國立交通大學

資訊科學與工程研究所

碩 士 論 文

隧 道 監 控 系 統 之 多 攝 影 機 車 輛 辨 識

Multi-Camera Vehicle Identification
in Tunnel Surveillance System

研 究 生：朱明初

指導教授：李素瑛　教授 、 陳華總　教授

中 華 民 國　一 百 零 二　年 七 月

隧道監控系統之多攝影機車輛辨識

研究生：朱明初　　指導老師：李素瑛 教授

陳華總 教授

國立交通大學資訊科學與工程研究所

摘 要

隧道內交通意外往往會造成巨大災害且難以處理，因此有大量監視攝影機裝設於隧道中，可即時發現事故並監控路況。但通常並沒有足夠的人力來觀看大量的監視器畫面，使得自動化監控系統的需求增加。本論文提出一種多攝影機車輛辨識系統，利用隧道內多攝影機的監視器畫面追蹤行車在隧道內的位置。

於單一監視器畫面中，使用 Haar-like 特徵偵測找出車輛，並取出 OpponentSIFT 影像特徵。接著，本論文提出的空間時間連續關係動態規劃(S^2DP)演算法，利用隧道內行車順序關係性，辨識前後兩台攝影機中所偵測到的車輛。此外亦提供兩種進階辨識方法，包含即時運算(RT)方法以及非即時加強處理(OR)。即時運算方法減少車輛配對之搜尋範圍，並快速比對兩攝影機內之車輛。而非即時方法針對空間時間連續關係動態規劃演算法中無法有效配對的行車做進一步處理。

實驗結果顯示所提出之多攝影機車輛辨識系統可得到滿意的準確程度，並優於其他相關演算法。

關鍵字：影像監控、隧道監控、多攝影機車輛辨識、智慧交通系統

Multi-Camera Vehicle Identification in Tunnel Surveillance System

Student: Ming-Chu Chu

Advisor: Prof. Suh-Yin Lee

Prof. Hua-Tsung Chen

Department of Computer Science,

National Chiao Tung University

Abstract

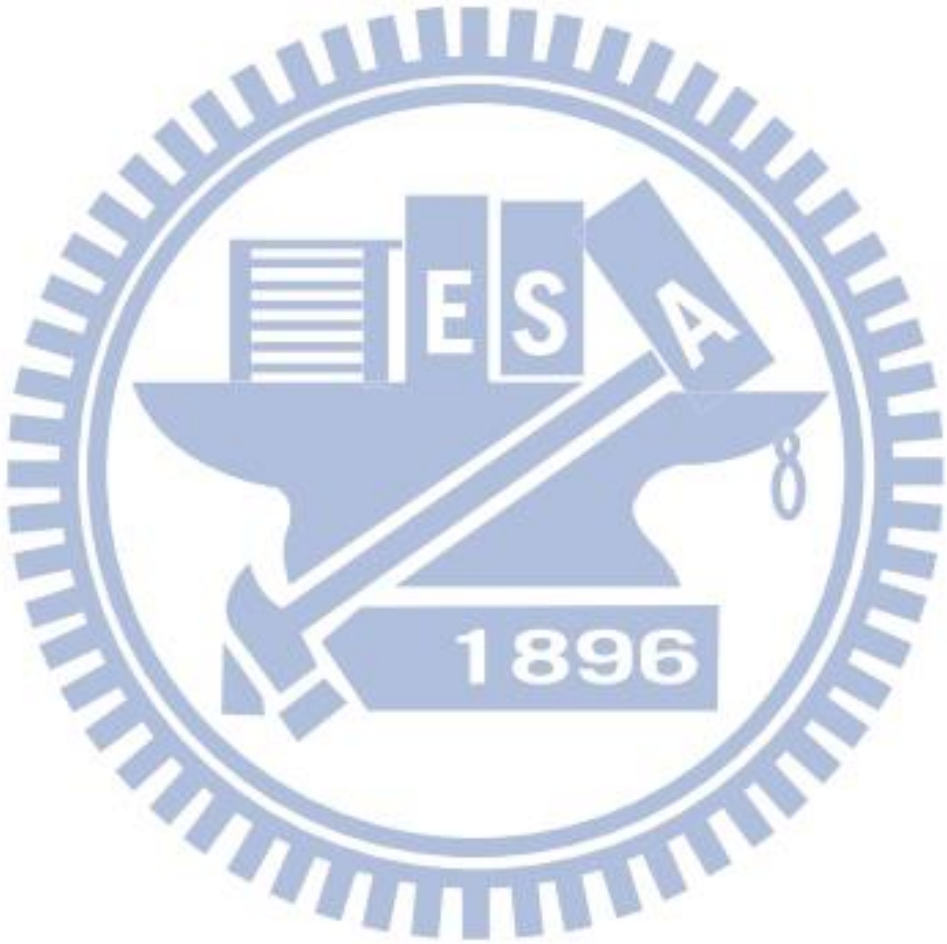
Surveillance cameras are widely equipped in tunnels to monitor the traffic condition and traffic safety issues. Identifying vehicles from multiple cameras within a tunnel automatically is essential to analyze traffic condition through the road. This thesis proposes a multi-camera vehicle identification system for tunnel surveillance videos.

Vehicles are detected using Haar-like feature detector and their image features are extracted using OpponentSIFT descriptor in single camera. The proposed Spatiotemporal Successive Dynamic Programming (S^2DP) algorithm identifies vehicles from two cameras by considering the ordering constraint in the tunnel environment. Next, two methods Real-Time (RT) algorithm and Offline Refinement (OR) algorithm are proposed for different requirements. The RT fast identifies vehicles in real-time by searching a limited range of candidates, and the OR refines the identification result from the S^2DP.

Comprehensive experiments on various datasets demonstrate the satisfactory

performance of the proposed multi-camera vehicle identification methods, which outperform state-of-the-art algorithms.

Keyword: video surveillance, tunnel surveillance, multi-camera vehicle identification, intelligent transportation system



Acknowledgements

I would like to thank my advisor Prof. Suh-Yin Lee and Prof. Hua-Tsung Chen who gave me strong advices, valuable comments and precious experiences on doing a research and gave me the chance to work on my own. This thesis cannot be as complete as now without their grateful assistance.

I would like to thank Mr. Chien-Li Chou for our extensive discussions on our research topic every day. I also would like to thank all staff in the Information System Laboratory, NCTU, for their support and encouragement.

Thanks to my father, my mother, and my brother. Thanks to my lovely family.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

In recent years, video cameras are equipped everywhere in our daily life due to the affordable price and easy installation of devices. People can record a mass amount of events passing though the scene by storing the data within videos. However, most of videos contain a lot of redundancies. What we care about is only very small subsets or portions of the videos with semantic meanings. For example, in a traffic surveillance system, what we are interested in is the traffic density of a road section [1] or whether an incident happened [2], and then we can select other path to avoid those sections. For cameras set on streets, we would like to know how many pedestrians passed [3] or if there is any unusual activity [4]; in our living room, system may send alarms if elderly people fall on floor or some dangerous events occur [5]. Therefore, methods to retrieve and summarize useful data efficiently are essential for handling the huge amount of videos.

Research topics on surveillance systems have been discussed in the last decades, for example, object detection, tracking and identification, scene understanding, and event detection [6] [7] [8] [9]. Although different applications are developed for different scenarios, many basic techniques can be applied to most of the surveillance videos. Object detection is usually the first step of a surveillance system to locate the region of interests (ROI), precisely, the object within a scene. After a sequence of video frames is processed, we need to know whether two detected objects from two frames are the same one. This is called object identification or object tracking. Trajectories and paths of detected objects are collected after tracking objects in a period of time. We can use the information to retrieve high-level semantics, such as

the number of vehicles passed by and events, car accidents for example.

A traffic surveillance system can reveal various kinds of information. We can monitor the traffic condition by analyzing the videos collected from cameras installed along roads and highways. Intuitively, traffic data like the level of driving speed or the number of vehicles passing by can be provided to drivers or used in navigation systems to avoid congestion areas. Another type of information is whether a specific event happens, like traffic accidents or traffic rule violation. The police nearby can receive alarms from the control center and come to the location quickly. For long-time monitoring, traffic data are stored in databases and we can discuss issues on a road section about congestion or retrieve historic records about some specific vehicles driven by criminals.

In recent years, researchers focus on tunnel surveillance since accidents in a tunnel may cause serious problems [10]. The traffic agency monitors the traffic condition of a tunnel using multiple surveillance cameras and tries to discover unusual events in real-time. However, that is not an easy task since in most of time monitoring is nothing interesting and makes workers hard to concentrate on the screens. Another shortcoming is that sometimes there are not enough cameras to cover the whole tunnel scene. There are temporal and spatial gaps between videos. Therefore, it is necessary to develop a computer system that can automatically provide precise and brief information. Figure 1-1 shows an example of a tunnel surveillance system that contains multiple cameras. Many surveillance systems on day-time traffic can be directly applied to tunnels because major features are the same as in tunnels. However, more challenges such as poor illumination conditions are present. It is worth considering more aspects on tunnels than on day-time traffic to achieve better

performance.

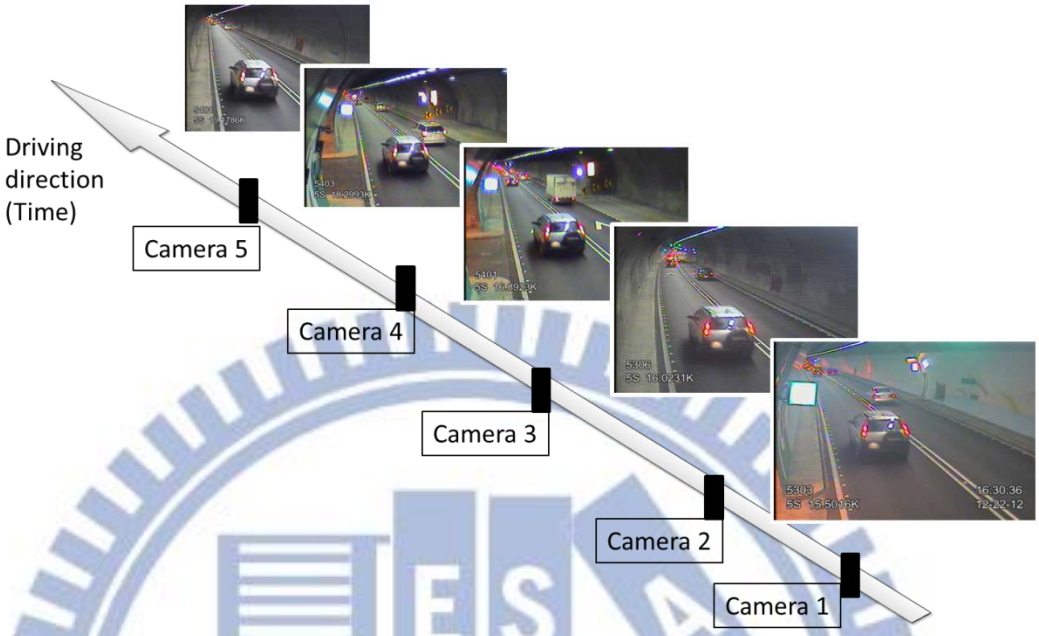


Figure 1-1. An example of a tunnel surveillance system.

There are several difficulties for traffic surveillance: low resolution, less frame rate, limited view, and camera sensor noises. Since we have to install a huge number of cameras and to store a mass amount of video data, the price of camera is relatively low and the size of data should be as small as possible. That lowers the quality of videos and makes the analysis of videos more challenging. In addition, there are some differences between surveillance systems on highways and on tunnels. Most systems on road can only work in day-time. In this case, many basic algorithms can be applied due to sufficient light. However, illumination effects usually exist in tunnels, producing more noise and unpredictable troubles. Fortunately, there are still some benefits in the environments of tunnels, like fewer lanes, more strict driving constraints, and the 24-hour system with the same lighting condition.

Another issue is that many traffic surveillance systems do not consider about information between two or more cameras. As a car is driven on a road through multiple cameras, we would first like to know whether two vehicles in two videos are the same. This task is called multi-camera object tracking or multi-camera object identification. It is more challenging than tracking in a single camera, because the view points, poses and lightings are different. These differences may make the visual appearances of an object different in different cameras. Sometimes we cannot track a vehicle across cameras because there are time and space gaps in between. However, we can still identify them by considering the features of each vehicle. A naïve method to identify vehicles between cameras is obtaining the unique license plate number using license plate recognition. However, it is usually not available in a surveillance system due to the poor quality of videos. Hence the robust image features are needed for multi-camera object identification.

In this thesis, we propose a multi-camera vehicle identification system in tunnels. First vehicle detection and tracking are performed in single tunnel surveillance video using Haar-feature-based cascade detector [11]. After images of vehicles are collected from videos, the visual features of images are then extracted. Features such as color histograms, Haar-like feature vector, SIFT-based feature points and template matching in pixel domain are studied and evaluated by experiments. Next we propose a multi-camera vehicle identification method to identify vehicles between two non-overlapping views of different cameras using calculated feature vectors. The first step is the Spatiotemporal Successive Dynamic Programming (S^2DP) algorithm that matches vehicles in two cameras. And then two different algorithms for real-time tracking and offline refinement are proposed for different requirements: the Real-Time (RT) and the Offline Refinement (OR) algorithms following S^2DP,

respectively. Finally the experiments and discussions on proposed methods are presented in this thesis.

The remaining of this thesis is organized as follows. In Chapter 2, we introduce the related work about surveillance video analysis. Chapter 3 describes the proposed tunnel surveillance system and the multi-camera vehicle identification methods. The experiment settings and experimental results are presented and discussed in Chapter 4. Finally, conclusions and discussions of future work are in Chapter 5.

Chapter 2. Related Work

In this chapter, we review the related literatures of video surveillance systems. A survey on methods of object detection, tracking, and multi-camera object identification is presented.

2.1 Video Surveillance Systems

Video surveillance systems are widely used nowadays since cameras are installed everywhere. Applications can be briefly divided by their operating environments, like in-door or out-door and for vehicles or pedestrians. Nevertheless most systems share similar characteristics and require common techniques.

Excellent video processing and computer vision techniques are necessary for a video surveillance system. Researchers try to retrieve useful information and understand the semantics from videos since it is intuitive for human obtaining information from what we see. Buch *et al*. [12] review the state-of-the-art computer vision techniques for urban traffic system. The common challenges include poor quality of data, wide range of operational conditions and environments, and real-time processing. Key techniques to establish a video-based traffic system are foreground segmentation, shadow removal, feature selection, object classification, and tracking, which are common research topics in computer vision. Zhang *et al*. [13] survey recent research on data-driven intelligent transportation systems (ITS). The main components of data are from videos, because people are more familiar with visual information. Applications like vehicles or pedestrian detection and tracking, behavior

analysis, incident detection, density estimation are discussed often. And systems can be extended by mixing data from different sensors like global positioning system (GPS), laser or infrared radars.

A traffic surveillance system can provide not only real-time processing like vehicle tracking or event detection, but also offline analysis, traffic condition investigation or database indexing for example. Feris *et al*. [6] propose a system for large-scale indexing of vehicles in a video surveillance system. The idea is providing a database of vehicles in urban environment, with semantic attributes like time, color, size, etc. This can help police easily search for suspicious vehicles and reduce efforts in criminal investigation process. They use some easy but strict constraints to automatically collect and generate a large amount of training patches for vehicle detectors of Haar-based features. Then attributes like time, color, driving direction, color, size, speed, are retrieved for each detected vehicles, and the information is stored into a database.

Sometimes we would like to know the semantic of a scene instead of a specific object, which leads to the topic of scene understanding. For example, the system can automatically discover the moving directions of crowds [8], or the driving paths of vehicles in a scene [14]. The trajectories and motion flows of moving objects can provide features of regular activities. Hence we can discover unusual events by analyzing outliers and anomalies. Atev *et al*. [14] and Morris and Trivedi [7] try to model the driving behaviors of a road scene in a surveillance video. Since a vast amount of cameras have already been set on road sections and activities might change often, it is necessary to develop an unsupervised way to build the model automatically. By collecting the trajectories of vehicles passing through the scene, both methods

apply clustering algorithms and treat each cluster as a usual activity. Morris and Trivedi [7] then train a Hidden Markov Model (HMM) for each activity that can be used as a trajectory classifier. Finally when a new trajectory is obtained, its distance to each cluster can be calculated and abnormal events can be detected. In another aspect, the normal behavior of a road scene can help object detection and tracking. Additional information could assist to predict where the object might be and where it would possibly move to.

