3D Modelling for Improved Visual Traffic Analytics

Eduardo R. Corral-Soto

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

MARCH 2018

Abstract

Advanced Traffic Management Systems utilize diverse types of sensor networks with the goal of improving mobility and safety of transportation systems. These systems require information about the state of the traffic configuration, including volume, vehicle speed, density, and incidents, which are useful in applications such as urban planning, collision avoidance systems, and emergency vehicle notification systems, to name a few. Sensing technologies are an important part of Advanced Traffic Management Systems that enable the estimation of the traffic state. Inductive Loop Detectors are often used to sense vehicles on highway roads. Although this technology has proven to be effective, it has limitations. Their installation and replacement cost is high and causes traffic disruptions, and their sensing modality provides very limited information about the vehicles being sensed. No vehicle appearance information is available. Traffic camera networks are also used in advanced traffic monitoring centers where the cameras are controlled by a remote operator. The amount of visual information provided by such cameras can be overwhelmingly large, which may cause the operators to miss important traffic events happening in the field.

This dissertation focuses on visual traffic surveillance for Advanced Traffic Management Systems. The focus is on the research and development of computer vision algorithms that contribute to the automation of highway traffic analytics systems that require estimates of traffic volume and density.

This dissertation makes three contributions:

The **first contribution** is an integrated vision surveillance system called 3DTown, where cameras installed at a university campus together with algorithms are used to produce vehicle and pedestrian detections to augment a 3D model of the university with dynamic information from the scene.

A **second major contribution** is a technique for extracting road lines from highway images that are used to estimate the tilt angle and the focal length of the camera. This technique is useful when the operator changes the camera pose.

The **third major contribution** is a method to automatically extract the active road lanes and model the vehicles in 3D to improve the vehicle count estimation by individuating 2D segments of imaged vehicles that have been merged due to occlusions.

Dedication

To Tatyana Ignatyev, Alexander Corral-Ignatyev, Eduardo R. Corral-Ibarra, Maria Teresa Soto-Camargo, and Virginia Camargo de Soto.

Acknowledgements

I would like to thank my PhD supervisor, Prof. James H. Elder for his patience and firm commitment to helping me become a better scientist, and for his continuous guidance and supervision throughout my PhD.

I also want to thank Bob Hou for being a great support in multiple aspects during my PhD at York, and to all my fellow students in the Elder lab for their help and valuable feedback during my presentations and meetings.

Thanks to my wife Tatyana, and to my two sons, Alexander and Eduardo, for their patience and support during all these years of hard work and sacrifices.

And many thanks to the examining committee members: Prof. Michael S. Brown, Prof. Petros Faloutsos, Prof. Gunho Sohn, Prof. Marin Litoiu, Prof. Steven Waslander, and Prof. James H. Elder.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

An intelligent transportation system (ITS) as defined by the Intelligent Transportation Systems Society of Canada (ITS Canada) is the application of advanced and emerging technologies (computers, sensors, control, communications, and electronic devices) in transportation to save lives, time, money, energy and the environment [1, 2].

Under the umbrella of ITS, Advanced Traffic Management Systems (ATMS) utilize diverse types of roadway sensor networks with the goal of improving mobility, safety, and productivity of transportation systems. These systems require information about the operational state and characteristics of the traffic, including traffic volume, vehicle speed, traffic density, occupancy, incidents, and weather conditions, which are useful in applications such as urban planning, collision avoidance systems, and emergency vehicle notification systems, to name a few. The COMPASS (`http://www.mto.gov.on.ca/english/traveller/trip/compass.shtml`) freeway traffic management system in Ontario, Canada is an example of ATMS that improves emergency assistance, provides timely traffic information, decreases motor vehicle collisions, increases safety assurance, reduces congestion as well as travel times and pollution.

Sensing technologies are an important part of Advanced Traffic Management Systems to enable the estimation of the traffic state. In particular, inductive loop detectors,

which are electrical wire loops embedded in the road surface (Figure 1.1), are typically used to sense travelling vehicles on city and highway roads with the objective of estimating the traffic state.

Although this technology has proven to be effective and accurate for detecting and counting vehicles, it has its own limitations: namely, their installation and replacement cost is high and causes traffic disruptions, especially if not installed correctly [1], and multiple loops are required to monitor different road lanes. Moreover, their sensing modality provides very limited information about the vehicles being sensed. No vehicle appearance information is available for applications such as appearance-based vehicle classification [3] or license plate recognition [4]. Other sensing technologies are available for ATMS applications, inlcuding, magnetometers, radar, video cameras, infrared and ultrasonic sensors, and LIDAR. Each of them has advantages and disadvantages as explained in detail in [1].

During the last decades, the use of Pan-Tilt-Zoom (PTZ) cameras in traffic monitoring systems has gained traction. These cameras are typically installed on poles or overpass bridges located next to or over the road at heights of up to 21 m [1]. Typical traffic monitoring centers are comprised of networks of several cameras that are controlled and monitored by a remote operator using switches and multi-paneled displays as shown in Figure 1.2(a). The amount of visual information provided by such camera networks can be overwhelmingly large, which may cause the operators to miss important traffic events happening in the field. Thus, computer vision systems are desirable for automating the surveillance process.

This dissertation is focused on visual traffic surveillance for Advanced Traffic Management Systems. More specifically, the focus is on the research and development of computer vision algorithms and methods that contribute to the automation of highway traffic analytics systems that require estimates of traffic volume (number of vehicles passing a reference point per unit of time), vehicle speed (distance travelled by a vehi-

Figure 1.1: Road traffic monitoring via inductive loops. Example indictive loop configuration where a diamond-shaped loop is installed on each lane of the road. Source: [1].

cle per unit of time, usually expressed in km/h), and traffic density (number of vehicles occupying a road lane per unit of length at a given point in time).

1.1 Problem Definition

The problem under consideration is simple: How to determine from a single camera the number of vehicles in highway traffic scenes. The main challenges that we identify are: 1.-Determining the mapping from the 3D world scene onto the image plane (i.e. camera calibration) given that the camera pose can be changed by the remote operator, 2.-The extraction of road lanes (which can be linear or curved), and 3.- The segmentation of vehicles that are occluded by other vehicles.

(a) (b)

Figure 1.2: Road traffic monitoring via surveillance cameras - (a) Example of human monitoring of a large number of traffic surveillance cameras. (b) Example image from a real traffic surveillance camera.

1.2 Summary of Contributions

This dissertation presents three technical contributions that attempt to address the problem stated above. These contributions are summarized as follows:

1. A distributed prototype surveillance system we call 3DTown [5] for sensing, interpreting and visualizing the real-time dynamics of urban life within the 3D context of a city. The main strategy is the automatic integration of 3D urban models with data from pan/tilt video cameras, environmental sensors and other real-time information sources used for the three-dimensionalization of pedestrians and vehicles tracked in 2D camera video, aided by the automatic real-time computation of camera pose relative to the 3D urban environment.

   This contribution was the result of a joint collaboration by the following team members (author names ordered as in [5]): Eduardo R. Corral-Soto, Ron Tal, Langyue Wang, Ravi Persad, Luo Chao, Chan Solomon, Bob Hou, Gunho Sohn, and James H. Elder. I was the main author in that paper [5], which was published in the Computer and Robot Vision (CRV), 2012 Conference.

2. A novel and effective single-view probabilistic method [6] that extracts the road lane lines/curves from highway images and exploits parallelism between coplanar curves to iteratively estimate the tilt angle and, if possible, the focal length of the surveillance camera on linear or curved roads (unlike typical methods based on vanishing points extracted from straight roads and orthogonal structures). Our method is based on establishing associations iteratively between pairs of corresponding points lying on the image projection of these curves.

3. A novel data-driven 3D generative reasoning method [7] that automatically segments the active road lanes (highway lane structure) from traffic video and models the vehicles in 3D to disaggregate individual vehicles from 2D image clusters formed due to dense traffic conditions. The method estimates the vehicle configuration and dimensions that provide the most likely account of the observed foreground pixels.

The proposed methods have been evaluated objectively and/or compared against state-of-the art methods.

Each of these three contributions is described in detail in the next chapters of this dissertation document, and each chapter contains its own introduction, geometry, algorithm, data sets and evaluation sections.

Chapter 2

First Contribution: 3DTown - The Automatic Urban Awareness Project

The work described in this chapter was the result of a joint collaboration by the following team members (author names ordered as in [5]): Eduardo R. Corral-Soto, Ron Tal, Langyue Wang, Ravi Persad, Luo Chao, Chan Solomon, Bob Hou, Gunho Sohn, and James H. Elder. I was the main author in that paper [5], which was published in the Computer and Robot Vision (CRV), 2012 Conference.

My main contributions in the 3DTown project were:

1. The computation of the camera projection matrix and the mapping between image locations and virtual ground plane (See sub-section 2.4.1).

2. The adaptation of an existing pre-attentive foreground extraction method to pedestrian and vehicle detection (See Appendix A).

3. A simple mechanism to detect pedestrian blobs via temporal processing of posterior maps coupled with smoothing, peak detection, and nearest neighbour-based tracking (Section 2.5).

4. Collaboration with other team members to get the 3DTown system to work in live mode (e.g. update rotation matrix, read images, compute updated pedestrian ground locations, and demo preparations).

5. Experiments, qualitative evaluations, and demonstrations.

The 3DTown project was a qualitative study focused on the development of a distributed system for sensing, interpreting and visualizing the real-time dynamics of urban life within the 3D context of a city. In this project, the 3DTown motivates the work presented in the subsequent chapters, which present more quantitative results.

At the heart of the 3DTown lies a core of algorithms that automatically integrate 3D urban models with data from pan/tilt video cameras, environmental sensors and other real-time information sources. A key challenge is the three-dimensionalization of pedestrians and vehicles tracked in 2D camera video, which requires automatic real-time computation of camera pose relative to the 3D urban environment. Here, I report qualitative results from a prototype system we call 3DTown, which is composed of discrete modules connected through pre-determined communication protocols. These modules consist of: 1) A 3D modeling module that allows for the efficient reconstruction of building models and integration with indoor architectural plans; 2) A GeoWeb server that indexes a 3D urban database to render perspective views of both outdoor and indoor environments from any requested vantage; 3) Sensor modules that receive and distribute real-time data; 4) Tracking modules that detect and track pedestrians and vehicles in urban spaces and access highways; 5) Camera pose modules that automatically estimate camera pose relative to the urban environment; 6) Three-dimensionalization modules that receive information from the GeoWeb server, tracking and camera pose modules in order to back-project image tracks to geolocate pedestrians and vehicles within the 3D model; 7) An animation module that represents geo-located dynamic agents as sprites; and 8) A web-based visualization module that allows a user to explore the resulting dynamic 3D visualization in a number of

interesting ways.

To demonstrate our system we have used a blend of automatic and semi-automatic methods to construct a rich and accurate 3D model of a university campus, including both outdoor and indoor detail. The demonstration allows web-based 3D visualization of recorded patterns of pedestrian and vehicle traffic on streets and highways, estimations of vehicle speed, and real-time (live) visualization of pedestrian traffic and temperature data at a particular test site. Having demonstrated the system for hundreds of people, we report our informal observations on user reaction, potential application areas and the main challenges that must be addressed to bring the system closer to deployment.

2.1 Introduction

In recent years 3D Geographic Visualization Environments such as Google Earth and Microsoft Virtual Earth have demonstrated the power of making 3D geographic information broadly available over the internet. However, these tools only provide information about the static infrastructure of the urban environment. Since cities are by definition centres of activity, this limits the utility of these databases for many potential applications. On the other hand, the number of surveillance video cameras installed in urban areas grows every year as cameras become less expensive and as the demand for security and monitoring systems grows. This includes networks of cameras installed at universities, airports, train stations, shopping malls, highways, etc.

With the potential benefit of having vast amounts of visual information comes the problem of viewing and analyzing the image data, which still today typically involves one or more human operators. This monitoring task can become overwhelming and inefficient if there are many cameras and few operators. For example if a pedestrian walks out of the field of view of camera A and enters the field of view of camera B, the operator may need some time to understand how the pedestrian is moving in the

3D world.

2.1.1 Prior Work

The problem of automatically mapping tracked agents in video to a 3D model has been studied previously in a number of different contexts. For example, Kanade et al ([8, 9]) developed a cooperative multi-sensor video surveillance and monitoring system (VSAM) for military applications that mapped the dynamics extracted from multiple distributed video cameras from different videos of a scene onto the appropriate areas of a common static 3D model of the corresponding scene. Their method allows the operator to see the projected scene dynamics in the context of the 3D model, thus obtaining a 3D situational awareness of the current dynamic environment. Some years later, a similar system was reported by Sawhney et al [10] with the added feature that parts of the 3D model were texture-mapped with the projected live video images from the cameras. The system was implemented on commodity graphics hardware.
At roughly the same time, another group [11] began the development of a similar system called Augmented Virtual Environments (AVE) where GPS devices, orientation sensors and video cameras contained in a tracking backpack that can be carried by a pedestrian were used to create augmented virtual environments. This system was later enhanced [12] with the introduction of background subtraction-based detection of moving objects followed by a pseudo-tracking step to extract track. More recently in [13], the authors divided the problem into four scenarios based on the number of cameras, overlap of their fields of view, and types of motion: direct mapping (pedestrians), overlapping cameras with complex motion (sports), sparse cameras with simple motions (traffic), and sparse cameras with complex motion (clouds). In that work, optical flow-based tracking and planar homographies are used to augment Aerial Earth Maps (AEM) with dynamic information from pedestrians and vehicles.

In practical video surveillance systems, the camera pose is often changed by the

operator. When this occurs, it is necessary to update the camera matrix in order to re-compute the geolocation of tracked objects in the 3D model and potentially perform automatic view updating of the model. None of the approaches discussed above focuses on this problem with the exception of [10] which describes a pose estimation method based on video image correspondences. However that method requires manual initialization and does not run in real-time.

In the 3DTown project, the goal was to develop a distributed system for sensing, interpreting and visualizing the real-time dynamics of urban life within the 3D context of a city. A clear advantage and main novelty in our system with respect to the approaches mentioned above is the introduction of an automatic on-line single-view method for updating the rotation matrix component of our virtual camera for the cases when the operator changes the pose.

Scenes contain several types of dynamic information. However we focus on typical, useful urban dynamic information, specifically walking pedestrians and moving vehicles captured by surveillance video cameras, and environmental signals coming from indoor temperature and motion sensors. In order to achieve our goal we have to integrate these different types of real-time dynamic information with a 3D virtual environment corresponding to an urban scene.

As a test site, we have selected a university campus that has a diverse mixture of building types, pedestrian, and vehicle traffic. Over a 3D texture-mapped terrain surface model acquired from Google Earth, we have constructed both exterior and interior models of all the campus buildings, using a diverse set of automatic, semi-automatic, and manual techniques (see below). Both recorded and live video data are acquired from pan-tilt-zoom (PTZ) video cameras situated at diverse sites over the campus. Tracking algorithms provide trajectories of pedestrians and vehicles in image coordinates. At a slower data rate, an automatic camera calibration algorithm computes the current pose of each camera relative to a world coordinate frame, allowing tracks

to be back-projected to the 3D model. Intersections of these backprojections with the ground plane determine the geolocation of tracked agents, which are then represented as sprites,thus allowing the dynamics to be visualized in 3D within a web-based Google Earth environment. We also introduce an on-line system for monitoring temperatures in an indoor 3D building model which enables the temperature visualization of the whole floor of a building at once. We describe our methods to create photo-realistic indoor and outdoor 3D models of buildings, and our augmentation of Google Earth 3D urban models in real-time from surveillance cameras. In the current version of the 3DTown, our work is mainly focused on detecting, tracking, geolocating and rendering agents within the field of view of a single camera rather than tracking agents between cameras.

The rest of the chapter is structured as follows: Section 2.2 describes the overall system architecture, Section 2.3 describes our virtual 3D world and how we create our own virtual building models, Section 2.4 describes how we map the PTZ camera 2D image points onto the virtual 3D world and how we automatically update the rotation matrix, Section 2.5 explains how we track pedestrians and vehicles, Sections 2.6 and 2.7 show system integration details and results, Section 2.8 includes our qualitative evaluation of the system, and Section 2.9 contains final comments and plans for future work.

2.2 System Architecture Overview

Figure 2.1 shows the overall architecture of our system. The inputs to our system are surveillance video images, signals from environmental sensors, terrain surface 3D models provided by Google Earth, virtual 3D building models and geographic vector data such as roads and parking lots from Geographic Information Systems (GIS) and other map databases. The object tracking module produces image coordinates of the tracked moving objects, and the camera projection matrix estimation module computes

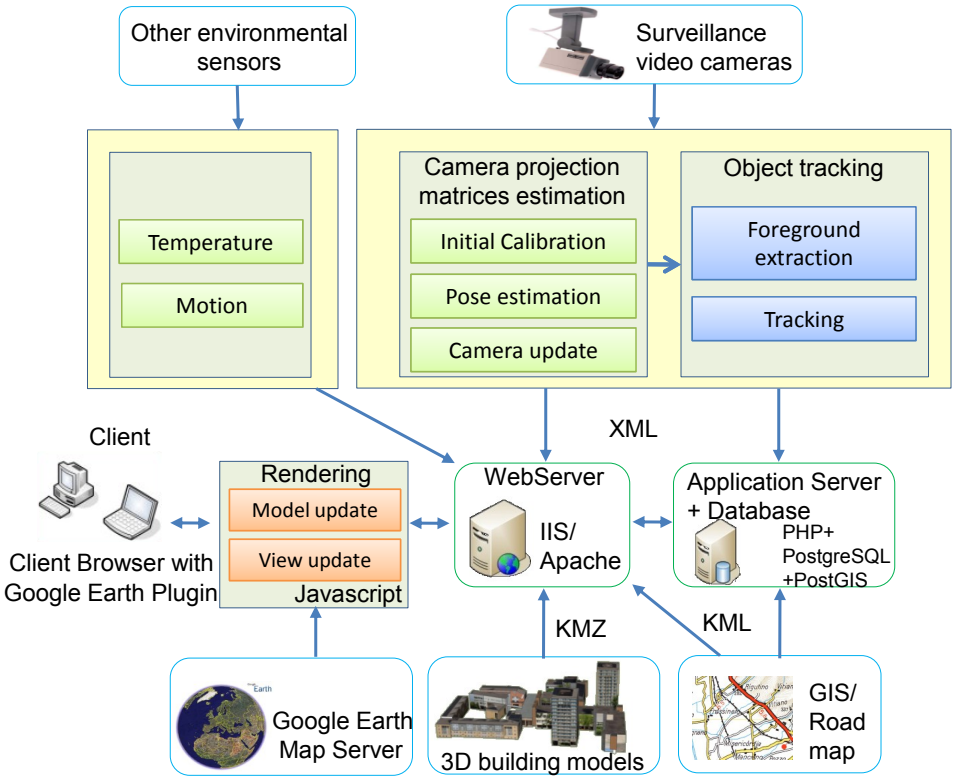


Figure 2.1: Simplified system architecture diagram.

an updated camera matrix that is used to map these image coordinates onto Universal Transverse Mercator (UTM) coordinates in the 3D virtual world. Both camera pose information and the object UTM coordinates are written asynchronously onto XML (Extensible Markup Language) files on a webserver which are read by other modules in order to perform the rendering. The inputs from the Google Earth map server are virtual aerial 3D maps of the terrain. The inputs from the 3D building models module are the building models that we created, and the GIS/Road map data are additional inputs that we may use in the future. The rest of the blocks in the diagram are related to data storage and rendering. The details of all these modules will be explained in the later sections below.

2.3 The Virtual 3D World

The Google Earth Plug-in and its Javascript Application Programming Interface (API) provide a platform to develop various 3D applications by allowing users to incorporate their own 3D building models and other 3D data via KML files, thus enabling users to create customized 3D virtual worlds. The 3D photo-realistic building model is the most important component in the virtual world since buildings are the dominant man-made objects in urban environments. During the last two decades, a number of algorithms and systems have been developed to construct 3D building models using various input data sources. With the development and wide application of Light Detection and Ranging (LIDAR) techniques, airborne and terrestrial laser scanning data has proven to be a valuable source of information for the construction of 3D building models. The 3D building models used in our system correspond to actual buildings from a university campus.

