國立臺灣大學電機資訊學院生醫電子與資訊學研究所

碩士論文

Graduate Institute of Biomedical Electronics and Bioinformatics
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

基於粒子濾波器方法搭配固定相機之 PTZ 攝影機系統進行人像追蹤：以生醫教師追蹤與視訊錄影為案例分析

A Pan-Tilt-Zoom Camera System for Human Tracking based on the Particle Filtering Method with a Static Camera: A case study of Biomedical Teacher Tracking and Video Recording

翁丞世

Cheng-Shih Wong

指導教授：傅楸善 博士

Advisor: Chiou-Shann Fuh, Ph.D.

中華民國 106 年 6 月

June 2017

誌謝

首先，能順利完成本篇論文首先要非常感謝傅楸善教授的指導，他總是不遺餘力地在研究過程中支持我並且給予幫助，同時也非常感謝捷揚光電提供合作的機會，支持我研究這個主題。而公司同仁們專業的意見也讓我受益良多，尤其要感謝鄭文欽先生、吳中皓先生、陳柏宏先生、鄭又涵小姐、邱峻堯先生的指導與幫助。

接著也感謝數位相機與電腦視覺實驗室的學長姊、同學和學弟妹們在研究期間的照顧。呂建明學長、林德垣學長無論在任何問題總會給予我適當的幫助，讓我在研究過程中順利度過難關。學長林奇賦；學姊熊育萱不吝分享各種學術研究以及人生經驗，讓我獲益匪淺。同學陳弘毅，在研究過程中互相扶持照顧，一同分享學業上的甘苦。學弟謝尊安、楊志柔、高咸培、曾子家、李宇倫，還有實驗室小弟李昺圜，實驗室因為你們的加入帶來不少歡樂，讓我忘卻課業上的煩悶。數位相機與電腦視覺實驗室的所有成員們對於幫助我完成這篇論文實在功不可沒。

最後感謝我親愛的女友黃榆涵、母親陳素蘭、父親翁啟明，還有許多陪在我身邊的人，總是毫無保留地支持我在學業上的任何決定，使我能專心投入課業，無後顧之憂，當我永遠的避風港。

在此僅將這段時間的研究成果匯集成本論文，獻給所有曾經關心、照顧與幫助我的所有人。

中文摘要

本論文開發一套即時的生醫教師追蹤與視訊錄影系統，能夠在室內環境中對特定目標教師以一廣角相機與一左右轉動上下傾斜與放大縮小 (PTZ, Pan-Tilt-Zoom) 相機來進行追蹤。

行人追蹤已經有許多的生活應用，例如:老年人看顧、線上即時會議、住宅安全監控。在此追蹤系統中，我們採用由偵測來追蹤的方法以粒子濾波的框架來實現。

首先我們必須指定欲追蹤目標，接著以粒子濾波來模擬目標位置的分布。我們訓練一卷積類神經網路來估計目標影像是否為人頭之機率，還有以背景相減法來做前景偵測，且以色彩直方統計計算候選與目標之相似度。最終以估計得最高機率位置為當前追蹤之目標，再迭代估計下一幀之位置。

關鍵字:視覺追蹤、教師追蹤、卷積類神經網路、色彩直方統計、偵測追蹤法、粒子濾波、背景相減法

Abstract

In this thesis, we develop a nearly real-time biomedical teacher tracking and video recording system to track a biomedical teacher in the indoor scene by one wide-angle camera and one PTZ (Pan-Tilt-Zoom) camera.

Human tracking has many applications such as eldercare, security surveillance, and online meeting. In this human tracking system, we employ tracking-by-detection in particle filter framework to track the target.

We have to specify which person to track first, and model the location of the target as a state distribution by particle filter. Moreover, we train a convolutional neural network as a head classifier to estimate the probability of human head, the motion detector with background subtraction, and color histogram is used to obtain the similarity between candidate and the target.

Keywords: visual tracking, human tracking, convolutional neural network, color histogram, tracking-by-detection, particle filter, background subtraction

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Overview

Human tracking is always an important application in computer vision. Human tracking can be applied to many conditions such as eldercare, security surveillance, remote online meeting, and teacher tracking and video recording during course. Biomedical teacher tracking and video recording are important because more and more classes are offered online such as Coursera and MOOC (Massive Open Online Course) [23]. Therefore, we develop this automatic biomedical teacher tracking and video recording system to save laborious and tedious manual camera pointing and recording.

However, human tracking is still a challenging problem because we only have initial target information to track the target, and background clutter, changing illumination, object occlusion, and non-rigid target will increase the difficulty of tracking.

In this thesis, we employ two cameras shown in Figure 1-1, elevated 2 meters high from the ground in Figure 1-2, to be our system sensors. The wide-angle camera, the lower part of the equipment, is used to capture the whole scene in the room and detect

and locate moving teacher, and the captured video can be simultaneously analyzed by the computer to locate the target teacher. The Pan-Tilt-Zoom (PTZ) camera, the upper part of the equipment, can receive the location information from the computer, then move the camera to the target position to obtain more precise information and closer view.

Our method is based on Breitenstein's work [4]. We apply tracking-by-detection in particle filter framework for this system, instead of using HOG (Histogram of Oriented Gradient) detector [7] for the detection, we train a convolutional neural network to measure the possibility of human as a domain-specific classifier, a moving detector, and use color histogram similarity as an instance specific discriminator.

Figure 1-1 (left) The frontal view of our equipment. (right) The 30-degree side view of our equipment. The green box locates the wide-angle camera, and the red box

locates the PTZ camera.

Figure 1-2 We elevate the cameras to around 2 meters high from the ground.

1.2 Tracking-by-Detection

Tracking-by-detection has been applied to tackle the difficulties mentioned above in recent researches, driven by the progress of object detection. To reduce the search space of the target, particle filter [10] is a popular framework to be used in tracking-by-detection. It can well deal with the misdetection (there is a person, but classifier does not recognize it) and false alarm (classifier recognize a non-human object as a person) of the classifier used in detection. Particle filter can be divided into three parts: sampling, motion estimation, and measurement.

In the particle filter framework, we have to select the target teacher to initialize the locations of particles at the center of selected bounding box at the first frame. For the subsequent frames, it samples the particles from the weights of particles in previous frame.

In motion estimation, we estimate the motion direction and velocity of each particle. We use first order linear difference equation to efficiently approximate the locations of each particle in current frame by the locations in last frame.

Finally, we have to measure the probability of each particle as the new weights in the current frame. We model the weight as proportional to the candidate bounding box as a human and similar to our target. Therefore, we train a convolutional neural network as a classifier to predict the probability of head in an image, and a motion detector with background subtraction to detect the foreground. Then, we use color histogram to represent the target, which we can use to calculate the similarity.

1.3 Convolutional Neural Network

Convolutional neural network has become a popular research field in computer vision since the AlexNet's [13] astonishing performance on ImageNet LSVRC (Large

Scale Visual Recognition Challenge) [16].

Because of the power of feature extraction and acceleration by GPU (Graphic Processing Unit) of convolutional neural network, we train a VGG (Visual Geometric Group) Net-like [15] network as a human head classifier to obtain the possibility of candidate bounding box.

1.4 Background Subtraction

Background subtraction is a fully-developed technique in computer vision, and is used widely in many applications. It can extract the moving objects by modeling the background image, then subtract the current image by the background image to obtain the foreground objects. To model the background image, there is a simple method which takes the average image of previous frames to be the background image. The main drawback of this method is that the camera should be fixed.

In our system, the camera is mounted on a specific location that meets the limitation of background subtraction. Therefore, we can use background subtraction to obtain the moving objects as the target candidates.

1.5 Thesis Organization

In the rest of this thesis, we will discuss the recent techniques for visual tracking in Chapter 2. We also give some background knowledge about our thesis in Chapter 3. In Chapter 4, the foundational methodology and techniques used in this thesis will be described. The experimental results and performance evaluation for tracking process are described in Chapter 5. We eventually give the conclusion of this thesis in Chapter 6.

Chapter 2 Related Works

2.1 Overview

In recent years, visual tracking is one of the popular problems in computer vision, and also has been applied to a wide range of applications in computer vision, such as human-computer interaction, security surveillance, and medical imaging. Several algorithms are developed for human tracking.

The general problem of visual tracking is to estimate the position and size of the target in the rest of frames by giving the initialized bounding box of a target object in the first frame of a video. Although object tracking has been studied for several decades, and much progress has been made in recent years, it remains a very challenging problem.

Numerous factors affect the performance of a tracking algorithm, such as illumination variation, occlusion, as well as background clutters, and there exists no single tracking approach that can successfully handle all scenarios. We will describe the popular methods for visual tracking in this section.

2.2 Kernel-Based Tracking

D. Comaniciu et al. proposed a method for real-time tracking of non-rigid objects by the mean shift iterations and finding the most probable target position in the frame [5], known as kernel-based object tracking [6].

Another improved kernel method proposed by N. S. Peng et al. [14] used an adaptive Kalman filters for filtering object kernel histogram to obtain the optimal estimate of object model.

2.3 Tracking-by-Detection

Some tracking algorithms use discriminative models built on binary classifier to detect the target object, also known as tracking-by-detection. H. Grabner et al. proposed an on-line AdaBoost feature selection algorithm [9] which can adapt the classifier while tracking the object.

Another literature based-on on-line Adaboost learning, B. Babenko et al. [2] use multiple instance learning instead of traditional supervised learning to avoid incorrectly

labeled training examples caused by self-bootstrap, which degrades the classifier.

Z. Kalal et al. proposed a novel tracking framework decomposing the long-term tracking process into tracking, learning, and detection [12].

2.4 Domain-Specific Tracking

Some researches track multiple persons in crowded scenes. M. D. Breitenstein et al. proposed an algorithm using the continuous confidence of pedestrian detectors and online trained, instance-specific classifiers as a graded observation model [4]. An approach, which guaranteed real-time tracking in high definition video is built on asynchronous HOG (Histogram of Oriented Gradients) with KLT (Kanade-Lucas-Tomasi feature tracker) tracking and Markov-Chain Monte-Carlo Data Association [3].

B. Wu and R. Nevatia [26] represent the human body as an assembly of body parts, then part detectors based on edgelet features can be learned by Adaboost classifier. By forming a joint likelihood model and combining an analysis of possible occlusions, it can provide the person observations used for tracking. S. Tang et al. proposed a joint people detector which explicitly exploits common patterns of person-person occlusions [19].

Chapter 3 Background

3.1 Overview

We will briefly introduce the fundamental knowledge of particle filter, artificial neural network, and convolutional neural network in this chapter. A special kind of particle filter, condensation [10], for visual tracking is described in Section 3.2 . In Section 3.3 , the artificial neural network as a computational model will be depicted. Moreover, the convolutional neural network which has been studied intensively recently, a special type of ordinary neural network, is mentioned in Section 3.4 . Last, a classic background subtraction method is introduced in Section 3.5 .

3.2 Particle Filter

Particle filter [24], [1] or Sequential Monte Carlo (SMC) methods are a collection of genetic-type particle Monte Carlo approaches to deal with filtering problems emerging in signal processing and Bayesian statistical inference. In computer vision, Condensation (Conditional Density Propagation) [10] is the utilization of Sampling Importance Resampling (SIR), one of particle filter methods, estimation to contour tracking.

The filtering problem consists of estimating the internal states in dynamical systems when partial observations are made, and random perturbations are present in the sensors as well as in the dynamical system. The objective is to compute the conditional probability (as known as posterior distributions) of the states of some Markov process, given some noisy and partial observations.