2.2 Object Detection and Tracking

A surveillance video contains both specific objects that we are interested in and other unrelated regions, and therefore object detection is usually the first step in a system. As objects are detected in each frame, we need to know whether two detections from two consecutive frames belong to the same object. This requires tracking of each object. Usually both steps are required in a video surveillance system to provide information for further processing.

2.2.1 Object Detection

Sometimes the objects might move, and background subtraction [15] [16] [17] can be applied to videos to obtain those moving objects. Still there are some object detection algorithms in image processing domain like face detection [11] and pedestrian detection [18] techniques, which can also be applied to surveillance video [9] [19] .

Background subtraction is a scheme to detect moving objects in videos from static cameras [15], which computes the difference between current frame and a background model, considering the idea that an object is likely different from the background. This is widely applied to surveillance systems since cameras usually are static, and the computation cost is relatively lower than object detection methods. The commonly used approaches include running Gaussian average [20] and Mixture of Gaussians (MoG) [17], which consider the background model as a normal distribution or a mixture of them in pixel domain and detecting foreground object if it is different from the distribution. Brutzer *et al*. [16] evaluate commonly used and state-of-the-art background subtraction algorithms. The testing scenarios contain dynamic background, darkening, light switch, noise, shadow, and recommended some state-of-the-art algorithms. However, they also state that most background subtraction methods would fail in conditions like noisy night and sudden lighting change, which are quite common in tunnel environments.

Another approach is appearance-based object detection, where the desired objects share common characteristics in an image. Examples include Haar-like feature with cascade classifier [11] and Histogram of Oriented Gradient (HoG) [18], which are widely used in video surveillance systems [6] [10] [19]. Unlike background subtraction methods that usually consider the changes between frames, appearance-based methods treat a video as individual images, not related to neighbor frames. They often require a training step to obtain the common property between objects, and then use sliding window approach to scan the frame and verify whether the region is an object. Appearance-based object detection often needs more computation cost since it requires exhaustive search on the image, but the accuracy is better because it can only find objects similar to the training set.

2.2.2 Object Tracking

Object tracking is to figure out where the object is in a video, and the sequence of tracking results can be arranged into a trajectory that presents the history of the movement. An intuitive way is to group detections from two consecutive frames that are in nearby locations. However, there might be noises around the observation. Hence Bayesian tracking [21] can be applied to tracking process. By applying probability formulation, we can track objects more precisely. The common solutions for Bayesian tracking are Kalman Filter and Particle Filter, which constantly update the tracker by taking new observations. Breitenstein *et al.* [19] propose a multi-person tracking framework using Particle Filter. It is very challenging for object tracking using simple grouping methods in a complex environment because detection cannot perform well. Therefore, filters are necessary in this case. The confidence of detected objects is considered for detections and this solves data association for multi-person tracking.

2.3 Multi-Camera Object Identification and Tracking

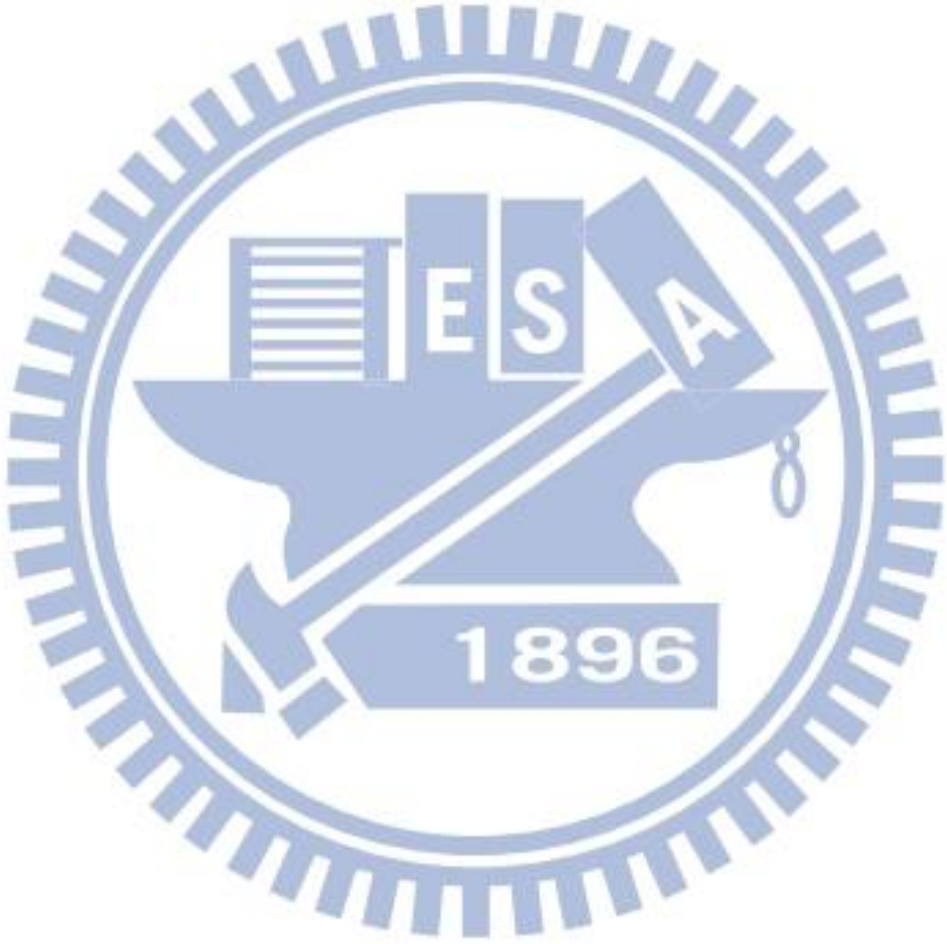
An area usually contains multiple surveillance cameras, and it is necessary to mix data from two or more places to obtain more complex information. However, multiple camera processing is far more difficult than single camera processing. For example, lighting, viewpoints, and background in two cameras can be different. Wang [22] reviews recent researches in multi-camera video surveillance. Key technologies in multi-camera systems are multi-camera calibration, computing topology of camera network, tracking, re-identification, and activity analysis. All of them face challenges

including various configurations, limited topology, large changes of viewpoints, and illumination conditions. Some of the technologies can be jointly solved. Just as in this thesis, we try to solve the object re-identification problem to achieve multi-camera tracking.

When tracking an object across non-overlapping views of two cameras, two major problems should be solved. Information may not be continuous because there are time and space gaps between two cameras, and the visual appearance changes due to different view angle and lighting of different cameras. Lian *et al*. [23] propose a method of tracking pedestrians across two cameras using only visual appearance. They calculate the CI-DLBP, the enhanced distance-based local binary pattern (LBP) features that include color information, for each pedestrian. And then match pedestrians from two cameras using the feature vectors to achieve multi-camera tracking. Instead of considering only LBP that describes structural information, color is also an important feature for matching objects from different camera views. They encode a detection of a pedestrian to a histogram of LBP in color space and match two histograms from two different cameras using Chi-square distance, and apply cumulative brightness transfer function (CBTF) to reduce the illumination effect in pre-processing steps.

Cabrera *et al*. [9] propose a method of multi-camera vehicle tracking-by-identification in a tunnel surveillance system. Like [6], they use Haar-like feature to detect vehicles in a single camera and obtain a feature vector with binary values of weak classifiers for all detections. Next they use Hamming distance to find the best matched pairs for each vehicle in both single camera tracking and multi-camera identification. The two major advantages of this method are: the

computation of Haar-like features can be efficiently obtained, and the lower cost in passing and comparing feature vectors with binary values. The authors state that color is not reliable in tunnels because it may be affected by lighting. However we think that color is still a strong feature for vehicles especially when the resolution is quite low in surveillance video. Some practical issues have not been discussed, such as system initialization and error handling, which is presented in this thesis.



Chapter 3. Multi-Camera Vehicle Identification

In this chapter, we illustrate the multi-camera vehicle identification framework in detail. An overview will be given in Section 3.1 and single-camera vehicle detection and tracking is described in Section 3.2. Next, possible image features for identification is presented in Section 3.3. The proposed multi-camera vehicle identification method and the real-time and offline identification are illustrated in Section 3.4 and 3.5, respectively.

3.1 An Overview of the Proposed Framework

Figure 3-1 shows the framework of the proposed system. Starting from a single camera, we first extract vehicles passing through the scene by object detection and tracking. After a period of time we can collect a set of vehicle patches. The number of detected vehicles in the first camera is always more than the number in the next camera, since there are vehicles passing the first camera but not the second one. In other words, when a vehicle is presented in the second camera, multi-camera tracking/identification is to search for the corresponding detection in the first camera that represents the same vehicle.

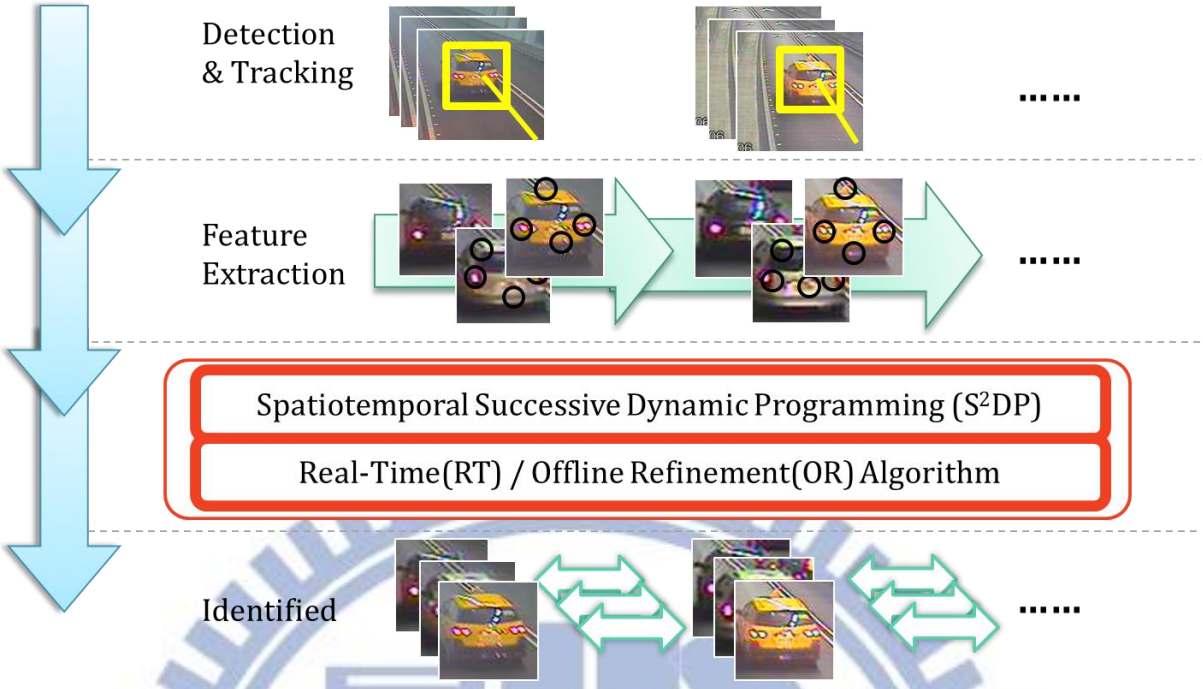


Figure 3-1. System framework.

In a single camera, all passing vehicles are discovered using Haar-like feature detector [11] for each frame. And within a small interval of a video sequence, vehicles are tracked by considering the location in the image of detections from consecutive video frames. As vehicles in tunnel travel in sequence, we can use a queue to store all tracked vehicles to preserve the order. Single-camera vehicle detection and tracking will be discussed in Section 3.2 in detail and a dataset of the detection results with five cameras are presented in Section 4.1.

For object recognition and identification, we extract features from the object images. Color and visual structure information are important features for vehicle identification in a low resolution surveillance video. Section 3.3 discusses some commonly used feature descriptors that encode the visual features, and Section 4.2 evaluates these descriptors.

After a period of time, we can collect a set of queues with detected vehicles from several cameras in the same tunnel. We need to initialize the system first to verify which vehicle is the first one that exists in all queues. Because there may be vehicles between two cameras, and when we start recording multiple cameras at the same time, there may be vehicles in latter cameras that do not appear in the previous ones. One solution is to start recording multiple cameras in different time, however it is still hard to correctly make one vehicle become the first entity in all detection queues, unless there is no vehicle in the tunnel. We accomplish the task by considering the ordering constraint within a tunnel. That is, one vehicle is behind another most of the time and the order rarely changes, as well as lane changing is often prohibited in a tunnel. Therefore, a multi-camera identification method is proposed based on the ordering constraints of vehicles. In the beginning the Spatiotemporal Successive Dynamic Programming (S^2DP) algorithm matches vehicles from two cameras. Next, the Real-Time (RT) algorithm can be applied after S^2DP for real-time vehicle tracking, and the Offline Refinement (OR) algorithm can further achieve higher accuracy from the results of S^2DP.

The S^2DP algorithm matches vehicles in two detection queues, and thus achieves multi-camera identification. However, it has to check a group of detections at the same time and cannot run in real-time. The RT can accomplish the identification task in real-time, picking up the most similar one in a small search window of the detection queue. Nevertheless, real-time algorithm usually shows lower accuracy, and is sometimes not suitable for database applications, which do not require real-time processing but the accuracy need to be assured. In this case we propose another offline identification method, OR, which is used to achieve higher accuracy by refining the results from S^2DP. The real-time and offline algorithms are discussed in

Section 3.5 and experiments are presented in Section 4.4.

3.2 Vehicle Detection and Tracking in Single Camera

The first step of the system is to obtain information from cameras independently, thus we will discuss single-camera vehicle detection and tracking method in this section. We use Haar-like feature detector [11], which is commonly used in face detection, to detect vehicles in a tunnel surveillance video. And the same method is used in [6] and [10] for vehicle detection. For a sequence of detections from consecutive video frames, we group them by considering the location of detections in x-y coordinate of frames to achieve vehicle tracking.

3.2.1 Vehicle Detection

We use Haar-like feature detector [11] to detect vehicles in a video frame. The Haar-like features are a set of binary values which consists of the differences between two or more sums of intensities in rectangle regions of a grayscale image, as shown in Figure 3.2. Since there are various sizes of rectangles, a 24*24 resolution image, which is the size of our training images, can obtain a great amount of features in the exhaustive set of rectangle regions. As the visual structures of vehicles are similar in surveillance videos, vehicles share the same subset of Haar-like features, which do not appear in other regions of the videos. Therefore, we can discover those shared Haar-like features by applying machine learning algorithms and obtain a vehicle detector/classifier for videos. The learning algorithm uses a cascade of Adaboost classifiers to choose proper weak classifiers, the Haar-like features, and forms a stronger classifier. The cascade consists of a number of stages that reject negative

examples. Finally if an image passing all cascading stages, the detector will classify it as positive detection. Figure 3.3 illustrates the cascading scheme. The method uses integral images to fast calculate features, considers only a small subset of all possible Haar-like features, and rejects negatives in a short time by cascading stages. As the result, the algorithm can be used in real-time detection.

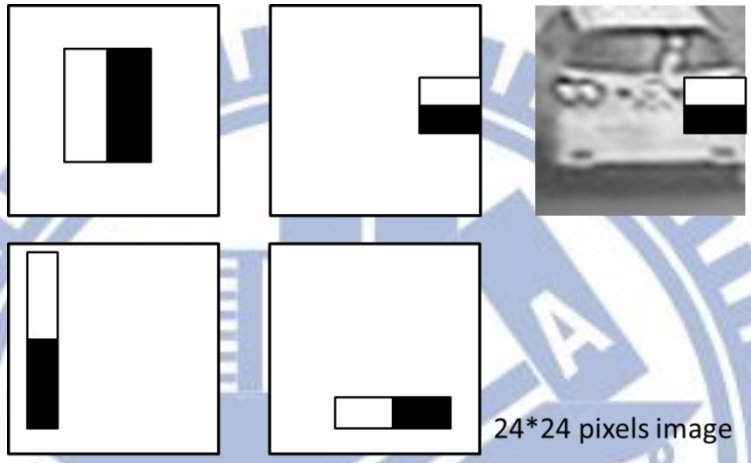


Figure 3-2. Haar-like features. Each feature is the difference between sums of intensities of two rectangle region.

