applied Kalman Filtering, in 1992 Second Edition, John Wiley andSons, Inc.

[13] G. Lakshmeeswari and K. Karthik, Survey on Algorithms for ObjectTracking in Video, 2016.

[14] G. Welch and G. Bishop, An Introduction to the Kalman Filter, p. 16,2006.

[15] Lu, Jiang, Ting Zhang, Qingquan Sun, Qi Hao, and Fei Hu. ”Binarycompressive tracking.” IEEE Transactions on Aerospace and ElectronicSystems 53, no. 4 (2017): 1755-1768.

[16] G. M. Rao and C. Satyanarayana, Visual Object Target Tracking Using Particle Filter: A Survey, International Journal of Image, Graphics andSignal Processing; Hong Kong, vol. 5, no. 6, pp. 5771, May 2013.

[17] KanadeLucasTomasi feature tracker, Wikipedia. 20-Dec-2018.

[18] A. Graves, GPU-accelerated feature tracking, in 2016 IEEE NationalAerospace and Electronics Conference (NAECON) and Ohio InnovationSummit (OIS), 2016, pp. 422429.

[19] N. Al-Najdawi, S. Tedmori, E. A. Edirisinghe, and H. E. Bez, Anautomated real-time people tracking system based on KLT featuresdetection, Int. Arab J. Inf. Technol., vol. 9, pp. 100107, 2012.

[20] G. Liu, S. Liu, K. Muhammad, A. K. Sangaiah, and F. Doctor, ObjectTracking in Vary Lighting Conditions for Fog Based Intelligent Surveil-lance of Public Spaces, IEEE Access, vol. 6, pp. 2928329296, 2018.

[21] KanadeLucasTomasi feature tracker, Wikipedia. 20-Dec-2018.

[22] Z. Chen, K. Wu, Y. Li, M. Wang, and W. Li, "SSD-MSN: An Improved Multi-Scale Object Detection Network Based on SSD," IEEE Access, vol. 7, pp.80622–80632, 2019, doi: 10.1109/ACCESS.2019.2923016.

[23] H. Nguyen, "Improving Faster R-CNN Framework for Fast VehicleDetection," Mathematical Problems in Engineering, Nov. 22, 2019.https://www.hindawi.com/journals/mpe/2019/3808064/(accessed Aug. 03,2020).

[24] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, and R.-S. Jia, "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications," IEEE Access, vol. 7, pp.133529–133538, 2019, doi: 10.1109/ACCESS.2019.2941547.

[25] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," 2018, ArXiv 2018, abs/1804.02767.

[26] T. W. Zhang, X. L. Zhang, J. Shi, and S. J. Wei, "Depthwise separable convolution neural network for high-speed SAR ship detection," Remote Sensing, vol. 11, no. 21, p. 2483, 2019.

[27] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," arXiv:1612.03144 [cs], Apr. 2017, Accessed: Apr. 10, 2021. [Online]. Available: http://arxiv.org/abs/1612.03144.

[28] Guy The AI. TheAIGuysCode/OIDv4_ToolKit. 2020. 2020. GitHub, July. 2020, [online] Available: https://github.com/theAIGuysCode/OIDv4_ToolKit.

[29] Z. Pan, S. Liu, and W. Fu, A review of visual moving target tracking, Multimed Tools Appl, vol. 76, no. 16, pp. 1698917018, Aug. 2017.

[30] J. Sun, A Fast MEANSHIFT Algorithm-Based Target Tracking System,Sensors; Basel, vol. 12, no. 6, pp. 82188235, 2012.

[31] S. G. Tzafestas, Introduction to Mobile Robot Control. Saint Louis,UNITED STATES: Elsevier, 2013.

[32] N. An, S. Sun, X. Zhao, and Z. Hou, Move like humans: End-to-endGaussian process regression based target tracking control for mobilerobots, in 2017 36th Chinese Control Conference (CCC), 2017, pp.69176921.

[33] R. Wang, L. Xiao-Dong, D.-X. Shi, and Z.-H. Tong, A Role BasedFramework for Robot Sensor Management, in ITM Web of Conferences;Les Ulis, Les Ulis, France, Les Ulis, 2017, vol. 12.

[34] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, ASurvey of Appearance Models in Visual Object Tracking, ACM Trans.Intell. Syst. Technol., vol. 4, no. 4, pp. 58:158:48, Oct. 2013.

[35] J. Sun, A Fast MEANSHIFT Algorithm-Based Target Tracking System,Sensors; Basel, vol. 12, no. 6, pp. 82188235, 2012.

[36] Jianhong Li, Ji Zhang, Zhenhuan Zhou, Wei Guo, Bo Wang, and QingjieZhao, Object tracking using improved Camshift with SURF method,in 2011 IEEE International Workshop on Open-source Software forScientific Computation, 2011, pp. 136141.

[37] R. Pandey and Neelendra Badal, “Understanding the Role of Parallel Programming in Multi-core Processor Based Systems,” Social Science Research Network,

Rochester, NY, SSRN Scholarly Paper ID 3350311, Mar. 2019. doi: 10.2139/ssrn.3350311.

