

國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

可運用於合作式視訊監控之攝影機分工協調技術

**A Study on Coordination of PTZ Cameras for
Cooperative Video Surveillance**

研 究 生：范博凱

指導教授：王聖智 博士

中 華 民 國 九 十 七 年 七 月

可運用於合作式視訊監控之攝影機分工協調技術

**A Study on Coordination of PTZ Cameras for Cooperative
Video Surveillance**

研 究 生：范博凱

Student : Po-Kai Fan

指 導 教 授：王聖智博士

Advisor : Dr. Sheng-Jyh Wang

國 立 交 通 大 學

電子工程學系 電子研究所碩士班



A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master

in

Electronics Engineering

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

可運用於合作式視訊監控之攝影機分工協調技術


研究生：范博凱

指導教授：王聖智 博士

國立交通大學

電子工程學系 電子研究所碩士班

摘要



在本論文中，我們提出一套應用於多台主動式攝影機之分工協調系統，對於空間中大約已知臉部之位置與朝向的人群，進行攝影機的分工與協調。每一台攝影機將會負責拍攝一小部分人群的臉部，並且設法調整攝影機的旋轉角度以及放大倍率，使人臉可以清晰地畫面中呈現。在此，我們對於人臉在畫面中清晰與否的評斷標準為：人臉是否正面朝向負責拍攝的攝影機，以及人臉在影像中的解析度。透過本系統，我們可以安排各個主動式攝影機的旋轉角度與放大倍率，盡可能地拍攝場景中所有人的臉部，以獲得理想的人臉拍攝角度與解析度，便於清楚地辨識每個人。

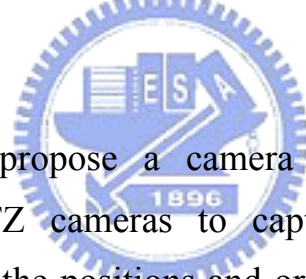
A Study on Coordination of PTZ Cameras for Cooperative Video Surveillance

Student: Po-Kai Fan

Advisor: Dr. Sheng-Jyh Wang

Department of Electronics Engineering, Institute of Electronics
National Chiao Tung University

Abstract



In this paper, we propose a camera coordination system that coordinates multiple PTZ cameras to capture the face pictures of monitored targets. Given the positions and orientations of people's faces in the 3-D space, this system dynamically controls the panning, tilting, and zooming of all PTZ cameras, trying to acquire better shots of targets' faces. The adopted criteria include people's facing directions with respect to the cameras and the resolutions of the facial images. Unlike other approaches, we do not limit our PTZ cameras to capture only one target at one time. Instead, the proposed system coordinates all PTZ cameras to capture as many high resolution frontal faces as possible. With this system, the faces in the scene can be better captured and the identity of each monitored target can be well discerned.

誌謝

在此要特別感謝我的指導教授 王聖智老師，在他的細心指導下，學習到了很多研究的方法與態度，除了課業上的知識外，也學習到許多待人接物與處理事物的技巧，使我在這兩年中有著長足的成長。感謝實驗室的全體夥伴，在我需要幫助時，給予我真誠的協助。同時，我也要感謝我的家人，因為有他們的支持，讓我可以安心地在學業上努力。最後，我要感謝晴駿，因為你的支持、鼓勵與陪伴，使我擁有今日的結果。

Content

Chapter 1.	Introduction.....	1
Chapter 2.	Backgrounds	3
2.1.	Surveillance Systems with PTZ Cameras	3
2.2.	Clustering Algorithms.....	8
2.2.1.	K-means Clustering	8
2.2.2.	Fuzzy C-Means Clustering	9
2.2.3.	Hierarchical Clustering	11
2.3.	Optimization	12
2.3.1.	Particle Swarm Optimization.....	12
2.3.1.1.	Classical PSO	12
2.3.1.2.	Discrete Binary PSO	14
2.3.2.	Differential Evolution	17
2.4.	Virtual Video Tool.....	20
Chapter 3.	Camera Coordination	23
3.1.	Problem Formulation	23
3.2.	Significance weight.....	28
3.2.1.	Weighting Update	30
3.2.1.1.	Rise State	30
3.2.1.2.	Hold State.....	30
3.2.1.3.	Decline State	31
3.2.2.	Upper Bound of Unclear Period	32
3.2.3.	Combining Weight with Evaluation.....	34
3.3.	Modified Discrete Binary PSO	34
3.3.1.	Particle Generation.....	35
3.3.1.1.	Feature Space	35
3.3.1.2.	Generation by Clustering	37
3.3.1.3.	Generation by the Latest Assignment	37
3.3.1.4.	Generation by Random Selection	38
3.3.2.	Optimization with Constraints	38
3.4.	Camera Assignment	43
3.4.1.	Who to Look?	43
3.4.2.	Where to Look?.....	43
3.5.	Camera Control.....	46
3.6.	Assignment Hold	48
3.7.	Overall Coordination System.....	49
Chapter 4.	Simulations	51

Chapter 5. Conclusions.....73
References.....74



List of Figures

Figure 2-1 The results of Micheloni’s proposed system [2]	4
Figure 2-2 Overview of Kim’s system [3]	4
Figure 2-4 The process of face focus of Hampapur’s system [4]	6
Figure 2-5 A face zoom sequence [4]	6
Figure 2-7 A virtual train station designed by Qureshi (a) over views (b) close-up views by PTZ cameras [6]	7
Figure 2-8 A clustering example (a) data points (b) clustering result.....	8
Figure 2-9 An example of the process of K-means clustering (a) centroids initialization (b)-(d) iteration (centroids recalculation).....	9
Figure 2-10 The comparison of (a) k-means clustering (b) fuzzy c-means cluster algorithm	10
Figure 2-11 An example of merging process	11
Figure 2-12 Sigmoid function.....	16
Figure 2-13 The block diagram of differential evolution algorithm [16]	17
Figure 2-14 A mutation example of a two dimensional minimization problem [15]..	18
Figure 2-15 An example of crossover [15]	19
Figure 2-16 OVVV system [19]	20
Figure 2-17 The synthetic frames with (right) and without (left) noises [19]	21
Figure 2-18 The synthetic frames of omniscams: panoramic (left) parabolic catadioptric (right) omni-directional cameras [19]	21
Figure 2-19 A ground truth examples: bounding box (left) and label map (right) [19]	22
Figure 3-1 Flow chart of our proposed camera coordination system	23
Figure 3-3 The illustration of W_{ij}	24
Figure 3-4 Normalized function of the bias angle	26
Figure 3-5 Normalized function of the face width.....	27
Figure 3-6 (a) lower (b) higher overall observation level.....	30
Figure 3-7 Variation of the significance weight over time.....	31
Figure 3-8 Illustration of the weighting variation for the case of time limitation	33
Figure 3-9 Illustration of the weighting variation for the case of weighting limitation	33
Figure 3-10 An example of coordinate normalization	36
Figure 3-11 The illustration of constraint repair of each iteration of binary PSO	40
Figure 3-12 Repair process for the no assignment case.....	42
Figure 3-13 Block diagram of the optimization process.....	42
Figure 3-14 A camera takes charge of three people.....	43

Figure 3-15 Representation of FOV by vectors	44
Figure 3-16 Angle bisector.....	45
Figure 3-17 Internal point of division of a line segment	45
Figure 3-19 Illustration of pan and tilt angles.....	47
Figure 3-20 Block diagram of camera adjustment.....	49
Figure 3-21 Overall block diagram of the proposed coordination system.....	50
Figure 4-1 The installation of PTZ cameras	51
Figure 4-2 Experimental results of the test sequence SEQ-1	55
Figure 4-3 All people's score curves of Sequence SEQ-1	56
Figure 4-4 All people's significance weights of Sequence SEQ-1	56
Figure 4-5 Person p8's score curve of Sequence SEQ-1 without applying significance weight.....	57
Figure 4-6 Illustration of the mechanism of unclear limitation	57
Figure 4-7 The corresponding score curve of Figure 4-6	58
Figure 4-8 Experimental results of the test sequence SEQ-2	60
Figure 4-9 All people's score curves of the sequence SEQ-2.....	61
Figure 4-10 Experimental results of the test sequence SEQ-3	63
Figure 4-11 All people's score curves of sequence SEQ-3	64
Figure 4-12 Experimental results of the test sequence SEQ-4	66
Figure 4-13 All people's score curves of Sequence SEQ-4	66
Figure 4-14 Experimental results of the test sequence SEQ-5	69
Figure 4-15 All people's score curves of Sequence SEQ-5	69

List of Tables

Table 4-1 Statistical results of all experimental sequences.....	70
Table 4-2 Comparison between the modified DBPSO and the original DBPSO	70
Table 4-3 Comparison between the modified DBPSO and the original DBPSO	71
Table 4-4 Comparison of iteration numbers (30 particles)	71
Table 4-5 Comparison of iteration number (20 particles).....	72



Chapter 1.

INTRODUCTION

A tremendous number of cameras have been surrounding us in our daily lives in recent years. We can see them in various places, like airports, train stations, subways, and convenience stores. Due to the increasing demands in security and safety, more and more researchers pay attention to the issues of video surveillance. Recently, the issues about multi-camera surveillance systems have attracted the attention of researchers. In a multi-camera system, more than one camera is installed within a certain area. The cameras located at different locations can help us in monitoring the targets from different observation angles. If PTZ (Pan-Tilt-Zoom) cameras, instead of static cameras, are used, the functionalities of video surveillance system can be even more versatile.

Before, a multi-camera system was composed of static cameras, whose pan angle, tilt angle, and field of view were fixed. Compared with a single camera, this kind of multi-camera system extends the monitoring region and angles of view. However, once if the monitored targets move away from the monitored region, we can no longer get clear images of the targets. Hence, recently, people start to use active cameras in their multi-camera systems.

The most popular type of active camera is the PTZ (Pan-Tilt-Zoom) camera. As implied by its name, a PTZ camera can actively adjust its pan angle, tilt angle, and zoom level. Many recently proposed multi-camera systems are composed of both static cameras and PTZ cameras. With the help of PTZ cameras, we can not only monitor a region with various angles of view, but can also more clearly capture the features of the monitored targets via the adjustment of the zoom level.

Up to now, many multi-camera systems equipped with PTZ cameras focus on the capturing of human faces. They assign PTZ cameras to zoom in on the target to get a close-up of the target's face. This can help in identifying the monitored target. However, existing systems usually assign each camera to focus on a single face at one time. If the number of targets are many more than the number of PTZ cameras, then these multi-camera systems may fail in taking good observations of all targets.

In this thesis, we develop a surveillance system that tries to simultaneously

observe as many high-resolution faces as possible. In Figure 1-1, we illustrate the task of the proposed system. In this example, there are 9 people in total. The triangles denotes PTZ cameras, the circles indicate people's locations, and the arrows represent the orientation of people's face. The proposed system will automatically assign these four PTZ cameras to take care of different groups of people so that the multi-camera system can capture as many high-resolution facial images as possible at every moment.

We first formulate the problem according to some criteria and we define the evaluation function. We also try to optimize the evaluation function in an efficient way. For the sake of cost and convenience, we simulate the proposed system by using virtual videos generated from the ObjectVideo Virtual Video (OVVV) software tool.

In this thesis, we will first discuss some related works and mathematical techniques in Chapter 2. In Chapter 3, we will present the proposed coordination system which use multiple active cameras to get as many clear people's face images as possible. Some experimental results are shown in Chapter 4. Finally, we give our conclusion in Chapter 5.

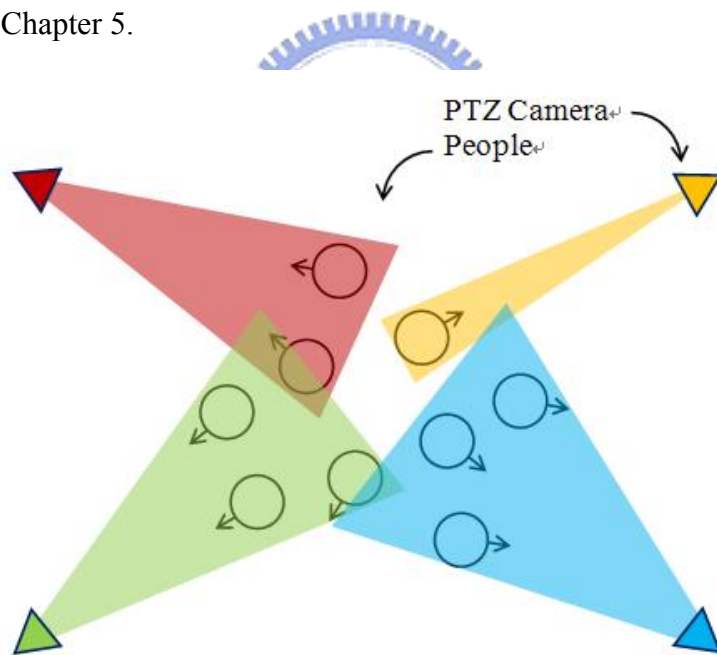


Figure 1-1 An example of camera coordination

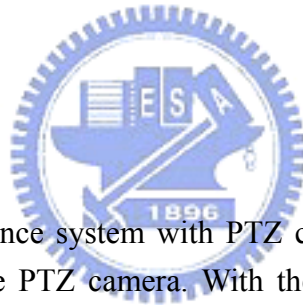
Chapter 2.

BACKGROUNDS

Although several multi-camera surveillance systems have already been proposed, we have not found any multi-camera system that offers similar functionalities as ours. Hence, we only mention a few articles that have discussed some issues similar to ours. In the proposed method, we use some mathematical techniques, such as clustering and optimization. Hence, we will also briefly introduce these mathematical techniques. In the end of this chapter, we will introduce the virtual video tool which we have made use of.

2.1. SURVEILLANCE SYSTEMS WITH PTZ

CAMERAS



In general, in a surveillance system with PTZ cameras, there are several static cameras and no less than one PTZ camera. With the PTZ cameras, we are able to carry out more intelligent surveillance, such as active monitoring. For example, if we are interested in people's faces, we may control the PTZ cameras to focus on someone's face and identify who the person is.

Most of these systems mainly focus on the capture of clear human images. For example, in [1] and [2], Micheloni proposed a system that contains a few static cameras and PTZ cameras. The resolution of the PTZ camera is higher than that of static camera. When a person appears, they estimate the 3-D location of the target and automatically control the pan angle and tilt angle of the PTZ cameras to capture the target's high-resolution images. In their approach, each PTZ camera focuses on the tracking of a single target. Their results are shown in Figure 2-1.

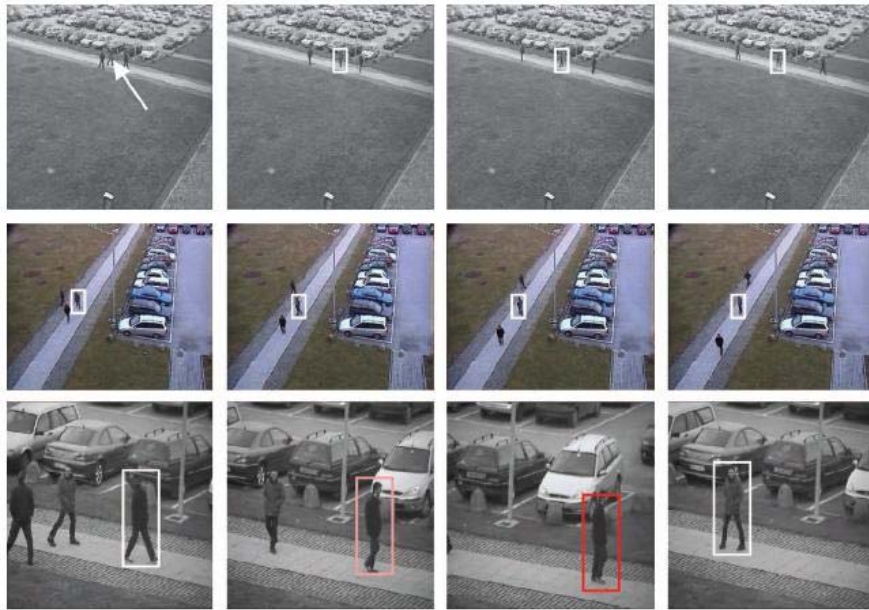


Figure 2-1 The results of Micheloni's proposed system [2]

On the other hand, [3] uses the cooperation of multiple PTZ cameras to reduce the spatial limit and to locate the targets' positions. This system is composed of two major parts: camera agents and a support module. Camera agents carry out image processing and camera control, while the support module coordinates all camera agents. The overview of this system is shown in Figure 2-2.

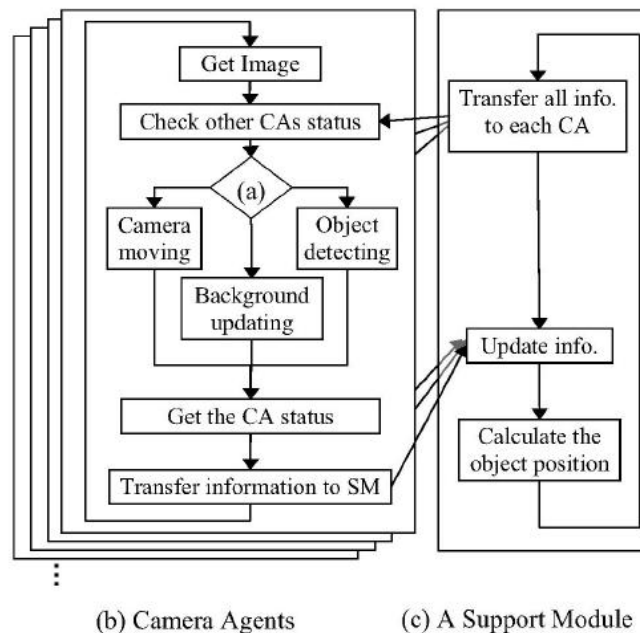


Figure 2-2 Overview of Kim's system [3]

In [4], the proposed surveillance system also contains multiple static cameras and PTZ cameras. The static cameras are used to estimate the 3D positions of the detected targets. Face detection is also used to determine whether a human face exists. Once if

a face exists, then they control a PTZ camera to capture a close-up of that face.

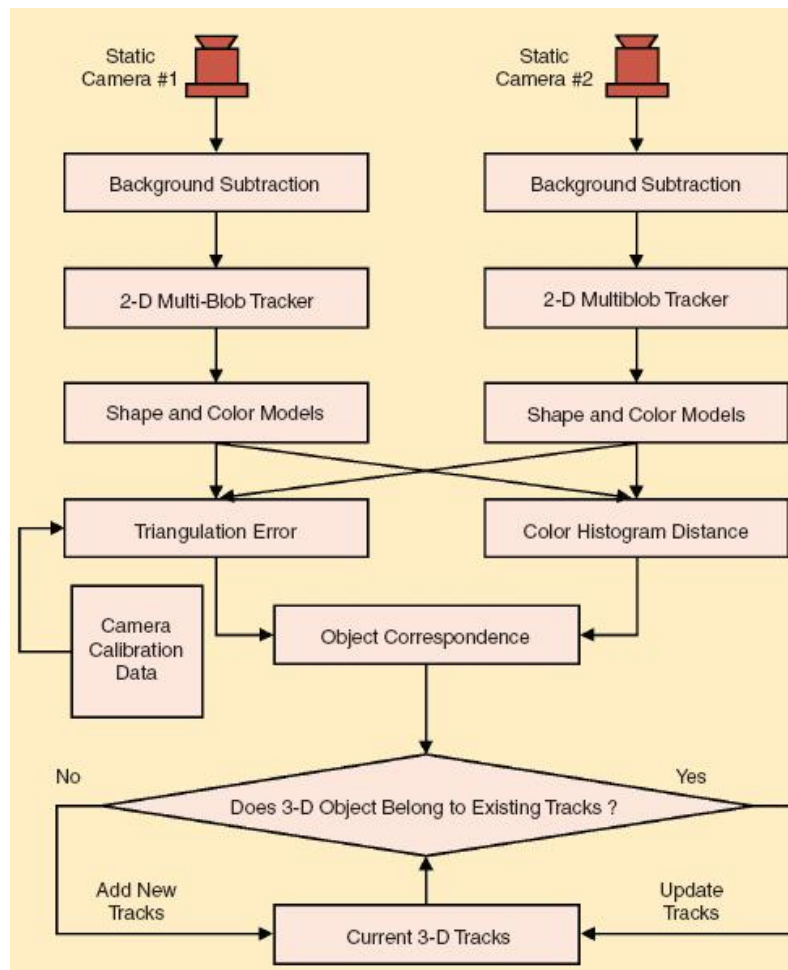


Figure 2-3 Block diagram of Hampapur's 3D tracker [4]

Figure 2-3 shows the 3D tracking process of the static cameras and Figure 2-4 shows how the system coordinates the static and PTZ cameras to accomplish face capturing. In Figure 2-5, we show the zoomed images captured by the PTZ camera.

