Robust and Consistent Human Tracking Within Camera and Across Multiple Cameras

Chun-Te Chu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2013

Reading Committee:
Jenq-Neng Hwang, Chair
Zicheng Liu
Linda Shapiro

Program Authorized to Offer Degree:
Department of Electrical Engineering

University of Washington

Abstract

Robust and Consistent Human Tracking Within Camera and Across Multiple Cameras

Chun-Te Chu

Chairperson of the Supervisory Committee:
Professor Jenq-Neng Hwang
Department of Electrical Engineering

This dissertation strives to develop a robust and consistent tracking system that tracks humans within a single camera and across multiple cameras. We present three main parts of our research: single camera tracking, multiple-camera tracking with overlapping views and multiple-camera tracking with nonoverlapping views.

In single camera tracking, we focus on solving the occlusion problem in order to perform reliable tracking within a camera's view. We present an innovative method, which uses projected gradient to facilitate multiple inter-related kernels, in finding the best match during tracking under predefined constraints. The adaptive weights are applied to the kernels in order to compensate for the adverse effect introduced by occlusion. An effective scheme is also incorporated to deal with the scale change issue during the tracking. We construct the single camera tracking system by embedding the multiple-kernel tracking into a Kalman filtering-based tracking module. Several simulation results have been done to show the robustness of the proposed multiple-kernel tracking and also demonstrate that the overall system can successfully track the targets under occlusion.

For multiple-camera tracking, we aim to reduce as much of the manual work as possible while providing the consistent tracking across multiple cameras. We first introduce our approach for tracking human across multiple cameras with overlapping

views. The camera link model, including homography matrix, brightness transfer function and tangent transfer function, are estimated automatically given the cameras' views. Vicinity cue, color cue and edge cue are considered while matching the people across cameras. When the cameras have nonoverlapping field of views, we propose an unsupervised learning scheme to build the camera link model, including transition time distribution, brightness transfer function, region mapping matrix, region matching weights, and feature fusion weights. Our unsupervised learning scheme tolerates well the presence of outliers in the training data. The systematic integration of multiple features enables us to perform an effective re-identification across cameras. The pairwise learning and tracking manner also enhances the scalability of the system. Several experiments and comparative studies demonstrate the effectiveness of our proposed method, and the complete system has been tested in a real-world camera network scenario.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my sincere appreciation to my advisor, Professor Jenq-Neng Hwang, for his guidance and support in my PhD study. He has always been supportive and given me the freedom to try various ideas. He has helped me keep pursuing my research goal and given me advices whenever I encountered difficulties.

I would like to thank the members in my PhD committee, Professor Mark Ganter, Professor Zicheng Liu, Professor Shwetak Patel, and Professor Linda Shapiro for their valuable suggestions and career advices.

Many thanks to Professor Mohamed El-Sharkawi and Professor Maya Gupta for providing the insightful discussions.

I am grateful to my colleagues and lab mates, Hsu-Yung Cheng, Chih-Wei Huang, Victor Gau, Po-Han Wu, and Shian-Ru Ke, who have provided huge help and friendship to me, especially during my first year. Without their kindness and great help, I could not settle down my life in US and focus on my research.

I thank all the present members in Information Processing Lab: Xiang Chen, Meng-Che Chuang, Shian-Ru Ke, Kuan-Hui Lee, Pei-An Lee, Youngdae Lee, Yongjin Lee, Ruizhi Sun, and Po-Han Wu. They have made the lab a great working place and provided precious suggestions to my research. Best wishes to all of them.

I also thank all my friends whom I have met in Seattle. It is they who provide great friendship so that I feel energetic in my life.

I would like to thank Jung-Sheng (Debbie) Su. She has always been around me during my good and bad times. I will never forget the valuable suggestion and encouragement she has given to me.

I will forever be thankful to my parents and brother. They have been supportive during my PhD study. Thanks to their understanding and constant encouragement, I am able to pursue the degree without worries. I clearly know they will always be there whenever I need help and suggestion. I am the luckiest person because of my dear family.

Dedication

To my family

Chapter 1 – Introduction

1.1 Introduction

Recently, the development of intelligent surveillance systems has attracted much of people's attention. Human tracking is one of the major tasks of video surveillance systems. In order to analyze the behavior of a specific person, the target's position needs to be correctly located in consecutive frames. Several approaches have been proposed to deal with the tracking problems [80]. While applying tracking to the real world scenarios, there are several sub-problems, such as occlusion, cluttering, fast motion, etc [24]. Among these sub-problems, occlusion is one of the most challenging and common problems in the real world scenarios. When the occlusion occurs, the relevant information of the target in the frame may be insufficient to locate the target, thus the tracker tends to lose the target. A surveillance system will never be reliable without having the ability to deal with occlusion.

Due to the limited field of view (FOV) of a camera, a target's information is no longer available once the target leaves the view of the camera. Hence, a surveillance system is required to have multiple networked cameras covering a range of areas, such as in airport, train station, and shopping mall, and the system needs to track humans across multiple cameras; that is, it is necessary to know while a person appears in one view, whether it has already shown up in another view or not. The ultimate goal is to solve the correspondence problem between the observed humans in different cameras. Normally, two different scenarios are considered separately when investigating multiple-camera tracking:

(1) Tracking across cameras with overlapping views: In this scenario, there is overlapping area between two cameras, like camera 1 and camera 2 in Fig. 1.1. Assume the cameras are synchronized, the person located in the overlapping area will appear

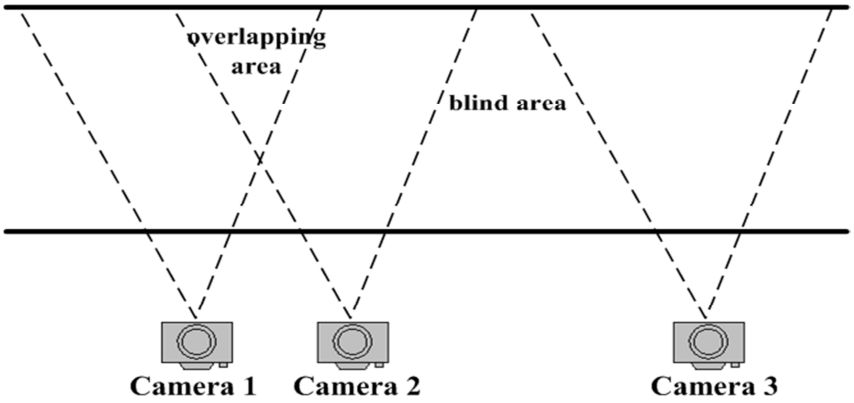


Figure 1.1: The topology of multiple cameras

simultaneously in two cameras. The calibration information is usually adopted for identifying the correspondence between the people observed in two cameras.

(2) Tracking across cameras with nonoverlapping views: When there is no overlapping area between two cameras, i.e., the views are totally disjointed like camera 2 and camera 3 in Fig. 1.1, the same person shows up at different time and locations. The variant lighting conditions, human poses, and viewing angles increase the difficulty of the re-identification.

This dissertation strives to construct a consistent tracking system that tracks humans within a single camera and across multiple cameras. In single camera tracking, the focus is on solving the occlusion problem in order to perform continuous tracking within the field of view of each camera. For the multiple-camera tracking, we aim to reduce as much of the manual work as possible while providing consistent tracking across multiple cameras. Specifically, unsupervised learning and updating schemes are applied to model the relationship, referred as *camera link model* in our work, between cameras without any human intervention. The pairwise learning and tracking manner enhances the scalability of the system. Several experiments and comparative studies based on public datasets and real-world scenarios demonstrate the effectiveness of our proposed system.

1.2 Contributions

The contributions of this dissertation are listed in the following:

- We propose an innovative method that uses projected gradient to facilitate multiple-kernel tracking in finding the best match under predefined constraints. Our formulation can be applicable to any cost function.
- Each kernel in our multiple-kernel tracking has an adaptive weight which enables the system to obtain better movement during optimum search, especially under occlusion situation. The by-product in tracking can be utilized to deal with the scale change of the target.
- The multiple-kernel tracking is combined with a Kalman filter to form a complete automatic tracking system.
- We utilize image registration technique to identify the overlapping area and the FOV lines automatically given the views of two cameras.
- We propose the tangent transfer function to deal with the change of edge direction due to variant perspectives.
- We formulate the camera link model estimation as an integer optimization problem and present an unsupervised approach to solve the problem.
- We consider the outlier issue in the training data which has not been clearly investigated in the previous works.
- We propose region matching method in order to make more detailed comparison between people.
- Our estimation can be generalized easily by adding more features or components in the camera link model.

1.3 Dissertation Roadmap

The dissertation is organized as followed:

Chapter 2: we review the recent works on previously proposed tracking systems including single camera tracking and multiple-camera tracking and briefly describe how our proposed methods improve in dealing with the problems on which we focus.

Chapter 3: our single camera tracking system is described in this chapter. First, we introduce the multiple-kernel tracking scheme that tracks a specific person in the crowd. This method, as our main contribution in single camera tracking, deals with the occlusion problem by applying multiple kernels to different human body parts. The constraints and adaptive weights are introduced to obtain better optimum solution. Afterward, we embed the multiple-kernel tracking into a Kalman filtering-based tracking system to enable fully automatic tracking. Extensive experiments and analysis of the proposed method are also provided.

Chapter 4: we present our approach for tracking human across multiple cameras with overlapping views. The camera link model, including homography matrix, brightness transfer function and tangent transfer function, are estimated automatically given the cameras' views. Vicinity cue, color cue and edge cue are considered while matching the people across cameras.

Chapter 5: we present our system for tracking human across multiple cameras with nonoverlapping views. We formulate the camera link model estimation as an optimization problem and apply a deterministic annealing algorithm to obtain the optimal solution. The reliable model is built based on this unsupervised scheme even under the presence of the outliers in the training data. We construct the complete system to track humans across the cameras deployed in the real world based on camera link models, which can be continuously updated in the testing stage in order to refine the model and to adapt the changes of the environment.

Chapter 6: we conclude this dissertation by summarizing our main contributions and also discussing the extension work which leads to potential research topics in the future.

Chapter 2 – Backgrounds and Related Works

In this chapter, we introduce several previous works related to the three major topics in this dissertation and briefly describe how our proposed methods improve in dealing with the problems we focus on.

2.1 Single Camera Tracking

Among the existing tracking algorithms, kernel-based object tracking [26] has recently gained more popularity for more robust tracking performance due to its fast convergence speed and relatively low computation. In our work, we focus on using kernel-based tracking to solve the occlusion problem. Since a thorough discussion on the general visual tracking is not within the major scope of this dissertation, here we only review the most relevant ones focusing on kernel-based tracking.

The mean-shift method, one of the most popular kernel-based tracking methods, was applied to the tracking problems to find the most similar location around the local neighborhood area in Comaniciu's works [25][26]. Collins et al. [27] used the difference of Gaussian and Lindeberg's theory to track the objects through the scale space. A sample-based similarity measurement combined with a fast Gaussian transform was proposed by Yang et al. [79] to fulfill the mean shift procedure during tracking. In Yilmaz's approach [81], they employed the asymmetric kernels that can adaptively change the scale and orientation to track the target. Other information, such as boundary cues, was also utilized in the kernel-based tracking systems [53][82].

All the above works only used a single kernel during tracking, and it would fail under occlusion because of the ill-observation of the object. In order to better represent the tracked target, multiple kernels have also been adopted these years. A different similarity measure was exploited in Hager's work [40], where they used Newton-style iterations to minimize the sum of squared difference (SSD) measure. In Martinez's work [58], the

human represented by multiple kernels, which denoted various body parts, were tracked by using a two-step approach. The global movement and local movement were alternatively applied to locate the target. Porikli et al. [65] used the multiple kernels centered at the high motion areas to enhance the performance when the tracked targets have fast motion. In [32], Fang et al. presented a fragments-based multiple-kernel tracking scheme. They divided the target into several fragments, and each fragment was tracked by a kernel. However, none of the above utilized the inter-relationship among kernels, which can provide useful information for tracking. In [63], Parameswaran et al. proposed a feature representation which considered multiple kernel centers. It implicitly embedded the geometrical information into the feature representation, but the relationships between the kernels were not explicitly constrained. Moreover, the adaptive weights for different kernels could not be imposed. Fan et al. [31] linked the multiple collaborative kernels by using predefined constraints. In their approach, the tracking was formulated as solving a least square problem. It may work under normal conditions in which there is little occlusion or none, but the result tends to deviate from the optimum easily under occlusion since the minimum cost value is no longer a small value. Moreover, the reliability weightings for different kernels were not considered, resulting in an unstable system when kernels are in great conflict. Additionally, their Gauss-Newton style iteration is limited to a least-squares problem formulation and may not be easily generalized for other kinds of cost functions such as K-L distance. Hence, we propose a projected gradient based multiple-kernel tracking to overcome the above problems [19][20].

2.2 Multiple-Camera Tracking

2.2.a Camera Link Model

There may be several *entry/exit zones*, defined as the areas that people tend to enter in or leave from within a camera's view, in a camera's view, and a path between a pair of

cameras actually connects one zone each in these two cameras. Given an identified pair of directly connected camera zones, the temporal/appearance relationships between a pair of entry/exit zones in two cameras corresponding to a path can be characterized by a *camera link model*. The model enables us to utilize particular features, which may not be invariant under different cameras, for person re-identification purposes. It consists of transfer functions or look-up-tables which need to be applied before comparing the features extracted from the humans in two cameras that may or may not have overlapping views. For instance, due to different lighting conditions and camera color responses, the same object may appear with different colors under different views. The brightness transfer function (BTF) [30][66], which stands for the mapping of color models between two cameras, can be applied to compensate for the color difference between two cameras before we compute the distance between the color histograms of two observations under two cameras. In our work, several components are included in the camera link model, which enables the system to match the humans with various features.

2.2.b Tracking Across Cameras with Overlapping Views

The field of view (FOV) line, as proposed by Khan et al. [50], represents how the view limits of one camera are seen in the others. Javed et al. [44] presented a system using the distance to FOV line to find the correspondence of the objects. By further exploiting Javed's method, the extracted motion trends and appearance of objects were combined in [45]. In Kang's method [47], the spatial-temporal homography was utilized to register multiple trajectories to track the objects across multiple cameras. Moller et al. [59] suggested a hierarchical relocation procedure based on the color information during the matching stage. In our work, we utilize image registration technique to identify the FOV lines automatically given the views of two cameras [18][87]. Identifying the FOV lines enables us to know the overlapping areas between two cameras. To compensate for the color deviation and perspective difference, brightness transfer function and tangent

transfer function are included in the camera link model and are utilized in the matching scheme.

2.2.c Tracking Across Cameras with Nonoverlapping Views

Tracking multiple people across uncalibrated cameras with disjoint views is a rather challenging problem. Some researchers aim to come up with distinctive features of the targets, such as SIFT, SURF, covariance matrix, etc [6][7][33][35][39]. The re-identification is done based on the assumption that these kinds of features are invariant under different cameras' views. However, the ideal features for describing humans have not been discovered yet, since the human appearance varies dramatically due to different perspectives, poses and illuminations. On the contrary, instead of relying on the complex feature representations, we focus on solving the tracking problem based on systematically building the link between cameras [22].

Normally, two components are included in a camera link model: transition time distribution and brightness transfer function (BTF). The works in [30][45][46][66] obtained the camera link model with a large amount of manual work. Their supervised learning schemes were not feasible when the scale of the camera network increased. There are approaches with unsupervised learning methods. Makris et al. [57] proposed an estimation method to build the transition time distribution based on the cross-correlation between the exit and entry time stamps of the objects. Their assumption on the single-mode distribution was inadequate, since it could not represent most cases in the real world. In [74], Tieu et al. presented an entropy-based method to build the distribution using Markov Chain Monte Carlo (MCMC) to find the optimal correspondence. It has been shown in [42] that their minimum-entropy assumption may not hold for practical applications. In Gilbert's work [36], similar to [57], they considered all the possible correspondences within a given time window which consisted of the mixture of true and false correspondences, and the number of false ones was usually much larger than the number of true ones; hence, a large amount of noise was included in the estimation

process resulting in an unreliable model. Huang et al. [42] introduced a method based on Gibbs sampling, where they used hard decisions to determine the correspondence during the whole estimation process, which may be easily trapped in poor local optima. Furthermore, they did not take into account the outlier issue in the training data.

People have tried to utilize multiple cues for the re-identification in building a multiple-camera tracking system. In [15], Chigunde et al. built a multi-camera tracking system based on the camera topology and objects' motion and shape information. The views of all the cameras were projected to a common ground plane from the homography. The manual calibration and the assumption of constant speed of the objects were required in their implementation. In Gilbert's work [36], the tracking relied on three cues: color, size, and transition time. The links were learned in an incremental scheme. In order to obtain a dependable link model, large amount of training data were needed. Javed [45][46] presented a multiple-camera tracking system which combined the temporal and color features. In [13], Chen et al. incorporated Markov Chain Monte Carlo into the training stage, and the transition time distribution and BTF were employed to perform the re-identification. Kuo et al. [52] proposed to associate multiple targets by on-line learning the discriminative appearance models consisting of color, texture, and shape features. They applied the Hungarian algorithm to solve the correspondence matrix given the testing data without taking into account the color deviation between cameras. Although all the above works considered multiple features during the matching, the cues were integrated either with uniform weights or empirically determined weights. We have included the feature fusion weights, learned in the training stage, in the model. Moreover, since the viewpoints are different in the cameras, the holistic appearance of the target may not be discriminative enough. In [6][33][83], the human body was divided into several regions where the features were extracted. However, the direct comparison between the features from the corresponding regions in two objects may cause some errors, since the corresponding regions may not cover the same area in the real world.

Therefore, the region mapping matrix and region matching weights are included in our camera link model, which enables better matching between the regions of human bodies.

Chapter 3 – Single Camera Tracking

3.1 Single Camera Tracking System Overview

By taking the advantage of the efficient Kalman filter tracking [14] and the robustness of multiple kernels tracking [19][20], we propose a fully automatic tracking system that does not need any human intervention (see Fig. 3.1). The input is the video frame, and the output is the tracking result for each person in the frame. First, to extract the foreground objects, the background model constructed from the mean frame among the several frames is subtracted from the current frame. We then apply the Otsu method [62], which dynamically determines the threshold, to the difference image to build the binary image. Morphological reconstruction filters [37] are then used to remove the small noise, connect the broken part, and smooth the shapes of the foreground objects. After that, for each object that has already been tracked in the previous frames, the Kalman filter prediction is performed. The next stage is to detect whether the object is under occlusion by checking if the predicted states of the tracked objects merge with one another. If it is not, the segmentation result is regarded as reliable, and the nearest segmentation result is selected as the corresponding measurement. Otherwise, we utilize multiple-kernel

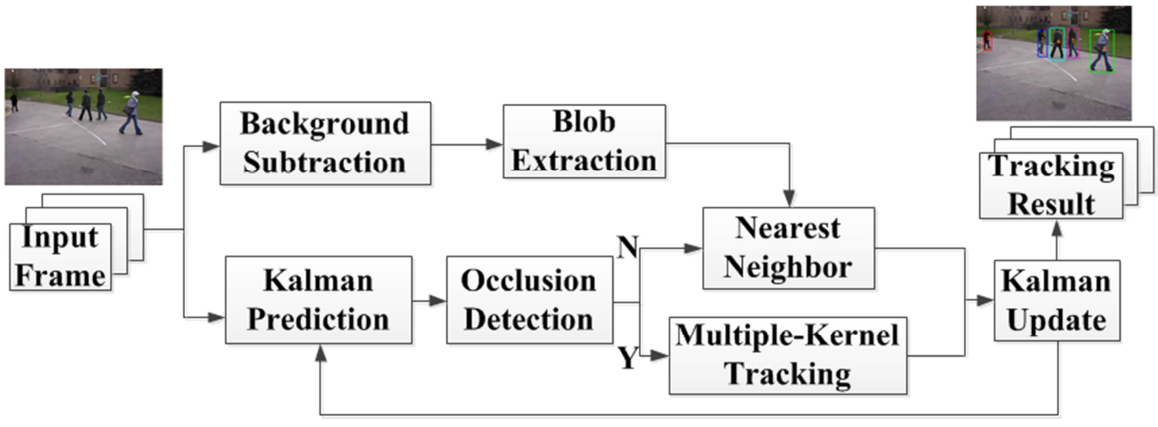


Figure 3.1: Single Camera Tracking System

tracking to get the measurement. By using the measurement either from the result of segmentation or the result of multiple-kernel tracking, we are able to update the Kalman filter and display the tracking result.

3.2 Multiple-Kernel Tracking

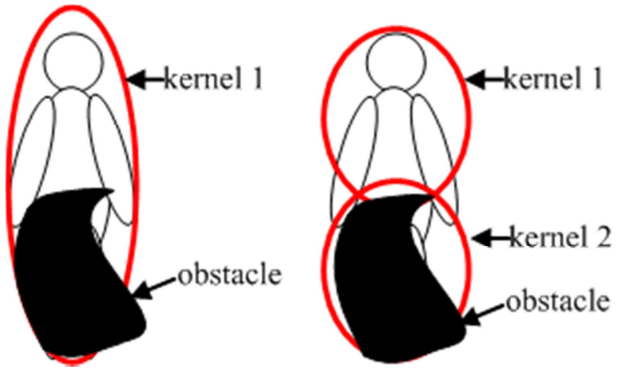
To achieve the successful tracking, the target needs to be located in consecutive frames. Assume the target model is known, and we can extract the candidate model at each location in a video frame. If we define the similarity measure between these two models, the objective is to find the candidate model that has the highest similarity.

3.2.a Single Kernel Tracking

In conventional kernel based tracking [26], a model is represented as the probability density function in the feature space, i.e., the histogram. During the histogram extraction, the amount of the contribution (weight) of a pixel is controlled by the value of a kernel function $k(\cdot)$, and this value is determined based on the distance between the pixel and the kernel center. To achieve successful tracking, the target needs to be located in consecutive frames by comparing the target model and the model extracted at the current location. Assume the target model is known, and a candidate model centered at a particular location in a video frame is extracted, the goal is to find the candidate model that has the highest similarity value, and the associated center location is the tracking results. In [26], the tracking problem is expressed as maximizing the predefined similarity function and can be reformulated as finding $\mathbf{x}$ that maximizes the density estimator $f(\mathbf{x})$

$$f(\mathbf{x}) = \frac{\sum_{i=0}^{N_h} \omega_i k(\left\| \frac{\mathbf{x}-\mathbf{z_i}}{h} \right\|^2)}{\sum_{i=0}^{N_h} k(\left\| \frac{\mathbf{x}-\mathbf{z_i}}{h} \right\|^2)}, \tag{3.1}$$

where $\mathbf{x}$ is the center of the kernel, which is normally the centroid of the object; $\mathbf{z_i}$ is the location of the pixel within the predefined range, and N_h is the number of pixels; ω_i is the weight for each pixel; $k(\cdot)$ is the kernel function which is usually a monotonously



(a) (b)

Figure 3.2: Illustration of the occlusion and kernels. (a) Single kernel with occlusion. (b) Two kernels with occlusion. Red ellipses represent kernels.

decreasing function. The mean shift procedure [25] is adopted to efficiently find the optima.

3.2.b Multiple-Kernel Cost Function

Alternatively, we can reformulate the problem from maximizing the similarity to minimizing the cost function, which can be designed to be inversely proportional to the similarity as shown in (3.2),

$$J(\mathbf{x}) \propto 1/simi(\mathbf{x}), \tag{3.2}$$

where $simi(\mathbf{x})$ is the similarity function at the location $\mathbf{x}$ in the state space domain.