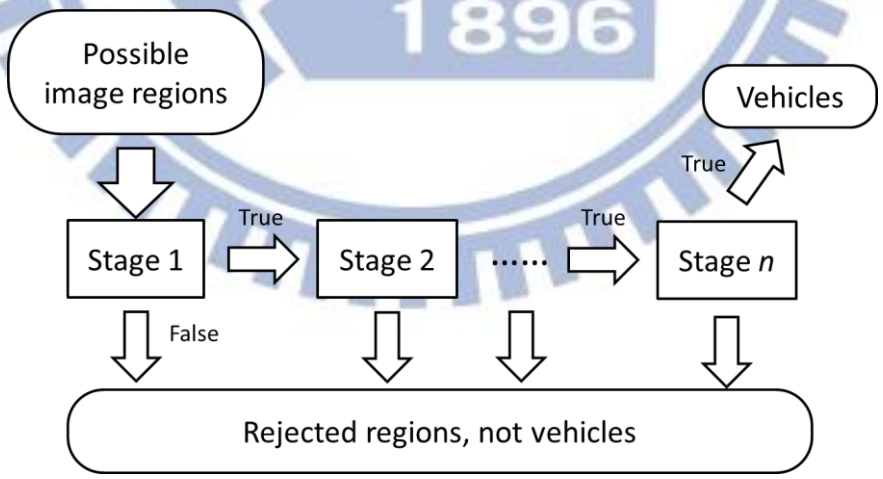


Figure 3-3. The cascading scheme of Haar-like feature detector.

The traditional method for object detection in surveillance video is background subtraction, because the objects usually move in the video. In fact, the vehicles always

move on the roads. However, reflections of headlights and taillights often exist on the wall of tunnels and those reflection regions might be treated as foreground objects in background subtraction methods. Another problem is that one vehicle may be occluded by another vehicle such that the algorithm may produce one big object instead of two. Figure 3-4 shows an example of problems in background subtraction using Mixture of Gaussians (MoG) [17] method, where the light reflections on walls are treated as objects. Hence, we choose feature detectors instead of background subtraction algorithms to achieve higher accuracy.

Figure 3-4. Detection fails using Mixture of Gaussians (MoG) background subtraction. Reflections on walls are treated as detected objects.

3.2.2 Vehicle Tracking

The goal of vehicle tracking is to find the corresponding detections of the same vehicle in different frames. Detection contains the image features, coordinates of the bounding box in image, detected time, and other related properties. We can keep track of detections which have similar properties. That is, if two detections from two consecutive frames have similar properties, these two detections present the same

vehicle. Since the detection is not accurate all the time and contains noise, it is common to apply Kalman Filter or Particle Filter [21] to predict the actual state of the object.

As vehicles always drive forward in tunnels, we simply take the coordinates of detections in an image as the properties for tracking. If two detections from two frames are in similar location within the image, we treat them as a single object. After one object is tracked in five consecutive frames, where the frames per second (FPS) of our video is about 10 and the vehicle is around the middle position of the video, the image of this object in the fifth frame is stored into the detection queue of the camera. Each camera maintains a queue of tracked vehicles, so that the ordering of passing vehicles can be recorded. The tracking step can also help us remove wrong detections because noises usually do not move. Every image stored in detection queue is resized to 40*40 pixels for further processing.

3.3 Feature Extraction

As we obtain a vehicle from one camera, we need to transfer the image patch into a feature vector that is proper for object identification. The goal is to select the best feature that can clearly identify whether two detections from two different cameras belong to the same one or not using distance metric. In this section we will introduce four possible feature descriptors for our multi-camera identification application. Experiments will be presented in Section 4.2.

The feature should minimize the effects on visual appearances in multi-camera identification: poses, color, illumination and noises. Figure 3-5 illustrates these

challenges. Another problem is that the images are in small resolution such that the details of a vehicle are hard to describe by features. Nevertheless we can take advantage of shorter computation time. In conclusion we choose Harris corner detection with OpponentSIFT [24] as our selected feature.

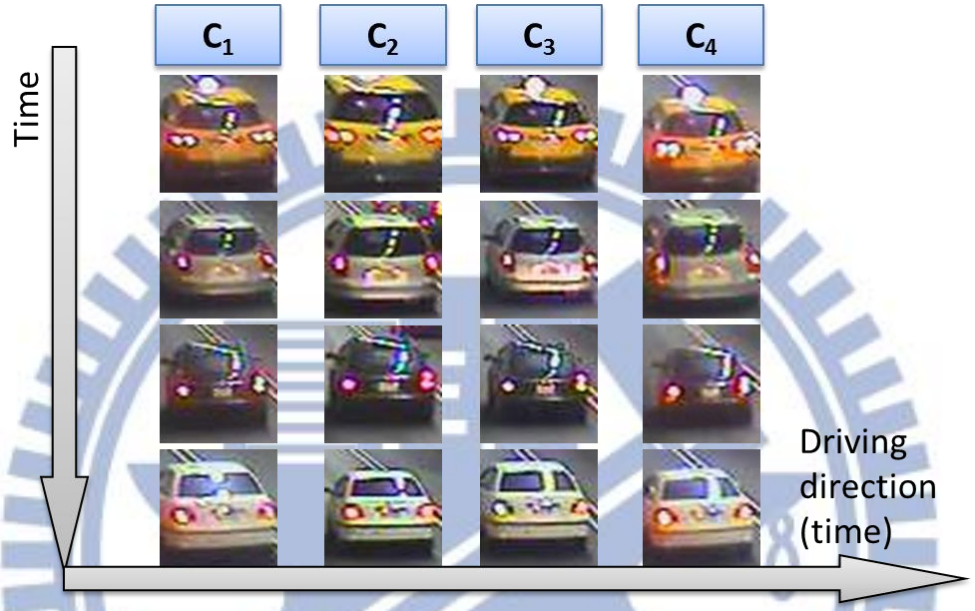


Figure 3-5. Examples of four vehicles in four different cameras. The size, pose, color, and reflections on windows are different in different cameras.

3.3.1 Image Intensity

To calculate the distance between two images, the simplest way is to obtain the sum of squared difference (SSD) of intensities between every pixel in them. In our application, the resolution of our detected vehicle is 40*40 for all images and the distance can be calculated in one pass for each color channel. Consider the application in RGB color space, and the image intensities (int) distance between two images I_1 and I_2 is

$$d_{int}(I_1, I_2) = \sum_{R,G,B} \sum_{x,y} \left(I_1(x,y) - I_2(x,y)\right)^2$$

where $I_i(x, y)$ is the intensity of image i at (x, y), and sum the values of all RGB channels.

3.3.2 Color Histograms

Histograms describe the distribution of intensities for an image. Here we tested histograms in three different color spaces: RGB, hue, and opponent color space. RGB histogram is a basic way to describe an image, since the input image is in RGB color space. Because color is one of the important features of vehicles, another selected color space is hue which describes the color distribution of an image. Hue is converted from RGB by

$$Hue = \begin{cases} 60(G-B)/(max(R,G,B) - min(R,G,B)) & if\ max(R,G,B) = R \\ 120 + 60(B-R)/(max(R,G,B) - min(R,G,B)) & if\ max(R,G,B) = G \\ 240 + 60(R-G)/(max(R,G,B) - min(R,G,B)) & if\ max(R,G,B) = B \end{cases}$$

Finally we choose histograms in opponent color space. The color space consists of three channels, the first two channels represent relation between RGB information and the third channel represents the grayscale intensity. Opponent color space can be converted from RGB using

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} (R-G)/\sqrt{2} \\ (R+G-2B)/\sqrt{6} \\ (R+G+B)/\sqrt{3} \end{pmatrix}$$

For every histogram, we extract 64 bins for each channel, therefore the RGB and opponent histogram has 192 bins and hue histogram has 64 bins. The distance between two histograms is the Chi-square distance

$$d_{CH}(H_1, H_2) = \sum_j \frac{\left(H_1(j) - H_2(j)\right)^2}{H_1(j)}$$

where $H_i(j)$ is the value of bin j in histogram H_i.

3.3.3 Haar-Like Feature Vector

Haar-like feature vector is obtained from the vehicle detector. It is a set of binary values with 143 dimensions, where we have 143 selected features in our trained detector. Cabrera *et al*. [10] stated that this feature can be used in multi-camera identification, without additional steps of feature calculation. Figure 3-2 shows an example of the feature. Hamming distance is used as distance metric since the feature vector contains only binary values, and the same vehicle would have similar result in the vector. The Hamming distance is computed by

$$d_{HV}(T_1, T_2) = \sum_j |T_1(j) - T_2(j)|$$

where T_i is the binary vector of image i, and $T_i(j)$ is the value of the jth element.

3.3.4 Keypoints Descriptors

Keypoints descriptors are commonly used in object recognition. Generally, some points in an image contain rich information. Those points are called keypoints, interesting points, or corners. Keypoints are extracted from the image, and then each point is represented with a specific descriptor. One of the famous keypoint extraction methods is Scale-Invariant Feature Transform (SIFT) [25], which is a powerful tool in many computer vision applications. SIFT finds keypoints using Difference-of-Gaussians (DoG) and describes each point using a histogram with orientation information around that point. The method can achieve high accuracy but the computation speed is relatively low if the image is large. Our selected keypoints descriptors are SIFT, ORB [26], and Harris corner detection with OpponentSIFT descriptor [24]. Figure 3-6 illustrates the keypoints descriptor algorithms.

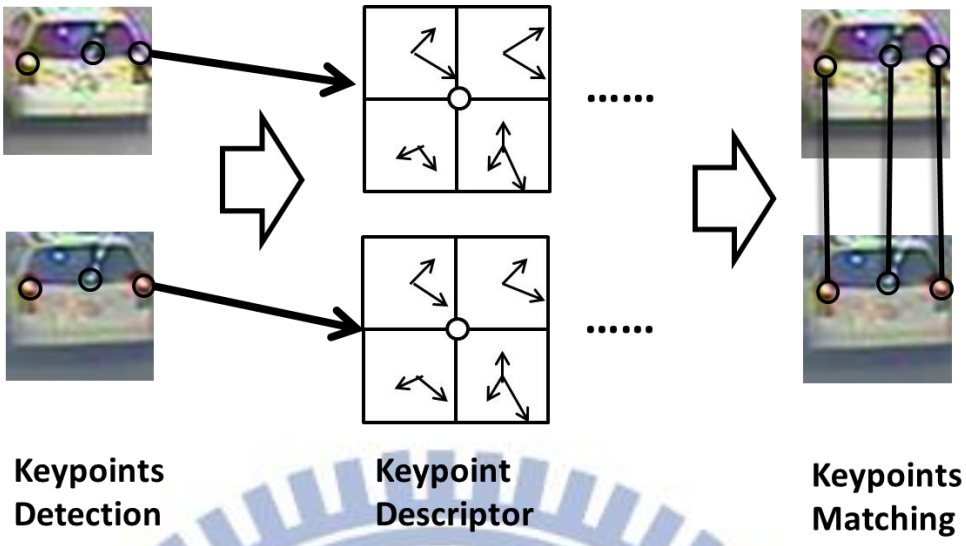


Figure 3-6. A general illustration of keypoints descriptors.

Oriented BRIEF (ORB) [26] tries to reduce the computation time while not reducing much of accuracy compared with SIFT. The algorithm starts by detecting FAST [27] keypoints in an image, which considers the intensity differences between one center pixel and its neighbors in a circular ring. Next ORB uses BRIEF [28] descriptors to encode each detected keypoint into a bit string. The orientation and rotation properties of an image are considered in both keypoint detection and description processes. The whole processing time is much faster than SIFT, since both keypoints detection and description methods are simpler. However, the performance is not as superior as SIFT.

Van de Sande *et al*. [24] evaluate different features for object recognition. They analyze the invariant properties of different color descriptors, histograms, moments, and SIFT descriptors, with test on public datasets. The authors state that OpponentSIFT, which computes SIFT descriptor in opponent color space, is recommended since it outperforms other descriptors and is invariant to light intensity changes and shifts. Although our application is a little bit different from the

experiments in their work, Section 4.2 shows similar result that OpponentSIFT has higher accuracy than other methods described in this section. In their detection process they use Harris-Laplace corner detector to obtain more accurate points without unnecessary points. Since our input image is far smaller and the object is located in the center, we use only Harris corner to reduce computation time.

For all keypoint descriptors, we compute the Euclidean distance between feature vectors of two images I_1 and I_2. First, we extract 10 keypoints, which are enough for our detected images with size of 40*40 pixels, in each image. And the feature vector is obtained using descriptors introduced before. For each point, SIFT descriptor is a vector of integers with 128 dimensions of orientation histogram, where ORB has 256 dimensions and OpponentSIFT has 3*128 dimensions of three channels in opponent color space. Next we use brute-force method to assign a corresponding point in I_2 with smallest Euclidean distance for a point in I_1. Therefore, we can obtain 10 distance values since there are 10 keypoints in an image, and one point in I_2 may have chance being assigned more than once. For I_2, we again use the same method to obtain 10 distance values corresponding to I_1. Finally the distance between the two images is the average value of the 20 distance values. The method can be formulated as

$$d_{KD}(P^{I_1}, P^{I_2}) = \frac{1}{2}\left(\frac{\left(\sum_i \min_j dist\left(P_i^{I_1}, P_j^{I_2}\right)\right)}{|P^{I_1}|} + \frac{\left(\sum_j \min_i dist\left(P_j^{I_2}, P_i^{I_1}\right)\right)}{|P^{I_2}|}\right)$$

where $P_i^{I_1}$ is the ith keypoint descriptor of image I_1 and $P_j^{I_2}$ is the jth keypoint descriptor of image I_2. $P^{I_1} = \{P_1^{I_1}, \dots, P_n^{I_1}\}$ and $P^{I_2} = \{P_1^{I_2}, \dots, P_m^{I_2}\}$ is the set of keypoint descriptors of I_1 and I_2, respectively. The $|\cdot|$ represents the number of elements in a set. The $dist\left(P_i^{I_1}, P_j^{I_2}\right)$ is the distance between $P_i^{I_1}$ and $P_j^{I_2}$. Note that

the Euclidean distance is applied on SIFT and OpponentSIFT descriptors, and Hamming distance is for ORB descriptors.

As our experimental result in Section 4.2, OpponentSIFT outperforms other features and is selected for further processing.

3.4 Multi-Camera Vehicle Groups Matching

This section illustrates the problem of multi-camera vehicle identification and describes the proposed algorithms for tunnel application. We collect vehicles from cameras as described in Section 3.2, and compute the feature vector for each vehicle in Section 3.3. Each camera maintains a queue of detected vehicles with computed feature vectors. Now, we can identify vehicles from different camera by matching feature vectors in two queues. This is called multi-camera identification, object re-identification [22], object matching [23], or multi-camera tracking-by-identification [9].

For each camera, we keep a queue of detected vehicle patches ordered by time. Different cameras are also ordered by time since they are set on road that vehicles must pass in order. If we focus on only two cameras, the identification is reduced to an assignment problem. That is, finding all matched pairs of vehicles from two queues with the smallest sum of distances, and every vehicle can be assigned only once. In Figure 3-7, assume that we have to assign three vehicles from camera C_1 and C_2. First we can compute the distance matrix using feature vectors described before, and then apply greedy method or some well-known solutions such as Hungarian algorithm [29] to find matching pairs of vehicles.

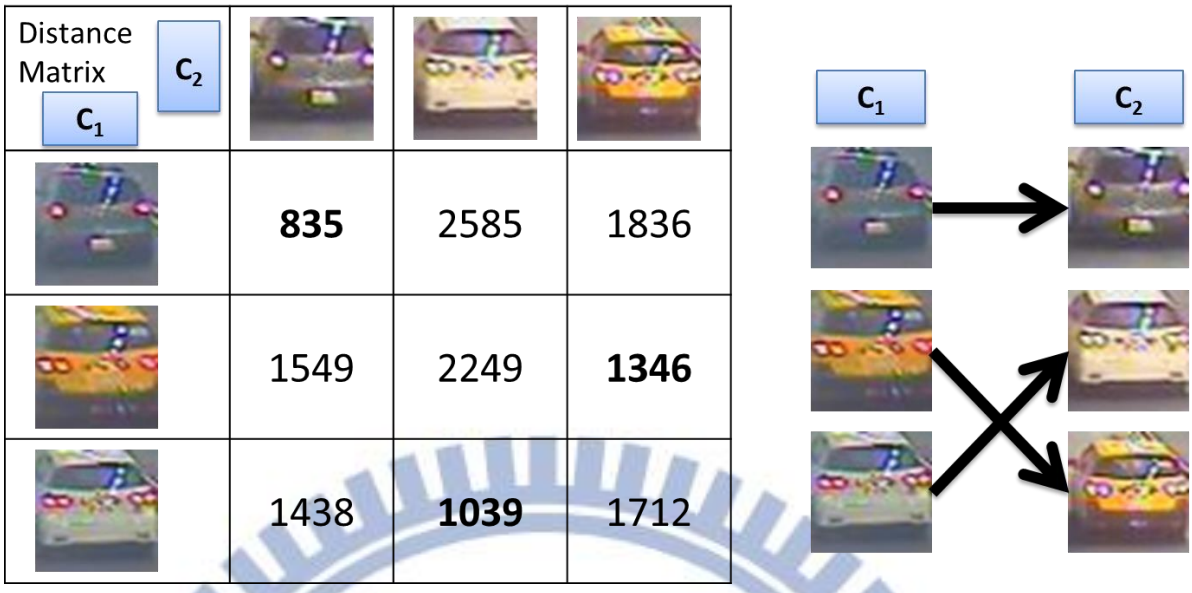


Figure 3-7. Assignment problem in multi-camera vehicle identification.