In [5], the authors use pairs of static cameras to estimate the depth information. The face position of the target is estimated by combining the depth information with the face detection results. Similarly, once if a face is detected, a PTZ camera is controlled to capture a clearer facial picture of the target. Some experimental results are shown in Figure 2-6.

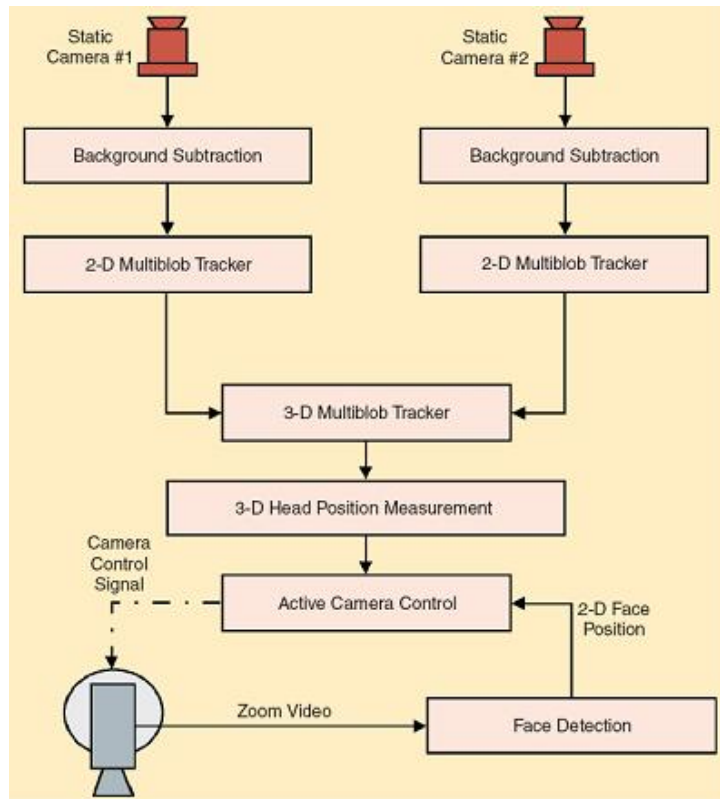


Figure 2-4 The process of face focus of Hampapur's system [4]

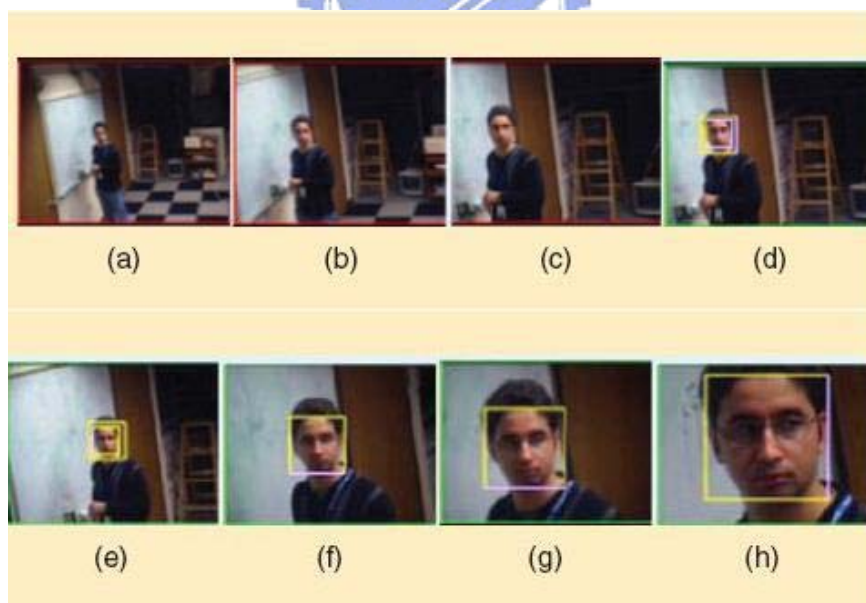
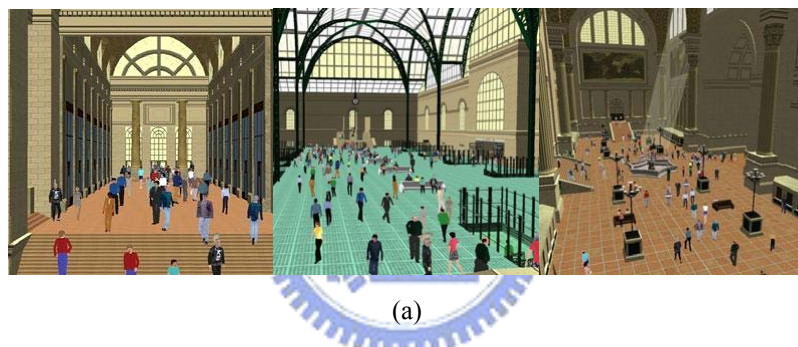


Figure 2-5 A face zoom sequence [4]

In [6] and [7], the authors use pairs of static cameras to estimate the depth information. The face position of the target is estimated by combining the depth information with the face detection results. Similarly, once if a face is detected, a PTZ camera is controlled to capture a clearer facial picture of the target. Some examples are shown in Figure 2-7.



Figure 2-6 The experiment results of [5]



(b)

Figure 2-7 A virtual train station designed by Qureshi (a) over views (b) close-up views by PTZ cameras [6]

2.2. CLUSTERING ALGORITHMS

Clustering can be thought as a kind of classification method. When there are several data which have some kinds of similar properties clustering methods can be used to explore the data and to group similar ones together under certain criteria. A clustering example is illustrated in Figure 2-8. In the literature, clustering has already been well developed and many different algorithms have been developed. We will discuss some commonly used algorithms in this section.

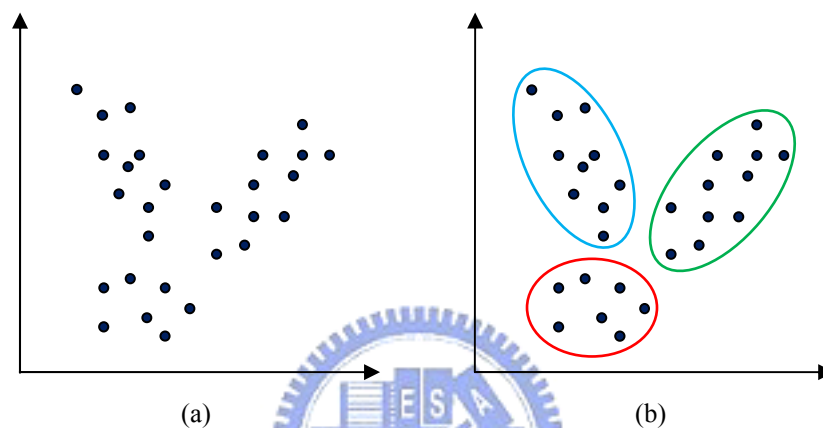


Figure 2-8 A clustering example (a) data points (b) clustering result

2.2.1. K-MEANS CLUSTERING

K-means is a simple and fast clustering algorithm. It was originally proposed in [8]. The main idea of K-means clustering is to iteratively minimize the variance of each cluster. At the beginning, k centroids are initialized and they represent the centers of clusters. Then, each datum is classified to a cluster according to the distances between the data point and the centroids. The data point is assigned to the cluster which has the shortest distance between its centroid and this data point. Finally, the mean of each cluster is calculated and is used to update the new centroid. The process is repeated until the positions of the centroids converge. The followings are the detailed steps of the k-means algorithm:

1. In the data space, choose k points as the initial centroids of clusters.
2. Assign each data point to the cluster which has the shortest distance between its centroid and that data point.
3. Recalculate the k centroids by averaging the data points.

4. Repeat Step 2 and Step 3 until these centroids are almost fixed. Then we get the final clustering result.

The advantages of the k-means method are its simplicity and low computational cost. It is very easy to implement the K-means algorithm. However, this method still has several disadvantages. For example, it is very sensitive to the choice of the initial centroids. It only minimizes the intra-cluster variance, but not the global variance. In other words, this method does not guarantee global minimization but only a local minimization. The global minimization depends on the appropriate selection of the initial centroids. There is an example of k-means clustering shown in Figure 2-9.

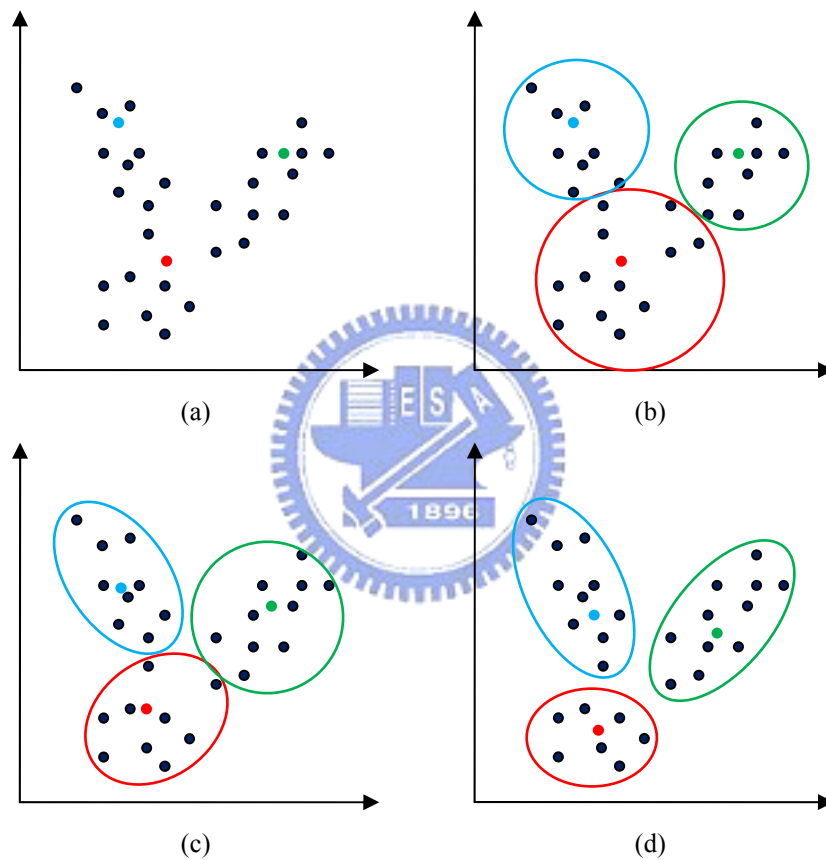


Figure 2-9 An example of the process of K-means clustering (a) centroids initialization (b)-(d) iteration (centroids recalculation)

2.2.2. FUZZY C-MEANS CLUSTERING

Fuzzy c-means clustering technique [9] is similar to k-means but it allows data to belong to more than one cluster. This is why it is called fuzzy. We illustrate the difference between k-means clustering and fuzzy c-means clustering in Figure 2-10. Here we consider 1-D data points and two clusters (red and green). For k-means

clustering, each data point only belongs to one cluster, as shown in Figure 2-10 (a). With fuzzy c-means clustering, however, each data point can belong to more than one cluster with different degrees of cluster membership, as shown in Figure 2-10 (b).

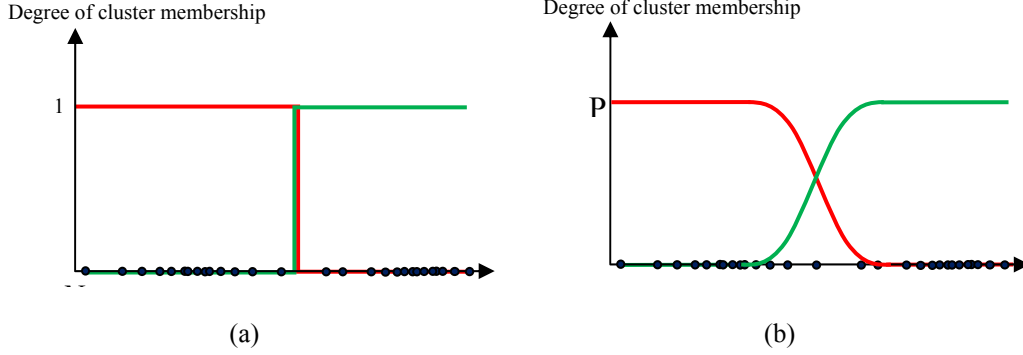


Figure 2-10 The comparison of (a) k-means clustering (b) fuzzy c-means cluster algorithm

The objective of fuzzy c-means clustering and k-means clustering are the same. That is, we find the clusters that minimize their variances. Similar to k-means, the fuzzy c-means clustering needs to define an initial condition and then iteratively update the cluster centers. However, the difference is that the fuzzy c-means clustering directly initializes the degrees of the data points in each cluster and update them in each iteration. The detailed fuzzy c-means algorithm is described as follows:

1. Initialize u_{ij} , the degree of x_i in the cluster j , where x_i is a data point.

2. Calculate each center c_j by means of the formula
$$c_j = \frac{\sum_{i=1}^N u_{ij}^m \cdot x_i}{\sum_{i=1}^N u_{ij}^m} .$$

3. Use this formula
$$u_{ij} = \left[\sum_{k=1}^C \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}} \right]^{-1}$$
 to update each u_{ij} .

4. Repeat Step 2 and Step 3 until $\max_{ij} \left\{ |u_{ij}^{(k+1)} - u_{ij}^{(k)}| \right\} < \varepsilon$

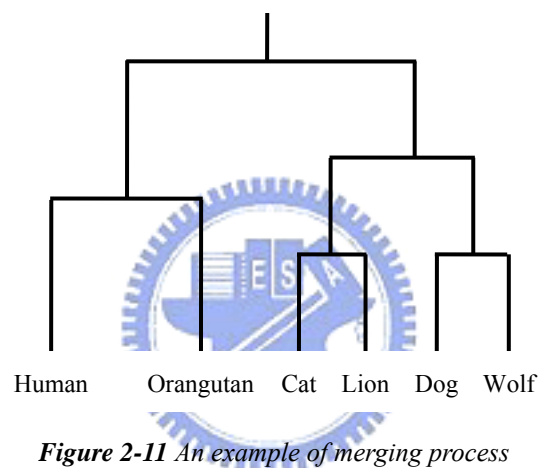
where m is a real number greater than 1, k is the iteration number, and ε is a real number between 0 and 1.

Although fuzzy c-means clustering requires more computations than k-means clustering, it usually can find better solution. However, it still possesses some problems of k-means clustering. For example, it can only find a local minimum. The

clustering result is also sensitive to the initialization of the degrees.

2.2.3. HIERARCHICAL CLUSTERING

Unlike k-means clustering and fuzzy c-means clustering, the hierarchical clustering algorithm [10] does not need to set the number of clusters. Compared with k-means (or fuzzy c-means) clustering, this method uses the concept of mergence, instead of the concept of partition. It considers each data point a cluster initially and then merges data points gradually to reach a proper set of clusters. Figure 2-11 illustrates the simple merging process. Here we take each creature as a data point and we gradually clustering these six creatures into clusters.



The followings are the detailed steps of the hierarchical clustering algorithm:

1. Consider each data point a cluster. Define the distances between each pair of clusters.
2. Find the pair of clusters which has the closest distance.
3. Merge the pair of clusters with the closest distance into a new cluster. The number of clusters reduces one.
4. Repeat Step 2 and Step 3 until the number of clusters reduces to a value we desire.

Generally, the hierarchical clustering method better suits the characteristics of data. It does not need assign the number of clusters and can always reach the same result. However, this method has a major problem: its high computational cost. Its complexity is at least $O(n^2)$. Besides, because of the mergence, this method cannot undo what have been done previously.

2.3. OPTIMIZATION

We usually encounter the optimization problem in our daily lives. For example, when we prepare a trip, we often ask how we can arrange our transportation to reduce the traveling time to the destination. This is a simple example of the optimization problem. Typically, an optimization problem can be formulated in mathematics. In general, we describe these problems by using an objective function with or without constraints. The objective function and the constraints are composed of several unknown parameters. Then, we try to find the selection of parameters that gets the minimum or maximum of the objective function. In other words, we want to find the values of parameters which make the value of the objective function minimal or maximal. Depending on the problems we want to solve, the objective function can be defined in different ways. The objective function may be linear or nonlinear and can be either continuous or discrete.

So far, many optimization algorithms have already been proposed, like gradient decent, linear programming, Lagrange multiplier, and Karush-Kuhn-Tucker (KKT) condition [11], etc. However, these derivative-based and linear constrained algorithms do not suit the problems that are nonlinear or cannot be differentiated. Hence, people devise some other algorithms for these kinds of optimization problems. Here, we briefly introduce two effective algorithms – Particle Swarm Optimization (PSO) and Differential Evolution (DE).

2.3.1. PARTICLE SWARM OPTIMIZATION

2.3.1.1. CLASSICAL PSO

Kennedy and Eberhart devise the particle swarm optimization algorithm, which is inspired by a sociological model [12][13]. Each particle represents a trial solution of the problem that we want to solve. In this algorithm, as implied by the name “particle swarm”, a large number of particles are generated. The PSO algorithm uses these particles to carry out multi-agent parallel search. Each particle has its own memory. They can “remember” their previous best positions that make the objective function minimal or maximal. In addition, the particles communicate with each other to get the best global position that achieves the global extreme in the past. One

particle moves to its next position according to its previous best position and the global best position in the past. The particles repeat the same steps and they gradually converge to the final position.

In mathematics, the objective function can be expressed as

$$f(\vec{x}) = f(x_1, x_2, \dots, x_n) \quad \text{Eq. 2-1}$$

where \vec{x} is the variable vector in the n-dimensional space. Here we assume that the problem we want to solve is a minimization problem and we would like to find a \vec{x}_* that minimizes Eq. 2-1.

First, a group of particles are initialized randomly. That is, we create a certain number of particles and allocate their initial positions and velocities randomly. The velocity defines where the corresponding particle should move to next time. The position and velocity of the i -th particle at Time t are denoted as \vec{x}_i^t and \vec{v}_i^t , respectively. The number of particles is initialized by the user. For each particle, we calculate the value of the objective function at its current position. Every particle keeps track of its best previous position that gets the extreme value of the objective function. We denote the best previous position as \vec{P}_i . In the meantime, we also record the globally best position, which is denoted as \vec{P}_g . Finally, the next velocity and position of each particle can be calculated by Eq. 2-2 and Eq. 2-3, respectively.

$$\vec{v}_i^{t+1} = \omega \cdot \vec{v}_i^t + c_1 \varphi_1 \cdot (\vec{P}_i - \vec{x}_i^t) + c_2 \varphi_2 \cdot (\vec{P}_g - \vec{x}_i^t) \quad \text{Eq. 2-2}$$

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1} \quad \text{Eq. 2-3}$$

where ω is the inertia factor, c_1 and c_2 are scalars, and φ_1 and φ_2 are random numbers generated from the uniform distribution over the interval [0, 1]. The aforementioned process is repeated until the stop criterion is reached. The followings are the pseudo code of the PSO algorithm.

PSEUDO CODE

Initialization: Initialize the positions (\vec{x}_i^0) and velocities (\vec{v}_i^0) of N particles randomly. Also initialize \vec{P}_i and \vec{P}_g .

Begin

While the stop criterion is not reached

For $i = 1$ to N

Evaluate the value of objective function for each particle: $f(\vec{x}_i^t)$

If $f(\vec{x}_i^t) < f(\vec{P}_i)$ **do**

$$\vec{P}_i = \vec{x}_i^t$$

End do

If $f(\vec{P}_i) < f(\vec{P}_g)$ **do**

$$\vec{P}_g = \vec{P}_i$$

End do

End for

For $i = 1$ to N

$$\vec{v}_i^{t+1} = \omega \cdot \vec{v}_i^t + c_1 \phi_1 \cdot (\vec{P}_i - \vec{x}_i^t) + c_2 \phi_2 \cdot (\vec{P}_g - \vec{x}_i^t)$$

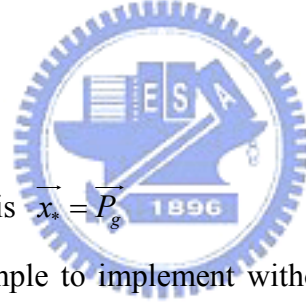
$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1}$$

End for

End while

End

Output: the optimal position is $\vec{x}_* = \vec{P}_g$



The PSO algorithm is simple to implement without heavy computation load. In addition, it can find the global optimum and does not depend on the form of objective function (or fitness function). It is an effective optimization algorithm. We can utilize PSO to deal with complex, high-dimensional and nonlinear optimization problems.