If we use a single kernel to track the object, the mean-shift tracking [25][26] can be adopted. However, when the target is occluded, an error occurs. This can be avoided by applying multiple kernels as shown in Fig. 3.2. If occlusion happens, the tracking performance is severely affected, since kernel 1 incorporates a large portion of irrelevant information (obstacle) to the target in Fig. 3.2 (a). However, once an additional kernel is added in Fig. 3.2 (b), although kernel 2 is nearly non-observable, the well-observed kernel 1 can still be used to compensate for the adverse effect resulting from the occlusion after some constraints linking the two kernels are introduced. Hence, for N

multiple kernels we define the total cost function $J(\mathbf{x})$ to be the sum of the N individual cost functions $\{J_i(\mathbf{x})\}$,

$$J(\mathbf{x}) = \sum_{i=1}^{N} J_i(\mathbf{x}). \tag{3.3}$$

In addition to the cost function, the constraint functions $\mathbf{C}(\mathbf{x}) = \mathbf{0}$ need to be imposed. The constraint functions confine the kernels based on their inter-relationships. For instance, if the object in Fig. 3.2 is a rigid body, kernel 1 will be always on top of kernel 2 (see Chapter 3.3 for our design). Hence, the problem would become searching the state $\hat{\mathbf{x}}$, such that

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) \qquad \text{subject to } \mathbf{C}(\mathbf{x}) = \mathbf{0}. \tag{3.4}$$

3.2.c Projected Gradient-based Multiple-Kernel Tracking

Starting from the state in the previous frame, in order to efficiently decrease the total cost function and keep the constraints satisfied during the state search, the movement vector $\delta\mathbf{x}$ is determined based on the projected gradient method [73], which is used to iteratively solve the constrained optimization problem. The intuitive idea is to project the gradient vector onto the space in which the values of the constraint functions remain unchanged. Moreover, in order to further handle the constraints, the second term, which guarantees the satisfaction of the constraints (i.e., ensuring $\mathbf{C}(\mathbf{x}) = \mathbf{0}$), is introduced (see Appendix A1). Hence, $\delta\mathbf{x}$ consists of two components $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$:

$$\begin{aligned} \delta\mathbf{x} &= \alpha(-\mathbf{I} + \mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}_\mathbf{x}^T)\mathbf{J}_\mathbf{x} + (-\mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}(\mathbf{x})) \\ &= \quad \delta\mathbf{x}_A \quad + \quad \delta\mathbf{x}_B \quad , \end{aligned} \tag{3.5}$$

where $\mathbf{x} \in \mathcal{R}^{\mathrm{n}}$ is the state vector; $\mathbf{C}(\mathbf{x}) = \begin{bmatrix} c_1(\mathbf{x}) \\ \vdots \\ c_m(\mathbf{x}) \end{bmatrix}$ consists of m constraint functions, and $c_i(\mathbf{x}): \mathcal{R}^{\mathrm{n}} \to \mathcal{R}$ is the *i-th* constraint function; $\mathbf{C}_\mathbf{x} \in \mathcal{R}^{n\times m}$ is the gradient matrix of constraint functions with respect to $\mathbf{x}$; $\mathbf{J}_\mathbf{x}$ is the gradient vector of the total cost function with respect to $\mathbf{x}$, and $\alpha > 0$ is the step size.

The iterative projected gradient search formulation has the following characteristics,

1. $\mathbf{x}$ is the final state from the previous frame; $counter = 0$.
2. **while** $counter < T$
3. $\quad counter = counter + 1$.
4. $\quad$ Evaluate $\mathbf{C}(\mathbf{x})$ and $J(\mathbf{x})$.
5. $\quad$ **if** $|\mathbf{C}(\mathbf{x})| < \boldsymbol{\varepsilon}_{\mathbf{C}}$ and $J(\mathbf{x}) < \varepsilon_J$ **then**
6. $\quad\quad$ End of the iteration.
7. $\quad$ **else**
8. $\quad\quad$ **if** $|\mathbf{C}(\mathbf{x})| < \boldsymbol{\varepsilon}_{\mathbf{C}}$ and $J(\mathbf{x}) \geq \varepsilon_J$ **then**
9. $\quad\quad\quad$ Compute $\delta\mathbf{x}_A$.
10. $\quad\quad\quad$ Apply $\mathbf{x} = \mathbf{x} + \delta\mathbf{x}_A$.
11. $\quad\quad$ **else**
12. $\quad\quad\quad$ **if** $|\mathbf{C}(\mathbf{x})| \geq \boldsymbol{\varepsilon}_{\mathbf{C}}$ and $J(\mathbf{x}) < \varepsilon_J$ **then**
13. $\quad\quad\quad\quad$ Compute $\delta\mathbf{x}_B$.
14. $\quad\quad\quad\quad$ Apply $\mathbf{x} = \mathbf{x} + \delta\mathbf{x}_B$.
15. $\quad\quad\quad$ **else**
16. $\quad\quad\quad\quad$ Compute $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$.
17. $\quad\quad\quad\quad$ Apply $\mathbf{x} = \mathbf{x} + \delta\mathbf{x}_A + \delta\mathbf{x}_B$.
18. $\quad\quad\quad$ **end if**
19. $\quad\quad$ **end if**
20. $\quad$ **end if**
21. **end while**

Algorithm 3.1. Projected gradient iteration in each frame for multiple-kernel tracking. $|.|$ takes entry-wise absolute value.

which are proved in Appendix A2:

i) $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$ are orthogonal to each other.

ii) Moving along the $\delta\mathbf{x}_A$ decreases the total cost function $J(\mathbf{x})$ and keeps the values of the constraint function vector $\mathbf{C}(\mathbf{x})$ unchanged.

iii) Moving along the $\delta\mathbf{x}_B$ can lower the absolute values of the constraint function vector $\mathbf{C}(\mathbf{x})$.

Based on the above characteristics, starting from the initial point in each frame, we evaluate $J(\mathbf{x})$ and $\mathbf{C}(\mathbf{x})$ and then interchangeably apply $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$ to decrease the cost while satisfying the constraints in each iteration. We stop the iteration if either the cost and the absolute values of the constraint functions are both below some given thresholds ε_J and $\boldsymbol{\varepsilon}_{\mathbf{C}}$ or the iteration count exceeds T (see Algorithm 3.1). Fig. 3.3 is a typical

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

Intermediate Steps	*Iteration 1*	*Iteration 2*	*Iteration 3*	*Iteration 4*
cost function	3.44892	2.25533	1.48434	1.329651
1st constraint function value (Eq. 3.12)	1	1	1	1
2nd constraint function value (Eq. 3.13)	0	0	0	0
update vector applied	$\delta\mathbf{x}_A$	$\delta\mathbf{x}_A$	$\delta\mathbf{x}_A$	$NULL$

(e)

Figure 3.3: An example of the intermediate steps. The two ellipses represent two kernels. (a) Before iteration 1 starts in the current frame (The initial locations of the two kernels are inherited from the final state in the previous frame.). (b)(c)(d) Updated results after iteration 1 to 3. Since the cost function is above the threshold ($\varepsilon_J = 1.4$), $\delta\mathbf{x}_A$ is applied until the cost function is below the threshold. (e) The corresponding values for the cost function and the constraint functions.

example for how the algorithm proceeds within one frame. Fig. 3.3 (a) shows the initial state, which is inherited from the final state in the previous frame, and Fig. 3.3 (b)~(d) are the subsequent iterations. The two constraint functions specify the geometrical relationship between two kernels, as given in (3.12) and (3.13). Based on the values of the cost function and the constraint functions (Fig. 3.3 (e)), $\delta\mathbf{x}_A$ is applied and according to characteristic (ii), the values of the constraint function remain as 1 and 0 through the iterations until the termination condition is satisfied.

We use the mean shift vector [26] with the opposite direction as our $\mathbf{J_x}$ in the implementation, while maintaining characteristic (ii) (Appendix A3). If there is no constraint (i.e., both $\mathbf{C(x)}$ and $\mathbf{C_x}$ are $\mathbf{0}$), then the movement in (3.5) becomes

$$\delta\mathbf{x} = -\alpha\mathbf{J_x}, \tag{3.6}$$

which is just the formulation of each kernel performing an independent mean shift update.

3.2.d Adaptive Cost Function

When there is occlusion, not all the kernels are reliable. Thus, we associate each kernel with an adaptively changeable weight value w_{i} in the calculation of the total cost function,

$$J(\mathbf{x}) = \sum_{i=1}^{N} w_i J_i(\mathbf{x}). \tag{3.7}$$

Therefore, the movement vector in (3.5) is modified to be

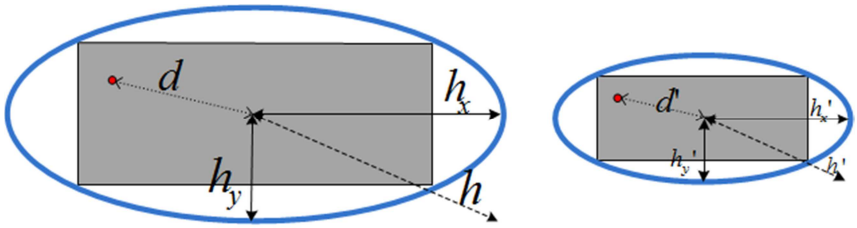
$$\delta\mathbf{x} = \alpha(-\mathbf{I} + \mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C_x^T})\mathbf{W}\mathbf{J_x} + (-\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C}(\mathbf{x})), \tag{3.8}$$

where $\mathbf{W} = \begin{bmatrix} w_1\mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & w_N\mathbf{I} \end{bmatrix}$ and $w_i \propto simi_i(\mathbf{x})$; $\mathbf{I}$ is an $\frac{n}{N} \times \frac{n}{N}$ identity matrix with n being equal to the dimension of the state space, and N is the number of kernels. The value w_i, which corresponds to the *i-th* kernel, is adaptively updated based on the individual similarity $simi_i$ and is normalized to make the sum equal to N. The similarity represents the degree of match between the color feature of the candidate and the target model in the single kernel. The higher similarity will give us the higher weight value, which corresponds to more confidence in this kernel. This would enable the system to produce a better movement toward the optimum when the kernels have conflicts.

3.2.e Scale Change Issue

The scale (size) of the video object will probably change if the object is moving toward or away from the camera. Unlike the other scale update methods [26][27][81], which need additional processes for dealing with scale variables, such as using additional kernels, we propose a simple while effective solution to overcome the scale change issue, as evidenced by the experimental results in Chapter 3.3.c.

The extracted histogram used for similarity measure of the object is highly dependent on the kernel bandwidth h. If the object size is changing, it is better to adaptively change the kernel bandwidth in order to obtain a histogram that matches the original one. If all



(a) (b)

Figure 3.4: (a) Original size. (b) Smaller size. The grey rectangle is the object. The blue ellipse represents the kernel with bandwidth h and h' which is equal to $\sqrt{h_x^2 + h_y^2}$ and $\sqrt{h_x'^2 + h_y'^2}$, respectively. The red dots, whose distances to the center are d and d', are the corresponding locations on the objects with two different sizes.

the corresponding pixels have similar weights, the histograms would be similar. For instance, Fig. 3.4 shows the same object with different sizes. The two corresponding pixels (red dots) have the same weights for the histogram calculation if $\frac{d}{h}=\frac{d'}{h'}$. Hence, the object size and the kernel bandwidth are positively correlated. Thus, the change of the kernel bandwidth h for maximizing the similarity (or minimizing the cost function) is utilized to infer the change of the object scale. We take the derivative of the density estimator in (3.1) with respect to h,

$$\nabla f(h) = \frac{\partial f(h)}{\partial h} = f(h)\left[\frac{-2\sum_i (g(v_i)v_i)}{h\sum_i k(v_i)} + \frac{2\sum_i (\omega_i g(v_i)v_i)}{h\sum_i \omega_i k(v_i)}\right], \tag{3.9}$$

where $g(v_i) = -k'(v_i)$ and $v_i = \left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2$.

We can see that all the terms in (3.9), including ω_i, v_i, $g(v_i)$, $k(v_i)$, and h, can be directly inherited from the tracking steps such as the mean shift vector and cost function computing. Thus, based on these by-products, the scale change factor can be efficiently estimated as,

$$\Delta s = \eta \frac{\nabla f(h)}{h f(h)} = \eta r(h), \tag{3.10}$$

where η is the scalar factor which takes into account both the sensitivity of the scale change and the smoothing factor. Δs is proportional to the gradient $\nabla f(h)$ normalized by

the current kernel bandwidth h and density estimation value $f(h)$ enabling the system to control the step size adaptively. The scale s_t at time t can be updated according to (3.11),

$$s_t = s_{t-1}(1 + \Delta s) \, . \tag{3.11}$$

If the average of all kernels' similarity values is above a certain threshold, the scale update takes place after the iteration in Algorithm 3.1 finishes. The scale change method is developed based on the assumption that the localization is done, i.e., the center of the kernel in the current frame and the center of the kernel in the previous frame are at the same point or two close points on the target, therefore the scale update should take place after the tracking iteration is completed. In this manner, the scale can be adaptively updated to reflect the appropriate size.

3.3 Experimental Results

In this section, we start with introducing our implementation details. Next, the experiments are divided into three parts: first, the scenarios are mainly in tracking a particular person in the crowd. We compared our proposed multiple-kernel tracking with two kernel-based tracking methods: multiple collaborative kernels [31] and single kernel mean shift [26], and two state-of-the-art trackers: on-line boosting [38] and superpixel tracking [76], by using several databases. Second, we will show the effectiveness of our approach for dealing with the scale changes of the tracked objects. Finally, robust tracking of multiple people simultaneously under occlusion is demonstrated.

3.3.a Implementation Setup

In the multiple-kernel tracking module, each object is represented by multiple inter-related kernels, and the overall state vector **x** is composed of the centroids of all the kernels $[x_1 \; y_1 \; \ldots \; x_N \; y_N]^T$, where N is the number of kernels, and (x_i, y_i) is the centroid of the i-th kernel. Theoretically, more kernels will give better results, since it can handle occlusion from various directions, especially in a crowded scene environment. However,

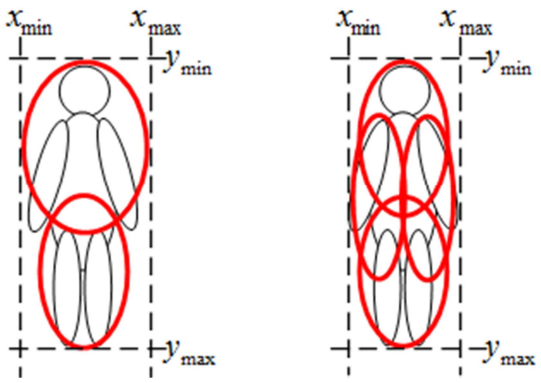


Figure 3.5: Layout for the multiple kernels. (a) 2 kernels. (b) 4 kernels. Kernels are represented as red ellipses. Note that the object is not necessarily viewed from the front. The figure here is just for the demonstration purpose.

due to the different sizes of the targets, the number of kernels should be limited. For example, if the size of the target is small, each kernel may cover only a small portion of the body. The extracted histograms may not be informative enough due to insufficient pixels. In this work, based on the symmetry and characteristics of the human body, we use two kernels or four kernels to represent an object, and the layouts are shown in Fig. 3.5. These kernels are generated systematically and automatically once the multiple-kernel tracking is activated. The layout and the constraints can be designed to reflect the target's physical characteristics. For instance, in the two-kernel setting, whether the viewing angle is from front or side, the system automatically generates the first kernel to represent the upper 55% of the object, while the other one covers the lower 55% of the object, so there is some overlapping between them. Since the upper and lower body parts of a human normally have rigid geometrical relationship, it is reasonable to formulate the constraints as (3.12) and (3.13):

$$c_1(x_i, x_j) = ((x_i - x_j)^2 - {L_{x,ij}}^2)/({L_{x,ij}}^2 + 1), \quad (3.12)$$

$$c_2(y_i, y_j) = ((y_i - y_j)^2 - {L_{y,ij}}^2)/({L_{y,ij}}^2 + 1), \quad (3.13)$$

where $L_{x,ij}$ and $L_{y,ij}$ are the distances which remain constant after the initialization,

unless the object size changes; and (x_i, y_i) and (x_j, y_j) are the centroids of kernels i and j, respectively. At the first frame, each kernel initializes its own target model based on its covering area, and the $L_{x,ij}$ and $L_{y,ij}$ values are automatically determined by setting (3.12) and (3.13) to zero; that is, the constraint functions are satisfied in the beginning: $c_1(x_i^{initial}, x_j^{initial}) = 0$ and $c_2(y_i^{initial}, y_j^{initial}) = 0$. The $L_{x,ij}$ and $L_{y,ij}$ values are adjusted in proportional to the scale change only when the size of the object changes. As shown in Fig. 3.5, the extremes of the kernels, x_{min}, x_{max}, y_{min}, and y_{max}, form the bounding box. The centroid of the tracking result is obtained based on (3.14),

$$x_c = \left(\frac{x_{min} + x_{max}}{2}\right), y_c = \left(\frac{y_{min} + y_{max}}{2}\right). \quad (3.14)$$

The α in (3.8) is set to 1 since the mean shift vector already provides adaptive step size. In Algorithm 3.1, the values of $\boldsymbol{\varepsilon_C}$ actually depend on the size of the object, and we assign the values in the range of 0.05 to 9 to each entry of $\boldsymbol{\varepsilon_C}$; ε_J is given by the value of $0.7 \times N$, where N is the number of kernels. The max number of iterations (T) is equal to 5. η in (3.10) is set to 9×10^3 empirically to reflect the appropriate amount of the scale change. The target's model and the scale will only be updated if the average of all kernels' similarity values is above 0.6. We use K-L distance [9] for all the similarity-related and cost computations through the implementation. To construct the histogram of the object, the HSV color space and roof kernel are employed.

3.3.b Experiments on Multiple-Kernel Tracking

First of all, in order to emphasize the robustness of our proposed multiple-kernel tracking method, we focus on tracking a specific person who is occluded, with potential scale change, in the crowd within the video. The comparisons with two other kernel-based methods, as presented in [31] and [26], and two state-of-the-art trackers, on-line boosting [38] and superpixel tracking [76], show the robustness and improvement of our proposed method. In this experiment, all the targets of interest and kernels are selected manually in the beginning.

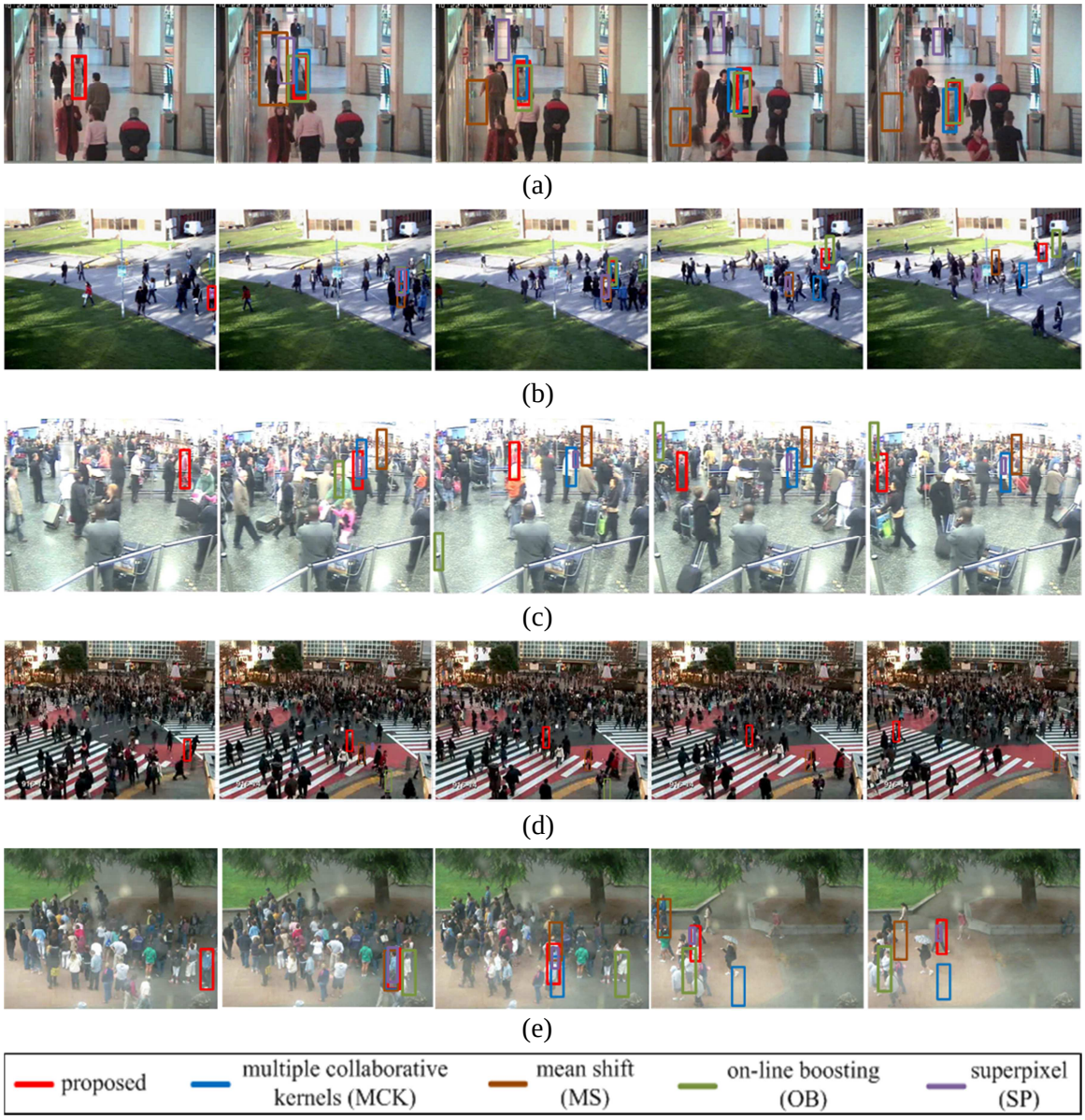


Figure 3.6: Tracking a person under occlusion. (a) *CAVIAR* Database: *OneStopMoveEnter1cor* sequence. Frame #1, #28, #73, #132, #164. (b) *PETS2010* Database: *S2.L2.View_001* sequence. Frame #344, #368, #381, #406, #429. (c) *i-LIDS* Database: *MCTTE0102* sequence. Frame #1, #71, #189, #322, #361. (d) Crowd Analysis Dataset. Frame #1, #61, #82, #93, #181. (e) Our own recorded video. Frame #1, #39, # 102, #444, #485.

Fig. 3.6 shows some of our test results. We have evaluated our approach using the CAVIAR Database [11], with the frame size of 384x288. We select some clips that have at least 5 people in the scene and pick the target that is continuously occluded during the movement. The target is marked by the bounding box for readers to see clearly. Fig. 3.6 (a) shows 5 representative frames out of the tracking results based on using our proposed method with two kernels and the competing methods. The corresponding trackers are shown in different colors. In frames #28 and #132, the target is occluded. Our method can effectively track the target (marked as red bounding box), while it results in larger errors by applying the methods in [31][38] or even losing the tracking of the target by using the methods in [26][76].

In addition to the CAVIAR database, we use the video from PETS 2010 Benchmark Database [64]. One of the objectives of creating this database is to track the individuals within the crowd, so it is perfect for our usage to prove the effectiveness of our approach. The frame size is 768x576, and it is taken under 7 frames per second. Fig. 3.6 (b) shows that our method with two kernels produces promising tracking and scale updating results. Moreover, the iLIDS database [75] is utilized to evaluate the tracking accuracy. It captures the scene in a busy airport. The frame size is 720x576. Our algorithm using four kernels clearly outperforms the others (Fig. 3.6 (c)). Some video clips for the crowd analysis [68], with the frame size of 480x360, are also used for the experiments in this work. In Fig. 3.6 (d), our tracker with two kernels tracks the target moving across the central area. The trackers we compared to lose the target, and some of them even drift out of the visible area.

We also use our own captured videos to do the experiments. These videos, with the frame size of 640x480, contain the tracked target changing the direction while walking in the crowd scene. In Fig. 3.6 (e), frames #39, #102, and #444 show the target is continuously occluded in the crowd. The proposed method using two kernels steadily tracks the specific person effectively.