The major difficulties are the problems in appearance changing of the same vehicle in different cameras, which would be solved by using OpponentSIFT. However, vehicles may look similar or are of the same type that exists in the tunnel within a small period of time. Matching vehicles using only appearance information may have errors sometimes. In other words, matching pairs of vehicles using only the distance matrix obtained by visual features, Figure 3-7 for example, cannot achieve suitable performance, especially when the distance matrix is quite large. Besides, there is a time gap between two cameras, so we can only roughly guess when a vehicle might pass the second camera. This requires a certain number of vehicles to be considered at the same time since we cannot know the exact time a vehicle passed by.

3.4.1 Spatiotemporal Successive Dynamic Programming (S^2DP)

The Spatiotemporal Successive Dynamic Programming (S^2DP) algorithm is proposed to solve the identification problem in tunnels by considering the ordering

constraint. Given a distance matrix of detected vehicles in two cameras, the S^2DP finds one vehicle from the first camera for each vehicle in the second camera.

Ordering constraint [30] exists in road that all vehicles are in some kinds of successive sequence, especially when lane changing is prohibited in long tunnels for traffic safety. More specifically, a car is possibly behind another one during the whole driving in the tunnel. The proposed methods take advantages of the ordering constraint in tunnels. Figure 3-8 illustrates the multi-camera identification problem by considering the ordering constraint. The detection queue of the first camera C_1 is always equal to or larger than the queue of the second camera C_2. The problem is to search one or a small group of vehicles in C_2 from candidates in C_1, and the number of candidates is greater than the number of matched objects. The number of candidates is determined by the actual distance on road between two cameras. More detected cars are stored if two cameras are far away from each other.

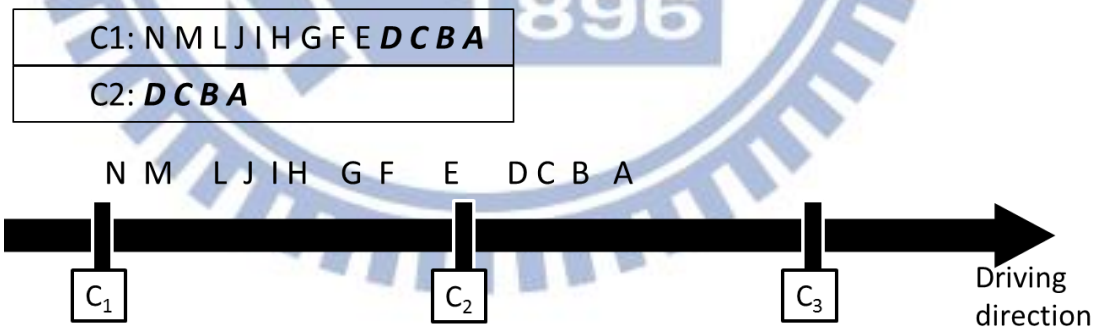


Figure 3-8. An example scenario on multi-camera vehicle identification. Assume each capital letter represents an individual vehicle.

In the beginning, we are not sure where the exact position of the desired candidate is in the queue, so we need to search in a window of proper size and position, with respect to the detection queue of previous camera. The size of the

window is important since the more the candidates are included, the more the noises are contained, which affects the performance. Therefore, the first thing to do is to find the proper location of the search window in the queue of first camera, and then we will have the ability to minimize the size of the window. We call this step as an initialization problem. As illustrated in Figure 3-8, the first vehicle A need to be correctly identified, thus vehicle B can be easily found by considering the ordering constraint. If, unfortunately, vehicle A in camera C_2 is matched to vehicle H in camera C_1, then vehicle B in C_2 can never find a correct matching since B is not behind H. Therefore, the initialization problem is an important issue when the ordering of objects is taken into account.

It is worth noting that not all vehicles in the second camera can find a corresponding detection in the first camera, and vice versa. Because the vehicle detection cannot certainly reach 100% of accuracy, there are always miss detections in all cameras. In addition, when each camera passes information to the central server using the Internet, there may have packet losses or disconnections sometimes that implicitly yield miss detections. Consequently the algorithm must incorporate the miss detection problem.

Since the ordering constraint is an important characteristic and the miss detection problem needs to be considered, we propose a dynamic programming algorithm S^2DP to solve the assignment problem. Assume that we have a $N_1^C \times N_2^C$ distance matrix D where the two axes contain N_1^C and N_2^C vehicles in camera C_1 and C_2, respectively. $D(i,j) = d_{KD}(P^{I_i}, P^{I_j})$ is the distance between two feature vectors of vehicle $1 \le i \le N_1^C$ in C_1 and vehicle $1 \le j \le N_2^C$ in C_2 using keypoints descriptor described in Section 3.3. The dynamic programming cost function f is

defined as

$$f(i,j) = min\begin{cases} f(i-1,j-1) + D(i,j) + \lambda * step \\ f(i-1,j) + \epsilon \\ f(i,j-1) + \epsilon \end{cases}$$

where ϵ is the miss-match penalty, λ is the Non-Spatiotemporal-Successiveness (NS^2) penalty, and $step$ is the number of steps from the last assignment. $D(i,j)$ is added to f if i is assigned to j, which is the first statement of the min function. Otherwise, a penalty ϵ is added to f for miss detection. After scanning the distance matrix D once using the cost function f, a corresponding cost matrix and a backtrack table can be obtained. The assignment results can be found by tracing the shortest path from $f(N_1^C, N_2^C)$ to $f(0,0)$ using the backtrack table.

Figure 3-9 shows an example of the S^2DP algorithm. In the dynamic programming backtrack table, we can find a minimum-cost path from the last element, which corresponds to row F and column C in Figure 3-9, to $f(0,0)$ easily. The horizontal and vertical paths represent miss detections and diagonal paths are matched vehicle pairs. The algorithm is similar to the longest common subsequence problem.

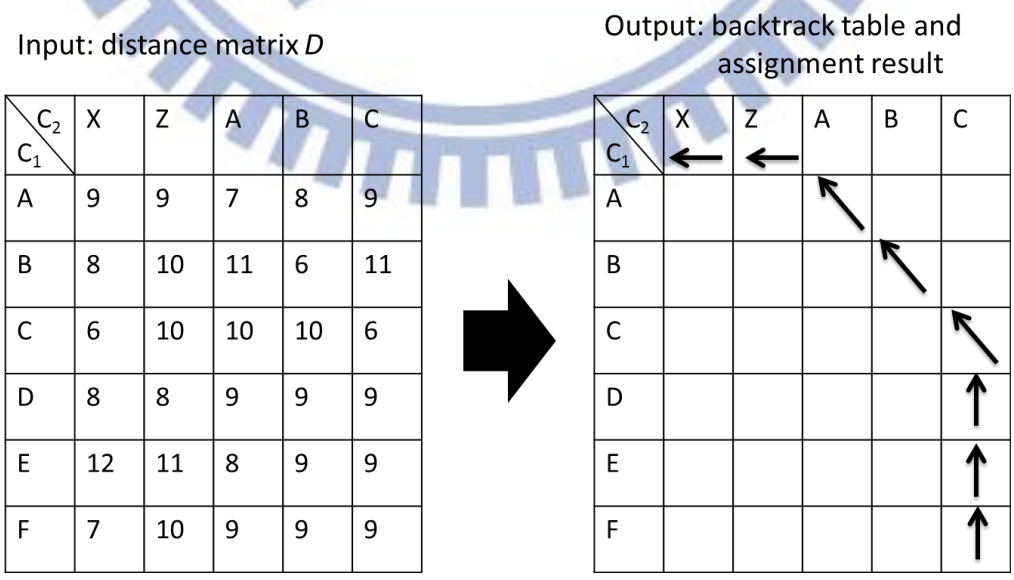


Figure 3-9. An illustration of S^2DP algorithm

The value of the miss-match penalty ϵ is determined by both the distance function d_{KD} and the proper threshold to claim miss detections. The total cost f will add ϵ twice if one assignment is skipped since one element in row and one in column have to be added to move to the same position. As illustrated in Figure 3-10, two ϵ or one assignment cost from location A to location B will be added to the cost function f. Therefore, we set ϵ as half of the maximum-allowed matching distance (threshold). We use a fixed value for all datasets in our experiments, and the discussion of this parameter will be presented in Section 4.5 with recommendations for default value.

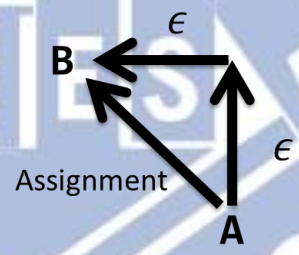


Figure 3-10. Decision of miss-match penalty ϵ. From location A to B in the cost function F, the total cost will add two ϵ or one assignment cost.

Another parameter is the Non-Spatiotemporal-Successiveness (NS^2) penalty λ for successive assignments. Assuming that we do not have enough detections in the camera C_2, Figure 3-11 shows a scenario when λ equals zero. This scenario occurs when two cameras are far away from each other in the initialization step. The total cost between the solid path and the dotted path are nearly the same, thus we may have chance to choose the dotted one, which is wrong, instead of the solid one when noise is included. If the number of elements in one axis is much less than the other, then the assignment may not be successive since the distance cannot provide enough information due to noise of visual features in the distance matrix. Here, the successive assignment means that assigned elements are next to each other, which satisfies the

ordering constraint that vehicles go through the tunnel in order and cannot disappear. Therefore, we select the path that assignments are more successive with the consideration of ordering constraint. For each assignment, we add the NS^2 penalty λ to the distance, multiplied by the number of steps from previous assignment. If the $f(i,j)$ is far away from previous assignment, then the penalty is big enough to make miss match as final decision. We define $\lambda = \alpha\epsilon$ that is related to ϵ and α is the weight. In our experiments $\alpha = 0.01$ is enough to accomplish the task.

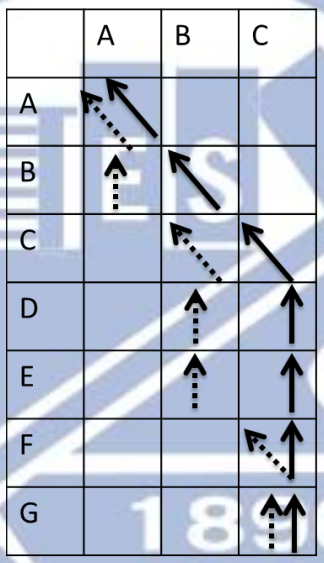


Figure 3-11. A scenario without Non-Spatiotemporal-Successiveness (NS^2) penalty λ (dotted path).

3.5 Real-Time and Offline Vehicle Identification

This section describes our real-time and offline identification process. The first step is S^2DP presented in Section 3.4. However, two problems show up in the S^2DP: cannot run in real-time, and cannot assign order-changed vehicles. We employ two additional algorithms to solve the two problems after the S^2DP.

First, the S^2DP cannot run in real-time since it requires a number of vehicles appear in both camera queues. For example, at least 15 vehicles are required to achieve reasonable accuracy in our experimental result in Section 4.2. Sometimes we would like to identify a vehicle immediately when it appears in the second camera, just like multi-camera vehicle tracking. If the initialization problem is solved by S^2DP, we can lower the size of the search window and identify the corresponding vehicle with a small number of candidates in the queue.

Another problem is that, if one vehicle changed its order in two cameras, the S^2DP would fail to assign this vehicle. Figure 3-12 shows an example of the situation. Vehicle *C* is in front of vehicle *B* in the second camera and *B* cannot find proper assignment because the ordering constraint does not allow *B* in C_2 to assign the one prior to *C* in C_1. Hence *B* in C_2 will get a "*no-match*" result. The problem will occur in two situations: one vehicle overtakes another one, or multiple driving lanes are in the video and the driving speed of one lane is faster than that of another one. Although in many tunnels overtaking of vehicles is not allowed, there are still vehicles not obeying the traffic rules. And in our dataset, we do not record which driving lane a detected vehicle was to make our work more flexible. Hence the problem is considered in this section. In fact the methods automatically detecting driving lanes in a surveillance video [14] [7] can be applied to our system for better vehicle detection.

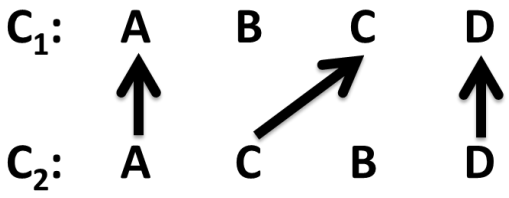


Figure 3-12. The order of vehicle B and vehicle C in camera C_1 are changed in another camera C_2. S^2DP algorithm can only assign one of them.

The remaining sections are organized as follows: in Section 3.5.1 we will describe the Real-Time (RT) algorithm for fast assignment of a proper candidates using sliding window approach. If real-time processing is not required, another Offline Refinement (OR) algorithm to solve the problem with higher identification accuracy is presented in Section 3.5.2.

3.5.1 Real-Time Identification

Assume we finish the S^2DP algorithm that all vehicles in the second camera are assigned. Then, if a new vehicle shows up in the second camera, we can assign a candidate to this vehicle in real time. Instead of considering a group of vehicles from two cameras in S^2DP, the Real-Time (RT) algorithm only takes one vehicle in the latter camera for identification. RT does not have to wait for a number of vehicles passing by, thus achieve real-time processing. This section describes a simple greedy algorithm that achieves real-time processing of the identification problem.

Given the computation result of the S^2DP and a feature vector of one newly detected vehicle in the second camera, the RT assigns one candidate vehicle to the newly detected one. As we finish the initialization step using S^2DP, a newly detected

vehicle in the second camera is possibly next to the last assigned one in the first camera. This property suggests us of using a sliding window approach to solve the problem.

The Real-Time (RT) algorithm runs as follows. First set a search window in the detection queue of the first camera just around the last assigned vehicle in the S^2DP, since a newly detected vehicle in latter camera is probably behind it according to the ordering constraint. Considering the example in Figure 3.8, the S^2DP matches vehicle A to vehicle D. As vehicle E is detected in camera C_2, the desired matching candidate in C_1 is probably around vehicle D. The size of the window is limited to a small value for real-time processing, for example, five vehicles in our experiments. Notice that the detection queue of the first camera always contains unassigned vehicles since it is prior than the second camera. If a vehicle appears in the second camera, we calculate the feature vector of the detection and compute the distance to all vehicles in the search window of the first camera. The one in the first camera with the smallest distance is picked up and treated as the assignment result. If all distance values are greater than the matching threshold, which is greater than 2ϵ in S^2DP, a no-match is assigned. Finally we slide the search window to the next one in the detection queue of the first camera, and this process loops again. Figure 3-13 describes the RT algorithm in detail. As an execution example shown in Figure 3-14, only five candidates in C_1 are considered when vehicle E is detected in camera C_2. The candidate with smallest distance is matched. Next five candidates are changed using sliding window approach for the next detected vehicle F in C_2. Note that only the candidates in the search window are taken into consideration. The distance values of others, for example, candidate J, are not computed and are represented using dots.

- Assume N^W candidates in the search window of the first camera
- For each newly detected vehicle in the second camera
  - Calculates the distances to N^W candidates
  - Picks up the candidate with the smallest distance value
  - Outputs the selected one if the distance is less than threshold
    - Otherwise assign a *no-match*
  - Shifts the search window in the first camera

Figure 3-13. The Real-Time (RT) algorithm.