A generic particle filter estimates the posterior distribution of the hidden states using the observation measurement process. For computational purposes, the propagation process must be set out in terms of discrete time t. Consider a state-space shown in the Figure 3-1.

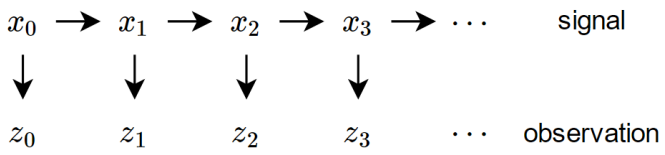

Figure 3-1 The filtering problem is to estimate sequentially the values of the hidden states x_t, given the values of the observation process $z_0, \cdots, z_t$ at any time step t.

The key idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights.

The state of the modeled object at time t is denoted x_t and its history is $X_t = \{x_0, x_1, \dots, x_t\}$. Similarly, the set of image features at time t is z_t with history $Z_t = \{z_0, z_1, \dots, z_t\}$. Note that no functional assumptions (e.g. linearity, Gaussianity, or unimodality) are made about densities in the general treatment, though particular choices will be made in due course in order to demonstrate the approach.

A somewhat general assumption [10] is made for the probabilistic framework that the object dynamics form a temporal Markov chain so that

$$p(x_t|X_{t-1}) = p(x_t|x_{t-1})$$

Given a continuous-valued Markov chain with independent observations, the conditional state-density p_t at time t is defined by

$$p_t(x_t) \equiv p(x_t|Z_t)$$

This represents all information about the state at time t that is deducible from the entire data-stream up to that time. The rule for propagation of state density over time is

$$p(x_t|Z_t) = k_t * p(z_t|x_t)p(x_t|Z_{t-1})$$

where

$$p(x_t|Z_{t-1}) = \sum_{x_{t-1}} p(x_t|x_{t-1})p(x_{t-1}|Z_{t-1})$$

and k_t is a normalization constant independent of x_t.

The effective prior $p(x_t|Z_{t-1})$ is actually a prediction taken from the posterior $p(x_{t-1}|Z_{t-1})$ from the previous time-step, onto which is superimposed one time-step from the dynamical model, which is expressed in $p(\mathbf{x}_t|Z_{t-1})$.

The Condensation algorithm [10] is based on factored sampling but extended to apply iteratively to successive images in a sequence. Given that the process at each time-step is a self-contained iteration of factored sampling, the output of an iteration will be a weighted, time-stamped sample-set, denoted $\left\{s_t^{(n)}, n = 1, \dots, N\right\}$ with weights $\pi_t^{(n)}$, representing approximately the conditional state-density $p(x_t|Z_t)$ at time t.

The prior $p(x_t|Z_{t-1})$ is derived from the sample set representation $\left\{\left(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}\right), n = 1, \dots, N\right\}$ of $p(x_{t-1}|Z_{t-1})$, the output from the previous time-step, to which prediction $p(x_t|Z_{t-1})$ must then be applied.

The algorithm samples new N particles $\left\{s'^{(n)}, n = 1, \dots, N\right\}$ from the samples at previous time-step samples $\left\{\left(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}\right), n = 1, \dots, N\right\}$ with associated weight $\pi_{t-1}^{(n)}$ as probability. In the prediction phase, the process is divided into deterministic drift and

stochastic perturbation. The deterministic drift corresponds to the motion of the target object, and the random perturbation is the Brownian motion of each particle.

Finally, the weights of new N particles should be estimated from the measured features z_t

$$\pi_t^{(n)} = k * p\left(z_t \middle| x_t = s_t^{(n)}\right)$$

where k is a normalization constant that $\sum_{n=1}^{N} \pi_t^{(n)} = 1$.

Once we have the new N particles and weights $\left\{\left(s_t^{(n)}, \pi_t^{(n)}\right), n = 1, \ldots, N\right\}$, we can let the particle with the highest weight $s_t'^{(n)}$ or the mean state $\mathcal{E}[f(x_t)] = \sum_{n-1}^{N} \pi_t^{(n)} f\left(s_t^{(n)}\right)$ as the tracking target object. Therefore, we can iterate the process successively to the rest of the states.

3.3 Artificial Neural Network

Artificial neural networks [20] known as multilayer perceptrons are computational model which consists of multiple layers with hundreds of neurons and thousands of weights (connections) that can be tuned to minimize a specific loss function depending on the problem we are trying to solve.

Denote a $L+1$ layer network that consists of input N-dimensional data $\{x_i, i = 1, \dots, N\}$ which is also the *0*-th layer, $\{a_i^l, i = 1, \dots, d^l\}$ for each neuron of layer $1 \le l \le L$ where the output layer is $l = L$ and weights $\{w_{ji}^l, 1 \le i \le d^{l-1}, 1 \le j \le d^l, 1 \le l \le L\}$ connecting the *i*-th output from $l-1$ layer to *j*-th input for l layer. The output of each neuron is usually designed as $a_j^l = \sigma(\sum_{i=1}^{d^{l-1}} w_{ji}^l \, a_i^{l-1} + b_j^l)$, where b_j^l is the bias of *j*-th neuron at *l*-th layer, and $\sigma(\cdot)$ is called activation function which is a non-linear function to activate the neuron, e.g. sigmoid function, hyperbolic tangent, and so on. There is a specific loss function $E(\mathbf{x}, \mathbf{y})$ of the neural network, e.g. cross-entropy error or root mean square error, and so on. An example of a neural network is shown in Figure 3-2.

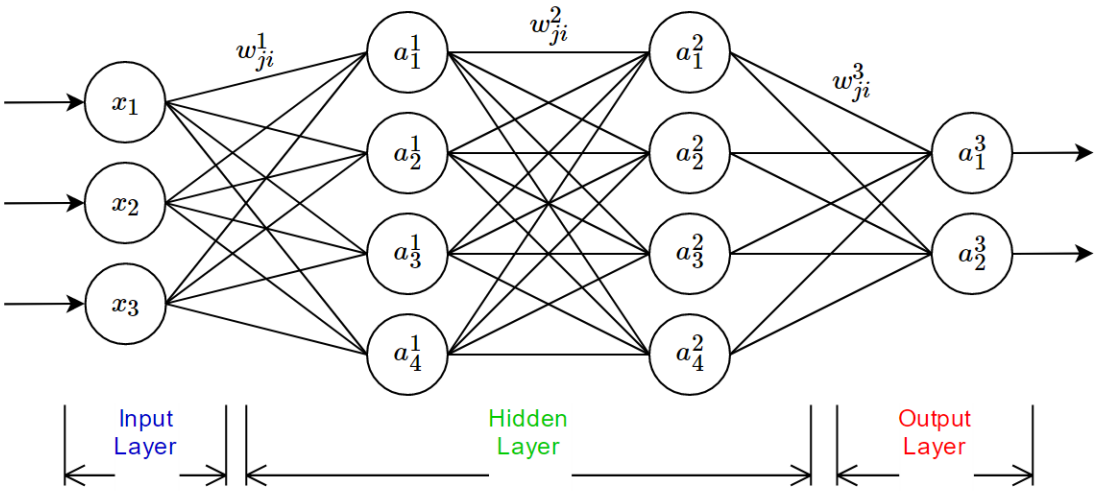

Figure 3-2 An example of a *4*-layer neural network with *3*-dimensional input $\mathbf{x}$ and two hidden layers with *4* neurons for each layer and *2*-dimensional output layer. There are

$(3+1)*4+(4+1)*4+(4+1)*2=46$ parameters to be tuned, and somehow show the degree of freedom of this model.

The primary method used to train the neural network is backpropagation [21] with stochastic gradient descent, then we can use it to minimize the loss function $E(\mathbf{x},\mathbf{y})$. The backpropagation can be split into two processes: forward pass and backward pass. In forward pass, the neural network passes the input data from input layer through the hidden layers to calculate the output value a_i of each neuron for each layer until the output layer, in which we can obtain the total error $E(\mathbf{x},\mathbf{y})$ of the current network for a batch of data. In backward pass, the gradient of each weight $\frac{\partial E}{\partial w_{ji}^{l}}$ and bias $\frac{\partial E}{\partial b_{j}^{l}}$ for layer $l=L,\dots,1$ are calculated backward by chain rule to reduce the loss of the network. For the deployment of neural network to application scenario, only forward pass is needed to make the prediction.

3.4 Convolutional Neural Network

Convolutional neural network [22], [8] is a special type of the ordinary feed-forward neural network, it also has weights and biases trained to minimize the loss function. In addition, convolutional neural network is biologically inspired by the visual cortex of

animal. Individual cortical neuron will stimulate a particular region of visual cortex called receptive field when it receives the information from outside world, and the similar concept used in convolutional neural network is shown in Figure 3-3. The main differences between ordinary neural network and convolutional neural network are local connectivity of neurons, subsampling (or called pooling), and parameter sharing.

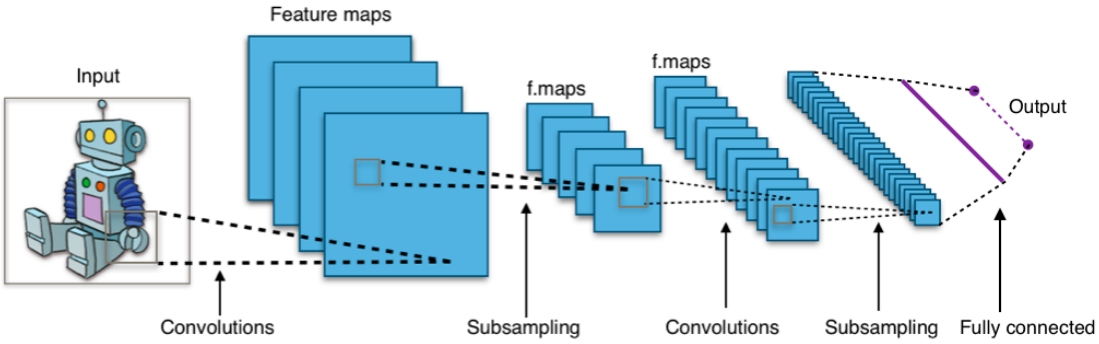

Figure 3-3 From [22], this is a typical architecture of convolutional neural network with input as an image, convolutional layers, subsampling (pooling) layers, and also there are fully connected layers at last several layers to classify the image.

The local connectivity, shown in Figure 3-4, is that each neuron of convolutional layers only connects to corresponding neurons of previous layer based on the local spatial relationship of training data. The convolutional layers are the core building block of convolutional neural network, and reduce considerable amount of parameters than the fully connected layers needed to be trained.

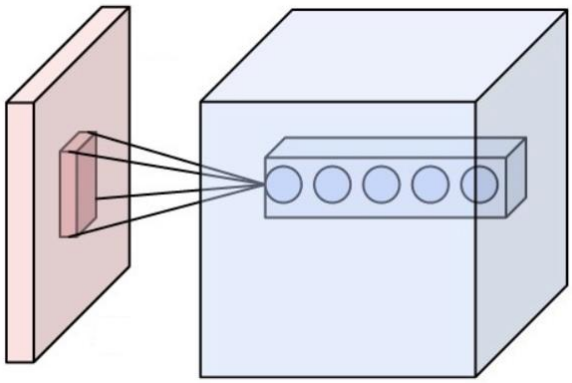

Figure 3-4 From [22], the neurons (blue) of convolutional neural network connect to corresponding neurons as its receptive field (red).