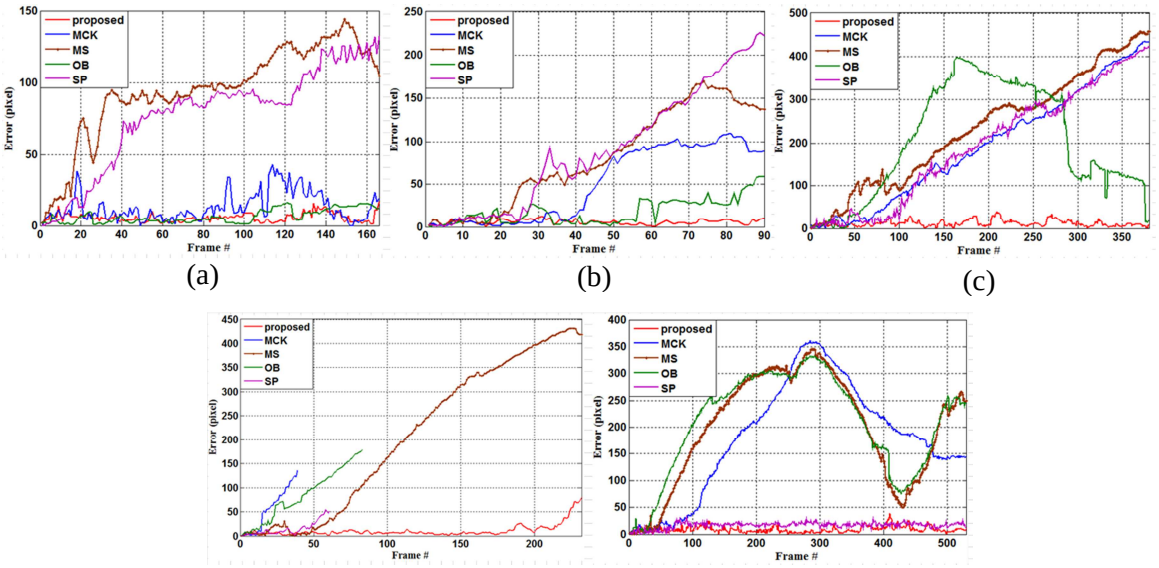


Figure 3.7: Errors (pixels) of the object center against the ground truth in Fig. 3.6 (a)~(e). Our proposed method (red) outperforms other methods.

Table 3.1: Average error (pixels) in Fig. 3.6 (a)~(e)

Method	Avg. Error (pixels)				
	CAVIAR	*PETS2010*	*i-LIDS*	*Crowd Analysis*	*Our own video*
proposed method	**5.62**	**5.73**	**11.43**	**10.81**	**8.75**
multiple collaborative kernels (MCK) [31]	13.27	50.51	192.19	55.60	177.21
mean shift (MS) [26]	94.95	81.40	231.59	205.55	199.18
on-line boosting (OB) [38]	6.53	17.86	200.74	84.17	208.34
superpixel (SP) [76]	77.62	90.55	192.13	15.02	16.70

To further evaluate the performance quantitatively, the absolute errors for each frame in the above scenarios are computed and shown in Fig. 3.7. The error is defined as the pixel distance between the target's centroid of the simulation result and the ground truth as provided by the CAVIAR database [11] for the scenario in Fig. 3.6 (a). For the PETS database, iLIDS database, crowd analysis dataset, and our own captured video, the ground truth is produced by using the Video Performance Evaluation Resource (ViPER)

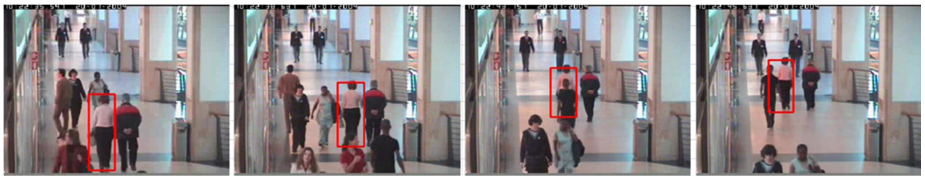
tool [29]. The deviations remain in the lowest values by using our method among all the scenarios, while the other trackers lose the targets during occlusion, resulting in large deviations in some or all of the cases. Note that sometimes small error does not necessarily mean an accurate tracking. For instance, in Fig. 3.6 (c), the on-line boosting tracker drifts away after the occlusion in the beginning and is trapped at the upper left corner. The target happens to move from right to left, resulting in the decrease of the error of the on-line boosting tracker (Fig. 3.7 (c)), but in fact the tracker still loses the target. In Fig. 3.7 (d), three compared trackers drift away and are out of the visible area, so their corresponding error curves terminate in the early stage. Table 3.1 further shows the average error among all the frames of the video clips. The quantitative error measurement in the table demonstrates the superior performance of our proposed method in dealing with occlusion.

3.3.c Experiments on Scale Change

In these experiments, we focus on tracking the targets that have obvious scale changes in the videos from different databases [11][64][75]. Similarly, all the targets are selected manually in the beginning and are tracked by our multiple-kernel tracking with the scale update. Fig. 3.8 exhibits the robustness of our proposed method. Since some of the other methods under comparison did not explicitly consider the scale change in their systems, we do not provide the comparison here. By efficiently using the by-product from the tracking procedure, i.e., the required terms for scale change factor computation in (3.9) can be inherited from the tracking steps in the associated optimization iterations, we not only perform the successful tracking but also effectively update the scale of targets accordingly.

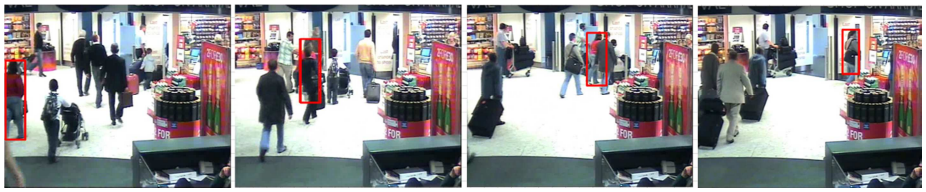
3.3.d Experiments on Single Camera Tracking System

In order to achieve fully automatic tracking, we further integrate the multiple-kernel tracking scheme into a Kalman filter based tracking system [14], where we use the



(a)

(b)

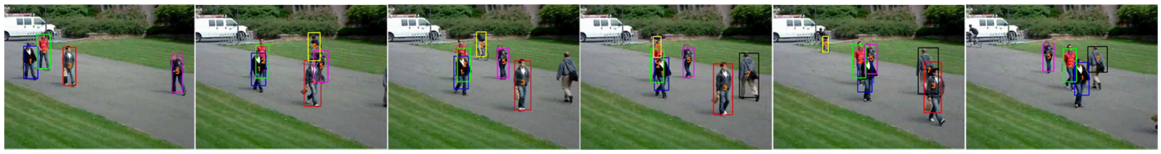


(c)

Figure 3.8: Tracking a person under occlusion with obvious scale change by using proposed method. (a) *CAVIAR* Database: *OneStopMoveEnter1cor* sequence. (b) *PETS2010* Database: *S2.L1.View_006* sequence. (c) *i-LIDS* Database: *MCTTE0201* sequence.

Kalman prediction as the initial location for the multiple-kernel tracking module and treat the result from the multiple-kernel tracking as the measurement for Kalman update. Several video clips are used for the performance evaluation. In order to make people differentiable, we tag them with numbers and different colored bounding boxes as well. Fig. 3.9 shows some of the results. In addition to our own captured videos, we use the video from AVSS 2007 Database [43] and PETS 2010 Benchmark Database [64]. From the background subtraction to the end of tracking, everything is fully automatic, and there is no need for any manual intervention. Note that the initialization of the multiple kernels is automatic as stated in Chapter 3.3.a.

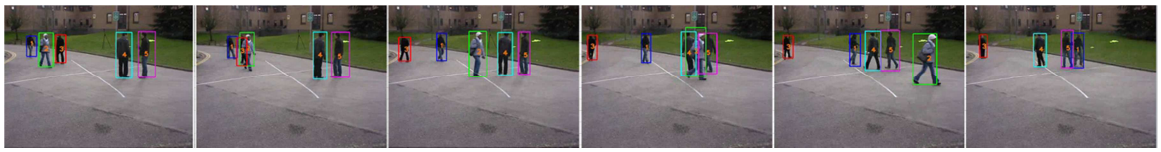
We further compare the performance, in terms of the average error, with two base-line trackers, mean shift tracking [26] and particle filtering [60], by using the same test video



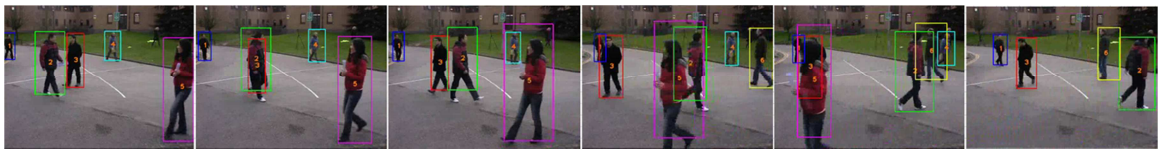
(a)

(b)

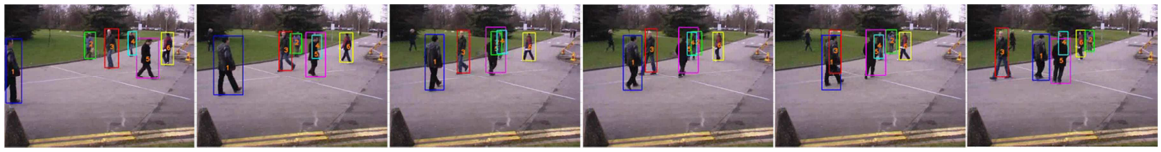
(c)



(d)



(e)



(f)

Figure 3.9: Tracking all individuals in the video simultaneously. Different people have been labeled with numbers and also with different colors bounding boxes. (a) Our own captured video. (b) *AVSS_AB_EVAL*. (c) *PETS 2010_S2.L1.View_001*. (d) *PETS 2010_S2.L1.View_008*. (e) *PETS 2010_S2.L1.View_008*. (f) *PETS 2010_S2.L1.View_007*.

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

Figure 3.10: Some tracking failure examples from mean shift or particle filter. In (c), the bounding box 3 (red) lost the person at left who has been occluded by the sign and the person at right. Since the box does not cover the area of the person at left, we remove this error value from the calculation of average error in Fig. 3.11.

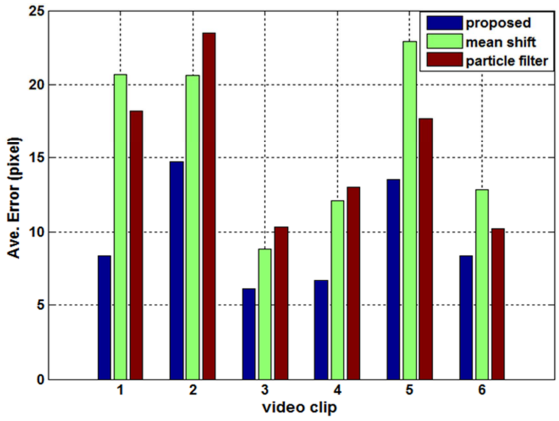
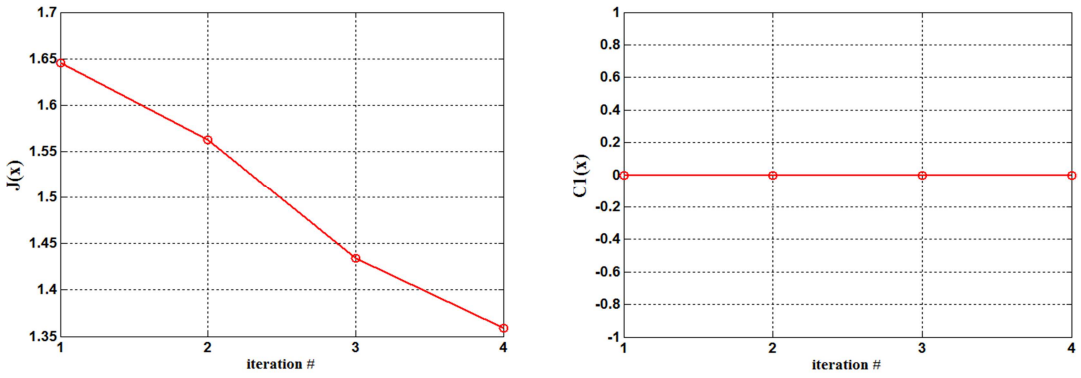


Figure 3.11: Comparison between our method (blue), single kernel mean shift (green), and particle filter (brown). The video clips 1~6 correspond to the results in Fig. 3.9 (a)~(f), respectively.

clips. The average error is obtained based on all the people in all the frames, and the error is defined as the pixel distance between the targets' centroids of the simulation results and the ground truths which are produced by using ViPER tool [29]. Originally, the mean shift method and particle filtering lose the targets occasionally during occlusion, and thus the trackers tend to drift away easily, resulting in larger errors. Some of their failure cases are shown in Fig. 3.10. In order to show the robustness and stability of our method, here we adopt another way to compute the average errors. For each person, we only consider the frames in which the corresponding bounding boxes of all three trackers overlap



(a) (b)

Figure 3.12: An example that shows the values of (a) cost function and (b) constraint function change as the iteration goes. Note that only one constraint function value is shown here. In this example, we can see it takes 4 iterations to make $J(\mathbf{x}) < \varepsilon_J$ ($\varepsilon_J = 1.4$).

with that person in the error computation. For instance, the frame in Fig. 3.10 (c) is excluded from the average error computation for the person labeled as 3, since the bounding box 3 totally leaves the corresponding person. Fig. 3.11 shows the quantitative comparison based on the above rule. We can see that even after discarding those frames having huge error values, our proposed method still attains better performance and higher stability since the occlusion results in larger deviation when using the other two methods even if they do not lose the target completely.

3.4 Discussion

3.4.a Projected Gradient Evaluation

The decomposition of the update vector $\delta\mathbf{x}$ ensures that the state vector reaches the optimum efficiently according to the characteristics proved in Appendix A2. Although some linear approximation has been exploited, the promising simulation results demonstrate that the characteristics still hold in practice. Fig. 3.12 shows a typical case of how the values of the cost function and one of the constraint functions change. Since the

(a) (b)

Intermediate Steps	*Iter. 1*	*Iter. 2*	*Iter. 3*	*Iter. 4*	*Iter. 5*
cost function	9.52	7.61	7.42	7.37	7.37
1st constraint function	0	0	0	0	0
3rd constraint function	0	0	0	0	0

(c)

Intermediate Steps	*Iter. 1*	*Iter. 2*	*Iter. 3*	*Iter. 4*	*Iter. 5*
cost function	9.52	7.86	9.02	7.80	9.16
1st constraint function	0	-0.681	0.178	-0.637	0.04
3rd constraint function	0	-0.883	-0.17	-0.77	-0.22

(d)

Figure 3.13: Two examples of the intermediate steps w/ ((a)(c)) and w/o ((b)(d)) the projected gradient. The four ellipses represent four kernels. The snapshots and the function values of the methods w/ and w/o the projected gradient are shown. The one with projected gradient gradually move to the target while the one w/o projected gradient swings from different states and does not converge. Note that there are 12 constraints in the four-kernel setting, and we only show two constraint functions here for illustration purposes.

cost function exceeds the threshold $\varepsilon_J(= 1.4)$ in Algorithm 3.1 and the constraint function is satisfied, the update takes place by moving along $\delta\mathbf{x}_A$ where one can see that the cost function drops to the lower level while the constraint function remains the same as the iterations proceed. In this example, the process stops at the 4th iteration since both the cost function and constraint function are below the thresholds.

The projected gradient in (3.8) plays an important role in both the accuracy and computation efficiency. The tracking accuracy comparison between the methods with and without projected gradient term is shown in Table 3.2. In the simulation without projected gradient, we discard the term $\mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}_\mathbf{x}^T$ in (3.8) while computing $\delta\mathbf{x}_A$, and

Table 3.2: Average error (pixels) comparison with and without projected gradient

Method	Avg. Error (pixels)				
	CAVIAR	*PETS2010*	*i-LIDS*	*Crowd Analysis*	*Our own video*
proposed method	**5.62**	**5.73**	**11.43**	**10.81**	**8.75**
without projected gradient	6.88	10.85	87.52	90.08	43.86

Table 3.3: Efficiency comparison for PETS2010 Database: S2.L2.View_001 sequence

Computation	*w/ projected gradient*	*w/o projected gradient*
ave. error (pixel)	7.3575	7.4008
T in algorithm 3.1	3	6
frames per second	10.19	5.869

other elements are left unchanged. Since the projected term is removed, the characteristic (ii) does not exist, and it is likely that the values of the constraint functions change while moving along the $\delta\mathbf{x}_A$. After that, applying $\delta\mathbf{x}_B$ may increase the cost function. Thus, the oscillation tends to happen and makes the tracking accuracy worse. The snapshots of the kernels' movements and the function values are shown in Fig. 3.13, which is a typical example of how the projected gradient leads to convergence through the iterations while the one without projected gradient does not approach the steady state point before the predefined maximum number of iterations is reached ($T = 5$). In Fig. 3.13 (l), one can see clearly the oscillation of either the kernels' movements or the function values when the projected gradient is not utilized. Projected gradient enables the decrease of the cost function while maintaining the satisfaction of the constraints, so in practice our proposed method usually does not need the computation of $\delta\mathbf{x}_B$.

In order to decrease the probability of oscillation occurring while not using projected gradient, one possible way is to reduce the step size (α) when applying the gradient vector. However, it takes more iterations for tracking, and it still fails in some cases. One can compare the computations of two methods, i.e., with and without projected gradient, by looking into (3.8). In each iteration, the method with projected gradient needs

additional computation of $\mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T$, and the one without projected gradient usually needs to compute $\delta\mathbf{x}_B$, which includes $\mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}$. Since the matrix inversion is normally the dominant computation and both of them have it in the computation, the computation cost in each iteration is similar in both cases. However, the required number of iterations is higher for the one without projected gradient in order to get a similar tracking performance. For instance, Table 3.3 shows the efficiency comparison between these two schemes using the video clips in the PETS database. We tried to adjust the parameters, and the number of maximum allowed iterations (*T*) in Algorithm 3.1 need to be set as 3 and 6 for w/ and w/o projected gradient, respectively, in order to obtain a similar tracking accuracy for both cases. Hence, the one with projected gradient attains higher computation efficiency in terms of processed frames per second.

Compared to Newton's method, gradient-based methods usually need more iterations to reach the optimum point. However, it has been proved by Fashing et al. [34] that "*the mean shift procedure with a piecewise constant profile g is equivalent to Newton's method applied to a density estimate using the shadow of g*." If we choose a piecewise linear profile k(·), such as the roof kernel, as we did in the implementation, the mean shift procedure will have a piecewise constant profile g $(g(\cdot) = -k'(\cdot))$; that is, doing mean shift update can have the same iterations just as Newton's method (i.e., the mean shift vector equals to the Newton step). Therefore, by combining the mean shift vector into our gradient–based approach (3.8) enables our method to attain both robust performance and efficiency.

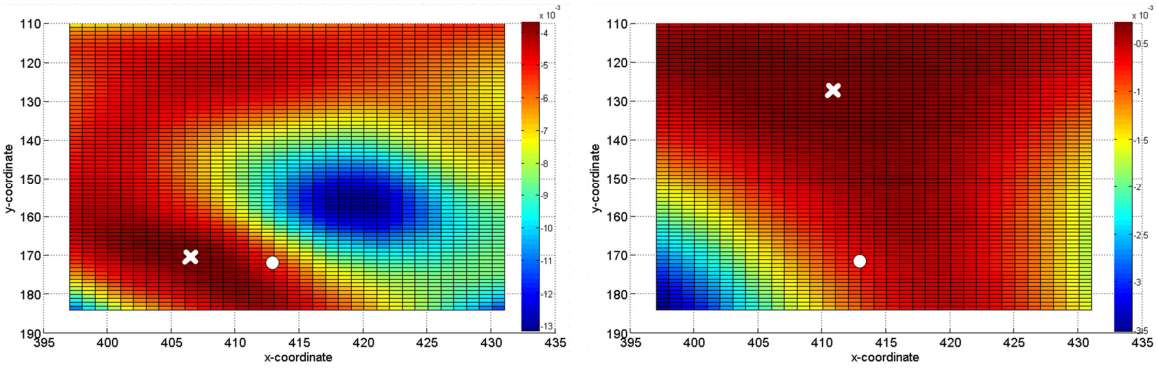
3.4.b Similarity Map

The proposed method considers multiple kernels with constraints to make the solution feasible during the optimization procedure. Take the same frame in Fig. 3.13 as an example, Fig. 3.14 (a) is the similarity map constructed under single kernel based on the negative value of the K-L divergence around the neighborhood area of the target. One can see that there is a large flat region with the similar values (dark red) which makes it

hard to locate the local maximum of the map. The local optimum, marked as a white cross, is far away from the ground truth, marked as a white dot, which causes the tracking error. In Fig. 3.14 (b), the same map is built based on the multiple kernels with constraints (Eq. (3.7)). The local optimum locates in the smaller region at the lower left corner, and the variation of the map values are more obvious which makes it suitable for gradient-based optimization. During the optimization iteration, it prevents the divergence of the tracking. It is not surprising to see that the estimated optimum (white cross) does not perfectly coincide with the ground truth. In practice, due to the appearance variation of the target, presence of the noise, and especially in the video with a lot of occlusion, we do not expect the experimental results can match the ground truth perfectly. However, it shows they are close enough to perform the successful tracking. In short, although our proposed method utilizes simple elements, e.g., HSV histogram and K-L divergence, the multiple constrained kernels effectively build a similarity map that enables us to do the successful tracking.

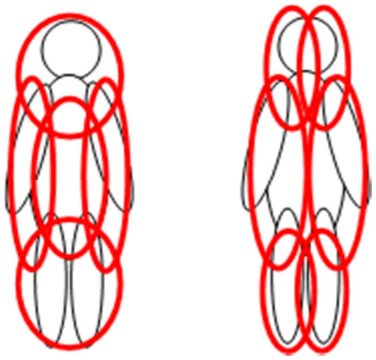
3.4.c Kernel Design

When we design the configuration of the kernels, the overlapping between a pair of the



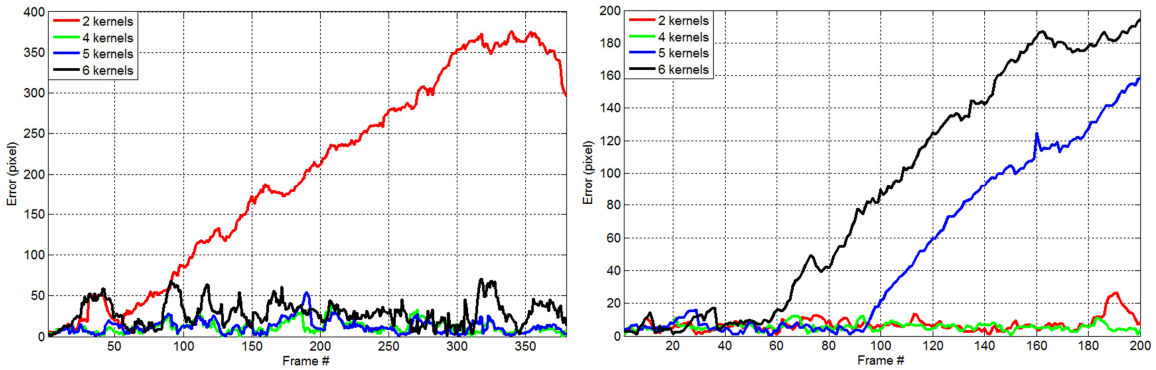
(a) (b)

Figure 3.14: Similarity map of the neighborhood area of Fig. 3.13. White dot is the ground truth of the target location. White cross is the local optimum of the map. (a) Compute the map based on single kernel. (b) Compute the map based on the multiple kernels with constraints.



(a) (b)

Figure 3.15: Layout for the multiple kernels. (a) 5 kernels. (b) 6 kernels.