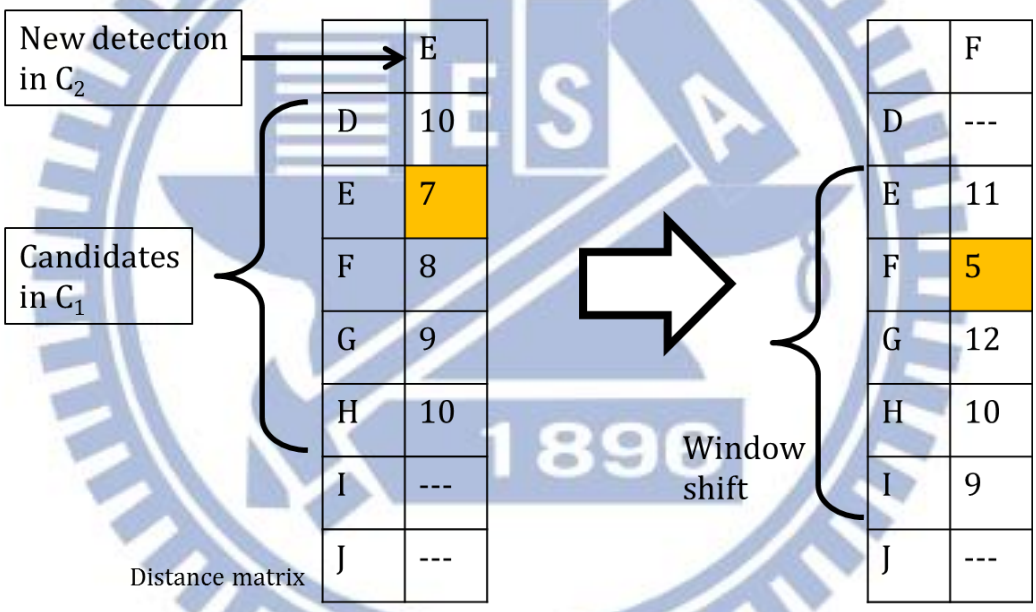


Figure 3-14. An example of RT. Each capital letter represents one vehicle, and the table is the distance matrix. Only the distance values of candidates are computed and presented using numbers.

The RT algorithm allows one candidate in the first camera being matched to two or more new detections in the second camera. It is strange and erroneous that one vehicle became two in another camera. However in our experiments, we found that error may propagate if we do not enforce the property and produces poor identification results. Consider the scenario in Figure 3-15 that vehicle X in the first

camera C_1 is assigned to vehicle W in the second camera C_2, which is an incorrect assignment. The algorithm would delete X from the candidate queue if we do not allow re-assignment. Next vehicle X in C_2 was detected, however X in the candidate queue had already been erased and the assignment would certainly incorrect. If X in C_2 still got another false assignment, then this error propagated. Therefore an incorrect assignment will possibly produce at least two errors in the final result.

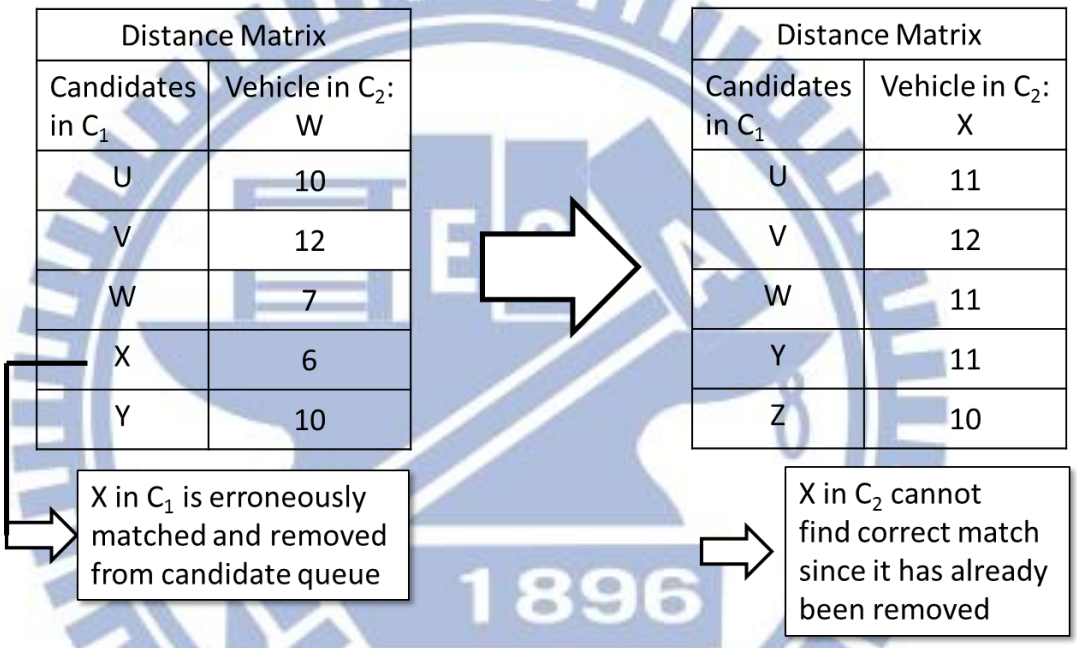


Figure 3-15. An example of error propagation if popping-candidate is allowed. In the beginning vehicle W is detected in C_2 and erroneous matched vehicle X in C_1. Next vehicle X is detected in C_2, but X has already been removed from candidates thus cannot obtain correct result.

Another characteristic is that the last assignment in the first camera has to be a correct assignment. Otherwise the initial position of the sliding window would be inappropriate and the correct candidate would never appear in the search window. Hence the whole real-time identification results would be wrong. To make sure the algorithm can correctly find the location of the sliding window, we discard the last three results in the S^2DP and re-assign them in the RT. Using the S^2DP algorithm can

make all assigned candidates close to each other by considering the NS^2 penalty. However, the vehicles in the end of the matched group cannot be matched correctly since the last few vehicles of the group have too little information to be matched. Consequently, we discard the last three assignments and re-assign them in the RT.

Finally, to enhance the effect of ordering constraint $\lambda * step$ is added to all distance values, where λ is the NS^2 penalty and $step$ is number of steps from previous assignment.

3.5.2 Offline Refinement

The RT algorithm can run in real-time, but the performance is limited because it only considers one vehicle for one assignment. As stated in the S^2DP, the ordering constraint is an important feature and we should consider a group of vehicles instead of one. Therefore, we develop another algorithm Offline Refinement (OR) achieve higher accuracy. Another problem is that the S^2DP algorithm cannot properly assign order-changed vehicles.

We propose the Offline Refinement (OR) algorithm to solve these problems. We found that order-changed vehicles can be solved using a second pass assignment. Figure 3-16 shows an example that vehicle B and E are order-changed vehicles and cannot be assigned by S^2DP. As vehicle B and vehicle E are both missed in camera C_1 and C_2, we can re-assign the missed vehicles. In fact all order-changed vehicles will possibly leave un-assigned in S^2DP. We can simply apply Hungarian algorithm to the vehicles that are missed in the S^2DP. That is, vehicle B and vehicle E in Figure 3-16 are matched using optimal assignment algorithm.

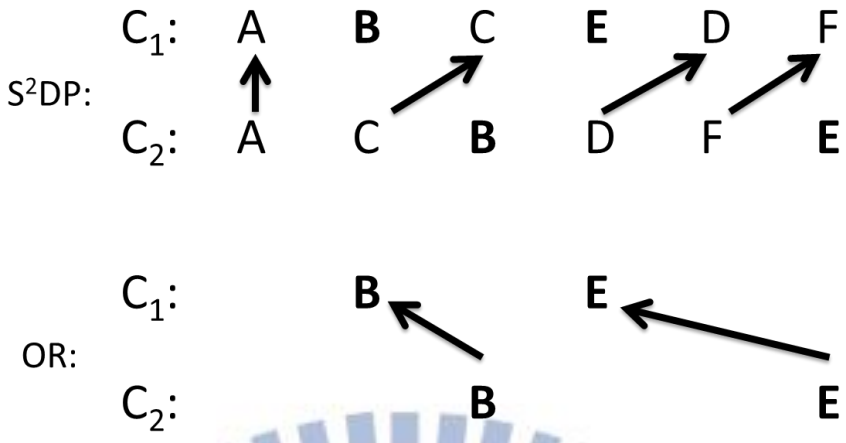


Figure 3-16. Example of the OR. The order of vehicle B and E changed, the S^2DP algorithm can only assign one of them. The OR can re-assign B and E.

The penalty $\lambda * step$ is added to each element in the distance matrix, where λ is NS^2 penalty, and $step$ is the number of detections between two vehicles, just like in S^2DP and RT. Similarly, after applying Hungarian algorithm in the OR, we will check the distance values of each assignment, and report miss detections if the value is greater than threshold 2ϵ, the same threshold value as in S^2DP.

Chapter 4. Experiments

This chapter presents the experiments on our proposed methods. First, we will introduce the manually labeled datasets for experiments in Section 4.1. Experiments on feature selection, the S^2DP multi-camera identification, and real-time and offline methods are presented in Section 4.2, 4.3 and 4.4, respectively. Finally the discussions on the threshold ϵ in our algorithm and the performance on other datasets are discussed in Section 4.5.

4.1 Datasets

The surveillance videos from five cameras in Hsuehshan tunnel, Taiwan [31] are collected. Every vehicle drives through five cameras from camera C_1 to camera C_5 and the driving distance between each camera is one kilometer in average. We manually label 195, 195, 148, 150, 173 vehicles in C_1 to C_5, respectively, and each labeled vehicle is a colored image of size 40*40. Assume that all vehicles are presented in C_1 and C_2, and C_3, C_4, C_5 including miss detections. The detection rate in each camera is 100%, 100%, 76%, 77%, and 89%, respectively. The numbers of order-changed vehicles in C_1 to C_5 with respect to C_1 is 0, 6, 10, 12, and 19. Here we call this dataset as HsuehShanTunnel (HSTunnel).

Another dataset HSTunnel_NO_MISS is a subset of HSTunnel. All five cameras contain 124 vehicles respectively, which is the intersection of all cameras in HSTunnel. Assume there is no miss detection in this dataset. The numbers of order-changed vehicles with respect to C_1 are 0, 0, 5, 4, and 8 in C_1 to C_5, respectively. HSTunnel_NO_MISS can give us the best performance of our experiments since one major issue, miss detection, is removed. Figure 4-1 shows the examples of each

camera in HSTunnel, and the resolution is 352*240. Table 4-1 summarizes the properties of the two datasets, and sample images are presented in Figure 4-2. The X symbol represents miss detection in Figure 4-2.

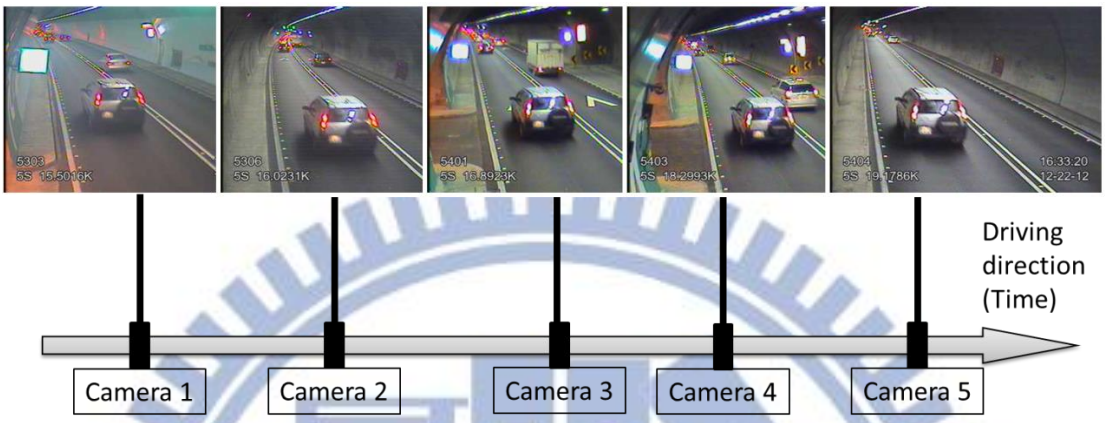


Figure 4-1. Five surveillance videos in Hsuehshan tunnel, Taiwan.

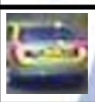
	HSTunnel					HSTunnel_NO_MISS				
ID	C_1	C_2	C_3	C_4	C_5	C_1	C_2	C_3	C_4	C_5
001										
002										
003										
004										
005			X	X		X	X	X	X	X
006										

Figure 4-2. Examples of HSTunnel and HSTunnel_NO_MISS. The X symbol represents miss detection. If one vehicle is miss-detected in one camera, it is removed from HSTunnel_NO_MISS, for example, vehicle 005.

Table 4-1. Properties of dataset HSTunnel.

Dataset	Camera	#Vehicles	Detection Rate	#order-changed w.r.t. C_1
HSTunnel	C_1	195	100%	0
	C_2	195	100%	6
	C_3	148	76%	10
	C_4	150	77%	12
	C_5	173	89%	19
HSTunnel _NO _MISS	C_1	124	100%	0
	C_2	124	100%	0
	C_3	124	100%	5
	C_4	124	100%	4
	C_5	124	100%	8

It is worth noting that this work is focused on multi-camera identification. We assume that the vehicle detection and tracking processes have already executed. Therefore, the datasets in our experiments contain only manually detected vehicle images from different cameras, instead of raw videos. Experiments on single-camera vehicle detection and tracking can be found in [6] and [9], which use similar methods as in our system.

4.2 Feature Selection

To choose a proper feature descriptor for multi-camera vehicle identification, HSTunnel_NO_MISS dataset is selected to evaluate the distinctiveness of each feature described in Section 3.3. All vehicles in the dataset exist in all cameras. That is, no miss detection occurs in this dataset. First, we choose two out of five cameras in HSTunnel_NO_MISS dataset and calculate the 124*124 visual distance matrix. Next, for each row of the distance matrix, we can obtain the rank of the vehicle (row). Rank

i means the corresponding vehicle in the next camera (column) is the ith smallest distance value in the row. For example, *rank 1* means the corresponding vehicle in the next camera has the smallest distance value with respect to all others in the row, which is the best result. *Rank 2* means the correct vehicle has the second smallest distance value, and so on. Therefore, the HSTunnel_NO_MISS contains five cameras and each feature can obtain ten execution results by exhaustively choosing two cameras out.

Same as the experiments in [23], we obtain the cumulative matching characteristic (CMC) curve. Assume that we have N vehicles presented as $V = \{V_1, \dots, V_N\}$, and the $rank(V_i)$ means the rank value of vehicle V_i where $1 \leq rank(V_i) \leq N$. The $CMC(r)$ of rank value r can be defined as:

$$CMC(r) = \frac{\#\{V_i | rank(V_i) \leq r\}}{N}$$

where $\#\{\cdot\}$ denotes the number of elements satisfying the condition. For $1 \leq r \leq N$, we can obtain the CMC curve.

For the Haar-like feature [11] used in feature extraction, we manually collect 900 grayscale images of vehicles from eight tunnel videos as positive set, which is different from dataset HSTunnel, and 1800 negative samples randomly sampled from 20 background images inside tunnels without vehicles. The resolution of training images is 24*24, where the resolution of our video is 352*240 pixels. The number of cascading stages is 10 and the total number of weak classifiers obtained from the training algorithm is 143. The trained detector is only used in the experiments of feature selection.

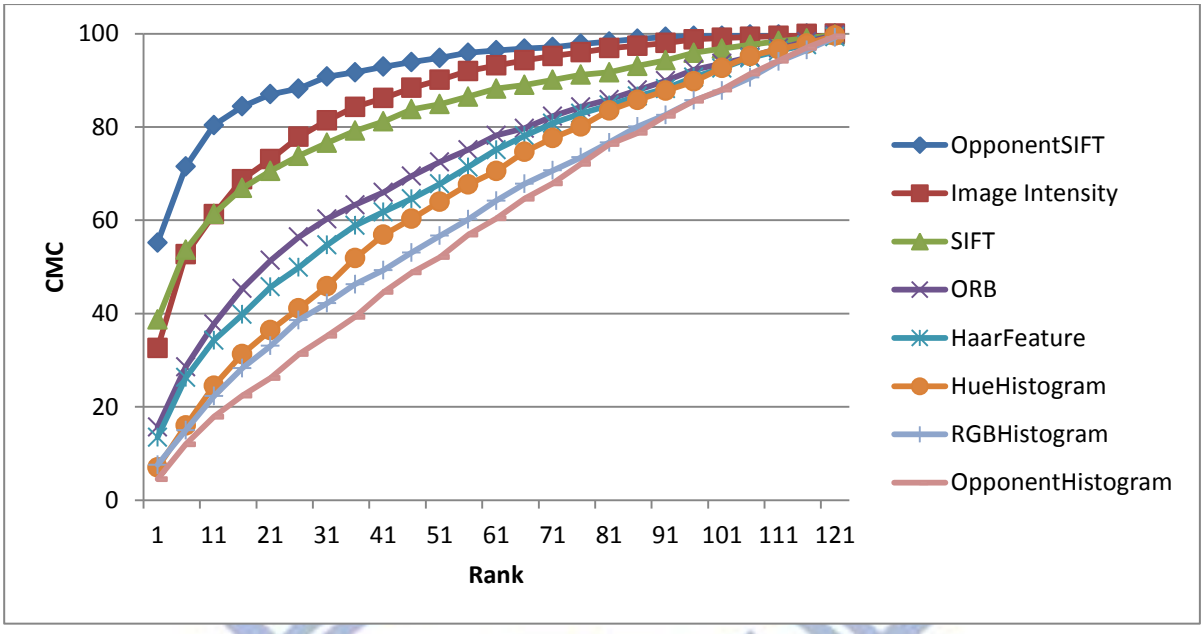


Figure 4-3. Average performance on different feature descriptors.