Recognizing the content of an image is trivial task to a human, even we shrink the image to quarter of the size, as shown in Figure 3-5. Similarly, the pooling layers are introduced to convolutional neural networks, which subsample the output of previous layer with given mask and specific strategy, for example max pooling shown in Figure 3-6.

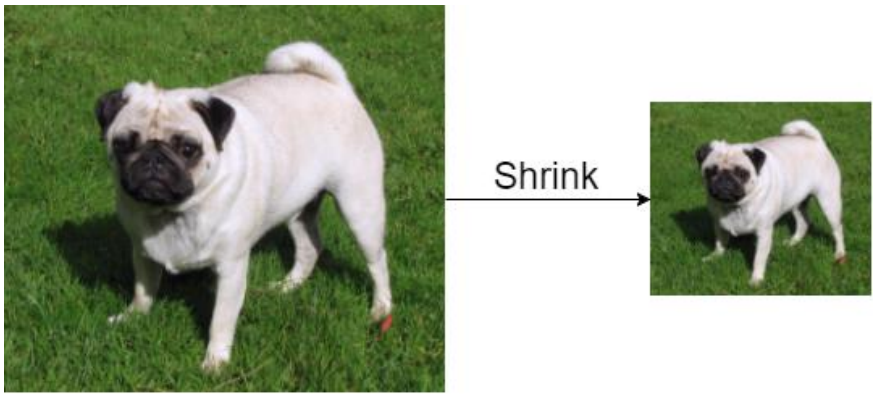

Figure 3-5 Even we shrink the image of a Pug dog to half of its original height and width,

we still can recognize that it is the same dog. Appropriately subsampling the image will not affect the content of the original image.

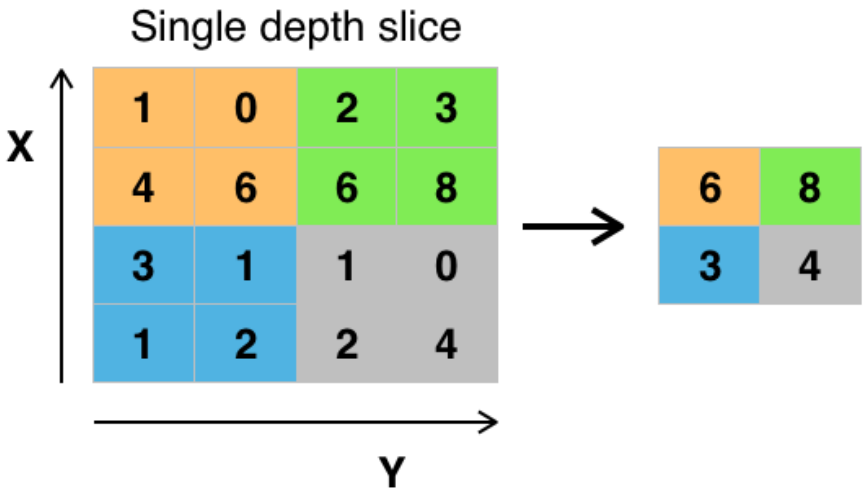

Figure 3-6 From [22], a max pooling with a 2×2 mask and the stride is 2.

In sharing parameter mechanism, the convolutional layer is modeled as a 3-dimensional cube (Figure 3-7) that each slice of the cube is the 2-dimensional convolution on the output of previous layer and the trainable parametric filter of that slice. The depth of the convolutional layer is the number of trainable filters we want to apply to the previous layer. By this strategy, we still can diminish the amount of parameters effectively.

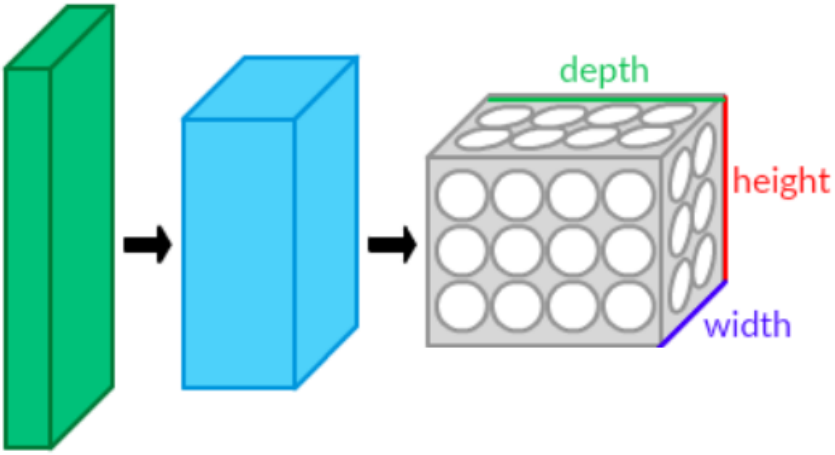

Figure 3-7 The 3-dimensional formation of convolutional layer, in which the slices along depth-axis are the outcome of convolution of previous layer and a trainable filter, like feature extractor.

By these basic properties of convolutional neural network, the number of parameters can be reduced significantly to avoid the overfitting, and better performance can be achieved in spatially relevant problems, e.g. image recognition, speech analysis, and natural language processing.

3.5 Background Subtraction

Background subtraction, known as moving detection and foreground detection, is a technique in computer vision that extracts the moving objects in video from a static

camera. A common method of background subtraction is to model a background image that the foreground can be obtained by 'subtracting' the current image by background image due to the camera is fixed.

In [17], the background is modeled by an adaptive mixture Gaussian models. They assume that the pixel value at time t is X_t, and each pixel is represented by a mixture of K Gaussian distributions. The probability of the occurrence of X_t at that pixel is

$$P(X_t) = \sum_{i=1}^{K} w_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

where K is the number of distributions; $w_{i,t}$ is the weight of i-th Gaussian; $\mu_{i,t}$ is the mean value of i-th Gaussian at time t; $\Sigma_{i,t}$ is the covariance matrix of i-th Gaussian at time t; and η is a Gaussian probability density function

$$\eta(X_t, \mu_{i,t}, \Sigma_{i,t}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1}(X_t - \mu_t)\right)$$

Each new X_t will be checked against the K Gaussians, and match with one of the Gaussians as the pixel value within the threshold 2.5 standard deviations of a distribution.

$$X_t \text{ in } i^{th} \text{ Gaussian} \Rightarrow \frac{(X_t - \mu_{i,t})}{\sigma_{i,t}} < 2.5$$

After that, the weight of K Gaussians will be updated by

$$w_{i,t} = (1-\alpha)w_{i,t-1} + \alpha(M_{i,t})$$

where $M_{i,t}$ is 1 and 0 representing whether X_t matches with i-th Gaussian, and α is a learning rate how fast the pixel value will be adaptive into the Gaussian. Then, each Gaussian parameter should be updated by

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)$$

where the second learning rate is

$$\rho = \alpha\eta(X_t \mid \mu_k, \sigma_k)$$

Finally, the first B distributions will be regarded as backgrounds where

$$B = \text{argmin}_b\left(\sum_{k=1}^{b} w_k > T\right)$$

and T is a threshold that larger T can adapt to the slightly shaking situations, e.g. leaves on the tree.

Chapter 4 Methodology

4.1 Overview

The detailed methods used in our tracking system is described in this chapter. In Section 4.2 , the tracking algorithm based on particle filter is presented and split into three parts: sampling (Subsection 4.2.1), motion estimation (Subsection 4.2.2), and measurement (Subsection 4.2.3). After that, the convolutional neural network used as a head classifier will be explained in Section 4.4 , and the maximal rectangle overlap rate with foreground candidates obtained by background subtraction is estimated in Section 4.3 . Eventually, the similarity estimated from color histogram is shown in Section 4.5 .

4.2 Particle Filter Framework

Our tracking algorithm is based on particle filter framework, which consists of three parts: sampling, motion estimation, and measurement. Specifically, the particle filter framework [10] is to estimate the propagation of state density over time

$$P(\mathbf{x}_t|Z_t) = k_t P(\mathbf{z}_t|\mathbf{x}_t) P(\mathbf{x}_t|Z_{t-1})$$

where

$$P(\mathbf{x}_t|Z_{t-1}) = \sum_{\mathbf{x}_{t-1}} P(\mathbf{x}_t|\mathbf{x}_{t-1})P(\mathbf{x}_{t-1}|Z_{t-1})$$

and k_t is the normalization constant. Clearly, $P(\mathbf{x}_{t-1}|Z_{t-1})$ is the state density sampled from previous time-step; $P(\mathbf{x}_t|\mathbf{x}_{t-1})$ is to model the object motion of state; and $P(\mathbf{z}_t|\mathbf{x}_t)$ is the measurement of the state, which is the weight of the corresponding particle.

The i-th particle at time-step t is considered as $\mathbf{x}_t^i = \left(x_t^i, y_t^i, s_t^i\right)$ to represent a bounding box of the frame at time-step t, where (x_t^i, y_t^i) is the center of that bounding box, and s_t^i is the scaling factor relative to an initial base width, which we will describe the detail later.

For the input video which we want to track, denote that I_t where $t = 0, 1, \ldots$ are the image sequences of that video. At the first frame I_0, we have to designate the bounding box of target teacher to track, and assume that (x_{ul}, y_{ul}) is the upper left point of the bounding box, and (w, h) are the width and height of that bounding box. In addition, let Ω denote the bounding box patch which we can use it to identify other image patch by similarity. To initialize the particle filter, we set base width $w_{base} = 100$, then we have $s_0 = \frac{w}{w_{base}}$, $x_0 = x_{ul} + 0.5 * s_0 w_{base}$, $y_0 = y_{ul} + 0.5 * s_0 w_{base}\left(\frac{h}{w}\right)$.

The initial N particles are set as

$$\mathbf{x}_0^i = (x_0, y_0, s_0), \forall \ 1 \leq i \leq N$$

and the initial weights of particles are

$$\pi_0^i = \frac{1}{N}, \forall \ 1 \leq i \leq N$$

That is all initial N particles have the same bounding box as the tracking bounding box at the first frame, and the weight of each particle is also the same. Once we have the initial particles, we can iterate the particle filter to obtain the conditional state-density $P(\mathbf{x}_t|Z_t)$ at time t with subsequent frames.

4.2.1 Particle Filter Sampling

For each subsequent time-step t, we sample the new N particles $\{\mathbf{x}_t^i\}_{i=1}^N$ from previous time-step particles $\{\mathbf{x}_{t-1}^i\}_{i=1}^N$ with corresponding weights $\{\pi_{t-1}^i\}_{i=1}^N$, that is the particles with higher weights will have higher probabilities to be sampled. On the contrary, the lower weight particles are less sampled.