2.3.1.2. DISCRETE BINARY PSO

The PSO algorithm mentioned above is originally operated in continuous domain. However, many optimization problems are actually in a discrete domain. Hence, Kennedy and Eberhart proposed the discrete binary version of PSO [14] for discrete optimization problems. The concept of discrete binary PSO algorithm is the same as the original PSO algorithm, except a few modifications over the original PSO algorithm. In the discrete binary space, the variables are only the integers 0 or 1. Hence we re-define the objective function (or fitness function) to Eq. 2-4:

$$f(\mathbf{x}) = f(x_1, x_2, \dots, x_n) \quad \text{Eq. 2-4}$$

where \mathbf{x} denotes an n -bit string, and x_k represents the k -th bit which is 0 or 1 in the bit string. Similarly, we want to find a \mathbf{x}_* to minimize Eq. 2-4. The position of the i -th particle and its d -th bit are denoted by \mathbf{x}_i and x_{id} . The definition of velocity is different from the original PSO. In continuous PSO, the velocity is defined for each particle. Here each dimension has its own velocity which is denoted by v_{id} . That is, each bit has its own velocity. Moreover, the velocity of the original PSO indicates where the corresponding particle moves to. However, when we discuss the velocity of binary PSO, we focus on each single bit and the meaning of velocities is changed. The meaning of velocity now represents the tendency of the corresponding bit to become 1. The larger the velocity is, the more likely the corresponding bit becomes 1. Besides, with the modification of the definition of velocity, the best previous position and the best previous global position are also treated in a bitwise manner. p_{id} denotes the best previous d -th bit of the i -th particles and p_{gd} denotes the best previous global d -th bit. Of course p_{id} and p_{gd} are either 0 or 1. With the above modifications, we rewrite the velocity updating formula to be

$$v_{id}^{t+1} = \omega \cdot v_{id}^t + \varphi_1 \cdot (p_{id} - x_{id}^t) + \varphi_2 \cdot (p_{gd} - x_{id}^t) \quad \text{Eq. 2-5}$$

where t represents the time instant, ω is the inertia factor, and φ_1 and φ_2 are random numbers generated from the uniform distribution over the interval $[0,1]$. The authors used probability to describe the tendency of bit change so the velocities have to be converted to the interval $[0, 1]$. They introduce the sigmoid function and modified the position-updating formula to be defined as below:

$$\begin{aligned} & \text{if } (\text{rand}(0,1) < S(v_{id}^{t+1})) \text{ then } x_{id}^{t+1} = 1 \\ & \text{else } x_{id}^{t+1} = 0 \end{aligned} \quad \text{Eq. 2-6}$$

where $\text{rand}(0,1)$ is a random number selected from the uniform distribution over $[0, 1]$, and S is a sigmoid function. Eq. 2-7 is the formula of the sigmoid function.

$$S(v) = \frac{1}{1 + e^{-v}} \quad \text{Eq. 2-7}$$

The logistic curve of the sigmoid function is shown in Figure 2-12. This function transfers the value of v_{id} into the interval $[0, 1]$.

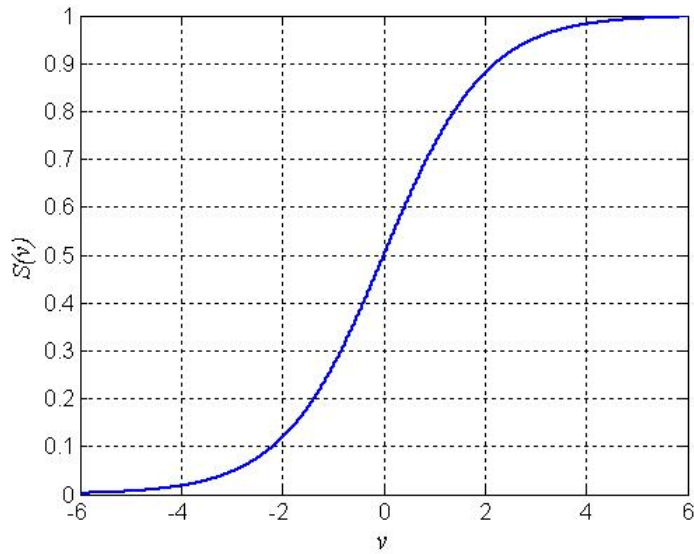


Figure 2-12 Sigmoid function

Basically, the binary PSO is very similar to the original PSO. Only the definition of velocity and the position-updating function are modified. The followings are the pseudo codes of the DBPSO algorithm.

PSEUDO CODE

Initialization: Initialize the positions (\mathbf{x}_i^0) and velocities (v_{id}^0) of N particles randomly. Also initialize \mathbf{p}_i and \mathbf{p}_g

Begin

While the stop criterion is not reached

For $i = 1$ to N

Evaluate the value of objective function of each particle: $f(\mathbf{x}_i^t)$

If $f(\mathbf{x}_i^t) < f(\mathbf{p}_i)$ **do**

$\mathbf{p}_i = \mathbf{x}_i^t$

End do

If $f(\mathbf{p}_i) < f(\mathbf{p}_g)$ **do**

$\mathbf{p}_g = \mathbf{p}_i$

End do

End for

For $i = 1$ to N

For $d = 1$ to n

$$v_{id}^{t+1} = \omega \cdot v_{id}^t + \varphi_1 \cdot (p_{id} - x_{id}^t) + \varphi_2 \cdot (p_{gd} - x_{id}^t)$$

$$\text{if } (\text{rand}(0,1) < S(v_{id}^{t+1})) \text{ then } x_{id}^{t+1} = 1$$

$$\text{else } x_{id}^{t+1} = 0$$

End for

End for

End while

End

Output: the optimal bit string is $\mathbf{x}_* = \mathbf{p}_g$

The binary PSO inherits the main concept from the original PSO. The particle swarm still has “memory” in the binary PSO and the particles move toward the region that so far provides the best solution. The DBPSO is also effective for solving the discrete binary optimization problems.

2.3.2. DIFFERENTIAL EVOLUTION

Differential evolution is a global optimization algorithm proposed by Storn and Price [15]. Like PSO, it is one kind of parallel searching techniques. It generates several numbers of trial parameter vectors at the same time and tries to find the optimum. DE inherits the ideas from genetic algorithm but it alters the classical crossover and mutation operations. The authors present a differential operator to generating new “offspring” for the searching of the optimum. The block diagram of the DE algorithm is shown in Figure 2-13.

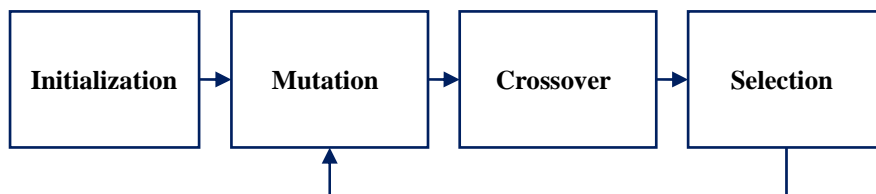


Figure 2-13 The block diagram of differential evolution algorithm [16]

In the initialization stage, a population is initialized. In other words, a number of D-dimensional parameter vectors are initialized. The i -th parameter vector in the g -th generation is denoted as x_i^g , and the population size is denoted as N . After the initialization, DE creates several candidates that may become parts of the population of the next generation. These candidates are generated by means of “mutation” and “crossover”. In the mutation stage, we use Eq. 2-8 to generate a “mutant” parameter vector for each target vector, x_i^g :

$$v_i^{g+1} = x_{r_1}^g + C \cdot (x_{r_2}^g - x_{r_3}^g) \quad \text{Eq. 2-8}$$

where C is a constant in $[0, 2]$ and r_1, r_2, r_3 are the random integers from 1 to N . In Figure 2-14, we show an example of mutation.

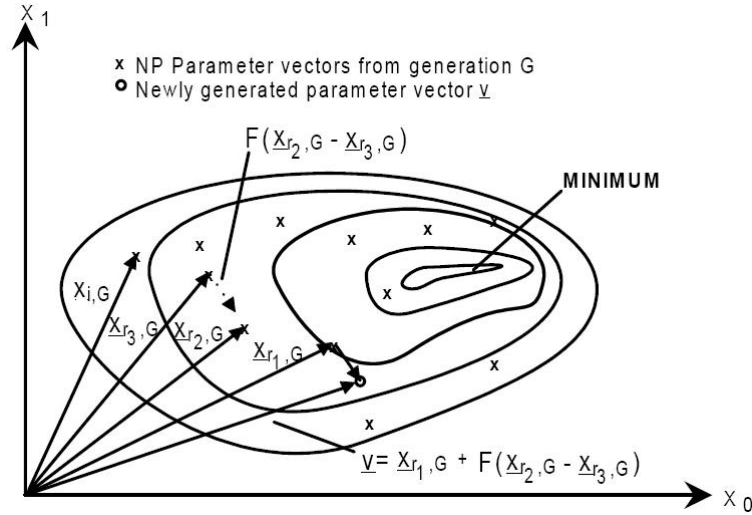


Figure 2-14 A mutation example of a two dimensional minimization problem [15]

Next, the mutant parameter vectors are carried out crossover to increase the variance. A trial parameter vector, u_i^g , is created for each target vector by means of crossover. It is generated based on the following equation:

$$u_{i,j}^{g+1} = \begin{cases} v_{i,j}^{g+1}, & \text{if } rand(0,1) \leq CR \text{ or } j = rand_{int}(1,D) \\ x_{i,j}^g, & \text{otherwise} \end{cases} \quad \text{Eq. 2-9}$$

where j is an integer from 1 to D that represents the value of the j -th dimension; $rand(0, 1)$ is a random real number generated from the uniform distribution over $[0, 1]$; $rand_{int}(1, D)$ is a random integer number chosen from $\{1, 2, \dots, D\}$; and CR represents the pre-defined crossover constant within the range $[0, 1]$. In Figure 2-15, we illustrate the crossover process.

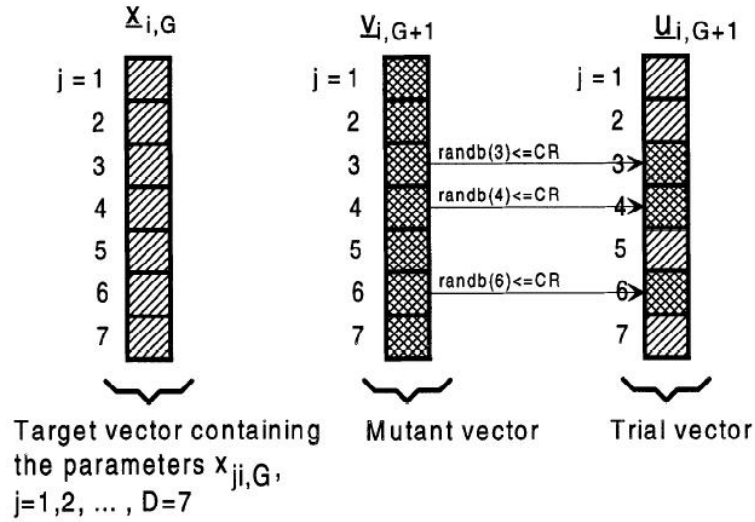


Figure 2-15 An example of crossover [15]

Finally, the selection process is performed to decide the next-generation population. Here we assume that we want to find the minimum of the objective function. A decision is made by comparing the target vector with the corresponding trial vector. If the trial vector produces the smaller value of objective function (or fitness function) than the target vector, the target vector will be replaced by the trial vector as the next-generation population. On the contrary, the target vector is retained. Eq. 2-10 formulates the selection process:

$$x_i^{g+1} = \begin{cases} u_i^{g+1}, & f(u_i^{g+1}) < f(x_i^g) \\ x_i^g, & \text{otherwise} \end{cases} \quad \text{Eq. 2-10}$$

where f is the objective function (or fitness function) to be minimized.

Differential evolution imitates the biological behavior and tries to find the global optimum of the multi-dimensional objective function in the continuous space. It is also easy to be implemented and is an effective global optimization algorithm.

2.4. VIRTUAL VIDEO TOOL

In general, we have to set up real cameras to verify the proposed surveillance system. From time to time, we need to change the experimental environments and the adjustment may cost a lot money and time. Hence, using virtual reality for experiments is another choice to release the dilemma. In the literature, there have been some examples, like [17] and [18], that use virtual reality tools to help the development of their surveillance systems.

In [19], Taylor *et al.* developed a virtual video tool for surveillance simulation and evaluation. They call it ObjectVideo Virtual Video (OVVV), which is a modification based on the game engine of Half-Life 2 by Valve Software. It can simulate static or active cameras and render video streams. In addition, it can also extract the ground truth from each camera automatically to help performance evaluation.

Figure 2-16 shows the block diagrams of the OVVV system. The camera server manages the virtual cameras which are defined by several camera parameters, including frame rate, orientation, location, and field of view (FOV). This system can render videos for each virtual camera. The PTZ server controls the PTZ parameters of each virtual camera. Because of the utilization of TCP/IP (Transmission Control Protocol/Internet Protocol) protocol, we can access the camera and PTZ servers via internet. We can get the videos generated by virtual cameras and adjust the PTZ parameters of each camera remotely through the video client and PTZ client. Moreover, we do not necessarily operate them on only one computer. In other words, we can manipulate them even on the computer where the OVVV system is not installed.

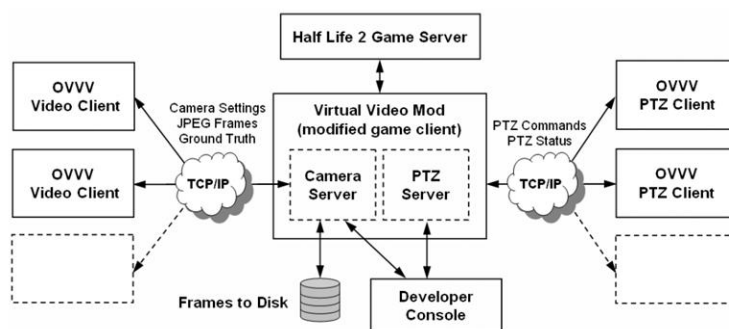


Figure 2-16 OVVV system [19]

OVVV system is not just a simple virtual video generator. In order to simulate real cameras, several kinds of noise and camera distortion can be added optionally, including additive pixel noise, video ghost, radial distortion, blur, defocus, and jitter. Users can also change the level of noise or distortion arbitrarily. Based on these functionalities, we'll be able to discuss the relationship between noise interference and the performance of the surveillance system. An example of noise addition is shown in Figure 2-17. Besides noise and distortion, the OVVV system can also simulate omni-cameras, such as panoramic and parabolic catadioptric omni-directional cameras. These two kinds of cameras views are shown in Figure 2-18. These functions can increase the usability for many kinds of surveillance experiments.

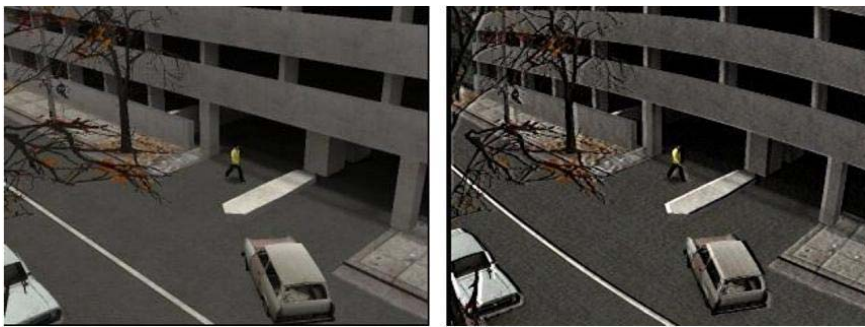


Figure 2-17 The synthetic frames with (right) and without (left) noises [19]



Figure 2-18 The synthetic frames of omnicams: panoramic (left) parabolic catadioptric (right) omni-directional cameras [19]

OVVV system does not only aim at simulation but evaluation. It can generate the ground truth to support the evaluation of surveillance systems. It includes both camera and target ground truth. The camera ground truth consists of camera center, camera orientation, horizontal FOV, and frame dimensions. The target ground truth consists of 3D world location of target center, target center on image, foreground label map, bounding box of an entire target, and bounding box of a visible target. Figure 2-19 shows an example of the ground truth. In the left figure of Figure 2-19, the dashed line represents the bounding box of an entire target and the solid one represents the

bounding box of visible target. The different bounding boxes help us to evaluate the performance of the system under the occlusion situation.



Figure 2-19 A ground truth examples: bounding box (left) and label map (right) [19]

Because the scenarios and scripts are simulated virtually, we can repeat the experiments with the same experimental environment to improve our surveillance system. In addition, we can acquire those sequences that are hard to make. We can also place cameras at any place and can control these cameras easily. Although eventually we still have to test our surveillance in the real world, the use of the OVVV tools can shorten the period of system development and increase the feasibility of the developed system. With the help of the OVVV system, we can greatly reduce the cost of development.



Chapter 3.

CAMERA COORDINATION

Figure 3-1 shows the flow chart of our proposed coordination system. In this thesis, focus on the coordination of multiple cameras. Here, we assume all pre-processes, like camera calibration, object detection, face detection, and object tracking, have already been done. Hence, the 3D locations of the targets and the orientations of the target faces are available beforehand. Here we utilize the ground truth of OVVV to accomplish these tasks. In this chapter, we'll discuss how to formulate the coordination problem and how to apply a suitable optimization tool to achieve the goal.

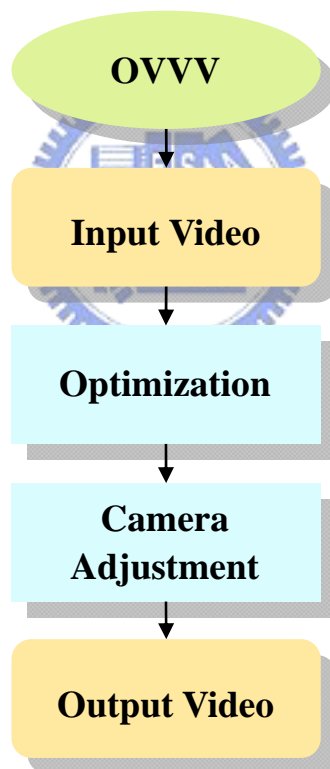


Figure 3-1 Flow chart of our proposed camera coordination system

3.1. PROBLEM FORMULATION

At the start, we define the problem that we want to solve. Unlike the articles we introduce in Section 2.1, we aim to capture as many frontal high-resolution facial images as possible during the presence of the monitored targets. In the proposed

algorithm, PTZ cameras are allowed to cover more than one target at each time, as long as the captured facial images are sufficiently clear. Moreover, we allow the tracking of a target can be handed over from one PTZ camera to another PTZ camera so that the face of that target can be better observed over time. In the proposed algorithm, we design our camera coordination system based on two major criteria: frontal shoot and high-resolution shoot.

To formulate these two criteria, we define the shoot angle θ_{ij} , and the face width W_{ij} . In θ_{ij} and W_{ij} , the subscript i denotes the i -th PTZ camera, while the subscript j denotes the j -th target. As shown in Figure 3-2, the shoot angle θ_{ij} represents the angle between the blue arrow $\overrightarrow{cam_{ij}}$ and the green arrow $\overrightarrow{face_j}$. $\overrightarrow{cam_{ij}}$ indicates the line connecting the i -th PTZ camera and the j -th target, while $\overrightarrow{face_j}$ indicates the facing orientation of the j -th target. As the j -th target is looking toward the i -th camera, we have a smaller shoot angle. On the other hand, as shown in Figure 3-3, the shot face width W_{ij} represents the width of the j -th target's face in the image captured by the i -th camera. A larger value of W_{ij} indicates a better observation of the j -th target in the i -th camera image.

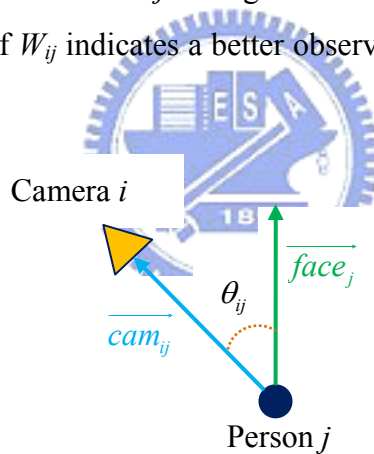


Figure 3-2 The illustration of θ_{ij}

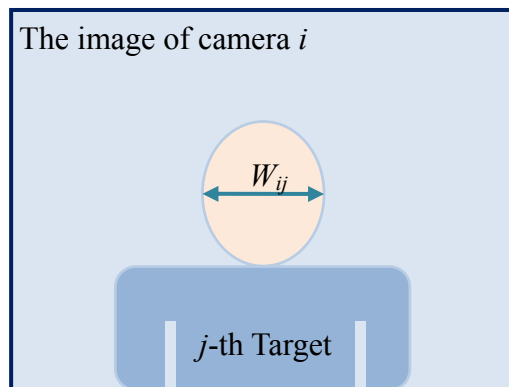


Figure 3-3 The illustration of W_{ij}

To simplify the computation of θ_{ij} and W_{ij} , all 3-D vectors are projected onto the ground plan to form 2-D vectors instead of calculating 3-D vectors directly. In other words, we only consider the 2D vectors here in order to simplify the computation. In the simplified forms, the shoot angle and the face width are defined as follows.

$$\theta_{ij} = \text{acos}\left(\frac{\overrightarrow{cam_{ij}} \cdot \overrightarrow{face_j}}{\|\overrightarrow{cam_{ij}}\| \|\overrightarrow{face_j}\|}\right) \quad \text{Eq. 3-1}$$

$$W_{ij} = f_{xi} \frac{\text{Face width in 3D space}}{D_{ij}} \quad \text{Eq. 3-2}$$

$$f_{xi} = \frac{\text{Image width}}{2 \cdot \tan\left(\frac{FOV_i}{2}\right)} \quad \text{Eq. 3-3}$$

In the definition of W_{ij} , f_{xi} denotes the focal length of the i -th PTZ camera in the horizontal direction, D_{ij} is the distance between the i -th camera and the j -th target, and FOV_i is the field of view of the i -th PTZ camera. Eq. 3-2 and Eq. 3-3 originate in the pinhole camera model. Originally Eq. 3-2 is used only when the face is on the center of image. However, we do not need a very precise face width in the image. Hence, we simply define the face width in an approximated way to simplify the computation.