2.3.1 Automatic Generation of 3D Photorealistic Prismatic Building Models

For 3DTown, we commissioned airborne LIDAR scans of a university campus from an altitude of $2300m$ with a point distance density of about $1.9m$. Terrestrial LIDAR data was also collected with a mobile mapping system. We use these laser scanning data to reconstruct 3D photorealistic prismatic building models using an established processing pipeline [14] consisting of: (1) point cloud registration, (2) automatic tie point collection, (3) geo-referencing, (4) surface modeling, and (5) texture mapping. We used the Iterative Closest Point (ICP) algorithm [15] for the automatic registration of 3D point clouds from multiple scans. In order to ensure that the ICP solution converged correctly, accurate tie points were collected automatically by generating 2D images from 3D point clouds and performing automatic image matching. From these

images, planar regions and their edges were extracted using a region growing method described in [16]. Those 2D edges were then transformed back to 3D to construct 3D planar surface [14]. For texture mapping, a photogrammetric collinearity condition based method was used [17]. The optical images of the buildings used in the texture mapping were captured using a digital camera. We solve for the position and orientation of the camera via least squares and the collinearity condition. The tie points between the photos and LIDAR data were collected by the user through an interactive Graphical User Interface developed by our team. The textures are mapped to the surface model using a nearest neighbour resampling algorithm. Figure 2.5 shows an example of a reconstructed 3D building model.

2.4 Coordinate Transformation Between Cameras and the 3D Model: Determining Camera Matrices

2.4.1 Back-Projection of Image Points onto the 3D World

In order to back-project tracked image locations $\vec{x}_i$ to corresponding 3D positions $\vec{X}_i$ in the world, we need to know both the internal and external parameters of the camera. The parameters are captured by the projection matrix P: $\vec{x} = P\vec{X}$, which can be written as $P = [KR|\vec{t}]$, where K is a 3×3 calibration matrix containing the intrinsic parameters of the camera, R is a 3×3 rotation matrix and $\vec{t}$ is a translation vector.

We make the assumption that our cameras are installed at fixed and known locations, so that the translation vector t is known. However, computation of P is non-trivial since for PTZ cameras, both K and R can change on the fly.

In the 3DTown we make the simplifying assumption that the focal length is fixed. For offline camera data, we select video segments where the zoom was not changed, and use an interactive calibration method [18] to estimate focal length and principal

point. We make the assumption that skew is negligible and scaling parameters in x and y directions are identical, allowing complete identification of the internal camera matrix K. For the live camera, the camera focal length is fixed at its minimum value, and the camera matrix K is estimated using a standard offline method [19].

To perform the back-projection, the remaining step is to compute the camera rotation matrix R, which may change dynamically as the camera pans and tilts. We describe our solution to this problem in the next section.

2.4.2 Estimating the Camera Pose via Manhattan Frames

We use a novel, automatic, on-line, model-free method to maintain a continuous estimate of the rotation matrix R of each camera. Our method estimates the rotation independently for every frame by considering R as the product of two matrices

$$R = R_{M \to UTM} \cdot R_M, \tag{2.1}$$

where R_M defines the rotation of the camera relative to the Manhattan frame (the canonical coordinate system defined by the orthogonal man-made structures in the scene), and $R_{M \to UTM}$ defines the orientation of man-made structures with respect to the UTM coordinate system. Since buildings are static, $R_{M \to UTM}$ need only be computed once. We now turn to the computation of R_M.

We can estimate the pose R_M of the camera with respect to the 3D scene structure by exploiting the rectilinear structure of typical urban environments. In particular, we exploit the so-called *Manhattan assumption* [20] , which states that an urban scene is generally dominated by planar surfaces that conform to three mutually orthogonal directions (e.g. vertical, streets and avenues). Under perspective projection, this regularity gives rise to convergence of the major lines of the urban environment onto a trio of vanishing points on the image plane. Coughlan and Yuille's algorithm [20] es-

timated the three principal Manhattan directions by modeling using a mixture model over the dense map of projected image gradients. Denis et al [18] later improved on these results using a sparser edge-based formulation. Here we introduce a line-based method that we find provides further improvement in accuracy and reliability.

All three of these methods optimize the rotation Ψ between the camera and the Manhattan frame of reference by maximizing the likelihood function over a set of linear perspective cues $E = \{\vec{E}_1, \ldots, \vec{E}_N\}$:

$$\Psi^* = \arg\max_{\Psi} p(E|\Psi) \;=\; \prod_i p(\vec{E}_i|\Psi). \tag{2.2}$$

The association of observations with the Manhattan directions is expressed through a mixture model:

$$p(\vec{E}_i|\Psi) \;=\; \sum_{m_i} p(\vec{E}_i|\Psi, m_i)p(m_i), \tag{2.3}$$

where m_i is the 'Manhattan cause' of the line (vertical, horizontal(1), horizontal(2), background) and $p(m_i)$ is the prior over causes.

To compute the linear perspective cues E_i, we first detect and localize image edges to sub-pixel accuracy [21]. These edges are then grouped into lines using a Hough transform technique [22] that uses a kernel-based voting scheme to propagate the uncertainty of edge observations onto the parameter domain. A common problem with Hough methods is the multiple response problem: multiple peaks detected in the Hough map that correspond to a single linear structure in the scene. In our approach, these multiple responses are avoided by employing a stepwise probabilistic subtraction method [23] that subtracts off the contributions of edge observations that correspond to previously detected lines.

The resulting set of detected image lines can be used to recover R_M by considering the *Gauss sphere* representation of the problem [24]. A detected line in the image plane together with the optical centre of the camera define an *interpretation plane*.

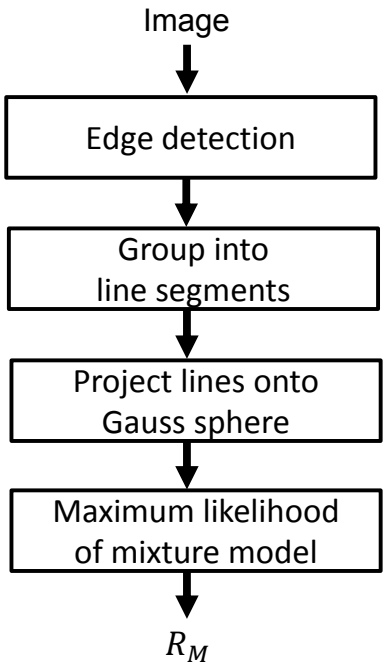


Figure 2.2: Estimating the camera rotation matrix.

The space formed by the normal vectors of all possible interpretation planes is called the *Gauss sphere*. Under perspective projection, the interpretation plane normals of parallel lines in the 3D scene are coplanar, and the normal vector to this plane lies in the direction of these parallel lines. Thus a good choice for the observable E_i used in the likelihood $p(\vec{E}_i|\Psi, m_i)$ is the angular error formed between the interpretation plane of an observed line and the putative 3D orientation vector of the corresponding Manhattan direction.

We learn the values of the priors $p(m_i)$ and the parameters of the distribution of the error functions using a public ground-truth database [18] and optimize the likelihood over the unknonwn Euler angles Ψ using a multi-start version of a BFGS gradient-descent algorithm [25]. Figure 2.2 summarizes the complete pipeline for camera pose estimation. The output of this algorithm is the rotation matrix that defines the pose of the camera with respect to the Manhattan frame, and the inverse R_M of this matrix thus defines the transformation from the camera coordinate system to the Manhattan

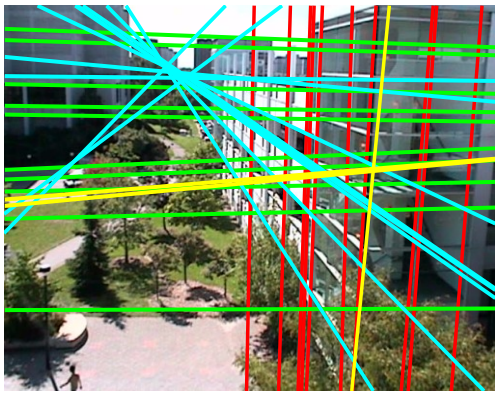


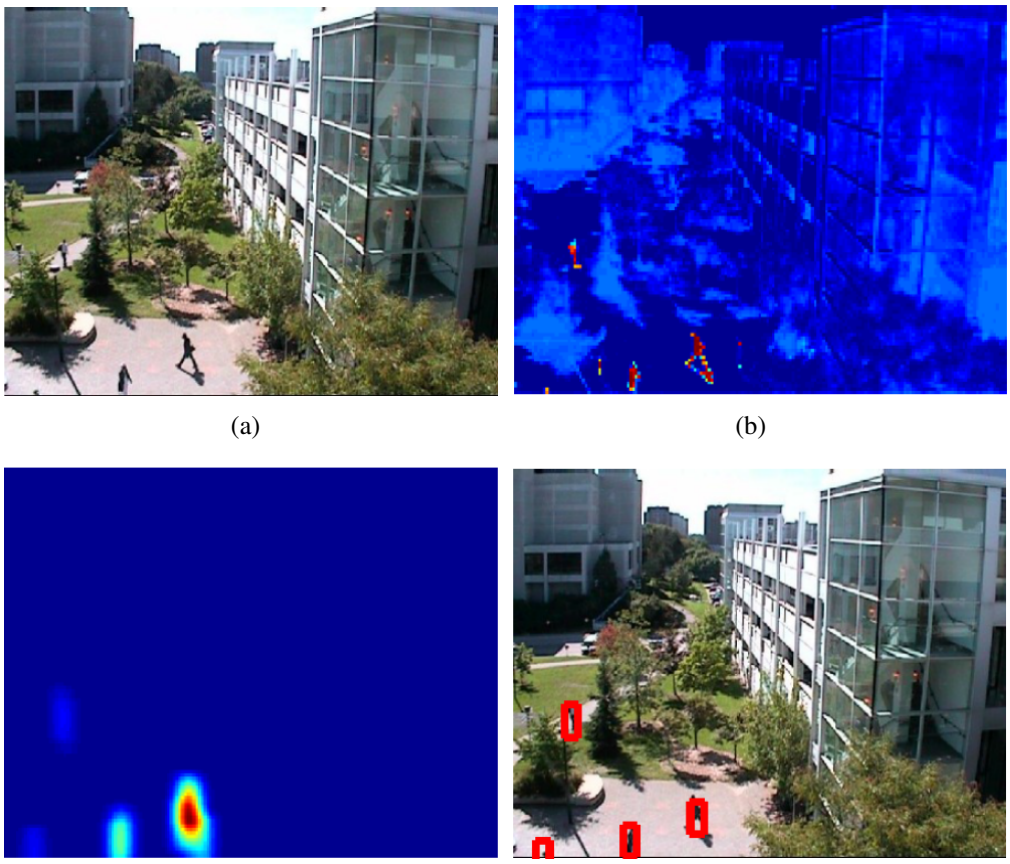
Figure 2.3: Example of Manhattan frame estimation. The cyan and green lines correspond to the two horizontal axes, red lines correspond to the vertical axis, and yellow lines are "non-Manhattan" lines that do not conform to any of the three principal directions.

coordinate system. An example result from the algorithm is shown in Figure 2.3. Estimation of camera pose takes roughly 8 seconds on a standard desktop computer. In practice, we find this is sufficiently fast for typical surveillance cameras, where cameras tend to stay in the same pose for minutes or even hours at a time. In order to improve the robustness of the system to noise and illumination changes, we perform time averaging of the computed rotations [26] over a window of 5 frames.

2.5 Scene Dynamics: Tracking Pedestrians and Vehicles

Detection and tracking of pedestrians and vehicles is based upon a probabilistic background subtraction computation which is described in Appendix A.

In order to track moving objects of interest while ignoring environmental motion (e.g. slow-moving trees), we have developed a simple but efficient and effective tracking algorithm. Let P^n and P^{n-1} represent the foreground posterior probabilities corresponding to the current and previous frames. Then the algorithm consists of 5 computational steps:



(a) (b)

(c) (d)

Figure 2.4: The stages of our tracking algorithm. a).-Input image, b).-Current foreground posterior map, c).-Thresholded, smoothed, normalized absolute difference of current and previous posterior maps, d).-Tracking boxes overlayed on the input image.

1. We compute the absolute two-frame difference $P_d^n = |P^n - P^{n-1}|$.

2. We zero out differences P_d^n below a threshold T

3. We apply a 2D Gaussian low-pass filter to P_d^n. This produces smooth blobs that can be tracked relatively easily (Fig.2.4c).

4. We extract the set of image locations $S^n = \{\vec{\mu}_1, \ldots, \vec{\mu}_{K^n}\}$ corresponding to the maxima of each Gaussian-like blob, where K^n is the number of objects detected in the current image. Note that K^n (and S^n) can change from image to image depending upon the objects that enter or exit the camera's field of view. Our experiments have shown that S^n can be computed effectively using 5-10 iterations of the Mean-Shift algorithm [27], however in practice, we find that a simple peak detection algorithm produces very similar results in a fraction of the time.

5. The actual tracking of the ith object $\vec{\mu}_i \in S^n$ at time n is performed by finding the nearest neighbour $\vec{\mu}_{j^*} \in S^{(n-1)}$ at time $n-1$, which is done by finding its index

$$j^* = \arg\min_j \parallel \vec{\mu}_i - \vec{\mu}_j \parallel, \forall j \in \{1, \ldots, K^{n-1}\}, \tag{2.4}$$

   from which we compute motion vectors $\vec{v}_i = \vec{\mu}_i - \vec{\mu}_{j^*}$ that can be used for temporal smoothing of the tracks and for speed estimation of the moving objects.

Figure 2.4(d) shows an example frame of the algorithm tracking four pedestrians.

2.6 Monitoring of Environmental Signals in Indoor Models

One potential application of 3DTown technology is for monitoring and analyzing energy usage, and how that relates to the flow of people through the city. As an initial step

toward this application, we have integrated within one of the buildings in our 3DTown demonstration a distributed sensing network that monitors the ambient air temperature. To visualize this information, we colour-code rooms of the building according to their temperature, with blues representing cooler temperatures (from 16^oC), and oranges and reds representing warmer temperatures (to 30^oC) - see Fig. 2.5(d).

2.7 Data Integration and Results

We have developed an intuitive web-based graphical user interface that allows the user to select an available real-time surveillance video camera located in the 3D model, activate the 3D visualization of tracked pedestrians and vehicles, change the 3D view of the scene, and query indoor temperature. Estimated 3D pedestrian and vehicle coordinates are stored in an XML file, which is read by the rendering program. Tracked pedestrians are rendered as simple 2D sprites and vehicles and buses as simple 3D models in DAE (Digital Asset Exchange) format that can be loaded and rendered directly in Google Earth. Figure 2.5 show examples of visualizations provided by our web-based user interface.

2.8 Preliminary Evaluation

Prior systems for 3D dynamic visualization of urban scenes [9, 10, 11, 13] have generally not been systematically and quantitatively evaluated, and there is no standard method for such an evaluation at this point. Ultimately, we intend to conduct a usability study based on a human-in-the-loop process within the context of a specific set of tasks. In this work, however, we can report some specific quantitative performance parameters for specific modules of our system, as well as qualitative observations gleaned from demonstrations to hundreds of observers.

(a) (b)

(c) (d)

(e) (f)

Figure 2.5: Example outputs from the 3DTown system. a) and b): Two different views of the 3D visualization of the four tracked pedestrians from Fig. 2.4-d. c) An example of vehicles tracking in the 3D virtual world, d) Example of room temperatures visualization. e) Example of camera pose update view before video camera pose change, f) Automatically-updated view after the video camera pose change.

The accuracy of our 3D building models is on the order of $5cm$. The average error of our automatic camera pose estimation module, measured on a standard public dataset (YorkUrbanDB) is 2.5 deg, which compares favourably with other published single-frame approaches [18].

While our tracking module operates at 8 frames per second on a standard PC, our camera pose algorithm takes about 8 seconds to estimate the camera pose from a single frame: thus it is useful for intermittent pan/tilt operation, but not for continuous smooth pursuit.

We have received informal subjective feedback on the system as a whole from over one hundred observers. The feedback was received from conference attendees during our 3DTown demos, and was based on observing the outputs from our system, both in live and off-line modes, and on our oral presentations. The most frequent observations are:

1. The ability to maintain the 3D dynamic model through pan/tilt shifts of the camera is seen as a very powerful and important capability.

2. Integration of outdoor with indoor models and environmental sensing is perceived as novel, interesting and useful.

3. Delays are sometimes introduced when retrieving model information from Google Earth, particularly when camera pose changes, and these delays are seen as objectionable. This suggests that a different geoserver framework optimized for dynamic visualization is needed.

4. The sprites are too simplistic and should be upgraded to fully articulated avatars.

5. Automatic labeling of actions (e.g., walking, running.) is perceived as a valuable feature for future versions of the system.

6. With only a single live camera, avatars appear and disappear when they enter

and leave the field of view, and this is seen as objectionable. More convincing live demonstration will depend upon a facility with multiple, ideally overlapping fields of view.

2.9 Conclusions and Future Work

In this chapter, we have reported qualitative results from our prototype 3DTown distributed system which allows real-time 3D visualization and analysis of scene dynamics for urban environments within a flexible distributed architecture. To demonstrate the system, we created a 3D model of a university campus integrated with Google Earth terrain data. Three-dimensionalization of data extracted from 2D video cameras is achieved by an algorithm that uses the Manhattan structure of the urban scene to automatically estimate the camera pose. This allows automatically tracked pedestrians and vehicles to be geo-located and thus represented as sprites in the 3D model. Our web-based interface allows the user to browse the 3D model and visualize the scene dynamics of a particular site in real-time while changing the view of the 3D visualization. If the pose of the video camera changes, our system will automatically update the corresponding projection matrix to maintain accurate geo-location of the scene dynamics.

Demonstrations of the 3DTown system for hundreds of people have yielded substantial informal feedback that has been helpful in planning future work, which will include 1) Expanding the system to include more cameras, thus providing complete coverage of an entire building or thoroughfare, 2) Improvements to our tracker, 3) Minimization of rendering delay by migrating the system to an OpenGL platform, currently under development, 4) Introduction of full-articulated avatars, 5) Incorporation of automatic action labelling, and 6) Introduction of a standard method for evaluating the efficacy of the system as whole.

The 3DTown project was an excellent starting point and motivator for the develop-

ment of algorithms and methods that contribute to the automation of visual surveillance systems, especially in the cases where the performance of traffic analytics systems needs to be maintained while the camera pose is allowed to change, which is a common scenario in highway traffic surveillance as we shall see in the next chapter of this dissertation.

Chapter 3

Second Contribution: Automatic Single-View Traffic Camera Calibration from Parallel Curves

An essential step toward the automation of vision-based traffic surveillance systems where the pose of the camera can be changed by the operator is the automatic estimation of the roadway geometry. Roadway estimation requires re-calibration of the surveillance camera, both, intrinsically and extrinsically to enable the bi-directional mapping between imaged objects and 3D world objects and ground plane. A second important requirement is to be able to segment the road lanes and potentially use this information as a prior for detecting and localizing vehicles traveling along their lanes.

In this chapter we present a novel and effective automatic single-view method [6] that: 1) extracts the road lane lines/curves from highway images, and 2) exploits the coplanarity and mutual parallelism between pairs of curves to iteratively estimate both the tilt angle and the focal length of the camera, which together with pre-computed intrinsic parameters, enable the mapping of points from image plane onto the ground plane in the world.

Typical methods for traffic camera calibration from a single view assume the existence of straight parallel lines from which vanishing points can be computed or an orthogonal structure known to exist in the scene. However, there are practical situations where these assumptions do not apply. Moreover, from a single family of parallel lines on the ground plane, there is insufficient information to recover a complete rectification.

In this chapter, we generalize these methods to scenes known to contain parallel curves. We establish an association between pairs of corresponding points lying on the image projection of these curves and compute a least-squares estimate of the focal length and the camera pose from the tangent lines of the associated points, allowing the computation of a planar homography to map image points onto the ground plane, which is needed in Chapter 4 to enable 3D modelling to improve traffic analytics. We evaluate the method on highway and sports track imagery and demonstrate its accuracy relative to a state-of-the-art vanishing point method.