4.2.2 Particle Filter Motion Estimation

We estimate the object dynamics of each particle as first order linear difference, which is

$$\mathbf{x}_t^i = A\left(\mathbf{x}_{t-1}^i - \mathbf{x}_{t-2}^i\right) + B\mathbf{w}_t + \mathbf{x}_{t-1}^i$$

where A, B are deterministic (velocity) and stochastic (Brownian motion, larger B with larger variance of particle distribution) motion coefficient, and $\mathbf{w}_t$ is an independent standard normal variable. That is, we assume that the transition state density is

$$P(\mathbf{x}_t|\mathbf{x}_{t-1}) \propto \exp(-\frac{1}{2}\|B^{-1}((\mathbf{x}_t - \mathbf{x}_{t-1}) - A(\mathbf{x}_{t-1} - \mathbf{x}_{t-2}))\|^2)$$

To be precise, we update the $\mathbf{x}_t^i$ by

$$x_t^i = 0.5\left(x_{t-1}^i - x_{t-2}^i\right) + 0.15w_{t,x} + x_{t-1}^i$$

$$y_t^i = 0.5\left(y_{t-1}^i - y_{t-2}^i\right) + 0.15w_{t,y} + y_{t-1}^i$$

$$s_t^i = 0.5\left(s_{t-1}^i - s_{t-2}^i\right) + 0.01w_{t,s} + s_{t-1}^i$$

$$\mathbf{x}_t^i = \left(x_t^i, y_t^i, s_t^i\right)$$

4.2.3 Particle Filter Measurement

After the particles at current time-step $\left\{\mathbf{x}_t^i\right\}_{i=1}^N$ are determined, we have to estimate

the weight of each particle by the measurement $P(\mathbf{z}_t|\mathbf{x}_t)$. To compute $P(\mathbf{z}_t|\mathbf{x}_t)$, we assume that

$$P(\mathbf{z}_t|\mathbf{x}_t) = \big(P_{head}(\mathbf{x}_t) + FG(\mathbf{x}_t)\big) * S(\mathbf{x}_t, \Omega)$$

where $P_{head}(\mathbf{x}_t)$ is the probability of the upper portion of the bounding box represented by $\mathbf{x}_t$ is human head or not; $FG(\mathbf{x}_t)$ is the probability of bounding box $\mathbf{x}_t$ being a foreground candidates; and $S(\mathbf{x}_t, \Omega)$ is the similarity between the patch surrounded by the bounding box $\mathbf{x}_t$ and the target patch Ω. Moreover, $P_{head}(\mathbf{x}_t)$ is predicted by a VGG-like [15] convolutional neural network; $FG(\mathbf{x}_t)$ is calculated by rectangle intersection area over union area with foreground objects generated by background subtraction; and $S(\mathbf{x}_t, \Omega)$ is estimated by the correlation coefficient of color histogram of two given patches. We will go through the detail in next subsection.

As a consequence, the weight of each particle $\{\pi_t^i\}_{i=1}^N$ can be obtained by

$$\pi_t^i = \frac{\Big(\big(P_{head}(\mathbf{x}_t) + FG(\mathbf{x}_t)\big) * S\big(\mathbf{x}_t^i, \Omega\big)\Big)}{\sum_{j=1}^N \big(P_{head}(\mathbf{x}_t) + FG(\mathbf{x}_t)\big) * S(\mathbf{x}_t^j, \Omega)}$$

Once we have the particles and its weight pair $\{(\mathbf{x}_t^i, \pi_t^i)\}_{i=1}^N$, the tracking target can be estimated by

$$\mathcal{E}[\mathbf{x}_t] = \sum_{i=1}^{N} \pi_t^i * \mathbf{x}_t^i$$

the weighted mean state at time-step t.

Finally, we can continue processing the subsequent frames iteratively. The flowchart of the tracking algorithm is shown in Figure 4-1.

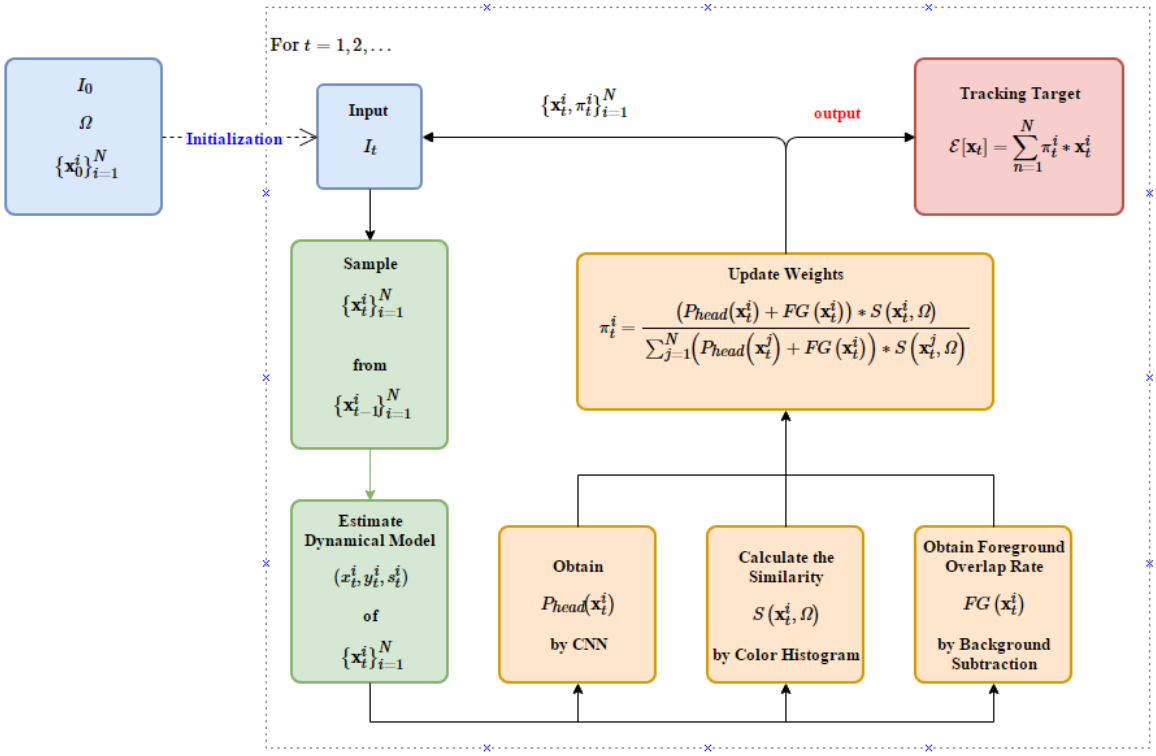

Figure 4-1 The flowchart of our tracking algorithm. The blue parts are the input images and initial information. The green parts are to make the prediction of prior distribution $P(x_t|Z_{t-1})$ of particle filter. The orange parts are to measure the posterior distribution $P(z_t|x_t)$ of particle filter. The red part is the output of tracking target.

4.3 Background Subtraction

We use background subtraction to detect the foreground candidates as shown in Figure 4-2, and assume that the probability of a bounding box $\mathbf{x}_t^i$ being a foreground object is proportional to $FG(\mathbf{x}_t^i)$ the maximal rectangle overlap rate with one of the foreground objects.

Denote that K foreground objects generated by background subtraction are bounding boxes $\{\Lambda_k\}_{k=1}^K$, then

$$FG(\mathbf{x}_t^i) = \max_k \frac{|\mathbf{x}_t^i \cap \Lambda_k|}{|\mathbf{x}_t^i \cup \Lambda_k|}$$

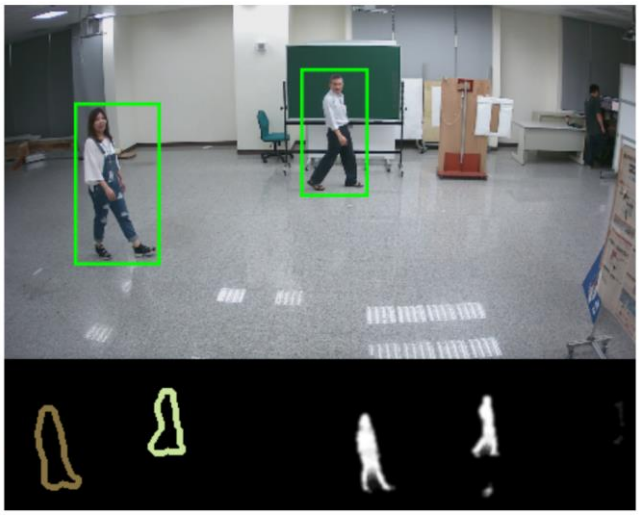

Figure 4-2 (a) The foreground candidates are represented as green bounding boxes. (b) The lower left part is the contours combined with connected components algorithms. (c)

The lower right part is the foreground pixels extracted by background subtraction.

4.4 CNN as a Head Classifier

In the measurement of particles of the tracking process in the particle filter framework, we have to estimate $P(\mathbf{z}_t|\mathbf{x}_t)$. We model the $P_{head}(\mathbf{x}_t^i)$ as a head classifier implemented by a 4-layer convolutional neural network to predict the probability of being a human head of an image patch in Figure 4-3.

Figure 4-3 The upper portion (green) of the input patch is used to predict the probability of human head $P_{head}(\mathbf{x}_t^i)$.

The human head dataset used to train our network is from [18], which collects 369,846 human heads annotated in 224,740 video frames from 21 Hollywood movies, shown in Figure 4-4. Moreover, we also label additional 1,400 heads and non-heads respectively to test the network in our scenario, as shown in Figure 4-5.

Figure 4-4 Here are some examples of training images. (a) The upper $12 \times 8 = 96$

images are head images. (b) The lower $12 \times 8 = 96$ images are non-head images.

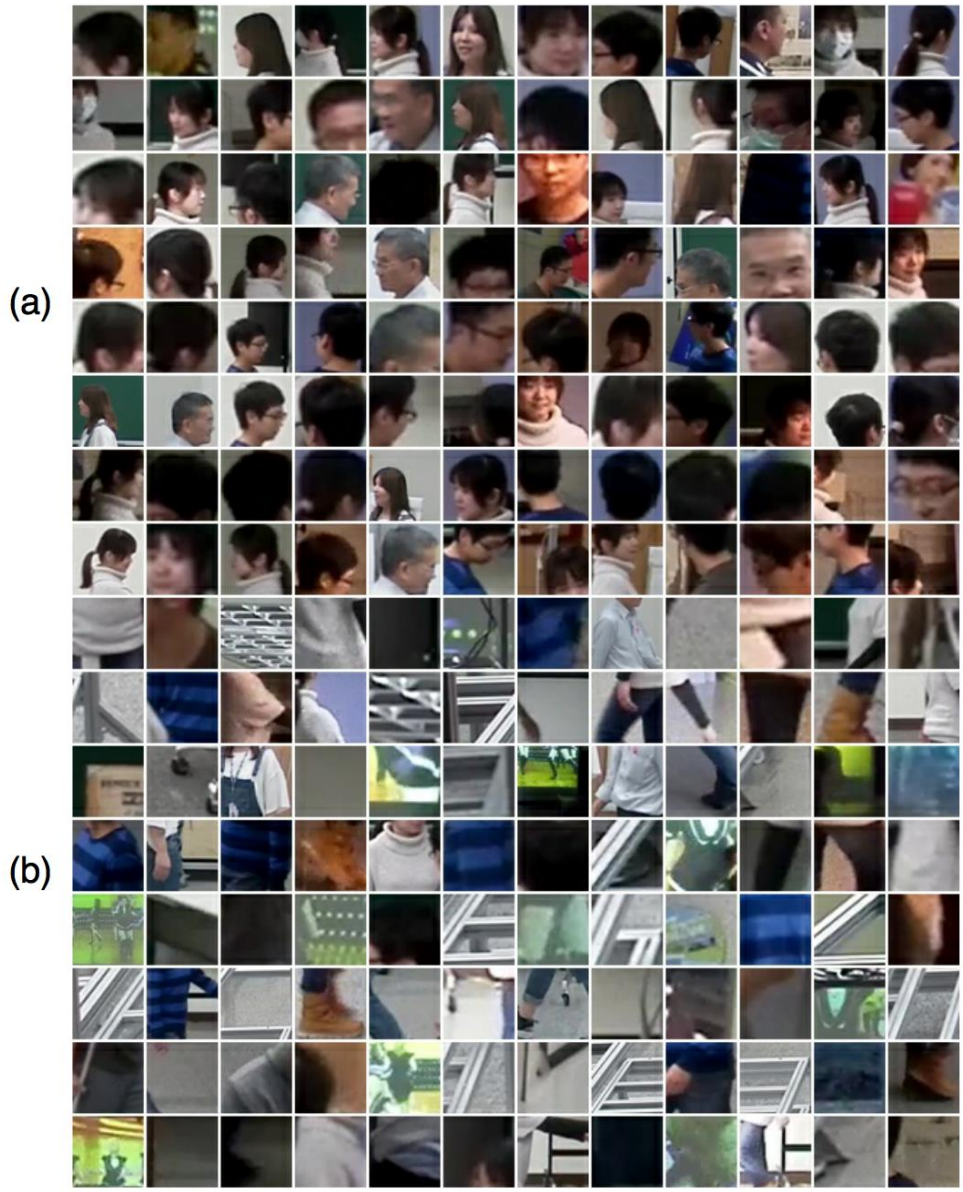

Figure 4-5 Here are some examples for testing in our scenario. (a) The upper $12 \times 8 = 96$ images are head images. (b) The lower $12 \times 8 = 96$ images are non-head images.