The shoot angle and the face width are two different physical quantities. In addition, the desired tendencies of the two quantities are different. Basically, we prefer to capture a facial image with a smaller shoot angle but a larger face width. Therefore, we apply two mapping functions $N_\theta(\cdot)$ and $N_w(\cdot)$ over θ_{ij} and W_{ij} to convert them into two normalized measures. The two different quantities can be unified after normalizing. Here we define the values to become larger after normalizing when the performance we desire is become well. In other words, we set higher “scores” for better capture situations. For example, we hope that θ_{ij} is as small as possible so that we can see more frontal face. Therefore, the value of $N_\theta(x)$ becomes larger as the x becomes smaller. These two mapping functions are defined as follows and are illustrated in Figure 3-4 and Figure 3-5.

$$N_\theta(x) = \begin{cases} -\frac{r_\theta \cdot k}{th_\theta} x + \frac{(1+r_\theta)k}{2}, & 0 \leq x < th_\theta \\ 0, & \text{otherwise} \end{cases} \quad \text{Eq. 3-4}$$

$$N_w(x) = \begin{cases} 0 & , x < th_{min} \\ \frac{r_w \cdot k}{th_{max} - th_{min}} x - \frac{r_w \cdot k \cdot th_{min}}{th_{max} - th_{min}} + \frac{(1-r_w)k}{2} & , th_{min} \leq x < th_{max} \\ \frac{(1+r_w)k}{2} & , x \geq th_{max} \end{cases} \quad \text{Eq. 3-5}$$

In Eq. 3-4 and Eq. 3-5, k is a positive constant that controls the dynamic range of $N_\theta(x)$ and $N_w(x)$. r_θ and r_w are real numbers within the range $[0, 1]$ and they control the slopes of $N_\theta(x)$ and $N_w(x)$. th_θ , th_{min} , and th_{max} are pre-defined thresholds. th_θ represents the worst situation that can be allowed for capturing the frontal face. th_{min} represents the minimum face width for clear observation. On the other hand, when the face width is wider than th_{max} , we think the facial image has achieved the level of perfect observation. These thresholds can be varied by the users for different applications.

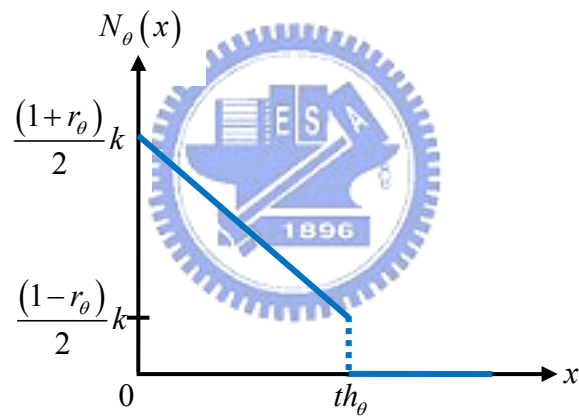


Figure 3-4 Normalized function of the bias angle

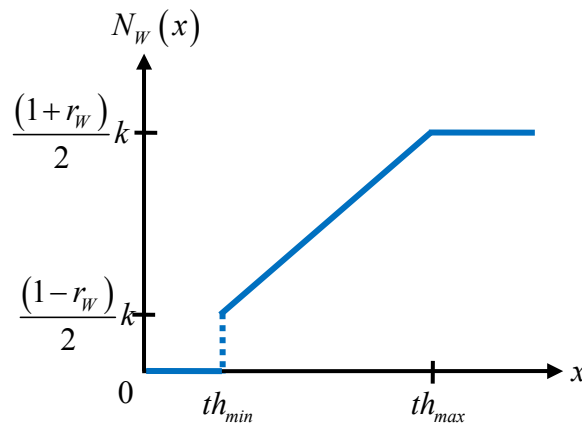


Figure 3-5 Normalized function of the face width

The physical meaning of th_θ is the worst situation for capturing the frontal face. In other words, we hardly clearly see (or identify) someone's face when the angle between face vector and camera vector exceeds th_θ . Similarly, th_{min} and th_{max} represent the worst and best case of face width in the image respectively. When the face width is smaller than th_{min} , we also hardly see the clear face because of the low resolution. Conversely, when the face width reaches or exceeds the threshold, th_{max} , we can clearly identify this face. The function of r_θ and r_W are to adjust the slopes of the linear part of the normalized functions and the maximal and minimal values of the normalized functions. It will affect the weightings of the orientation and clearness. For example, if r_W becomes smaller, the largest and smallest values of the normalized function will be closer and the difference between them is smaller. That means the discrimination of the face resolution is decreased. Under the extreme condition, if we let the r_W be zero (and it will make the slope zero), any face width will get the same normalized value. That makes no difference no matter what the face width is after the normalization.

The goal is that our system finds a camera coordination way to make each θ_{ij} as small as possible while make each W_{ij} as large as possible. With the definitions of N_θ and N_W , we then define $Eval()$ (Eq. 3-6) for the face capture of the j -th target by the i -th camera. It is defined to evaluate the different coordination. The large the value of $Eval()$ is, the better the performance of coordination is.

$$Eval(AP) = \sum_{i=1}^m \sum_{j=1}^n ap_{ij} N_\theta(\theta_{ij}) N_W(W_{ij}) \quad \text{Eq. 3-6}$$

In Eq. 3-6, m and n are the number of cameras and targets respectively. AP denotes a set of camera assignments and is defined as Eq. 3-7:

$$AP = \{ap_{ij}\}, \quad i = 1, 2, \dots, m, \quad j = 1, 2, \dots, n \quad \text{Eq. 3-7}$$

ap_{ij} represents the binary assignment parameters. ap_{ij} is equal to 1 if the i -th camera is assigned to monitor the j -th target, and ap_{ij} is equal to 0 otherwise. Hence, for a camera assignment AP , $Eval(AP)$ represents the overall observation levels of the n targets by all m cameras. When more targets can be better observed by their corresponding cameras, with smaller shoot angles and larger face widths, we have a larger $Eval(AP)$. Hence, the goal of the proposed camera coordination system is simply to find the optimal camera assignment that reaches the largest $Eval(AP)$. Moreover, as these n targets keep moving within the monitored scene, we need to

adaptively adjust the assignment of cameras to achieve the most preferable observation.

Besides, to simplify the problem, we also add one extra constraint over Eq. 3-6. The constraint is stated in Eq. 3-8:

$$\sum_{i=1}^m ap_{ik} = 1, \quad k = 1, 2, \dots, n \quad \text{Eq. 3-8}$$

This constraint implies that we only take into account the camera view that is assigned to the target even though that target may also appear in some other views.

Because there are two criteria, one target has two observation level, the level of orientation (shoot angle) and the level of clearness (face width). They are the values of the two normalized functions, N_θ and N_w , respectively. The zero values of the normalized functions mean that the situation of orientation or clearness is too bad to identify the target's face. Here, we multiply these two scores together to form the final score. This is because we consider these two scores to be dependent. We consider that if one of the scores for a target is low, we will not be able to clearly see that target even though the other score is high. Hence, as one score is high but the other one is low, the final score is still low. In addition, when the performance of orientation or clearness is lower than a threshold, according to Eq. 3-4 or Eq. 3-5, the value of Eq. 3-6 (total score) is set to zero.

We add all the targets' overall observation levels together to evaluate the performance of camera coordination for all targets. Obviously, according to the mapping functions we define, the value of the evaluation function (Eq. 3-6) will become larger if the performance of coordination gets better. That is to say, more frontal and higher resolution faces. Thus, the goal is that we want to find a set of camera assignment (assigned parameters), AP , which makes the evaluation function maximal. In other words, we want to find an optimal AP here.

3.2. SIGNIFICANCE WEIGHT

In theory, we can always find an optimal AP for the evaluation function at any time instant. However, people's behavior is highly diverse. It is very likely that even with the optimal camera assignment we still cannot clearly capture all people's faces at some time instants. In addition, the evaluation function takes all people into

account and the evaluation function is actually a tradeoff among all cameras. It may happen that some people's observation levels are sacrificed to gain other people's observation levels. Hence, the proposed system cannot guarantee that all people's faces are always clearly observed.

Because the evaluation function takes all people into account, sometimes the tradeoff situation happens when we carry out the optimization. It means that maybe some people's observation level is sacrificed to increase some other people's observation level. The increased value of evaluation function may be larger than the sacrificed value so the system will prefer this kind of coordination during the optimization process. Figure 3-6 shows an example of the optimization tradeoff. In Figure 3-6, two different cases are illustrated. Compared with Figure 3-6(a), Camera 2 in Figure 3-6(b) cannot capture Person 3's frontal face and we lose some scores on it. However, the FOV of Camera 1 becomes small because Camera 1 only needs to take charge of Person 1 and Person 2. As the FOV becomes smaller, the scores of Person 1 and Person 2 increase. The total increased amount is larger than the decreased amount and the total scores become higher.

The cases that the system cannot always cover all people's faces are unavoidable. However, we still hope to clearly see the unclear faces in the next moment. We hope we'll be able to clearly see all people's face during some periods of time and try to capture as many frontal high-resolution facial images as possible.

To deal with this problem, we assign each target a significance weight to represent the priority of that target. In other word, it represents the importance of the target. This weight will increase if the target hasn't been clearly observed in the past few moments. On the contrary, if that target has already been clearly observed for a while, we decrease its significance weight. Here, target's "clearness" is defined by his/her observation level. The zero observation level means that the corresponding target cannot be observed at all.

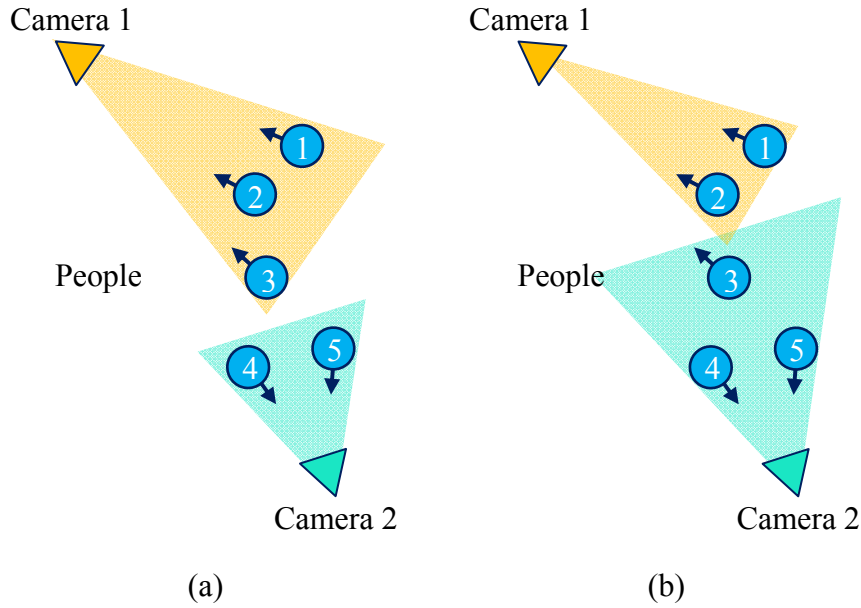


Figure 3-6 (a) lower (b) higher overall observation level

3.2.1. WEIGHTING UPDATE

The usage of significance weight is to help the clear capture of targets' faces. The values of weights are closely related to the situation that targets cannot be clearly captured. The trend of significance weight roughly follows the states of the clearness. Here, we design the adjustment of significance weight to include three major states: rise, hold, and decline.

3.2.1.1. RISE STATE

The weight increases continuously in the rise state. When the face of a target cannot be clearly captured, we linearly increase its significance weight. When the weight is raised, the system will pay more attention to that target and it's more likely that the target can be better observed. The value of weight is 0 initially. When a target's face is unclear, his or her weight starts to increase. If the unclear situation is continuous, the value will also increase continuously. It will stop increasing when the unclear situation is improved.

3.2.1.2. HOLD STATE

Once if the system has adjusted its camera coordination to take clear facial

picture of that target, the significance weight will be held at a high value for a while. At this time, we stop to increase the value of the significance weight because we have already clearly seen the target's face. Although we stop to increase the weight, we do not decrease the value of weight immediately. This is because we hope we can clearly see the person's face for a while, but not just a short glimpse. Hence, the significance weight is held in the holding state for a pre-defined period to ensure the target's face can be clearly observed for a long enough period.

3.2.1.3. DECLINE STATE

After keeping a period of “hold”, the significance weight of the target is decreased gradually as long as the target's face can be clearly captured. This is because we have paid attention to the target for a long enough period in the holding state and the target is no longer as important as before. Similar to the rise state, we reduce the value linearly. The value will be continuously reduced to zero as long as the target's face can be clearly captured continuously.

These three states are alternately taken place until the weight comes back to the initial state, that is, the zero value. Once a target's face becomes unclear, his/her weight is in the “rise” state again. As the target's face can be clearly observed, the state switches to “hold” for a while. If the face is continuously clear after a period of time, the state will switch to the “decline” state. However, if the face becomes unclear suddenly during the “hold” or “decline” state, it will switch back to the “rise” state to enforce a higher priority in capturing the clear image of that face.

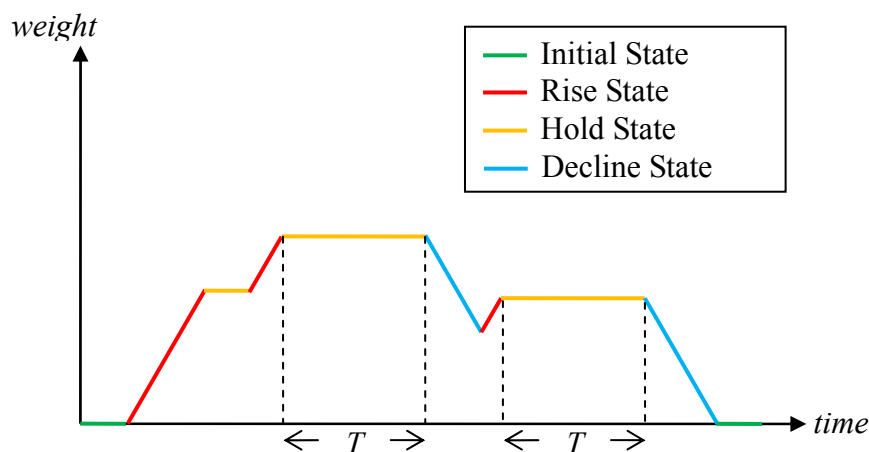


Figure 3-7 Variation of the significance weight over time

An example of the switching of these three states is illustrated in Figure 3-7. At the beginning, the target's face is not clear within the “rise” state. T represents the

holding time. As illustrated in Figure 3-7, whenever the unclear condition happens, the “rise” state takes place again. On the other hand, as the target can be clearly observed for a while, the weight drops to zero in the “decline” state.

3.2.2. UPPER BOUND OF UNCLEAR PERIOD

Although the significance weight can help us in alleviating unclear observation, it still takes a while for an unclear observed target to get clearly observed. In some situations, the lag can be too long for practical usage. Hence, we put an upper bound over the unclear period. If the time period that a target hasn’t been clear observed exceeds a pre-defined threshold, its significance weight is dramatically raised to a very large value. This pushes the camera coordination system to take quick response to take good care of that target.

As we dramatically raise the weight to a very large value, the people who are unclear before will have a very high priority to be clearly captured. Similar to the hold state mentioned in Section 3.2.1.2, this large value is also held for a long enough period. However, this period can be different than the aforementioned “hold” period. Moreover, the value of the weight is reset to zero this time when the high-value hold process ends, as illustrated in Figure 3-8.

On the other hand, we also take the value of significance weight into account. When the value of the weight increases to a certain level but the corresponding target is still not captured well, we also adjust the target’s significance weight to a very high value, as illustrated in Figure 3-9.

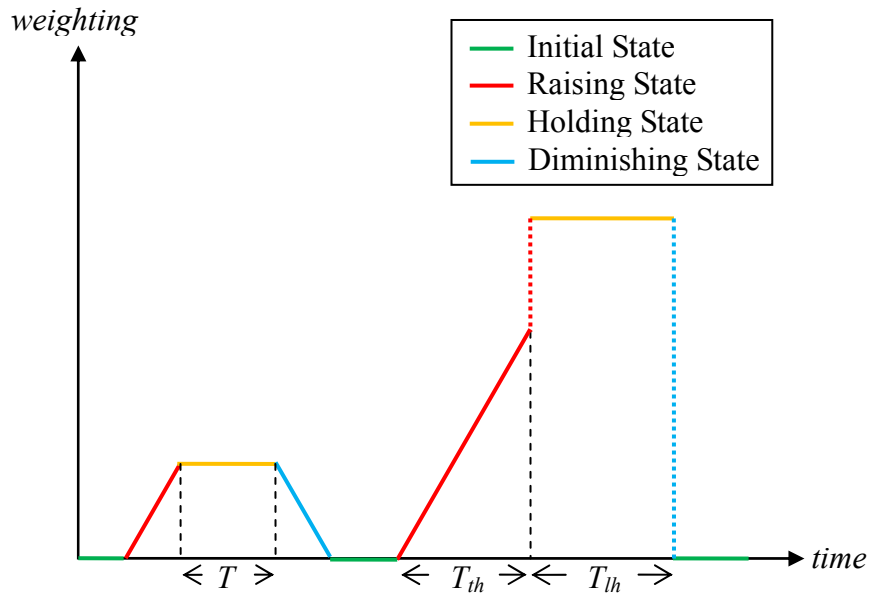


Figure 3-8 Illustration of the weighting variation for the case of time limitation

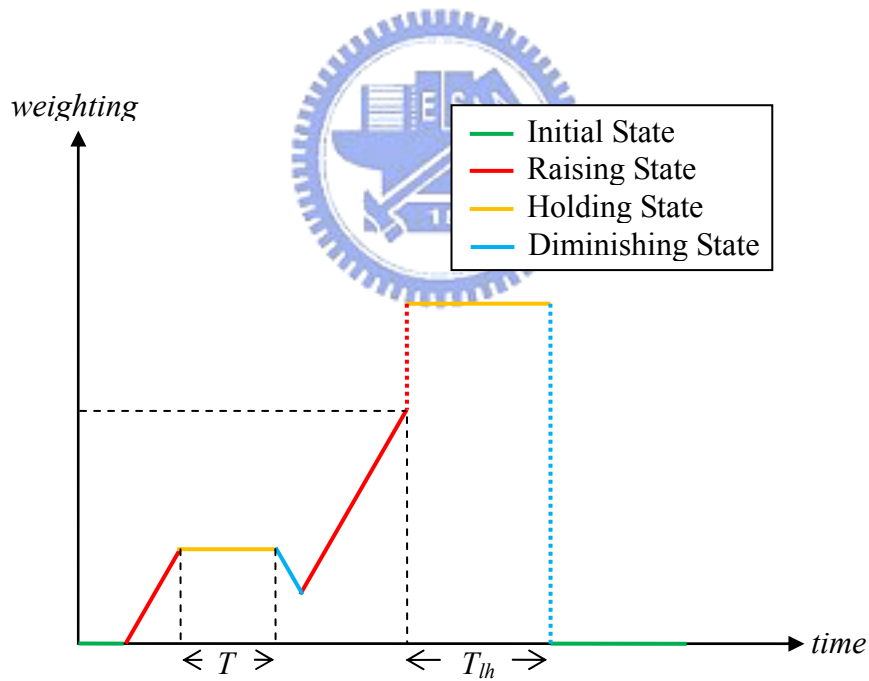


Figure 3-9 Illustration of the weighting variation for the case of weighting limitation

3.2.3. COMBINING WEIGHT WITH EVALUATION

Because we want to take significance weight into account when we adjust the camera coordination, we incorporate significance weight into the evaluation function. Our object is that a target with higher weight will have a higher priority to be clearly captured. To realize the concept of importance weight, we add penalty term into the definition of $Eval()$. If a target is assigned to a camera which cannot clearly capture his/her face by an AP , the evaluated value of the AP will be added a penalty term. This causes a value to be deducted from the original evaluated value. Hence, we redefine the evaluation function as below:

$$Eval(AP) = \sum_{i=1}^m \sum_{j=1}^n ap_{ij} (N_{\theta}(\theta_{ij}) N_w(W_{ij}) - pv_{ij}) \quad \text{Eq. 3-9}$$

where the penalty term pv_{ij} is defined as

$$pv_{ij} = cf_{ij} \cdot sw_j \cdot c_p \quad \text{Eq. 3-10}$$

In Eq. 3-10, sw_j stands for the significance weight of the j -th target, cf_{ij} represents the clear factor of the j -th target with respect to the i -th camera, and c_p is a controlling parameter. The clear factor cf_{ij} is equal to 0 if the j -th target can be clearly observed by the i -th camera. Otherwise, cf_{ij} is equal to 1. Apparently, the penalty value is determined by the significance weight. The higher the weight is, the larger the penalty value is. With the inclusion of the penalty term, the camera coordination system can automatically pay more attention to these targets with larger significance weights.

