A PEDESTRIAN TRACKING SYSTEM

by

BOFU LIU

(Under the Direction of Hamid R. Arabnia)

ABSTRACT

Pedestrian detection and tracking are very popular in image processing and computer vision areas. The main purpose of this paper is to compare different pedestrian detection approaches and to build a pedestrian tracking system. This paper trains two different detectors one is based on Histogram of Oriented Gradient feature combined with Support Vector Machine and another one is based on a neural network that imitates the Faster R-CNN then compares their detection results. Both detectors are trained on a dataset that combines part of the INRIA dataset and images taken on the outside of campus. It includes people with different postures, walking in any directions and brightness. And this paper also completes a pedestrian re-identification function using GrabCut segmentation technique in HSV color space to recognize same people. At last, this paper achieves tracking a specific target in a video by using Kernelized Correlation Filters (KCF).

INDEX WORDS: pedestrian detection, pedestrian tracking, support vector machine, Histogram of Oriented Gradient (HOG), Faster R-CNN, GrabCut, HSV color space, Kernelized Correlation Filters (KCF)

A PEDESTRIAN TRACKING SYSTEM

by

BOFU LIU

BS, BEIJING UNIVERSITY OF TECHNOLOGY, CHINA, July 2015

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2017

A PEDESTRIAN TRACKING SYSTEM

by

BOFU LIU

Major Professor:	Hamid R. Arabnia
Committee:	Kyu Hyung Lee
	Yi Hong

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
December 2017

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

With the increase of public awareness of safety and the prevention of their own security, pedestrian tracking system gains multitudinous concern from the public. This system can not only automatically detect and track pedestrians, but also reduces the cost of letting people monitor the videos all the time. Due to this reason, pedestrian tracking system has been put into use in abundant places, including the parking lot, subway station, exhibition, and laboratory. Especially, pedestrian tracking system plays an irreplaceable role in some specific places. In the bank or mall, it provides customers' movement information to the police department to prevent crime; on the street or in the parking lot, it analyses pedestrians' actions and routes to prevent traffic accidents and helps planning traffic; in people's own houses, it alarms the police when detecting illegal intruders. Furthermore, as the development of autopilot cars, the pedestrian tracking system can not only recognize the sudden appearance of pedestrians and by tracking their tracks, the car system is able to predict pedestrians' future locations to calculate the best avoiding solution. For those advanced applications, human detection and tracking are two of the most basic and core technologies.

Human detection not only is a popular topic in computer vision and pattern recognition areas but also gets a wide range of applications in the artificial intelligence area. In today's research, human detection uses machine learning technologies based on three different categories: 1) template match, 2) human modeling, 3) statistical

classification. Template match and human modeling are too complex and involve a variety of unpredictable variables. Statistical classification achieves a better result by extracting features from target then apply pattern recognition to classify those features among all three methods. Histogram of Oriented Gradient (HOG) is one of the most widely used features. Although HOG feature descriptor has a large number of dimensions that causes huge calculations, it describes both the gradient intensity and the distribution of the gradient direction in partial areas of the image. Meanwhile, with the development of deep learning, Convolutional Neural Network has become another popular technology for human detection. This paper also trains a human detector based on a neural network which imitates the Faster R-CNN (Faster Region-based Convolutional Network) [1]. And the results of using HOG with SVM and using the neural network to detect pedestrian are compared in this paper.

Pedestrian tracking is also a highly focused and very challenge problem owing to the complexity of body movement and background, light change in the video and different types of features of the human body. Nowadays, there are four widely used approaches for human tracking: 1) region-based tracking, 2) contour-based tracking, 3) feature-based tracking, 4) model-based tracking. However, its low computational efficiency makes it difficult to process real-time videos, since it requires more parameters and calculations when compared with other algorithms. To increase the computational efficiency, the Kernelized Correlation Filter (KCF) builds training samples by applying cyclic shifts to all the original samples and imports circulant matrices to simplify the calculations [2]. KCF shows higher efficiency and accuracy rate when compared with TLD and Struck [3], [4].

This paper puts forward a pedestrian tracking system which has three important components. The first part detects pedestrian using two different methods. One is a statistical classification method that uses motion-based detection to detect moving objects from the constant background, then trains a linear SVM classifier based on HOG feature to recognize pedestrian among those detected objects. Another method is to build a Faster Region-based Convolutional Neural Network to detect the pedestrian. The second part is pedestrian re-identification that uses the same label to mark the same person when this person comes into the video again by using automatic GrabCut segmentation technique [5] [6] in HSV color space. The third part tracks pedestrian by applying KCF to a specific target to achieve fast speed tracking.

The rest of the paper has four parts and are organized as follows. Section 2 presents related works of human detection and pedestrian tracking. Section 3 presents basic works of the system including the structure of the system and the database. Section 4 presents people detection using HOG and SVM. Section 5 presents people detection using Faster R-CNN. Section 6 presents pedestrian re-identification. Section 7 presents tracking using Kernelized Correlation Filters. Section 8 presents the results. At last, section 9 presents the conclusions and future work.

CHAPTER 2

RELATED WORKS

2.1 Moving Object Detection

It is known to all that the first and most important step for pedestrian tracking is human detection. It separates pedestrians from a complex background and gets their accurate locations. Although detection is on the lowest level of the whole tracking system, the accuracy of tracking and other functions is depending on this part. Since human detection is based on body features such as, aspect ratio, square measure, and gradient feature etc., and surroundings around targets are both changeable and various, the difficulty of detection is highly increased. In present, background subtraction [7], temporal differencing [8] and optical flow [9] are three basic detection methods with high effectiveness and feasibility under statistic background situation.

2.1.1 Background Subtraction

Background subtraction is a normally used method for moving object segmentation under constant background condition. It uses the difference of gray value between the current frame and the background frame to detect moving areas. Firstly, build a background model by using previously stored or real-time updated background images to do the statistical modeling on every pixel. Then calculate the changing pixels when comparing current image with the background model. And areas made up of those changed pixels are considered as foreground objects.

Set $B(x, y)$ as the background pixel value at location (x, y), $C(x, y)$as the pixel at location (x, y) in the current frame, T is a threshold value which indicates the boundary between foreground pixels and background pixels. And the value of T is decided by the background model. So the foreground pixel $F(x, y)$, should meet the requirement in following:

$$F(x, y) = \begin{cases} 1, |C(x, y) - B(x, y)| > T \\ 0, |C(x, y) - B(x, y)| \leq T \end{cases} \quad (2-1)$$

However, the environment is always changing, for example, the illumination conditions are different from each period, leaves or flags fly with the wind. Those changes influence the accuracy of background segmentation. To eliminate all the influencing factors as much as possible and get the most accurate background model, several approaches are presented. The most used methods are statistical averaging [10], W4 [11] and linear prediction [12].

How those methods work are described in the following:

(1) Statistical Averaging:

The best condition for statistical averaging is moving objects with a simple background. There are two mainly used filters for getting the background: median filter and mean filter. Median filter finds the median value of multiple background images and mean-filter calculates the mean of multiple background images. Set B as the background model and C as each frame, k is the order of every frame and N is the number of frames used to get the background model. In the following is the equation of mean filter:

$$B = \frac{1}{N}(C_k + C_{k-1} + C_{k-2} + \cdots + C_{k-N+1}) \quad (2-2)$$

(2) W4:

W4 is a simple background model algorithm which applies minimum grayscale value, maximum grayscale value and maximum time difference to model every pixel in the scene. By updating the background periodically, it can detect multiple moving objects from complex outdoor scenes like the shopping mall, square and parking lot. But W4 is very sensitive to light changing, it is necessary to do regional subtraction and morphological filtering to reduce the influence.

(3) Linear Prediction:

Linear prediction predicts the future background image in the next frame based on pixel values in previous frames. While this makes linear prediction unable to apply to the real-time system because the predictor of the filter is based on sampling covariance on every frame due to huge calculation cost.

2.1.2 Temporal Differencing

Temporal differencing uses the difference between the corresponding pixels of two or three consecutive images to calculate the motion area. It detects moving pedestrian by assuming that the gray value of every pixel is a constant value. The requirement for being a foreground pixel which is similar to the requirements in equation 2-1are in the following:

$$\mathrm{F(x,y)} = \begin{cases} 1, |f_{i+1}(x,y) - f_i(x,y)| > T \\ 0, |f_{i+1}(x,y) - f_i(x,y)| \leq T \end{cases} \qquad (2-3)$$

T is a pre-set threshold value, $\mathrm{F(x,y)} = 1$ is considered as foreground or brightness changing, $f_{i+1}(x,y) - f_i(x,y)$ is the absolute difference between two

continuous frames. This approach has less calculation cost when compared to other method and can adapt to dynamically changed scenes. However, it cannot detect the whole body of moving objects that means many empty holes may be left in the motion area, especially for objects that move slow and has similar overall pixel values. Temporal differencing demands constant background or at least minor changing that induce edge broken by moving objects whose contrast with the background is not that strong. Also, it would increase the difficulty of the subsequent recognition and tracking. Furthermore, it is impossible to detect pedestrian if this pedestrian is quiescent.

2.1.3 Optical Flow

Optical flow is a kind of instantaneous velocity field that contains object moving information. It describes the velocity distribution of each pixel in the image. Optical flow focuses on the relationship between the change of every pixel gray value and the velocity distribution. When detecting the target, it gives each pixel an initial velocity vector and analyses the image dynamically based on current velocity distribution. Then when the target shows up in the image, the velocity vectors of the target can be easily distinguished from background velocity vectors. But if the optical flow in the entire image is constantly changing, it means no moving object is in the image.

Although optical flow is suitable not only for the static or dynamic background but also on background videoed by moving camera, it has complex calculation and unable to meet the requirements of real-time recognizing. At the same time, the analyzation of optical flow can be affected by noise, shadow, and blocking which makes it be adequate for the target with slow speed and image with less noise.

2.2 Pedestrian Tracking Algorithms

Object tracking is also another important problem after pedestrian detection that makes up the insufficient of pedestrian detection. The basic concept for tracking algorithm is to predict the future result based on the current result. And tracking algorithms are classified into different categories based on various strategies.

2.2.1 Model-based Tracking

Model-based tracking algorithm uses a straight line, projection or three-dimensional models to describe details of the human structure. And it must build a human 3-dimensional model before tracking starts. Then it predicts the future posture model and calculates the similarity between the predicted model and actual posture model to get the human posture in the current frame. By analyzing the posture of pedestrian, this method can track pedestrian no matter how the posture changes. However, building 3-dimensional model would cost a lot of time and its accuracy is highly depends on how accurate the human model is. It is difficult to achieve real-time tracking pedestrian using model-based tracking.

2.2.2 Region-based Tracking

This tracking method considers each portion of the human body as multiple areas which is composed of the head, trunk, and limbs. Each part is corresponding to many different areas. Region-based tracking builds models for the human body and background so that pixels that belong to human body would be divided into different body parts. And it tracks human by tracking each body part. But its performance is restricted when the target object is partially eclipsed.

2.2.3 Contour-based Tracking

Contour-based tracking uses closed contour to describe moving pedestrian and the contour can be updated automatically and continuously. Unless the Region-based tracking, this method can keep tracking the target even if the target is partially covered by something, yet the initialization is highly difficult.

2.2.4 Feature-based Tracking

Feature-based tracking extracts several kinds of features from the target and lets features in one image correspond to another image. Then it compares the similarity between object's feature set in the current image and target's feature set in the reference image. If the similarity is very high, then this object is the tracking target. It reduces the influence of target covering, while the robustness of the algorithm depends on feature extraction.

Another way for feature-based tracking is template matching. Firstly, saving a target template as the standard for recognition. Then calculate the similarity with every interested area in the current image. The area with the highest similarity is considered as the location of that target in the current image.