(a) (b)

Figure 3.16: Errors (pixels) of the object center against the ground truth. Results of using different number of kernels are shown in different colors. Red: two kernels; Green: four kernels; Blue: five kernels; Black: six kernels. (a) i-LIDS Database. (b) Crowd Analysis Dataset.

kernels is allowed. However, the overlapping area between any pair of the kernels should be limited; otherwise, if the occlusion happens, it would be easily for both kernels to suffer from the ill-observable condition. For human tracking, it is intuitive to design two kernels, one for the upper part and one for the lower part, since (i) It is reasonable to assume that the human remains upright when they are moving, so the geometrical relationship of upper and lower body parts rarely changes. (ii) We have observed that for many cases of the occlusion between humans, the occlusion happens from the bottom of

the human body. When the lower part is occluded, the upper part may still be well-observed which helps track the target correctly. For more complex scenarios (e.g. Fig. 3.6 (c)) that the occlusion happens from different directions, four kernels are adopted for handling the occlusion from various directions. Theoretically, with more kernels representing the target, more information is available to enable the better tracking, especially in the crowded scene environment. We compare the results from using different numbers of kernels. The layouts of the five and six kernels are shown in Fig. 3.15. Fig. 3.16 (a) shows the tracking errors of *i-LIDS* video (Fig. 3.6 (c)) under different numbers of kernels. In this scenario, the scene is more cluttered and the target is usually occluded from the various directions. The one with two kernels loses the target while the ones with four, five and six kernels successfully locate the target. However, due to the different sizes of the targets, the number of kernels should be bounded. If we assign more kernels to the target, each kernel will cover only a small portion of the target (assume there are only limited overlapping areas between kernels). The extracted histograms may not be informative enough due to insufficient pixels, especially when the target is small, like the scenario in Fig. 3.6 (d). Fig. 3.16 (b) shows the tracking errors of *Crowd Analysis* video (Fig. 3.6 (d)). One can see the error of four-kernel is slightly better than the error of two-kernel, but when we add more kernels, the performance is worse due to the information deficiency for some kernels. In this case, instead of using statistical-based feature, other features may be included with more kernels setting.

Chapter 4 – Multiple-Camera Tracking with Overlapping Views

In our work, there are two different scenarios in dealing with tracking across multiple cameras (Fig. 1.1). This chapter introduces our approach for multiple-camera tracking with overlapping views.

4.1 Overview of the System

Suppose there are N_c cameras C_i, $i = 1 \sim N_c$; for each pair of cameras with overlapping views (e.g. C_1 and C_2 in Fig. 4.1), the spatial relationship can be built by constructing the FOV lines. This task can be done automatically by employing a feature detection and matching technique [55]. After finding the correspondence points between two views, we can determine the ground plane homography, which is the coordinate transform in two images from different views [1]. In this way, we can obtain the FOV lines between two views; i.e., where the boundaries of one view in the other view are located. In the mean time, the overlapping area in both views can thus be identified automatically (see Fig. 4.1) [18], and the visibility map $V_i^j(\cdot,\cdot)$ between a pair of cameras can be established. Adopting the notation from [50], let $V_i^j(x,y) = 1$ denote the fact that the location at (x,y) in camera C_i is also visible in camera C_j; otherwise, $V_i^j(x,y) = 0$. If the *n-th* person in camera C_i is located at (u,v), the camera subset that also sees this *n-th* person can be expressed as

$$C_i(n) = \{C_j \mid V_i^j(u,v) = 1, \forall j \neq i\}. \tag{4.1}$$

Moreover, the FOV line along side s of camera C_i showing in camera C_j is defined as $\ell_j^{i,s}$ and $s = \{left, top, right, bottom\}$.

For camera C_i, when the *n-th* person O_i^n enters the view from side s, the system decides if this position is in the overlapping area by building the set $C_i(n)$. If it only appears in

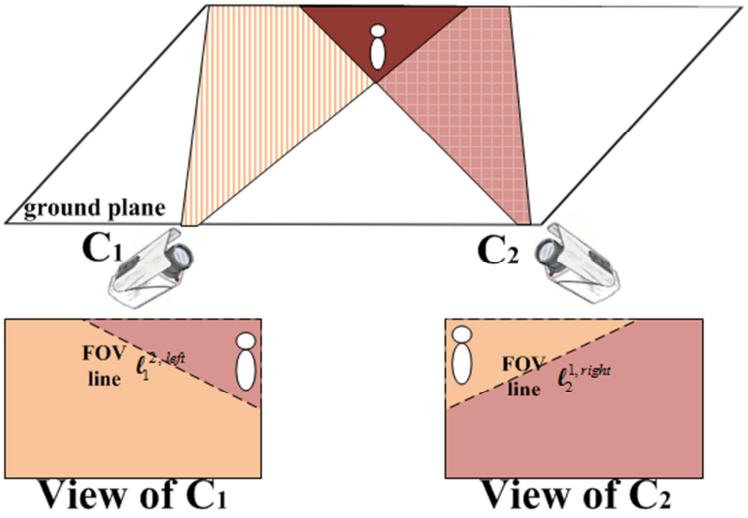


Figure 4.1: Camera topology and FOV lines

camera C_i, the set $C_i(n)$ is empty, and the system gives it a new global label. Otherwise, the person O_i^n can be seen by the other cameras in the set $C_i(n)$. Then the goal is how to search and compare all the people in $C_j \in C_i(n)$ in order to assign the same label to the person. In our work, three kinds of clues are effectively exploited. As introduced in Chapter 2.2.a, the camera link model consists of several components that allow us to utilized different features to match the humans in two cameras. Here, the camera link model includes the homography matrix, brightness transfer function, and tangent transfer function. In the next section, we describe how they are applied to the associated features.

4.2 Feature Representation and Transfer Function

4.2.a Vicinity Cue

The distance from a person to an FOV line is the first clue that can be used to remove impossible people in $C_j \in C_i(n)$ [50]. In each camera $C_j \in C_i(n)$, we form the candidate set in camera C_j,

$$CS_j^{i,n,s} = \{O_j^k \mid dis_v(O_j^k, \ell_j^{i,s}) < T_d, \forall\, O_j^k \; in \;\; C_j\}, \tag{4.2}$$

where $dis_v(O_j^k, \ell_j^{i,s})$ is the minimum distance between feet location of the *k-th* tracked object in camera C_j and the FOV line along side s of camera C_i showing in camera C_j. T_d

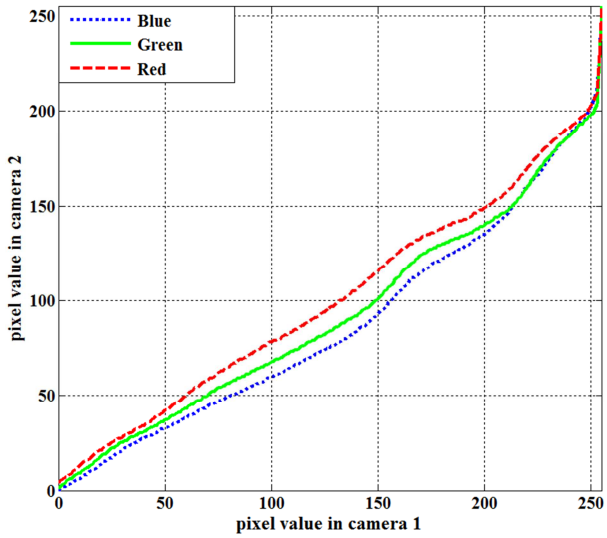


Figure 4.2: Brightness transfer function (BTF). Three channels are shown separately.

is a predefined threshold based on the scene. This set discards those people far away from the entering FOV line. We keep the distance values of the ones close to the FOV line for the matching later. If we use the homography to transform the feet location to its corresponding point in the other view, it may not give us the exact location due to the imperfection of the homography and the scale difference of two views. Especially when there are several candidates, it would be difficult to make the association. Moreover, since there may be several people whose distances to the FOV line are similar, we cannot only count on the vicinity cue. Therefore, besides the cues related to spatial location, more information is necessary.

4.2.b Color Cue

A color histogram can serve as a robust clue when searching for the best match among the candidate set $CS_j^{i,n,s}$ created from the previous step. However, even the same object may be seen differently due to the cameras' color difference. The color deviation can be modeled as a brightness transfer function (BTF) [30][87]. The BTF is applied to compensate for the color difference between two cameras before we compute the distance between the color histograms of two observations. For each two cameras with overlapping FOVs, we use the cumulative histograms of the overlapping area to compute the

brightness transfer function, $f_{BTF}(\cdot): \mathbb{R}^d \rightarrow \mathbb{R}^d$, where d is the dimension of a histogram vector. One example of the BTF is shown in Fig. 4.2, where the color correspondences are displayed for all the three channels. We can interpret that brightness is darker in camera 2 than in camera 1. For each object O_j^k in the candidate set $CS_j^{i,n,s}$, we evaluate the distance between the histograms of O_j^k and O_i^n, denoted by $\mathbf{h}_j^k$ and $\mathbf{h}_i^n \in \mathbb{R}^d$, respectively, with the BTF involved. The distance is denoted as $dis_c(\mathbf{h}_j^k, f_{BTF}(\mathbf{h}_i^n))$, where $f_{BTF}(\mathbf{h}_i^n)$ means the histogram of the *n-th* object in camera C_i after applying the BTF.

4.2.c Edge Cue

Assuming the cameras are synchronized, the edge feature is an effective feature for re-identification. However, the edge direction on a person may also change due to the difference of perspective. Here we proposed the transformation between the edge directions (i.e., the tangent values). We denote the coordinates of the views of two cameras as (x_i, y_i) and (x_j, y_j); the homography transform obtained in the FOV line detection stage can be shown as

$$\begin{bmatrix} \lambda x_j \\ \lambda y_j \\ \lambda \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}. \tag{4.3}$$

We can rewrite (4.3) into an explicit form:

$$\lambda = gx_i + hy_i + 1, \tag{4.4}$$

$$x_j = (ax_i + by_i + c)/\lambda = (ax_i + by_i + c)/(gx_i + hy_i + 1), \tag{4.5}$$

$$y_j = (dx_i + ey_i + f)/\lambda = (dx_i + ey_i + f)/(gx_i + hy_i + 1), \tag{4.6}$$

Taking the derivative of (4.5) and (4.6) with respect to x_i,

$$\frac{\partial x_j}{\partial x_i} = \frac{\left(a+b\frac{\partial y_i}{\partial x_i}\right)(gx_i+hy_i+1)-(ax_i+by_i+c)\left(g+h\frac{\partial y_i}{\partial x_i}\right)}{(gx_i+hy_i+1)^2}, \tag{4.7}$$

$$\frac{\partial y_j}{\partial x_i} = \frac{\left(d+e\frac{\partial y_i}{\partial x_i}\right)(gx_i+hy_i+1)-(dx_i+ey_i+f)\left(g+h\frac{\partial y_i}{\partial x_i}\right)}{(gx_i+hy_i+1)^2}. \tag{4.8}$$

Dividing (4.8) by (4.7), we get

$$\frac{\partial y_j}{\partial x_j} = \frac{\left(d+e\frac{\partial y_i}{\partial x_i}\right)(gx_i+hy_i+1)-(dx_i+ey_i+f)\left(g+h\frac{\partial y_i}{\partial x_i}\right)}{\left(a+b\frac{\partial y_i}{\partial x_i}\right)(gx_i+hy_i+1)-(ax_i+by_i+c)\left(g+h\frac{\partial y_i}{\partial x_i}\right)} = \frac{\left(d+e\frac{\partial y_i}{\partial x_i}\right)-y_j\left(g+h\frac{\partial y_i}{\partial x_i}\right)}{\left(a+b\frac{\partial y_i}{\partial x_i}\right)-x_j\left(g+h\frac{\partial y_i}{\partial x_i}\right)}. \tag{4.9}$$

Let m_i and m_j denote the tangent value $\frac{\partial y_i}{\partial x_i}$ and $\frac{\partial y_j}{\partial x_j}$, respectively, and (4.9) becomes

$$m_j = \frac{(d+em_i)-y_j(g+hm_i)}{(a+bm_i)-x_j(g+hm_i)}. \tag{4.10}$$

Since (4.10) enables us to map the tangent value from one camera to the other, we call it the tangent transfer function (TTF). The tangent value of a point is obtained from the gradient vector extracted by the Sobel operator based on the fact that the tangent direction is always perpendicular to the gradient direction. Given a point (x_i, y_i) in camera C_i and the corresponding tangent value m_i, we apply (4.3) and (4.10) to get tangent value m_j. We use the histogram of oriented tangent (HOT) as our feature to express edge directions. Hence, we can get the tangent value at each point on the object after transferring to the other camera, and the HOT can be built. Comparison is made by computing the distance $dis_e(\mathbf{t}_j^k, \widetilde{\mathbf{t}}_i^n)$, where $\mathbf{t}_j^k \in \mathbb{R}^r$ is the HOT of object O_j^k, $\widetilde{\mathbf{t}}_i^n$ denotes the HOT of *n-th* object in camera C_i after applying the TTF, and r is the dimension of the histogram.

In our simulation, we assume the cameras are set up at the high elevation places, which is the normal set up in public places, so the object height is much smaller than the distance between the ground plane and the camera. Hence, the ground plane homography can be utilized as the approximation as the global homography between two views [86]; that is, all the pairs of corresponding points share the same homography matrix. However, the performance can also be enhanced if multiple layers of planar homographies are employed. In this way, for each point on the object, we can choose the homography corresponding to the plane on which the point lies in order to obtain the tangent value transformation. The multiple layers homographies have also been considered in several works [3] [51].

4.3 Object Matching and Transfer Function Update

4.3.a Object Matching

Due to the quantization of the histogram and the imperfection of the transfer function, the histogram after applying transfer function will not match perfectly to the correct one. To alleviate this effect, the Earth Mover's Distance (EMD) [70] is chosen to compute the distance between two histograms since it can tolerate well some amount of deformations that shift features in the feature space. Here, we adopt the efficient EMD algorithm [54] in order to have robust comparison and computation efficiency as well.

By combining the three cues above, we can write our likelihood function and determine the best match by selecting the one with maximum likelihood among all $O_j^k \in CS_j^{i,n,s}$:

$$\widehat{K} = \arg\max_k [\exp(-(\frac{dis_v\left(O_j^k, \ell_j^{i,s}\right)^2}{{\sigma_v}^2} + \frac{dis_c\left(\mathbf{h}_j^k, f_{BTF}(\mathbf{h}_i^n)\right)^2}{{\sigma_c}^2} + \frac{dis_e\left(\mathbf{t}_j^k, \widetilde{\mathbf{t}_i^n}\right)^2}{{\sigma_e}^2}))]. \quad (4.11)$$

Therefore, we assign the label of the $\widehat{K}$-*th* object in camera C_j to the *n*-*th* object in camera C_i . The parameters are determined empirically as $\sigma_v = 15, \sigma_c = 2, \sigma_e = 2$ in our experiments.

4.3.b Camera Link Model Update

The environment change may lead to the necessity of the modification of the model, e.g., change of the lighting condition requires a different BTF. Therefore, in the testing stage, the camera link model needs to be continuously updated. For each pair of cameras with overlapping views, whenever a matched pair is detected, the information of this pair can be utilized to update the camera link model.

An instant brightness transfer function $f_{BTF}^{instant}$ is constructed and included in the updated BTF as (4.12).

$$f_{BTF}^{new} = (1-\rho) f_{BTF}^{old} + \rho f_{BTF}^{instant}, \quad (4.12)$$

where the update rate ρ is predetermined based on the environment. If the lighting condition changes fast, we can set the rate higher.

The original homography matrix is extracted from the initial stage described in the beginning of Chapter 4.1. Since there may be noise that affects the accuracy of the homography matrix computation, the matrix needs to be refined during tracking procedure in order to build a more reliable TTF. As long as a matching pair with high enough likelihood value in (4.11) is identified, we use this positive result to update the matrix, since they have high probability to be the correct correspondence objects. Assume the original transform matrix is obtained from two set of corresponding matching feature points with size *M* in two camera views:

$$P = \{(x_i, y_i),\ i = 1{\sim}M\},\ Q = \{(x_i', y_i'),\ i = 1{\sim}M\}, \tag{4.13}$$

where the point (x_i, y_i) in camera 1's frame matches the point (x_i', y_i') in camera 2's frame. Suppose we have a positive result; that is, the object at $(\check{x}_k, \check{y}_k)$ in camera 1 is labeled as the same object at $(\check{x}_k', \check{y}_k')$ in camera 2 with high enough likelihood in (4.11), we add this pair to the original set. Hence, we have the a set of matching points with size *M+1*. The updated homography can be extracted by using the least square method [1]. Since the positive result is the location that the object really passes by, it is highly probable that other objects may pass by that point in the future. By taking this into account, the homography transform matrix is more reliable; therefore, the TTF would make less deviation around those positions.

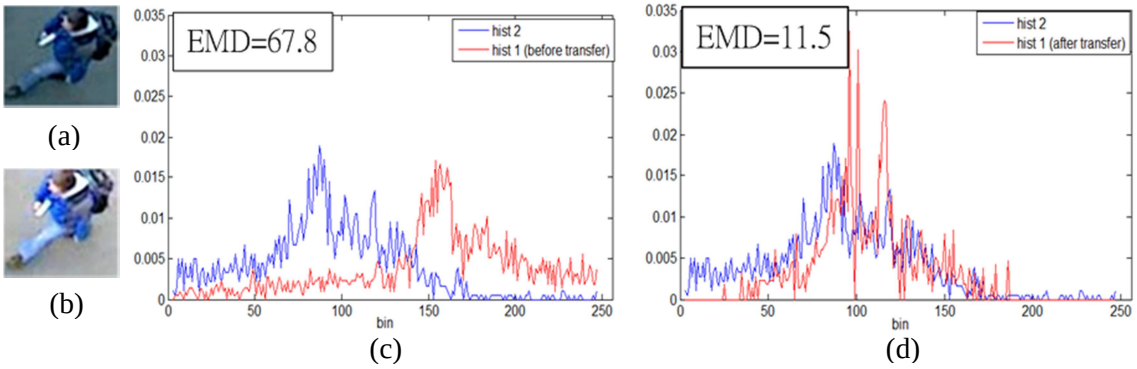
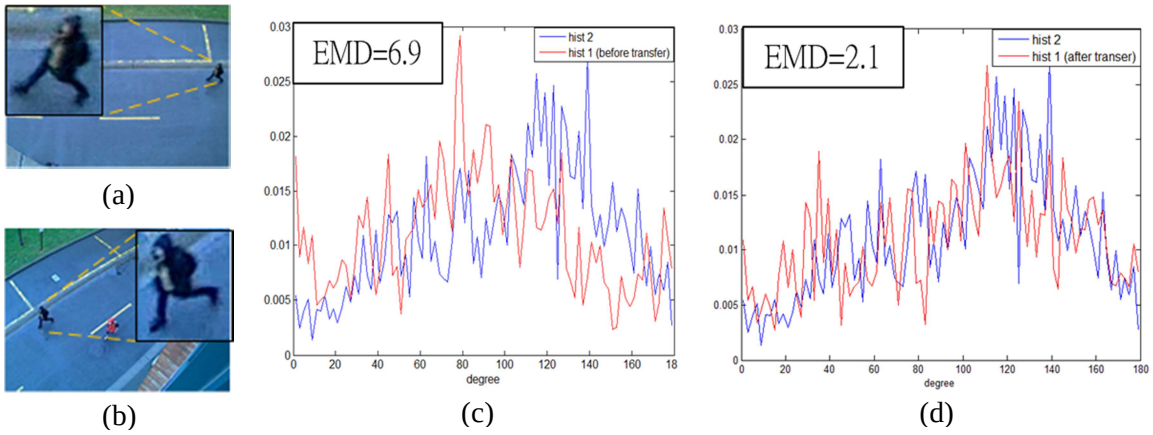


Figure 4.3: Histogram comparison (only one channel is shown here). (a) and (b) show the same person observed in different cameras. It is the same person as the one with ID 2 (green box) in the Fig. 4.5 (c)(d). The hist2 (blue curve, corresponding to (a)) is unchanged. The hist1 (red curve, corresponding to (b)) before and after transfer are shown in (c) and (d), respectively. The distance is smaller after applying BTF.



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

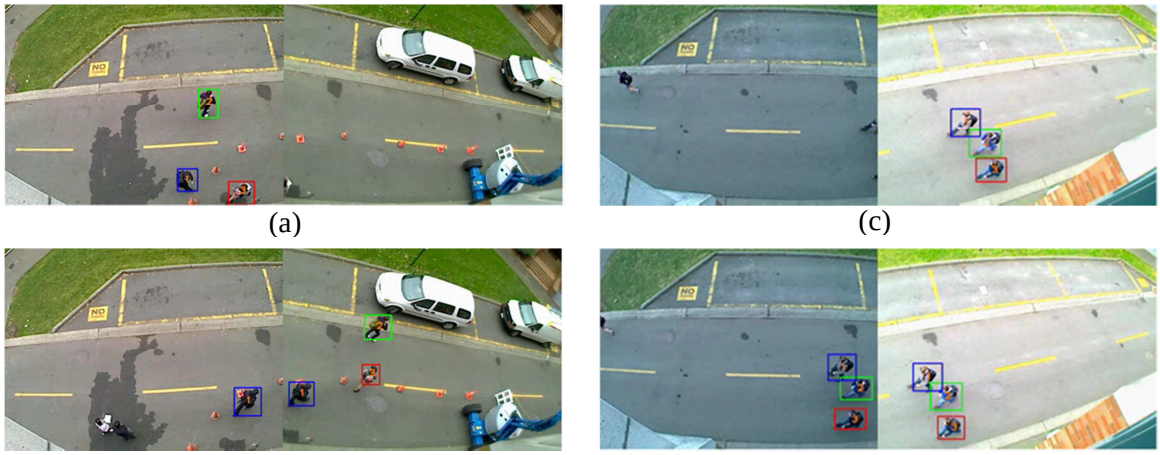
Figure 4.4: Histogram comparison. (a) View of camera 1. (b) View of camera 2. Note that the same objects are observed from different perspectives. (c)(d) Histogram comparison. The hist2 (blue curve, corresponding to (a)) is unchanged. The hist1 (red curve, corresponding to (b)) before and after transfer are shown in (c) and (d) respectively. The distance is smaller after applying TTF.

4.4 Experimental Results

In this section, we will first show the effectiveness of our histogram transfer functions including BTF and TTF. After that, we evaluate the accuracy of our multiple-camera tracking scheme based on several video clips. All the test videos are captured from two cameras with overlapping views from the high elevation places. The image size is 640x480. Some of the simulation results are shown to demonstrate the efficacious tracking across cameras by using the proposed method.

4.4.a Transfer Function Evaluation

The effectiveness of the transfer functions can be shown in the Fig. 4.3 and Fig. 4.4. We can see the distance between the histograms extracted from the same person in different views is much lower after applying the transfer functions. In our implementation, we use RGB color space, and each channel has 32 bins. The histograms from the three channels are concatenated together into a single vector. The edge direction is divided into 30 bins within the range from 0 to 180 degree. We increase the number of bins in the figures just for demonstration purpose.



(a) (c)

(b) (d)

Figure 4.5: Two overlapping views from two cameras are shown in the same row. (a) Three people are going to enter the right view. (b) All people appear in the right view and are labeled correctly. (c) Three people enter the left view in the same time. (d) All people appear in the left view and are labeled correctly.

(a) (b)

(c) (d)

Figure 4.6: Two overlapping views from two cameras are shown in the same row. The system successfully labels the same person with the same color bounding boxes in two cameras' views. (a)-(d) are four representative frames in chronological order.

4.4.b Tracking Across Cameras

The proposed method has been integrated in our single camera tracking system [20], which makes the tracking reliable even under occlusion or segmentation error. In this

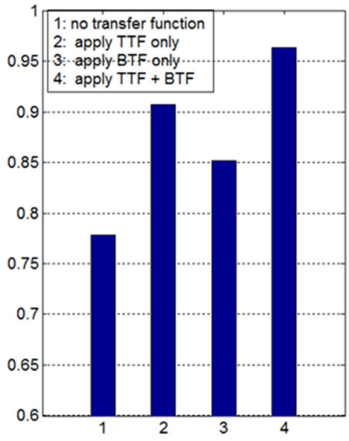


Figure 4.7: Matching accuracy when using three features. 1: Without transfer function. 2: Apply TTF only. 3: Apply BTF only. 4: Proposed method, apply TTF + BTF.

Table 4.1: Accuracy comparison

Method	**vicinity cue only [50]**	**vicinity cue + color cue (w/o BTF) [59]**	**vicinity cue + color cue + edge cue**
Accuracy	72%	79%	96%

section, we present the performance during matching. The video size in each camera is 640x480. Fig. 4.5 (a)(b) shows the result of tracking 3 people across the cameras, and all of them are labeled correctly. In order to make people differentiable, people are labeled with numbers and different colors of bounding boxes as well. In Fig. 4.5 (c)(d), the color difference between two cameras is even larger. Three people walk into the left view nearly at the same time. Since the geometrical distance to FOV line is similar, we count on the color and edge clues which lead us to the correct results. There are more people walking across the cameras views in Fig. 4.6. Some of them even pass the FOV line nearly at the same time which makes matching more challenging. Fig. 4.7 concludes the accuracy of the matching. The accuracy is defined as:

$$accuracy = \frac{\#\ of\ people\ correctly\ labeled}{\#\ of\ people\ walking\ across\ the\ cameras} \tag{4.14}$$

During the matching procedure, all the three cues are considered. We can see the performance is the best if we apply both transfer functions.