Our network architecture refers to VGG (Visual Geometry Group) 16-layer network [15], but the problem we want to solve which is to classify a given image into head or

non-head which is much easier than what VGG wants to solve i.e. to classify the input image into 1000 categories. For this reason, we diminish the size of the network from 16-layer to 4-layer. We gain some advantages from reducing the size of network, such as less overfitting, and computational and memory space efficiency. We have trained three different sizes of network, which are 8-layer, 5-layer, and 4-layer networks for comparison shown in Table 4-1. We adopt 4-layer network due to similar accuracy but much less computation.

Models	**Training Set Accuracy**	**Validation Set Accuracy**	**Training Time per Image (milliseconds)**	**Prediction Time per Image (milliseconds)**
8-layer	0.939	0.895	2.612	1.224
5-layer	0.911	0.891	0.352	0.194
4-layer	0.907	0.894	0.121	0.057

Table 4-1 The comparison between three different-size networks shows that more powerful model obtains lower error in the training set, but the gaps in validation set are not as great as before. Obviously, the size of the network is proportional to the execution time in both training and testing phase.

In training phase, we use stochastic gradient descent to train our network. We use a

popular deep learning framework, Caffe [11], to setup and deploy our neural network. Finally, we choose the 4-layer network to be the head classifier of our tracking system to predict $P_{head}(\mathbf{x}_t^i)$. We can roughly examine the performance of the head classifier in tracking video by estimating the probabilities of grids split from the video and showing the result as a heat map in Figure 4-6 and Figure 4-7.

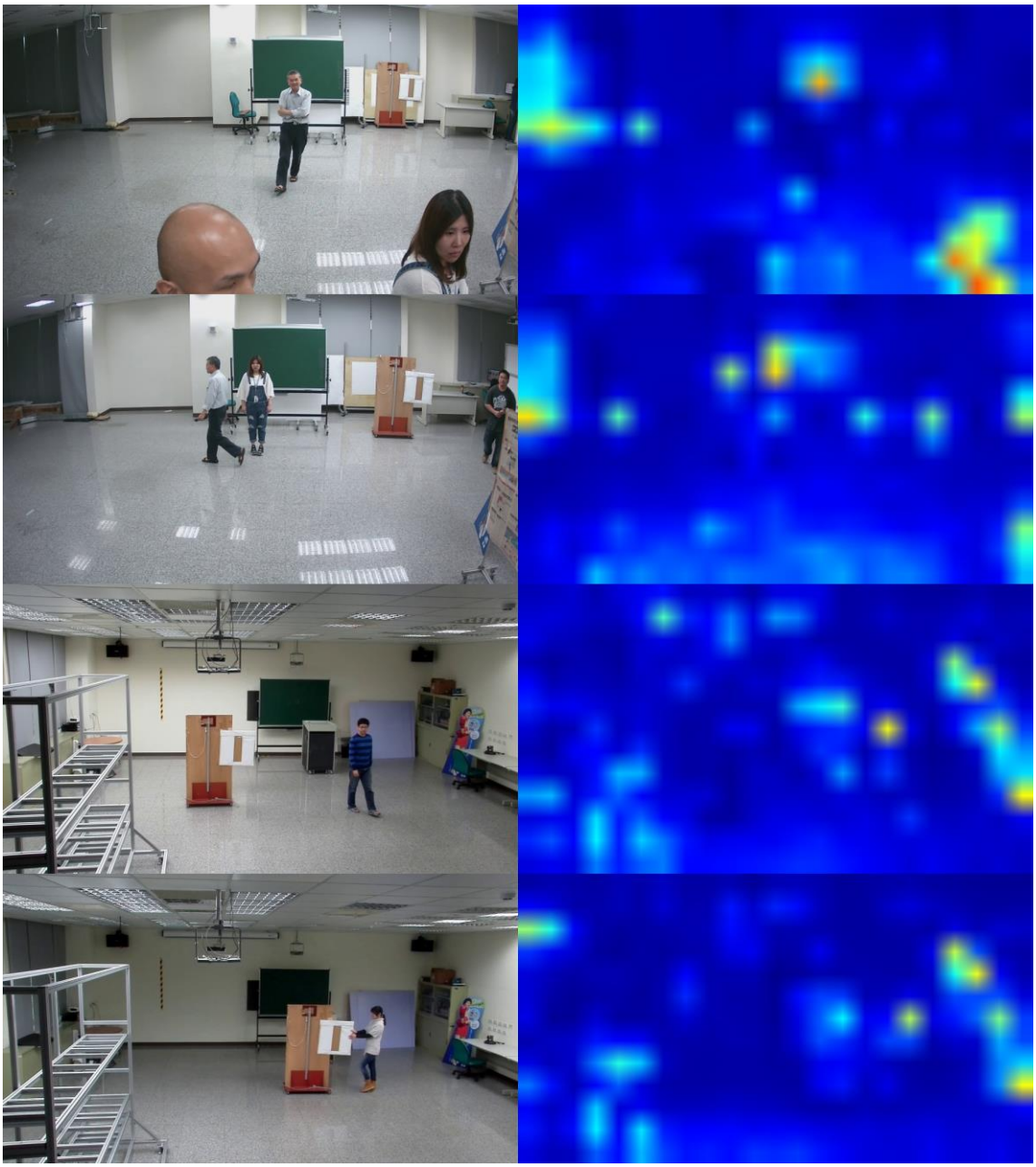

Figure 4-6 Some good results of our head classifier are shown as heat map that the positions with head at the left side correspond to the higher temperature at the right side, and vice versa. There are some misdetections at low intensity locations.

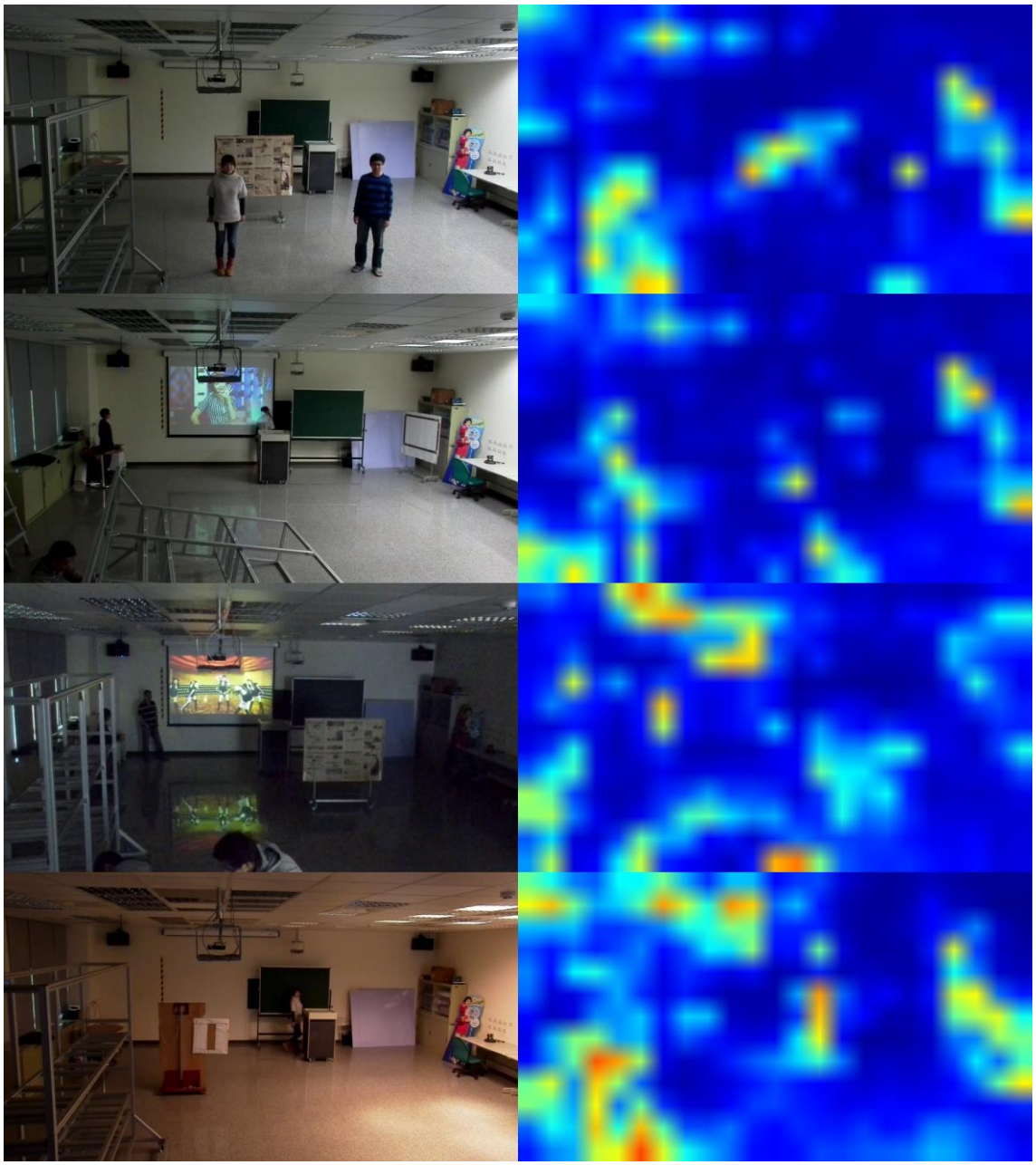

Figure 4-7 Some defect results of our head classifier are shown as heat map that many non-head positions at the left side also correspond to higher temperature at the right side as misdetections.

4.5 Color Histogram

The final measurement in particle filter framework is to estimate the similarity between the target teacher Ω and the bounding box of current particle x_t^i. We convert the color space of the given image from RGB (Red, Green, and Blue) to HSV (Hue, Saturation, and Value) and take the Hue channel to calculate the color histogram of specific blocks of the input patch, where we divide the patch into 4×3 blocks in Figure 4-8.

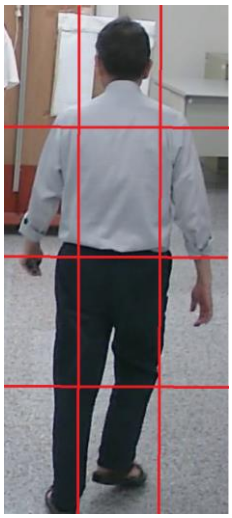

Figure 4-8 The given patch is divided into 4×3 blocks.