3.3. MODIFIED DISCRETE BINARY PSO

In Eq. 3-9, we redefine our problem and want to find an AP to maximize the evaluation function. In mathematics, this is simply an optimization problem. Unfortunately, Eq. 3-9 has a nonlinear and non-differentiable form. To find the optimal AP , these classical optimization algorithms, like the gradient descent algorithm, cannot be used. Instead, we adopt the particle swarm optimization algorithm [12] mentioned in Section 2.3.1 to tackle this problem. Due to the binary nature of the assignment parameters, we actually adopt the discrete binary particle swarm optimization proposed in [14]. Moreover, since we have added one constraint in the evaluation function, we further make some modifications over the discrete

binary particle swarm optimization algorithm to tackle the problem.

According to the discrete binary PSO algorithm, the evaluation function Eq. 3-9 is equal to the objective function in Eq. 2-4. We want to find an AP that maximizes the evaluation function. AP is equivalent to the \mathbf{x} in Eq. 2-4. Here, we can consider it as a bit string composed of a set of ap_{ij} . An AP can be thought as a particle position too. In the first step of DBPSO, we will generate a number of AP 's first.

3.3.1. PARTICLE GENERATION

In the modified DBPSO, each particle represents a possible AP . In the original form of PSO, particles are randomly generated in the initial stage. The use of random particles increases the probability of finding the global optimum. However, this also causes a large number of iterations. To speed up the computations, we develop two simple but effective schemes to generate particles. At the first scheme, we utilize clustering to generate a reasonable initial guess about AP and use it to produce particles. In addition to the initial clustering over the monitored targets, we also utilize the temporal information in the second scheme to speed up the optimization process in subsequent frames.



3.3.1.1. FEATURE SPACE

In our approach, we consider that the people with similar characteristics should be assigned to the same camera. The characteristics we think are people's positions and orientations. For the sake of efficiency, people who are close to each other and have similar face orientations are more likely to be assigned to the same camera. Hence, we use the clustering technique first for the design of camera coordination.

We first create a feature space and convert people's characteristics into this space. In other words, the characteristics of each person correspond to a data point in the feature space. Here, we define the feature space based on people's positions and orientations. We then perform clustering over the data points. For people's positions, we consider the 2D coordinates (X,Y). For people's orientations, we use the inner product of the camera vector and the face vector. The camera vector and face vector are illustrated in Figure 3-2. Here, we do not directly use the inner product of these two vectors. Instead, we check the cosine of the included angle between these two vectors.

However, position and orientation are very different physical quantities. Hence, we further normalize these two quantities. Since the value of the cosine of the included angle is within the range $[-1, 1]$, we normalize the positions to be within the same range. That is, the origin of the (X, Y) coordinates is translated to the center of the space. Then, the new coordinates of X and Y are divided by the half width of the space to get the normalized coordinates (X', Y') , as illustrated in Figure 3-10.

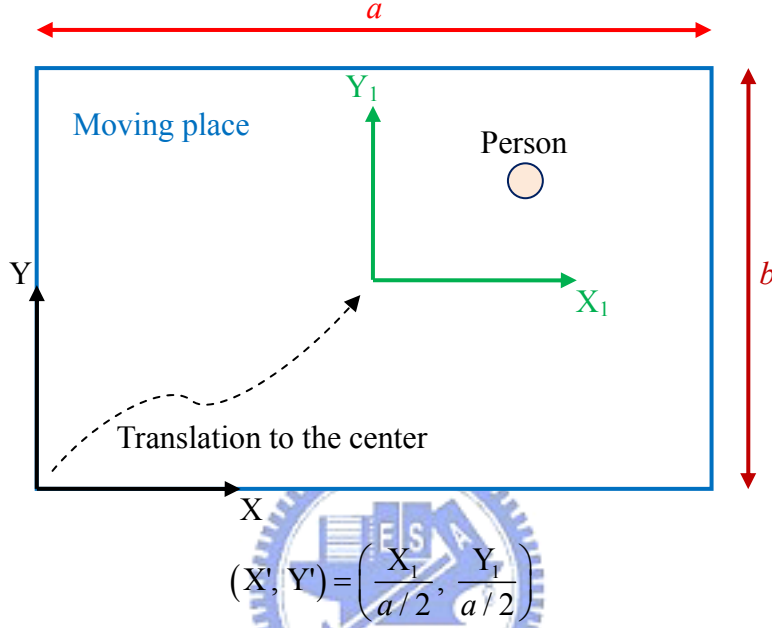


Figure 3-10 An example of coordinate normalization

In Figure 3-10, we use a to normalize the position because it is longer than b . With this normalization, the values of X' and Y' are in the range $[-1, 1]$.

Assume m is the number of cameras. We define the dimension of the feature space to be $m+2$. For example, as we install four PTZ cameras, the feature space has 6 dimensions and each target corresponds to a 6-D data point as expressed in Eq. 3-11:

$$(\lambda X', \lambda Y', IP_1, IP_2, IP_3, IP_4) \quad \text{Eq. 3-11}$$

where λ is a scalar to balance between positions and orientations. In Eq. 3-11, X' and Y' represent the normalized coordinates of the target on the ground plane. Both X' and Y' have the range $[-1, 1]$. IP_i represents the normalized inner product between $\overrightarrow{face_j}$ and $\overrightarrow{cam_{ij}}$. IP_i has the range $[-1, 1]$. Moreover, because the orientation characteristic has m dimensions but the position one only has two, we use λ to balance it. In this case, we choose $\lambda = 2$.

3.3.1.2. GENERATION BY CLUSTERING

After converting each target into a data point in the $m+2$ dimensional feature space, we choose the k-means clustering algorithm [8] to cluster n targets into m clusters. After clustering, we assign each group to a camera. Since the centroid of each group represent the mean of the clusters, we use these centroids to help the assignment of cameras. For each camera, the values of IP 's of the centroids are compared with each other. We assign to that camera the group of feature points which has the smallest IP . For example, assume we have four PTZ cameras and four groups. For Camera 1, four IP_i 's are compared and the group which has the smallest IP_i is assigned to Camera 1. The same process is carried out for the remaining cameras.

This clustering creates the initial guess about the optimal camera assignment. That is to say, we obtain an initial AP (a set of ap_{ij}). We then randomly generate a few particles around the initial AP . For example, assume there are six people and three PTZ cameras. Initially, the first and second, third and fourth, and fifth and sixth people are assigned to Camera 1, Camera 2, and Camera 3, respectively. Here, we change one target's assignment at one time. If the first target is selected, we then randomly choose a camera for the target to be assigned to. Of course, since the original assignment of the first target is Camera 1, Camera 1 is excluded from the random selection process. After the random selection, we get a new AP , which is very close to the initial AP .

In the previous paragraph, we only change one target's assignment to generate a new AP . In practice, if the numbers of people and particles increases, we may change the assignment of more targets to increase the randomness. We can also repeat the above process several times to generate a number of different random AP s as a portion of the initial particles.

3.3.1.3. GENERATION BY THE LATEST ASSIGNMENT

In addition to the initial clustering over the monitored targets, we also utilize the temporal information to speed up the optimization process in subsequent frames. In general, the time interval between two consecutive frames is small and we can reasonably assume the variations over targets' positions and orientations are small between successive frames. Hence, the optimal AP at the previous time instant can be used as the initial guess of AP at the current time instant. Here, the latest optimal AP

can be obtained from latest result of the DBPSO. Then, we use it to generate a number of different AP 's. The generating process is the same as that in Section 3.3.1.2.

3.3.1.4. GENERATION BY RANDOM SELECTION

Even though we have used the initial clustering and the temporal prediction to speed up the computations of DBPSO, the optimization process may easily fall into a local optimum if we only use these particles generated around the initial guess. Hence, in our implementation, a portion of particles are still randomly generated. These random AP 's are generated by choosing the assignment from a uniform distribution, with mutually independent people assignment. For example, we assume there are four PTZ cameras and each target has four possible assignments. Then, all these four choices are equally probabilistic.

3.3.2. OPTIMIZATION WITH CONSTRAINTS

The DBPSO [14] technique is performed after the generation of initial particles. The objective function is defined in Eq. 3-9. Here we want to find a set of assignment parameters, an AP , which maximizes the objective function. However, the DBPSO proposed by Kennedy and Eberhart in [14] does not concern the problem with constraints. Hence, we also make some modification over DBPSO to fit for our constrained problem.

In our approach, the main optimization process basically follows the algorithm mentioned in Section 2.3.1.2. First, the best previous position of each particle, $p_{k,ij}$, and the best global position, $p_{g,ij}$, are calculated. Second, the velocity of each ap_{ij} is updated by Eq. 2-5 and we rewrite it with respect to the assigned parameters:

$$v_{k,ij}^{t+1} = \omega \cdot v_{ij}^t + \varphi_1 \cdot (p_{k,ij} - ap_{k,ij}^t) + \varphi_2 \cdot (p_{g,ij} - ap_{k,ij}^t) \quad \text{Eq. 3-12}$$

where t represent the time instant, k is the index of particle, ω is the inertia factor, and φ_1 and φ_2 are random numbers generated from the uniform distribution in the interval between 0 and 1. We choose the inertia factor from the range [0.8 1]. Finally, the positions of particles are updated according to Eq. 2-6. Here, we only have to replace the parameter x with ap . Eq. 3-13 is the rewritten formula:

$$\begin{aligned} & \text{if } (rand(0,1) < S(v_{k,ij}^{t+1})) \text{ then } ap_{k,ij}^{t+1} = 1 \\ & \text{else } ap_{k,ij}^{t+1} = 0 \end{aligned} \quad \text{Eq. 3-13}$$

where $rand(0,1)$ is a random number selected from a uniform distribution in $[0, 1]$, and S is the sigmoid function. No matter whether the objective function is a maximization or minimization problem, the velocity and position updating functions are the same. All we need to do is to replace the parameter representation.

Similar to the DBPSO algorithm, we repeat this process until the stop criterion is reached. However, due to the constraint that the assigned parameters must obey Eq. 3-8, we slightly modified the original DBPSO to take into account the constraint. As before, the positions of particles are updated according to Eq. 3-13. However, because the random factor is used, we cannot assure that the assigned parameters will be 0 or 1. Naturally the constraint formulated in Eq. 3-8 is not guaranteed to be obeyed. Once if the constraint fails, we try to “repair” the assigned parameters to make them fit the constraint. The illegal situation occurs as one target is either assigned to more than one camera or assigned to no camera after the DBPSO process.

To “repair” the assignment means that we have to design a method to alter the “wrong” particles of the DBPSO. For the case that some target is assigned to more than one camera, we modify the assigned parameters in the ap_{ik} set for the target k and make only one assigned parameter be 1. The simplest approach is to randomly select one assigned parameter to be 1 among the assigned parameters which are 1 for this target. However, as we physically implemented this simple method, the experiments showed that the particles usually do not converge quickly in a few iterations. Hence, we developed an alternative method that has a better performance. In the following, we’ll explain how we modify the original DBPSO to fit the constraint.

According to the concept of PSO, the particles will gradually move toward the best position based on its previous best experience and the best global experience. We utilize this concept and take these two positions into account as we carry out the repair process.

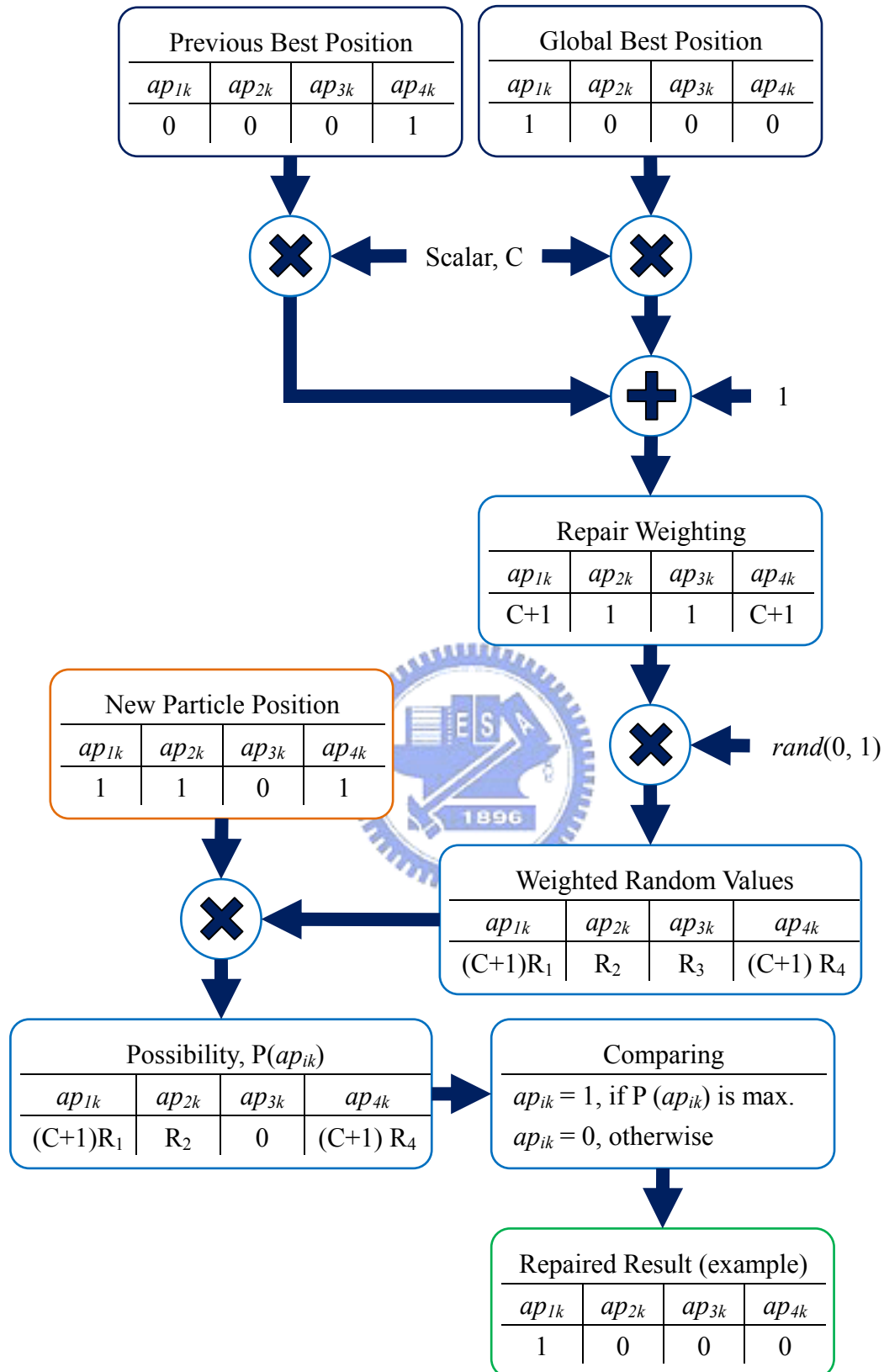


Figure 3-11 The illustration of constraint repair of each iteration of binary PSO

When the assigned parameters for someone have multiple 1's, only a single "1" will retain while the others will be set zero. Every assigned parameter has its

possibility to be 1 during the process of selection. In our approach, the assigned parameter which has the highest possibility of becoming 1 is retained. Here, if the assigned parameter is 1 and its best previous value or the best previous global value is 1, it has higher possibility to retain as 1 in the repair process. That is to say, we tend to choose the best local or global position when we encounter the constraint violation.

In the repair process, we randomly generate the possibilities for each assigned parameter at the start. Then, we determine their weights according to its best previous position and the best previous global position. Finally, the possibilities of the assigned parameters are compared with each other. The assigned parameter that has the highest possibilities will be kept while the others are set zero. Figure 3-11 shows an example of the repair process. Here we assume that there are four PTZ cameras that aim at the k -th target. We multiply the best previously assigned value and the best global assigned value by a scalar, C , and add them together to form the repair weight. We also add an extra one for all repair weights to keep them from being 0. Then, every repair weighting is multiplied by a random number in $[0, 1]$ to get the weighted random value for each assigned parameter. After multiplying the weighted random value with the new assignment, we compare the possibilities and keep the assigned parameter whose possibility is the highest.

On the other hand, if the assigned parameters of a target are all zero, we slightly modify the repair process. As illustrated in Figure 3-12, we directly take the weighted random values as the possibility and choose one ap to be 1 by comparing this possibility. Compared with the original process, we only leave out the multiplication of the unrepaired assigned parameters. The core concept is basically the same.

In the repair process, C is a weighting factor that affects the tendency of the repair process. The value of C cannot be negative. If C becomes larger, the system will prefer to choose the best previous assignment or the best global assignment. On the contrary, if C becomes smaller, the repair process will be more like a random selection. Finally, in Figure 3-13, we show the block diagram of the modified optimization process.

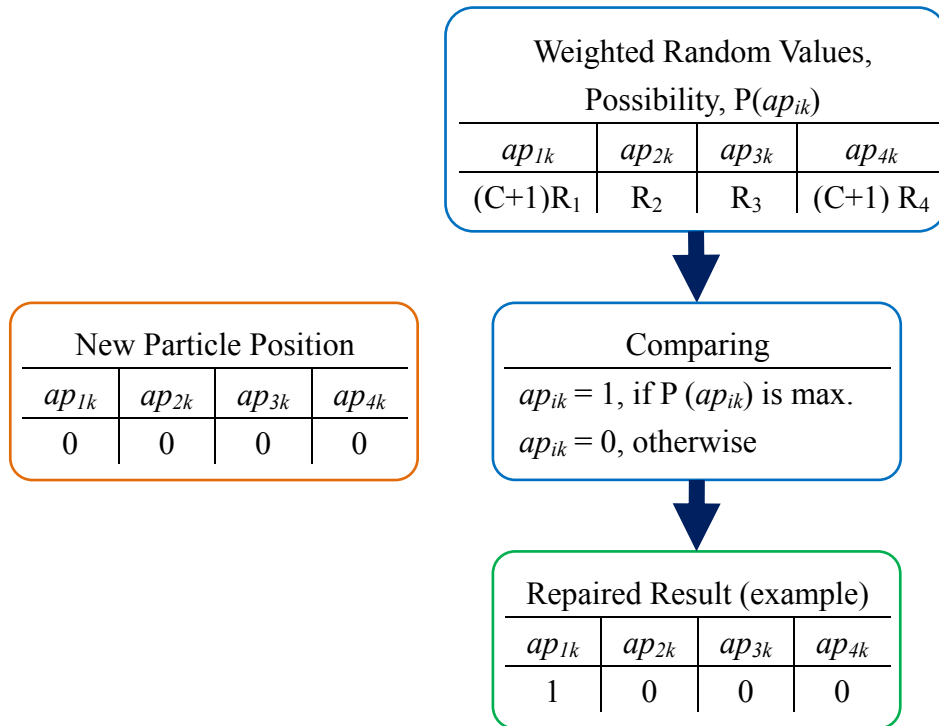


Figure 3-12 Repair process for the no assignment case

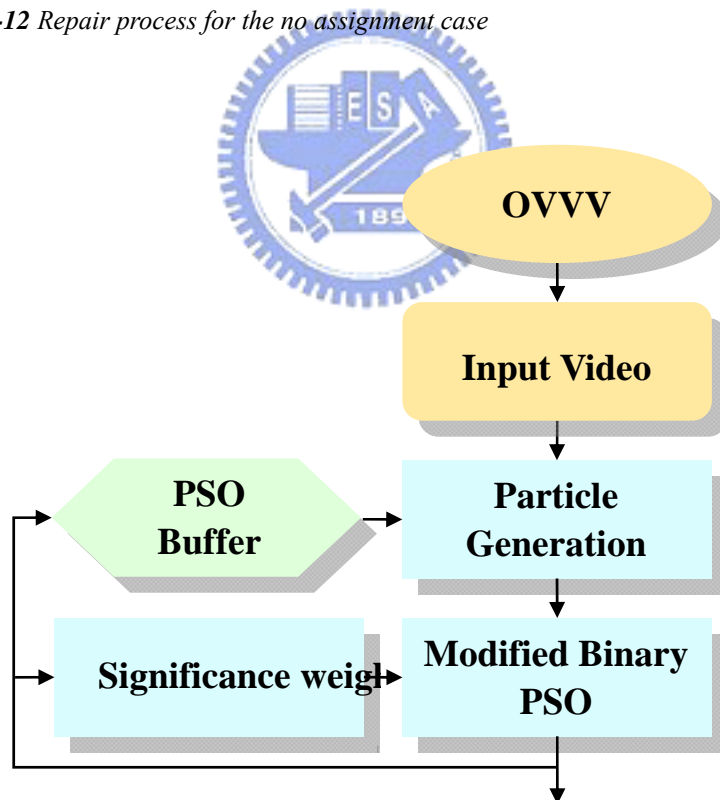


Figure 3-13 Block diagram of the optimization process

3.4. CAMERA ASSIGNMENT

After carrying out the modified DBPSO, we get an optimal AP . This AP is the globally optimal result found by the optimization process. So far, the cameras still have no idea about who to look and where to look. Hence, we have to further convert AP into the assignments of cameras.