3.1 Introduction and Prior Work

Automatic rectification of imagery to a dominant scene plane is an important subproblem in many applications, including surveillance [28], geodatabases [29], autonomous driving [30], and sports videography [31]. Single-view methods typically rely upon prior knowledge of the features lying on these planar surfaces, such as straight lines or orthogonal structures (e.g., [23, 32, 33, 34, 35]). However these methods fail when orthogonal structures is not dominant in the image or when there are no straight lines from which to extract vanishing points. Here we make the observation that it is not the linearity of the visible features used in vanishing point methods that affords information about the surface attitude, but rather their parallelism. This is important because there are many practical cases where the features are parallel curved lines, e.g., highways, race tracks, railway tracks, industrial conveyor belts etc. In this chapter we

introduce a technique to perform automatic traffic camera calibration in such cases. As a target application, we focus on the problem of rectifying highway images taken from pole-mounted cameras. Rectification in this application is an important step toward accurate estimation of traffic flow and vehicle speed, and the associated estimated camera parameters enable the mapping between image plane and ground plane, which is useful for traffic analytics such as vehicle counting as will be explained in Chapter 4 of this dissertation document.

Traffic surveillance is one of the main applications for automatic camera rectification [32, 36]. Some methods [37, 38, 39, 40] use vehicle motion trajectories in the image to estimate the ground plane orientation; however, these have the disadvantage that recalibration after a PTZ shift may take considerable time if traffic is sparse. The majority of static methods assumes that straight lines or rectangular patterns or textures are available for vanishing point estimation [32, 34, 41, 42, 43].

Prior work has explored concentric circle calibration rigs for multi-view camera calibration [44, 45]. For roadway analysis, Masoud & Papanikolopolous [46] have reported an interactive method for recovering camera parameters that includes the modelling of concentric curves bounding traffic circles. While their work demonstrates the potential for using curves to rectify roadway imagery, their approach was largely manual: the number of curves was assumed known, and control points for each of the curves were provided to the algorithm, sidestepping the difficult problems of feature detection and grouping.

In this chapter we present a much more general and fully-automatic, non-parametric single-image approach to projective rectification of planar scenes containing a system of parallel curves, as arises commonly in highway traffic and sports track video. In spirit, our approach is related to prior work on *elations*, which are projections of (normally rectilinear) coplanar features related by translation in the plane, from which vanishing points and lines can be inferred [19, 47, 48]. In our case, however, the features

are not rectilinear but curved, and they are related not by translation but by systems of dilations (see below).

To train and evaluate our method, we construct a human-labelled dataset of highway camera images in which the parallel curves in the image projecting from lane dividers and highway markers in the scene are identified. Our algorithm proceeds in three stages:

1) *Local feature detection.* Orientation features are detected using local eigenvector analysis, and a classifier is trained to distinguish features lying on curvilinear roadway boundaries from other local features in the scene. 2) *Feature grouping.* A set of probabilistic grouping cues are learned to infer extended curves as connected components of these local orientation features. 3) *Rectification and outlier removal.* Extracted curves are assumed to comprise a subset of inliers that are mutually parallel when back-projected, as well as a subset of non-parallel outliers. To estimate the camera parameters, we form an objective function based upon the average deviation from parallelism between all pairs of inlier curves. Rectification then consists of minimization of this objective, alternating with adjustments in the inlier/outlier assignments.

In summary, the primary contributions presented in this chapter are: 1) A probabilistic method for extracting useful curvilinear features from highways and sports tracks. 2) A novel, effective and fully automatic calibration and rectification algorithm that applies to planar surfaces featuring general parallel curves. 3) Demonstration that the method generalizes, without relearning, to a completely different application domain (a running track).

3.2 Datasets and Geometry

In this section we describe both, the geometry and the image datasets that we used in this project. The main dataset includes highway traffic surveillance images captured by actual pan-tilt-zoom (PTZ) cameras installed at highway poles and overpass bridges.

A secondary evaluation set of images from running tracks (sports) was also used for validation and generalization testing purposes.

3.2.1 Geometry

We assume a projective pan-tilt-zoom (PTZ) camera model where the camera can be expressed by a 3×4 homogeneous camera matrix $P = KR[I \mid -C]$ [19], where $C = [0, 0, D]^T$ is the camera centre, D is the distance of the optical centre of the camera from the ground plane along the optic axis (Fig. 3.2(a)), I is the 3×3 identity matrix, and K and R are the 3×3 instrinsic parameter and rotation matrices respectively, expressed as follows:

$$K = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & \sin(\phi) & -\cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

Here, α_x and α_y are the horizontal and vertical focal lengths in pixels, s is the skew factor, (p_x, p_y) is the principal point, ϕ, θ, and γ are the tilt, pan (or yaw), and roll angles respectively, all in radians.

We consider the problem of rectifying to a ground plane, where the visible features consist of smooth coplanar parallel curves, and let the camera rotate about the ground plane X-axis as shown in (Fig. 3.2(a)). Without loss of generality, we identify the X-axis of the world coordinate system with the x-axis of the camera, so that the y-axis of the camera is the projection of the Y-axis of the world coordinate system (Fig.

3.2(a)), which corresponds to setting $\theta = 0$ and $\gamma = 0$ in eq. (3.2). This assumption is reasonable given that the horizon line in typical highway traffic surveillance images is approximately horizontal as can be seen in Fig. 3.1. We assume zero skew ($s = 0$), square pixels ($\alpha_x = \alpha_y = \alpha$), and we locate the principal point at the centre of the image: $(px, py) = (0, 0)$. Given these assumptions, the matrices K and R reduce to:

$$K = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & \sin(\phi) & -\cos(\phi) \end{bmatrix}. \tag{3.3}$$

Under these conditions, points $[X, Y]^T$ on the ground plane project to points $[x, y]^T$ on the image plane according to $\lambda[x, y, 1]^T = H[X, Y, 1]^T$, where λ is a scaling factor, and the homography H (obtained from P, see [19]) is given by

$$H = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha\cos\phi & 0 \\ 0 & \sin\phi & D \end{bmatrix}. \tag{3.4}$$

Here α is the focal length in pixels, D is the distance of the optical centre of the camera from the ground plane along the optic axis, and ϕ is the tilt angle of the camera relative to the ground plane: $\phi = 0$ when the camera points straight down at the ground surface and increases to $\pi/2$ as the camera tilts up toward the horizon, as it rotates about the ground plane X-axis (Fig. 3.2(a)). Conversely, points in the image can be backprojected to the ground plane using the inverse of this homography, $[X, Y, 1]^T = \lambda H^{-1}[x, y, 1]^T$, where

$$H^{-1} = \begin{bmatrix} \alpha^{-1} & 0 & 0 \\ 0 & (\alpha\cos\phi)^{-1} & 0 \\ 0 & -(\alpha D)^{-1}\tan\phi & \frac{1}{D} \end{bmatrix}. \tag{3.5}$$

Figure 3.1: Example highway images provided by the Ministry of Transportation of Ontario.

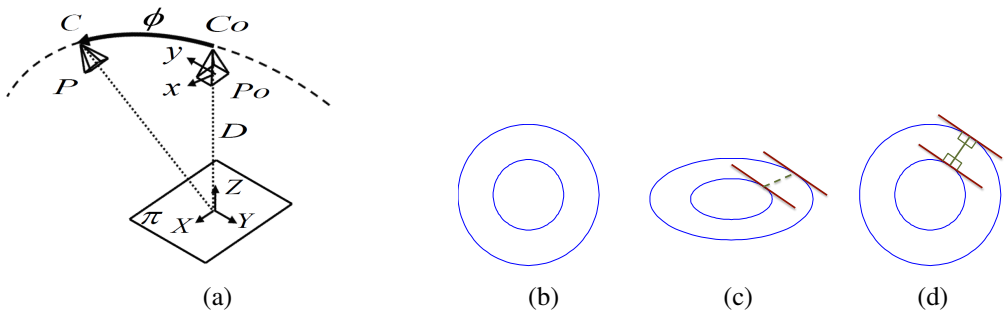


Figure 3.2: (a) Camera setup - see text for details. (b) Parallel curves in the scene plane. (c) Compression in the y-dimension preserves the 1:1 mapping of parallel tangent lines but breaks the constraint that the tangents be orthogonal to the line connecting the points. (d) Only rectification with the correct tilt angle ϕ and the correct focal length α will fully restore parallelism.

Substituting, in Euclidean coordinates this backprojection becomes:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \frac{f_s}{1 - f_p y} \begin{bmatrix} x \\ f_e y \end{bmatrix}, \tag{3.6}$$

where $f_s = \alpha^{-1} D$, $f_p = \alpha^{-1} \tan\phi$ and $f_e = 1/\cos\phi$. Here, f_s is a *scaling factor* that determines the isotropic scaling of the backprojection into metric scene coordinates. f_e is the affine *vertical expansion factor* that determines the extent to which the image is vertically stretched to undo the foreshortening. f_p is the *perspective factor* that reverses the effect of linear perspective, restoring affine properties, e.g., that parallel lines remain parallel.

As a final step, we can apply the homography H of Eqn. (3.4) with a tilt angle of $\phi = 0$ to the scene points $[X, Y]^T$ computed using (3.6), transferring these scene points to image points $[x_r, y_r]^T$ taken by a "bird's eye" virtual camera, yielding a rectified plan view of the ground surface seen from a height D:

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \frac{1}{1 - f_p y} \begin{bmatrix} x \\ f_e y \end{bmatrix}. \tag{3.7}$$

This rectification equation can alternatively be expressed in terms of a homography: $\lambda[x_r, y_r, 1]^T = H_r[x, y, 1]^T$, where

$$H_r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{cos\phi} & 0 \\ 0 & -\alpha^{-1}\tan\phi & 1 \end{bmatrix}. \tag{3.8}$$

Note that based on the geometrical assumptions described above, this homography has two degrees of freedom (i.e. α and ϕ), which are sufficient for transforming the input data in such a way that parallel curves on the ground plane become parallel on a virtual bird's eye camera view as will be explained in Section 3.3.2.

To understand how projection will transform parallel curves visible on the ground surface, we appeal to the definition typically attributed to Leibniz (1692-4) [49]: two curves are considered parallel if one is a constant distance d along the normal from the other. In the language of mathematical morphology, one curve is the erosion or the dilation of the other. (Note that the curves are not related by a translation, and hence the theory of elations [19, 47, 48] does not directly apply.) This means that for every point on the first curve, there is a corresponding point on the second curve such that 1) the line connecting the two points is normal to both curves and 2) the tangent lines through the two points are parallel [50]. Understanding the projection of smooth parallel curves thus entails understanding the projection of these pairs of parallel tangent lines.

A tangent line $\mathbf{L}$ on the ground plane can be represented by the normalized homogeneous vector $\mathbf{L} = [A, B, 1]^T$. The projection $\mathbf{l} = [a, b, 1]^T$ of this line to the image is given by [19] $\mathbf{l}/\lambda = H^{-T}\mathbf{L} \rightarrow \lambda\mathbf{L} = H^T\mathbf{l}$. Substituting (3.4) into this equation allows us to express the tangent line $[A, B, 1]^T$ on the ground plane in terms of its projection

$(a, b, 1)^T$ on the image plane:

$$\lambda \begin{bmatrix} A \\ B \\ 1 \end{bmatrix} = \begin{bmatrix} a\alpha \\ b\alpha \cos\phi + \sin\phi \\ D \end{bmatrix}. \tag{3.9}$$

Now consider two tangent lines $\mathbf{L}$ and $\mathbf{L}'$ from corresponding points on two parallel curves. To be parallel, the coordinates of the two lines must satisfy the relation $A'/A = B'/B$. Substituting from (3.9), we have:

$$\frac{a'}{a} = \frac{b'\alpha \cos\phi + \sin\phi}{b\alpha \cos\phi + \sin\phi}, \tag{3.10}$$

and rearranging, we obtain:

$$f_p = \alpha^{-1} \tan\phi = \frac{ab' - a'b}{a' - a}. \tag{3.11}$$

Thus we observe that the perspective factor f_p can be computed directly from the image coordinates of the two tangent lines projecting from corresponding points on the parallel ground plane curves. From Eqn. (3.6) it can be seen that this is sufficient information to restore the scene curves to their parallel state. However, on its own, Eqn. (3.11) is insufficient to uniquely determine the tilt angle ϕ and focal length α. In particular, there remains a one-dimensional family of solutions corresponding to the unknown vertical expansion factor f_e.

In principle, it is possible to estimate the internal parameters of a pan-tilt camera, including the focal length α, using point correspondences from a set of images taken with different pan-tilt settings [51]. However, for applications such as traffic surveillance, requiring a series of large pan-tilt shifts to recalibrate every time the focal length changes is undesirable, as it may interrupt the normal control protocol and real-time video analytics.

Fortunately, if the curves are not straight we have not exhausted the information available from a single image. Recall that for the curves to be parallel, not only must the corresponding tangent lines be parallel, they must also be orthogonal to the line connecting the corresponding points (Fig. 3.2(b-d)). In particular, letting $(X, Y)^T$ and $(X', Y')^T$ be the Euclidean representation of the two corresponding tangent points, we must have $[X' - X, Y' - Y]\,[B, -A]^T = 0$. Substituting from Eqns (3.6) and (3.9), and simplifying, we obtain:

$$cos^2\phi = \frac{\delta_y a}{\delta_x\,(b + f_p)}, \tag{3.12}$$

where $\delta_x = w'x' - wx$, $\delta_y = w'y' - wy$, $w = (1 - f_p y)^{-1}$ and $w' = (1 - f_p y')^{-1}$. With the constraint that $0 \leq \phi \leq \pi/2$ (Fig. 3.2), Eqn. 3.12 uniquely determines the tilt angle ϕ, allowing the focal length α to be computed directly from Eqn. (3.11).

An example may make this computation clearer. Suppose that the two curves are concentric (parallel) circles in the scene plane (Fig. 3.2(a)). On projection, these circles appear as ellipses compressed along the y axis in the image, and these ellipses are *not* parallel (Fig. 3.2(b)) since the lines connecting pairs of points with parallel tangents are not normal to the curves. The curves will remain non-parallel ellipses even after correction for the perspective factor f_p, due to the uncorrected expansion factor f_e. The only solution to the rectification problem that will make all tangent pairs parallel and orthogonal to the line connecting them must use the correct tilt angle ϕ and the correct focal length α to correct for both perspective f_p and expansion f_e. In general, the parallel curves may be much more complex, but the same principle applies.

Thus the presence of parallel curvature on the ground plane represents an opportunity for more complete rectification. The flip side is that as the curvature decreases and the curves become straight, the problem becomes ill-posed. Although the image can still be rectified up to the vertical compression factor, the estimates for focal length $\hat{\alpha}$ and tilt $\hat{\phi}$ will generally be unreliable.

Without some metric knowledge, the scaling factor f_s must remain unknown. However, for many applications (e.g., highway surveillance), camera height is known, so that given both tilt angle ϕ and focal length α, knowledge of the sensor dimensions (pixel pitch) will in principle suffice for metric estimation on the ground plane. We leave this as future work.

Due to noise, a single pair of corresponding tangents will in practice be insufficient to render an accurate rectification. Instead, we seek a least-squares solution over a large number of corresponding tangent pairs over multiple parallel curves. We turn to this problem now.

3.2.2 Highway Dataset

The first and main dataset used in this chapter was created based on a set of long videos provided by the Ministry of Transportation of Ontario (`http://www.mto.gov.on.ca/english/`). The 480×640 pixel resolution color images were captured by traffic surveillance pant-tilt-zoom (PTZ) cameras installed at different highway intersections as shown in Fig. 3.1. This dataset is challenging: 1) The cameras are uncalibrated. No information about the cameras was provided to us, 2) The images are originally analog, then digitized and compressed, 3) Pan, tilt and zoom vary widely, 4) Quality of the road markings varies widely, 5) Light conditions vary widely.

We randomly partitioned 20 highway videos into training and test datasets of 10 videos each. From each video we extracted and hand-labeled 10 images, sampled sparsely in time. For each of these sampled images, a binary mask was produced, which contains the binary labels of the pixels belonging to road curves and markings (foreground) and backround, which were used to train a binary classifier as described later in this chapter.

3.2.3 Running Track Dataset

We created a second, very short image dataset using our own camera to capture images of a running track that contain coplanar parallel curves. The main goals of this dataset were: 1) to validate the algorithms presented in this chapter using a calibrated camera (intrinsic and extrinsic parameters), and 2) to evaluate how the algorithms trained with highway data generalize with images from a different domain (sports). Here 712×1072 pixel resolution color images were captured with a Nikon D90 camera, which was mounted on a tripod at a height of 3.74 m from the ground, and the camera roll was minimized (i.e. approx. zero roll) using a level. A total of 7 images of the running track were captured. These images were taken from a variety of locations in the stadium stands, at tilt angles ϕ of 60, 65 and 70 deg. One example of these images is shown in Fig. 3.10. The camera was calibrated (intrinsic parameters) in the lab using a standard method [52]: the focal length was estimated at $\alpha = 812$ pixels, and the principal point was estimated at $(px, py) = (4, 8)$ with respect to the centre of the image. These parameter values were used in subsection 3.4.3.

3.3 Algorithm

Given a single image I and a scene plane containing parallel curves, we wish to estimate the camera parameters ϕ and α in order to rectify the image data by means of the homography H_r from Eqn.(3.8). This estimate will be based on maximizing the parallelism of rectified image curves. To compute this objective, we need to detect and group local features into extended curves, and associate pairs of curves hypothesized to be parallel.

For the sake of concreteness, we will focus here on highway imagery. To optimize accuracy and make our assumptions explicit, we adopted a probabilistic supervised learning approach, randomly partitioning 20 640×480 highway videos captured by

various highway cameras into training and test datasets of 10 videos each. From each video we extracted and hand-labeled 10 images, sampled sparsely in time. A risk here is that we will over-learn the statistics specific to this dataset. To assess this, we will also apply the method, without relearning, to a completely different application domain (sports videography).

Our automatic rectification algorithm has two main stages: 1) curve extraction, and 2) rectification.

3.3.1 Curve Extraction

Curve extraction consists of local feature detection and grouping. The parameters for both stages are learned from manually labeled lane marks and lane dividers in our highway dataset.

Local feature detection. We use a standard corner detector [53] to extract image features. At each image location i, we construct the 2×2 matrix C_i and compute its eigenvectors $\mathbf{e}_i^1$ and $\mathbf{e}_i^2$ and associated eigenvalues λ_i^1 and λ_i^2. For a smooth curve, the eigenvectors $\mathbf{e}_i^1$ and $\mathbf{e}_i^2$ encode the normal and tangent vectors, respectively. We define an appearance vector $\mathbf{d}_i = [\eta_i, \lambda_i^1, b_i]^T$ where $\eta_i = \lambda_i^2/\lambda_i^1$ is the ratio of the eigenvalues and b_i is the pixel brightness.

We use this appearance vector to determine whether a feature lies on ($\omega_i = 1$) or off ($\omega_i = 0$) a smooth curve on the scene plane. Assuming conditional independence of the appearance features, the likelihoods are approximated as the product of the marginals (Fig. 3.3), and the likelihood ratio L can be written as

$$L_i = \frac{p(\lambda_i^1|\omega_i = 1)p(b_i|\omega_i = 1)p(\eta_i|\omega_i = 1)}{p(\lambda_i^1|\omega_i = 0)p(b_i|\omega_i = 0)p(\eta_i|\omega_i = 0)}. \tag{3.13}$$

After discarding features for which $\log L_i < 0$, we thin the features using non-maximum suppression in the $\mathbf{e}_i^1$ direction, normal to the curve. The result is a sparse set of local

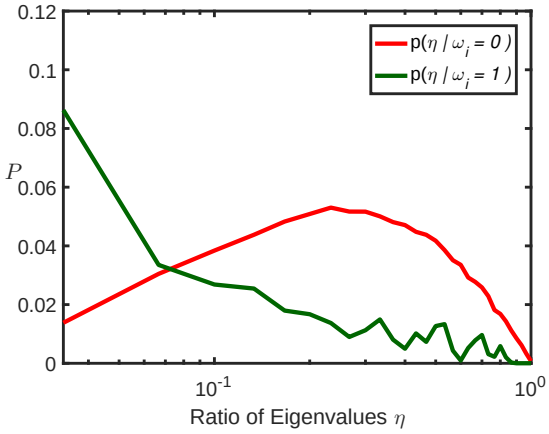
features (Fig. 3.4 (a)).