Table 4-2. Average performance on CMC values of different feature descriptors.

Feature Descriptor	CMC(1)	CMC(3)	CMC(5)	CMC(10)
Image Intensity	32	44	50	60
Haar-Features	14	20	24	33
RGB Histogram	8	11	14	21
Hue Histogram	7	13	14	23
Opponent Histogram	5	10	11	17
SIFT	39	50	51	60
ORB	16	25	27	36
OpponentSIFT	**55**	**67**	**70**	**79**

Figure 4-3 depicts the average performances over the CMC value and different rank values, and Table 4-2 shows the CMC values with some rank value. The selected features are image intensity, Haar-like feature vector, color histograms, and keypoints descriptors. For color histograms, RGB, hue, and opponent color space are used in the

experiments. And for keypoints descriptors, SIFT, ORB, and OpponentSIFT are tested. We can clearly observe that OpponentSIFT outperforms other feature descriptors in Figure 4-3. The result is similar to the recommendation in [24]. In conclusion, we choose OpponentSIFT as our feature descriptor.

For tunnel surveillance, OpponentSIFT outperforms SIFT since OpponentSIFT considers color information but SIFT does not. As described in the previous section, the color and structure information of a vehicle are powerful visual information that should be considered.

All color histogram descriptors perform poor in the experiments. Color is a strong feature in vehicles. However, many vehicles are mostly in the same color, especially in our experiments that we look at 124 vehicles at the same time. Therefore, using only color information is not sufficient for multi-camera identification.

Finally, the Haar-feature vector used in [9] cannot achieve good result in CMC value of the experiment. It is used in vehicle classifier, which means all vehicles share the similar characteristics. The Haar-feature vector we trained contains only 148 dimensions with binary values, and it is not sufficient to describe all vehicles since most of the values are the same.

4.3 Multi-Camera Vehicle Groups Matching

This section presents the experiments on the multi-camera vehicle groups matching algorithms. By using the OpponentSIFT as feature descriptor, the assignment algorithms can be applied to match vehicles from two cameras.

Assume that we have N_1^C vehicles in the first camera and N_2^C vehicles in the second camera and $N_2^C \leq N_1^C$. In the following experiments, we set N_1^C as 50 and N_2^C from 1 to 50. Considering the order constraint in tunnels, the larger N_2^C is, the higher accuracy can be achieved. For each vehicle in the second camera, a vehicle in the first camera is assigned to it or is assigned as "*no-match*" if all candidates are not suitable. The accuracy is the percentage of correct assignments over all N_2^C vehicles in the second camera.

Figure 4-4 shows an example of assignment result. The first two rows are the detected ID of vehicles in capital letters, and the third row is the output from some algorithm. For every vehicle in C_2, the algorithm chooses one best-match vehicle from C_1. The first four results are correct. The 3rd result is correct because vehicle C is not in camera C_1 and the algorithm assigns a *no-match* to it. Finally, the last two results are examples of incorrect assignment: vehicle F in C_2 matches vehicle E, and vehicle G in C_2 is claimed as *no-match* in C_1 but vehicle G does exist. Therefore, the accuracy of this example is $4/6 = 67\%$. The experiments on these two cameras are represented as (C_1, C_2), where C_1 is the first camera and C_2 is the second camera.

C_1	A,	B,	D,	E,	F,	G
C_2	A,	B,	C,	D,	F,	G
Result	A,	B,	*No-match,*	D,	E,	*No-match*

Figure 4-4. Example of an assignment result on (C_1, C_2). The first row is the candidate queue contains detections in camera C_1, the second row in the second camera C_2, and the third row is an example of execution result.

The experiments proceed as follows. First we randomly set a number as the starting index in two cameras, the following N_1^C vehicles in the first camera and N_2^C vehicles in the second camera are used in the experiment. It is necessary because the order of vehicles in our dataset cannot change, and the number of vehicles is greater than N_1^C. We randomly select 15 starting index and run 15 times independently, and take the average accuracy as the final result. After that we increase the value of N_2^C by one, and randomly execute 15 times again. Finally N_2^C is tested from 5 to 50, and N_2^C is incremented by one after each iteration.

Three methods are evaluated in the following experiments: S^2DP, S^2DP without NS^2 penalty, and Hungarian algorithm. To evaluate the effectiveness of NS^2 penalty, we include the result that NS^2 penalty λ is set to zero. As a common solution of assignment problems, the Hungarian algorithm is selected as the baseline for performance comparison. The miss-match penalty ϵ in S^2DP is set to 450 in all experiments, where discussions on the value of ϵ are presented in Section 4.5.1. Both HSTunnel and HSTunnel_NO_MISS are tested in the experiments.

Figure 4-5 and Figure 4-6 show the results on every camera setting in HSTunnel and HSTunnel_NO_MISS, respectively. The x-axis is the value of N_2^C and the y-axis is the corresponding accuracy. In HSTunnel, all methods perform poor in camera (C_3, C_5) and (C_4, C_5) when N_2^C is small. Table 1 shows that camera C_5 contains 173 vehicles, whereas camera C_3 contains only 148 vehicles. In other words, in total 25 vehicles in the second camera (C_5) cannot find corresponding candidates in the first camera (C_3) because of miss detections. Therefore, the performance on (C_3, C_5) may decrease if the number of vehicles in second camera is not enough, same as in (C_4, C_5). Camera (C_2, C_3) does not have such problem even there are 47 vehicles miss in C_3

since miss-detected vehicles in C_3 are not candidates. The algorithms do not execute on miss-detected vehicles in the second camera, as illustrates in Figure 4-4, thus the performance on (C_2, C_3) does not have significant differences compared with (C_3, C_5). Figure 4-6 on HSTunnel_NO_MISS does not have this effect since there is no miss detection in HSTunnel_NO_MISS.

Figure 4-7 shows the results on HSTunnel and HSTunnel_NO_MISS. The x-axis is the value of N_2^C and the y-axis is the accuracy. All methods achieve higher accuracy in HSTunnel_NO_MISS than HSTunnel since there is no miss detection in HSTunnel_NO_MISS. The Hungarian algorithm does not work well because the visual feature is not robust enough to provide sufficient information. With NS^2 penalty, the S^2DP algorithm can achieve higher accuracy when N_2^C is below 45 in HSTunnel_NO_MISS and 39 in HSTunnel, respectively, and achieves similar performance when N_2^C value is near N_1^C. The S^2DP algorithm can reach 90% and 80% accuracy when N_2^C is greater than 6 and 30 in HSTunnel, respectively. Note that in HSTunnel the performance of S^2DP drops when N_2^C is greater than 45, because more order-changed vehicles are included. As described in Section 3.5, the S^2DP cannot correctly identify order-changed vehicles.

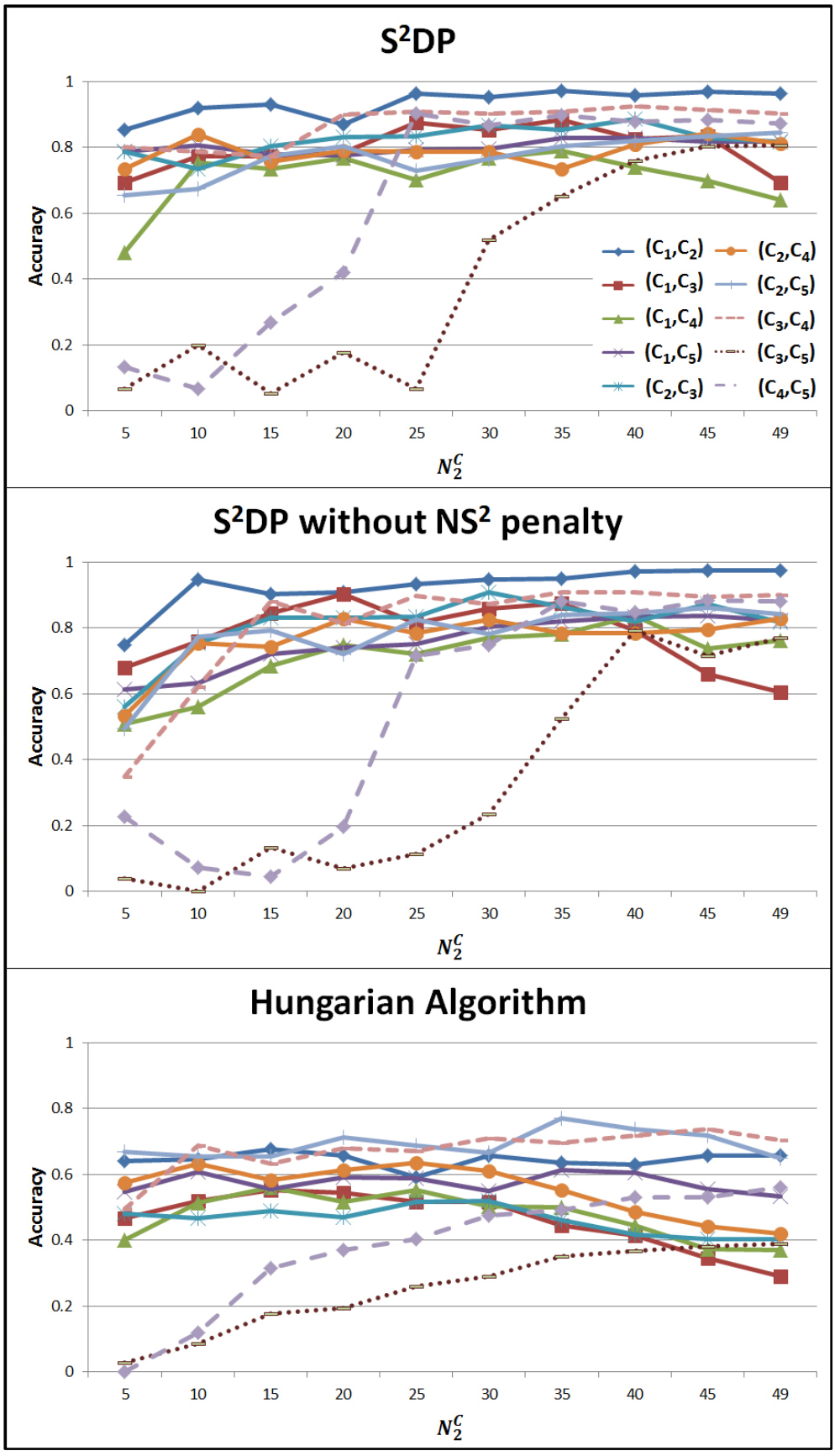


Figure 4-5. Experimental result of vehicle groups matching algorithms on HSTunnel. The x-axis is the number of vehicles in C_2 assigned, and y-axis is the accuracy.

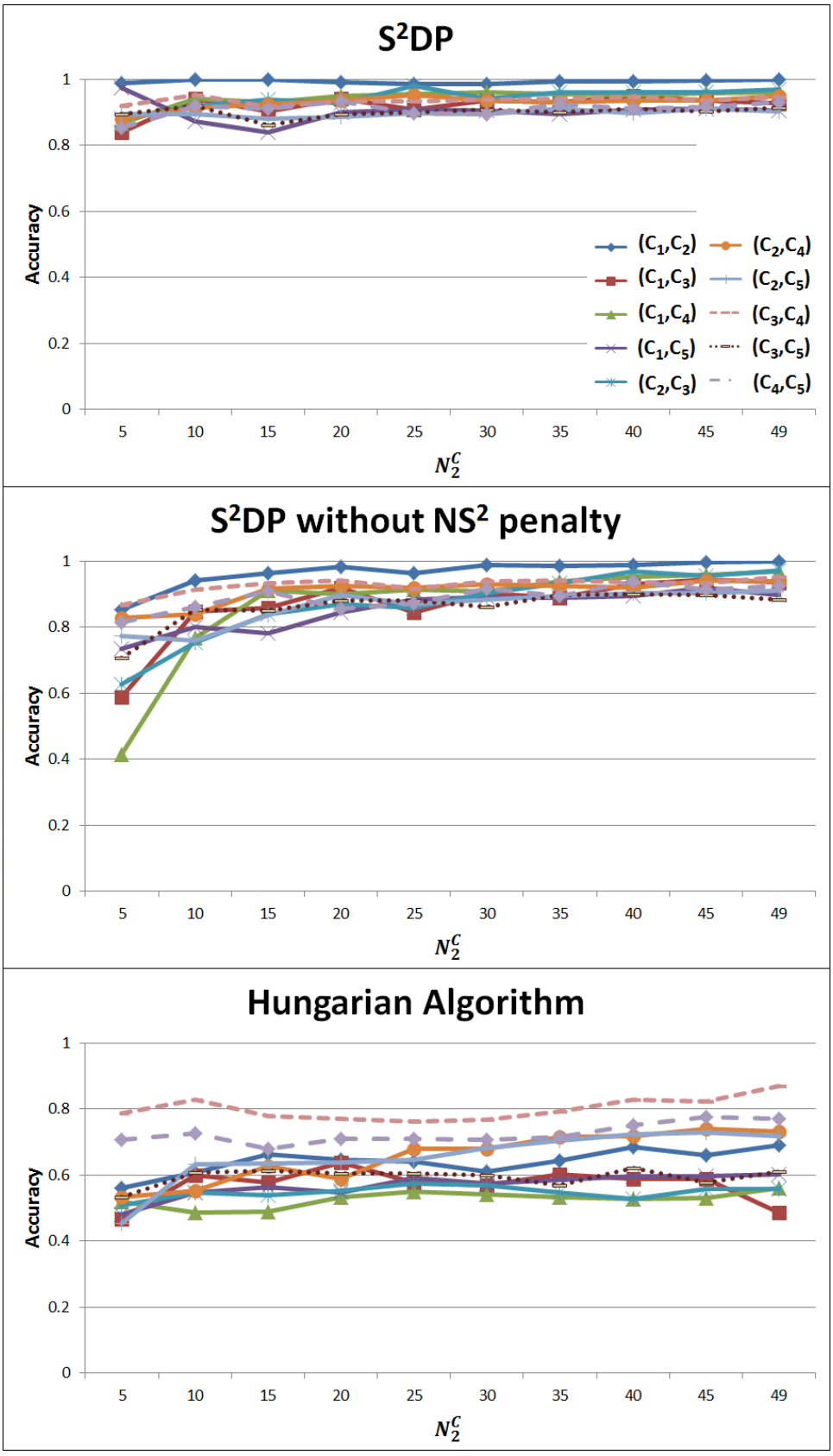


Figure 4-6. Experimental result of vehicle groups matching algorithms on HSTunnel_NO_MISS. The x-axis is the number of vehicles in C_2 assigned, and y-axis is the accuracy.

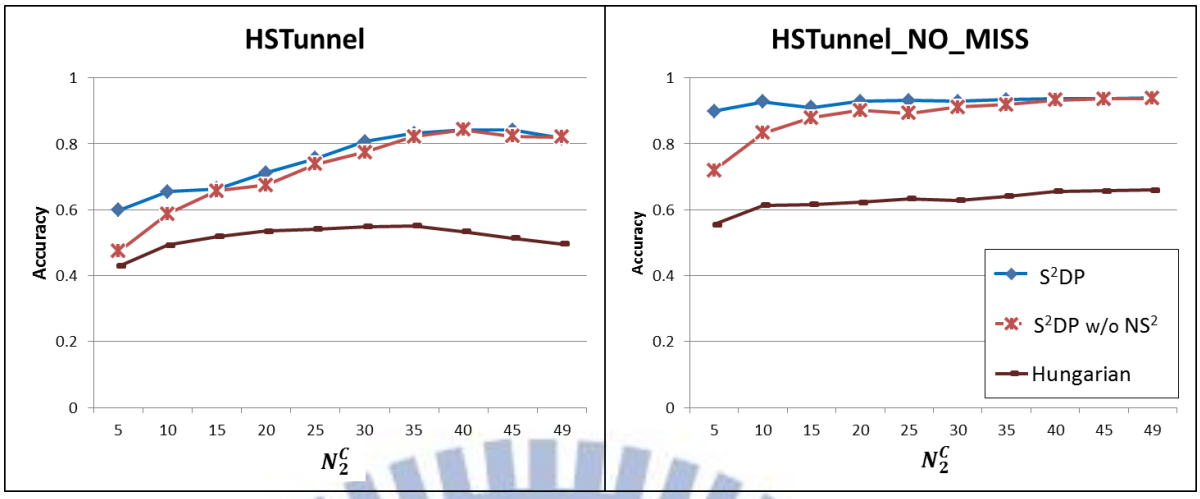


Figure 4-7. Average accuracy of multi-camera vehicle groups matching algorithms.