3.4.1. WHO TO LOOK?

Every camera assignment should be determined before estimating the focusing location. Because ap_{ij} decide whether the i -th camera should take charge of the j -th target, it is very easy to know the assignment of targets. We simply need to collect the values of assigned parameters which are one and then get the assignment for each camera.

3.4.2. WHERE TO LOOK?

In our system, one PTZ camera does not only capture one target's facial image. Hence, we have to adjust the field of view (FOV) of the camera to capture all the people it wants to capture. In addition, the camera must also adjust its FOV to get as high resolution as possible.

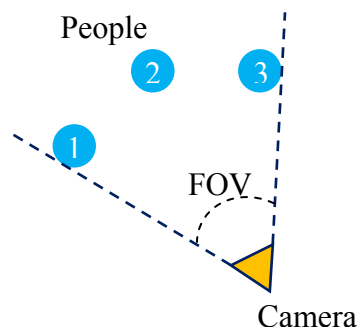


Figure 3-14 A camera takes charge of three people

Figure 3-14 is an example where a PTZ camera takes charge of three people. In this kind case, we determine FOV of the camera first. This FOV cannot be too small to capture all the people the camera wants to take charge of. On the other hand, we

hope this FOV can be as small as possible so that we can capture higher resolution images of the faces. In Figure 3-14, we illustrate the selection of FOV that offers the best resolution in the capture of all three persons' faces. Here, we check every pair of targets and find that the pair of Person 1 and Person 3 determine the minimal FOV for the camera. In this case, there are three possible combinations: the pair of Person 1 and Person 2, the pair of Person 1 and Person 3, and the pair of Person 2 and Person 3. Only the FOV formed by Person 1 and Person 3 can cover all these three people and it is the minimal FOV we want to have for this camera.

The FOV of each pair of targets can be easily estimated. We can calculate the camera-to-person vector first by the positions of the camera and person. Then, we utilize the inner product and the triangular function to calculate the FOV for that pair.

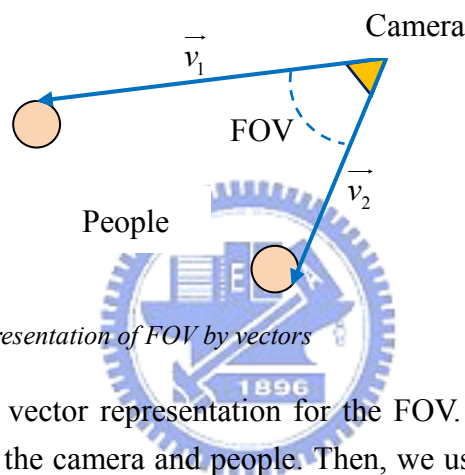


Figure 3-15 Representation of FOV by vectors

Figure 3-15 shows the vector representation for the FOV. We can calculate \vec{v}_1 and \vec{v}_2 by the positions of the camera and people. Then, we use Eq. 3-14 to estimate the FOV of the pair of people.

$$\text{FOV} = \text{acos} \left(\frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \cdot \|\vec{v}_2\|} \right) \quad \text{Eq. 3-14}$$

After the FOV is determined based on the best combination of target pair, based on the above process, we use angle bisector theorem to decide the direction in which the camera should focus on. Figure 3-16 illustrates an angle bisector of a triangle. The angle bisector theorem is that the angle bisector divides the opposite side into two parts whose ratio is the same as the ratio of the adjacent sides. We can formulate it by Eq. 3-15.

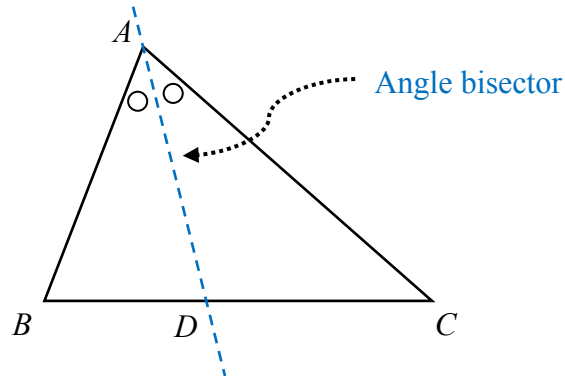


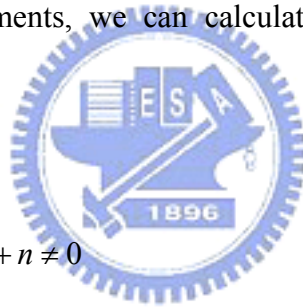
Figure 3-16 Angle bisector

$$\frac{\overline{AB}}{\overline{AC}} = \frac{\overline{BD}}{\overline{CD}} \quad \text{Eq. 3-15}$$

In addition, if we have a point that divides a line into two segments and we know the length ratio of the two segments, we can calculate the coordinate of that point according to Eq. 3-16:

$$\frac{\overline{BD}}{\overline{CD}} = \frac{m}{n} \quad \text{Eq. 3-16}$$

$$P_D = \frac{mP_C + nP_B}{m+n}, \quad m+n \neq 0$$



where P_B and P_C are the end point coordinates of the line segment and P_D represents the coordinate of the point of division, as illustrated in Figure 3-17.

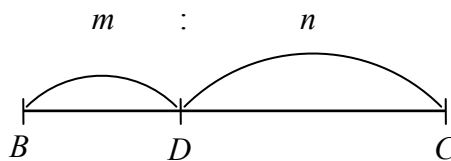


Figure 3-17 Internal point of division of a line segment

There is an analogy between Figure 3-15 and Figure 3-16. We can think that point A corresponds to the camera position while point B and C correspond to the people's positions. Obviously, the angle bisector corresponds to the optical axis of the camera. In order to make the optical axis coincide with the angle bisector, the camera should directly focus on the intersection point of the optical axis and the line through the two people. In other words, that point should project on the image center (or principal

point). This intersection point can be calculated by Eq. 3-15 and Eq. 3-16, according to the positions of the camera and people.

3.5. CAMERA CONTROL

After we know the focusing direction and the FOV of cameras, we can adjust the PTZ parameters to capture people's faces. We adjust the PTZ parameters based on a reference coordinate system. Figure 3-18 illustrates the pan and tilt angles of a PTZ camera with reference to an axis and a plane, respectively. The camera center is at the origin of the coordination system. The absolute pan angle is defined with respect to the X axis and the tilt angle is defined with respect to the X-Y plane.

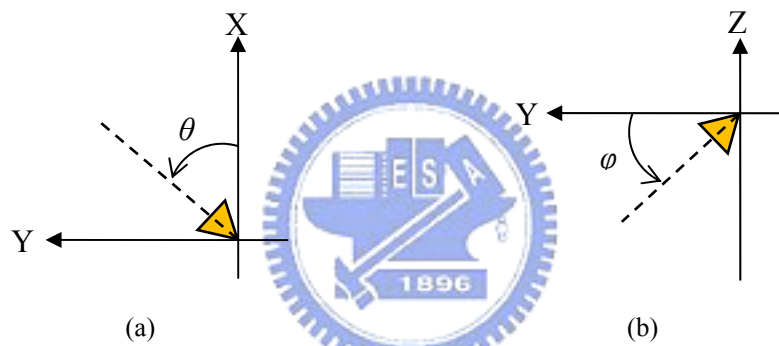


Figure 3-18 (a) pan and (b) tilt angles with reference to the reference coordinate system

In general, the PTZ cameras can adjust its pan and tilt angles with reference to some reference axes or planes. We only need to calculate the pan and tilt angles with regard to the references axis or plane to control the PTZ cameras. Here we illustrate the pan and tilt angle together in Figure 3-19. O is the origin of this coordinate system and is also the camera center. P is the focused point and P' is the point that P projects on the XY plane. The pan angle, θ , is determined by the angle between positive x-axis and $\overline{OP'}$. Because we can get the coordinates of P', the pan angle can be calculated by Eq. 3-17:

$$\theta = \text{atan}\left(\frac{y_{p'}}{x_{p'}}\right) \tag{Eq. 3-17}$$

where $x_{p'}$ and $y_{p'}$ are the x and y coordinates of point P', respectively. Because the x and y coordinates of the point P' are the same with P, we can rewrite Eq. 3-17 as Eq. 3-18:

$$\theta = \text{atan}\left(\frac{y_p}{x_p}\right) \quad \text{Eq. 3-18}$$

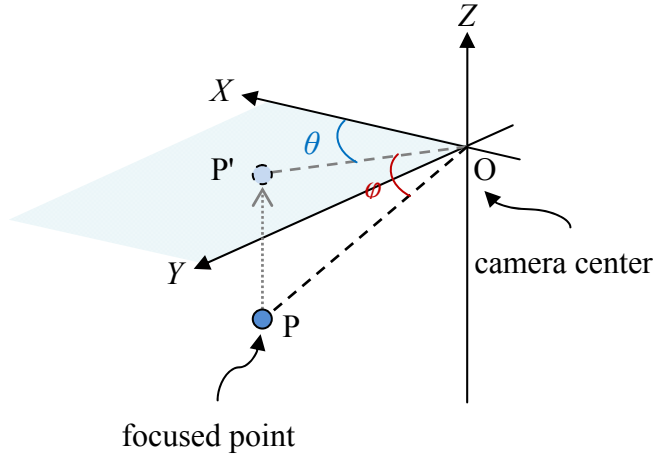


Figure 3-19 Illustration of pan and tilt angles

On the other hand, the tilt angle, φ , is determined by the angle between the X-Y plane and \overline{OP} . In other words, φ is the angle between \overline{OP} and $\overline{OP'}$. Hence, the tilt angle can be obtained by Eq. 3-19:

$$\varphi = \text{atan}\left(\frac{|z_p|}{\sqrt{x_{p'} + y_{p'}}}\right) \quad \text{Eq. 3-19}$$

where $x_{p'}$ and $y_{p'}$ are the x and y coordinates of the point P' and z_p is the z coordinate of point P. We can also rewrite Eq. 3-19 because the x and y coordinates of point P' are the same with P. Eq. 3-20 states the tilt angle in terms of the coordinates of P:

$$\varphi = \text{atan}\left(\frac{|z_p|}{\sqrt{x_p + y_p}}\right) \quad \text{Eq. 3-20}$$

where x_p , y_p , and z_p are the x, y, and z coordinates of the point P, respectively.

Eq. 3-18 and Eq. 3-20 are based on the coordinates of the point P, with the origin of the coordinate system being the camera center. We have to convert the coordinates of point P into the coordinates with respect to the camera center before we calculate the pan and tilt angles of the camera. So far, we have determined the pan and tilt parameters of the PTZ camera. The zoom parameter will be discussed below.

In general, the PTZ camera seldom sets the FOV directly. On the contrary, it often sets the focal length to adjust its FOV. Therefore, we need to convert FOV into

the focal length for the control of the zoom parameter. The formulas that convert the FOV of a rectilinear lens camera to the focal lengths are stated in following equations:

$$f_x = \frac{\text{image width}}{2 \cdot \tan\left(\frac{\text{horizontal FOV}}{2}\right)} \quad \text{Eq. 3-21}$$

$$f_y = \frac{\text{image height}}{2 \cdot \tan\left(\frac{\text{vertical FOV}}{2}\right)} \quad \text{Eq. 3-22}$$

where f_x and f_y are the horizontal and vertical focal lengths, respectively. According to these equations and the FOV obtained in Section 3.4, we can calculate the focal length of each PTZ camera.

3.6. ASSIGNMENT HOLD

However, there is usually a time lag before a PTZ camera moves to its destination. If we change the camera parameters too rapidly, the cameras might not correctly move to the destinations we want. For example, a PTZ camera may be requested to pan right at the n -th frame but pan left at the $(n+1)$ -th frame. However, since the PTZ camera needs a time lag to move to the destination. The camera does not actually move to the destination of the n -th frame but instead starts to change the direction at the $(n+1)$ -th frame. A quick change of commands may not only cause failures in target tracking but also cause unpleasant observation in target tracking.

A too-fast swing of camera's pose results from a quick change of the camera assignment. In other words, it is caused by the fast variations of the optimal results. Hence, in our system, we intentionally hold the result of optimization over a number of frames so that the moving states of the PTZ cameras become more consistent and more smooth over successive frames. Figure 3-20 shows the block diagram of the camera adjustment. Holding the result of optimization only means that the cameras keep taking charge of the same people within the holding frames. Camera assignment still needs to be carried out at each frame because people are still moving and their positions are changing. Although the holding procedure might cause some errors, the errors are under control because typically people don't have abrupt change in movement during a short period. Of course, some exceptions are unavoidable and from time to time some people's faces might not be captured well. When this kind of situation happens, the mechanism of significance weight starts to function. The details of the significance weights have discussed in Section 3.2.

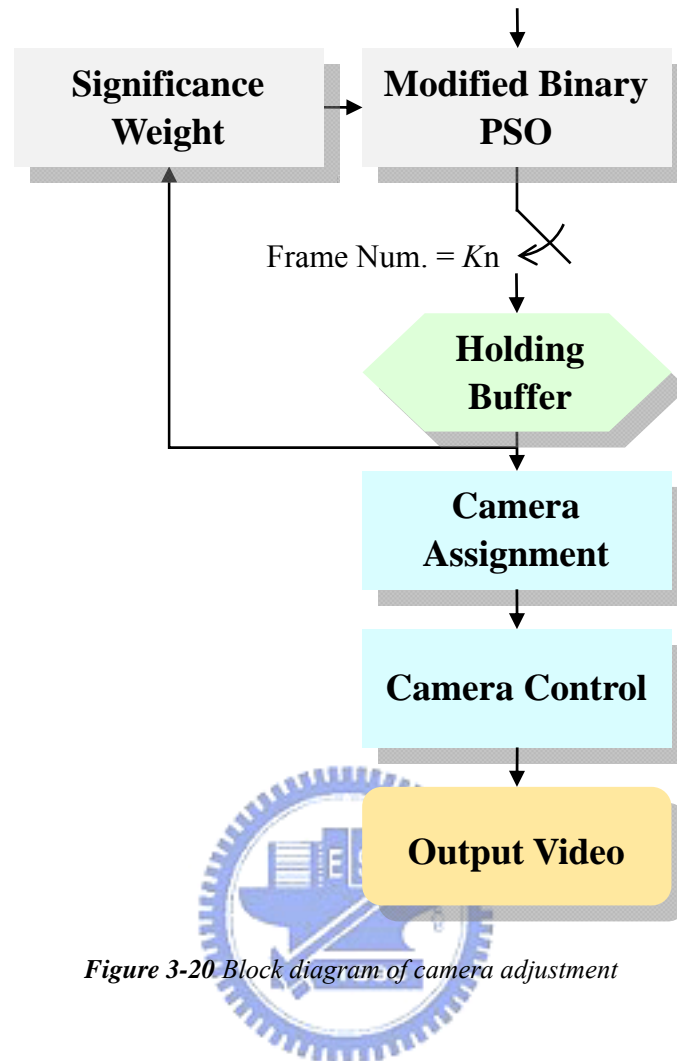


Figure 3-20 Block diagram of camera adjustment

3.7. OVERALL COORDINATION SYSTEM

To sum up, we show the overall block diagram of the proposed coordination system in Figure 3-21. There are two main parts in the system – optimization and camera adjustment. In the optimization stage, we try to find out an optimal solution to capture as many frontal high-resolution people’s faces as possible in the scene. Here, we design an evaluation function to achieve this goal and simultaneously avoid some people from being unobserved for a long time. We find the best coordination by using the modified discrete binary particle swarm optimization. Based on the coordination, we control all the PTZ cameras on the camera adjustment stage. The pan, tilt, and zoom parameters are calculated according to the result of modified DBPSO and the PTZ cameras move to the assigned destinations to capture targets’ facial image. The holding process is also added to avoid too-fast swings. .

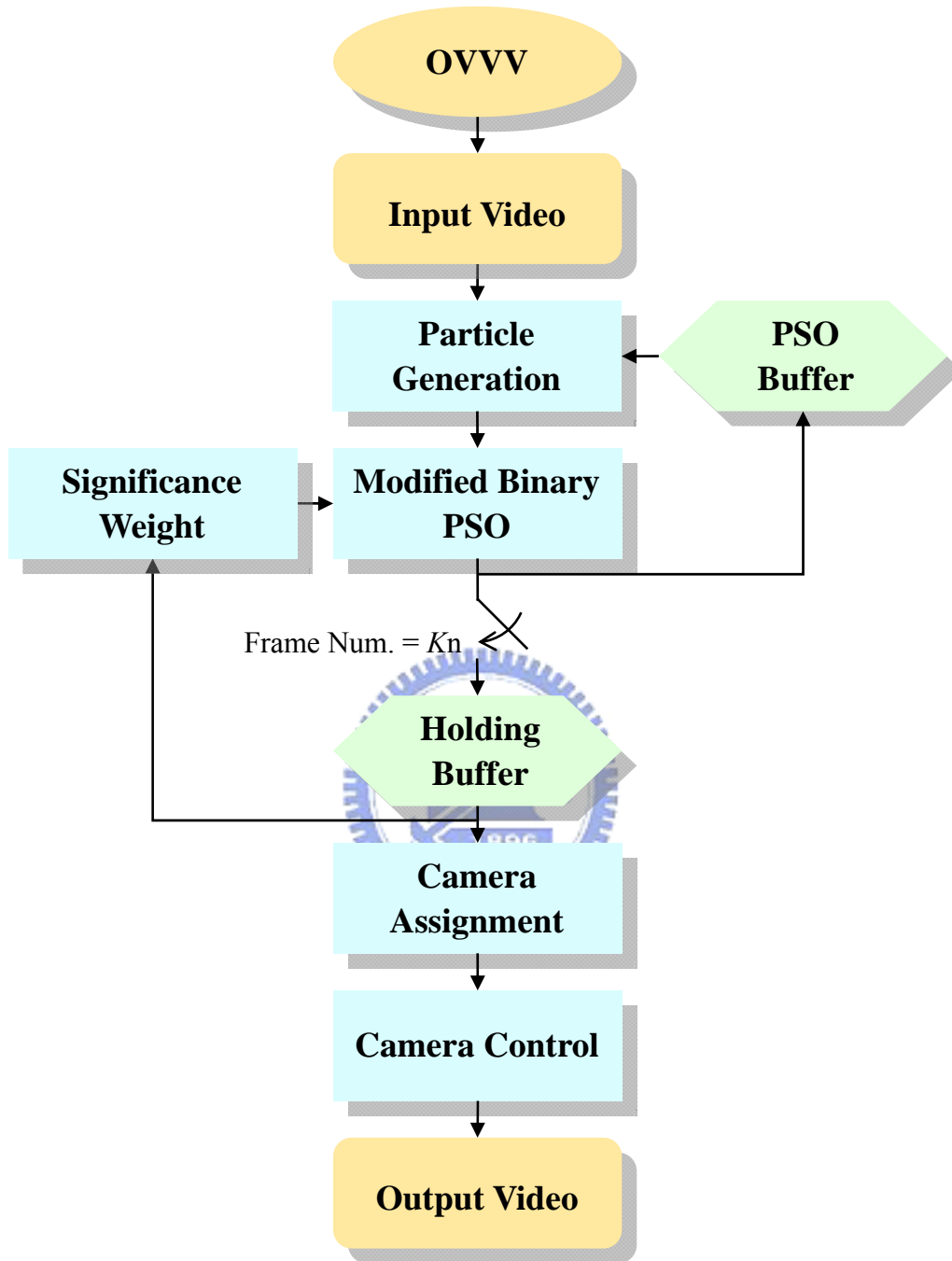


Figure 3-21 Overall block diagram of the proposed coordination system

Chapter 4.