CHAPTER 3

BASIC WORKS

3.1 Workflow

The pedestrian tracking system presented in this paper has three basic parts including pedestrian detection based on SVM and Neural Network, pedestrian re-identification, pedestrian tracking based on Kernelized Correlation Filters (KCF). Figure 1 shows the system structure.

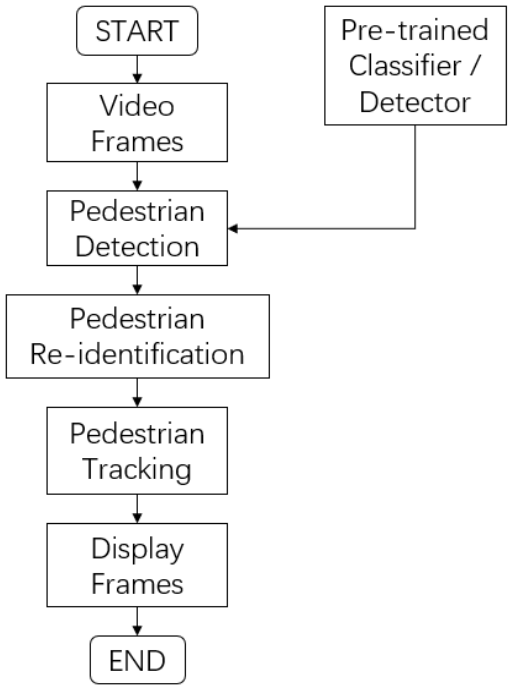


Figure 1 System workflow. The main structure of the whole pedestrian tracking system that contains pedestrian detection, re-identification, and tracking parts.

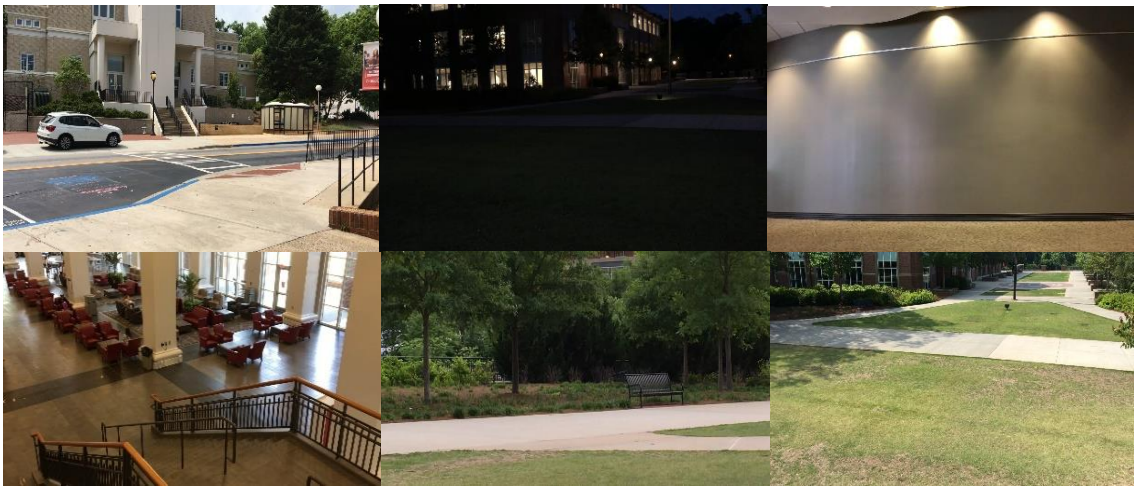
3.2 Dataset

The dataset contains 3,990 images in total which includes 2,442 positive images and 1,548 negative images. 1,917 training images are from INRIA dataset, and rest of the 2,073 images consist of 1,812 positive images and 261 negative images that photographed own my own. Those 1,812 images are photographed under different situations which are listed in the following. (1) Different filming locations: lawn, roadway, corridor, and stairs; (2) light intensity: filming during daytime, sunset, indoor light, nighttime with different degrees of brightness; (3) pedestrian positions: different filming angles, walking in various directions and running. Since pedestrian tracking system would be much more necessary during nighttime. I believe by adding both negative and positive images taken at night into the dataset, the system would adapt to night environment and be able to detect and track pedestrian during the night. Figure 2 exhibits some example images from the dataset.

(a) Positive samples from INRIA dataset.

(b) Negative samples from INRIA dataset.

(c) Self-taken positive samples.



(d) Self-taken negative samples.

Figure 2 Examples of the dataset. (a) and (b) show the example images from INRIA dataset. Rest of the images in (c) and (d) are examples taken by me, including indoor, outdoor, daytime, midnight and different camera angles.

CHAPTER 4

PEDESTRIAN DETECTIONS WITH HOG AND SVM

Nowadays, two most popular technologies for pedestrian detection are Histogram of Oriented Gradient features combined with Support Vector Machine and Deep Learning using the convolutional neural network. This section focuses on using HOG feature to detect pedestrian from the moving objects detected from the video and can be separated into two steps, movement detection, and pedestrian recognition. Figure 3 shows the basic steps.

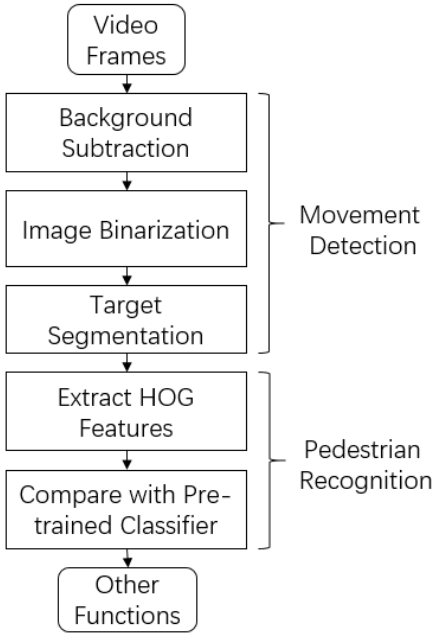


Figure 3 Structure of motion-based pedestrian detection. It contains the movement detection and pedestrian recognition as two main parts.

4.1 Movement Detection

Movement detection segments foreground from the background. Assuming background is static so that any meaningful object movement would be considered as foreground. The system trains a background model based on Gaussian Mixture Model to detect foreground objects. After getting the information of moving objects, HOG features are extracted from every moving object and compared with a pre-trained classifier to recognize pedestrians among them.

Gaussian Model is a background based statistical model under the category of background subtraction. It is widely used in background modeling and can be divided into two categories: Gaussian Single Model and Gaussian Mixture Model. Gaussian Single Model applies to the background only when the color distribution is more concentrated and can be described by a probability distribution model. But when the color distribution is dispersed, instead of Gaussian Single Model, Gaussian Mixture Model which uses multiple distribution models describing the background together would be more appropriate. Meanwhile, it allows moving objects to exist in the background during modeling and can handle backgrounds that are not completely stationary, like flag or leaves that flying with wind and water ripples. For example, when a leaf is swinging back and forth with the wind, it leaves its position rapidly. Pixels' information when this leaf leaves this position is described by a Gaussian Model, meanwhile, pixels' information when this leaf comes back is also described by another gaussian model. So that pixel in the new frames would be considered as background no matter which gaussian model they match with. This increases the robustness of the model by preventing the model from seeing leaf movement as foreground.

Basic steps of Gaussian Mixture Model:

(1) Definition of pixel model:

Assume K is the number of models contained in the Gaussian Mixture Model that is in the range of 3 to 5, (x, y) is pixel location, t is a serial number of every frame and 0 is the first frame. $w_i(\mathrm{x},\mathrm{y},\mathrm{t})$ is weight, $u_i(\mathrm{x},\mathrm{y},\mathrm{t})$ is mean value, $\sigma_i(\mathrm{x},\mathrm{y},\mathrm{t})^2$ is the variance of each model and must satisfy $\sum_{i=1}^{K} w_i(\mathrm{x},\mathrm{y},\mathrm{t}) = 1$.

Since each pixel is described by multiple models, then pixel p can be described as $\mathrm{P(p)} = \{[w_i(\mathrm{x},\mathrm{y},\mathrm{t}), u_i(\mathrm{x},\mathrm{y},\mathrm{t}), \sigma_i(\mathrm{x},\mathrm{y},\mathrm{t})^2]\}, \mathrm{i} = 1, 2, 3 \ldots, \mathrm{k}$.

(2) Updating parameters and foreground detection:

Step 1: if the pixel value at location (x, y) in current frame satisfies

$$|\,\mathrm{I}(\mathrm{x},\mathrm{y},\mathrm{t}) - u_i(\mathrm{x},\mathrm{y},\mathrm{t})| \le \lambda \cdot \sigma_i(\mathrm{x},\mathrm{y},\mathrm{t}) \qquad (4-1)$$

then this pixel matches the model which means it is background pixel and moves into step 2. Otherwise, this pixel is a foreground pixel and moves into step 3.

Step 2: correcting the weight of the model. Weight increment is defined as

$$\mathrm{dw} = \alpha \cdot \big(1 - w_i(\mathrm{x},\mathrm{y},\mathrm{t}-1)\big) \qquad (4-2)$$

α is the update rate which is manually set and makes the model to have robustness when the background changes very slowly.

And new weight would be

$$w_i(\mathrm{x},\mathrm{y},\mathrm{t}) = w_i(\mathrm{x},\mathrm{y},\mathrm{t}-1) + dw \qquad (4-3)$$

$$= w_i(\mathrm{x},\mathrm{y},\mathrm{t}-1) + \alpha \cdot \big(1 - w_i(\mathrm{x},\mathrm{y},\mathrm{t}-1)\big) \qquad (4-4)$$

Turn into Step 4 directly.

Step 3: if the new pixel does not match any of the models, then create a new model with a small weight and its mean value equals to this new pixel value. If the number of models has already reached the maximum number, then delete the model which has the lowest importance.

Step 4: normalize weight.

$$w_i(x,y,t) = \frac{w_i(x,y,t)}{\sum_{j=1}^{k} w_i(x,y,t)}, (i = 1,2,\dots,k) \qquad (4-5)$$

(3) Calculate the importance of Gaussian Models:

Because a background model should have a large weight which means a high probability of being background and low variance which means minimum pixel value change. The importance of a model can be defined as $\mathrm{I} = \frac{w_i(x,y,t)}{\sigma_i(x,y,t)}$. Sort all the models by their importance from large to small. Furthermore, set a threshold value T to delete models with importance lower than T would increase the efficiency.

After getting the background model, the system does Image Binarization on every video frame. It compares the pixel values in the current frame with the background model to find foreground pixels and a binary mask with the same size as current frame is created. In this mask, values of foreground pixels are set to 1 which is white and rest of the pixels that are considered as background are set to 0 which is black. Since all the detected objects are described as white irregular areas on the mask, boundary boxes that contain each object are created. Assume the location of the leftmost pixel, the rightmost pixel, the highest pixel and the lowest pixel of the object area are defined as following: (x_l, y_l), (x_r, y_r), (x_t, y_t) and (x_b, y_b). The height and width of the box are $\mathrm{H} = |x_b - x_t|, \mathrm{W} =$

$|y_r - y_l|$. The location of the top-left pixel on the boundary box is (x_t, y_l). Figure 4 shows an example of getting the boundary boxes. The white area indicates the moving object and black area is background. Four red dots on the yellow boundary box indicate the marginal pixels and the blue dot is the top-left pixel of the boundary box. With the boundary boxes, the system segments every object from the frame called Target Segmentation and prepare them for the pedestrian classification.