4.4 Real-Time and Offline Vehicle Identification

We evaluate the proposed real-time and offline identification algorithms in this section. In the first step we collect a set of vehicles in the second camera and run the S^{2}DP algorithm to solve the initialization problem. Next we can apply the proposed real-time assignment or offline refinement algorithms.

4.4.1 Real-Time Identification

Similar to the experiments in Section 4.3, we randomly select one starting point and apply the real-time RT algorithm, as described in Section 3.5.1. The final result is the average accuracy of the 15 rounds of execution with random starting points.

Table 4-3 summarizes the experimental settings and Figure 4.8 illustrates an example of experiments on HSTunnel. In Figure 4.8, the solid lines represent the number of vehicles used in the S^{2}DP, and the dotted lines for the RT. Assume that

camera C_1 starts at vehicle index *i* and C_2 at *j*, where each vehicle is given an index numbered from 0 to 194 in HSTunnel (see Table 4-1). The S^2DP algorithm identifies j^{th} to $(j+29)^{th}$ vehicles in C_2, and the RT identifies $(j+27)^{th}$ to $(j+59)^{th}$ vehicles. The $(j+27)^{th}$ to $(j+29)^{th}$ are re-identified in RT. Finally, in total 60 vehicles are identified.

Table 4-3. Experimental settings of the real-time methods.

Dataset	S^2DP		RT		Total matched
	#vehicle in C_1	#vehicle in C_2	#vehicle in C_1	#vehicle in C_2	
HSTunnel	50	30	35	33	60
HSTunnel _NO_MISS	50	20	30	28	45

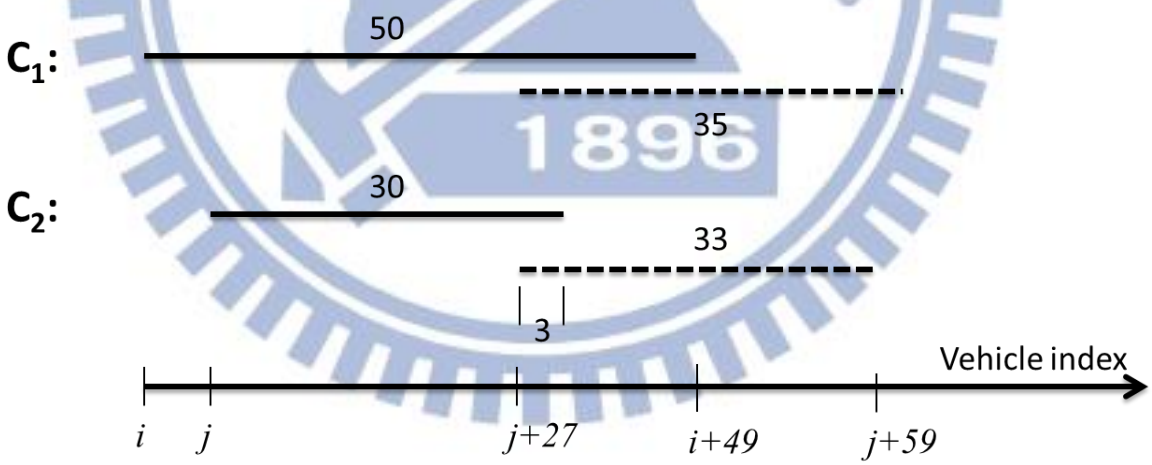


Figure 4-8. Example of real-time experiments on HSTunnel. The solid lines represent number of vehicles in the S^2DP and dotted lines for the RT. Assume camera C_1 starts at vehicle index *i* and C_2 at *j*.

We use different settings on HSTunnel and HSTunnel_NO_MISS in the experiments, since the performances of S^2DP are different in the two dataset in

Section 4.3. For HSTunnel, the S^2DP will assign 30 vehicles in the second camera from 50 candidates in the first camera, where the accuracy reached 80% in the experiments as described in Section 4.3. Next, in the RT, we assign the 28^{th} to 60^{th} vehicles in the second camera from 35 candidates. Note that our RT algorithm re-assigns the last three vehicles in the S^2DP, and therefore the S^2DP assigns 30 vehicles and the RT starts on the 28^{th} one. For candidates in the first camera, the RT starts with the one that is assigned to the 28^{th} vehicle of the second camera, and the following 35 candidates are searched in the RT. Finally, 60 vehicles are assigned in both S^2DP and RT, and all of the 60 vehicles are taken into consideration when computing the accuracy. For HSTunnel_NO_MISS, the S^2DP assigns 20 vehicles from 50 candidates, and the RT assigns 18^{th} to 45^{th} vehicles from 30 candidates. The accuracy considers for all 45 vehicles. For HSTunnel_NO_MISS, the S^2DP assigns 20 vehicles from 50 candidates, and the RT assigns 18^{th} to 45^{th} vehicles from 30 candidates.

To demonstrate the effect on different properties of RT algorithm, some parts are removed from RT in the experiments. Three properties in RT are: NS^2 penalty λ, re-assignment of the last three results from S^2DP, and multiple assignments on one candidate. Therefore, we introduce three variants of RT algorithm. The first one is RT-w1 which sets NS^2 penalty λ to zero so the effect on this penalty is discarded. Next the RT-w2 further discards the multiple-assignments property from RT-w2 and one candidate can be assigned only once. Finally, the RT-w3 discards all the three properties.

Table 4-4 and Table 4-5 show the performances of different methods on HSTunnel and HSTunnel_NO_MISS, respectively. Each value in the table is average

accuracy of vehicle identification of two cameras (column) using different method (row). Note that the Hungarian algorithm, which is the baseline method, is an offline method. In both HSTunnel and HSTunnel _NO_MISS, all RT methods outperform Hungarian algorithm. The performance of real-time RT outperforms RT-w1, -w2, and -w3 methods. Note that camera settings (C_3, C_5) and (C_4, C_5) perform poor in HSTunnel. The reason is the same as mentioned in Section 4.3, where a number of miss detections exist in candidates of C_3 and C_4. Especially the RT in (C_3, C_5), most desired candidates are not in the search window, only 16% correctness can be obtained.

Table 4-4. Average accuracy of real-time methods on HSTunnel.

Method	HSTunnel										
	C_1, C_2	C_1, C_3	C_1, C_4	C_1, C_5	C_2, C_3	C_2, C_4	C_2, C_5	C_3, C_4	C_3, C_5	C_4, C_5	Avg
Hungarian algorithm	0.64	0.39	0.35	0.60	0.37	0.48	0.68	0.68	0.42	0.53	0.51
S^2DP + RT-w3	0.70	0.76	0.66	0.63	0.73	0.72	0.61	0.64	0.20	0.49	0.61
S^2DP + RT-w2	0.74	0.78	0.71	0.70	0.76	0.74	0.74	0.65	0.21	0.54	0.66
S^2DP + RT-w1	0.90	0.83	0.72	0.71	0.82	0.75	0.77	0.58	0.24	0.51	0.68
S^2DP + RT	**0.92**	**0.86**	**0.76**	**0.80**	**0.84**	**0.78**	**0.78**	**0.65**	**0.37**	**0.60**	**0.74**

Table 4-5. Average accuracy of real-time methods on HSTunnel_NO_MISS.

Method	HSTunnel_NO_MISS										
	C_1, C_2	C_1, C_3	C_1, C_4	C_1, C_5	C_2, C_3	C_2, C_4	C_2, C_5	C_3, C_4	C_3, C_5	C_4, C_5	Avg
Hungarian algorithm	0.72	0.64	0.55	0.60	0.56	0.73	0.75	0.83	0.61	0.78	0.68
S^2DP + RT-w3	0.65	0.57	0.51	0.56	0.51	0.61	0.61	0.79	0.52	0.60	0.59
S^2DP + RT-w2	0.77	0.73	0.67	0.70	0.61	0.80	0.73	0.83	0.65	0.71	0.72
S^2DP + RT-w1	0.91	0.83	0.80	0.83	0.73	0.86	0.78	0.91	0.75	0.83	0.82
S^2DP + RT	**0.93**	**0.87**	**0.88**	**0.84**	**0.80**	**0.88**	**0.85**	**0.92**	**0.75**	**0.84**	**0.86**

4.4.2 Offline Refinement

Our offline algorithm OR is compared with Hungarian algorithm, S^2DP, and the state-of-the-art algorithm, Hungarian Voting (HV) [9], for multi-camera vehicle identification in tunnels.

The Hungarian Voting algorithm [9] for multi-camera identification runs as follows. In the beginning, it pushes N^H detected vehicles into respective queues for two cameras, and obtains the $N^H \times N^H$ distance matrix. Next the Hungarian algorithm is applied to the distance matrix, and the assigned row-column pairs are treated as votes of valid results. The matched pair (of vehicles from two cameras) with highest votes is returned as result and the two corresponding elements are removed from queues. Finally two following detections in two cameras are pushed into two queues and loop the process. We set $N^H = 10$ in our experiments which has reasonable performance.

In addition, the Hungarian Voting algorithm cannot solve the initialization problem described in Section 3.5. HV performs poorly in HSTunnel which requires solving the problem first. Therefore, we apply the S^2DP algorithm in the first step, followed by HV for offline assignments. This method is called S^2DP+HV and is used in HSTunnel dataset. S^2DP+HV is not included in experiments on HSTunnel_NO_MISS since this dataset does not require solving the initialization problem, and the performance is the same as HV.

Similar to the experiments in Section 4.3, we leave N_1^C candidates in the first camera and assigned by N_2^C vehicles in the second camera. Here N_1^C is set as 65, and N_2^C is set as 60. Table 4-6 shows the experimental results on HSTunnel and HSTunnel_NO_MISS, respectively. The tested methods are S^2DP, Hungarian algorithm, OR, Hungarian-voting from related work, and S^2DP+HV.

As shown in Table 4-6, OR outperforms S^2DP and Hungarian Voting. As stated before, the performance of Hungarian-voting drops dramatically in HSTunnel, and it is even worse than baseline Hungarian algorithm. The Hungarian Voting does not include the initialization problem, so that proper candidates would not present in the search window in the whole processing. The result of the refined S^2DP+HV method is shown in the last column of Table 4-6, yet our proposed method still outperforms S^2DP+HV.

Table 4-6. Comparison of average accuracy using different offline methods.

Dataset	Hungarian algorithm	Hungarian voting (HV)	S^2DP+HV	**S^2DP**	**OR**
HSTunnel	0.524	0.320	0.718	**0.865**	0.820
HSTunnel _NO_MISS	0.630	0.848	N/A	0.939	**0.971**

4.5 Discussions

The parameter settings of our proposed algorithms are discussed in this section. In addition, two more datasets are included to further verify our methods.

4.5.1 Miss-Match Penalty

The miss-match penalty ϵ plays an important role in our algorithm. Section 3.4.2 describes the semantic meaning of ϵ that the value is around half of the matching (assignment) distance threshold. However, it is still difficult to determine the maximum distance in applications.

Take a look at the example of the distance matrix in Figure 4-9. Ideally the algorithm will assign vehicles at (1, 3), (2, 4) and (3, 5) since the three distance values are smaller than others. In fact, most values in the distance matrix are greater than those matched points. This gives us inspiration to use the average value of the distance matrix as matching threshold, which equals to 2ϵ. The detail of miss-match penalty is discussed in Section 3.4.1.

		1	2	3	4	5
		Y	Z	A	B	C
1	A	9	9	**5**	8	9
2	B	8	10	11	**4**	11
3	C	6	10	10	10	**5**
4	D	8	8	9	9	9
5	E	12	11	8	9	9
6	F	7	10	9	9	9

Figure 4-9. An example of distance matrix. Most distance values are greater than those matched ones.

We test our S^{2}DP with different values of ϵ. The experiment assigns 30 vehicles in the second camera from 65 candidates in the first camera by increasing the value of ϵ by 10 each time. The result is the average accuracy of 15 iterations of random starting points.

Figure 4-10 shows the experimental results on HSTunnel and HSTunnel _NO_MISS, where the x-axis is miss-match penalty ϵ and the y-axis is the average accuracy. In HSTunnel_NO_MISS, the ϵ value greater than 440 gets similar performance since there is no miss detection. The best ϵ value is 430 in HSTunnel and the performance reduces when ϵ becomes larger. Because HSTunnel contains miss detections, the algorithm cannot claim misses if the ϵ is too large and thus decreases accuracy. Especially in both (C_3, C_5) and (C_4, C_5) which contain many miss detections, the accuracy drops when ϵ is greater than 450.

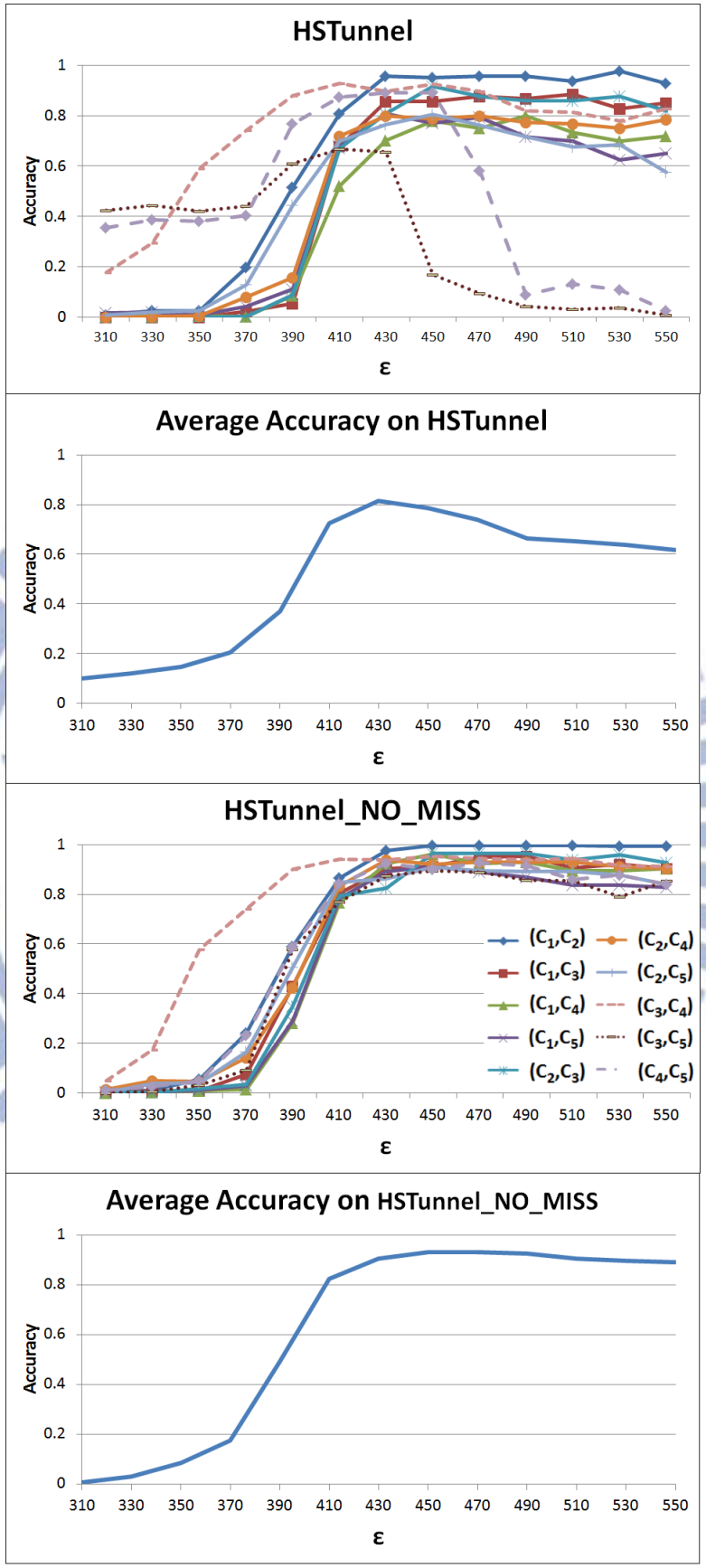


Figure 4-10. Experiments on miss-match penalty ϵ of two datasets. The x-axis is the value of penalty, the y-axis is corresponding accuracy.