Feature grouping. In spirit, our rectification method is local, as it relies only upon pairs of local tangent vectors. In practice, however, grouping the local features into global curves is important, for two reasons. First, for some applications, the local features are not aligned. For example, in the highway imagery shown in Fig. 3.4(c-d), the lane marks are offset, so that a normal from one will not necessarily intersect another. If we first group these lane marks into curves, these intersections can be determined by interpolation.

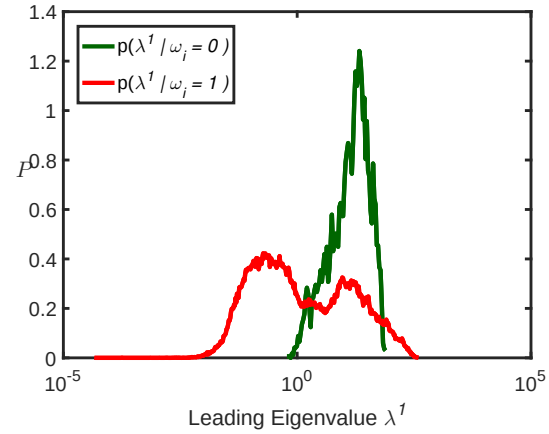
A second benefit of grouping is outlier removal. In practice, only some of the local features will lie on parallel curves from the dominant scene plane, and identifying outliers is an important part of making the method work. Outlier identification is greatly facilitated by first grouping the local features into curves, as normally a curve will be either wholly an inlier or wholly an outlier.

We found that a simple grouping method was sufficient for this application. From the training dataset, we learn the minimum-area $L \times W$ rectangular search window (Fig. 3.4(b)) that, when based at each ON curve feature, is guaranteed to include at least one other feature from the same curve, which helps determine the maximum angle θ_0 between the leading eigenvectors of these two features. (Learned parameters: $L = 10$ pixels, $W = 5$ pixels, $\theta_0 = 39$ deg.) The grouping algorithm then proceeds in three stages: 1) We instantiate a graph $G(V, E)$, where vertices V represent the local features and the edge set E is initially empty. 2) We tour the graph, searching the window based at each local feature and adding an edge to any other vertex representing a feature within the search window and satisfying the maximum angle constraint. 3) We extract curves $\mathbf{C} = \{C_1, C_2, \ldots, C_M\}$ as connected components of G.

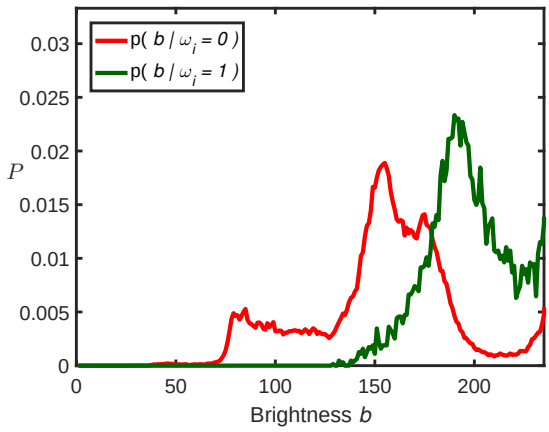
Figure 3.4(c) shows the resulting curves for an example highway image. Note that the lane marks are segmented as individual, short curves. In order to group these into global curves, we repeat the same procedure using a larger search window, again



(a)

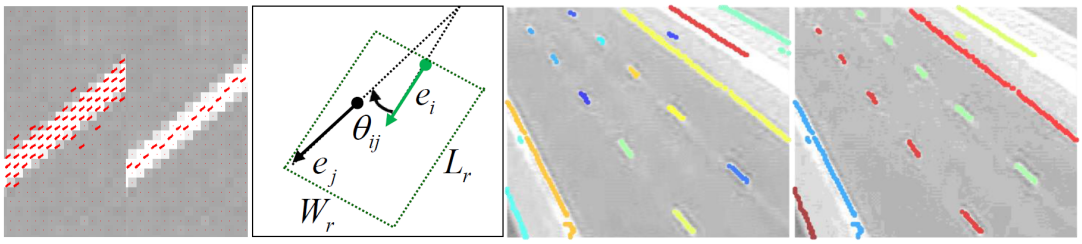


(b)



(c)

Figure 3.3: Likelihood distributions for local features.



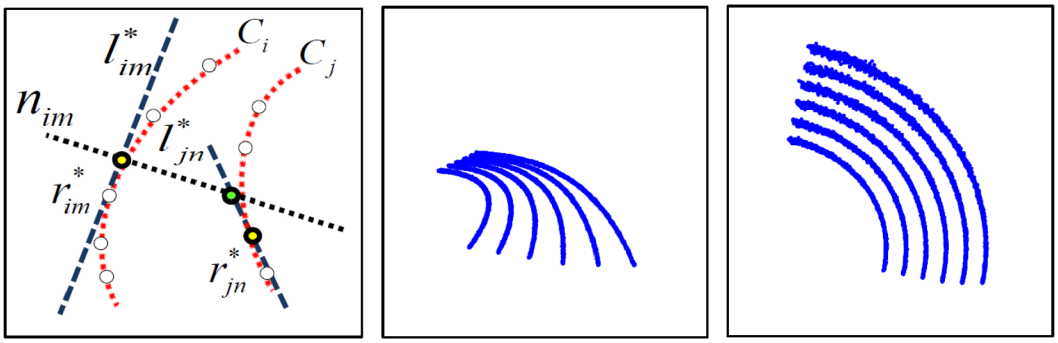
(a) (b) (c) (d)

Figure 3.4: Curve extraction. (a) Local features before and after thinning. Each vector indicates the direction of the leading eigenvector. (b) Processing window used in the curve segmentation process, (c) First-level segmentation, (d) Second-level segmentation. Each color represents a subset of grouped feature vectors.

learned (independently) from the training data (Learned parameters: $L = 145$ pixels, $W = 25$ pixels, $\theta_0 = 20$ deg.). Figure 3.4(d) shows the resulting global curves for the same highway image.

3.3.2 Rectification: Iterative Estimation of Tilt Angle and Focal Length

The input to our rectification algorithm is a set $\mathbf{C}$ of curves, each consisting of a set of local features $f_i = (\mathbf{r}_i, \mathbf{l}_i)$ represented by their location $\mathbf{r}_i = (x_i, y_i, 1)^T$ in the image and tangent line $\mathbf{l}_i = (a_i, b_i, 1)^T$, in homogeneous form. Our goal is to estimate the camera parameters ϕ and α, and therefore the homography H_r, which can be used to rectify the input image (Eqn.(3.8)). Each iteration t of the algorithm consists of four main steps: 1) Transformation of local features using the current estimate H_{rt} of the homography, 2) Pairwise association of local features on parallel curves, 3) Re-estimation of the homography, and 4) Outlier removal. The steps are interdependent: feature association and outlier removal depend upon the estimated homography, and the estimated homography depends upon feature association and outlier removal. In this sense, our rectification algorithm can be considered a generalization of the iterative



(a) (b) (c)

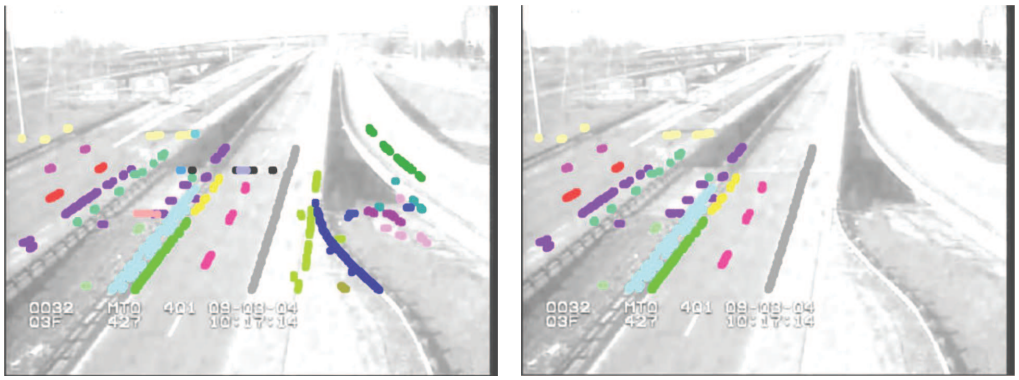
Figure 3.5: Tangents association (a)- Association of feature vectors from different curves - see text for details. (b)- Example synthetic input with localization noise $\sigma_r = 1$ pixel and no angle noise. (c)- Rectified output. Note the noise amplification along the rectified curves. .

closest point method [54, 55].

Transformation. The current estimate of the homography H_{rt} is applied to the local feature map to yield approximately rectified features: $\mathbf{p}^*_i = H_{rt}\mathbf{p}_i$, $\mathbf{l}^*_i = H^{-T}_{rt}\mathbf{l}_i$.

Association. The goal of this step is to associate each local feature f_{im} on a curve C_i with one other local feature f_{jn} on each of the other curves $C_j, j \neq i$ in the image. Roughly speaking, we wish to select the feature f_{jn} that lies nearest the normal line for f_{im} (Fig. 3.5 (a)). In practice, we measure this distance along the tangent line for f_{jn}, which takes into account the curvature of C_j. If no feature on C_j lies within an association tolerance of $T_a = 41$ pixels (learned from training images) of the normal line for f_{im}, no association is made. The outcome of this process is, for each pair of curves $(C_i, C_j), j \neq i$ in the image, a bipartite matching between a number K_{ij} of local features on the two curves. The critical property of each match is the angular deviation $\theta_{ijk}, k \in [1, \ldots, K_{ij}]$ between the associated tangent lines.

Re-estimation. Were the homography H_{rt} correct, and in the absence of noise, all of the associated features would be parallel. Thus to estimate the focal length α and tilt angle ϕ we minimize the sum of squared angular deviations θ_{ijk} over all corresponding tangent lines and all pairs of curves in the image using a standard iterative nonlinear



(a) (b)

Figure 3.6: Outlier removal Example highway feature vectors before (a) and after (b) outlier removal. .

optimization method. To initialize the optimization we evaluated two methods: 1) Coarse grid search and 2) Mean of the parameters estimation from the training set ($\bar{\phi} = 70$ deg, $\bar{\alpha} = 788$ pixels). Both methods proved equally accurate, but of course, using a single, well-chosen initial estimate is faster.

Outlier removal. The re-estimation process can fail if there are many outlier curves. In this work, an outlier curve is a curve that is not approximately parallel to the rest of the curves in the image. To detect and remove outliers, we define a simple, yet effective measure of total deviation ϵ_i for each of the M curves C_i in the image:

$$\epsilon_i = \frac{1}{M-1} \sum_{j \neq i} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} |\theta_{ijk}| \,. \tag{3.14}$$

Curves C_i for which ϵ_i exceeds a threshold of 30 deg are discarded, and the total deviation ϵ_i is recomputed for the remaining curves. This process is repeated until all curves lie within the threshold. We find that this method works reasonably well in practice (Fig. 3.6).

3.4 Results

We evaluated our algorithm on three datasets: 1) Synthetic data consisting of parallel curves with added noise and known ground truth, 2) Highway images taken by a variety of uncalibrated highway cameras (unknown α and ϕ), and 3) Images of a curved running track taken by a calibrated camera (known α and ϕ). Dataset 3 was acquired with a Nikon D90 camera, calibrated using a standard method [52]: the focal length was estimated at $\alpha = 812$ pixels. We used this camera model when generating the synthetic Dataset 1.

3.4.1 Experiment 1. Synthetic Data

For our first experiment, we assumed a viewing distance $D = 40$m and a 640×480 pixel image, and simulated 6 concentric circular arcs centred at the origin of the scene plane, with equally-spaced radii ranging from 30 to 55 m, and angular subtense of 90 deg. The rotation of the arcs around the origin of the scene plane was randomized over samples. Each of these arcs was projected analytically to the image using our camera model, and then represented by a field of feature vectors localized to sub-pixel accuracy with an arc length spacing of 1 pixel. We used this dataset to verify the method and assess sensitivity to additive Gaussian iid noise in in both the position and angle of the local features, as a function of tilt angle. Fig.3.5 (b-c) shows an example stimulus before and after rectification.

Fig. 3.7 shows results for noise (Figs. 3.7(a-b)) and tilts (Figs. 3.7(c-d)). The method produced estimates of tilt angle $\hat{\phi}$ and focal length $\hat{\alpha}$ that are unbiased and accurate. However, with high levels of noise and/or small tilt angles, the method becomes biased: both tilt angle and focal length are underestimated.

We believe that this bias is due to amplification of noise in the objective function induced by rectification. From Eqn. 3.7, one can see that the perspective factor f_p will

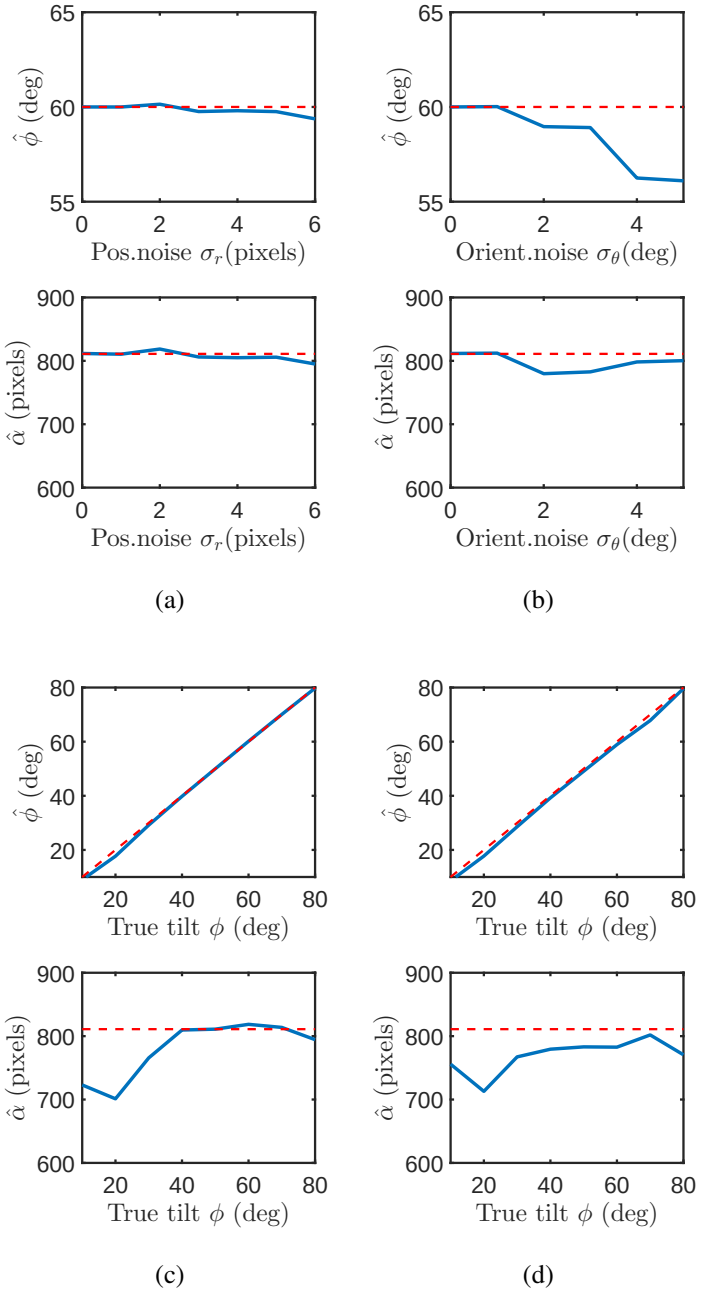


Figure 3.7: Experiment 1 results. Top plots show estimated tilt angle $\hat{\phi}$, bottom plots show estimated focal length $\hat{\alpha}$. Red dashed curves indicate ground truth values, blue curves indicate estimated values. (a) Variation with std. dev. of position noise σ_r. (b) Variation with std. dev. of angle noise σ_θ. (c) Variation with tilt angle ϕ for position noise of $\sigma_r = 1$ pixel. (d) Variation with tilt angle ϕ for angle noise of $\sigma_\theta = 3$ deg.

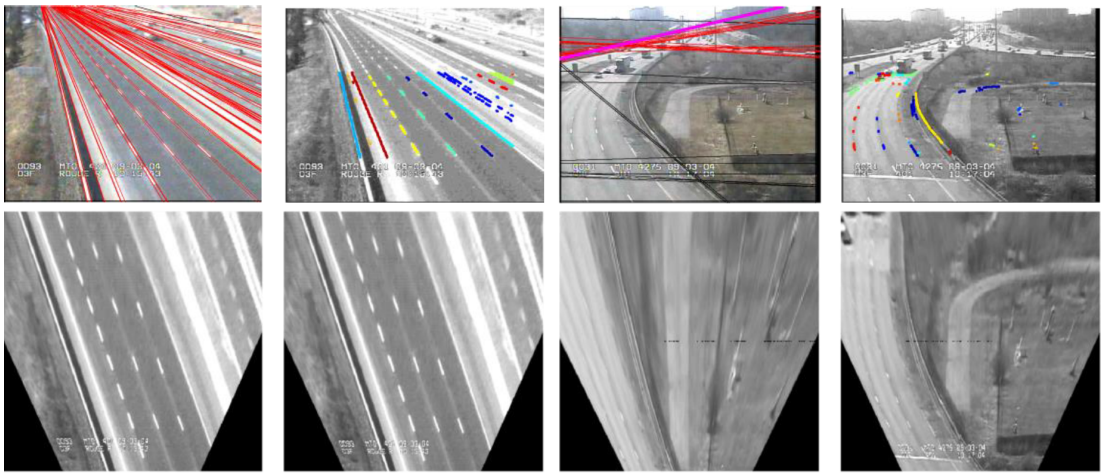
cause noise to be attenuated or amplified depending upon the sign of y. However, noise in the rectified y coordinate y_r will increase monotonically with the vertical expansion factor f_e, inducing a bias to smaller values, and hence smaller tilt angles ϕ. Since tilt angle and focal length α are inversely coupled through the perspective factor $f_p = \alpha^{-1} \tan \phi$, this induces a compensating decrease in the estimate of focal length α. This effect becomes more pronounced for smaller tilt angles, where the perspective distortion does not impose as strong a constraint on the rectification. In future work, we hope to correct for this high-noise bias by explicitly modelling the propagation of noise across rectification.

3.4.2 Experiment 2. Highways

For our second experiment, we applied our algorithm to the highway test dataset, which is quite challenging: 1) The images are originally analog, then digitized and compressed, 2) Pan, tilt and zoom vary widely (see Fig. 3.1), 3) Quality of the road markings varies widely, 4) Weather and light conditions vary widely. This dataset thus forms a realistic test of the algorithm's potential.

We compare our curvilinear method against a state-of-the-art method for linear vanishing point extraction [23]. This method extracts the vanishing points from the detected families of imaged parallel lines assumed to lie on the ground plane and the resulting horizon line, which can be used to estimate the projective factor f_p. We emphasize that while the linear method should work well for straight highways, it is not expected to work well for curved highways.

Since we do not have ground truth here, we are somewhat constrained with respect to quantitative evaluation. However, two forms of evaluation are still possible. 1) We can qualitatively evaluate whether rectification correctly parallelizes the curves. 2) We can compare parameter estimates for the fully automatic algorithm against parameters estimated with our curvilinear rectification algorithm but given hand-labelled curves as



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

Figure 3.8: Highway rectification examples. Top row: estimated features, bottom row: resulting rectified image. (a-b) Linear [23] and proposed curvilinear method applied to straight road. Both methods perform well. (c-d) Linear [23] and proposed curvilinear method applied to curved road. While the linear method fails completely, the curvilinear method computes a reasonable rectification.

input. While this second method serves to estimate errors induced by the segmentation and grouping stages, we emphasize that it does not evaluate errors introduced in the rectification stage.