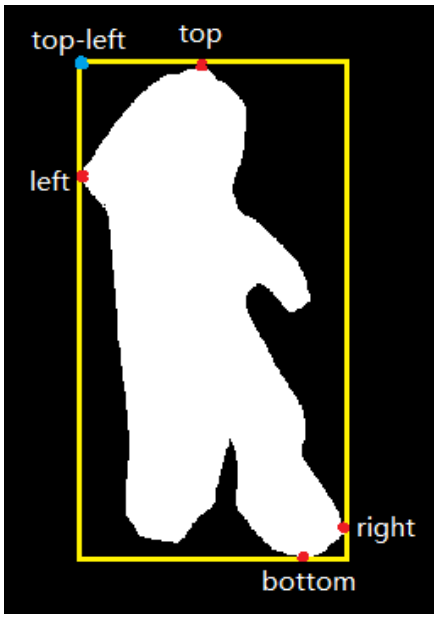


Figure 4 Boundary box on the mask. The white area contains the foreground pixels and the black area contains the background pixels. The initial location of the boundary box is the top-left pixel, width and height of the boundary box are decided by the location of the most top, bottom, left and right pixels.

4.2 Pedestrian Classification

Movement detection returns boundary boxes and by segmenting areas enclosed in those boxes, the system applies a pre-trained classifier to recognize which of them contain pedestrian. If an area is determined as having pedestrian in it, its information would be stored in the system. To do so, there are two steps: 1) feature extraction, extracting a specific feature from target region which makes this region different from other regions; 2) training classifier based on extracted feature to find an appropriate dividing line that separates targets from rest of the objects. Among current methods, HOG feature combined with Support Vector Machine (SVM) is one of the most effective and balanced approaches.

4.2.1 Feature Extraction

HOG feature has many advantages when compared with other features. On one hand, HOG is operated on the local grid cells which makes it maintain good invariance with both geometric and optical deformation of the image. On the other hand, it allows pedestrian has minor body movements if they can maintain an upright posture. The extraction steps are as follows:

(1) Changing the original images into grey scale images.

Normalization can adjust the image contrast, suppress noise and reduce the influence of shadow and light changes in the image. Suppose $\mathrm{I}(x, y)$ is the pixel value at (x, y), Max is the maximum value of all the pixels and Min is the minimum value of all the pixels, the actual pixel value after normalization is:

$$I(x, y) = (I(x, y) - Min)\frac{255}{(Max - Min)} \qquad (4-6)$$

(2) Applying filter $[-1, 0, 1]$ and $[-1, 0, 1]^T$ on the image to calculate the horizontal gradient $G_x(x, y)$ and the vertical gradient $G_y(x, y)$.

$$G_x(x, y) = I(x + 1, y) - I(x - 1, y) \quad (4 - 7)$$

$$G_y(x, y) = I(x, y + 1) - I(x, y - 1) \quad (4 - 8)$$

(3) Calculating the gradient $\mathrm{G}(x, y)$and the gradient direction $\alpha(x, y)$.

$$\mathrm{G}(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2} \quad (4 - 9)$$

$$\alpha(x, y) = tan^{-1}\left(\frac{G_y(x, y)}{G_x(x, y)}\right) \quad (4 - 10)$$

(4) Creating gradient histogram in a cell.

Firstly, divide the image into cells with size 8×8 and $[-\pi/2\ \pi/2]$ degree into 9 bins with 20° in each bin. Then calculate the histogram in every cell based on each pixel's gradient value in every bin which is a 9-dimensional feature vector. Figure 5 shows how the bins are divided into a cell. Suppose the count of bin 8 would plus 1 if the gradient direction of a pixel is 35°.

(5) Calculating feature vector in a block.

Considering every 4 adjacent cells as a block. So that by putting features of all 4 cells together, a 36-dimensional feature vector can be gained as the feature vector for a block.

(6) Collecting the HOG feature.

Scanning the input image using the block cell by cell and putting the feature vector of all blocks together to obtain the HOG feature of the target. Assuming, there is an image with size 64×128 and every 8×8 pixels

make up a cell. Each cell has 9 features and each block contains 4 cells that lead to 36 features in a block. Set scan step to 8 pixels, so the number of features is $36 \times (64 \div 8 - 1) \times (128 \div 8 - 1) = 3780$. Figure 6 presents an illustration of HOG feature.

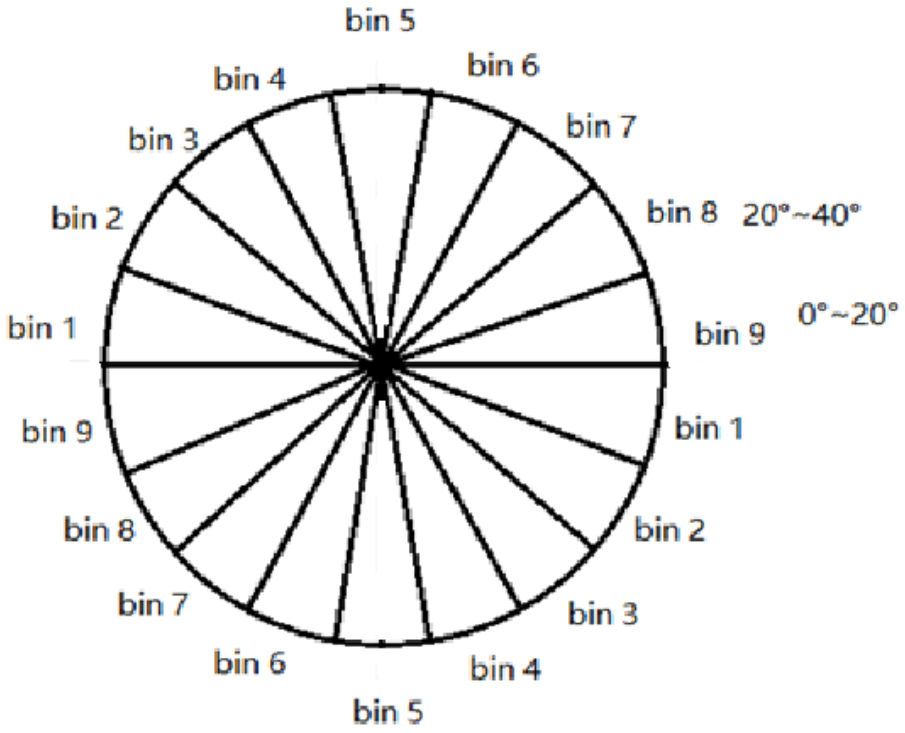


Figure 5 Gradient directions. In each cell, $[-\pi/2\ \pi/2]$ is divided into 9 bins with 20° in each bin. For example, if the direction is 10° which is in the range of 0° to 20°, then it belongs to bin 9.

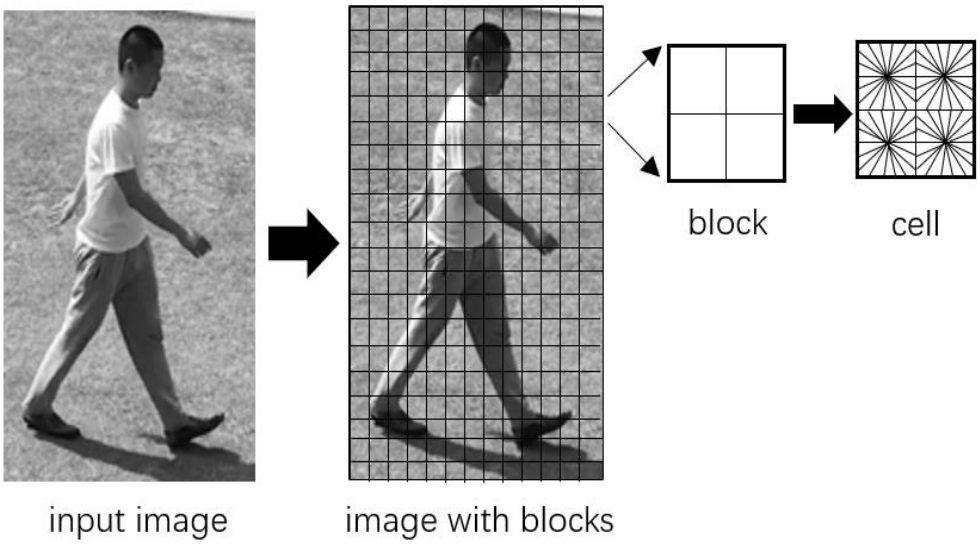


Figure 6 Illustration of HOG feature. The image is firstly divided into a lot of blocks, each block contains four cells, each cell is an 8×8 matrix that has 64 pixels, and those pixels generate 9 features.

4.2.2 Training Classifier

The classifier is trained with a huge set of samples to recognize the category of each unknown object. For the sample set, it consists of a set of sample feature and two labels including positive label and negative label. Support Vector Machine is the most famous approach for training a classifier. The goal of SVM is converting the problem to linear problem, then find the best distinguish line. Its advantages are as following:

(1) Low risk and good suitability.

(2) Less influenced by the feature dimension.

(3) The non-linear problems can be converted to the linear problems.

(4) Various model types that suitable for many traditional methods.

Pedestrian recognition is a simple linear classification problem, so a linear SVM $f(x) = x\beta + b$ is chosen as the kernel. Set the label of positive samples as 1 and the label of negative samples as 0. Then input both positive samples and negative samples with their labels into the SVM to train a classifier.

CHAPTER 5

PEDESTRIAN DETECTIONS WITH FASTER R-CNN

Faster R-CNN is a regional convolutional neural network but has shorter running time than R-CNN and Fast R-CNN [13]. It uses a Regional Proposal Network (RPN) which is a nearly cost-free fully convolutional network because RPN shares the same full-image convolutional features with the detection network. Furthermore, Faster R-CNN and RPN share their convolutional features by using “attention” mechanism also reduces the running time.

The steps for training a Faster R-CNN are [1]:

(1) Training RPN and initialized with ImageNet model.

(2) Initializing the parameters based on ImageNet model and use the RPN trained in step 1 to generate proposals. Then train a Fast R-CNN by setting those proposals as the input.

(3) Using the Fast R-CNN from step 2 to initialize a new RPN. But set the learning rate in conv layers which are shared by Fast R-CNN and RPN to 0. So that only those layers specific to RPN are updated. Then retrain the updated RPN.

(4) Combing layers that are specific to Fast R-CNN with shared layers to create a unified network. Then train this unified network. After training, this network would be able to detect the target.

Figure 7 shows the basic structure of Faster R-CNN. There are four main parts in the structure: conv layers, RPN, ROI pooling, and classification.

Each part has its own functions:

(1) For conv layers, it consists of a basic combination of conv + relu + pooling layers to extract image feature maps which are shared by RPN layers and fully connected layers.

(2) As for RPN, it generates region proposals. After getting the feature maps, there are two branches. One is using Softmax to classify anchors are either foreground or background. Another line calculates the bounding box regression deviation. At the proposal layer, it combines foreground anchors and bounding box regression deviation to get more accurate proposals and eliminate proposals which are too small or out of boundary.

(3) For ROI pooling layer, it collects feature maps and proposals to extract proposal feature maps.

(4) Classification distinguishes the category of proposals using proposal feature maps and gets the final location of the detection window.

Since Faster R-CNN is such a powerful neural network, I built a neural network that imitates Faster R-CNN. And the structure is shown in Figure 8. Because in my dataset all the objects are larger than 16×16, so the input size is set to 32×32. And there are 32 filters of size 3×3. A max pooling layer is created with pool size 2×2 and stride 2×2.