Table 4-7 and Table 4-8 show the performance on using half of average distance of distance matrix as ϵ and one of the best result in Figure 4-10. The first column is the tested cameras, the second and the third columns show the half value of average distance in the distance matrix and the corresponding accuracy using this value. The fourth and fifth columns are the best result in Figure 4-10. Finally the last column shows the difference between the third and the fifth column. Although the performance drops when using average distance as 2ϵ, the average differences in accuracy are 0.042 and 0.060 in HSTunnel_NO_MISS and HSTunnel, respectively. As a result, we can use average value in distance matrix as 2ϵ with at most 6% accuracy loss. Besides, we can use the half value of average distance in the distance matrix as initial condition and find the best value manually.

Table 4-7. Comparison on the average-distance strategy and the best case of miss-match penalty ϵ on HSTunnel.

Camera	AvgDistance ÷2 as ϵ	Accuracy	Best ϵ	Accuracy	Difference in accuracy
(C1,C2)	427	0.957	530	0.976	0.019
(C1,C3)	426	0.858	540	0.896	0.038
(C1,C4)	427	0.702	540	0.789	0.087
(C1,C5)	428	0.807	460	0.822	0.015
(C2,C3)	430	0.807	450	0.915	0.108
(C2,C4)	431	0.800	440	0.813	0.013
(C2,C5)	430	0.764	460	0.833	0.069
(C3,C4)	419	0.924	420	0.924	0.000
(C3,C5)	429	0.656	420	0.873	0.217
(C4,C5)	430	0.891	420	0.927	0.036
Average	N/A	0.817	N/A	0.877	**0.060**

Table 4-8. Comparison on the average-distance strategy and the best case of miss-match penalty ϵ on HSTunnel_NO_MISS.

Camera	AvgDistance ÷2 as ϵ	Accuracy	Best ϵ	Accuracy	Difference in accuracy
(C1,C2)	425	0.976	460	0.998	0.022
(C1,C3)	425	0.904	460	0.956	0.052
(C1,C4)	426	0.924	450	0.962	0.038
(C1,C5)	428	0.891	450	0.911	0.020
(C2,C3)	430	0.824	480	0.971	0.147
(C2,C4)	430	0.938	500	0.957	0.019
(C2,C5)	430	0.864	440	0.916	0.052
(C3,C4)	418	0.944	460	0.956	0.012
(C3,C5)	429	0.876	440	0.922	0.046
(C4,C5)	430	0.924	420	0.931	0.007
Average	N/A	0.907	N/A	0.948	**0.042**

4.5.2 More Datasets

We further test proposed algorithms on two more tunnel datasets. Figure 4-11 is some examples of datasets HSTunnel2 and BGTunnel.

Figure 4-11. Examples of datasets HSTunnel2 and BGTunnel.

The HSTunnel2 contains two cameras in Hsuehshan tunnel, Taiwan [31], but in different driving direction of HSTunnel. We manually labeled 65 vehicles in the first camera and 60 in the second camera, and there is no miss detection and no order-changed vehicle included. The first five vehicles in the second camera are absent in the first camera, which satisfies the condition of initialization problem described in Section 3.4.1. The driving distance between two cameras is about two kilometers.

The BaGuashanTunnel (BGTunnel) dataset contains two cameras in Baguashan tunnel, Taiwan [31], which contains 65 vehicles in the first camera and 60 vehicles in the second camera. Five miss detections and six order-changed vehicles are included. This dataset is more challenging than HSTunnel2 due to the different camera viewpoints. In addition, the color information is poor since every vehicle looks monotonic in the tunnel.

Four methods are tested on HSTunnel2 and BGTunnel: real-time RT, offline OR, baseline Hungarian algorithm, and S^2DP+HV modified from related work. Table 4-9 shows that RT and OR outperform S^2DP+HV and are superior to Hungarian algorithm in both HSTunnel2 and BGTunnel datasets.

Table 4-9. Average identification accuracy on HSTunnel2 and BGTunnel datasets.

Method	HSTunnel2	BGTunnel
RT (proposed)	**0.933**	**0.700**
OR (proposed)	**0.983**	**0.767**
Hungarian algorithm	0.695	0.317
S^2DP+HV (related work)	0.915	0.450

Finally, the performance of all methods in dataset HSTunnel, HSTunnel2, and BGTunnel3 are summarized in Figure 4-12. The x-axis is the datasets with different methods and the y-axis is the average accuracy. The OR outperforms all other methods. The performances of RT and S^2DP+HV are similar, but RT can run in real-time and S^2DP+HV cannot.

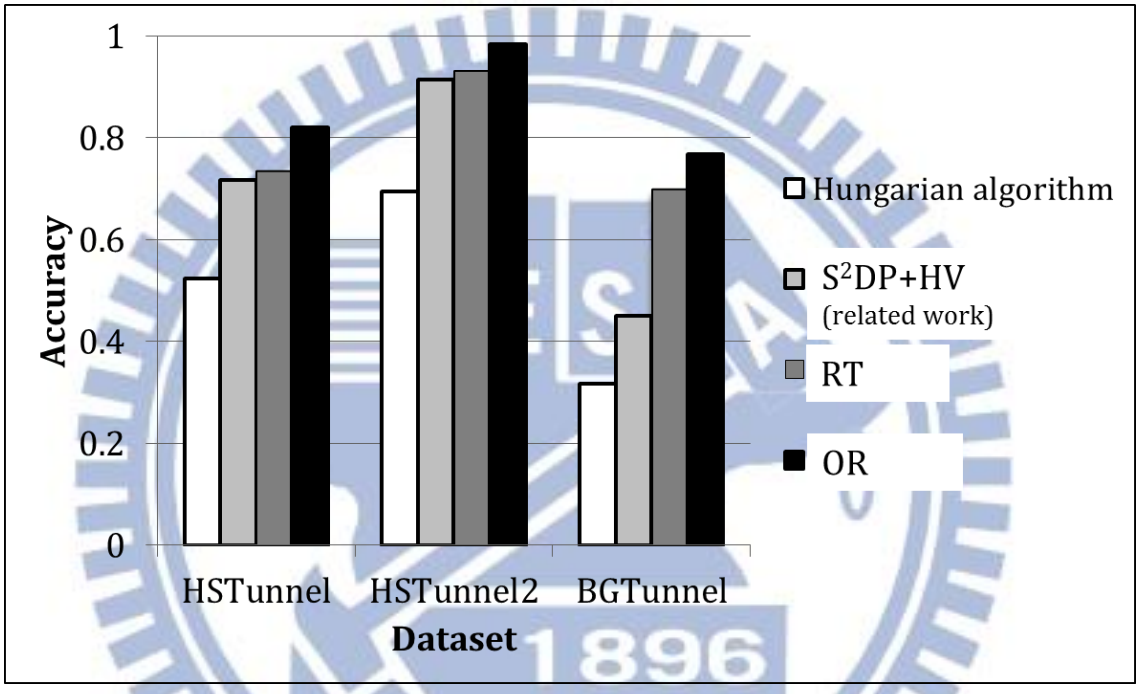


Figure 4-12. Overall performance of all methods in datasets HSTunnel, HSTunnel2, and BGTunnel. The x-axis is the datasets and the y-axis is the average accuracy.

Chapter 5. Conclusion and Future Work

This thesis has proposed a system for multi-camera vehicle identification in tunnel surveillance videos. As vehicles drive through a tunnel, they appear in all surveillance cameras. By applying the proposed algorithms, vehicles from different cameras can be identified.

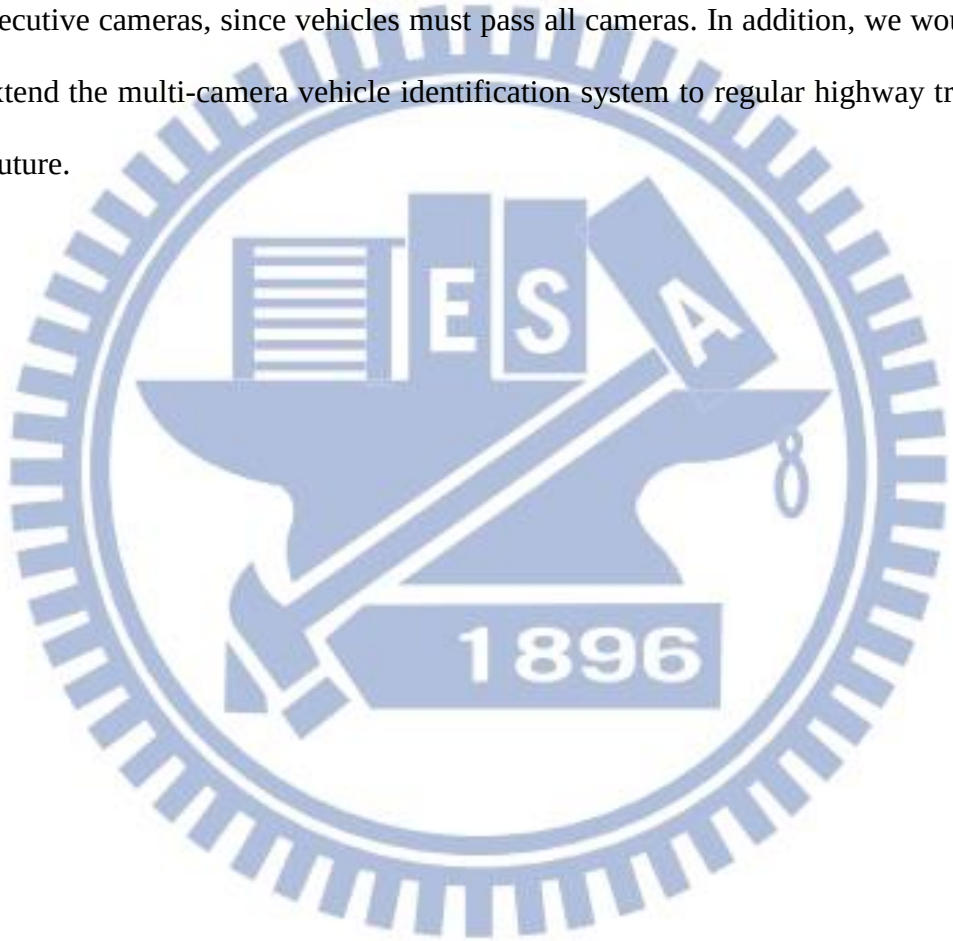
In the beginning, vehicles are detected with Haar-like feature detector and transformed to a visual feature vector using OpponentSIFT descriptor in an individual camera video. After the system collects a set of vehicles in multiple cameras for a while, the proposed Spatiotemporal Successive Dynamic Programming (S^2DP) algorithm is applied to identify vehicles across two cameras by considering the ordering constraint in a tunnel.

Usually there are two major requirements in a multi-camera traffic system, real-time tracking and offline identification. Therefore, two algorithms are proposed for the two purposes, respectively. The Real-Time RT algorithm gives an assignment strategy for fast candidate selection and can be used in multi-camera tracking-by-identification. Another algorithm is the Offline Refinement OR that refines the result of the S^2DP using Hungarian algorithm that achieves better performance. Meanwhile, practical issues such as initialization problem and error handling are taken into consideration in our proposed system.

Comprehensive experiments on every part of the system are provided using three manually labeled tunnel surveillance datasets. The three datasets include not only labeled vehicles, but also miss detected vehicles and order-changed vehicles. The

proposed RT and OR algorithms demonstrate good performance on different datasets, and both outperform state-of-the-art algorithms.

Mixing information from more than two cameras is left for future work. A miss-detected vehicle in one camera can be recovered using the information from the cameras in behind. Alarms can be triggered if one vehicle is miss-detected in consecutive cameras, since vehicles must pass all cameras. In addition, we would like to extend the multi-camera vehicle identification system to regular highway traffic in the future.



Bibliography

[1] B. Tian, Q. Yao, Y. Gu, K. Wang and Y. Li, "Video Processing Techniques for Traffic Flow Monitoring: A Survey," in *Proc. IEEE Int'l Conf. Intelligent Transportation Systems*, 2011.

[2] M. S. Shehata, J. Cai, W. M. Badawy, T. W. Burr, M. S. Pervez, R. J. Johannesson and A. Radmanesh, "Video-Based Automatic Incident Detection for Smart Roads: The Outdoor Environmental Challenges Regarding False Alarms," *IEEE Transactions on Intelligent Transportation Systems,* vol. 9, no. 2, pp. 349-360, Jun. 2008.

[3] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743-761, Apr. 2012.

[4] T. Gandhi and M. M. Trivedi, "Pedestrian Protection Systems: Issues, Survey, and Challenges," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, no. 3, pp. 413-130, Sep. 2007.

[5] C. Rougier, J. Meunier, A. St-Arnaud and J. Rousseau, "Robust Video Surveillance for Fall Detection Based on Human Shape Deformation," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 21, no. 5, pp. 611-622, May 2011.

[6] R. S. Feris, B. Siddiquie, J. Petterson, Y. Zhai, A. Datta, L. M. Brown and S. Pankanti, "Large-Scale Vehicle Detection, Indexing, and Search in Urban Surveillance Videos," *IEEE Transactions on Multimedia,* vol. 14, no. 1, pp. 28-42, Feb. 2012.

[7] B. T. Morris and M. M. Trivedi, "Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2287-2301, Nov. 2012.

[8] I. Saleemi, L. Hartung and M. Shah, "Scene Understanding by Statistical Modeling of Motion Patterns," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[9] R. R. Cabrera, T. Tuytelaars and L. Van Gool, "Efficient Multi-Camera Vehicle Detection, Tracking, and Identification in a Tunnel Surveillance Application," *Computer Vision and Image Understanding*, vol. 116, no. 6, pp. 742-753, Jun. 2012.

[10] R. R. Cabrera, T. Tuytelaars and L. Van Gool, "Efficient Multi-Camera Detection, Tracking, and Identification Using a Shared Set of Haar-Features," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

[11] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proc. IEEE CS. Conf. Computer Vision and Pattern Recognition*, 2001.

[12] N. Buch, S. A. Velastin and J. Orwell, "A Review of Computer Vision Techniques for the Analysis of Urban Traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920-939, Sep. 2011.

[13] J. Zhang, F. Y. Wang, K. Wang, W. H. Lin and C. Chen, "Data-Driven Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624-1639, Dec. 2011.

[14] S. Atev, G. Miller and N. P. Papanikolopoulos, "Clustering of Vehicle

Trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 647-657, Sep. 2010.

[15] M. Piccardi, "Background subtraction techniques: a review," in *Proc. IEEE Int'l Conf. Systems, Man, Cybernetics*, 2004.

[16] S. Brutzer, B. Hoferlin and G. Heidemann, "Evaluation of Background Subtraction Techniques for Video Surveillance," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

[17] C. Stauffer and W. E. L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," in *Proc. IEEE CS. Conf. Computer Vision and Pattern Recognition*, 1999.

[18] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. IEEE CS. Conf. Computer Vision and Pattern Recognition*, 2005.

[19] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier and L. Van Gool, "Online Multiperson Tracking-by-Detection from a Single, Uncalibrated Camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1820-1833, Sep. 2011.

[20] C. R. Wren, A. Azarbayejani, T. Darrell and A. P. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780-785, Jul. 1997.

[21] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174-188, Feb. 2002.

[22] X. Wang, "Intelligent Multi-Camera Video Surveillance: A Review," *Pattern*

Recognition Letters, vol. 34, no. 1, pp. 3-19, Jan. 2013.

[23] G. Lian, J. H. Lia, C. Y. Suen and P. Chen, "Matching of Tracked Pedestrians Across Disjoint Camera Views Using CI-DLBP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 7, pp. 1087-1099, Jul. 2012.

[24] K. E. A. van de Sande, T. Gevers and C. G. M. Snoek, "Evaluating Color Descriptors for Object and Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1582-1596, Sep. 2010.

[25] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004.

[26] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[27] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *Proc. European Conference on Computer Vision*, 2006.

[28] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Proc. European Conference on Computer Vision*, 2010.

[29] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83-97, 1955.

[30] P. F. Felzenszwalb and R. Zabih, "Dynamic Programming and Graph Algorithms in Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 721-740, Apr. 2011.

[31] "Taiwan Area National Freeway Bureau, MOTC," [Online]. Available: http://www.freeway.gov.tw.