We divided the test dataset into a subset of straight highway segments, and a subset where the highway is curved. For the straight subset, simultaneous estimation of both tilt angle ϕ and focal length α is under-constrained. To evaluate the methods, we therefore estimate the perspective factor f_p and use a nominal value for the vertical expansion factor of $f_e = 1$ in Eqns. 3.6 and 3.7 in order to rectify the imagery.

Fig. 3.8(a-b) shows example rectifications for linear and curvilinear methods. While results appear satisfactory in both cases, a paired t-test [56] comparing the error in the estimated perspective factor f_p for 30 different images reveals that the curvilinear method is significantly more accurate, $t(69) = 4.4, p = .00004$.

Fig. 3.8(c-d) shows example rectifications of a curved highway for both linear and

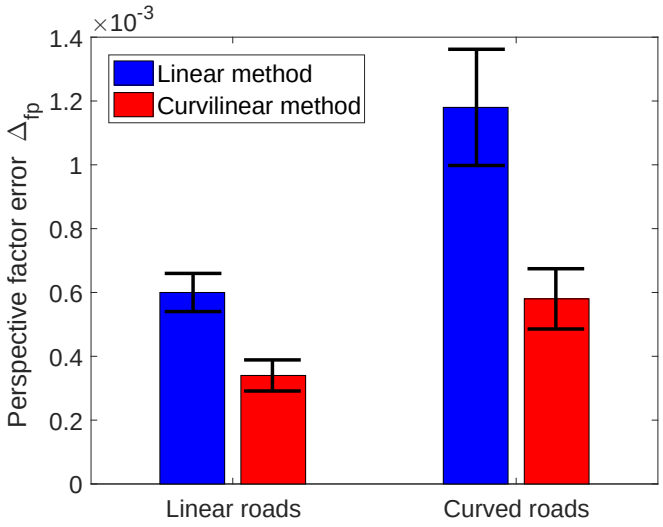


Figure 3.9: Experiment 2 results (highway images): comparison of error in the estimated perspective factor f_p using the linear and curvilinear methods. Errors were computed with respect to ground-truth estimated with hand-labeled data.

curvilinear methods. In this case, the linear method fails catastrophically, while the curvilinear method succeeds in computing a reasonable rectification. Quantitatively over the curved roads in our dataset, the curvilinear method again performs significantly better, $t(29) = 2.5, p = .02$, (Fig. 3.9). Mean accuracy of the estimated parameters for the curvilinear method applied to curved highways in our test dataset is shown in Table 3.1. We emphasize that these errors are relative to *estimated* ground truth, computed using our rectification algorithm on hand-labeled data. Average running time of our non-optimized Matlab implementation on a standard dual-core laptop computer was 15.3 sec for curve extraction, and 41.1 seconds for rectification (30 iterations). We expect run time could easily be improved by an order of magnitude with some optimization.

3.4.3 Experiment 3. Running Track

For our final experiment, we used a calibrated camera to acquire 7 images of a running track. These images were taken from a variety of locations in the stadium stands, at

Table 3.1: Mean absolute errors for estimated tilt and focal length for curvilinear method. Road results are for curved highways only and are relative to *estimated* ground truth, using our rectification algorithm on hand-segmented data. Track results are based on actual ground truth.

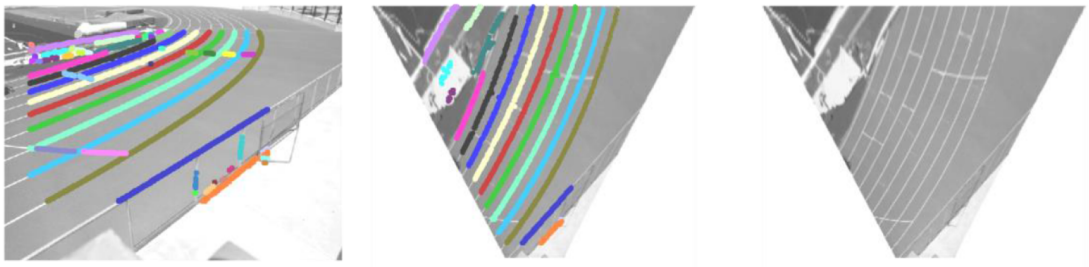
	Mean tilt err Δ_ϕ		**Mean focal length err Δ_α**	
	deg	%	pixels	%
Roads	2.8	4.0	13.7	1.7
Track	1.2	1.8	14.4	1.8

tilt angles of 60, 65 and 70 deg. We applied the same algorithm used for the highway images in Experiment 2, except for omitting the second grouping step, since the curves of the running track are continuous. Importantly, we did not relearn the statistical parameters for feature detection and grouping. Given the substantial differences in the quality of the imagery and the application domain, Experiment 3 provides a good test of how well the method generalizes.

An example result is shown in Fig. 3.10, where (a) shows the results of curve segmentation, and (b) and (c) show the results after outlier removal and rectification. Here we have ground truth tilt and focal length, allowing us to evaluate the accuracy of the method in absolute terms. Table 3.1 shows these results. Mean absolute tilt angle error Δ_ϕ was 1.2 deg (1.8%) and mean absolute focal length error Δ_α was 14.4 pixels (1.8%). We believe that the improved accuracy relative to the highway dataset is primarily due to the superior image quality.

3.5 Conclusions

Methods for single-view traffic camera calibration and image rectification generally assume systems of parallel lines or an orthogonal structure in the scene. In this chapter, we have introduced a method that applies to scene planes that may not contain linear structure, but do contain parallel curves. We have shown that these curves provide sufficient information for the estimation of both focal length and tilt angle. Our



(a) (b) (c)

Figure 3.10: Experiment 3 results. (a) Output from curve segmentation stage prior to outlier removal. (b) Rectified image with inlier curves. (c) Rectified image.

experiments have demonstrated the efficacy of the method, particularly in situations where conventional methods fail.

Although highway surveillance is our main target application, we have shown that the method generalizes well to sports tracks without relearning. In fact, the proposed method was evaluated recently by Elassal & Elder [57] in different scenarios, where the reported absolute tilt angle error was 0.55 deg for highway images, and 1.45 deg for an indoor pedestrian dataset. Other potential application domains include autonomous driving, curved conveyor systems for industrial automation and non-contact fingerprint analysis.

Future work may address 1) the analysis of small biases induced by nonlinear propagations of error, 2) estimation of additional camera parameters (e.g., camera roll), 3) estimation of metric properties based upon prior knowledge of sensor placement or object size and 4) generalizations to scenes with multiple systems of parallel curves, e.g., highway interchanges. 4) evaluate RANSAC [58] outlier rejection and explore whether performance for RANSAC would justify the added complexity.

The probabilistic curve extraction method and the iterative estimation of the homography parameters (α and ϕ) can be coupled with other computer vision techniques to enable automatic visual traffic analytics applications such as vehicle counting, as will be explained in Chapter 4.

Chapter 4

Third Contribution: Slot Cars: 3D Modelling for Improved Visual Traffic Analytics

The method introduced in Chapter 3 enables us to automatically extract road lines and curves from highway images and compute estimates of the camera tilt angle and focal length (if the road has curvature). In this chapter, we couple these estimates with a foreground extraction-based traffic object detection method (described in Appendix A), to build an end-to-end vision-based traffic analytics prototype system that can count vehicles in the scene.

A major challenge in visual highway traffic analytics is to disaggregate individual vehicles from clusters formed in dense traffic conditions as illustrated in Fig. 4.1. In this chapter, we introduce a data-driven 3D generative reasoning method to tackle this segmentation problem. The proposed method is comprised of offline (learning) and online (inference) stages. In the offline stage, given camera intrinsic parameters and height, we use the parallelism method described in Chapter 3 to estimate highway lane structure and camera tilt to project 3D models to the image. In the online stage,

Figure 4.1: Example vehicle clusters formed in dense traffic conditions (Top row), and corresponding results from a 2D segmentation algorithm (Bottom row).

foreground vehicle cluster 2D segments are extracted using the motion and background subtraction method described in Appendix A. For each 2D segment, we use a data-driven MCMC method to estimate the vehicles' configuration and dimensions that provide the most likely account of the observed foreground pixels. We evaluate the method on two highway datasets and demonstrate a substantial improvement on the state of the art.

4.1 Introduction and Prior Work

In many traffic surveillance installations, camera placement is oblique. As a consequence, vehicles project to the image in clusters and are often only partially visible due to occlusion. Disaggregating these clusters into individual vehicles is central to attaining accurate vehicle counts. Tracking and appearance cues are highly fallible, as vehicles may move at similar speeds and have similar colours (Fig. 4.1). Size cues are also tricky, as vehicle size can vary by an order of magnitude, from motorcycles to tractor-trailers.

Attempts have been made to solve this problem using 2D spatiotemporal constraints. One idea is to use variations in velocity within a segment or variations in

the shape of an image segment over time to identify the multiple vehicles within a cluster [59, 60, 61]. Unfortunately, since highway speeds are highly regulated, velocity is not a very effective segmentation cue for highway traffic: it is very common for neighbouring vehicles to be travelling at the same speed and in the same direction. An image segment projecting from a single vehicle may also vary in shape over time due to shadows, specularities and projective distortions, making this a weak cue as well. Bouvié et al. [59, 60] attempt to strengthen this method with the constraint that local image features projecting from each vehicle form a convex group in the image. However, this approach is also limited, since an individual vehicle could project a non-convex group, while a cluster of vehicles could project a convex group.

To overcome these limitations, we propose a 3D approach. 3D model-based methods for object verification (motorcycles, horses) [62] and for make/model vehicle recognition [63, 64] have recently proven effective on datasets where the objects are fairly isolated and already localized in the image. However, these methods do not address the thorny problem of detecting and individuating multiple mutually occluding objects in highly cluttered scenes, which is the problem we must solve to achieve accurate traffic analytics in rush-hour conditions.

The 3D method from [29] fuses geographic data (streets and static elements) from a specific urban environment with thousands of internet geotagged images of the same scene that are processed using Structure from Motion [65, 66], and refined using publicly available LIDAR point clouds of the urban environment. The resulting 3D model is used to generate labels for 3D vehicle detection. Although effective, the method is relatively complicated in the sense that it requires multiple sensors and images of the same location, and is limited to operate on specific urban areas for which LIDAR and geographic information is available.

To address these problems, we take as inspiration the earlier work of Song & Nevatia [67]. Their insight was that 3D models of common vehicles combined with knowl-

edge of intrinsic and extrinsic camera parameters could be used to reason about the 3D configuration of vehicles most likely to account for observed clusters in the image. This is a powerful approach, and has the advantage that vehicle categorization and traffic volume measurement can potentially be solved simultaneously.

However, there are two main limitations of this 3D model approach that we address in this dissertation. The first is how the 3D models are defined. The Song & Nevatia algorithm employed three categories of vehicle (sedan, SUV, truck), and assumed they occur with equal probability. This is clearly unrealistic for highway traffic, where diverse vehicle categories are possible. For example, in our datasets, we have enumerated 13 different semantic vehicle categories and found that the prior distribution is far from uniform (Fig. 4.2). To address this challenge, we propose an automatic clustering approach, optimizing the number of clusters to maximize the accuracy of traffic volume measurements.

The second limitation is the complexity of the configuration space. In the Song & Nevatia approach, the unknowns included the number, category, location and orientation of the vehicles in the scene. The combinatorial complexity of this configuration space led them to propose a rather complex coarse-to-fine search, terminating in a fine-grained MCMC stage. To address this problem, we take advantage of recent methods for automatically recovering the lane structure of the highway [6] (Chapter 3). Existing lane detection methods such as [68] extract curves from bird's eye view images using a homography computed from manually-defined control points, and use spline curve models and probabilistic frameworks to detect lanes. Here we use the method presented in Chapter 3, which estimates the homography parameters automatically, and couple it with a baground subtraction method to automatically extract active road lanes. This allows us to fix the pose of each 3D vehicle proposal, and to limit the search over location to a 1D space. As a result, a single MCMC search stage is sufficient to recover optimal configurations. We refer to our method as 'Slot Cars' because the

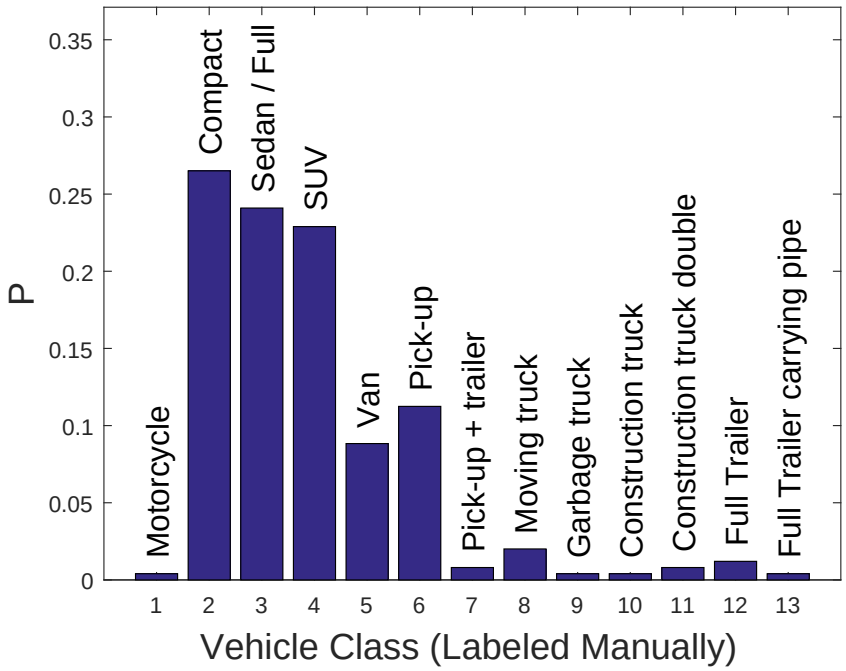


Figure 4.2: Distribution of vehicle classes for our training dataset.

Figure 4.3: Example 3D cuboid model.

MCMC algorithm slides the 3D models one-dimensionally along each lane to identify the most probable configuration.

While Song & Nevatia employed complex 3D CAD models for their vehicles, we elect to employ simpler cuboid models (Fig. 4.3) that have been used effectively in recent work on camera calibration [69], vehicle detection [70] and fined-grained vehicle recognition [71]. In addition, while Song & Nevatia relied on an orthographic projection approximation, we assume full perspective projection, as it adds negligable complexity and should provide more accurate results. The most important difference in our method with respect to Song & Nevatia, is the coupling of our method from Chapter 3 with 3D modelling to improve visual traffic analytics.

4.2 Datasets and Geometry

We recorded two highway traffic datasets at different highways and on different days. Both datasets were recorded with a Sony Nexus 6 camera at 1440×1080 pixel resolution and 30fps. For Dataset 1, we labeled (2D bounding boxes and IDs) 1,072 frames ($\sim$ 36 sec), and for Dataset 2, we labeled 494 frames ($\sim$ 16 sec). We employed the

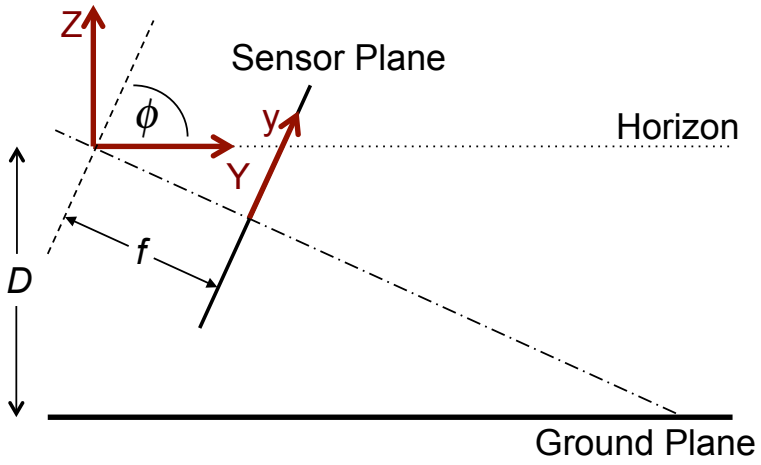


Figure 4.4: Camera geometry. Both the X-axis of the world frame and the x-axis of the image frame point out of the page.

first 566 labelled frames of Dataset 1 as training data and used the last 506 labelled frames for evaluation. Dataset 2 was used solely for evaluation, serving to assess the ability of the algorithm to generalize to different conditions.

The camera was calibrated in the lab using standard procedures [52]: The focal length was estimated to be $f = 1,142$ pixels and the principal point (p_x, p_y) was found to be centred vertically and displaced by only 3.5 pixels to the right horizontally. Skew was assumed to be zero, and pixel aspect ratio was assumed to be unity. For both datasets, the camera was mounted on a tripod on an overpass overlooking a highway; Fig. 4.4 shows the geometry in profile. We measured the camera height above the ground plane to be approximately $D = 8.01$ meters for Dataset 1, and $D = 8.36$ meters for Dataset 2. The roll angle of the camera was minimized using the camera's internal electronic levelling gauge, and we assume it to be zero in the following. We down-sampled the video to 360×270 pixel resolution prior to processing to reduce computation time. Both datasets were hand-labelled to identify a bounding box and semantic category for each vehicle in each frame. A unique ID was assigned to each unique vehicle, tracked across frames.

We assume a planar horizontal ground surface and adopt a right-hand world coordinate system $[X, Y, Z]$ centred at the camera, where the Z-axis is in the upward normal direction (Fig. 4.4). Without loss of generality, we align the x-axis of the image coordinate system with the X axis of the world coordinate system (both out of the page in Fig. 4.4). For notational simplicity we locate the centre of the image coordinate system at the principal point.

Under these conditions, a point $[X, Y]^T$ on the ground plane projects to a point $[x, y]^T$ on the image plane according to

$$\lambda[x, y, 1]^T = H[X, Y, 1]^T, \tag{4.1}$$

where λ is a scaling factor and the homography H is given by ([72], Page 328, Eqn. 15.16):

$$H = \begin{bmatrix} f & 0 & 0 \\ 0 & f\cos\phi & -fD\sin\phi \\ 0 & \sin\phi & D\cos\phi \end{bmatrix} \tag{4.2}$$

where ϕ is the tilt angle of the camera relative to the ground plane: $\phi = 0$ when the camera points straight down at the ground surface and increases to $\pi/2$ as the camera tilts up toward the horizon.

Conversely, points in the image can be back-projected to the ground plane using the inverse of this homography, $[X, Y, 1]^T = \lambda H^{-1}[x, y, 1]^T$, where

$$H^{-1} = (fD\cos 2\phi)^{-1} \begin{bmatrix} D & 0 & 0 \\ 0 & D\cos\phi & fD\sin\phi \\ 0 & -\sin\phi & f\cos\phi \end{bmatrix} \tag{4.3}$$

In Euclidean coordinates this backprojection can be written as:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \frac{D}{f\cos\phi - y\sin\phi} \begin{bmatrix} x \\ y\cos\phi + f\sin\phi \end{bmatrix} \tag{4.4}$$

This inverse homography will be used to back-project the lane boundaries detected and grouped in the image back to the ground plane.

Our method will also involve projection of 3D cuboid vehicle models resting on the ground plane to the image for comparison with detected foreground segments. For this comparison, we employ the 3×4 homogeneous camera projection matrix P: $\mathbf{x} = P\mathbf{X}$, where $P = KR[I \mid \mathbf{0}]$, and K and R are the 3×3 instrinsic parameter and rotation matrices respectively. Given our assumptions, the intrinsic matrix K reduces to:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

and the rotation matrix R reduces to:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & \sin(\phi) & -\cos(\phi) \end{bmatrix} \tag{4.6}$$

We measured ground truth values for the tilt angle ϕ using an SPI digital protractor: $\phi = 81.2$ deg for Dataset 1, $\phi = 73.4$ deg for Dataset 2. These ground truth values will be used to validate the camera tilt estimates made automatically from the imagery (see below).