We also made the comparison between our proposed scheme and other two different approaches. In [50], the vicinity cue is the only feature considered during matching, and in [59], the color information without applying BTF is also utilized in matching stage. Table 4.1 shows the improvement of the accuracy by using our method.

One can see that the accuracy of using “Vicinity cue + Color cue (w/o BTF)” in Table 4.1 (0.79) is similar to its of using all the three cues without applying transfer functions in Fig. 4.7 (0.77). If the perspective difference between two cameras is large, such as Fig. 4.4, utilizing the edge cue without applying TTF will sometimes even deteriorate the accuracy. There are several videos with large perspective difference in our test set. It can explain why the accuracy is nearly the same after adding one more cue while the transfer functions are not employed.

Chapter 5 – Multiple-Camera Tracking with Nonoverlapping Views

In this chapter we present our approach for tracking humans across cameras with nonoverlapping views.

5.1 System Overview

The overall system shown in Fig. 5.1 is divided into the training stage and the testing stage:

5.1.a Training Stage

The system learns camera link models in an unsupervised manner. We consider several terms in the model, including transition time distribution, brightness transfer function, region mapping matrix, region matching weights, and feature fusion weights.

If there exists a path allowing people to travel between two cameras without passing through any other cameras, we call the two cameras directly-connected. As introduced in Chapter 2.2.a, a path actually connects two entry/exit zones in a pair of directly-connected cameras (see Fig. 5.4). Given the camera topology, the directly-connected camera pairs and the corresponding entry/exit zones can be manually identified easily. Training takes place in a pairwise fashion between all the corresponding entry/exit zones pairs of directly-connected cameras. For each pair of entry/exit zones, the training data, i.e., two observation sets from two zones, is collected. Each entry/exit observation contains temporal, color and texture features of a person who is entering/leaving the field of view (FOV) of a particular camera. If people manually identify the correct correspondences between the observations in these two sets, the camera link model can be straightforwardly estimated in a supervised manner. For example, the transition time values between pairs of correct correspondences can be computed based on the difference between the entry time stamps and exit time stamps, from which the transition time distribution can be estimated. However, as the scale of the camera network is getting larger, supervised learning of camera link models is not feasible since a large amount of

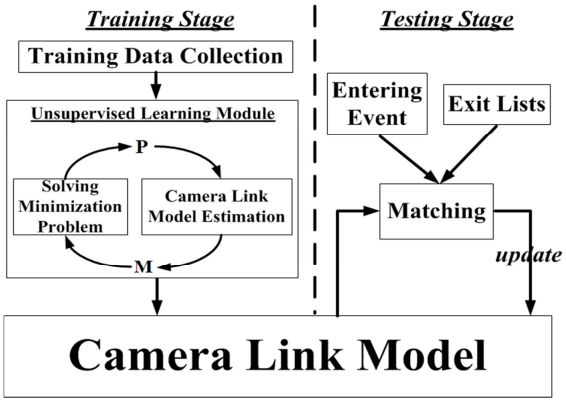


Figure 5.1: System overview

human effort is required. Hence, we learn the camera link model based on an unsupervised scheme [17] in the training stage. The system automatically identifies the correspondences in the training data and estimate the camera link mode.

Denote X and Y as the exit and entry observations, within a specific time window, from a pair of corresponding entry/exit zones in a pair of directly-connected cameras, respectively:

$$\mathbf{X} = [\mathbf{x}_1 \quad \cdots \quad \mathbf{x}_{N_1}],\ \mathbf{Y} = [\mathbf{y}_1 \quad \cdots \quad \mathbf{y}_{N_2}], \tag{5.1}$$

where $\mathbf{x}_i$ and $\mathbf{y}_j$ are exit and entry observations, and N_1 and N_2 are the numbers of the observations. Inspired by the concept of feature point matching between two images [16][56], we formulate the identification of the correspondence as finding an $(N_1 + 1) \times (N_2 + 1)$ binary matrix $\mathbf{P}$, where each row and column stands for an exit and entry observation, respectively. The entry P_{ij} in $\mathbf{P}$ is 1 if $\mathbf{x}_i$ corresponds to $\mathbf{y}_j$; otherwise, it is 0. The $(N_1 + 1)th$ row and the $(N_2 + 1)th$ column represent the outliers, i.e., an entry observation in one camera is not from the other, or an exit observation from one camera never enters the other. Note that $P_{N_1+1,\,N_2+1}$ has no physical meaning, so all the following discussion will exclude it automatically.

Analogous to the affine transformation in 2D image points matching [16][56], the camera link model serves as the transformation between the observations from two cameras. The correspondence matrix $\mathbf{P}$ is estimated based on the distances between the feature vectors from the observation sets, and the camera link model (transformation) is required to compensate for the deviation between the cameras during the distance

calculations. Moreover, the camera link model $\mathbb{M}$ can be estimated given a correspondence matrix **P**. Hence, it is reasonable to employ an EM-like approach to iteratively estimate the matrix **P** and the camera link model $\mathbb{M}$ (Fig. 5.1). In each iteration, **P** is obtained by solving a minimization problem given the most recently estimated $\mathbb{M}$. After that, model $\mathbb{M}$ is estimated based on the newly updated **P**. In Chapter 5.2 and 5.3, we will introduce the formulation of the minimization problem and the estimation of the camera link model including transition time distribution, brightness transfer function, region mapping matrix, region matching weights, and feature fusion weights. One will see that our estimation procedure is quite general. People can always add more features and the corresponding transformations in the camera link model. The unsupervised learning scheme will take care of the estimation of the unknown parameters adaptively and systematically.

5.1.b Testing Stage

In the testing stage, each camera C_i, $i = 1 \sim N_C$ maintains an exit list $\boldsymbol{L}_{i,k}$ for each entry/exit zone *k*, within a T_{max}-second interval. It consists of the observations $O_{i,k}$ of the people who have left the FOV from zone *k* within this T_{max}-second interval.

$$\boldsymbol{L}_{i,k} = \{O_{i,k}^1, O_{i,k}^2 \dots O_{i,k}^{|\boldsymbol{L}_{i,k}|}\} \tag{5.2}$$

Whenever a person enters a camera's view, the system finds the best match among the people in the exit lists corresponding to the linked zones of the directly-connected cameras. Based on the camera link model, the matching distance between two observations O_1 and O_2 can be computed as the weighted sum of distances:

$$match\_dist = \sum_{i=1}^{N_{feature}} \alpha_i \times feature\_dist_i\,(O_1, O_2), \tag{5.3}$$

where α_i is the weight for the distance $feature_{dist_i}(\cdot,\cdot)$ corresponding to the feature *i*. In our work, four different features ($N_{feature} = 4$), namely, temporal, holistic color, region color and region texture features, are utilized to compute distances. If the lowest distance is smaller than a certain threshold, the label handoff is performed; otherwise, we will treat it as a new person within the camera network. The re-identification results can be further used to update the camera link models.

5.2 Minimization Problem Formulation

Given the most recently estimated camera link model $\mathbb{M}$, the optimum correspondence matrix $\widehat{\mathbf{P}}$ can be obtained by solving a constrained minimization integer programming problem:

$$\widehat{\mathbf{P}} = \arg\min_{\mathbf{P}} J(\mathbf{P}) \tag{5.4}$$

$$\text{s.t.} \qquad P_{ij} \in \{0,1\} \quad \forall\, i \le N_1 + 1, j \le N_2 + 1, \tag{5.5}$$

$$\sum_{i=1}^{N_1+1} P_{ij} = 1 \;\; \forall\, j \le N_2, \qquad \sum_{j=1}^{N_2+1} P_{ij} = 1 \;\; \forall\, i \le N_1, \tag{5.6}$$

where $J(\cdot)$ is the objective function to be minimized. The constraints (5.5) and (5.6) enforce one-to-one correspondence (except for the outlier row and column). By incorporating the soft-assign [16] instead of hard decision all the time, the problem is relaxed by substituting constraint (5.5) with (5.7)

$$P_{ij} \ge 0 \quad \forall\, i \le N_1 + 1, j \le N_2 + 1. \tag{5.7}$$

In this way, the variables P_{ij} are continuous real numbers indicating how likely it is that the *i*-th exit and the *j*-th entry observations are a matched pair. Moreover, the relaxation reduces the chance of getting trapped in poor local minima during the optimization search. By incorporating the deterministic annealing method, the solution eventually converges at a binary permutation matrix [16]. The objective function $J(\cdot)$ comprises several cost functions, and each of them stands for a distance function between the exit and entry observations associated with one specific feature, e.g., time, color, texture. In the following, we will introduce the cost functions considered in the objective function and explain how a camera link model associates with different features.

5.2.a Temporal Feature

According to our observations, people tend to follow similar paths in most cases due to the presence of available pathways, obstructs, or shortest routes. Thus, the transition time t forms a certain distribution $f_{tran}(t)$. Given previously estimated $\mathbf{P}$, we can get the transition time values $\mathbf{T}_{tran} = \left[t_1^{exit} \quad \dots \quad t_{N_1}^{exit}\; t_1^{entry} \quad \dots \quad t_{N_2}^{entry}\right]$, where

$$t_i^{exit} = \sum_{j=1}^{N_2} P_{ij}(y_j^t - x_i^t) \qquad \forall\, i \le N_1, \tag{5.8}$$

$$t_j^{entry} = \sum_{i=1}^{N_1} P_{ij}(y_j^t - x_i^t) \qquad \forall\, j \le N_2, \tag{5.9}$$

and y_j^t and x_i^t represent the time stamps of the observations $\mathbf{y}_j$ and $\mathbf{x}_i$, respectively. The transition time is always positive if two cameras have no overlapping area, because the entry time of a person should be greater than the exit time of the correct correspondence from the other camera. Also, the outliers should lead to zero transition time. For example, if the *k*-th exit observation is an outlier, P_{kj} should be zero for all $j \le N_2$ resulting in $t_k^{exit} = 0$ according to (5.8). Hence, the optimal solution of **P** satisfies the constraints $t_i^{exit} \ge 0$, $t_j^{entry} \ge 0$ and $P_{ij} = 0$ *if* $y_j^t - x_i^t \le 0$, which are required to be included in the problem formulation. A set of valid time values $\mathbf{T}_{valid}$ that excludes the outliers, i.e., takes only the nonzero entries in $\mathbf{T}_{tran}$, is further established, and the transition time distribution $f_{tran}(\cdot)$ is built based on the kernel density estimation:

$$\mathbf{T}_{valid} = \{\hat{t} | \hat{t} \ne 0, \hat{t} \in \mathbf{T}_{tran}\} = [t_1 \quad \cdots \quad t_{N_{valid}}], \tag{5.10}$$

$$f_{tran}(t) = \frac{1}{N_{valid}} \sum_{i=1}^{N_{valid}} \frac{1}{\sigma_{tran}\sqrt{2\pi}} exp(-\frac{(t-t_i)^2}{2\sigma_{tran}^2}), \tag{5.11}$$

where σ_{tran}^2 is the predefined variance of the Gaussian kernel. For each possible correspondence, we compute the likelihood value $f_{tran}(y_j^t - x_i^t)$ given the model and consider the maximum likelihood estimation, i.e., use $(1 - f_{tran}(y_j^t - x_i^t))$ as the individual cost. Thus, the total cost can be written as:

$$\begin{aligned} cost_{time} &= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} P_{ij}(1 - f_{tran}(y_j^t - x_i^t)) \\ &= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} P_{ij} feature\_dist_1(\mathbf{x}_i, \mathbf{y}_j). \end{aligned} \tag{5.12}$$

5.2.b Holistic Color Feature

The same object may appear differently under two cameras with nonoverlapping views due to illumination changes and different camera color responses. The color deviation can be modeled as a brightness transfer function (BTF) [30]. One example of the BTF is shown in Fig. 4.2. The BTF is applied to compensate for the color difference between two cameras before we compute the distance between the holistic color histograms of two

observations. Thus, the total cost function for the holistic color feature is:

$$cost_{holistic\_color} = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij} D\left(f_{BTF}\left(\mathbf{y}_j^h\right), \mathbf{x}_i^h\right) = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij} feature\_dist_2(\mathbf{x}_i, \mathbf{y}_j), \quad (5.13)$$

where $\mathbf{y}_j^h \in \mathbb{R}^d$ and $\mathbf{x}_i^h \in \mathbb{R}^d$ are the holistic color histograms of the observations, $\mathbf{y}_j$ and $\mathbf{x}_i$; $D(\cdot)$ is the distance function between two histograms; $f_{BTF}(\cdot): \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the BTF, with *d* being the total bin number of the color histogram.

5.2.c Region Color and Texture Feature

Since the viewpoints vary in two cameras, some parts of the human body may only be seen in either one of the cameras' views. Hence, we divide the human into multiple regions for more detailed comparison. However, the corresponding regions do not always cover the same area of the human due to different viewpoints (see Fig. 5.2). We observe that the entering (or exiting) directions of different people at an entry/exit zone of a fixed camera are similar, so we use a mapping matrix to link the regions between two bodies. The histogram extracted from one region of human leaving from the first camera can be

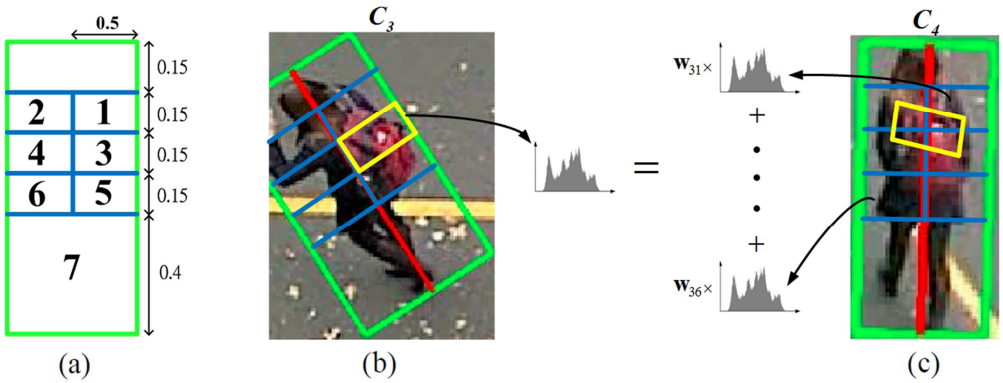


Figure 5.2: (a) Green box is the bounding box of the target. The target is divided into 7 regions (exclude the head) based on the shown ratios. Each region has predefined label number. (b) An exit observation in camera C_3. (c) An entry observation of the same person in camera C_4. The red line is the principal axis. Region 3 in (b) and region 3 in (c) do not cover the same areas. The yellow rectangles cover the same areas on the target. The histogram extracted from region 3 in (b) can be modeled as the linear combination of the histograms extracted from six regions in (c), and the coefficients are $\boldsymbol{w}_3 = [w_{31} \ldots w_{36}]^T$.

modeled as the linear combination of the histograms extracted from multiple regions of human entering into the second camera. The snapshots for a person exiting one camera and entering another are shown in Fig. 5.2 (b) and (c), respectively.

First, the principal axis of a person is identified by applying principal component analysis to the human silhouette which is obtained from the single camera tracking module [20]. After that, the whole body is automatically divided into head, torso, and leg regions based on the predefined ratios (Fig. 5.2(a)). We discard the head region in the region matching, since this part usually has lower discriminability due to its relatively small area and similar hair/face color information. The torso is further divided into six regions, and the mapping matrix will be trained for linking two six-regions from a pair of people. Because the leg region usually changes little under different perspectives, we compute the distance between the two whole leg regions without further dividing it. We denote the region color histograms extracted from the region k of the observations $\mathbf{x}_i$ and $\mathbf{y}_j$ as $\mathbf{x}_i^{rh_k} \in \mathbb{R}^d$ and $\mathbf{y}_j^{rh_k} \in \mathbb{R}^d$, respectively, where $k = 1\sim7$. As shown in Fig. 5.2, the regions 1 to 6 are from the torso, and region 7 is from the leg. We denote the mapping matrix $\mathbf{W}_{map} \in \mathbb{R}^{6\times6}$ as:

$$\mathbf{W}_{map} = [\mathbf{w}_1 \dots \mathbf{w}_6], \tag{5.14}$$

where $\mathbf{w}_k \in \mathbb{R}^6$ is the weighting for linear combination.

Moreover, since some regions may not be visible under both cameras' views, they should be assigned with smaller weights in the region feature distance computation. The cost function is the weighted sum of the distances from all 7 regions:

$$\begin{aligned} & cost_{region\_color} \\ &= \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij}\left[\sum_{k=1}^{6} q_k \times D\left(\mathbf{y}_j^{map_k}, \mathbf{x}_i^{rh_k}\right) + q_7 \times D\left(f_{BTF}(\mathbf{y}_j^{rh_7}), \mathbf{x}_i^{rh_7}\right)\right] \\ &= \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij} feature\_dist_3(\mathbf{x}_i, \mathbf{y}_j), \end{aligned} \tag{5.15}$$

where $\mathbf{y}_j^{map_k} \in \mathbb{R}^d$ is the linear combination of the torso region color histograms $\left[\mathbf{y}_j^{rh_1} \dots \mathbf{y}_j^{rh_6}\right] \in \mathbb{R}^{d\times6}$ with weights $\mathbf{w}_k$ after applying the BTF,

$$\mathbf{y}_j^{map_k} = \left[f_{BTF}(\mathbf{y}_j^{rh_1}) \dots f_{BTF}(\mathbf{y}_j^{rh_6})\right]\mathbf{w}_k, \tag{5.16}$$

and $\mathbf{q} = [q_1 \dots q_7]^T$ denote the weights for all 7 region distances. Note that all the seven regions are included in the distance computation, but only the torso regions are considered for the region mapping by using the mapping matrix $\mathbf{W}_{map}$.

The texture feature is considered in a similar manner. The local binary pattern (LBP) [61] is utilized as the texture feature and is expressed as *r*-dimensional LBP histograms $\mathbf{x}_i^{rLBPh_k} \in \mathbb{R}^r$ and $\mathbf{y}_j^{rLBPh_k} \in \mathbb{R}^r$, where $k = 1{\sim}7$. Hence, the cost function is:

$$
\begin{aligned}
&cost_{region\_texture} \\
&= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} P_{ij} \left[\sum_{k=1}^{6} q_k \times D\left(\mathbf{y}_j^{mapLBP_k}, \mathbf{x}_i^{rLBPh_k}\right) + q_7 \times D\left(\mathbf{y}_j^{rLBPh_7}, \mathbf{x}_i^{rLBPh_7}\right) \right] \\
&= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} P_{ij} feature\_dist_4(\mathbf{x}_i, \mathbf{y}_j),
\end{aligned} \tag{5.17}
$$

where $\mathbf{y}_j^{mapLBP_k} \in \mathbb{R}^r$ is the linear combination of torso region LBP histograms $\left[\mathbf{y}_j^{rLBPh_1} \dots \mathbf{y}_j^{rLBPh_6}\right] \in \mathbb{R}^{r \times 6}$ with weights $\mathbf{w}_k$.

$$
\mathbf{y}_j^{mapLBP_k} = \left[\mathbf{y}_j^{rLBPh_1} \dots \mathbf{y}_j^{rLBPh_6}\right] \mathbf{w}_k \tag{5.18}
$$

Since the LBP is robust to the brightness change [61], the BTF is not applied here.

5.2.d Maxima Entropy Principle

In deterministic annealing, a widely used iterative scheme to solve optimization problems, the procedure starts with emphasizing high "uncertainty", measured as the entropy, of the entries in $\mathbf{P}$, i.e., to maximize the entropy. The importance of the maximum entropy principle is gradually decreased by increasing a parameter β [16][69] through the iterations. Thus, the cost function is written as the negative of the entropy:

$$
cost_{entropy} = \frac{1}{\beta} \sum_{i=1}^{N_1+1} \sum_{j=1}^{N_2+1} P_{ij} \log P_{ij}. \tag{5.19}
$$

In the early stage of the training process, the factor β starts with a low value that raises the importance of this cost function with respect to the overall objective function $J(\mathbf{P})$, enabling the value P_{ij} to move freely in the space for searching the optimum. β is then gradually increased to lower the importance and eventually leads to convergence. The function (5.19) can also be seen as a barrier function [10] for the constraint defined in (5.7). More discussions about the parameter β are in Chapter 5.6.

5.2.e Outlier Cost

Since the presence of outliers is considered, there is an additional term which controls how large the distance we can tolerate before treating a particular observation as an outlier. The penalty term is:

$$cost_{outlier} = -\theta \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} P_{ij}, \tag{5.20}$$

where θ is a control factor. If θ is set large, for instance, the estimation process is allowed to tolerate larger distance before treating one as an outlier. More discussions about the parameter θ are in Chapter 5.6.

5.2.f Estimation of Correspondence Matrix

By incorporating all the cost functions above, our final problem formulation becomes a convex optimization problem:

$$\widehat{\mathbf{P}} = \arg\min_{\mathbf{P}} J(\mathbf{P}) \tag{5.21}$$

$$\text{s.t.} \qquad P_{ij} \geq 0 \quad \forall\, i \leq N_1 + 1, j \leq N_2 + 1, \tag{5.22}$$

$$\sum_{i=1}^{N_1+1} P_{ij} = 1 \;\; \forall\, j \leq N_2, \qquad \sum_{j=1}^{N_2+1} P_{ij} = 1 \;\; \forall\, i \leq N_1, \tag{5.23}$$

$$t_i^{exit} = \sum_{j=1}^{N_2} P_{ij}(y_j^t - x_i^t) \geq 0 \qquad \forall\, i \leq N_1, \tag{5.24}$$

$$t_j^{entry} = \sum_{i=1}^{N_1} P_{ij}(y_j^t - x_i^t) \geq 0 \qquad \forall\, j \leq N_2, \tag{5.25}$$

$$P_{ij} = 0 \;\; if \;\; y_j^t - x_i^t \leq 0 \qquad \forall\, i \leq N_1, j \leq N_2. \tag{5.26}$$

The objective function $J(\mathbf{P})$ is the combination of the above cost functions:

$$\begin{aligned} J(\mathbf{P}) = {} & cost_{time} + cost_{holistic\_color} + cost_{region\_color} \\ & + cost_{region\_texture} + cost_{entropy} + cost_{outlier}. \end{aligned} \tag{5.27}$$

Given the current camera link model $\mathbb{M} = \{f_{tran}, f_{BTF}, \mathbf{W}_{map}, \mathbf{q}\}$, the objective function is formulated, and $\mathbf{P}$ is updated by solving (5.21)~(5.27). Instead of incorporating the barrier function for constraints (5.24) and (5.25) as is done in our previous work [17], we use a projection-based convex optimization method [10] to deal with the constraints (5.24)~(5.26) since it leads to faster convergence in practice. First of all, given the current estimated camera link model, the objective function $J(\mathbf{P})$ is convex in $\mathbf{P}$, so the optimum can be obtained by equating the derivative to zero,

$$\frac{\partial J(\mathbf{P})}{\partial P_{ij}} = 0 \quad \rightarrow \quad P_{ij} = \begin{cases} e^{(\beta(\theta - dist_{ij})-1)} & \text{if } \ i \leq N_1 \text{ and } j \leq N_2, \\ e^{-1} & \text{if } \ i = N_1 + 1 \text{ or } \ j = N_2 + 1 \end{cases} \tag{5.28}$$

where $dist_{ij} = \sum_{k=1}^{N_{feature}} feature\_dist_k\,(\mathbf{x}_i, \mathbf{y}_j)$ is the sum of distances. After that, we project the solution onto the feasible space in which all the constraints are satisfied. Specifically, (i) Assign the zero values to particular elements to satisfy the constraints (5.24)~(5.26), like

$$P_{ij} = \begin{cases} 0, & y_j^t - x_i^t \leq 0, \ \ \forall\, i \leq N_1 \text{ and } j \leq N_2 \\ P_{ij}, & otherwise \text{ (include outlier row and column)} \end{cases}. \tag{5.29}$$

(ii) Perform alternately row-column normalization based on Sinkhorn's theorem [16][71] for constraints (5.23). (iii) The cost function (5.19) (maximum entropy) can be seen as a barrier function for the constraint (5.22) that makes the solution exponential as shown in (5.28), which is always nonnegative, so that the constraint (5.22) always holds. In this way, **P** can be updated based on the most recently estimated camera link model $\mathbb{M}$ and the current value of β.