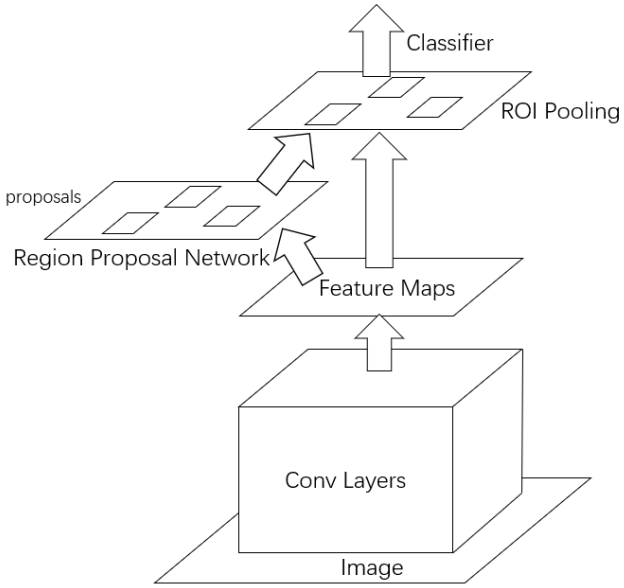


Figure 7 Basic structure of Faster R-CNN. Faster R-CNN is a single, unified network and the Region Proposal Network serves as the 'attention' of this unified network

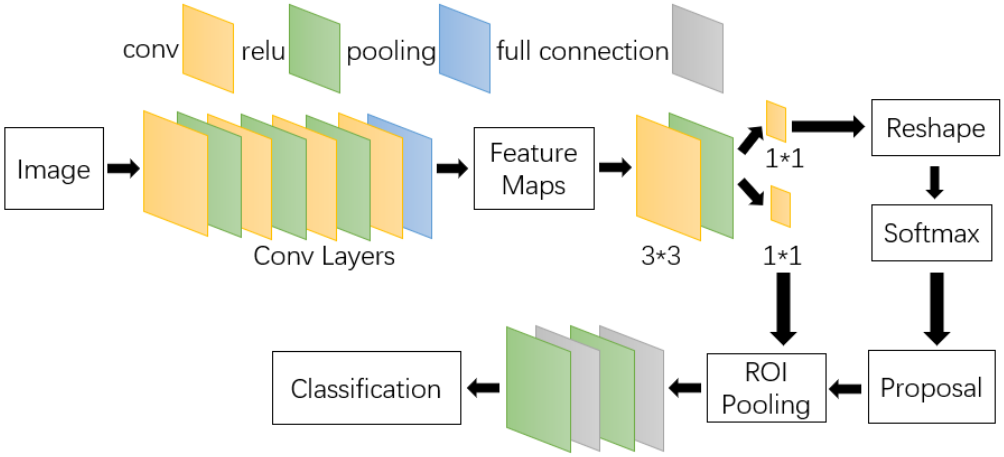


Figure 8 Workflow of my neural network. This network has three convolutional layers and the training process is learned from Faster R-CNN.

CHAPTER 6

PEDESTRIAN RE-IDENTIFICATION

Pedestrian Re-identification is to identify a pedestrian who has disappeared from the video for a while then comes back into the video again. Due to actual environment, different camera heights and pedestrian postures, re-identification have several problems. It is hard to recognize faces in the video, since sometimes people may show their back to the camera, their facial feature changes, and multiple people may have similar facial feature. But most people do not wear same color clothes that makes re-identification using cloth color become possible. To re-identify pedestrian, the system firstly extracting the pedestrian from the background and creating the color histogram of images then compares every histogram to find the most similar pair.

6.1 Pedestrian Extraction

The background is one of the adverse elements that affect the accuracy of pedestrian re-identification. Because background pixels are automatically included when using HSV histogram to identify pedestrian clothes color. To reduce the impact of background, extracting the pedestrian from the background is necessary. This paper uses an iterative image segmentation algorithm that called GrabCut which combines Graph Cut algorithm and statistics to segment foreground from the background. As shown in Figure 9 the pedestrian that is considered as the foreground is segmented from the image and the background is set to black.

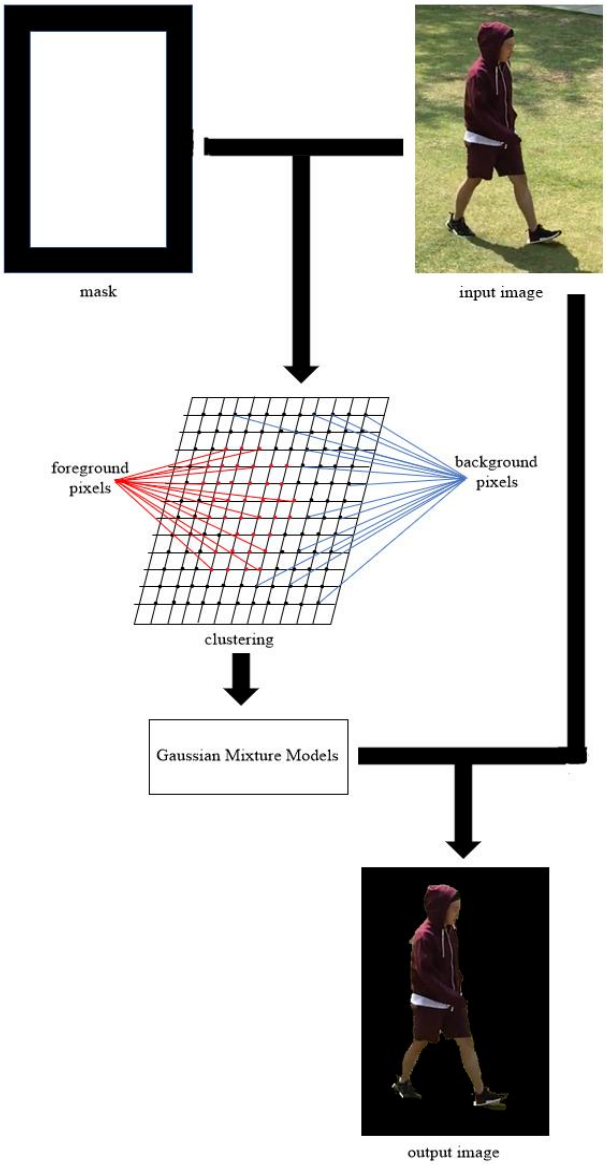


Figure 9 Workflow of GrabCut. The process and an example result of using GrabCut to get the foreground. The black and white picture on the top-left part is the mask which is set by the system automatically. The black area is the background area where all the pixels are considered as background, all the pixels in the white area are considered as possible foreground. By doing the clustering and modeling, a foreground Gaussian Mixture Model and a background Gaussian Mixture model are generated. Then apply the

models on the input image, and the pedestrian which is the foreground would be extracted.

The basic steps for GrabCut are as follows:

(1) Setting two areas: the background area where all the pixels are marked as background pixels, and the unknown area where contains pixels that can be either foreground pixels or background pixels. Those two areas are used to be selected manually, but they are automatically selected in this system by setting a rectangle area at the center of the input image that has 80% of the height and width of the input image.

(2) Putting the pixels which are in the unknown area into foreground class and all the background pixels into background class to create an initial image segmentation.

(3) The foreground and background classes are modeled as Gaussian Mixture Models using color clustering. Assume there are K clusters, z is the image data, and the first cluster $C_1 = \text{foreground} \cup \text{background}$, μ_1 is the mean value and $\sum_1$is the covariance of the cluster C_1.

For i = 2~K,:

(i) find the cluster C_n from cluster C_2 to cluster C_K which has the largest eigenvalue and its associated eigenvector e_n.

(ii) split C_n into two different sets along the mean values projection on the eigenvector, so that let $C_i = \{x \in C_n : e_n^T z_n \leq e_n^T \mu_n\}$ and update the original cluster with the other half $C_n^* = C_n - C_i$.

(iii) compute $\mu_n^*, \sum_n^* \mu_i$, and $\sum_i$.

(4) Every pixel in the foreground class and background class assigned most probable Gaussian component in the foreground Gaussian Mixture Models and background Gaussian Mixture Models. This creates a new Gaussian Mixture Models.

(5) Applying the new models on the image to find a new classification of foreground and background pixels.

(6) Repeat step 4 and 5 until the classification converges.

6.2 HSV Color Histogram

When dealing with colors, RGB and HSV are two commonly used color models. In RGB model, every color is the combination of different shades of red (R), green (G) and blue (B). Although this model is suitable for defining color on hardware, like television and multimedia devices, the way that RGB model simulates the color is different with how human sees the color. So, the distance in RGB color space does not represent differences in visual perception of human eyes [14]. Compared with RGB model, HSV model meets human's perception of color [15]. As shown in Figure 10, it has three properties which are hue (H), saturation (S) and value (V). The range of hue is 0°~360° and start from red which is 0°, green is 120° and blue is 240°. Saturation describes the closeness to the spectral color in a range of 0%~100%. The value represents the brightness of the color from 0% (black) to 100% (white).

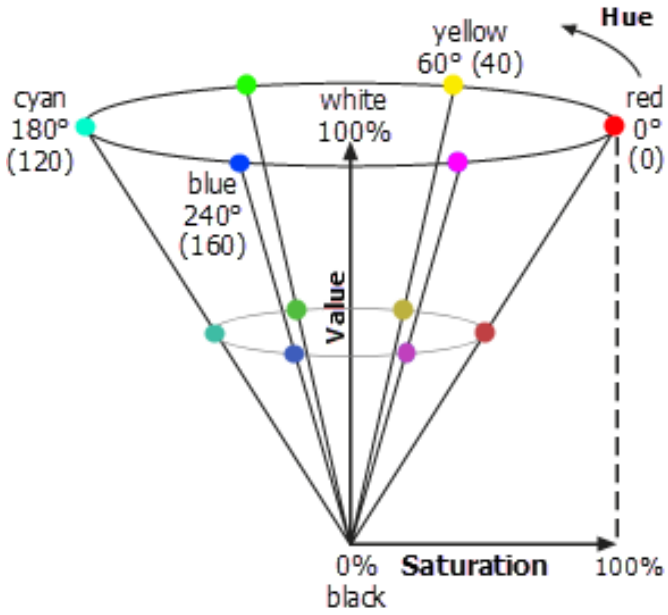


Figure 10 HSV color space. It has three properties, hue (H), saturation (S) and value (V). Hue decides the color, saturation presents the purity of the color and value indicates the brightness of the color.

Steps for creating HSV color histogram:

(1) Converting the image to HSV color space:

Assume R, G and B are the three channels of an RGB color space image, Max and Min are the maximum value and minimum value of R, G, and B. According to the definition of HSV color space, set $H \in [0, 360]$, $S \in [0, 1]$ and $V \in [0,1]$. Assume, $\text{Max} = \max(R, G, B)/255$, $\text{Min} = \min(R, G, B)/255$.

For each pixel, the conversion rules are as following:

$$H(i,j) = \begin{cases} 0°, & \text{if Max} = \text{Min} \\ 60° \times \dfrac{G-B}{Max-Min} + 0°, & \text{if Max} = \text{R and G} \geq \text{B} \\ 60° \times \dfrac{G-B}{Max-Min} + 360°, & \text{if Max} = \text{R and G} < \text{B} \\ 60° \times \dfrac{B-R}{Max-Min} + 120°, & \text{if Max} = \text{G} \\ 60° \times \dfrac{R-G}{Max-Min} + 240°, & \text{if Max} = \text{B} \end{cases} \quad (6-2)$$

$$S(i,j) = \begin{cases} 0, & if\ Max = 0 \\ \dfrac{Max-Min}{Max} = 1 - \dfrac{Min}{Max}, & otherwise \end{cases} \quad (6-3)$$

$$V(i,j) = \text{Max} \quad (6-4)$$

Now the image is converted from a RGB image to a HSV image.

(2) Doing the quantization on every pixel:

For Hue channel, H = { 0, if h∈[345,15], 1, if h∈[15,25], 2, if h∈[25,45], 3, if h∈[45,55], 4, if h∈[55,80], 5, if h∈[80,108], 6, if h∈[108,140], 7, if h∈[140,165], 8, if h∈[165,190], 9, if h∈[190,220], 10, if h∈[220,255], 11, if h∈[255,275], 12, if h∈[275,290], 13, if h∈[290,316], 14, if h∈[316,330], 15, if h∈[330,345] }.