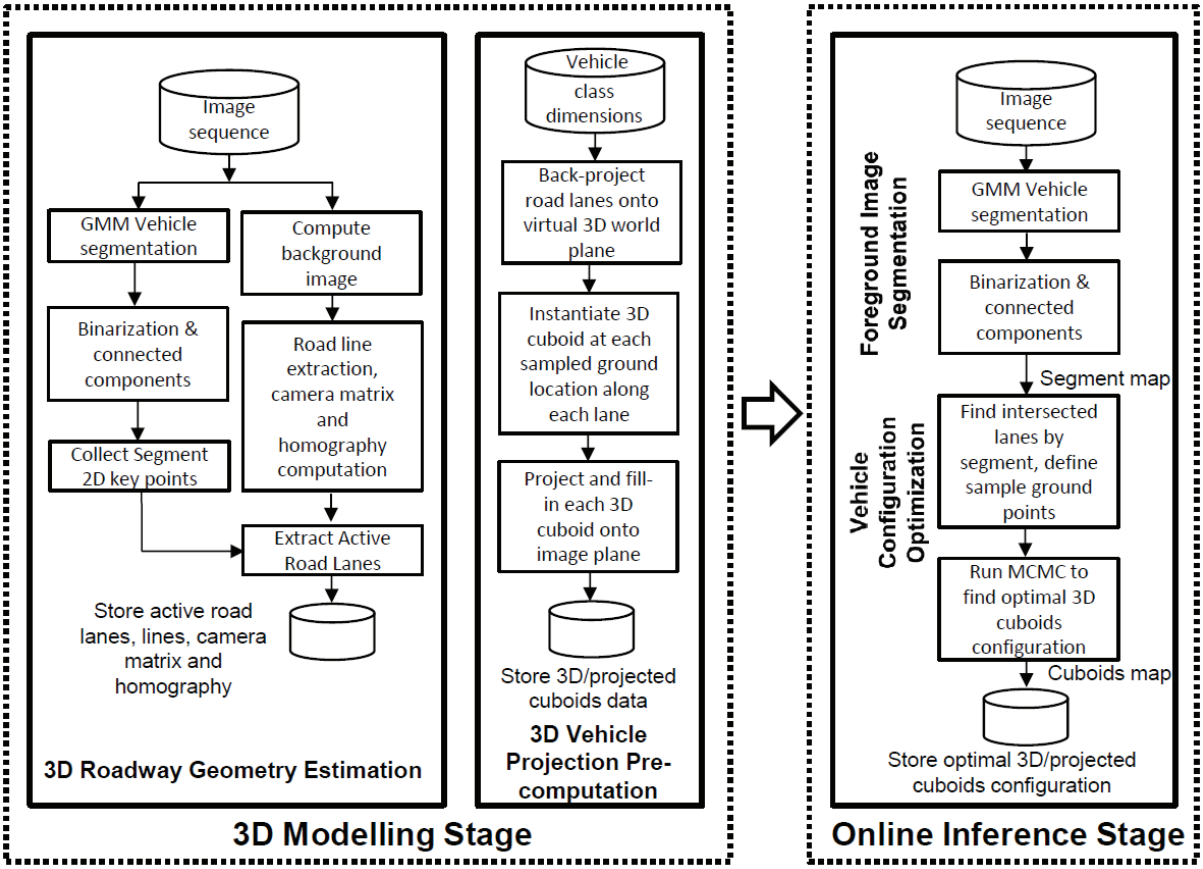


Figure 4.5: Algorithm overview.

4.3 Algorithm

Our method consists of a 3D modelling stage and an inference stage (Fig. 4.5). The 3D modelling stage runs in off-line mode, and the goal is to pre-compute active road lanes, homography parameters, and 3D models for different vehicle classes. The inference stage runs in on-line mode, and uses these pre-computed quantities to compute 2D vehicle segments and disaggregate these segments into individual vehicles using image projections of 3D models. These stages are explaned next.

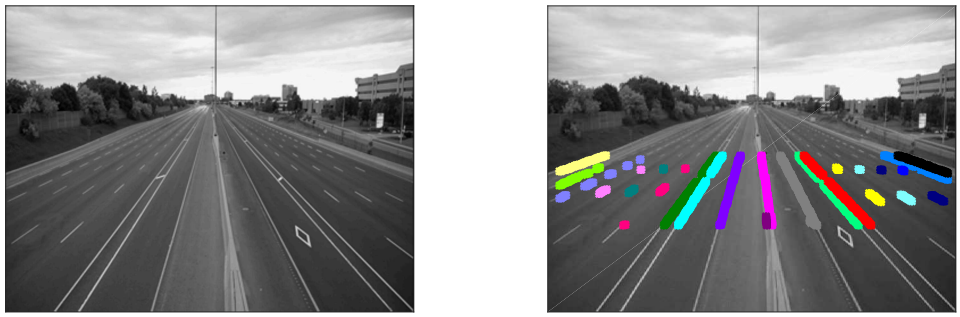
4.3.1 3D Modelling Stage

In the 3D Modelling Stage, the geometry of the ground plane lane structure of the highway is first established. This is then used to learn the image appearance of 3D cuboid vehicle models populating these lanes.

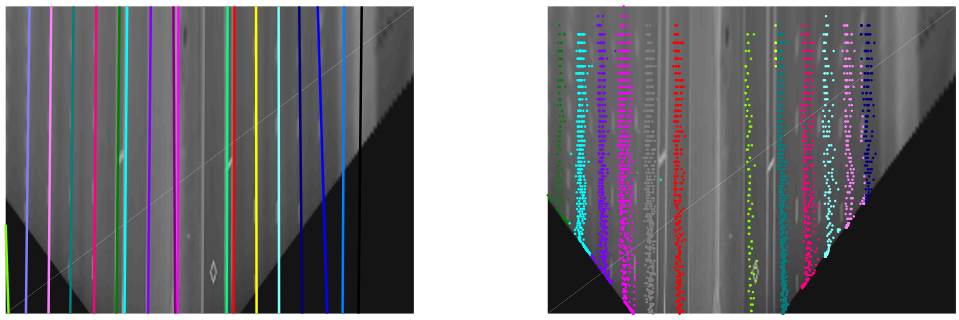
3D Roadway Geometry Estimation

Our goal is to facilitate traffic analytics by automatically estimating the projective relationship between the highway ground plane and the image, and to automatically recover the lane structure of the roadway. To achieve this, we use a 'preview video' consisting of the 200 frames immediately preceding each labelled dataset to burn in an online mixture model background subtraction algorithm [73] in order to estimate a reliable background image that clearly shows the lane structure, without occlusions from vehicles (Fig. 4.6(a)). We then employ the parallelism method presented in Chapter 3 for automatic single-view calibration and rectification. This algorithm first detects and groups local oriented structure into longer curve segments (Fig. 4.6(b)). (Although the highway shown here is straight and the focal length is known, the method can handle curved highways). These segments are then used together with knowledge of the camera height and intrinsic parameters to automatically estimate the camera tilt angle ϕ that maximizes the parallelism of the curve segments when they are back-projected to ground plane coordinates using the inverse homograph H^{-1}. (Fig. 4.6(c)).

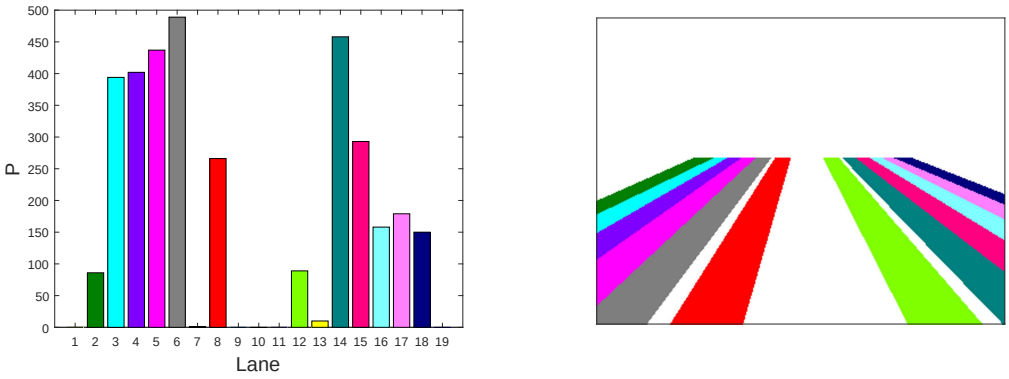
Note that the extracted curve segments include the lane boundaries but also other parallel curves generated by the meridian, parallel lane markings for the HOV lanes, shoulder, etc. In order to distinguish the traffic lanes from these other structures, we first identify as candidate lanes the curvilinear strips between all adjacent pairs of parallel curves. We then use the foreground segments detected in the preview video to identify which of these candidate lanes is active. Due to the oblique pose of the camera, the lowest point in each of these segments tends to lie close to the ground plane.



(a) (b)



(c) (d)



(e) (f)

Figure 4.6: 3D roadway geometry estimation for dataset 1 - (a) Background image recovered from initial 200 frames, (b) Line segments detected and grouped in the background image, (c) Rectified background image with initial lane boundary estimates overlaid, (d) Rectified background image with vehicle locations used to identify active lanes, (e) Distribution of traffic over lanes, (f) Labelling of active lanes in the image.

We therefore identify the ground plane location of each segment by the back-projection of its lowest point in the image and then accumulate these points over time for each of the candidate lanes (Fig. 4.6(d)). Since some of these foreground segments will actually correspond to multiple vehicles spanning multiple lanes, the frequency distribution of these points (Fig. 4.6(e)) cannot be taken as an accurate estimate of traffic volume, but it is sufficient to identify the active lanes. In our system, we treat as active any lane containing more than 10 points (Fig. 4.6(f)).

3D Vehicle Projection Pre-Computation

The estimation of 3D roadway geometry gives us the potential to transfer observations and hypotheses between the 3D scene and the 2D image . We will use this technique to implement an analysis-by-synthesis approach in which 3D hypotheses of vehicle configurations on the roadway are evaluated in terms of how well their image projections align with detected image foreground segments. The first step is to determine the exact 3D vehicle models to employ.

3D Vehicle Classes

In their 3D modelling approach, Song & Nevatia employed CAD models for three vehicle categories (sedan, SUV, truck). For highways, the distribution of vehicles types is more diverse (Fig. 4.2), and there is considerable variation in dimensions and shape within each class. For these reasons, we elect to use simpler 3D cuboid models and to learn optimal dimensions for these models from training data. Specifically, we first manually identify the subset of segments in the training dataset that involve only one vehicle that is fully-contained within a rectangular processing region of interest (See white rectangle in Fig. 4.10 (a) and (b)). The next step is to determine the lane for each segment by the lowest point in the segment. We then define a ground plane origin for the modelling of the segment as the back-projection of the projection of the centroid of

the image segment onto the midline of the lane. Next, we instantiate a set of 3D cuboid models uniformly sampling a range of plausible dimensions and locations, centred at the origin (Table 4.1). Finally, we identify the set of parameters that maximized the intersection-over-union (IOU) of the image projection of the cuboid with the observed image segment. The model dimensions for the larger vehicles, such as trucks, were also estimated manually following a similar procedure.

Table 4.1: 3D cuboid model sampling on training data. Locations are relative to ground plane origin (see text).

Parameter	Min (m)	Max (m)	Resolution (m)
Location	-3.0	3.0	1.0
Length	2.2	23.0	0.2
Width	0.8	2.6	0.2
Height	1.1	5.0	0.2

Figures 4.8 and 4.8 show the distribution of the 3D vehicle model dimensions for the test set 1, and test set 2 datasets respectively. And Fig. 4.7 shows the resulting distribution of cuboid dimensions over the training dataset. Note the broadness of the distribution, particularly in the length dimension. To partition the distribution into more compact categories, we ran the k-means algorithm [74] for $k = [1 \ldots 10]$, repeating $n = 1000$ times with random initial conditions for each value of k and selecting the solution for each value of k that minimizes the average intra-cluster variance. The number k of vehicle categories was selected to optimize the accuracy of traffic volume estimates on our training dataset: we found that $k = 4$ yields optimal performance (Fig. 4.7). We discuss this optimization in more detail in Section 4.4.

Table 4.2 shows the vehicle dimensions for the four cluster centres. Although there is not a 1:1 mapping between these clusters and semantic vehicle categories, Class 1 corresponds roughly to a compact car, Class 2 to an SUV, passenger van or pickup truck, Class 3 to a cube van and Class 4 to a semi-trailer.

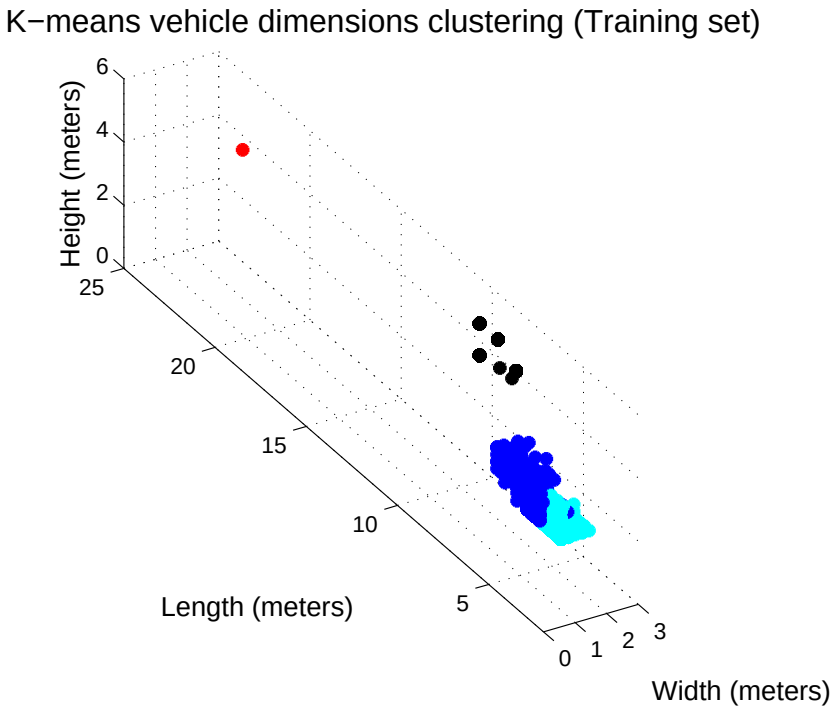


Figure 4.7: K-means clustering of labeled vehicle dimensions in the training dataset.

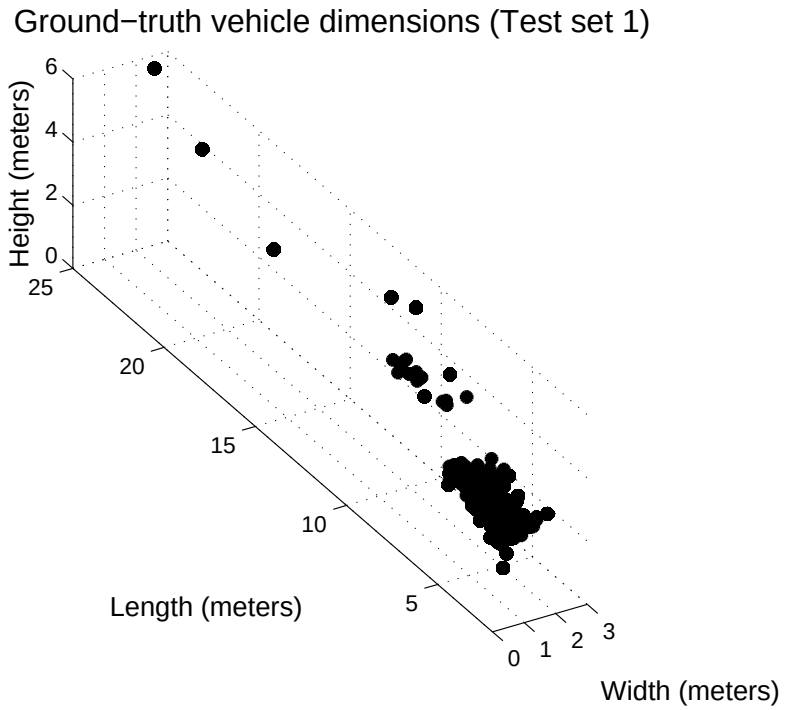


Figure 4.8: Labeled vehicle dimensions in the test set 1.

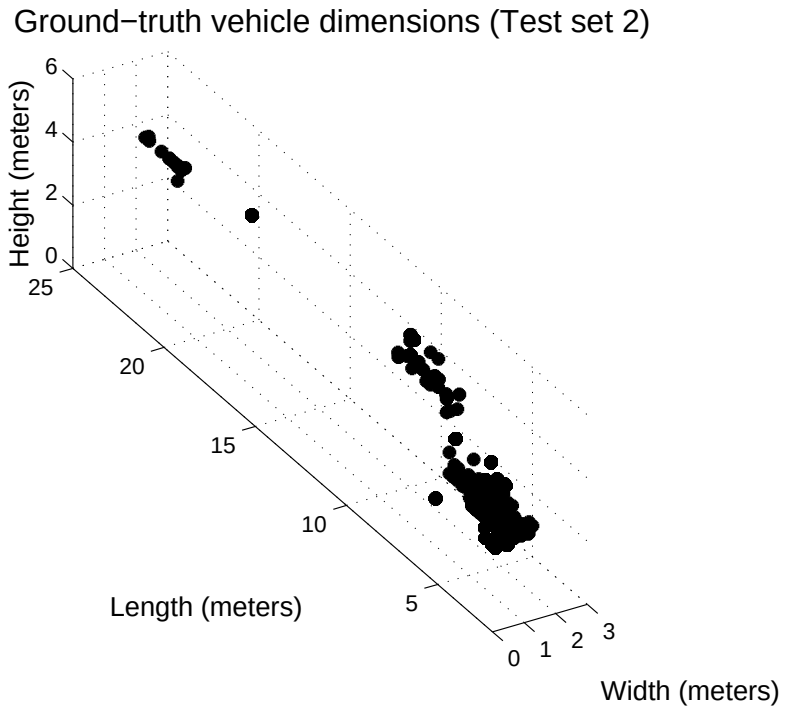


Figure 4.9: Labeled vehicle dimensions in the test set 2.

Table 4.2: Dimensions for the four vehicle classes automatically learned from our training data.

Class	Length (m)	Width (m)	Height (m)
1	4.2	1.7	1.5
2	5.7	1.8	1.7
3	9.2	2.6	4.6
4	23.0	2.6	4.0

Projection of 3D Models to the Image

Having estimated the camera matrix P and the homography H relating the ground plane to the image, the lane structure in ground plane coordinates, and the 3D cuboid model classes, we can now pre-compute an estimate of the expected image appearance (silhouette) of each vehicle class for each traffic lane and for each location along each lane. For the sake of efficiency, we assume that each vehicle will be centred within a lane. We sample lane locations at 1m resolution.

4.3.2 Online Inference Stage

Foreground Image Segmentation

We restrict our analysis to vehicles lying within or at least intersecting a region of interest in the lower portion of the video frame, to avoid very distant vehicles near the horizon (Fig. 4.10(a)).