5.3 Camera Link Model Estimation

The camera link model $\mathbb{M}$ is estimated given the newly updated correspondence matrix **P**. Since each P_{ij}, in the range from 0 to 1, indicates how likely the *i*-th exit and the *j*-th entry observations are a matched pair, the whole estimation process is based on the probability concept, i.e., the soft decision is made instead of the hard decision.

5.3.a Estimate Transition Time Distribution

Given the current **P**, the set of transition time $\mathbf{T}_{valid} = [t_1 \quad \cdots \quad t_{N_{valid}}]$ is established via (5.8)~(5.10), and the $f_{tran}(\cdot)$ is estimated based on (5.11).

5.3.b Estimate Brightness Transfer Function

Two histograms built from the corresponding people in two cameras are used to estimate $f_{BTF}(\cdot)$ [30][66]. To find the corresponding cumulative histograms, which have been shown to be effective for building the BTF [30][66], we calculate the weighted sum

among all the histograms from observation sets **X** and **Y** separately, where the weights are set as value P_{ij}.

$$\mathbf{h}_{\mathbf{x}}^{holistic} = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij}\mathbf{x}_i^h, \quad \mathbf{h}_{\mathbf{y}}^{holistic} = \sum_{j=1}^{N_2}\sum_{i=1}^{N_1} P_{ij}\mathbf{y}_j^h. \tag{5.30}$$

In addition to the holistic color histograms, the region color histograms are also considered.

$$\mathbf{h}_{\mathbf{x}}^{region} = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij}\sum_{k=1}^{7} q_k\mathbf{x}_i^{rh_k}, \tag{5.31}$$

$$\mathbf{h}_{\mathbf{y}}^{region} = \sum_{j=1}^{N_2}\sum_{i=1}^{N_1} P_{ij}(\sum_{k=1}^{6} q_k\left[\mathbf{y}_j^{rh_1} \dots \mathbf{y}_j^{rh_6}\right]\mathbf{w}_k + q_7 \times \mathbf{y}_j^{rh_7}). \tag{5.32}$$

Note that the region matching weights **q** is considered here because not all the regions are well-observable under two views. Thus, two cumulative histograms $\mathbf{h}_{\mathbf{x}}$ and $\mathbf{h}_{\mathbf{y}}$ are obtained:

$$\mathbf{h}_{\mathbf{x}} = \mathbf{h}_{\mathbf{x}}^{holistic} + \mathbf{h}_{\mathbf{x}}^{region}, \quad \mathbf{h}_{\mathbf{y}} = \mathbf{h}_{\mathbf{y}}^{holistic} + \mathbf{h}_{\mathbf{y}}^{region}. \tag{5.33}$$

In this way, the histograms of the outliers will not be considered, and the histograms of the corresponding parts are included in the cumulated histograms $\mathbf{h}_{\mathbf{x}}$ and $\mathbf{h}_{\mathbf{y}}$. After normalizing the histograms, $\mathbf{h}_{\mathbf{x}}$ and $\mathbf{h}_{\mathbf{y}}$ are used to estimate $f_{BTF}(\cdot)$ [30][66].

5.3.c Estimate Region Mapping Matrix

To estimate $\mathbf{W}_{map}$, both the region color and region texture features are utilized, and BTF should be applied when dealing with color features. Let the region color and texture histogram vectors of the *j*-th entry and the *i*-th exit observations be concatenated in the following way:

$$\mathbf{A}_j = \begin{bmatrix} f_{BTF}(\mathbf{y}_j^{rh_1}) \dots f_{BTF}(\mathbf{y}_j^{rh_6}) \\ \mathbf{y}_j^{rLBPh_1} \dots \mathbf{y}_j^{rLBPh_6} \end{bmatrix} = \left[\mathbf{a}_{j1} \dots \mathbf{a}_{j6}\right] \quad \forall j \le N_2\,, \tag{5.34}$$

$$\mathbf{B}_i = \begin{bmatrix} \mathbf{x}_i^{rh_1} \dots \mathbf{x}_i^{rh_6} \\ \mathbf{x}_i^{rLBPh_1} \dots \mathbf{x}_i^{rLBPh_6} \end{bmatrix} = [\mathbf{b}_{i1} \dots \mathbf{b}_{i6}] \quad \forall i \le N_1, \tag{5.35}$$

where $\mathbf{A}_j \in \mathbb{R}^{(d+r)\times 6}$ and $\mathbf{B}_i \in \mathbb{R}^{(d+r)\times 6}$; $\mathbf{a}_{jk} \in \mathbb{R}^{(d+r)}$, $\mathbf{b}_{ik} \in \mathbb{R}^{(d+r)}$ for $k = 1\sim 6$. The leg region, denoted as region 7, with the feature vectors $\mathbf{a}_{j7}$ and $\mathbf{b}_{i7}$, is not considered here because the mapping matrix is mainly for the torso region. We want to minimize the

weighted sum of the mapping error with the weights $\{P_{ij}\}$:

$$\mathbf{W}_{map} = \arg\min_{\underline{\mathbf{W}}} \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij}\left\|(\mathbf{A}_j\underline{\mathbf{W}} - \mathbf{B}_i)\mathbf{Q}\right\|_F^2 \tag{5.36}$$

$$\text{s.t.} \quad \underline{\mathbf{W}} = [\underline{\mathbf{w}}_1 \dots \underline{\mathbf{w}}_6],\ \underline{\mathbf{w}}_k \in \mathbb{R}^6,\ \mathbf{Q} = \begin{bmatrix} \sqrt{q_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{q_6} \end{bmatrix},$$

$$\underline{\mathbf{w}}_k \geq \mathbf{0},\ \left\|\underline{\mathbf{w}}_k\right\|_1 = 1 \quad \forall\, k = 1\sim 6,$$

where $\|\cdot\|_F$ stands for the Frobenius norm, and the current region matching weights $\mathbf{q}$ is used to form $\mathbf{Q}$. After some manipulations, it is equivalent to solve each vector $\underline{\mathbf{w}}_k$ separately by minimizing another objective function (5.37) toward each $\underline{\mathbf{w}}_k,\ k = 1\sim 6$:

$$\mathbf{w}_k = \arg\min_{\underline{\mathbf{w}}_k}\ g_{\mathbf{W}}(\underline{\mathbf{w}}_k) \tag{5.37}$$

$$\text{s.t.} \quad g_{\mathbf{W}}(\underline{\mathbf{w}}_k) = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij}\left\|\mathbf{A}_j\underline{\mathbf{w}}_k - \mathbf{b}_{ik}\right\|_2^2,$$

$$\underline{\mathbf{w}}_k \geq \mathbf{0},\ \left\|\underline{\mathbf{w}}_k\right\|_1 = 1.$$

Note that the factor q_k is omitted since the vectors $\underline{\mathbf{w}}_k$ can be solved separately without involving q_k.

5.3.d Estimate Region Matching Weights

The weights for different regions are determined by the Matcher Weighting method [72], where the weights are assigned based on the errors of the estimation results. Define the estimation error of the mapping vector $\mathbf{w}_k$ as

$$\varepsilon_k^{error} = g_{\mathbf{W}}(\mathbf{w}_k) \quad k = 1\sim 6, \tag{5.38}$$

$$\varepsilon_7^{error} = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij}\left\|\mathbf{a}_{j7} - \mathbf{b}_{i7}\right\|_2^2, \tag{5.39}$$

where $g_{\mathbf{W}}(\mathbf{w}_k)$ is shown in (5.37). Then $\mathbf{q} = [q_1 \dots q_7]^T$ is calculated as

$$q_k = \frac{1/\varepsilon_k^{error}}{\sum_{m=1}^{7} 1/\varepsilon_m^{error}} \quad k = 1\sim 7. \tag{5.40}$$

The weights q_k are inversely proportional to the corresponding estimation error. If the error is low, it means the region is well-observable in both views, which results in small region matching error, so the weight should be large.

5.3.e Estimate Feature Fusion Weights

The feature fusion weights $\boldsymbol{\alpha}$ are applied in the testing stage (see eq. (5.3)). The estimation of the feature fusion weights is performed after $\mathbf{P}$ converges to a binary matrix. According to the value of $\mathbf{P}$, i.e., value 1 and value 0 indicate the estimated match and nonmatch, respectively, the matched pairs and the nonmatched pairs are obtained. Two sets are then defined: the positive set includes the normalized distances of multiple features between estimated matched pairs: $\mathbf{D}_p = \{\mathbf{z}_i^p\}, i = 1 \sim N_p$, and the negative set includes those between nonmatched pairs: $\mathbf{D}_n = \{\mathbf{z}_i^n\}, i = 1 \sim N_n$, where $\mathbf{z}_i^p \in \mathbb{R}^4$ and $\mathbf{z}_i^n \in \mathbb{R}^4$ denote the vectors of the normalized distances of four features ($feature\_dist_i$): time, holistic color, region color, and region texture; N_p and N_n are the numbers of the matched pairs and the nonmatched pairs, respectively. The feature fusion weights are determined based on the degree of separation between the distributions of the values in the positive set and negative sets. Fig. 5.3 shows an example, where for each feature i, we denote the mean and standard deviation of the distribution of the feature distance as μ_i^p, μ_i^n, σ_i^p, and σ_i^n, respectively. The separation is measured as *d-prime* metric [12][72]:

$$d_i = \frac{\mu_i^n - \mu_i^p}{\sqrt{(\sigma_i^n)^2 + (\sigma_i^p)^2}}. \tag{5.41}$$

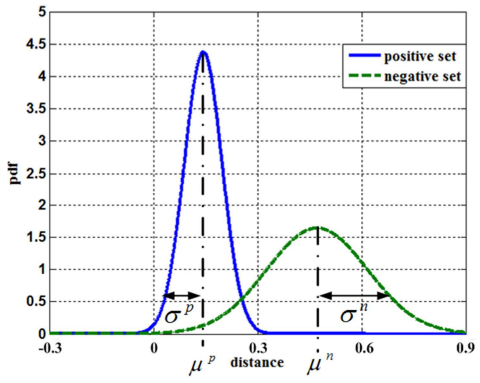


Figure 5.3: An example of the distributions of the distance values in the positive set and negative set. Blue solid curve is the pdf of positive set, and green broken line is the pdf of negative set.

The weight α_i can thus be calculated as:

$$\alpha_i = \frac{d_i}{\sum_{m=1}^{4} d_m}. \tag{5.42}$$

The larger the distance between the distributions of positive and negative sets, the better the differentiability of the feature is, resulting in a larger weight.

5.4 Unsupervised Learning and Update

In this section, we describe the complete unsupervised learning procedure in the training stage and the automatic camera link model update scheme in the testing stage.

5.4.a Unsupervised Learning Procedure

The deterministic annealing [16][69] is employed in our estimation process to obtain the optimum binary matrix **P** and the camera link model jointly. The optimization process is performed in a continuous space ($P_{ij} \geq 0$) which allows the fuzzy and partial matching between the observations instead of jumping in the space of binary permutation matrices (hard decision). The fuzziness is controlled by the deterministic annealing procedure which forces the randomness at the early stage and makes it gradually converge at the later stage based on the adjustment of β.

The unsupervised learning scheme is shown in Algorithm 5.1. In the beginning, the

---

1. Initialize $\boldsymbol{P}$, camera link model $\mathbb{M}$, $\beta = \beta_0$.
2. **While** ($\beta < \beta_f$)
3. **For** $l = 1$ *to* K
4. Formulate the objective function J based on the current estimated model.
5. Find the stationary point by solving $\frac{\partial J}{\partial P_{ij}} = 0$.
6. Project the solution to the space where constraints (5.24)~(5.26) are satisfied.
7. Perform alternately row-column normalization.
8. Sequentially estimate f_{tran}, f_{BTF}, $\boldsymbol{W}_{map}$, and $\boldsymbol{q}$ based on $\boldsymbol{P}$.
9. **End For**
10. Update $\beta = \beta \times \beta_r$. ($\beta_r > 1$)
11. **End While**
12. Estimate $\boldsymbol{\alpha}$.

Algorithm 5.1: Iteration of the unsupervised learning module in the training stage

matrix **P** and camera link model $\mathbb{M}$ are initialized, and β is set as β_0, a predefined small value. The outer iteration continues as long as β is smaller than β_f, a predefined large number. In each inner iteration, the objective function $J(\cdot)$ is formulated based on the most recently estimated camera link model, and a new **P** is updated by solving the problem (5.21)~(5.27). The newly updated **P** is then used to estimate a new camera link model. The multiple iterations ($K > 1$) in the inner loop enables us to obtain a better solution under the same β value before proceeding to the next β value. In each outer iteration, β is increased by multiplying with a constant scalar $\beta_r > 1$. After the outer iteration loop is finished, i.e., $\beta \geq \beta_f$, **P** becomes a binary matrix, and the feature fusion weights $\boldsymbol{\alpha}$ are further calculated as (5.42).

5.4.b Camera Link Model Update

Due to the presence of the outliers in the training data, the estimation of **P** may contain some wrong correspondences, so the camera link model needs to be further refined in the testing stage. Moreover, the environment change may lead to the necessity of the modification of the model, e.g., change of the lighting condition requires a different BTF. Therefore, in the testing stage, the camera link model needs to be continuously updated. For each link, whenever a matched pair is detected, if the matching distance is lower than a certain threshold, we believe they are likely to be a real matched pair, and the information of this pair can be utilized to update the camera link model. The threshold is set as 0.4 in our experiments.

To update the transition time distribution, we add the transition time of the detected matched pair into the set $\mathbf{T}_{valid}$, and the $f_{tran}(\cdot)$ can then be further estimated by (5.11). Denote $|\mathbf{T}_{valid}|_{max}$ as the maximum number of elements that set $\mathbf{T}_{valid}$ is allowed to hold. Whenever the number of elements in $\mathbf{T}_{valid}$ exceeds $|\mathbf{T}_{valid}|_{max}$, the element corresponding to the person who has the earliest entry time is discarded.

For the BTF, region mapping matrix and region matching weights, the update takes place by first getting an instant model $\Omega_{instant}$ based on the estimation procedure in Chapter 5.3.b~5.3.d, where **P** is an 1-by-1 matrix and is set to 1, followed by the update

equation (5.43)

$$\Omega_{new} = (1-\rho)\Omega_{old} + \rho\Omega_{instant}, \tag{5.43}$$

where $\Omega \in \{f_{BTF},\ \mathbf{W}_{map},\ \mathbf{q}\}$, and ρ controls the update rate.

For updating the feature fusion weights $\boldsymbol{\alpha}$, the feature distances $feature\_dist_i^{\boldsymbol{p}}$ are obtained from the newly matched pair. If the exit lists include other people besides the matched one, we can collect the distances as $feature\_dist_i^{\boldsymbol{n}}$ for the negative set. The parameters $\mu_i^{\boldsymbol{S}}$ and $\sigma_i^{\boldsymbol{S}}$ are updated as (5.44) given the available distance values,

$$\mu_{i\ new}^{\boldsymbol{S}} = (1-\rho)\mu_{i\ old}^{\boldsymbol{S}} + \rho feature\_dist_i^{\boldsymbol{S}}, \tag{5.44}$$

$$\left(\sigma_{i\ new}^{\boldsymbol{S}}\right)^2 = (1-\rho)\left(\sigma_{i\ old}^{\boldsymbol{S}}\right)^2 + \rho(feature\_dist_i^{\boldsymbol{S}} - \mu_{i\ new}^{\boldsymbol{S}})^2, \tag{5.45}$$

where $\boldsymbol{S} \in \{p, n\}$ and $i = 1\sim4$. Once the mean and variance are updated, $\boldsymbol{\alpha}$ is then adjusted by (5.41) and (5.42).

5.5 Experimental Results

This section is divided into three parts: First, the correctness of our camera link model estimation and the comparative study with other methods are presented. Second, the camera link model is applied in a multiple-camera tracking system. The experiments, conducted based on real-world video scenarios, show the effectiveness of our proposed system. Finally, we provide the discussion for the sensitivity of the parameters used in our unsupervised learning scheme.

5.5.a Camera Link Model Estimation

We set up four cameras C_1~C_4 around the Electrical Engineering building. The FOVs of the cameras are spatially disjointed, and the cameras do not need any calibration in advance. The topology of the cameras, as shown in Fig. 5.4, is the only prior knowledge which can be easily obtained in real application. There are four links (four pairs of connected entry/exit zones), shown as blue lines, are established in between the four cameras. The corresponding entry/exit zones are marked by red ellipses.

Given the training sets from the four cameras, the camera link models are estimated.

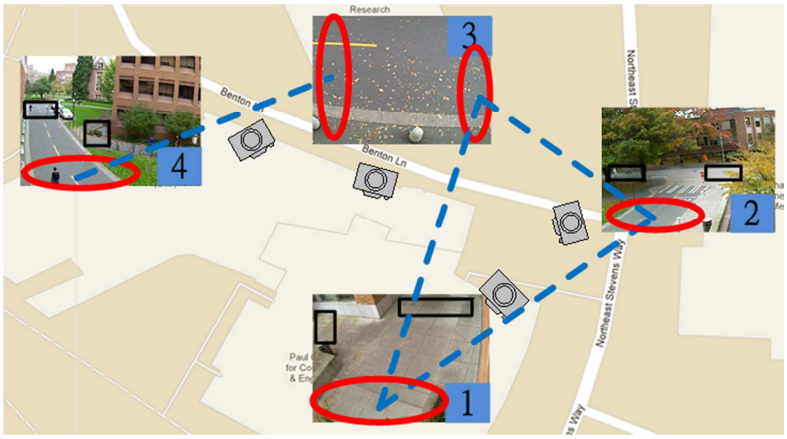


Figure 5.4: Camera topology. Four links are denoted as blue broken lines, and the corresponding entry/exit zones are denoted as red ellipses. Black rectangles are the other entry/exit zones which do not have any link between them.

The parameters in Algorithm 5.1 are set as following: $\beta_0 = 0.001$, $\beta_r = 1.2$, $\beta_f = 150$, $K = 4$. σ_{tran} in (5.11) is set as 0.5. θ in the outlier cost function (5.20) is set within 1~2. Euclidean distance is employed for the distance function $D(\cdot)$ in our work. The initial values of **P** and the camera link model are assigned as follows:

$$P_{ij} = \begin{cases} 0, & y_j^t - x_i^t \le 0, \ \ \forall\, i = 1{\sim}N_1, j = 1{\sim}N_2 \\ \varepsilon, & otherwise \text{ (include outlier row and column)} \end{cases}. \tag{5.46}$$

where ε is set as 0.01 in our simulations, and **P** is normalized before the process starts; f_{tran} is initialized based on (5.8)~(5.11) by using the normalized **P**; f_{BTF} is initially set as an identity function, i.e., assume no color deviation initially; an identity matrix is used as the initial $\mathbf{W}_{map}$; **q** is set as uniform.

Fig. 5.5 (a) shows the estimation result of the transition time distribution between camera C_2 and camera C_3. There are two 12-minute videos collected from two cameras as training data, including 104 exit observations from camera C_2 and 40 entry observations from camera C_3, where 33 pairs of people are matched pairs, i.e, there are total 78 outliers (54%). The results of other approaches [36][42][57][74] and the ground truth are also shown in the figure. The ground truth is obtained by manually labeling the correspondences and estimate the transition time based on (5.11), namely supervised learning as in [45] and [66]. Table 5.1 shows the quantitative error report of the above

simulations. The error is calculated as the distance between the estimated distribution and the ground truth. Our estimation has the smallest error to the ground truth.

The estimation result of the transition time distribution between camera C_1 and camera C_2 is shown in Fig. 5.5 (b). In the same period of 12-minute videos collected as training data, there are 30 exit observations from camera C_1 and 52 entry observations in camera C_2. There are 18 pairs of matched pairs, i.e., 46 of the total observations are outliers (56%). According to the error in Table 5.1, our method again outperforms the others.

In addition to our self-recorded data, we use the video clips in a multiple-camera tracking scenario from an international benchmark, i-LIDS dataset [75], where the cameras are set up in a busy airport area. Fig. 5.6 is the estimation result of the transition time distribution between two cameras in the dataset. There are 66 exit observations and 59 entry observations collected from two 6-minute videos of two cameras, where only 41 pairs of matched pairs are present, resulting in 43 outliers (34.4%). We can see that multiple modes present in the distribution since people tend to travel with varying speeds due to different amount of luggage they carry. The estimation provided by our proposed method attains the lowest error.

In methods [36] and [57], the correspondences are not explicitly solved, so large amount of noise is involved in the estimation results showing the rough trend of the

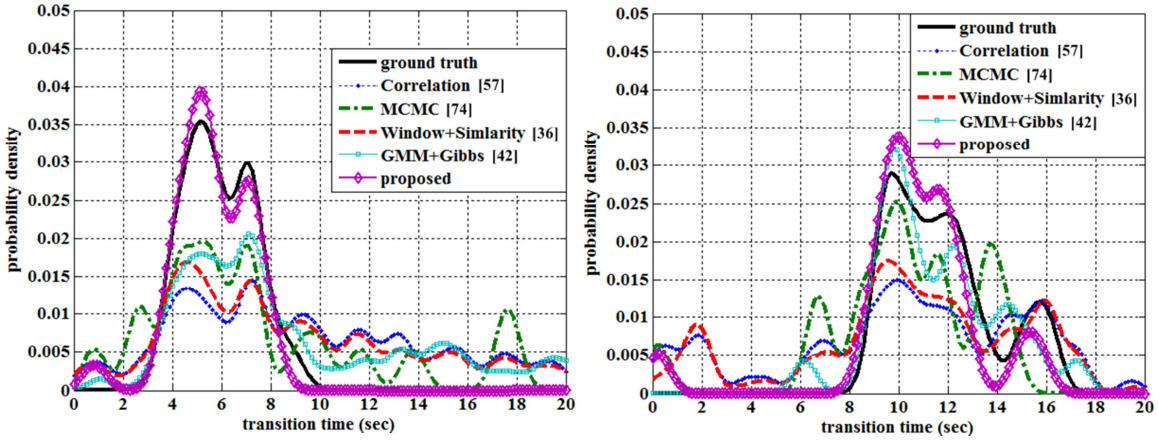


Figure 5.5: Comparison of the distributions of the transition time. (a) between camera C_2 and camera C_3. (b) between camera C_1 and camera C_2.

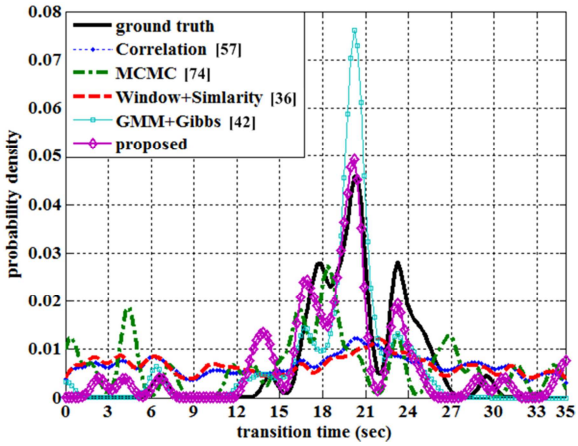


Figure 5.6: Comparison of the distributions of the transition time for iLIDS dataset.

transition time distribution without the clear modes. Although the methods in [42] and [74] try to the estimate the correspondences as latent variables, their sampling-based schemes always make hard decisions and are thus easily trapped in poor local optima. Moreover, since the outliers are not explicitly taken care of, the estimation results are easily deteriorated by the wrong correspondences. Since random sampling schemes are involved in these two methods, we report the best results from them after several trials in the above comparison.