For Saturation channel, S = { 0, if s∈[0,0.15], 1, if s∈[0.15,0.4], 2, if s∈[0.4,0.75], 3, if s∈[0.75,1] }.

For Value channel, V = { 0, if v∈[0,0.15], 1, if v∈[0.15,0.4], 2, if v∈[0.4,0.75], 3, if v∈[0.75,1] }.

Combining H, S, V channels of each pixel at location (i, j) into channel G. It is the color index for each pixel:

$$G(i,j) = 16 * H(i,j) + 4 * S(i,j) + V(i,j) \quad (6-1)$$

(3) Assigning the weight to each pixel:

The weight of every pixel is decided by the distance between itself and the center pixel of the image. Suppose H and W are the height and width of the image, $C_x \; and \; C_y$ are the location of the center pixel in the image, set a circle area at location (C_x, C_y) with radius:

$$\mathrm{R} = \sqrt{(H/2)^2 + (W/2)^2} \qquad (6-2)$$

Dist is the distance between the center pixel and pixels at location (i, j):

$$\mathrm{Dist} = \sqrt{(C_x - i)^2 + (C_y - j)^2} \qquad (6-3)$$

If Dist is in the range of circle area which means Dist is smaller than R, then set the weight of the pixel at location (i, j):

$$\mathrm{weight} = 1 - \left(\frac{Dist}{R}\right)^2 \qquad (6-4)$$

Otherwise, the weight would be 0.

(4) Calculating the histogram based on step 2 and step 3.

Since the highest value of $\mathrm{G}(\mathrm{i}, \mathrm{j})$ is 255 using equation 6-1, the horizontal axis is divided into 256 bins and setting U as the index of each bin. So, the height of the bin with index U in the histogram is the sum of the weight of pixels with $\mathrm{G}(\mathrm{i}, \mathrm{j}) = U$.

6.3 Histogram Similarity

Since the histogram can be seen as the vector of its original image, this paper uses Pearson Correlation Coefficient and Cosine Similarity which are two most commonly used methods to calculate the similarity of HSV histograms between the target image and pre-labeled images.

6.3.1 Pearson Correlation Coefficient

Pearson correlation coefficient measures the linear correlation between two vectors. The range of the correlation result is from -1 to 1, where -1 means totally negative linear correlation, 1 is totally positive linear correlation and 0 represents there is no linear correlation between the two vectors. When the value of correlation is approaching 1 that means the two vectors have high correlation and they are very similar. Otherwise, they are not that alike.

Assume A and B are the two vectors and they have the same size, m and n are the height and width of A and B, $\overline{A}$ and $\overline{B}$ are the mean value of A and B, the similarity between A and B is:

$$\text{similarity} = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \overline{A})^2)(\sum_m \sum_n (B_{mn} - \overline{B})^2)}} \qquad (6-5)$$

6.3.2 Cosine Similarity

Cosine similarity calculates the similarity between two vectors by measuring the cosine value of them. And the vector size is not in consideration, only the direction of the vector is included in the comparison. The cosine of 0° is 1, and no angle would have a cosine greater than 1 or smaller than -1. The cosine of the angle between the two vectors determines if they are pointing the same direction. If they have the same direction, the cosine would be 1; if the two vectors are vertical, the cosine value would be 0. When the two vectors have totally opposite directions, the cosine would be -1. Since the intersection degrees under most situations are less than 90°, the cosine value is in the range of 0 to 1. Also, if the closer the cosine is to 1, those two vectors are more similar to each other.

Suppose A and B are the two vectors, n is the number of elements in each vector, A_i and B_i are the elements in vector A and B at location i. The equation of calculating the similarity using Cosine Similarity:

$$\text{similarity} = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \qquad (6-6)$$

After getting many similarity values based on Pearson correlation coefficient or Cosine similarity, finding the histogram who has the highest similarity when compared with the target histogram among all the histograms which are based on pre-labeled images, then assign its label to the target image.

CHAPTER 7

PEDESTRIAN TRACKING

Kernelized Correlation Filters (KCF) is a high-speed tracking-by-detection method that trains a target detector during tracking. It uses the detector to detect the object at the predicted location is the target or not. Then KCF adds the result into its training set to update its target detector. While training the detector, it always sets target area as positive sample and area around target are set to negative. Moreover, to reduce the influence of having all negative samples with same return value without considering their distance from target center, KCF sets samples' return value in the range of [0,1]. The smaller the distance between sample and target center is, the closer the value is from 1, otherwise, the weight would get closer to 0.

All the samples in the training set are obtained by cyclic shifts on target samples. For example, with a 1-dimensional vector $x = [0, 0, 1]$. It shifts one element at a time and the results are $y = [1, 0, 0]$, $z = [0, 1, 0]$. If put x, y, and z into a matrix P that would be $P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ and it is called circulant matrix. Moreover, for a 2-dimensional image, cyclic shifts in both vertical and horizontal directions lead to a variety of different samples. Figure 11 shows results of applying cyclic shifts on target sample which is the image at the center to get negative samples.

Figure 11 Applying cyclic shifts on target sample. The image at the center is the target sample. And surrounding images are samples after cyclic shifts.

Steps of KCF are described in the following:

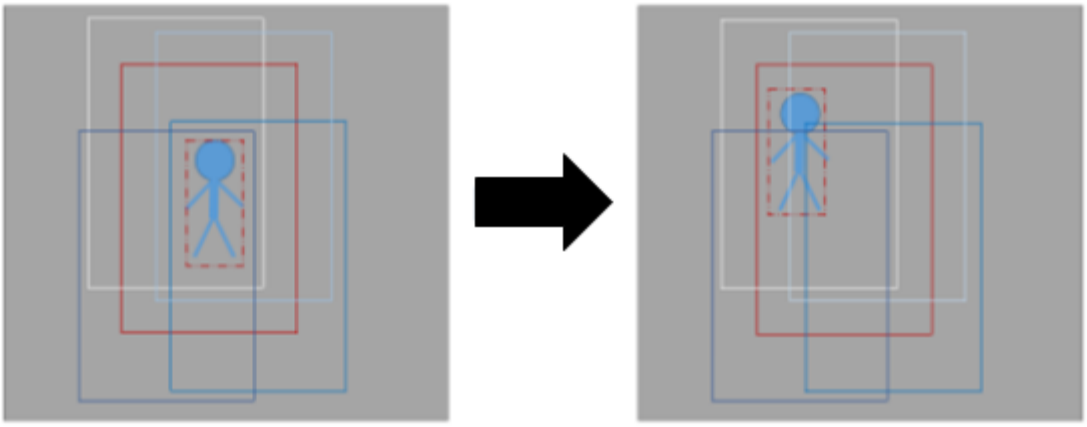
(1) Setting the target area and create negative samples based on cyclic shifts on the current video frame. As shown in Figure 12 (a), the red dotted box is the actual location of the target, the red solid box is padding, and rest of the boxes are samples after cyclic shifts on padding area.

(2) Using samples from step 1 to train a classifier and when the training is completed, enter next video frame.

(3) Doing sampling in the red solid box area and apply cyclic shifts again to get samples in the next video frame as shown in Figure 12 (b).

(4) Calculating the response in all the padding areas except the red solid box area using the pre-trained classifier. The area with the highest response contains the target. And in Figure 12 (b) that would be the white box on the top left of the frame who has the highest response.

(5) Repeat step 1 to 4 to keep training and detection.



(a) Current frame (b) Next frame

Figure 12 Target tracking. The pedestrian in current frame moves into the white box so that the white box would have the highest response.

CHAPTER 8

RESULTS

The pedestrian tracking system has many different parts, each part involves different technologies. Except for the final performance of the system, the test results of each part are also necessary. To achieve the best results in every part of the process, I tested the performance of each part and compared the results between the technologies used in the same part.

8.1 Results of Movement Detection

Gaussian Mixture Models was found to has several disadvantages during movement detection. Although it detects objects that move fast and with a small shape, when an object moves very slow or suddenly stops for a few seconds, the system would lose that object. That is because Gaussian Mixture Models is very sensitive to illumination changing in the whole image. As shown in Figure12, (a) to (f) show the differences in varying brightness. In Figure 13 (a), the light stays still for a long period which leads to minor changing as shown in Figure 13 (b). However, if the brightness of the whole image increases or decreases in a sudden, it would let the system recognize the whole frame or part of the frame as foreground. In Figure 13 (d) which is the mask of Figure 13 (c), a target has been detected and a yellow box is drawn around it, but this object is absolutely part of the background. In Figure 13 (d) and Figure 13 (f), they have a huge difference when compared with Figure 13, although there is no obvious changing in the actual background when comparing Figure 13 (a) with Figure 13 (c) and Figure 13

(f). This comparison shows that Gaussian Mixture Model is very sensitive to the brightness change of large areas.

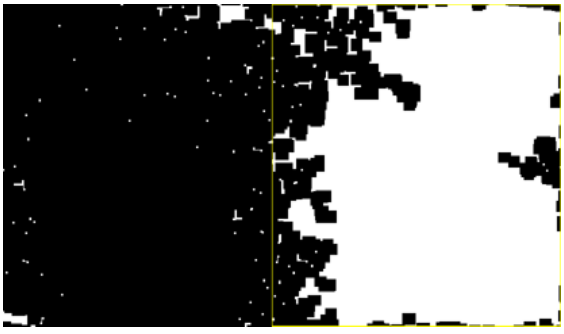
(a) Background under normal condition

(b) mask of (a)

(c) Background with brightness change

(d) mask of (c)

(e) Background with huge brightness change



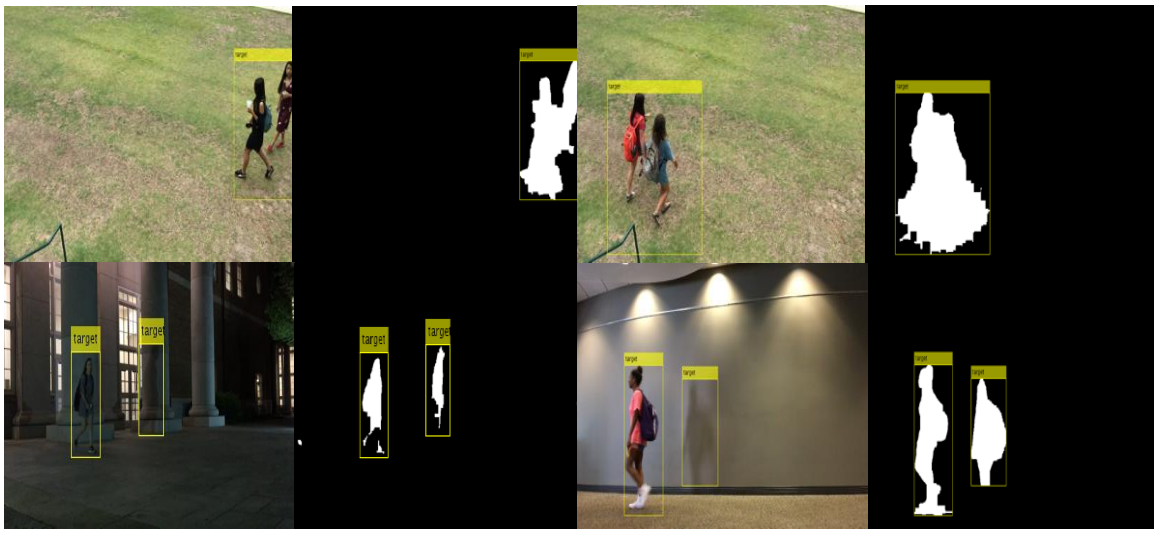
(f) mask of (e)

Figure 13 Backgrounds and binary masks under different brightness. Although the backgrounds are the same, their masks show a huge difference between each other.