We employ the foreground segmentation method described in Chapter A.2. The goal is to independently label each pixel in the image as foreground or background. The background subtraction component of the algorithm is based on a 2D adaptive Gaussian mixture model for pixel colour that ignores the luma channel to minimize responses to shadows. The motion component is a simple two-frame difference. The probabilistic combination of these two cues imbues the algorithm with a degree of invariance to traffic speed, since the background subtraction works well for slower

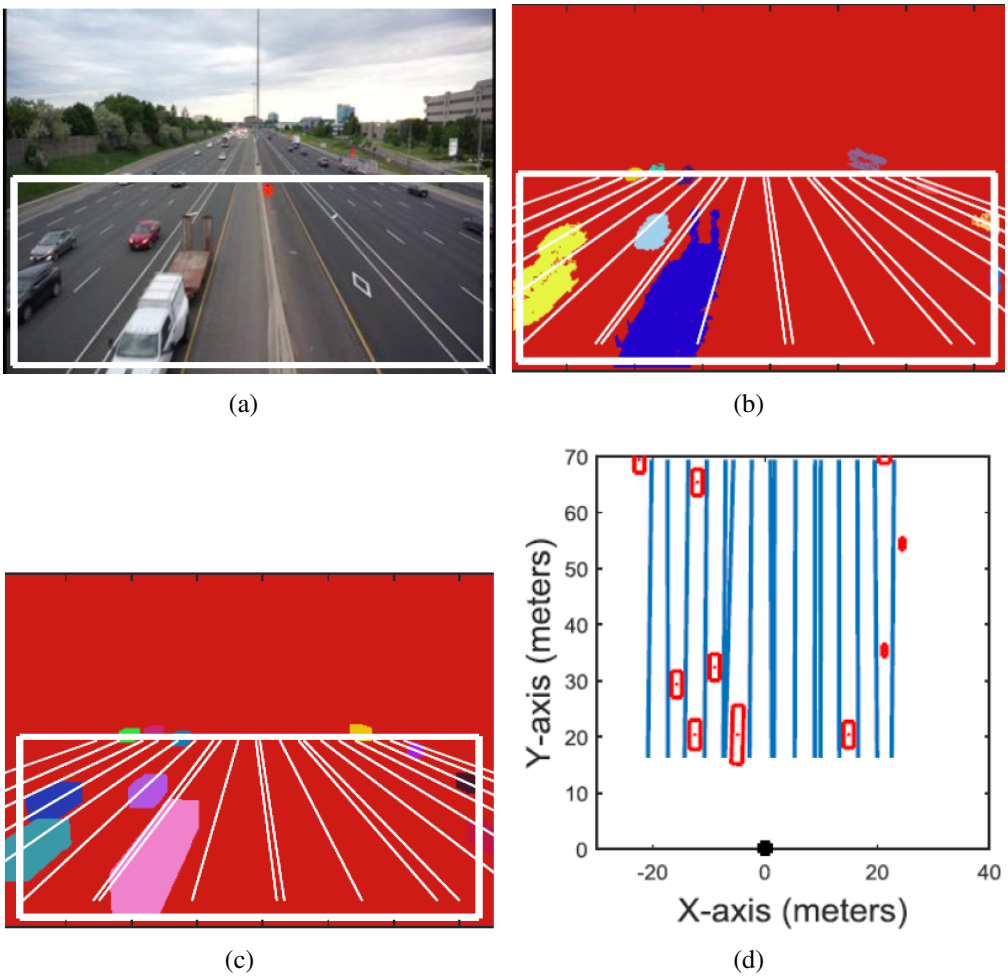


Figure 4.10: Online inference (a) Example frame from the training dataset. The white box indicates the ROI. (b) Foreground image segments computed using the GMM algorithm. (c-d) Maximum probability configuration of cuboids returned by our MCMC algorithm.

speeds and the motion detection works well for higher speeds. Note, however, that for stalled traffic, the background subtraction algorithm will eventually begin to interpret the stopped vehicles as background.

Marginal conditional likelihoods for the two cues were learned from the labelled training data and combined under a naïve Bayes assumption, and priors were learned from the proportion of the image occupied by ground truth bounding boxes. An initial segmentation is then determined by applying a threshold $p_0 = 0.08$ to the posterior ratio, and the resulting 8-connectivity [75] foreground components that exceed a criterion area $A_0 = 60$ pixels are identified as foreground segments. These thresholds were optimized to maximize the IOU of detected segments with ground-truth bounding boxes on the training dataset. Fig. 4.10(b) shows the detected foreground segments for an example video frame. In the following we will use the label GMM (Gaussian Mixture & Motion) to identify this foreground segmentation method.

3D Vehicle Configuration Optimization

Given a foreground image segment we wish to estimate the number of vehicles most consistent with the shape and size of the image segment, as well as the lane, location and class of each of these vehicles. As a measure of consistency we employ the intersection over union (IOU) of the image segment with the union of projected 3D cuboid vehicle models.

We first identify the lanes overlapped by the image segment and compute the centroid of each lane's portion of the segment. These centroids are then back-projected to the ground plane using our inverse homography H^{-1} (Eqn. 4.4) and the closest sample point on each lane's ground plane midline is identified as the origin for the search. The search space consists of seven locations, centred on the origin and spaced at 1m intervals along the midline.

If we knew that a certain image segment was created by k vehicles in k specific

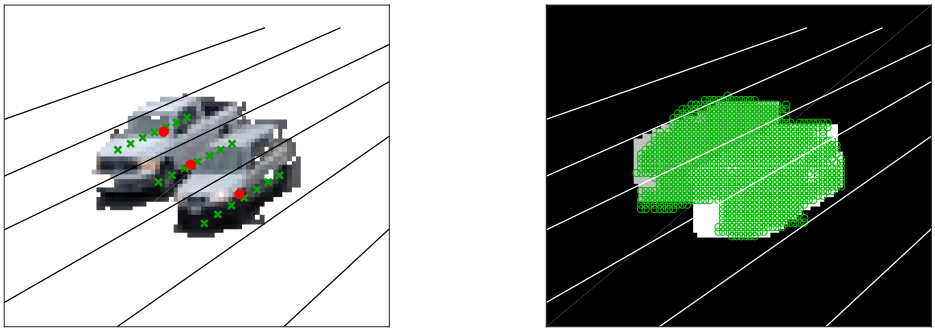
lanes, there would still be a total of $(7 \times 4)^k = 28^k$ possibly configurations, given 7 locations per lane and 4 vehicle classes. Thus, given a foreground image segment that overlaps n lanes, the total number N of possible configurations is given by

$$N = \sum_{k=1}^{n} \frac{28^k n!}{(n-k)!k!} \tag{4.7}$$

We find that foreground image segments can overlap up to five active lanes, resulting in a total of more than 20 million possible configurations: too many to explore exhaustively, especially online. Instead, we employ a Markov Chain Monte Carlo (MCMC) [67, 74, 76] method to explore the more probable regions of the configuration space within a reasonable amount of time.

We initialize the chain with a configuration computed using a simple greedy algorithm (Fig. 4.11). We first identify the lane with the largest overlap with the foreground image segment. We then exhaustively search the 28 possible location/class solutions within this lane, committing to the solution that maximizes the IOU with the whole segment. We then proceed to the lane intersecting the largest remaining unexplained portion of the foreground image segment and determine the solution in this lane that maximizes the IOU of the foreground image segment with the union of the two selected model projections. Note that it may be the case that for a particular lane no solution increases the IOU; in this case we assume no vehicle exists in this lane. This process continues until all intersected lanes have been considered.

Given this initial solution, we run MCMC, using the IOU as a model for the probability of each proposed configuration. Possible moves include: adding a vehicle in an unoccupied lane overlapped by the image segment, removing a vehicle, moving a vehicle to an adjacent location within a lane, and incrementing or decrementing the class of a vehicle by 1. Given this list of possible moves it is clear that any possible configuration can be reached. We imposed a time budget of 1 second per segment, which we found can accommodate 506 iterations of MCMC. We take as our solution



(a) (b)

Figure 4.11: Greedy initialization of our MCMC algorithm - (a) Green x show sampled traffic lane midline points and red dots indicate the centroids of the intersection of the foreground image segment with each lane. (b) The initial configuration selected by the greedy algorithm - Green area corresponds to the segment's area. Note that the furthest lane remains unoccupied since adding a vehicle there reduces the total IOU.

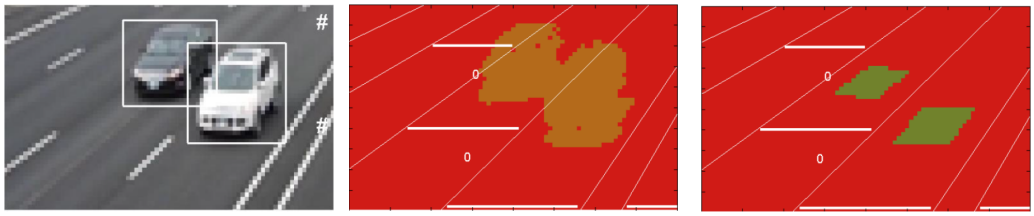
the maximum probability configuration in the chain. Fig. 4.10(c-d) shows the configurations selected for the segments (4.10(b)) in an example frame of the training video, in image and ground plane coordinates. We label our algorithm GMM3D to capture the combination of our GMM foreground segmentation algorithm with our 3D analysis-by-synthesis optimization of the vehicle configuration for each foreground segment.

4.4 Results

We evaluate our proposed GMM3D method for traffic counting and compare against the state of the art on two test video clips. Camera tilt angles were automatically estimated to be $\phi = 78.3$ deg. (error = -2.9 deg.) for Dataset 1, and $\phi = 69.5$ deg. (error = -3.9 deg) for Dataset 2. We use three measures of performance: 1) Accuracy of the total traffic flow over the duration of the clip (unique vehicle count), 2) Mean absolute error (MAE) per frame, and 3) Accuracy of estimated vehicle dimensions.

4.4.1 Total Traffic Volume

We employ our GMM3D system to estimate total traffic volume (unique vehicle count) over our two test datasets, and compare against the particle method of Barcellos et al. [60]. The Matlab code is available online (obtained from `https://www.researchgate.net/publication/278714978_Matlab_code_A_Novel_Video_Based_System_for_Detecting_and_Counting_Vehicles_at_User`). In the Barcellos method, a vehicle is detected and tracked as a group of particles, approximated by its convex hull. For each lane, a 'virtual loop' on the image is identified by hand. Any intersections between the convex hull representation of a vehicle and one or more virtual loops are identified. While one vehicle may intersect with more than one virtual loop, only the lane with the greatest intersection has its counter incremented. Barcellos et al. do not provide detailed rules for determining the location and width of these virtual loops. We therefore defined them to be qualitatively similar to those shown in their paper ([60], Figs. 8-9). Since our GMM3D model assigns each vehicle to a specific lane, traffic counting is more straightforward. For each lane we identify a virtual gate in the image that is the projection of a ground plane line orthogonal to the lane boundaries. For each identified 3D vehicle model, we extract its footprint, i.e., the contact surface between the 3D model and the ground plane, which lies entirely within one lane (see Fig. 4.12(c)). A lane counter is then incremented whenever a frame with a footprint on the gate follows a frame with no footprint on the gate. (This method works as long as two different vehicles never cross a line in two consecutive frames, which would require vehicles speeds that exceed speeds we observe in our datasets). We optimized the placement of these gates by maximizing their overlap with the ground truth boxes over the test datasets. Quantitative results are summarized in Table 4.3 and Figure 4.13. Our 3D method generates an average error of 12%, much lower than the mean error of the 2D method of Barcellos et al. (31%). Note that the 2D method consistently underestimates the traffic volume, due to a failure to disaggregate multi-



(a) (b) (c)

Figure 4.12: Virtual gates example. (a) Crop of example video frame showing vehicles and 2D bounding boxes. (b) GMM Foreground segment and (c) GMM3D cuboid footprints, with virtual gates shown in white.

ple vehicles that appear as a single image segment (Fig. 4.12(b)). This highlights the importance of 3D modelling for accurate traffic analytics.

Table 4.3: Total traffic volume estimation results.

		Count		Error %	
Test set	Ground-truth	Barcellos et al. [60]	GMM3D	Barcellos et al. [60]	GMM3D
1	75	53	85	-29%	13%
2	59	39	53	-34%	-10%

4.4.2 Per-Frame Traffic Volume

To assess the value of 3D modelling for traffic analytics, we compare the performance of two variations of a 2D method against two variations of our 3D method. In the 2D methods, the number of foreground segments is used as an estimate of the number of vehicles in the frame. We consider two foreground segmentation algorithms: the 2D method described in Appendix A, and the Principal Component Pursuit (PCP) method [77] that has been reported in previous work to outperform Gaussian mixture background subtraction algorithms; code obtained from `sites.google.com/a/istec.net/prodrig/Home/en/pubs/incpcp`. Against these we compare two versions of our 3D algorithm: one using the GMM foreground segmentation as

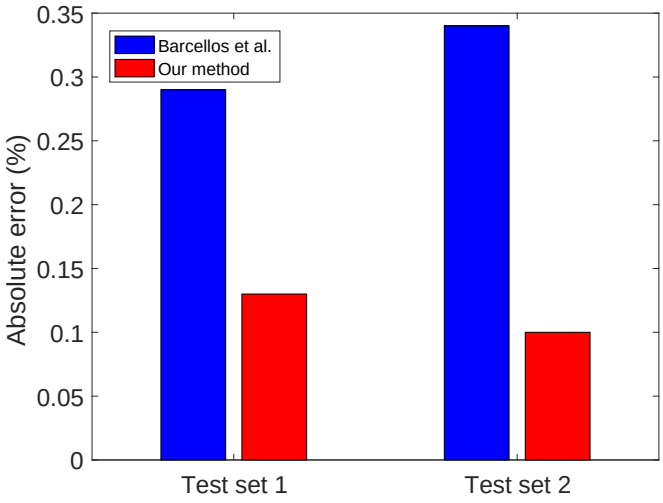


Figure 4.13: Total traffic flow evaluation.

input, labelled GMM3D, and the second using the PCP foreground segmentation as input, labelled PCP3D.

Table 4.4 and Figure 4.14 report the results from our proposed 3D method compared to those produced by using the 2D method described in Appendix A. Similarly, Table 4.5 and Figure 4.15 report the comparison against the 2D PCP method from [77]. Overall, we observe that the 3D modelling stage consistently helps achieve lower error levels, which meets our expectations in terms of contributions from 3D modeling.

Table 4.4: Per-frame traffic volume mean absolute error (MAE %). Our proposed GMM3D method against the 2D method described in Appendix A

	Test set 1	Test set 2
2D	14.8%	30.9%
3D	10.1%	20.2%

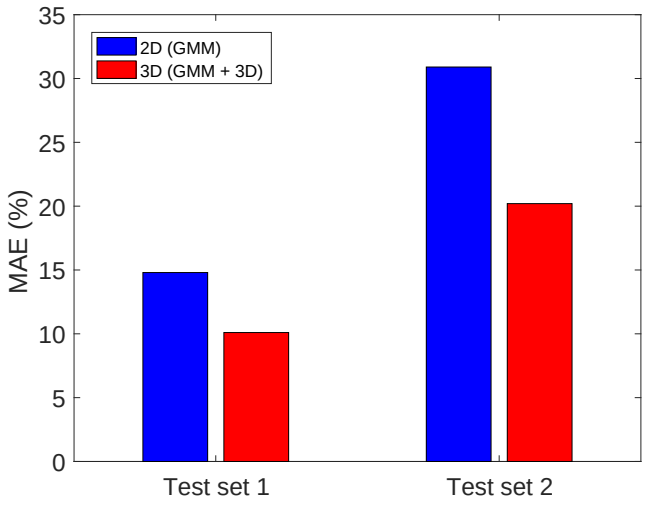


Figure 4.14: Per-frame traffic volume evaluation (MAE %). Our proposed 3D method against the 2D method described in Appendix A.

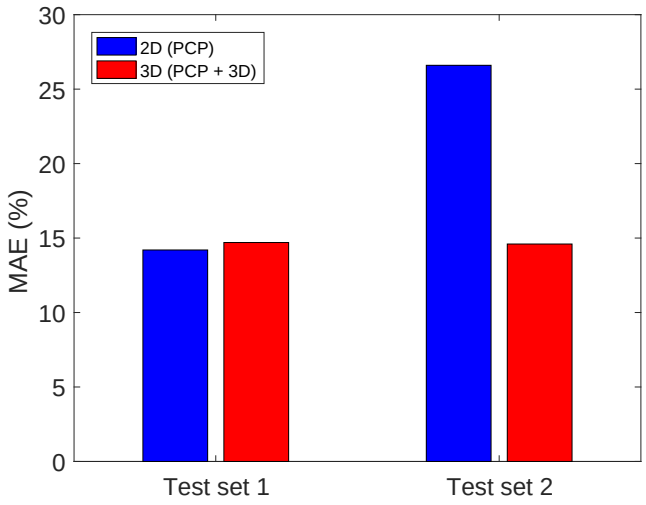


Figure 4.15: Per-frame traffic volume mean absolute error (MAE %). Our proposed 3D method against the Principal Component Pursuit (PCP) method [77]

Table 4.5: Per-Frame Traffic Volume Mean Absolute Error (MAE %). Our proposed 3D method against the Principal Component Pursuit (PCP) method [77].

	Test set 1	Test set 2
2D	14.2%	26.6%
3D	14.7%	14.6%

4.4.3 Vehicle Classification and Dimensions Estimation

In addition to improving the accuracy of traffic volume estimates, the proposed GMM3D method produces a rough estimate of the vehicle dimensions using the four size classes shown in Table 4.2. To evaluate the accuracy of these estimates, we analyze the classes assigned to vehicles at the time when they cross a virtual gate (as explained in Section 4.4.1). We compare against ground truth estimates of vehicle dimensions, estimated by hand, and employing two measures of accuracy.

First, we consider categorical accuracy. Here we identify the ground truth category as the cluster whose centre lies closest to the ground truth dimensions in a Euclidean sense. Table 4.8 shows the confusion matrices for our two test sets. Results are fairly good, and most errors involve assignment to an adjacent category. We note that confusions between Categories 1 and 2 are not surprising, given that that the two corresponding training data clusters appear to be contiguous (Fig. 4.7). This motivates our second measure of performance. Here we measure the average error in estimated vehicle dimensions, compared with a baseline estimate that uses the mean dimensions over the training set for every test vehicle (See table 4.6 and Fig. 4.16 for test set 1, and table 4.7 and Fig. 4.17 for test set 2). Our GMM3D method can be seen to generally produce surprisingly accurate estimates, within 24cm in all cases except for the length estimation in Test Set 1, which may be due to the quantization of the length dimensions of a few larger vehicles. See Table 4.8, and compare the dimensions of the longest vehicles in Test Set 1 (Fig. 4.8) against the closest mean dimensions in Fig. 4.7.

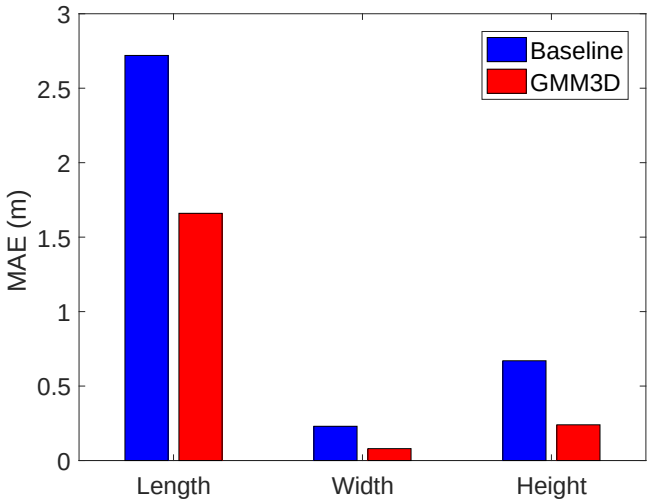


Figure 4.16: Vehicle dimensions estimation error for test set 1.

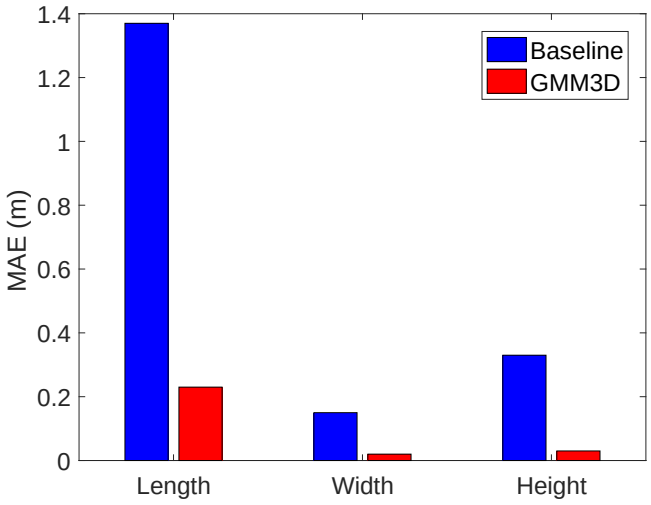


Figure 4.17: Vehicle dimensions estimation error for test set 2.

Table 4.6: Mean absolute error (MAE) for estimated vehicle dimensions on test set 1.