[38] L. Wang et al., "A Unified Optimization Approach for CNN Model Inference on Integrated GPUs," arXiv:1907.02154 [cs], Jul. 2019, Accessed: Jul. 05, 2020. [Online]. Available: http://arxiv.org/abs/1907.02154. doi: 10.1145/3337821.3337839.

[39] X. Zhang, F. Reveriano, J. Lu, X. Fu, and T. Zhang, "The Effect of High Performance Computer on Deep Learning: A Face Expression Recognition Case," in 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Aug. 2019, pp. 40–42, doi: 10.1109/CSE/EUC.2019.00017.

[40] M. A. S. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, and R. Buyya, "HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges," ACM Comput. Surv., vol. 51, no. 1, pp. 1–29, Apr. 2018, doi: 10.1145/3150224.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. doi: 10.1145/3065386.

[42] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556 [cs], Apr. 2015. doi: 10.1109/ACPR.2015.7486599.

[43] C. Szegedy et al., "Going Deeper with Convolutions," arXiv:1409.4842 [cs], Sep. 2014. doi: 10.1109/CVPR.2015.7298594.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv:1512.03385 [cs], Dec. 2015. doi: 10.1109/CVPR.2016.90.

[45] Xiaodi Fu, J. Lu, X. Zhang, X. Yang, and I. Unwala, “Intelligent In-Vehicle Safety and Security Monitoring System with Face Recognition,” in 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Aug. 2019, pp. 225–229, doi: 10.1109/CSE/EUC.2019.00050.

[46] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” arXiv:1610.02357 [cs], Apr. 2017. doi: 10.1109/CVPR.2017.195.

[47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” arXiv:1506.02640 [cs], May 2016. doi: 10.1109/CVPR.2016.91.

[48] W. Liu et al., “SSD: Single Shot MultiBox Detector,” arXiv:1512.02325 [cs], vol. 9905, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” arXiv:1311.2524 [cs], Oct. 2014. doi: 10.1109/CVPR.2014.81.

[50] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” arXiv:1506.01497 [cs], Jan. 2016. doi: 10.1109/TPAMI.2016.2577031.

[51] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv:1804.02767 [cs], Apr. 2018. doi: 10.4236/ce.2012.33059.

[52] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” arXiv:1411.4038 [cs], Mar. 2015. doi: 10.1109/TPAMI.2016.2572683.

[53] G. Sharma, D. Liu, E. Kalogerakis, S. Maji, S. Chaudhuri, and R. Mvech, “ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds,” 2020, doi:

10.1007/978-3-030-58571-6_16.

[54] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” arXiv:1505.04597 [cs], May 2015. doi: 10.1007/978-3-319-24574-4_28.

[55] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” arXiv:1612.03144 [cs], Apr. 2017. doi: 10.1109/CVPR.2017.106.

[56] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid Scene Parsing Network,” arXiv:1612.01105 [cs], Apr. 2017. doi: 10.1109/CVPR.2017.660.

[57] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” arXiv:1703.06870 [cs], Jan. 2018. doi: 10.1109/TPAMI.2018.2844175.

[58] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation,” arXiv:1802.02611 [cs], Aug. 2018. doi:10.1007/978-3-030-01234-2_49.

[59] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path Aggregation Network for Instance Segmentation,” arXiv:1803.01534 [cs], Sep. 2018. doi: 10.1109/CVPR.2018.00913.

[60] H. Zhang et al., “Context Encoding for Semantic Segmentation,” arXiv:1803.08904 [cs], Mar. 2018. doi: 10.1109/CVPR.2018.00747.

[61] L. Nwosu, H. Wang, J. Lu, I. Unwala, X. Yang, and T. Zhang, “Deep Convolutional Neural Network for Facial Expression Recognition Using Facial Parts,” in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), Nov. 2017, pp. 1318–1321, doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.213.

[62] S. Liu and W. Deng, “Very deep convolutional neural network based image classification using small training sample size,” in 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Nov. 2015, pp. 730–734, doi: 10.1109/ACPR.2015.7486599.

[63] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[64] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” arXiv:1512.00567 [cs], Dec. 2015, Accessed: Aug. 03, 2020. [Online]. Available: http://arxiv.org/abs/1512.00567. doi: 10.1109/CVPR.2016.308.

[65] Z. Chen, K. Wu, Y. Li, M. Wang, and W. Li, “SSD-MSN: An Improved Multi-Scale Object Detection Network Based on SSD,” IEEE Access, vol. 7, pp. 80622–80632, 2019, doi: 10.1109/ACCESS.2019.2923016.

[66] H. Nguyen, “Improving Faster R-CNN Framework for Fast Vehicle Detection,” Mathematical Problems in Engineering, Nov. 22, 2019. https://www.hindawi.com/journals/mpe/2019/3808064/ (accessed Aug. 03, 2020). doi: 10.1155/2019/3808064.

[67] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, and R.-S. Jia, “Mini-YOLOv3: Real-Time Object Detector for Embedded Applications,” IEEE Access, vol. 7, pp. 133529–133538, 2019, doi: 10.1109/ACCESS.2019.2941547.

[68] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, “Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications,” IEEE Access, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.