8.2 Results of HOG and SVM

In this part, I trained a classifier using Support Vector Machine based on Histogram of Oriented Gradient feature. This classifier is tested on a test set that contains 300 negative samples and 300 positive samples, and the classification result for assigning each image into the correct category is 91%. Although this result is good, the performance decreases when combining the classifier with motion-based object detection. Figure 14 shows the detection results with their masks. In Figure 14 (a) are successful detection results. Foreground pixels are set to 1 in the mask of each image and background pixels are set to 0. As you can see, there is a white shadow with car shape in one of the masks, the system ignores this object since it is not a pedestrian and only marks the real pedestrian. Although motion-based pedestrian detection has a good performance when the object is moving, there are still some situations it cannot deal with. As shown in Figure 14 (b), when multiple pedestrians are walking side by side at very close intervals, the system can detect those pedestrians, but it is very hard to detect each one of them separately. Also, when the light is on the side of the pedestrian which creates a shadow with the shape of a human being, the system detects the shadow as a pedestrian.

(a)



(b)

Figure 14 Detection results. (a) are the successful results and (b) are results of the failure detection. Black and white images are the mask of the original images. In the successful results, the system did not recognize the mask of a car as the pedestrian. However, in the failure results, the system accidentally detected the shadow as a pedestrian.

8.3 Results of Neural Network

Compared with detection using Histogram of Oriented Gradient features with Support Vector Machine, neural network based on Faster R-CNN has a better performance on pedestrian detection. To evaluate the performance of the detector, the test images are set to include indoor and outdoor, day and night, sunny time and cloudy, from one person up to five people in the same image at a time, different camera angles, and different postures like running, walking and cycling. Figure 15 and Figure 16 are test results that tested on a detector trained by a neural network that has 2 conv layers based on a dataset with 240×135 as the maximum image size. Figure 15 shows some successful results of using that detector. The objects that are detected as a pedestrian are surrounded by a yellow rectangle and labeled with the tag which is 'Pedestrian'. And all the pedestrians appear in each image in Figure 15 are detected correctly.

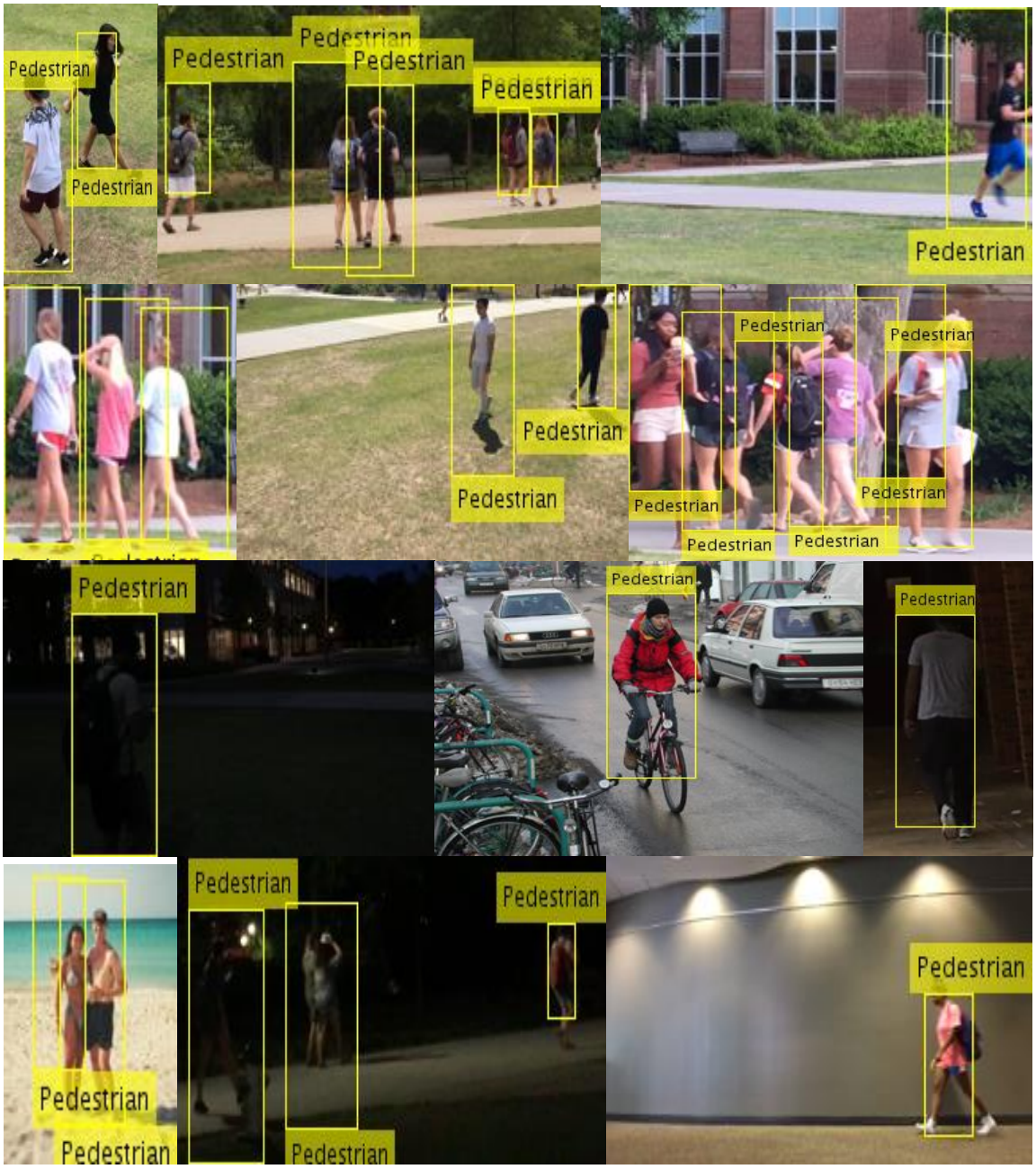


Figure 15 Successful results for pedestrian detection using the neural network. All the people in those images are marked by a yellow rectangle and labeled as a pedestrian.

However, the detector still misses target which detects less pedestrian than the total number of pedestrian and detection failure that using multiple rectangles to mark one person or detecting part of the background as a pedestrian under some conditions. Figure 16 shows some unsuccessful results of the detector. There are several possibilities for the cause of miss detection, one is the color of the target is very similar to the background that makes the system unable to distinguish the target from the background, another reason is the target size is too small to be detected. As for detection failure, the possible reasons are the color feature and texture feature create an area that is very similar to a human, like a shadow.

Figure 16 Unsuccessful results of the detector trained by the neural network. The system may use multiple boundary boxes to describe one pedestrian, detect part of the background as the pedestrian and cannot detect the pedestrian.

To enhance the performance of the detector, I trained several detectors based on a different number of conv layers. Table 1 presents the detection results. The first 4 rows in the table are detectors trained and tested on datasets which all images are downsampling to a maximum size 240×135, D5 is trained on the same dataset size but the test images are 480×270 and D6 is trained and tested on datasets that the size is 480×270. Failure rate means the percentage of the area which does not contain pedestrian is detected as a pedestrian or a pedestrian is detected but marked with multiple detection boxes. The missing rate is defined as when the number of the detected pedestrian is less than the actual number of the pedestrian in the video without failure detection.

When compared the first four rows, the detector D2 which is trained with 2 convolutional layers has the highest correct rate that is 66.17% and no matter reducing or increasing the number of convolutional layers, the correct rate decreases. When comparing D2 with D5 and D6, we can see that detectors trained and tested on the same size images have better detection results than the detector trained and tested on a different size.

The best correct rate in table 1 is 66.17%, I thought the dataset is a possible reason that led to this result. Since all images from the INRIA dataset have very complex content, it might have added a lot of noise to the training and affected the training process. Therefore, I had taken another two videos and extracted 584 images to form a dataset from the first 25 seconds of those two videos. 194 images from rest of the videos are set to be the test images. The detectors are also trained on images with size 240×135 and a different number of convolutional layers. The results are shown in table 2. As you can see, the highest performance has reached 91.44% when using 3 convolutional layers to train

the detector. After removing images from the INRIA dataset and using only images that have simple content as training samples, the training results have been significantly improved.

Table 1 Results of the detectors trained under different settings

Detector	Correct Rate	Missing Rate	Failure Rate
1 conv layer (D1)	22.93%	6.51%	70.56%
2 conv layers (D2)	66.17%	13.23%	20.60%
3 conv layers (D3)	39.50%	17.28%	43.22%
4 conv layers (D4)	51.83%	33.37%	14.80%
2 conv layers (D5)	49.57%	16.74%	33.69
2 conv layers (D6)	54.25%	33.91%	11.84%

Table 2 Results of the new detectors.

Number of Layers	Correct Rate	Missing Rate	Failure Rate
1 Layer	63.18%	2.57%	34.25%
2 Layers	90.41%	8.02%	1.57%
3 Layers	91.44%	7.36%	1.2%
4 Layers	67.98%	3.08%	28.94%

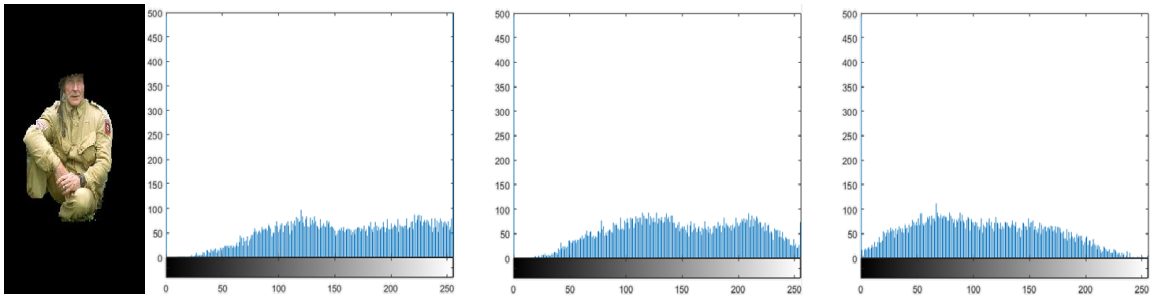
8.4 Results of Pedestrian Re-identification

I measure the performance of re-identification by comparing the histograms of the input images under different conditions. In Figure 17, it shows the results of GrabCut on pedestrian images. As you can see, automatic GrabCut has very good performance on segmenting the main body of the pedestrian from the background. But when a small part of the background is surrounded by the foreground, this area is marked as foreground and would not be eliminated. Furthermore, some accessories like hat and shoes are eliminated is because the difference between them and the background is not that great. Since the system only considers the color of the main body, small background areas that left behind, and the loss of some accessories on the pedestrian would not be able to affect the re-identification result.

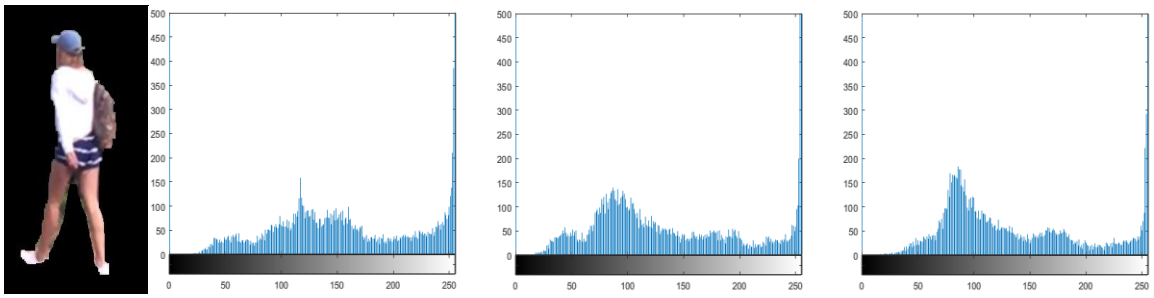
Figure 17 Results of automatic GrabCut. The pedestrians are extracted from the background and only a very small part of the background left.