The color histogram of the four blocks in middle patch will be separately calculated into 90 bins, and concatenate together as $90 \times 4 = 360$ bins to reserve spatial

characteristic. Then, the 360 bins will be normalized to sum up to 1, and become a 360-dimensional descriptor of the input patch. The similarity $S(\Omega, \mathbf{x}_t^i)$ between two patches is estimated by the correlation of two descriptors.

Chapter 5 Experimental Results

5.1 Overview

In this chapter, we show the experimental results of our tracking algorithm. We will explain how we evaluate the performance of tracking process in Section 5.2 . Furthermore, the tracking results and some observations of experiments are shown in Section 5.3 . The tracking system is built on the environment in Table 5-1.

Experiment Environments	
Operating System	Ubuntu 16.04
Central Processing Unit	Intel® Xeon® E3-1231 3.4GHz
Random Access Memory	16 Gigabytes
Programming Language	C++ with OpenCV 3.2
Graphic Processing Unit	Nvidia GeForce GTX 950
Deep Learning Framework	Caffe

Table 5-1 The environment to develop and conduct experiments is shown in this table. We use OpenCV to implement most computer vision tasks and employ Caffe as our deep learning framework to predict the probability of head.

5.2 Evaluation

We collect five indoor videos in Figure 5-1 to test our tracking algorithm with manual labeled tracking target in each frame. For quantitative evaluation, we use precision plot and success plot from [25].

Figure 5-1 Five testing videos labeled with the ground truth in blue bounding box at the first frame are shown here to initialize the tracking process.

We use precision plot to calculate the Euclidean distance between the center of the bounding boxes of labeled ground truth and the tracking result, then evaluate the accuracy under given thresholds. After that, the ultimate performance is estimated as the area under the precision curve. Denote the center of ground truth rectangle as $\mathbf{x}_g$, and the center of tracking result as $\mathbf{x}_t$. The distance is computed as

$$D(\mathbf{x}_g, \mathbf{x}_t) = \|\mathbf{x}_g - \mathbf{x}_t\|$$

Success plot computes the rectangle overlap rate by rectangular intersection over union between the bounding box of labeled ground truth and the tracking result. The same as the precision plot, the accuracy is also evaluated by thresholds, and the area under the success plot is considered as the final performance for comparison. Denote the rectangle of labeled ground truth as r_g and the tracking result as r_t, the rectangle overlap rate is defined as

$$Overlap\ \ rate = \frac{|r_g \cap r_t|}{|r_g \cup r_t|}$$

An example of precision plot and success plot of one of the testing videos are shown in Figure 5-2.

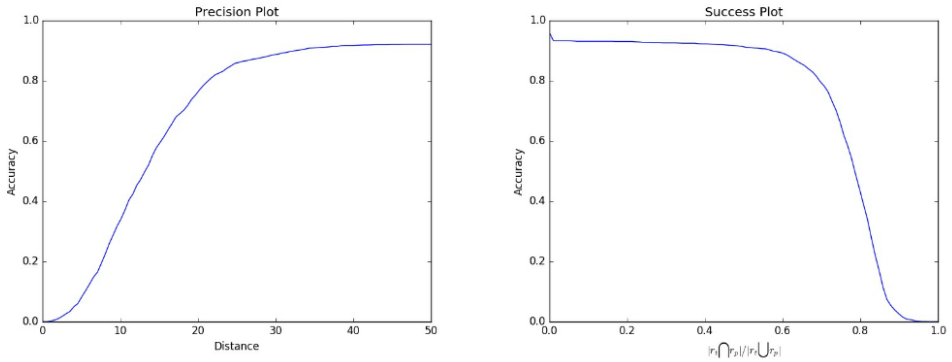

Figure 5-2 (a) The precision plot of detected and ground truth box center distance. (b) Success plot of overlapping rate (intersection/union of detected and ground truth boxes) of our tracking algorithm.

5.3 Results

We initialize the tracking algorithm with the ground truth bounding box of the first frame of each testing video, then generate the tracking result of subsequent frames. The snapshots of one of our tracking results are shown in Figure 5-3.

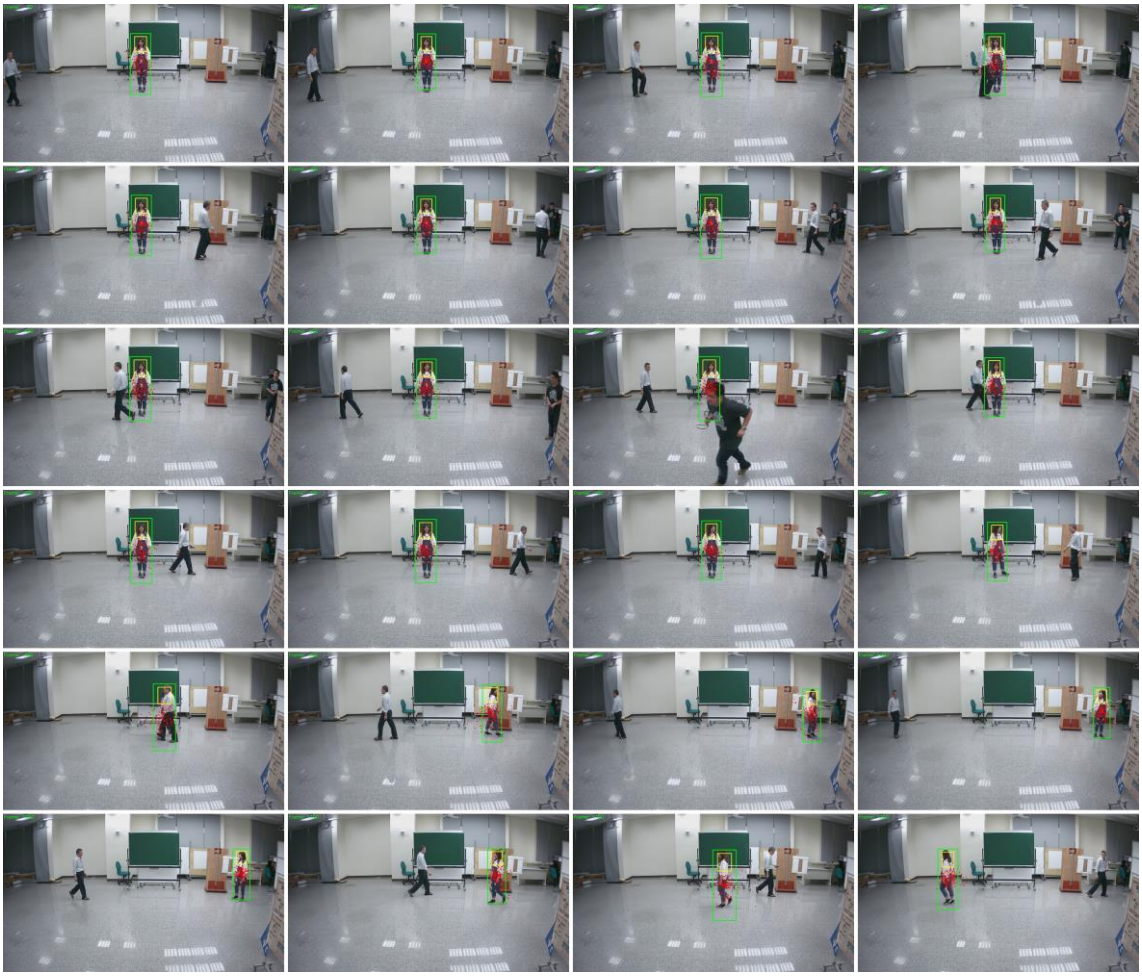

Figure 5-3 The snapshots of our tracking result show that we can track the designated girl frame by frame in the video successfully. The green bounding box is the tracking result predicted from the red particles, and the yellow rectangle is the predicted head position.

Moreover, our tracking algorithm can deal with occlusion conditions, such as the target person is occluded by another person in Figure 5-4. Two people cross each other in Figure 5-5. The target person walks behind a barrier in Figure 5-6.

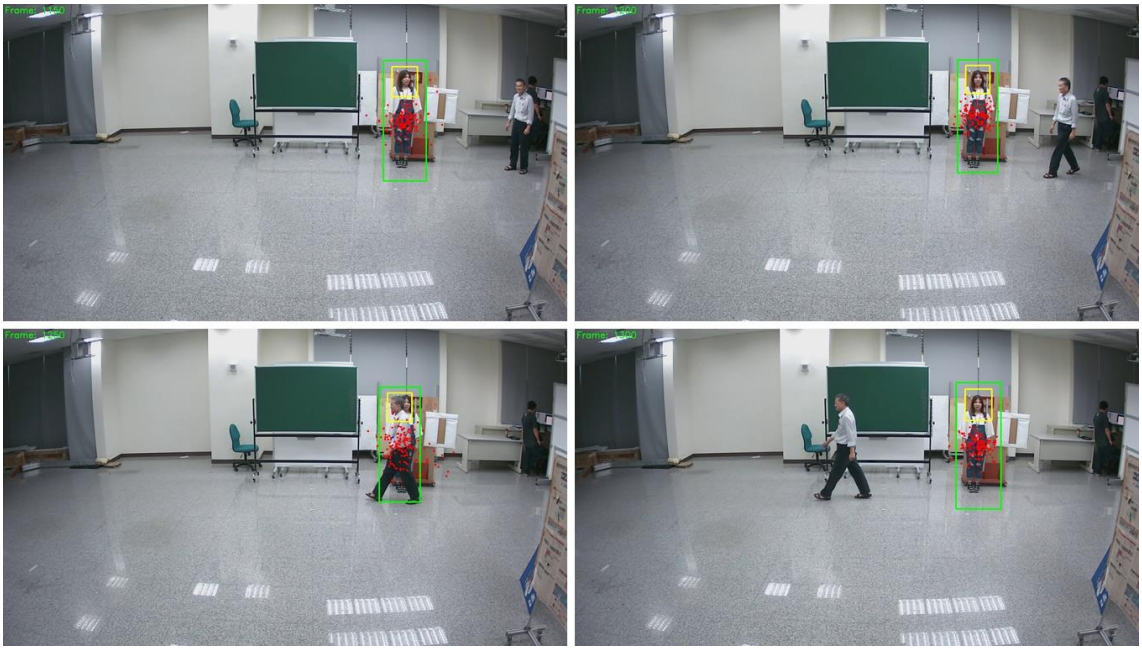

Figure 5-4 The tracking target person is occluded by another person walking across.

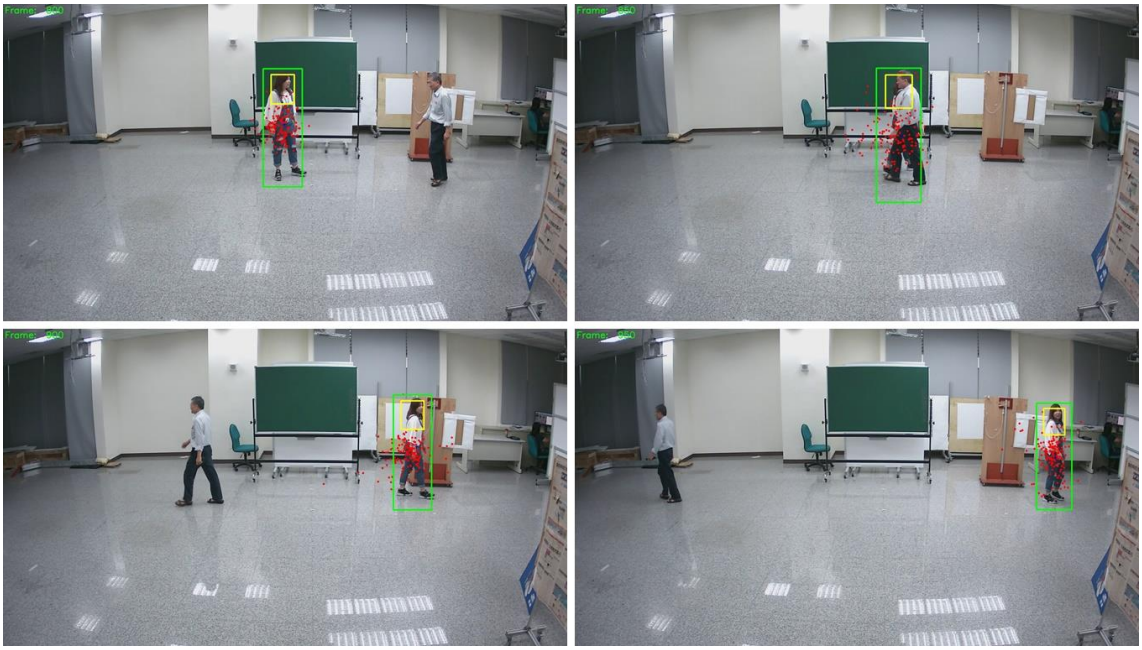

Figure 5-5 Two people cross each other.