Fig. 5.7 gives an example of the estimation results of the brightness transfer functions. Four curves correspond to four different links, and only the BTFs of the blue color channel of each are shown here for demonstration purpose. We can see that the color deviation does exist between different cameras. We do not compare the results with the ground truth, since it is not easy for human to determine the ground truth of region mapping matrix $\mathbf{W}_{map}$ and region matching weights $\mathbf{q}$, which are necessary for obtaining

Table 5.1: Error of the transition time distribution

Error Comparison	Camera C_2 & C_3	Camera C_1 & C_2	i-LIDS Dataset
Correlation [57]	0.1074	0.0787	0.1170
MCMC [74]	0.0824	0.0773	0.1208
Window+Similarity [36]	0.0952	0.0688	0.1229
GMM+Gibbs [42]	0.0738	0.0368	0.0861
Proposed	**0.0158**	**0.0339**	**0.0586**

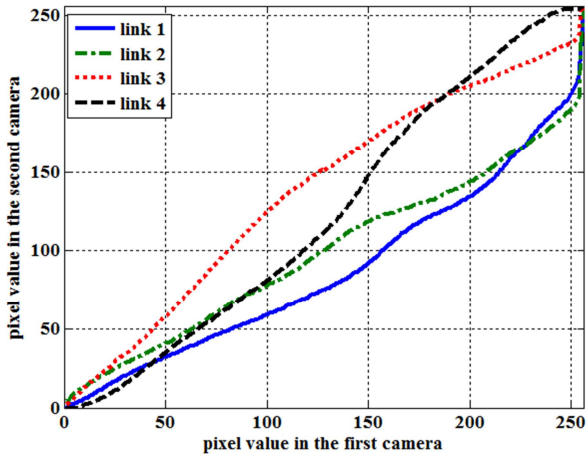


Figure 5.7: Brightness transfer function. Note only one channel is shown here for demonstration purpose.

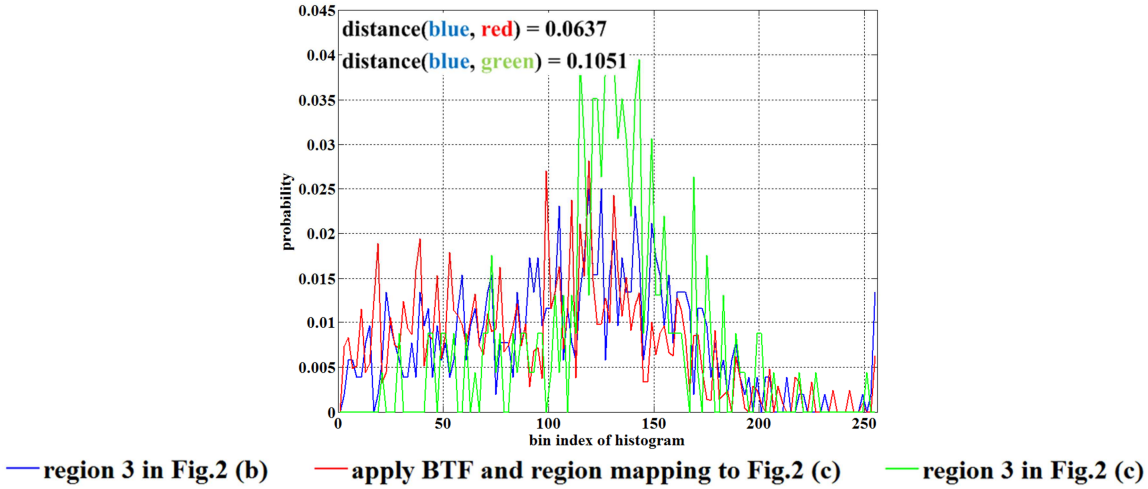


Figure 5.8: Histogram comparison. Blue: the histogram from region 3 in Fig. 5.2 (b). Red: the histogram after applying BTF and region mapping matrix to Fig. 5.2 (c). Green: the histogram from region 3 in Fig. 5.2 (c). Note only one channel is shown here for demonstration purpose.

the ground truth BTF (see Chapter 5.3.b.). Hence, the ground truth of BTF is not shown here. However, we can justify the effectiveness of BTF and region mapping matrix $\mathbf{W}_{map}$ by examining the region histogram matching. For example, the histogram (blue curve in Fig. 5.8) of region 3 in Fig. 5.2(b) is compared with the ones with and without applying camera link model. The red curve in Fig. 5.8 is the one after applying BTF and region mapping matrix to the histograms in Fig. 5.2(c). The green curve is the histogram of

region 3 in Fig. 5.2(c). By applying the correct camera link model, BTF and region mapping matrix $\mathbf{W}_{map}$, the one with the camera link model applied gets the better matching which gives preferable similarity measurement between the same objects under two cameras.

The histogram extracted from region 3 in Fig. 5.2(b) can be modeled as the linear combination of the histograms in Fig. 5.2(c) with the mapping weights $\mathbf{w}_3$ shown in Table 5.2. The yellow rectangles cover the same areas in Figs. 5.2(b) and (c), which accounts for nearly 85% of weight falling in region 1 to 4. Moreover, since the left part of the target in camera C_3 is not well-observable in camera C_4 due to the different perspectives, and the right part of the target in camera C_3 is well-observable in camera C_4, it explains the region matching weight values $\boldsymbol{q}$ (Table 5.3) are higher for the regions 1, 3, and 5 (q_1, q_3, q_5; right part of the torso) than the regions 2, 4, and 6 (q_2, q_4, q_6; left part of the torso).

5.5.b Tracking Test Over Multiple Cameras

We further implement an automatic multiple-camera tracking system that tracks humans across cameras based on the learned camera link models. The method in [20] is utilized to accomplish the tracking within a single camera. One can see in Fig. 5.4, the uncertainty of the exit and entry events increases the difficulty of the tracking. For example, a person exiting from camera C_1 can enter into camera C_2 or camera C_3. In our 20-minute testing video, there are 336 people appearing in the deployed camera network. When we use only temporal and holistic color feature similar to the methods in [13] and [46], the re-

Table 5.2: Vector $\boldsymbol{w}_3$ of the region mapping matrix

$\boldsymbol{w}_3$	w_{31}	w_{32}	w_{33}	w_{34}	w_{35}	w_{36}
Value	0.295	0.2	0.185	0.15	0.14	0.03

Table 5.3: Region matching weights $\boldsymbol{q}$

$\boldsymbol{q}$	q_1	q_2	q_3	q_4	q_5	q_6	q_7
Value	0.17	0.142	0.16	0.099	0.224	0.095	0.11

Table 5.4: Re-identification accuracy

Method	temporal and holistic color features only	uniform fusion weight	proposed system
Accuracy	68.9%	72.8%	79.5%

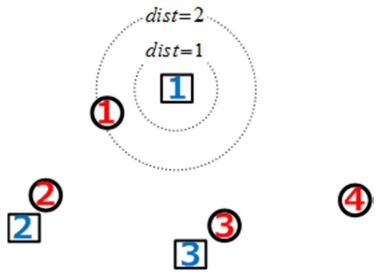
identification accuracy, shown in Table 5.4, is 68.9%, which shows that the temporal and holistic color features are important cues. The re-identification accuracy is defined as the fraction of the people being correctly labeled. If we consider the region features with the equally-weighted multi-cue integration, i.e., feature fusion weights are uniform, the accuracy increases to 72.8%, which demonstrates the effectiveness of the proposed region features. By further incorporating the adaptive feature fusion weights, our system achieves 79.5% accuracy. Compared to the existing multiple-camera tracking system [13][46], our proposed system enhances the performance by considering region matching and feature fusion weights.

In [28][33][39][84], Cumulative Match Characteristic (CMC) curve was used for evaluating the performance of the re-identification. The matching is performed across two sets of humans. For each person in the first set, the distances to all the people in the second set are calculated and ranked. In our work, each person entering the view will only be matched against the ones leaving the other views before, so the number of people who will be matched are highly dependent on the scenario hence are usually not the same. Therefore, the CMC curve is not suitable for evaluation purpose here.

5.6 Discussion

5.6.a Visualization of parameter θ

θ is a control factor for indicating how large the distance we can tolerate when we determine whether a particular observation would likely to be an outlier or not. In the estimation process, the entries of matrix **P** is calculated as (5.28). One can see that the entries of the outlier row and column are always e^{-1}, while the others are determined by



(a)

$$\text{dist}_{ij} = \begin{array}{c} \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \mathbf{y}_4 \end{array} \\ \begin{bmatrix} 1.6 & 3.9 & 3.5 & 5 \\ 3.8 & 0.4 & 3.4 & 6 \\ 3.97 & 3.91 & 0.4 & 4.6 \end{bmatrix} \end{array}$$

(b)

Figure 5.9: Example of feature points expressed in 2D space. (a) Three exit observations and four entry observations, represented by squares and circles, respectively. (b) The distances between each pair.

the total distance ($dist_{ij}$) of all the features. For instance, in Fig. 5.9(a), there are three exit observations and four entry observations, represented by squares and circles, respectively. Note that we assume the camera link model has been applied in order to map them into the same space, which is expressed as 2D space here only for visualization purpose. The dotted circles stand for the distances to the first exit observation. Fig. 5.9(b) is the distances between each pair of points. Fig. 5.10 is the plot for the P_{ij} value as the function of the distance when $\beta = 1$. We can see if β is fixed, when θ is getting larger, it tolerates larger distance before the P_{ij} becomes smaller than e^{-1}. It means for a particular observation, it has larger radius to cover more observations as the potential matches, so

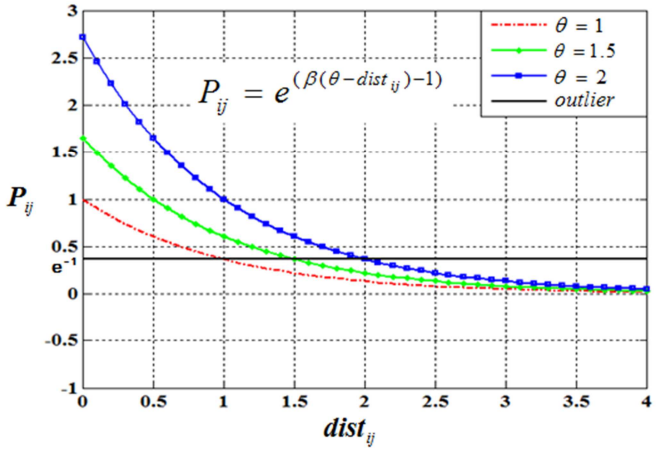


Figure 5.10: P_{ij} value as the function of the distance when $\beta = 1$ under different θ values.

the probability of an observation being recognized as an outlier is reduced. For example, the distance between $\mathbf{x}_1$ and $\mathbf{y}_1$ is 1.6, and P_{ij} would be greater than e^{-1} if $\theta = 2$ and smaller than e^{-1} if $\theta = 1$.

The row-column normalization is performed after we evaluate the matrix **P** based on (5.28) and (5.29). Given $\theta = 1.5$ and $\beta = 1$, the matrix **P** before and after the row-column normalization are shown as (5.47) and (5.48), respectively.

$$\mathbf{P} = \begin{bmatrix} 0.3329 & 0.0334 & 0.0498 & 0.0111 & 0.3679 \\ 0.0369 & 1.1052 & 0.055 & 0.0041 & 0.3679 \\ 0.0311 & 0.033 & 1.1052 & 0.0166 & 0.3679 \\ 0.3679 & 0.3679 & 0.3679 & 0.3679 & \mathrm{X} \end{bmatrix} \tag{5.47}$$

$$\mathbf{P} = \begin{bmatrix} 0.4751 & 0.0304 & 0.0437 & 0.0319 & 0.4188 \\ 0.0342 & 0.6548 & 0.0313 & 0.0076 & 0.272 \\ 0.0294 & 0.02 & 0.6417 & 0.0316 & 0.2774 \\ 0.4612 & 0.2948 & 0.2833 & 0.9289 & \mathrm{X} \end{bmatrix} \tag{5.48}$$

Note that when the distance between an observation pair is greater than θ, it does not imply they will never be considered as a match, but they have relatively lower probability to be a match; that is, θ is never a hard threshold. For instance, the distance between $\mathbf{x}_1$ and $\mathbf{y}_1$ is 1.6, which is greater than θ, but the corresponding $P_{11} = 0.4751$ is even larger

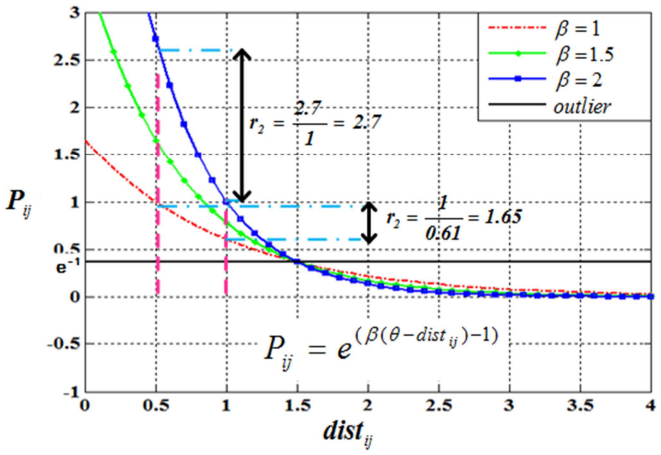


Figure 5.11: P_{ij} value as the function of the distance when $\theta = 1.5$ under different β values. If the distances between two pairs of observations are 0.5 and 1, the inter-variance of corresponding P_{ij} when $\beta = 2$ is much larger than the inter-variance of corresponding P_{ij} when $\beta = 1$, i.e., $r_2 > r_1$.

than the outlier entries, i.e., $P_{15} = 0.4188$ and $P_{41} = 0.4612$ in (5.48). In this case, this pair of observations will still be involved in the camera link model estimations due to the nonzero value of P_{11}. This kind of soft decision allows the system to search the optimum more effectively, especially during the early stage of the estimation process. During the early stage, the correct observation pairs may not be close enough in the feature space since the camera link model has not been well-estimated yet. By setting θ properly, the system will consider those correct matches in updating the model which leads to the right direction during the optimum search.

If θ is too small, unless the transformation between the correct matches can be neglected, i.e., they are already close in the beginning, the estimation tends to recognize most observations as outliers. If θ is too large, the system will include too many wrong pairs (the noise) for the estimation and lead to some sub-optimal directions during the optimum search.

5.6.b Visualization of parameter β

While θ controls the determination of outliers, β is a factor for the fuzziness. Fig. 5.11 shows the similar curves as in Fig. 5.10 but with β being the variable instead. θ is set as 1.5, so all the curves intersect at distance equal to 1.5. Since the row-column normalization takes place after solving **P** via (5.28) and (5.29), the fuzziness of the resulting matrix **P** is related to the ratio between the original values (before row-column normalization is applied) of the entries within the same row or column. For example, given two pairs of observations whose distances are smaller than θ, say 0.5 and 1, respectively, before the normalization is performed, the difference between their corresponding values (P_{ij}) will be larger when β is 2 than when β is 1 ($r_2 > r_1$); that is, with the same difference of the distances, the inter-variance is enlarged by the exponential function, especially with large β. On the other hand, when the distance is larger than θ, the outlier entry, equal to e^{-1}, will dominate when β is large. In summary, in the early stage of the estimation, β is small which creates a relatively loose constraint in searching the optimum. In the later stage, as β is larger, either an entry with small

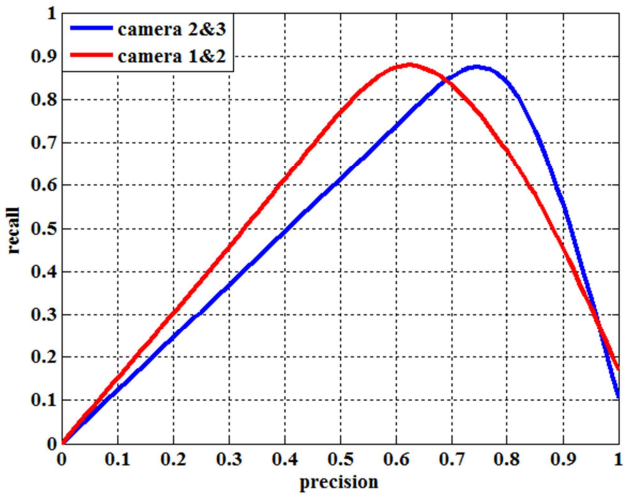


Figure 5.12: Precision-Recall for two links. Blue: based on the training data from camera C_2 and camera C_3. Red: based on the training data from camera C_1 and camera C_2.

distance or an outlier entry will start to dominate the others within the same row or column while computing the row-column normalization which results in convergence to a binary matrix.

5.6.c Accuracy of the Estimation of the Corresponding Matrix

After Algorithm 5.1 finishes, the matrix **P** converges to a binary matrix. Thus, the final estimation of the correspondences and the camera link model are obtained. If the correspondences are estimated correctly, the camera link model is more reliable. We define the precision and recall rate of the estimation of the correspondences as the following:

$$precision = \frac{\#\ of\ correctly\ identified\ pairs}{\#\ of\ pairs\ identified\ as\ correspondences}, \tag{5.49}$$

$$recall = \frac{\#\ of\ correctly\ identified\ pairs}{\#\ of\ nonoutlier\ pairs\ in\ the\ training\ set}. \tag{5.50}$$

Assume observations A and B are identified as a matched pair, i.e., $P_{AB} = 1$, then wrong correspondence happens in two cases: (i) A and B are both nonoutliers but they are not correct matches to each other. (ii) If at least one out of A and B is an outlier. If the precision is high, the noise come from the wrong correspondences would be low. The estimated model is more correct. If the recall rate is high, the estimated model obtained

from more correct correspondences is more generalized so as to avoid the issue of overfitting. Hence, these two measurements can also represent the quality of the estimation results. We do not perform the comparisons with other methods as is done in Chapter 5.5.a in terms of these two performance metrics since those competing methods either do not estimate correspondences explicitly [36][57] or they do not enforce one-to-one correspondence [42][74]. Therefore, it would not give useful information to compute the precision and recall based on their methods.

As discussed in Chapter 5.6.a, θ can be used as a soft threshold to determine the outliers, so we analyze how it affects the estimation accuracy in terms of precision and recall. By adjusting θ values, varying in the range of 0.5 to 2.5, we run the estimation process for each θ value, and two precision-recall curves of the estimations based on the training data between cameras C_2 & C_3 and cameras C_1 & C_2 are shown in Fig. 5.12. Unlike the conventional binary classification in which the recall and precision are usually trade-off, we observe that the recall and precision both drop when θ is too small or θ is too large. When θ is small, it poses more strict criterion for matching the observations, so the number of correctly identified pairs is decreased, resulting in an ill-estimated model which produces the false positive (wrong correspondences). Hence, the recall and precision decreases. When θ is too large, the system relaxes the soft threshold and introduces much more noise during the estimation process. The noise results in wrong correspondences between nonoutliers causing the drop of recall rate. Moreover, more outliers are identified as corresponding pairs, and these false positives reduce the precision. Therefore the precision and recall rates are decreased. By proper selection of θ value, we can attain 80% of precision and 70%~85% of recall rates even though there are more than 50% of the training data are outliers.

5.6.d Distance Metric Learning Based Human Re-identification

Recently, there are several works focusing on learning a distance metric between each pair of cameras (entry/exit zones) for person re-identification purpose [28][41][84][85]. The idea is to obtain a linear transformation **K** which maps the original features to a new feature space so that the plain Euclidean distance computation can be applied in this new

feature space to effectively differentiate different people; that is, if we denote two original feature vectors $\mathbf{v}_i \in \mathbb{R}^n$ and $\mathbf{v}_j \in \mathbb{R}^n$, the goal is to learn a matrix $\mathbf{K} \in \mathbb{R}^{n \times m}$, where $n \geq m$, such that the Euclidean distance after projection $\left\|\mathbf{K}^T\mathbf{v}_i - \mathbf{K}^T\mathbf{v}_j\right\|_2$ is small if $\mathbf{v}_i$ and $\mathbf{v}_j$ are the same person; otherwise, the value should be large. In all of these works in human re-identification, supervised labeling of the training data is required. When the data set is large or the number of the camera pairs increases, the labeling needs huge amount of manual effort, which makes this approach infeasible.

In our work, we do not compare our work with those using metric learning approaches because of several reasons. First, those approaches need supervised learning which is different from our unsupervised learning scheme. Moreover, they did not propose complete systems. All those works focused mainly on matching without considering the tracking part. It may not be fair if we do the comparison. Since our modeling is more related to the works [36][46][57][74], we only compare with these methods.

On the other hand, one can consider incorporating the distance metric learning into our estimation scheme. If the color information is used as the features, as is done in [28][41][84][85], the linear mapping can be analogous to the brightness transformation between two cameras. If we define the cost function as

$$\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} P_{ij}\left\|\mathbf{K}^T\mathbf{v}_i - \mathbf{K}^T\mathbf{v}_j\right\|_2^2, \tag{5.51}$$

by including (5.51) in our objective function (5.27), we can solve the minimization problem with respect to the new objective function. Moreover, given the current estimated $\mathbf{P}$, one can include it in whichever optimization formulation is used in [28][41][84][85] to estimate $\mathbf{K}$. Therefore, distance metric learning can be performed through our camera link model estimation procedure without any manual labeling. This part of research will be included in our future work.

Chapter 6 – Conclusions and Future Work

6.1 Conclusions

In this dissertation, we proposed a robust and consistent human tracking system which tracks people within single camera and across multiple cameras. Three major topics have been investigated: single camera tracking, tracking across cameras with overlapping views, and tracking across cameras with nonoverlapping views.

We propose an innovative method that uses projected gradient to facilitate multiple-kernel tracking in finding the best match under predefined constraints. Since some of the kernels are not observable, the adaptive weights are employed to the kernels to lower the importance of the ones being occluded while enhance it of the well-observable ones. Moreover, a simple yet effective approach is presented to deal with the scale change issue. Finally, the multiple-kernel tracking is combined with a Kalman filter to form a complete automatic tracking system. Based on the experimental results, the multiple-kernel tracking can successfully track the specific target under severe occlusion, and the overall system also demonstrates the promising results for tracking every individual simultaneously.

We have built a system for tracking humans across multiple cameras with overlapping views. We utilize image registration technique to identify the FOV lines automatically given the views of two cameras. In order to compensate for the color deviation and perspective difference, the brightness transfer function and tangent transfer function are utilized. The camera link model, including homography matrix, brightness transfer function and tangent transfer function, is applied when we calculate the distance function between two people regarding to vicinity cue, color cue, and edge cue. The update stage yields the camera link model more reliable over time. The promising results from several test videos demonstrate the effectiveness of our proposed method.

Finally, we have built a tracking system that tracks humans across multiple cameras with disjointed FOVs based on the application of camera link models. The camera link

model, including transition time distribution, brightness transfer function, region mapping matrix, region matching weights, and feature fusion weights, is correctly estimated by an unsupervised scheme even under the presence of outliers in the training data. Our estimation procedure can be generalized easily by adding more features or components in the camera link model. Several experiments and comparative studies demonstrate the effectiveness of our proposed method. The proposed tracking system is applied in a self-deployed camera network in the real world. We show the use of region matching and systematically-determined feature fusion weights can further enhance the tracking accuracy. The pairwise learning and tracking manner enhances the scalability of the system.

6.2 Future Work

The first future work is to incorporate the distance metric learning in our camera link model estimation procedure. Recently, distance metric learning approaches in human re-identification [28][41][84][85] have gained some popularity and demonstrated promising accuracy. However, the supervised learning fashion prevents it from real application. It is possible to incorporate this method with our estimation procedure by treating the linear transformation in distance metric as one of the component in the camera link model as we discussed in Chapter 5.6.d. In this way, we believe the system can improve the matching accuracy while still saving large amount of manpower.

In addition to the above, we want to keep enhancing the scalability of our system. In our proposed system, we assume the topology of the cameras, i.e., the geographical relationship between the cameras, is obtained as the prior knowledge. This information is straightforward when the scale of the camera network is moderate. However, as the number of the cameras increases, getting the topology is not a trivial work. Given a camera network consisting of multiple cameras, two pieces of topology information are required before the camera link model estimation can be performed: (i) The system needs to identify which pairs of cameras have link models between them, i.e., which pairs are directly connected. Wrong links or redundant links deteriorate the tracking performance

easily, due to the increased searching range resulting in reduced recall rate and increased false positives, not to mention the exponentially increased computational complexity. (ii) According to our observations, the link actually only connects two entry/exit zones in a pair of directly-connected cameras; that is, if a person is traveling between two cameras, he/she will likely leave from one particular zone and enters into the other. Hence, in order to avoid too many outliers, the training data used in camera link model estimation (and the subsequent re-identification tracking) should only include the observations happening in these two specific zones instead of including all the observations in the camera. Therefore, to identify which specific zones are linked together is another critical issue.