After extracting the pedestrian from the background, the system calculates the color histograms of each pedestrian. I compared the histograms of each pedestrian and the similarity between every two images based on RGB color space and HSV color space. Figure 18 shows the RGB histograms separately from left to right in the order of red, green and blue channels. As you can see, although the general color of each pedestrian is very simple which are yellow, white, black, red and grey, their histograms for RGB channels have different shapes and it makes color comparison very difficult. Because the two images must have a very small difference in each channel, otherwise, their synthetic similarity would be much lower than the similarity in any channel. For example, when using Pearson Correlation Coefficient to calculate the similarity of every channel, the similarity between Figure 18 (a) and (b) in each channel are 88.46%, 13.49% and 0. Only channel R has a very high similarity and rest of the channels have a very low similarity that indicates the two pedestrians are totally different. However, Figure 18 (e) and (f) are the same person, the only difference is the position of this pedestrian. However, there are two peaks that are obvious in Figure 18 (f) and only one obvious peak in Figure 18 (e). The similarity of Figure 18 (e) and (f) in RGB channels are 51.55%, 76.12%, and 43.87%. 51.55% and 43.87% are much lower than the threshold value (75%) of identifying two people as the same person. Furthermore, the similarity of Figure 18 (c) and (d) in RGB channels are 44.17%, 68.75 %, and 77.83%. There is one channel meet the requirement (75%) of being considered as same person and another channel is very close to that requirement. Although pedestrians in Figure 18 (c) and (d) are two completely different people that one of them wears black color clothes and another one wears purple clothes, the similarity between them is much higher than the similarity between Figure 18 (e) and

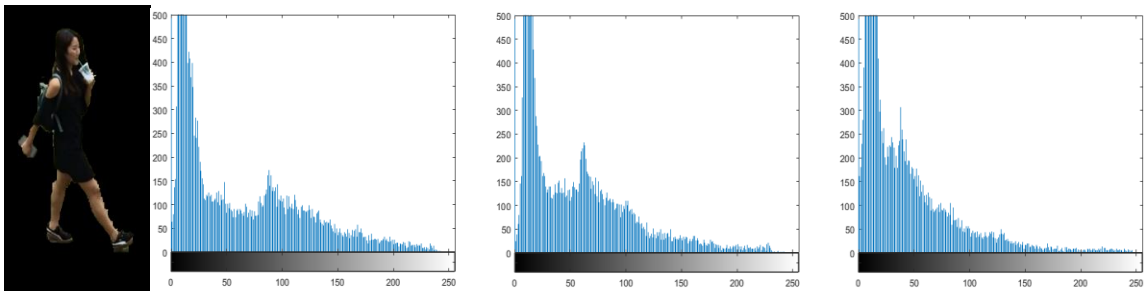
(f) which contains the same person wearing same color clothes. From those two comparisons, pedestrian re-identification based on RGB color space is unreliable and not stable. Due to different postures and brightness, a person would be identified as two different people and two strangers would be considered as the same person.



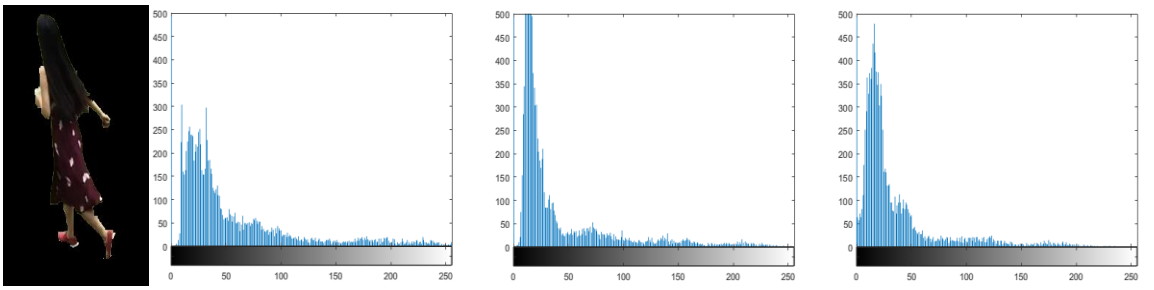
(a)



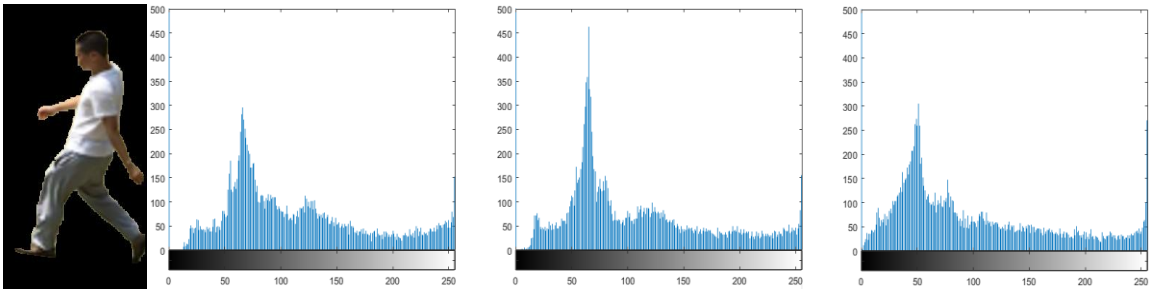
(b)



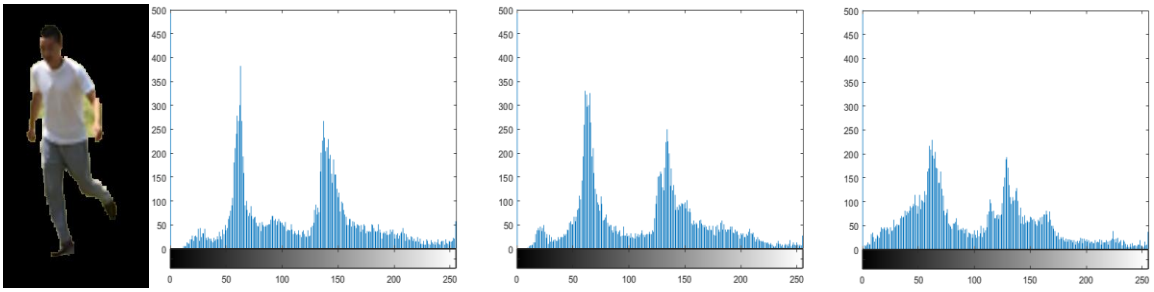
(c)



(d)



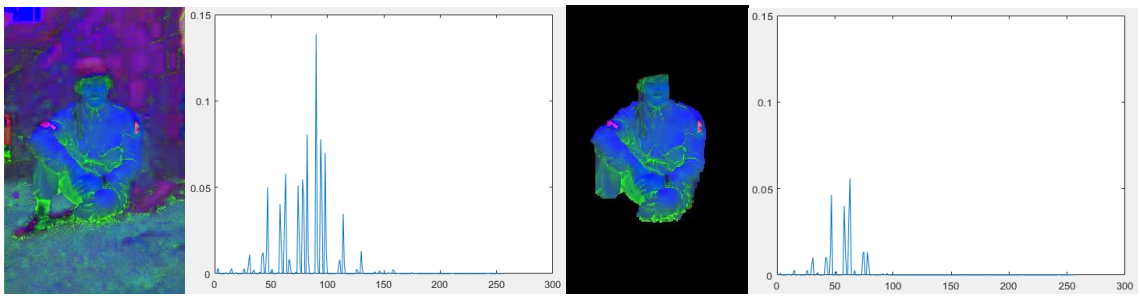
(e)



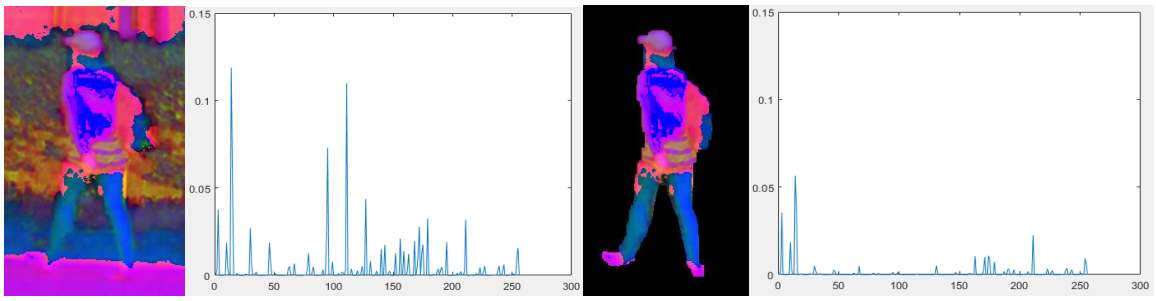
(f)

Figure 18 Histograms based on RGB channels. In the first column is the images only contain the foreground. And rest of the histograms from left to right are the red channel, the green channel and the blue channel.

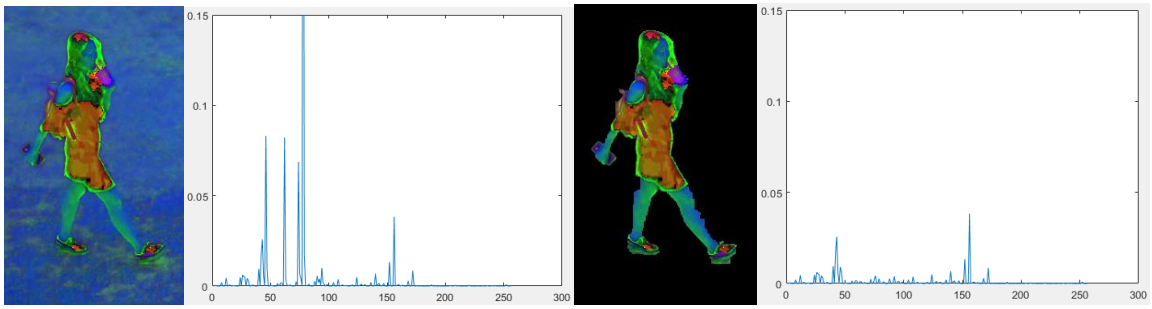
Pedestrian re-identification based on HSV color space is a more reliable method when compared with using RGB color space, due to the same principle of color recognition by human eyes and containing more information then RGB color space. Figure 19 compares the histograms of the pedestrian in HSV color space. Histograms on the left side are based on original HSV images that have the background and on the right side are histograms based on pedestrian without the background. And there is a big difference when comparing the histograms on the left side with the right side due to the existence of the background. As you can see, the shape of the histogram in the fourth column of Figure 19 (a) has shrunk a lot compared with the histogram in the second column and almost all the peaks in the range of 20~170 in Figure 19 (b) disappear. Figure 19 (c), (d), (e) and (f) share the same background and all of them have several peaks located in the range of 50~100 on the horizontal axis that is much higher than any other peaks. Moreover, those peaks are all gone after eliminating the background, but rest of the peaks are retained that proves the assumption of considering those high peaks indicate the background.