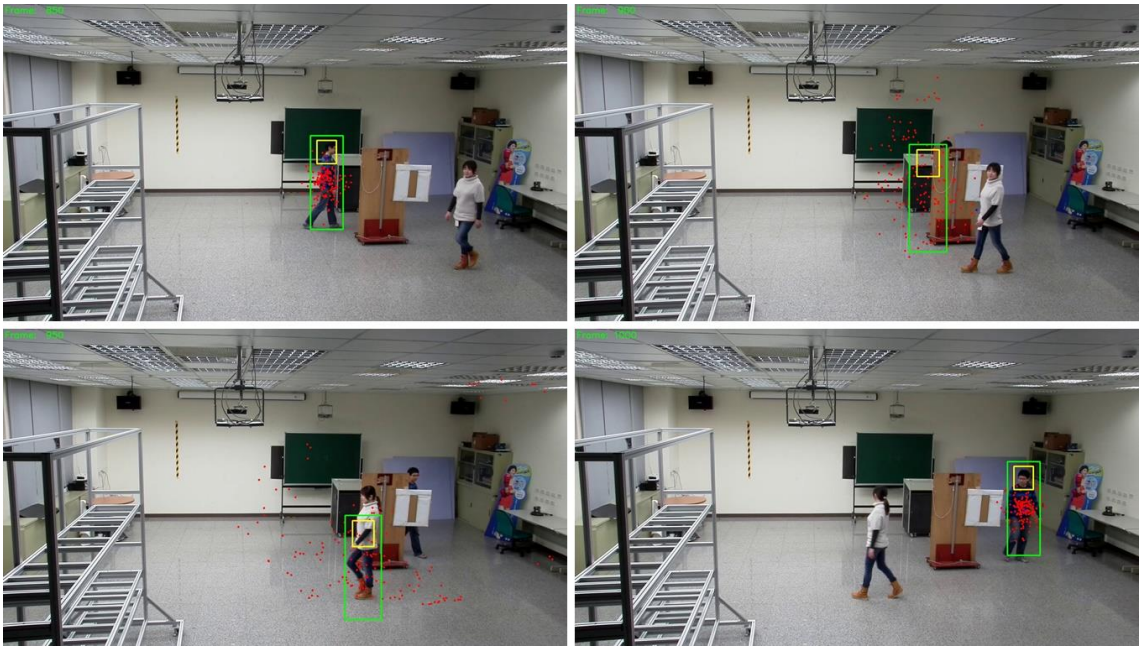

Figure 5-6 The tracking target person walks behind a barrier board. When the tracking target disappears in the view, the particles will spread over the scene. Once the tracking target appears again in the view, the particles can re-lock the target soon.

However, there are some difficult conditions that the tracking result will drift or fail, such as low light environments in Figure 5-7 or sudden light change in Figure 5-8. The reason for these drawbacks may be caused by using color to estimate the similarity with the target, so the poor illuminance tolerance is an issue of this algorithm.

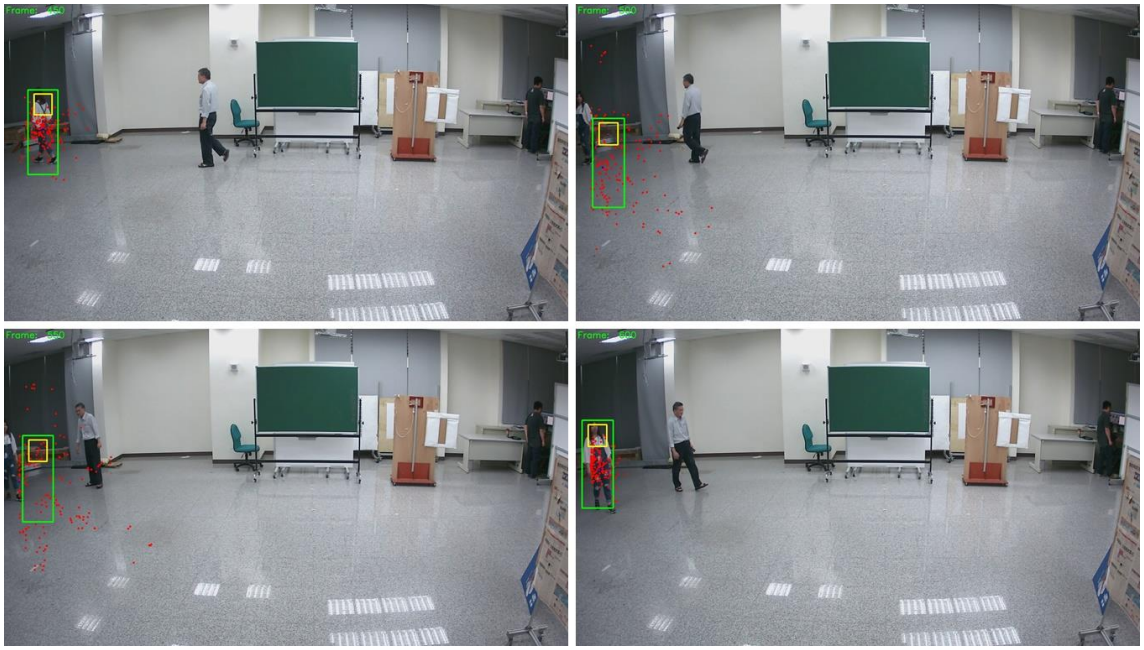

Figure 5-7 The target will be lost when she go into the dark area until she walks back to normal light area.

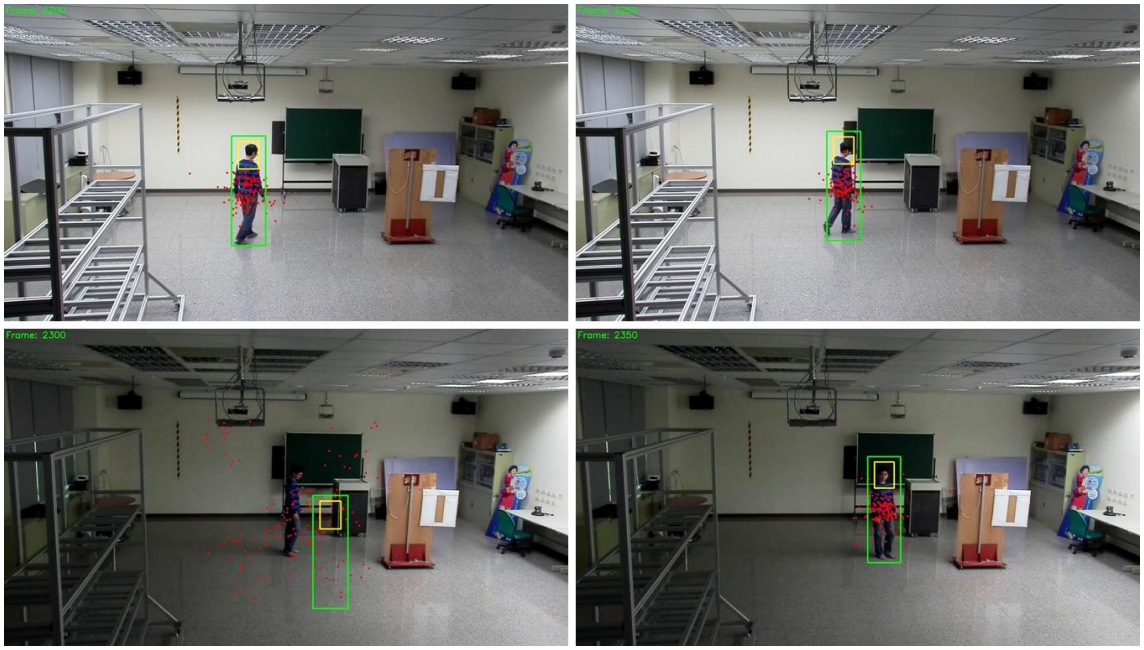

Figure 5-8 The tracking target is lost momentarily when the light changes suddenly.

We compare performance of different parameter settings of our tracking algorithm. We set different numbers of particles as $30, 50, 70, 100, \text{and } 150$ of the particle filter framework in Figure 5-9 and Table 5-2. Obviously, increasing the number of particles will approximate $P(\mathbf{x}_t|Z_t)$ more accurately. However, the computational cost is proportional to the number of particles. Therefore, we have to choose between accuracy and efficiency: 50 on Nvidia TK1.

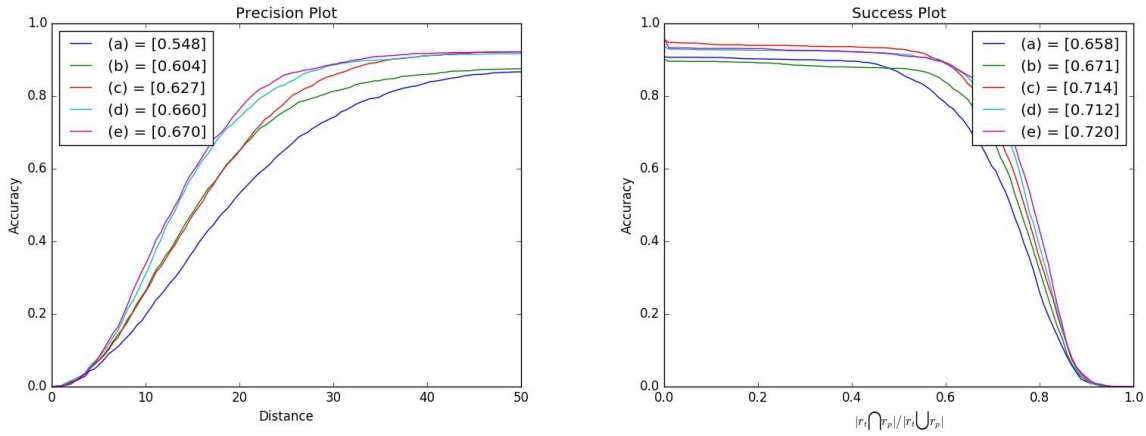

Figure 5-9 The precision plot and success plot of different numbers of particles. The numbers of particle filter are (a) 30 $(\text{PAUC} = 0.548, \text{SAUC} = 0.658)$, (b) 50 $(\text{PAUC} = 0.604, \text{SAUC} = 0.671)$, (c) 70 $(\text{PAUC} = 0.627, \text{SAUC} = 0.714)$, (d) 100 $(\text{PAUC} = 0.660, \text{SAUC} = 0.712)$, (e) 150 $(\text{PAUC} = 0.670, \text{SAUC} = 0.729)$. SAUC: Success plot Area Under Curve.