SIMULATIONS

In this chapter, we will show and discuss our experimental results. We utilize OVVV to simulate and evaluate the proposed coordination system. We create some indoor environments where a few people are moving around. We use four PTZ cameras, which are mounted on the ceiling in the four corners of the room, to simulate the setup of the proposed coordination system. Figure 4-1 illustrates the installation of these four PTZ cameras in the room. These PTZ cameras are calibrated and we know their 3D positions. Moreover, the camera ground truth provided by OVVV is also used to help camera calibration.

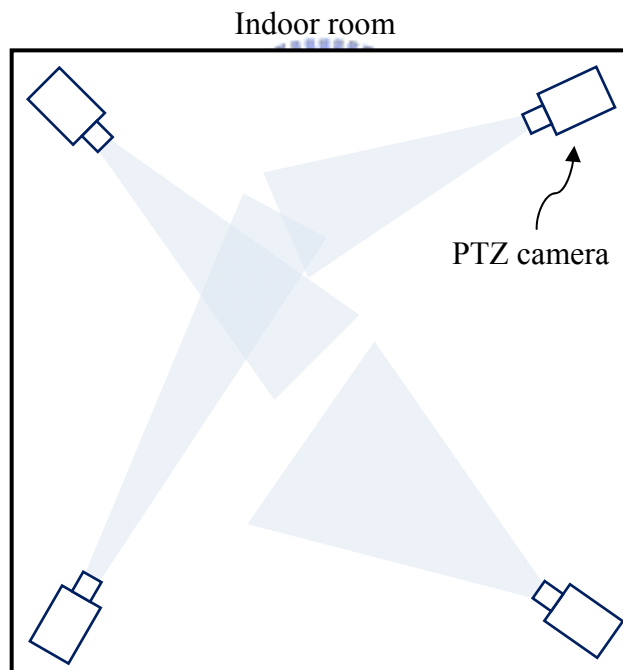
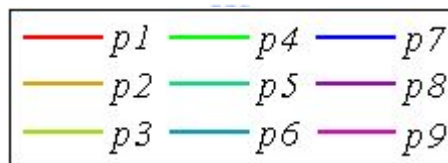


Figure 4-1 The installation of PTZ cameras

In this thesis, we mainly focus on the coordination of cameras. Here, we assume the tracking of people has already been done by some means. That is to say, we have already known each target's position in advance. Since the ground truth provided by OVVV contains people's 3D coordinates on the ground, we use it directly as the results of people tracking. However, since OVVV does not provide the ground truth about the 3-D position of people's faces, we simply define an average adult height and add this height to the people's ground positions to approximate the face positions. The

face directions are estimated by the two consecutive frames. We assume that people always look ahead so their facing direction can be estimated by two consecutive face positions. With the above assumptions, we are able to each target's face position and orientation, as illustrated in Figure 3-2.

Figure 4-2 shows the experiment results of the test sequence SEQ-1. In this sequence, people walk in groups initially. Then some people leave their partners and join another group. The first figure illustrates the color labeling for the 9 people in the scene. Each person is assigned a color and we use this color to plot a bounding box for that person. In the following figures, we show a few sets of images captured at different time instants. At each time instant, eight images are captured. They are captured by four static cameras and four PTZ cameras, locating at different positions. In each figure, the left four frames are captured by the static cameras, while the right four frames are captured by the PTZ cameras. The use of static cameras can help the reader to easily realize the relations among these nine people. On the other hand, the images captured by the PTZ cameras demonstrate the results of camera coordination.



Color Labels



Time 15



Time 35



Time 55



Time 65



Time 80



Time 95



Time 110



Time 120



Time 135

Figure 4-2 Experimental results of the test sequence SEQ-1

In Figure 4-2, the person $p8$ is not captured well at Time 65 because of the tradeoff in optimization, as mentioned in Section 3.2. In fact, this situation started at Time 61, where the score of $p8$ drops to zero, as shown in Figure 4-3. In Figure 4-3, we show all people's score curves from Time 1 to Time 150, with the best score being 10 while the worst one being 0. Moreover, in this simulation, we set the holding time to 10. That is, we always hold the camera assignment for 10 frames to see whether there is any chance that the observation may get improved. Unfortunately, for this case, the observation of $p8$ is not improved during the following 10 time instants. Hence, at Time 71, a new assignment is ignited and the score of $p8$ gets raised afterward. Please note that the significance weight of $p8$ keeps increasing during the period from Time 61 to Time 70. Hence, at Time 71, $p8$ has already accumulated a large value of significance weight and this offers $p8$ a very high priority in camera assignment.

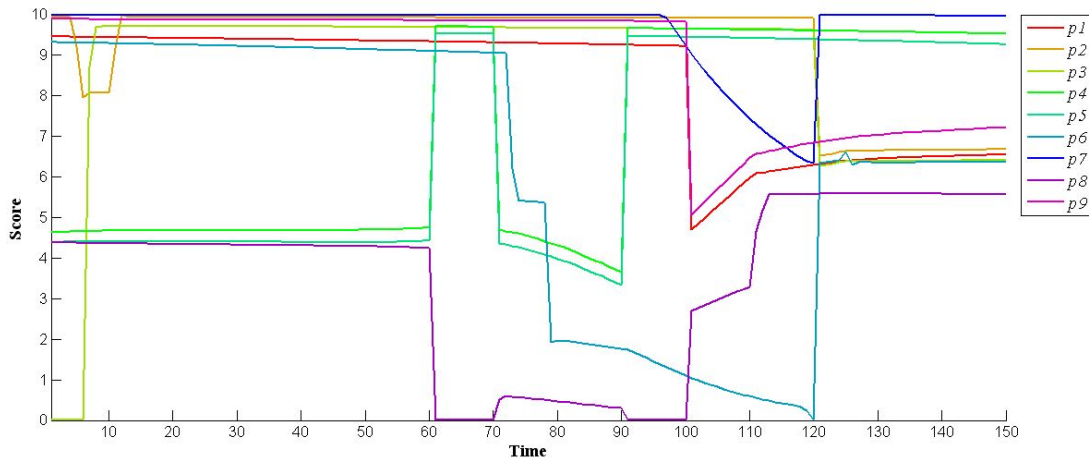


Figure 4-3 All people's score curves of Sequence SEQ-1

The variations of weights are illustrated in Figure 4-4. The weight of $p8$ is raised starting from Time 61. The significance weight is switched to the holding state at Time 71 until the face of $p8$ can be captured. After his face can be well captured, the weight diminishes to zero at Time 84.

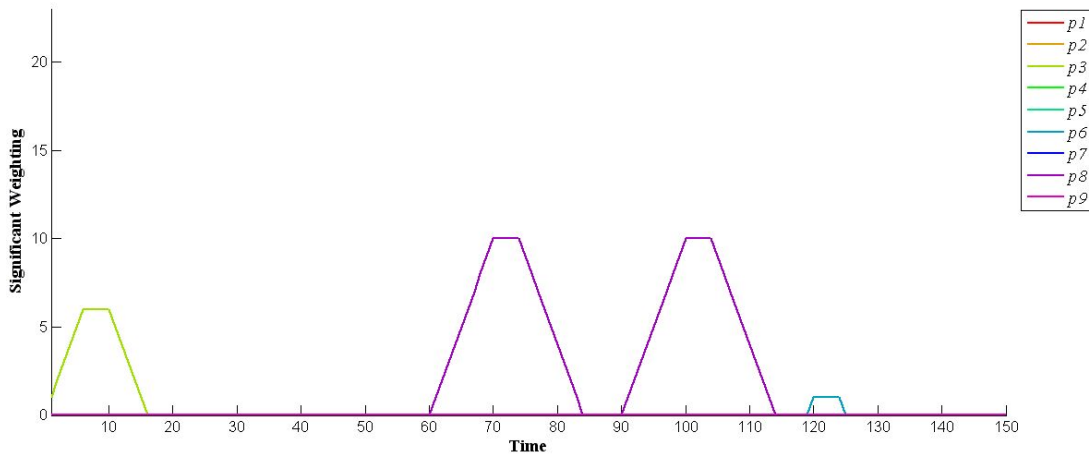


Figure 4-4 All people's significance weights of Sequence SEQ-1

If we do not apply the concept of significance weight to the coordination system, the face of $p8$ will not be seen for a long time. Figure 4-5 shows $p8$'s score curve without using the significance weight. We can see that $p8$ is not well captured during a long period starting from Time 51 till Time 120. During this period, he gets only zero score and his face is not observed at all. If he happens to stay in this scene only during this period, then we'll never be able to see his face. However, with the use of significance weight, the period of invisibility can be shortened and the probability of miss can be lowered.

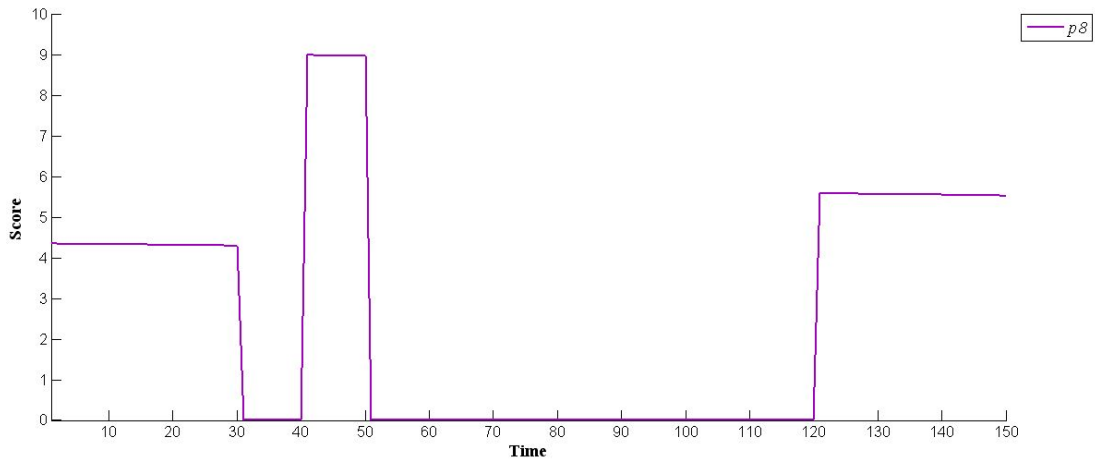


Figure 4-5 Person $p8$'s score curve of Sequence SEQ-1 without applying significance weight

In this thesis, we set the threshold of tolerable period to 20 frames. That is, as long as the score value of a target has been continuously zero for 20 successive frames, we'll assign that target a very high priority so that our system will try to capture a clear picture of that target as soon as possible. Here, we adopt the mechanism mentioned in Section 3.2.2. In order to verify the functionality of this mechanism, we intentionally slow down the increment rate of the significance weight. In this case, the significance weight of the unobserved target does not grow fast enough and the period of being unobserved can easily exceed 20 frames. On the other hand, the weight threshold is set to be 20. As shown in Figure 4-6 and Figure 4-7, the value of weight is dramatically raised to 30 at Time 75 because the weight threshold is reached. The value of weight keeps rising until Time 80 since the holding buffer changes the assignment at Time 81. On the other hand, the value of weight is also dramatically raised to 30 at Time 111 because the unobserved time period has reached the threshold of tolerable period.

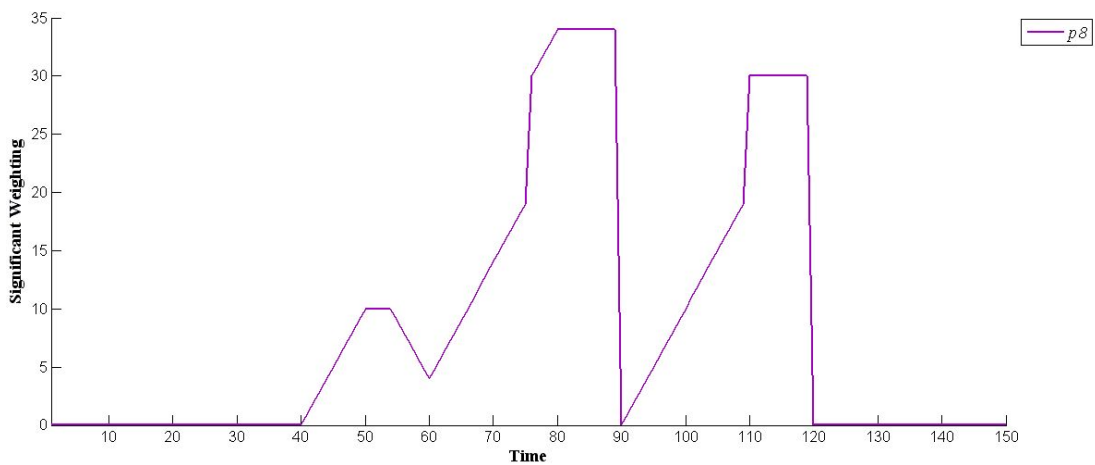


Figure 4-6 Illustration of the mechanism of unclear limitation

Figure 4-7 shows the corresponding score curve of Figure 4-6. The person gets out of the zero score at Time 80 and keeps non-zero score for 10 successive time units.

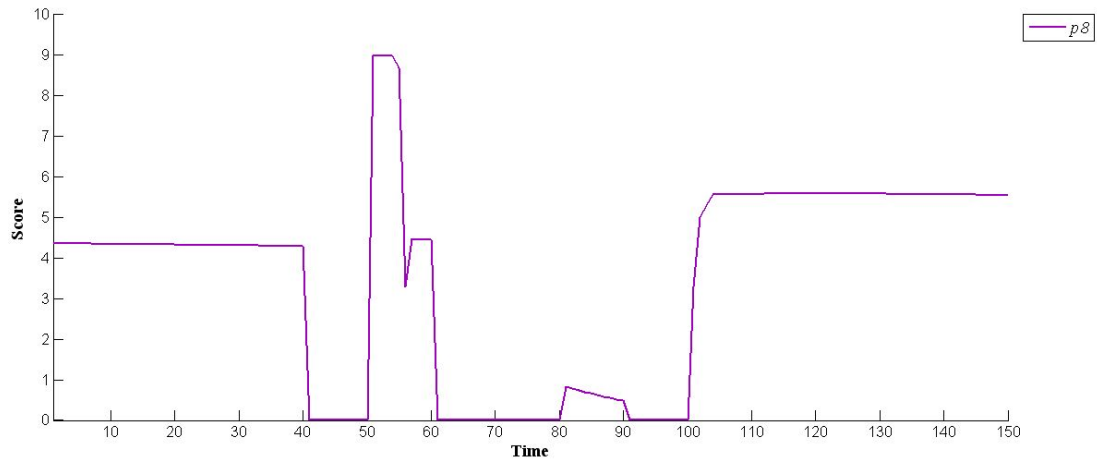
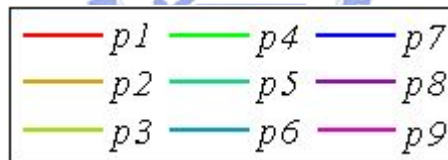


Figure 4-7 The corresponding score curve of Figure 4-6

Figure 4-8 shows the experimental results of the sequence SEQ-2. In SEQ-2, there are nine people in the scene. Different from SEQ-1, the people in SEQ-2 do not walk in group. They all walk on different routes with different directions. It is designed to simulate random walking people.



Legend



Time 15



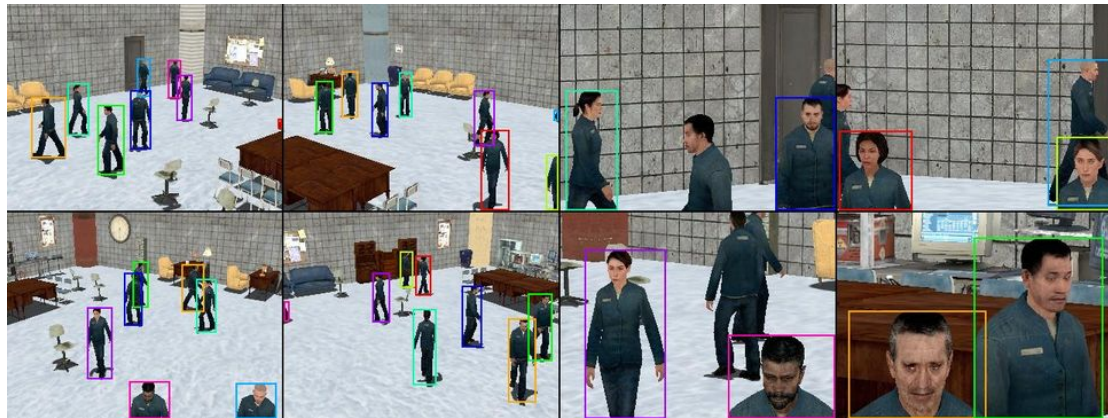
Time 65



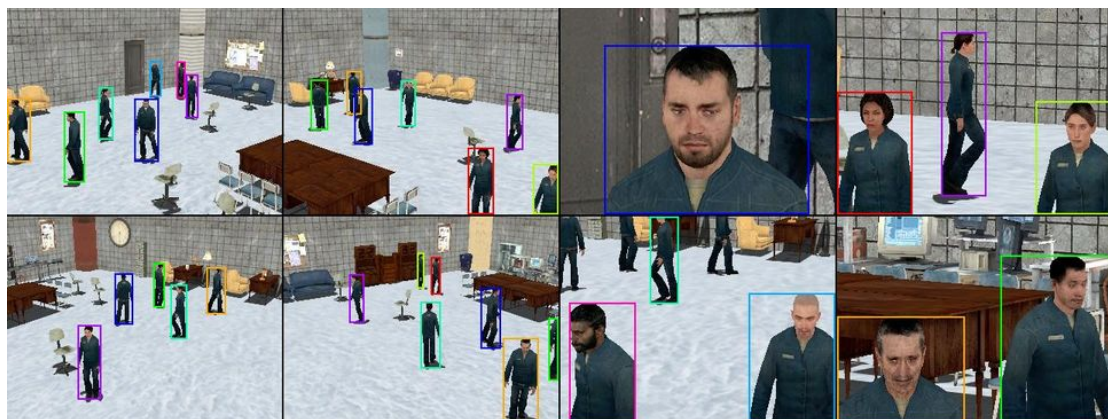
Time 95



Time 105



Time 125



Time 150

Figure 4-8 Experimental results of the test sequence SEQ-2

Figure 4-9 shows all people's score curves of the sequence SEQ-2. We can apparently see some differences between the score curves of SEQ-1 and SEQ-2. The people's scores of SEQ-1 are mostly higher than five while the scores of SEQ-2 are not. This is because the people in SEQ-2 walk in chaos. However, since we want to capture as many frontal people's faces as possible at each time instant, some people may not be well captured. Although the results are not as satisfactory as that of SEQ-1, each target's score is still higher than five over a certain period of time. This means we have clearly captured the frontal faces of all targets for a period of time. This will help us in recognizing the identity of each person in the monitored scene.