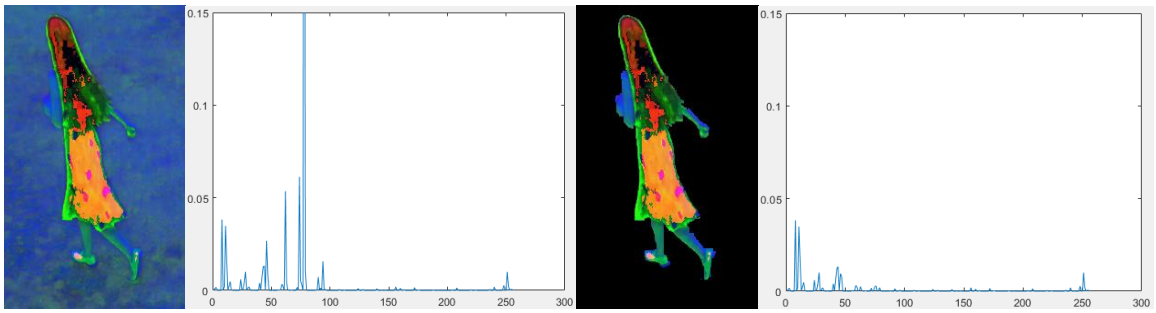
(a)


0.15
0.1
0.05
0
0 50 100 150 200 250 300

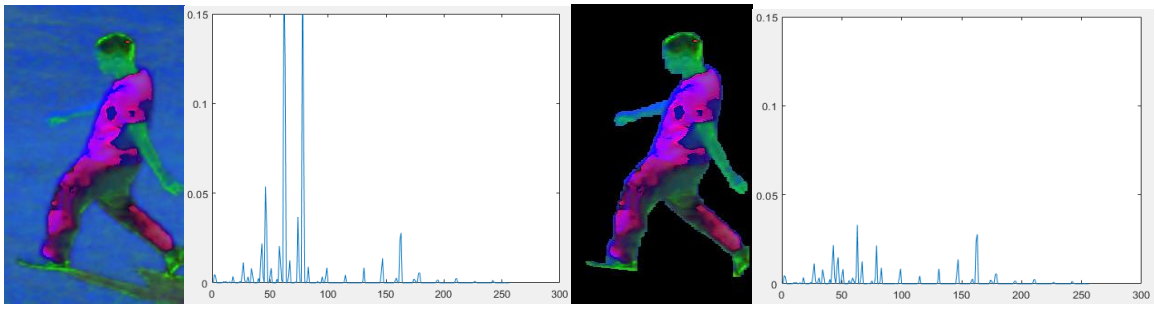
(b)


0.15
0.1
0.05
0
0 50 100 150 200 250 300

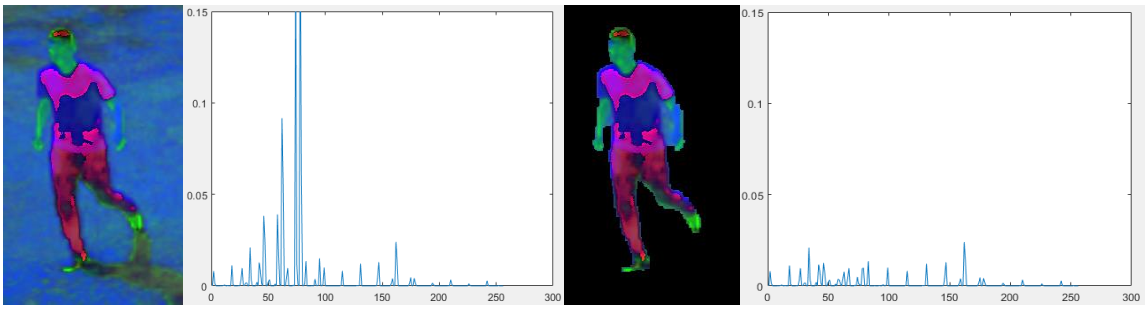
(c)


0.15
0.1
0.05
0
0 50 100 150 200 250 300

(d)


0.15
0.1
0.05
0
0 50 100 150 200 250 300

(e)



(f)

Figure 19 Histograms based on HSV channels. The first column is the original images in HSV color space and the third column contains the images in HSV color space after applying the GrabCut algorithm. The second and fourth columns are their histograms.

The similarity between two pedestrians is calculated using Pearson Correlation Coefficient (PCC) and Cosine Similarity (CS) based on images from Figure 19 and results are shown in table 3 and table 4. Table 3 shows the similarity between images with background while Table 4 shows the similarity based images without background. In both Table 3 and Table 4, although Pearson Correlation Coefficient and Cosine Similarity are different methods, the similarities calculated by PCC and CS between two test images are almost the same. There is no better between these two methods.

As shown in Table 3, it still has some false re-identifications. Meanwhile, there is no false re-identification in Table 4. Figure 19 (e) and (f) are successfully identified as the same person and Figure 19 (c) and (d) are considered as different people. It draws a conclusion that background has a negative effect on pedestrian re-identification. When comparing Table 4 with Table 3, we can see that most of the similarities are decreasing

after the background is removed. Especially, the images that have the same background would have higher similarity rate when comparing to each other even if the pedestrian contained in the images are not the same person. But there is an exception that only the similarity between Figure 19 (a) and (e) increases. After careful observation, the reason is that the color of the military uniform in Figure 19 (a) and the skin color in Figure 19 (e) are very similar. Therefore, even though the background is removed as a disturbing factor, the proportion of similar colors of the two people is increased, leading to an increase in similarity without any decrease. In general, comparing the similarity between each pedestrian without the background in HSV color space has better performance than RGB color space.

Table 3 Similarity of images with background

similarity	a	b	c	d	e	f
a(PCC)	100%	11.04%	28.86%	26.84%	41.95%	43.54%
a(CS)	0°	79.82°	72.04°	73.47°	63.32°	62.42°
b(PCC)	11.04%	100%	2.61%	0.97%	3.71%	2.61%
b(CS)	79.82°	0°	86.54°	87.76°	84.65°	83.54°
c(PCC)	28.86%	2.61%	100%	98.10%	71.6%	80.37%
c(CS)	72.04°	86.54°	0°	11.16°	43.88°	36.27°
d(PCC)	26.84%	0.97%	98.10%	100%	65.85%	76.36%
d(CS)	73.47°	87.76°	11.16°	0°	48.52°	40.09°
e(PCC)	41.95%	3.71%	71.6%	65.85%	100%	81.36%
e(CS)	63.32°	84.65°	43.88°	48.52°	0°	34.82°
f(PCC)	43.54%	2.61%	80.37%	76.36%	81.36%	100%
f(CS)	62.42°	83.54°	36.27°	40.09°	34.82°	0°

Table 4 Similarity of images without background

similarity	a	b	c	d	e	f
a(PCC)	100%	1.25%	15.13%	12.76%	49.16%	28.97%
a(CS)	0°	86.76°	78.71°	80.34°	58.69°	70.46°
b(PCC)	1.25%	100%	0	6.25%	8.39%	4.86%
b(CS)	86.76°	0°	90°	83.62°	81.52°	83.08°
c(PCC)	15.13%	0	100%	26.48%	26.05%	18.55%
c(CS)	78.71°	90°	0°	71.95°	71.25°	75.03°
d(PCC)	12.76%	6.25%	26.48%	100%	14.20%	8.28%
d(CS)	80.34°	83.62°	71.95°	0°	78.27°	81.13°
e(PCC)	49.16%	8.39%	26.05%	14.20%	100%	79.24%
e(CS)	58.69°	81.52°	71.25°	78.27°	0°	35.52°
f(PCC)	28.97%	4.86%	18.55%	8.28%	79.24%	100%
f(CS)	70.46°	83.08°	75.03°	81.13°	35.52°	0°

When using this method in a video, it shows results of re-identify the same person. Figure 20 shows some screenshots of marking the same person who has left the video for a while then comes back with the same label. As you can see, people who appear in both images from the first column and the second column or third column and fourth column in the same row are assigned with the same label. Pedestrian re-identification performs the best when the background is very simple, if the background is very complex like images in Figure 21, it may causes miss re-identify. In the fourth row of Figure 20, three pedestrians in red, black and blue color clothes are labeled with ‘A’, ‘B’ and ‘C’. And in the last row, they are labeled with same labels as they first came into the video when each two of these three pedestrians appear in the same image again.

Figure 20 Results of re-identification. Same pedestrians are labeled with their original label when they come back into the video again.

8.5 Results of Pedestrian Tracking

The tracking system tracks the target once the target is set manually. In Figure 21, a series of images are presented to describe the process of tracking. The blue rectangle with a ‘Target’ tag represents the tracking box. The box keeps following the targeted pedestrian as it moves. And when the pedestrian leaves the video, the blue box would also disappear. The fourth row in Figure 21 shows that the system keeps tracking the pedestrian even if the pedestrian changes his walking direction into the opposite.

Figure 21 Results of tracking. The tracking box tracks the pedestrian while he or she is moving in the video. Every three images in the same row show a process of how the tracking system tracks a specific pedestrian.

CHAPTER 9

CONCLUSIONS

In this paper, I describe the framework of the pedestrian tracking system and provide the approaches and implementation results of my current work. The results of using HOG with SVM and Faster R-CNN are compared under different conditions. Furthermore, this system can detect a pedestrian and assign the same label to the people who re-enter the video after a short period. And when a specific pedestrian is set to be the target, the system tracks this target until it leaves the video.

My future work will focus on increasing the accuracy of pedestrian re-identification by combining more features together. And in the future, the system would be able to get target information, such as moving speed, how many time does this target enter the video and how much calories a specific target consumes.

REFERENCES

[1]Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, *"Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"*, Computer Vision and Pattern Recognition, 2015.

[2]João F. Henriques, Rui Caseiro, Pedro Martins, Jorge Batista, *"High-Speed Tracking with Kernelized Correlation Filters"*, IEEE, 2014.

[3] Z. Kalal, K. Mikolajczyk, and J. Matas, *"Tracking-learning-detection"*, TPAMI, 2012.

[4] S. Hare, A. Saffari, and P. Torr, *"Struck: Structured Output Tracking with Kernels"*, ICCV, 2011.

[5] Zhentang Jia, Guiming He. *"A Fast Algorithm for Moving Video Object Segmentation[J]",* Journal of Image and Graphics, 2002, 7(11): 1123-1127.

[6] D. Khattab, H. M. Ebied, A. S. Hussien, andM. F. Tolba, *"Automatic GrabCut based on unsupervised clustering for image segmentation",* Intelligent Systems' 2014 Advances in Intelligent Systems and Computing Volume 323, 2015, pp 579-592.

[7] C.BenAbdelkader and L.Davis, *" Detection of People Carrying Objects : a Motion-based Recognition Approach",* 5th IEEE International Conference on Automatic Face and Gesture Recognition ,May, 2002.

[8] A. F. Bobick, and J. W. Davis, *"The Recognition of Human Movement Using Temporal Templates",* PAMI, Vol. 23, No. 3, 2001.

[9] S.S. Beauchemin and J.L.Barron, *"The Computation of Optical flow",* ACM

Computing Surveys, Vol.27, 1995, pp 433 – 466.

[10] Rother C, V. Kolmogorov, and A. Blake, *""GrabCut": interactive foreground extraction using iterated graph cuts",* ACM Transactions on Graphics, vol. 23, no. 3, pp. 309–314, 2004.

[11] Hongwen Lin, Dan Tu, Guohui Li. *"Moving Objects Detection Method Based on Statistical Background Model[J]",* Computer Engineering. 2003, 29(16): 97-99.

[12] Haritaoglu I, Harwood D, Davis L S. *"W4: Real-Time Surveillance of People and Their Activities[J]",* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 8209-830.

[13] Girshick, Ross. *"Fast R-CNN",* Proceedings of the IEEE International Conference on Computer Vision. 2015.

[14] Kilian Q. Weinberger, Lawrence K. Saul, *"Distance metric learning for large margin nearest neighbor classification",* Journal of Machine Learning Research, vol. 10, pp. 207–244, 2009.

[15] Loris Bazzani, Marco Cristani, Alessandro Perina, Vittorio Murino. *"Multiple-shot person re-identification by chromatic and epitomic analyses",* Pattern Recognition Letters, Vol.33, PP. 898–903, 2012.