	Avg. Time (ms)	Avg. DL Time (ms)	AUC of Precision	AUC of Success
(a)	27.574	18.498	0.548	0.658
(b)	36.264	28.239	0.604	0.671
(c)	48.499	39.698	0.627	0.714
(d)	64.248	55.998	0.660	0.712
(e)	99.768	90.402	0.670	0.720

Table 5-2 The results of different numbers of particles. The numbers of particle filter are (a) 30, (b) 50, (c) 70, (d) 100, (e) 150.

We also compare the performance with different motion parameters B for $B = 5, 15, 25, 35, \text{and } 45$ pixels under lower number of particles $N = 50$ setting, shown as Figure 5-10 and Table 5-3, and higher number of particles $N = 150$ setting in Figure 5-11 and Table 5-4. We observe that the motion parameter affects the performance significantly, when the number of particles is low. We have to take the situation into account. On the other hand, increasing the number of particles can reduce the unstable tracking, but the performance is still worse if the particles are too excited (faster motion).

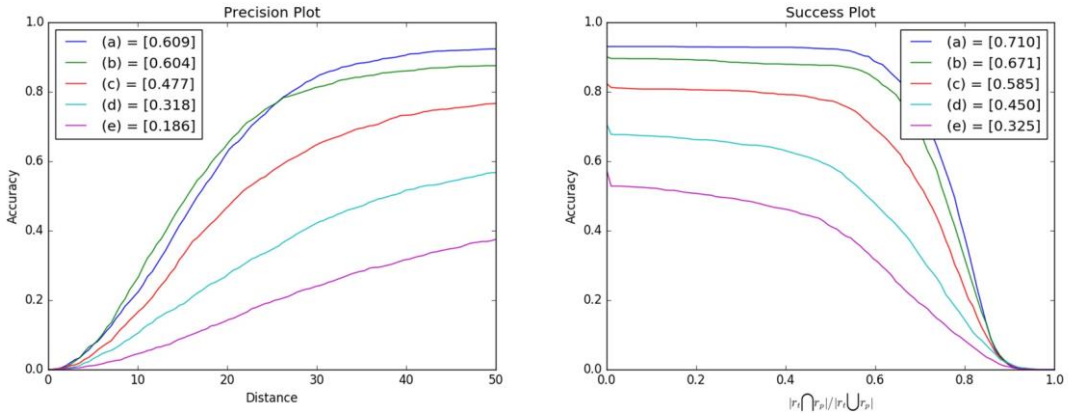

Figure 5-10 The precision plot and success plot of different motion estimation parameters under lower number of particles $N = 50$ setting. The motion parameters B are (a) 5 $(\text{PAUC} = 0.609, \text{SAUC} = 0.710)$, (b) 15 $(\text{PAUC} = 0.604, \text{SAUC} = 0.671)$, (c) 25 $(\text{PAUC} = 0.477, \text{SAUC} = 0.585)$, (d) 35 $(\text{PAUC} = 0.318, \text{SAUC} = 0.450)$, (e) 45 $(\text{PAUC} = 0.186, \text{SAUC} = 0.325)$ pixels.

	Avg. Time (ms)	**Avg. DL Time (ms)**	**AUC of Precision**	**AUC of Success**
(a)	37.290	28.285	0.609	0.710
(b)	36.264	28.239	0.604	0.671
(c)	38.836	29.817	0.477	0.585
(d)	41.035	32.129	0.318	0.450
(e)	40.691	31.716	0.186	0.325

Table 5-3 The results of different motion estimation parameters under lower number of particles $N = 50$ setting. The motion parameters B are (a) 5, (b) 15, (c) 25, (d) 35, (e) 45 pixels. For lower number of particles ($N = 50$), Brownian motion distribution variance B should be lower, such as 5.

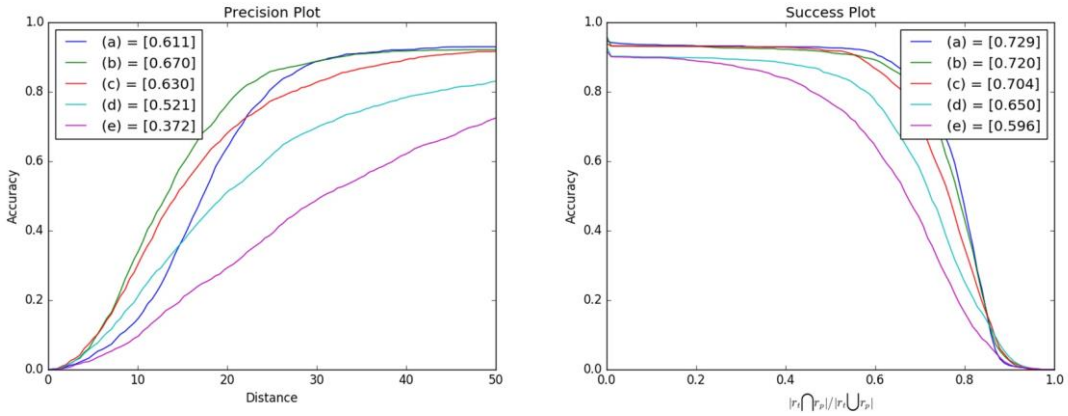

Figure 5-11 The precision plot and success plot of different motion estimation parameters under higher number of particles $N = 150$ setting. The motion parameters B are (a) 5 $(\text{PAUC} = 0.611, \text{SAUC} = 0.729)$, (b) 15 $(\text{PAUC} = 0.670, \text{SAUC} = 0.720)$, (c) 25 $(\text{PAUC} = 0.630, \text{SAUC} = 0.704)$, (d) 35 $(\text{PAUC} = 0.521, \text{SAUC} = 0.650)$, (e) 45 $(\text{PAUC} = 0.372, \text{SAUC} = 0.596)$ pixels.

	Avg. Time (ms)	**Avg. DL Time (ms)**	**AUC of Precision**	**AUC of Success**
(a)	89.225	81.181	0.611	0.729
(b)	99.768	90.402	0.670	0.720
(c)	94.169	86.243	0.630	0.704
(d)	95.394	87.439	0.521	0.650
(e)	96.797	88.806	0.372	0.596

Table 5-4 The results of different motion estimation parameters under higher number of particles $N = 150$ setting. The motion parameters B are (a) 5, (b) 15, (c) 25, (d) 35, (e) 45 pixels. For higher number of particles $(N = 150)$, Brownian motion distribution variance B should be lower, such as 5, but still stable with larger B.

Chapter 6 Conclusion

In this thesis, we develop a nearly real-time teacher tracking and video recording system, which can track a target teacher in the classroom successfully. The tracking algorithm can handle general tracking issues, such as the target person is occluded by other person or barriers; or the target person disappears from the camera view for a short time.

However, there are some known drawbacks that our algorithm is not perfect under some difficult conditions, such as the low illuminance area, sudden light change, or colored ambient lighting. These issues are caused by the simple modeling of target object appearance similarity as correlation coefficient of color histograms.

In general, our tracking system can work smoothly under some constraints, and track the target teacher correctly.

Chapter 7 References

[1]. M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking,” *IEEE Transactions on Signal Processing*, volume 50, No. 2, pp. 174 – 188, 2002.

[2]. B. Babenko, M. H. Yang and S. Belongie, “Visual Tracking with Online Multiple Instance Learning,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, pp. 983 – 990, 2009.

[3]. B. Benfold and I. Reid, “Stable Multi-Target Tracking in Real-Time Surveillance Video,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* Colorado Springs, Colorado, pp. 3457 – 3464, 2011.

[4]. M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, “Online Multiperson Tracking-By-Detection from a Single, Uncalibrated Camera,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 33, No. 9, pp. 1820 – 1833, 2011.

[5]. D. Comaniciu, V. Ramesh, and P. Meer, “Real-time Tracking of Non-Rigid Objects Using Mean Shift,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Carolina, volume 2, pp. 142 – 149, 2000.

[6]. D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-Based Object Tracking,” *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, volume 25, No. 5, pp. 564 – 577, 2003.

[7]. N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, California, volume 1, pp. 886 – 893, 2005.

[8] F. F. Li, "CS231n: Convolutional Neural Networks for Visual Recognition," http://cs231n.github.io/convolutional-networks/, 2017.

[9]. H. Grabner, M. Grabner, and H. Bischof, "Real-Time Tracking via On-line Boosting," *Proceedings of British Machine Vision Conference*, Edinburgh, UK, volume 1, pp. 47 – 56, 2006.

[10]. M. Isard and A. Black, "CONDENSATION – Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, volume 29, No. 5, pp. 5 – 28, 1998.

[11]. Y. Q. Jia, S. Evan, D. Jeff, K. Sergey, L. Jonathan, G. Ross, G. Sergio, and D. Trevor, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv: 1408.5093*, https://arxiv.org/pdf/1408.5093.pdf, 2014.

[12]. Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 6, No. 1, pp. 1409 – 1422, 2010.

[13]. A. Krizhevsky, I. Sutskever, and G. E Hinton, “Imagenet Classification with Deep Convolutional Neural Networks,” *Advances in Neural Information Processing Systems*, pp. 1097 – 1105, 2002.

[14]. N. S. Peng, J. Yang, and Z. Liu, “Mean Shift Blob Tracking with Kernel Histogram Filtering and Hypothesis Testing,” *Pattern Recognition Letters*, volume 26, No. 5, pp. 605 – 614, 2004.

[15]. K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv: 1409.1556*, https://arxiv.org/pdf/1409.1556.pdf , 2014.

[16]. Standford Vision Lab, “ImageNet Large Scale Vision Recognition Challenge,” http://www.image-net.org/challenges/LSVRC/, 2017.

[17]. C. Stauffer and W. E. L. Grimson, “Adaptive Background Mixture Models for Real-Time Tracking,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Fort Collins, CO, volume 2, pp. 246 – 252, 1999.

[18]. T. H. Vu, A. Osokin, and I. Laptev, “Context-Aware CNNs for Person Head Detection,” *Proceedings of International Conference on Computer Vision,* Santiago, Chile, pp. 2893 – 2901, 2015.

[19]. S. Tang, M. Andriluka, A. Milan, K. Schindle, S. Roth, and B. Schiele, “Learning People Detectors for Tracking in Crowded Scenes,” *Proceedings of International Conference on Computer Vision*, Sydney, Australia, pp. 1049 - 1056, 2013.

[20]. Wikipedia, “Artificial Neural Network,” https://en.wikipedia.org/wiki/Artificial_neural_network, 2017.

[21]. Wikipedia, “Backpropagation,” https://en.wikipedia.org/wiki/Backpropagation, 2017.

[22]. Wikipedia, “Convolutional Neural Network,” https://en.wikipedia.org/wiki/Convolutional_neural_network, 2017.

[23]. Wikipedia, “Massive Open Online Course,” https://en.wikipedia.org/wiki/Massive_open_online_course, 2017.

[24]. Wikipedia, “Particle Filter,” https://en.wikipedia.org/wiki/Particle_filter, 2017.

[25]. Y. Wu, J. Lim, and M. H. Yang, “Online Object Tracking: A Benchmark,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Oregon, Portland, pp. 2411 – 2418, 2013.

[26]. B. Wu and R. Nevatia, “Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet Based Part Detectors,” *International Journal of Computer Vision*, volume 75, No. 2, pp. 247 – 266.