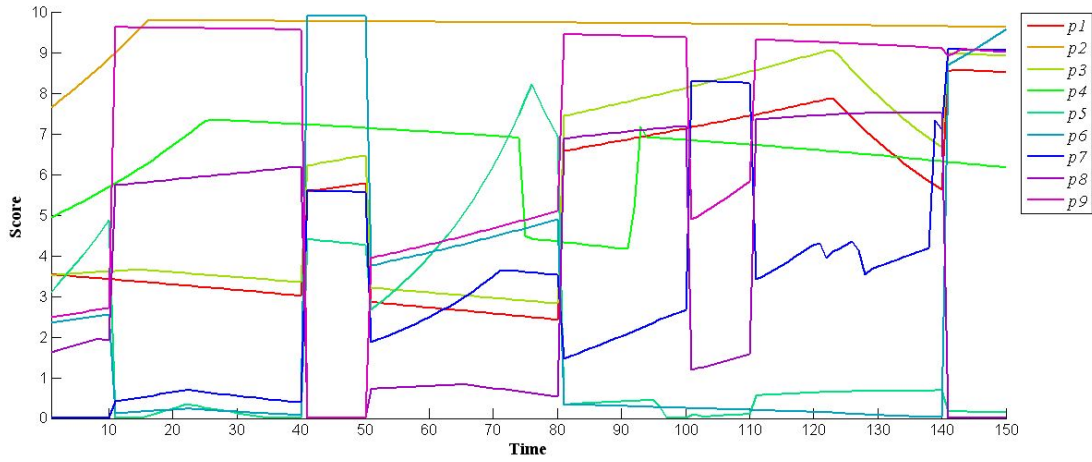
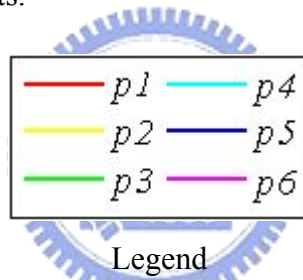


Figure 4-9 All people's score curves of the sequence SEQ-2

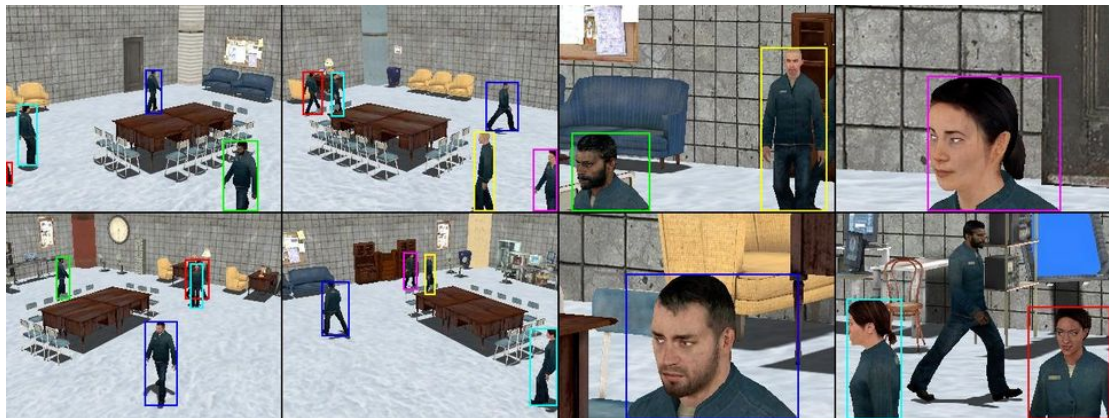
Figure 4-10 shows the experimental results of the sequence SEQ-3. There are six people in SEQ-3. They walk around a table, which can be seen by the static cameras. They do not walk in groups. Some people sometimes walk closely but sometimes walk alone. SEQ-3 is used to simulate the scenario that people walk around some obstacles, like tables or cabinets.



Time 50



Time 90



Time 120



Time 130



Time 170



Time 280

Figure 4-10 Experimental results of the test sequence SEQ-3

Figure 4-11 shows all people's score curves of Sequence SEQ-3. The scores are high at most of the time. However, we can notice that people's scores go down between Time 110 and Time 160. This is because people are changing their directions around that time. The results show that people are clearly captured except the instants of changing directions.

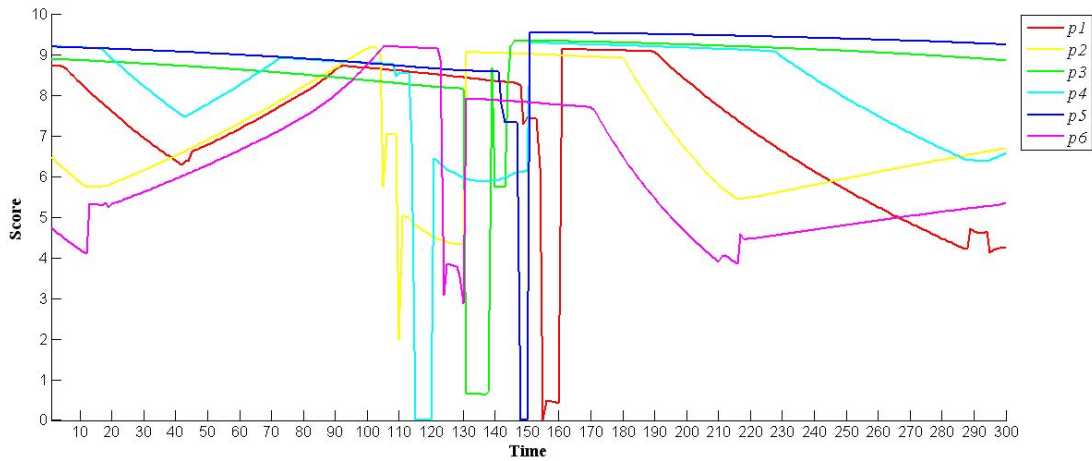
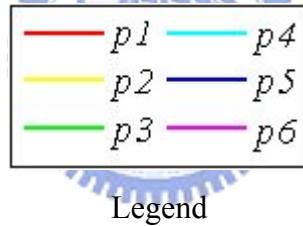


Figure 4-11 All people's score curves of sequence SEQ-3

Because the people in the sequence SEQ-3 walk around a big table, they keep similar conditions for a while. Hence, the PTZ cameras do not need to change their states rapidly. In SEQ-4, we test a rapid changing case. There are six people in SEQ-4 and they only walk around a chair within a small region. They do not walk in certain kinds of groups, either. Figure 4-12 shows the experimental results of SEQ-4. We can notice that the PTZ cameras switch to take charge of different people more frequently.



Time 25



Time 55



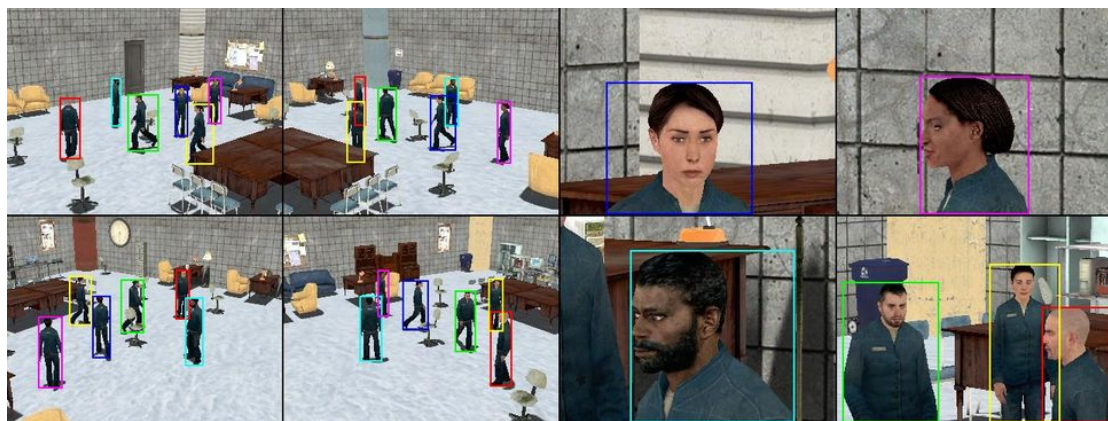
Time 115



Time 145



Time 225



Time 280

Figure 4-12 Experimental results of the test sequence SEQ-4

Figure 4-13 shows all people's score curves of SEQ-4. The chances that people's scores go down in a short period of time are increased apparently. The status of "going down" is similar to that in SEQ-3 (see Figure 4-11) but the frequency of "going down" is much higher in SEQ-4.

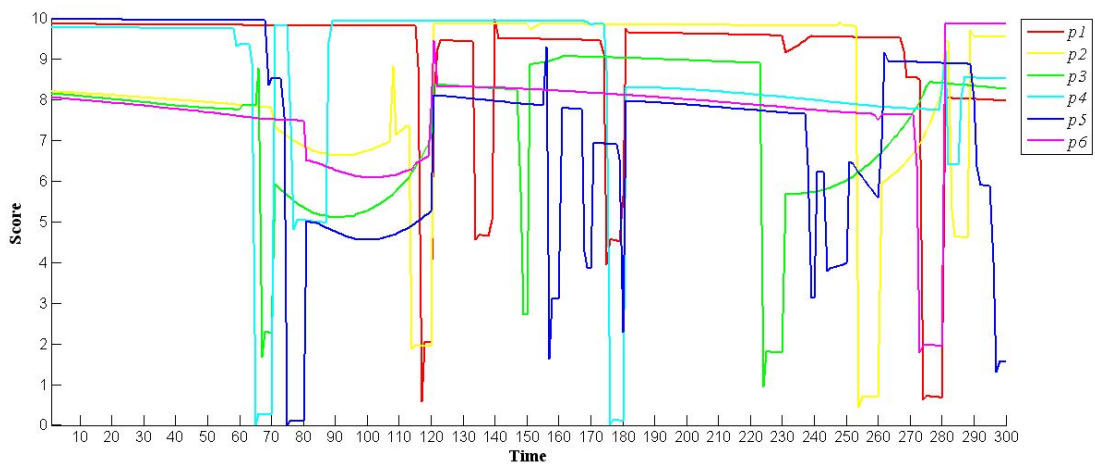









Figure 4-13 All people's score curves of Sequence SEQ-4

Figure 4-14 shows the experimental results of the sequence SEQ-5. There are seven people in the scene of SEQ-5. SEQ-5 is somewhat similar to SEQ-1. The people also walk in groups in SEQ-5. However, the people in SEQ-5 meet together and change partners more frequently.

	<i>p1</i>		<i>p5</i>
	<i>p2</i>		<i>p6</i>
	<i>p3</i>		<i>p7</i>
	<i>p4</i>		

Legend



Time 75



Time 125



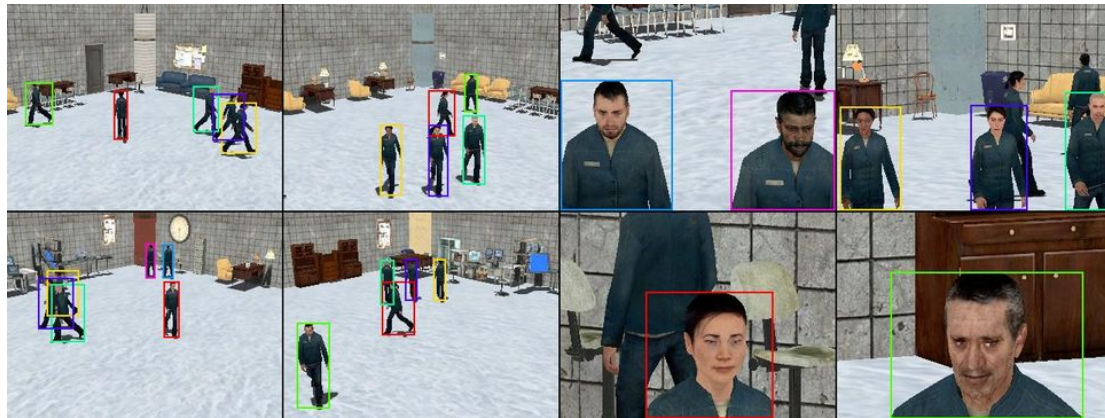
Time 195



Time 240



Time 250



Time 275

Figure 4-14 Experimental results of the test sequence SEQ-5

Figure 4-15 shows all people's score curves of SEQ-5. Compared to Figure 4-3, we notice that the number of people whose scores go down are increased. Nevertheless, these scores still keep high at most of the time.

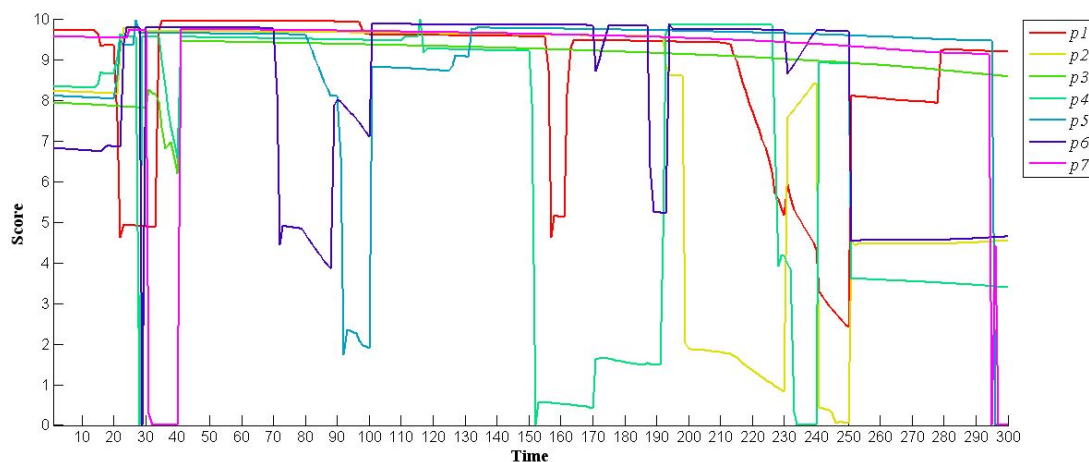


Figure 4-15 All people's score curves of Sequence SEQ-5

Table 4-1 shows the statistical results of all experimental sequences. The “average score” is the average score for all people in a sequence. The “unclear rate” is the average ratio of the zero-score time units over the total time units. The highest average score is 10 and the lowest is 0. Here, the dimension of each image frame is 320×240 for all experiments. We set the value of th_{max} , th_{min} , and th_{θ} to be 50, 15, and $\pi/2$, respectively. The upper bound of the “unclear period” is set to 20.

According to the average score of all sequences, we can approximate the average face width in the image and the average shoot angle with respect to the camera. The average face width is 45 pixels and the average shoot angle is 13 degrees. According to the average unclear rate, the probability that we cannot identify a target is about 1.61%.

Table 4-1 Statistical results of all experimental sequences

	Average Score (min, max) = (1, 10)	Unclear Rate (%)
SEQ-1 (9)	7.5355	2.00
SEQ-2 (9)	5.0644	3.93
SEQ-3 (6)	7.6656	0.56
SEQ-4 (6)	8.0526	0.17
SEQ-5 (7)	8.3151	1.38
Average	7.3266	1.61

We compare the performance of the proposed modified DBPSO with that of the original DBPSO. The performance is tested based on the same sequences. The optimization processes are implemented with 30 particles and the processes stop after 30 iterations. The results are compared in Table 4-2. In Table 4-2, 25 percents of the results generated by the proposed modified DBPSO are better than that of the original one. Only 1.4 percents of the results are worse than that of the original DBPSO. In Table 4-3, we shorten the iteration number from 30 to 20. It can be seen that as the number of iterations decreases, the proposed modified DBPSO method performs even better.

Table 4-2 Comparison between the modified DBPSO and the original DBPSO
(30 iteration)

	Better (%)	Equal (%)	Worse (%)
SEQ-1 (9)	55.00	42.50	2.50
SEQ-2 (9)	58.50	37.00	4.50
SEQ-3 (6)	1.39	98.61	0.00
SEQ-4 (6)	0.00	1.00	0.00
SEQ-5 (7)	12.22	87.78	0.00
Average	25.42	73.18	1.40

Table 4-3 Comparison between the modified DBPSO and the original DBPSO
(20 iteration)

	Better (%)	Equal (%)	Worse (%)
SEQ-1 (9)	76.00	24.00	0.00
SEQ-2 (9)	82.00	11.50	6.50
SEQ-3 (6)	8.89	91.11	0.00
SEQ-4 (6)	3.06	96.94	0.00
SEQ-5 (7)	23.75	76.25	0.00
Average	38.74	59.96	1.30

Table 4-4 shows the comparison of iteration number between the original and the modified binary PSO. The iteration number of the modified DBPSO is the iterations executed until the result converges. We compare the number of iterations that can achieve the same results by using the original DBPSO. The results are shown in Table 4-4. Here we use 30 particles to carry out this comparison. According to the statistics, the modified binary PSO in average saves 92-percent iterations. We do the comparison again by using 20 particles and show the results in Table 4-5. Because of the decrease in particle number, more iterations are needed to find the optimal solution. In comparison, the proposed modified DBPSO only need slightly more iterations to find the optimum.

Table 4-4 Comparison of iteration numbers (30 particles)

	MDBPSO (Iteration)	DBPSO (Iteration)	Reduction (%)
SEQ-1 (9)	2.0000	40.2167	95.03
SEQ-2 (9)	2.0000	52.8778	96.22
SEQ-3 (6)	2.0000	9.7472	79.48
SEQ-4 (6)	2.0056	8.4138	76.23
SEQ-5 (7)	2.0028	16.0344	87.53
Average	2.0017	25.4580	92.14

Table 4-5 Comparison of iteration number (20 particles)

	MDBPSO (Iteration)	DBPSO (Iteration)	Reduction (%)
SEQ-1 (9)	2.0000	50.5444	96.04
SEQ-2 (9)	2.0050	65.2833	96.94
SEQ-3 (6)	2.0028	15.8583	87.39
SEQ-4 (6)	2.0000	12.7194	84.28
SEQ-5 (7)	2.0031	22.8781	91.26
Average	2.0022	33.4567	94.02



Chapter 5.

CONCLUSIONS

In this thesis, we construct a coordination system of PTZ cameras to control multiple PTZ cameras to capture as many frontal high-resolution facial images as possible. We coordinate multiple PTZ cameras to capture people's faces according to their face resolutions on the images and the directions of the frontal faces with respect to the cameras. We propose the significance weight for each person and the coordination is affected by these weightings. The significance weight is proposed because the situation that someone's face cannot be clearly captured is hardly avoided. The proposed system aims to control PTZ cameras to clearly capture those targets that haven't been clearly observed before. In the proposed system, we define the evaluation function according to the above factors. We then try to find an optimal solution for the evaluation function. In other words, we attempt to find the best coordination approach to capture people's faces. We utilize discrete binary particle swarm optimization technique to find the solution. Due to some requirements of our system, we further modify the DBPSO algorithm to improve the effectiveness. Finally, the PTZ parameters of PTZ cameras are adjusted according to the result of optimization.

Because different persons may have different demands, some parameters of the proposed coordination system can be adjusted by users. Users can define their own thresholds. For example, the threshold of face width in the images is adjustable. In addition, the proposed system does not limit the number of PTZ cameras and their placement. No matter how many PTZ cameras there are, we can still use the proposed algorithm to accomplish the task of camera coordination.

REFERENCES

- [1] C. Micheloni, G. L. Foresti and L. Snidaro, "A cooperative multicamera system for video-surveillance of parking lots," *IEE Symposium on Intelligence Distributed Surveillance Systems*, pp. 1-5, Feb. 2003.
- [2] C. Micheloni, G. L. Foresti and L. Snidaro, "A network of co-operative cameras for visual surveillance," *IEE Proceedings on Vision, Image and Signal Processing*, vol. 152, pp. 205-212, April 2005.
- [3] Nyoun Kim, Ig-jae Kim and Hyoung-gon Kim, "Video Surveillance Using Dynamic Configuration of Multiple Active Cameras," *IEEE International Conference on Image Processing*, pp. 1761-1764, Oct. 2006.
- [4] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti, "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking," *IEEE Signal Processing Magazine*, vol. 22, pp. 38-51, Mar. 2005.
- [5] A. Khiat, S. Yous, T. Ogasawara and M. Kidode, "Combining Fixed Stereo and Active Monocular Cameras into a Platform for Security Applications," *IEEE Int. Conf. on Robotics and Biomimetics*, pp. 1134-1139, Dec. 2006.
- [6] Faisal Z. Qureshi and Demetri Terzopoulos, "Surveillance Camera Scheduling: A Virtual Vision Approach," *Multimedia Systems*, vol. 12, pp. 269-283, Dec. 2006.
- [7] Faisal Z. Qureshi and Demetri Terzopoulos, "Surveillance in Virtual Reality: System Design and Multi-Camera Control," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, Jun. 2007.
- [8] J. B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of 5-th Berkeley Symposium on Mathematical Statics and Probability*, Berkeley, University of California Press, pp. 281-297, 1967
- [9] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, vol. 3, pp. 32-57, 1973.
- [10] S. C. Johnson, "Hierarchical Clustering Schemes," *Psychometrika*, vol. 2, pp. 241-254, 1967.
- [11] Edwin K.P. Chong and Stanislaw H. Zak, "An Introduction to Optimization, second edition," *Wiley Press*, Jul. 2001.
- [12] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, 1995.
- [13] J. Kennedy, R. C. Eberhart and Y. Shi, "Swarm Intelligence," *Morgan Kaufmann*

Academic Press, 2001.

- [14] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, pp. 4104-4109, Oct. 1997.
- [15] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, Dec. 1997.
- [16] S. Das and A. Abraham and Amit Konar, "Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives," *Advances of Computational Intelligence in Industrial Systems*, Studies in Computational Intelligence, pp. 1-38, 2008
- [17] F. Qureshi and D. Terzopoulos, "Towards Intelligent Camera Networks: A Virtual Vision Approach," *In Proc. VSPETS 05*, pp. 177-184, 2005.
- [18] X. Desurmont, J-B. Hayet, C. Machy, J-F. Delaigle and J-F. Macq, "On the performance evaluation of tracking systems using multiple pan-tilt-zoom cameras," *Videometrics IX, part of the IS&T/SPIE Symposium on Electronic Imaging 2007*, pp. 28-30, Jan. 2007.
- [19] G. R. Taylor, A. J. Chosak, and P. C. Brewer, "OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, June 2007.