Thanks to the development of the online map services nowadays, e.g, Bing Maps and Google Maps, etc, we present an approach to solve the above problems by retrieving the information from the online map [23]. By providing the GPS locations of uncalibrated cameras and incorporating with Google Maps and Google Street View, the system can automatically identify the camera links within the camera network. In the future, we will further investigate the stability of this approach, and the visual data should also be considered to validate the information from online maps. We look forward to integrating this work into our system, so that the system will be self-organized and will scale up efficiently.

Bibliography

[1] A. Agarwal, C. V. Jawahar, and P. J. Narayanan, “A survey of planar homography estimation techniques,” IIIT Technique Report, 2005.

[2] A. Ali and K. Terada, “A framework for human tracking using kalman filter and fast mean shift algorithms,” *IEEE Intl. Conf. Computer Vision Workshops*, pp. 1028-1033, 2009.

[3] D. Arsic, E. Hristov, N. Lehment, B. Hornler, B. Schuller, and G. Rigoll, “Applying multi layer homography for multi camera person tracking,” *ACM/IEEE Intl. Conf. on Distributed Smart Cameras*, 2008.

[4] R. V. Babu, P. Perez, and P. Bouthemy, “Robust tracking with motion estimation and kernel-based color modeling,” *IEEE Intl. Conf. Image Processing*, vol. 1, pp. 717-720, 2005.

[5] R. V. Babu, P. Perez, and P. Bouthemy, “Robust tracking with motion estimation and local kernel-based color modeling,” *Image and Vision Computing*, vol. 25, issue 8, pp. 1205-1216, 2007.

[6] S. Bak, E. Corvee, F. Bremond, and M. Thonnat, “Person re-identification using Haar-based and DCD-based signature,” *IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance*, 2010.

[7] M. Bauml and R. Stiefelhagen, “Evaluation of local features for person re-identification in image sequences,” *IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance*, Sep, 2011.

[8] C. Beleznai, B. Fruhstuck, B. Horst, “Human tracking by fast mean shift mode seeking,” *Journal of Multimedia*, vol. 1, pp. 1-8, 2006.

[9] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[10] S. Boyd and L. Vandenberghe, Ch8.1 in Convex Optimization, *Cambridge University Press*, 2004.

[11] CAVIAR: Context Aware Vision using Image-based Active Recognition, EC founded CAVIAR project/IST 2001 37540, Available: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[12] K. Chen and Y. Hung, "Multi-cue integration for multi-camera tracking," *IEEE Intl. Conf. on Pattern Recognition*, pp. 145-148, 2010.

[13] K. Chen, C. Lai, Y. Hung, and C. Chen, "An adaptive learning method for target tracking across multiple cameras," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.

[14] H. Cheng and J. Hwang, "Multiple target tracking for crossroad tracking utilizing modified probabilistic data association," *Signal Processing*, vol. 89, issue 9, pp. 1844-1849, 2009.

[15] A. Chilgunde, P. Kumar, S. Ranganath, and H. Weimin, "Multi-camera target tracking in blind regions of cameras with non-overlapping fields of view," *British Machine Vision Conference*, 2004.

[16] H. Chiu and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, pp. 114-141, 2003.

[17] C. Chu, J. Hwang, Y. Chen, and S. Wang, "Camera link model estimation in a distributed camera network based on the deterministic annealing and the barrier method," *Proc. IEEE Conf. on ASSP*, pp. 997-1000, March, 2012.

[18] C. Chu, J. Hwang, K. Lan, and S. Wang, "Tracking across multiple cameras with overlapping views based on brightness and tangent transfer functions," *ACM/IEEE Intl. Conf. on Distributed Smart Cameras*, 2011.

[19] C. Chu, J. Hwang, H. Pai, and K. Lan, "Robust video object tracking based on multiple kernels with projected gradients," *IEEE Conf. on Acoustics, Speech and Signal Processing*, pp. 1421-1424, May, 2011.

[20] C. Chu, J. Hwang, H. Pai, and K. Lan, “Tracking human under occlusion based on adaptive multiple kernels with projected gradient,” *IEEE Trans. on Multimedia*, 2013.

[21] C. Chu, J. Hwang, S. Wang and Y. Chen, “Human tracking by adaptive Kalman filtering and multiple kernels tracking with projected gradients,” *ACM/IEEE Intl. Conf. on Distributed Smart Cameras*, 2011.

[22] C. Chu, J. Hwang, J. Yu, and K. Lee, “Tracking across nonoverlapping cameras based on the unsupervised learning of camera link models,” *ACM/IEEE Intl. Conf. on Distributed Smart Cameras*, 2012.

[23] C. Chu, K. Lee, and J. Hwang, “Self-organized and scalable camera networks for systematic human tracking across nonoverlapping cameras”, *IEEE Conf. on Acoustics, Speech and Signal Processing*, Vancouver, 2013.

[24] D. M. Chu and A. W.M. Smeulders, “Thirteen hard cases in visual tracking,” *IEEE AVSS*, 2010.

[25] D. Comaniciu, and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002.

[26] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564-577, May 2003.

[27] R. T. Collins, “Mean-shift blob tracking through scale space,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 234-240, 2003.

[28] M. Dikmen, E. Akbas, T. S. Huang and N. Ahuja, “Pedestrian recognition with a learned metric,” *Proc. ACCV*, 2010.

[29] D. Doermann and D. Mihalcik, “Tools and techniques for video performance evaluation,” *IEEE Intl. Conf. Pattern Recognition*, pp. 167-170, 2000. Available: http://viper-toolkit.sourceforge.net/

[30] T. D'Orazio, P. Mazzeo, and P. Spagnolo, "Color brightness transfer function evaluation for non overlapping multi camera tracking," *ACM/IEEE Intl. Conf. on Distributed Smart Cameras,* pp. 1-6, 2009.

[31] Z. Fan, Y. Wu, and M. Yang, "Multiple collaborative kernel tracking," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 502-509, 2005.

[32] J. Fang, J. Yang, and H. Liu, "Efficient and robust fragments-based multiple kernels tracking," *Intl. J. Electron. Commun.*, vol. 65, pp. 915-923, 2011.

[33] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani, "Person re-identification by symmetry-driven accumulation of local features," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 2360-2367, 2010.

[34] M. Fashing, and C. Tomasi, "Mean shift is a bound optimization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 471-474, 2005.

[35] N. Gheissari, T. B. Sebastian, P. H. Tu, J. Rittscher, and R. Hartley, "Person reidentification using spatiotemporal appearance," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1528-1535, 2006.

[36] A. Gilbert and R. Bowden, "Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity," *Proc. ECCV*, pp. 125-136, 2006.

[37] R. C. Gonzalez, and R. E. Woods, *Digital Image Processing*, Springer, ch.6, 2006.

[38] H. Grabner and H. Bischof, "On-line boosting and vision," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 260-267, 2006.

[39] D. Gray and H. Tao, "Viewpoint invariant pedestrian recognition with an ensamble of localize features," *Proc. ECCV*, pp. 262-275, 2008.

[40] G.D. Hager, M. Dewan, and C. V. Stewart, "Multiple kernel tracking with SSD," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 790-797, 2004.

[41] M. Hirzer, P. Roth, M. Kostinger and H. Bischof, “Relaxed pairwise learned metric for person re-identificaiton,” *ECCV*, 2012.

[42] C-C. Huang, W-C. Chiu, S-J. Wang, and J-H. Chang, “Probabilistic modeling of dynamic traffic flow across non-overlapping camera views,” *IEEE Intl. Conf. on Pattern Recognition*, pp. 3332-3335, 2010.

[43] i-Lids dataset for AVSS 2007. Available: *http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html*

[44] O. Javed, Z. Rasheed, O. Alatas, and M. Shah, “KNIGHT: A real time surveillance system for multiple overlapping and non-overlapping cameras,” *IEEE Intl. Conf. Multimedia and Expo*, vol. 1, 2003.

[45] O. Javed, Z. Rasheed, O. Alatas, and M. Shah, “Tracking across multiple cameras with disjoint views,” *Proc. IEEE Conf. on Computer Vision*, France, pp. 952-957, 2003.

[46] O. Javed, K. Shafique, Z. Rasheed, and M. Shah, “Modeling inter-camera space-time and appearance relationships for tracking across non-overlapping views,” *Computer Vision and Image Understanding*, pp. 146-162, 2008.

[47] J. Kang, I. Cohen, and G. Medioni, “Continuous tracking within and across camera streams,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, 2003.

[48] V. Karavasilis, C. Nikou, and A. Likas, "Visual tracking by adaptive kalman filtering and mean shift", in *Proc. SETN*, pp.153-162, 2010.

[49] A. Kembhavi, B. Siddiquie, R. Miezianko, S. McCloskey, and L. Davis, “Incremental multiple kernel learning for object recognition,” *IEEE Intl. Conf. on Computer Vision*, p. 638-645, 2009.

[50] S. Khan and M. Shah, “Consistent labeling of tracked objects in multiple cameras with overlapping fields of view,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1355-1360, Oct 2003.

[51] S. M. Khan and M. Shah, “Tracking multiple occluding people by localizaing on multiple scene planes,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, 2009.

[52] C. C. Kuo, C. Huang, and R. Nevatia, “Inter-camera association of multi-target tracks by on-line learned appearance affinity models,” *ECCV*, 2010.

[53] I. Leichter, M. Lindenbaum, and E. Rivlin, “Tracking by affine kernel transformations using color and boundary cues,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 1, Jan, 2009.

[54] H. Ling, and K. Okada, “An efficient earth mover’s distance algorithm for robust histogram comparison,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, pp. 840-853, 2007.

[55] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Intl. Journal of Computer Vision*, pp. 91-110, 2004.

[56] J. Maciel and J. Costeira, “A global solution to sparse correspondence problems,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 187-199, Feb 2003.

[57] D. Makris, T. Ellis, and J. Black, “Bridging the gaps between cameras,” *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 205-210, 2004.

[58] B. Martinez, L. Ferraz, X. Binefa, and J. Diaz-Caro, “Multiple kernel two-step tracking”, *IEEE Intl. Conf. Image Processing*, pp. 2785-2788, 2006.

[59] B. Moller, T. Plotz, and G. A. Flink, “Calibration-free camera hand-over for fast and reliable person tracking in multi-camera setups,” *Intl. Conf. on Pattern Recognition*, pp. 1-4, 2008.

[60] K. Nummiaro, E. Koller-Meier, and L. V. Gool, “An adaptive color-based particle filter,” *Image and Vision Computing,* pp. 99-110, 2003.

[61] T. Ojala, M. Pietikainen and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classificaiton with local binary patterns,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, Jul 2002.

[62] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Transaction Sys., Man., Cyber. On Robotics*, pp. 62-66, 1979.

[63] V. Parameswaran, V. Ramesh, and I. Zoghlami, “Tunable kernels for tracking,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 2179-2186, 2006.

[64] PETS 2010: Thirteenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 2010. Available: http://pets2010.net/

[65] F. Porikli and O. Tuzel, “Multi-kernel object tracking,” *IEEE Intl. Conf. Multimedia and Expo.*, pp. 1234-1237, 2005.

[66] B. Prosser, S. Gong, and T. Xiang, “Multi-camera matching using bi-directional cumulative brightness transfer functions,” *British Machine Vision Conference*, 2008.

[67] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, “More efficiency in multiple kernel learning,” *ICML*, 2007.

[68] M. Rodriguez, J. Sivic, I. Laptev and J. Audibert, “Data-driven crowd analysis in videos,” *IEEE Intl. Conf. on Computer Vision*, pp. 1235-1242, 2011.

[69] K. Rose, “Deterministic annealing for clustering, compression, classification, regression, and related optimization problems,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210-2239, 1998.

[70] Y. Rubner, C. Tomasi, and L. J. Guibas, “The earth mover’s distance as a metric for image retrieval,” *Intl. Journal of Computer Vision*, 40(2), pp. 99-121, 2000.

[71] R. Sinkhorn, “A relationship between arbitrary positive matrices and doubly stochastic matrices,” The Ann. Math. Statist., pp. 876-879, 1964.

[72] R. Snelick, U. Uludag, A. Mink, M. Indovina and A. Jain, “Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, Mar 2005.

[73] Jan A. Snyman, *Practical Mathematical Optimization*, New York: Springer Science + Business Media, 2005, chapter 3.

[74] K. Tieu, G. Dalley, and W. Grimson, “Inference of non-overlapping camera network topology by measuring statistical dependence,” *Proc. IEEE Conf. on Computer Vision*, vol. 2, pp. 1842-1849, 2004.

[75] UK Home Office, “The Image Library for Intelligent Detection Systems (i-LIDS): Multiple Camera Tracking (MCT),” 2008. *http://www.homeoffice.gov.uk/science-research/hosdb/i-lids/*

[76] S. Wang, H. Lu, F. Yang, and M. Yang, “Superpixel tracking,” *IEEE Intl. Conf. Computer Vision*, 2011.

[77] G. Welch and G. Bishop, “An introduction to the kalman filter,” *Technical Report, University of North Carolina at Chapel Hill*, 1995.

[78] J. Wright, Y. Peng, Y. Ma, A. Ganesh, and S. Rao, “Robust Principal Component Analysis: Exact recovery of corrupted low-rank matrices by convex optimization,” *Neural Information Processing Systems (NIPS)*, Dec, 2009.

[79] V. Yang, R. Duraiswami, and L. Davis, “Efficient mean-shift tracking via a new similarity measure,” *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 176-183, 2005.

[80] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” ACM Computing Surveys, vol. 38, Issue 4, 2006.

[81] A. Yilmaz, “Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1-6, 2007.

[82] H. Zhang, W. Huang, Z. Huang, L. Li, “Affine object tracking with kernel-based spatial-color representation,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 293-300, 2005.

[83] Y. Zhang and S. Li, “Gabor-LBP based region covariance desciptor for person rei-identification,” *IEEE Intl. Conf. on Image and Graphics*, pp. 368-371, 2011.

[84] W. Zheng, S. Gong and T. Xiang, “Person re-identification by probabilistic relative distance comparison,” *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 649-656, 2011.

[85] W. Zheng, S. Gong and T. Xiang, “Transfer re-identification: From person to set-based verification,” *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 649-656, 2012.

[86] J. Zhou and B. Li, “Homography-based ground detection for a mobile robot platform using a single camera,” *Proc. IEEE Conf. on Robotics and Automation*, May 2006.

[87] L. Zhu, J. Hwang, and H. Cheng, “Tracking of multiple objects across multiple cameras with overlapping and non- overlapping view,” *IEEE Intl. Symposium on Circuits and Systems*, pp. 1056-10360, Taipei, 2009.

[88] Z. Zhu, Q. Ji, K. Fujimura, and K. Lee, “Combining kalman filtering and mean shift for real time eye tracking under active IR illumination,” *IEEE Intl. Conf. Pattern Recognition*, vol. 4, pp. 318-321, 2002.

Appendix A: Projected Gradient-based Multiple-Kernel Tracking

A1 : Decomposition of the movement vector in multiple-kernel tracking

The projection matrix $\mathbf{A}$ which projects the vector onto the space in which the values of the constraint functions remain intact is expressed as $(\mathbf{I}-\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C_x^T})$. After applying $\mathbf{A}$ to $-\mathbf{J_x}$, the gradient descent direction, the first term $\delta\mathbf{x}_A$ is $(-\mathbf{I}+\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C_x^T})\mathbf{J_x}$ [73]. In addition to this, the constraint functions $\mathbf{C}(\mathbf{x})=\mathbf{0}$ need to be satisfied; hence, another moving vector $\delta\mathbf{x}_\mathrm{B}$ enabling the constraints to hold is introduced; that is, to make the constraint functions approach zero when moving along $\delta\mathbf{x}_\mathrm{B}$,

$$\mathbf{0}=\mathbf{C}(\mathbf{x}+\delta\mathbf{x}_\mathrm{B})\approx\mathbf{C}(\mathbf{x})+\mathbf{C_x^T}\delta\mathbf{x}_\mathrm{B}\ . \tag{A.1}$$

Thus, $\delta\mathbf{x}_\mathrm{B}=-\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C}(\mathbf{x})$.

A2: Characteristics of the projected gradient vector

The characteristics of the projected gradient vector can be proved in the following:

i)

$$(\delta\mathbf{x}_A)^T\delta\mathbf{x}_B \tag{A.2}$$

$$=(\alpha(-\mathbf{I}+\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C_x^T})\mathbf{J_x})^T(-\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C}(\mathbf{x}))$$

$$=\alpha(-\mathbf{J_x}^T+\mathbf{J_x}^T\mathbf{C_x}((\mathbf{C_x^T C_x})^{-1})^T\mathbf{C_x^T})(-\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C}(\mathbf{x}))$$

$$=\alpha(\mathbf{J_x}^T\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C}(\mathbf{x})-\mathbf{J_x}^T\mathbf{C_x}((\mathbf{C_x^T C_x})^{-1})^T\mathbf{C_x^T C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C}(\mathbf{x}))$$

$$=\alpha\left(\mathbf{J_x}^T\mathbf{C_x}(\mathbf{C_x^T C_x})^{-1}\mathbf{C}(\mathbf{x})-\mathbf{J_x}^T\mathbf{C_x}((\mathbf{C_x^T C_x})^{-1})^T\mathbf{C}(\mathbf{x})\right)$$

$$=0\ .$$

Hence, $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$ are orthogonal to each other.

ii)

Let δJ_A and $\delta\mathbf{C}_A$ denote the changes of functions $J(\mathbf{x})$ and $\mathbf{C}(\mathbf{x})$ by moving along $\delta\mathbf{x}_A$, respectively. Assume the local linear approximation:

$\delta J_A \approx \mathbf{J_x}^T \delta \mathbf{x}_A = \mathbf{J_x}^T \alpha(-\mathbf{I} + \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\mathbf{J_x}$ (A.3)

$=-\alpha \mathbf{J_x}^T(\mathbf{I} - \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\mathbf{J_x}$.

Let matrix $\mathbf{M} = \begin{bmatrix} \mathbf{C_x}^T\mathbf{C_x} & \mathbf{C_x}^T \\ \mathbf{C_x} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{C_x}^T \\ \mathbf{I}^T \end{bmatrix} [\mathbf{C_x} \quad \mathbf{I}] = \mathbf{K}^T\mathbf{K}$, which is always positive semi-definite. Moreover, since $\mathbf{C_x}^T\mathbf{C_x}$ is also positive semi-definite, the generalized Schur complement $(\mathbf{I} - \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)$ is also positive semi-definite. Thus, $\mathbf{J_x}^T(\mathbf{I} - \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\mathbf{J_x} \geq 0$ and $\delta J_A \leq 0$. The cost function always decreases if moving along the direction $\delta \mathbf{x}_A$. Moreover,

$\delta \mathbf{C}_A \approx \mathbf{C_x}^T \delta \mathbf{x}_A = \mathbf{C_x}^T \alpha(-\mathbf{I} + \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\mathbf{J_x}$ (A.4)

$=\alpha(-\mathbf{C_x}^T + \mathbf{C_x}^T\mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\mathbf{J_x}$

$= \alpha(-\mathbf{C_x}^T + \mathbf{C_x}^T)\mathbf{J_x} = \mathbf{0}$.

Thus, it will not change the values of the constraint functions by moving along the direction $\delta \mathbf{x}_A$.

iii)

Let $\delta \mathbf{C}_B$ denote the changes of functions $\mathbf{C}(\mathbf{x})$ by moving along the $\delta \mathbf{x}_B$. Assume the local linear approximation:

$\delta \mathbf{C}_B \approx \mathbf{C_x}^T \delta \mathbf{x}_B = \mathbf{C_x}^T(-\mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C}(\mathbf{x})) = -\mathbf{C}(\mathbf{x})$. (A.5)

Thus, no matter what the current values of constraint functions are, they will always go toward the value zero, i.e., make our problem constraints hold $\mathbf{C}(\mathbf{x}) = \mathbf{0}$.

A3: Using mean-shift vector as gradient vector

To minimize the cost function $J(\mathbf{x})$ is equivalent to maximizing $-J(\mathbf{x})$. Similar to the derivation in [26], we take the linear approximation of $-J(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}_0$ and obtain the following equation after some manipulations,

$$-J(\mathbf{x}) = h(\mathbf{x}_0) + c_1 \sum_i \omega_i k\left(\left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2\right) \quad (A.6)$$

$$= h(\mathbf{x}_0) + f_K(\mathbf{x}),$$

where $J(\mathbf{x}) = \sum_{i=1}^{m} q_i log\left(\frac{q_i}{p_i(\mathbf{x})}\right)$ is K-L distance between two histograms: $\mathbf{p}(\mathbf{x}) = [p_1(\mathbf{x}) \ldots p_m(\mathbf{x})]$ is *m-bin* candidate model and $\mathbf{q} = [q_1 \ldots q_m]$ is the target model. $h(\mathbf{x}_0)$

and c_1 can be seen as constants (independent to $\mathbf{x}$); $\omega_{\mathrm{i}} = \sum_{u=1}^{m} \frac{q_{\mathrm{u}}}{p_{\mathrm{u}}(\mathbf{x}_0)} \delta[b(\mathbf{z_i}) - u]$ and $\mathbf{z_i}$ is the pixel location.

Thus, we can get $\mathbf{J_x} = \frac{\partial J}{\partial \mathbf{x}} = -\nabla f_K(\mathbf{x})$. (A.7)

According to [25], the mean shift vector corresponding to maximizing the function $f_K(\mathbf{x})$ can be expressed as:

$$\mathbf{m}(\mathbf{x}) = c_2 \times \frac{\nabla f_K(\mathbf{x})}{f_G(\mathbf{x})},\ c_2 > 0, \tag{A.8}$$

where $f_G(\mathbf{x}) = \sum_i \omega_{\mathrm{i}} g\left(\left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2\right)$ and $g(\cdot) = -k'(\cdot)$. Due to the positivity of ω_{i} and the monotonous decreasing of the kernel profile $k(\cdot)$ (i.e., $k'(\cdot)$ is negative), $f_G(\mathbf{x})$ is always positive. Hence, from (A.7) and (A.8) we can get

$$\mathbf{J_x} = c_3 \times \big(-\mathbf{m}(\mathbf{x})\big),\ c_3 > 0; \tag{A.9}$$

that is, $\mathbf{J_x}$ and $\big(-\mathbf{m}(\mathbf{x})\big)$ are aligned. Therefore, for each kernel, it is reasonable to use the mean shift vector with the opposite direction as our $\mathbf{J_x}$.

In the characteristic (ii) in Appendix A2, if we use $-\mathbf{m}(\mathbf{x})$ in calculating $\delta\mathbf{x}_A$,

$$\begin{aligned} \delta J_A \approx \mathbf{J_x}^T \delta\mathbf{x}_A &= \mathbf{J_x}^T \alpha(-\mathbf{I} + \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\big(-\mathbf{m}(\mathbf{x})\big) \\ &= -\frac{\alpha}{c_3}\mathbf{J_x}^T(\mathbf{I} - \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\mathbf{J_x} \le 0, \end{aligned} \tag{A.10}$$

which will not affect the non-positivity of δJ_A. Similarly, the values of the constraint functions will not change.

$$\begin{aligned} \delta\mathbf{C}_A \approx \mathbf{C_x}^T\delta\mathbf{x}_A &= \mathbf{C_x}^T\alpha(-\mathbf{I} + \mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T)\big(-\mathbf{m}(\mathbf{x})\big) \\ &= \alpha(-\mathbf{C_x}^T + \mathbf{C_x}^T)\big(-\mathbf{m}(\mathbf{x})\big) = \mathbf{0} \end{aligned} \tag{A.11}$$

Vita

Chun-Te Chu was born in Taipei, Taiwan. He received his Bachelor of Science degree in the Department of Electrical Engineering from National Taiwan University in 2006. In 2008, he joined Information Processing Lab at University of Washington and pursued his Ph.D degree. He got a Master of Science degree in the Department of Electrical Engineering from University of Washington in 2010. In 2013, he earned a Doctor of Philosophy degree in the Department of Electrical Engineering at University of Washington. His research interests are in computer vision, machine learning, and video/image processing.