	Length(mm)	Width(mm)	Height(mm)
Baseline	2.72	0.23	0.67
3D	1.66	0.08	0.24

Table 4.7: Mean absolute error (MAE) for estimated vehicle dimensions on test set 2.

	Length(mm)	Width(mm)	Height(mm)
Baseline	1.37	0.15	0.33
3D	0.23	0.02	0.03

Table 4.8: Confusion matrix for vehicle classification.

Test Set 1	GMM3D 1	GMM3D 2	GMM3D 3	GMM3D 4
GT 1	0.73	0.26	0	0
GT 2	0.24	0.72	0.04	0
GT 3	0.09	0.09	0.63	0.18
GT 4	0.09	0.09	0.27	0.54
Test Set 2	GMM3D 1	GMM3D 2	GMM3D 3	GMM3D 4
GT 1	0.88	0.11	0	0
GT 2	0.22	0.77	0	0
GT 3	0	0	1	0
GT 4	0	0	0	1

4.5 Conclusions and Future Work

In this chapter, we successfully coupled our method from Chapter 3 with 3D modelling, and have demonstrated that our 3D analysis-by-synthesis approach can be used effectively to disaggregate clusters of vehicles in highway traffic video, leading to improved estimates of traffic volume and vehicle dimensions. Future improvements may derive from the incorporation of learned likelihoods and priors into the MCMC search, tracking over time, accommodating lane changes, training and evaluation on larger and more diverse datasets such as [71] and [78], and efficient implementation to allow real-time deployment.

On the other hand, the proposed system has limitations. The adopted 2D segmentation methods assume that the vehicles move. Therefore the system would stop working if the traffic becomes stationary for a relatively long time, perhaps due to a traffic jam. The system would also fail in snow conditions or very high traffic density when the lane markings become occluded. Using four vehicle classes limits the accuracy of vehicle dimension estimates. However, with a larger labelled training dataset, I am hopeful that more fine-grained classification would be possible.

Chapter 5

Discussion and Future Work

In this dissertation, I have proposed computer vision algorithms and methods that contribute to the automation of visual traffic surveillance for Advanced Traffic Management Systems. In particular, the algorithms presented in Chapter 3 (Automatic Single-View Calibration and Rectification from Parallel Curves) are desirable in the cases where the remote camera operator changes the pose of the camera. The algorithm clearly relies on the assumption that a road contains lane markings and that these markings are visible (i.e. that there is no traffic jam with several stationary vehicles occluding the markings on the image, or that there is no snow covering the road). In these cases, other image cues and techniques could be utilized such as vehicle motion cues (e.g. integrated motion responses over time), and detection and segmentation of stationary vehicles (i.e. traffic jams), perhaps using deep learning approaches. The method could also be extended to use multiple views and temporal information.

In addition, the Slot Cars method (Chapter 4) makes use of the results from this single view parallelism method, and assume that each vehicle is centered and aligned with a lane. This greatly simplifies the problem of finding the vehicles' configuration (i.e number of vehicles, their classes, and locations along their identified lanes). However, this constraint is a problem when vehicles change lanes. In these cases, the method

oversegments (usually two smaller vehicles are detected in these cases) the data in the frames where the vehicle transitions from one lane to another. A refinement step could be introduced where the orientation and location of the vehicles are allowed to vary perhaps using learned prior distributions or regression methods. We confirmed the advantage of reasoning in 3D to produce improved estimates of the number of vehicles, their classes, and their locations.

The 3DTown system proposes a framework where the information from multiple cameras can be integrated into a *single* virtual 3D environment where the operator can visualize the information of all cameras and sensors in a single interface as opposed to having large panels with separate monitors from each camera where it is difficult to understand and track moving objects that exit the field of view of one camera and enter the field of view of a different camera. In 3DTown, the same moving object would be rendered as a single avatar making it easy for human operators to track visually.

5.1 Improving Run Time

Most of the algorithms described in this dissertation were implemented in Matlab. The main focus was on the validation of the theoretical concepts discussed in this dissertation. Although vectorization was used in some cases, the code includes several for loops that often encapsulate a relative large number of operations, and Matlab is known to become slow when for loops are used.

More specifically, in Chapter 3 (Automatic Single-View Calibration and Rectification from Parallel Curves), the average run time of our non-optimized Matlab implementation on a standard dual-core laptop computer was 15.3 sec for curve extraction, and 41.1 seconds for rectification (30 iterations).

In Chapter 4 (Slot Cars: 3D Modelling for Improved Visual Traffic Analytics) we pre-computed (to reduce the run time) the projections of individual vehicles given the pre-selected classes and dimensions. The most computationally intensive stage is the

MCMC sampling. This is because the process is iterative: in each iteration a vehicle configuration is sampled and projections merged to compute the intersection over the union with the input 2D image segment. We used 506 iterations, which required roughly 1 second per segment of compute time.

In the 3DTown contribution, the tracking module operates at 8 frames per second and the camera pose algorithm takes about 8 seconds to estimate the camera pose from a single frame; thus it is useful for intermittent pan/tilt operation, but not for continuous smooth pursuit. There were also delays related to retrieving model information from Google Earth, particularly when camera pose changed.

In general, we would like the algorithms described in this document to run in real time. To achieve this goal a number of improvements are required:

Software implementation

1. Translate the Matlab code into C.
2. Make use of multiple cores for parallel processing.
3. Use CUDA C [79] implementations for GPU processing.
4. Use vectorization or libraries for hardware-accelerated matrix operations.
5. Use integral images [80] whenever possible.

Architectural modifications

1. In Chapter 3, the use of corner detector and eigenvectors could potentially be replaced with standard image gradients. However this would require learning the likelihood terms for the gradients.
2. In Chapter 4, each 2D segment in the image is processed sequentially by MCMC. Instead, multiple instances of the MCMC algorithm could be run to process each of the 2D segments in parallel.

3. In Chapter 2 (3DTown project), the use of a different geoserver framework that is optimized for dynamic visualization needs to be explored (e.g. Unity 3D).

5.2 Improving Accuracy

There are a number of improvements that could be made to improve the accuracy of the proposed algorithms:

1. Improvements to probabilistic framework for the MCMC algorithm. Currently the MCMC uses the Intersection Over the Union (IOU) as an objective that has to be maximized. Incorporating a probabilistic mechanism that takes into account learned likelihood functions and priors could produce more accurate results and faster convergence. We leave this for future work.

2. Simplification of the vehicle configuration optimization mechanism by using anchors followed by 3D cuboid regression as in [81].

3. Tracking over time. The Slot Cars method counts vehicles when they intersect defined virtual gates. At the same time, the method performs vehicle classification (dimensions); however, no tracking system is in place at this point. Having a tracking system would help improve the tracked classes and vehicle dimensions over time as the vehicles move in the image.

4. Accommodating lane changes. As explained above, the system needs to allow the orientation and lateral location of each vehicle to vary. Currently, in the system, if a vehicle changes lanes, it will be likely oversegmented as two small vehicles on adjacent lanes.

5. Larger and more diverse labeled data sets. The amount of (labeled) data used in this work is quite limited, especially the number of images and scenarios

used in the Slot Cars method. In the future, we need to collect and label many more images from more highways and illumination/weather conditions. Datasets created by other research labs [71, 78] should be used as well.

6. Deep Learning implentation. Deep convolutional neural networks could potentially be trained to estimate tilt angle and focal length and count vehicles. Future work will involve comparing the performance of this approach with the approach I have taken here, and determining whether and how the two approaches could be combined, for example by combining the presented algorithms with deep learning pixel-level semantic segmentation methods such as [82] to segment both the road and the vehicles, or even 2D bounding box methods such as [81] or [83].

Appendix A

Foreground Extraction-Based 2D Segmentation of Traffic Objects

In this dissertation, I employed the foreground extraction method proposed by Elder et al. [73] to segment pedestrians and vehicles from traffic video. This chapter focuses on the work done to adapt that method to pedestrian detection application [5] and vehicle detection application [7].

A.1 Overview of the Elder et al. Foreground Extraction Method

The foreground extraction method proposed by [73] is part of a biologically-inspired solution that combines pre-attentive low-resolution sensing for detection with shiftable, high-resolution, attentive sensing for confirmation and further analysis that was developed in the context of indoor human face detection.

The system employs layered probabilistic modeling and spatial integration of relatively simple but complementary cues, namely, foreground extraction, image differences (i.e. motion), and color.

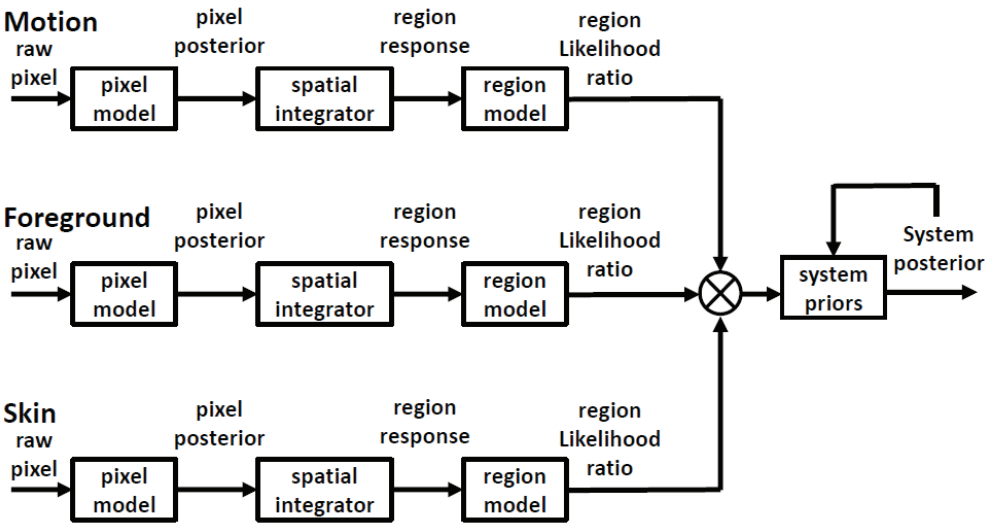


Figure A.1: Pre-Attentive system diagram taken from [73].

As shown in Figure A.1, the system has three modalities: Motion (two-frame image difference), Foreground extraction, and Skin (Color). For each modality, the first processing stage is the computation of pixel-level posterior probabilities, which are then integrated spatially over a rectangular window using L_γ normalization [73]. Finally, the likelihood distributions for the spatially-integrated cues are modelled by a mixture of 3 Gaussians. These likelihoods are then combined in a Bayesian fashion to produce a posterior probability map.

In the foreground extraction modality, each image pixel color is modelled as a mixture of two multivariate normals $\eta(\vec{\mu}_i, \Sigma_i)$, $i \in \{1, 2\}$ corresponding to background and foreground processes. The system uses Principal Component Analysis (PCA) [74] to perform image color space dimensionality reduction from 3D (RGB) to 2D to improve robustness to brightness variations.

Maximum likelihood estimates of the mean $\vec{\mu}_i$, covariance matrix Σ_i and mixing coefficient ω_i of both mixture components are updated on each frame using an incremental approximation of the EM algorithm [84]. Note that the mixing coefficients ω_i

for a specific pixel represent the prior probability of foreground and background at that pixel. It is assumed that on average, over time, each pixel corresponds more frequently to background than foreground, and thus the larger mixing coefficient is associated with the background process.

Representing foreground and background hypotheses as H and $\bar{H}$ respectively, the prior probabilities are thus given by

$$p(H) = min\left(\omega_1, \omega_2\right), p(\bar{H}) = max\left(\omega_1, \omega_2\right). \tag{A.1}$$

For each pixel of every incoming image, the posterior probability of the hypothesis H that a pixel belongs to the foreground given its colour $\vec{X}$ is computed as follows:

$$P(H|\vec{X}) = \frac{P(\vec{X}|H)P(H)}{P(\vec{X}|H)P(H) + P(\vec{X}|\bar{H})P(\bar{H})}. \tag{A.2}$$

A.2 Adaptation to Pedestrian and Vehicle Detection

A number of modifications were necessary to apply this algorithm to long-range pedestrian and vehicle detection:

1. **Removal of color modality**. After evaluating the system with highway traffic surveillance video, it was determined that the color modality was not useful in the traffic surveillance application due to the high variation in the color of the vehicles (unlike human face colors).

2. **Removal of spatial integration**. After extensive experimentation with highway traffic videos, it was determined that the integration window stage used in [73] was causing vehicles to be merged into single segments.

3. **The likelihood functions for the motion modality were re-learned from traffic training data**. Non-parametric representations of the *ON* and *OFF* likeli-

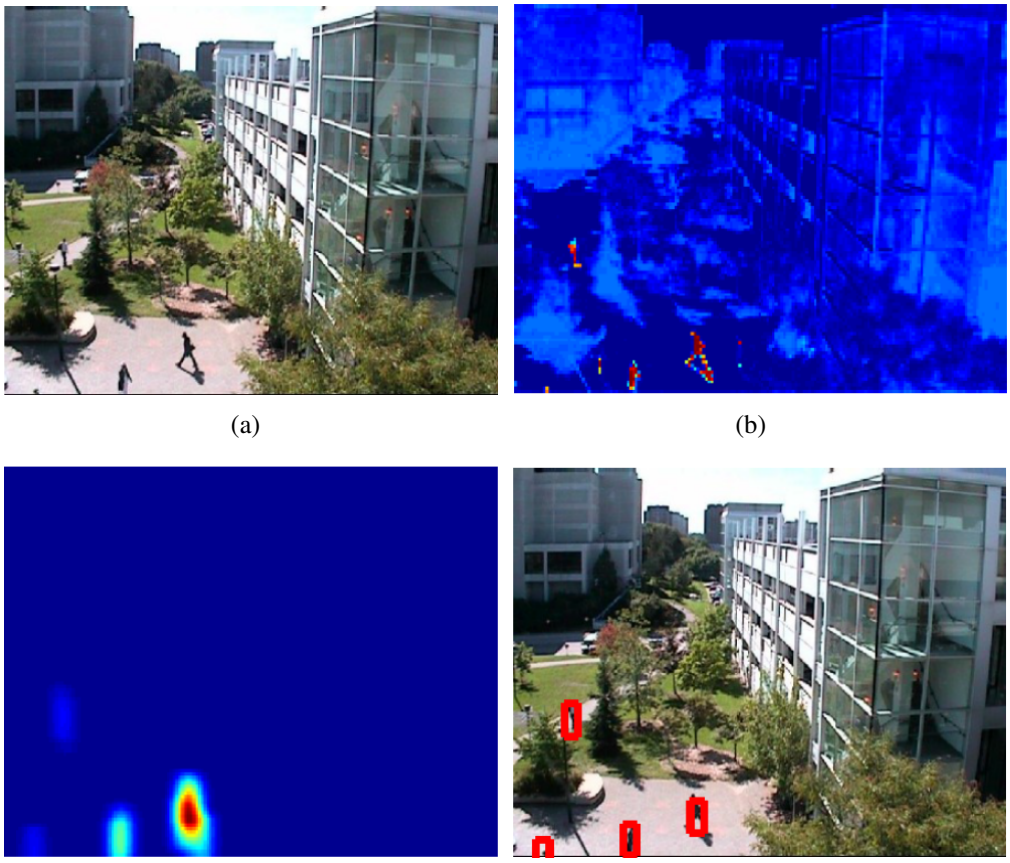
hood functions were learned using ground-truth 2D bounding boxes.

4. **The mixture model parameters were re-learned**. The 3-component mixture model parameters for the motion and foreground posterior distributions were re-learned.

5. **Adjustments to the foreground extraction method**. The foreground extraction stage modifications were: 1) *Burn-in:* The initial Gaussian mixture model parameters (mean, variance, and mixture weights) were re-learned using a video segment from the training dataset (see Section 4.2), 2) In the PCA option [73], the eigenvectors were re-learned from traffic data using the training set and the ground-truth 2D bounding boxes, and no dimensionality reduction was performed. Instead, the image pixel color rotations were performed using all three eigenvectors.

A.2.1 Computing 2D Image Object Segments

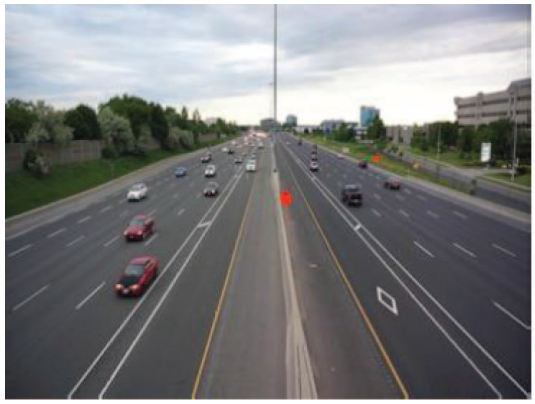
To compute 2D traffic object segments, I binarized the posterior map produced by the modified version of the Elder et al. method using a threshold $T = 0.0867$: Any pixel locations with probability greater than this threshold were set to 1, otherwise, they were set to 0. The threshold was determined by maximizing the area of the precision-recall curve using the training set ground-truth bounding boxes. The connected components from the binary array were then labeled using the Matlab function $bwlabel$, followed by one iteration of boundary removal (function $bwboundaries$), both with 8-connectivity kernel on each connected component. Small fragments were removed using a segment area threshold of $T_a = 60$ pixels, also learned by maximizing the precision-recall curve area. Only the segments that intersected the ROI were considered. Figure A.2 shows an example pedestrian detection application [5] of the modified Elder et al. method. Further, Figure A.3 shows example results for traffic vehicle segmentation.



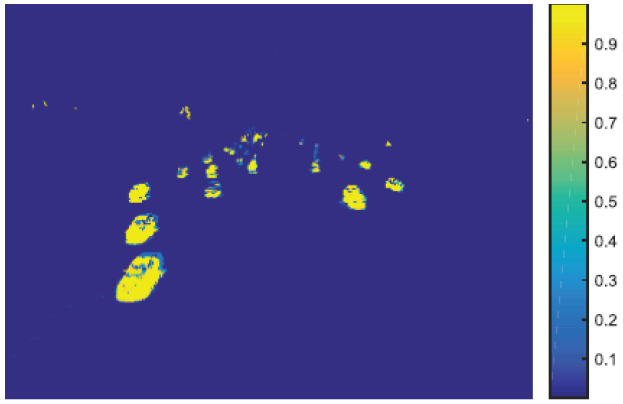
(a) (b)

(c) (d)

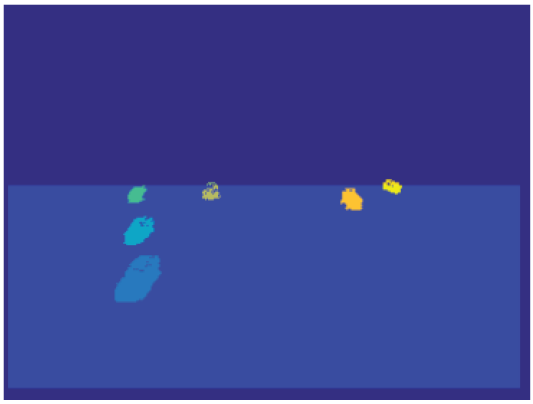
Figure A.2: Adapting the Elder et al. foreground extraction method to a pedestrian detection and tracking system used in the 3DTown project [5]. a) An image captured by a surveillance PTZ camera, b) Posterior probability map computed using the modified version of Elder et al., c) A smoothened version of the posterior map (See Chapter 2 for details), d) pedestrian 2D bounding boxes .



(a)



(b)



(c)

Figure A.3: Adapting the Elder et al. foreground extraction method to the highway vehicle detection method from [7]. Top: An image captured by a traffic surveillance camera, Middle: The corresponding posterior probability map computed using our modified version of Elder et al. The colors represent probabilies, and Bottom: 2D segmentation and rectangular region of interest (ROI) shown in lighter blue.

References

[1] ONTARIO TRAFFIC MANUAL COMMITTEE. **Advanced Traffic Management Systems**. In *Book 19*. Ministry of Transportation Ontario, 2007. 1, 2, 3

[2] FEDERAL HIGHWAY ADMINISTRATION (FHWA). **Freeway Management and Operations Handbook. Final Report**. U.S. Department of Transportation, 2003. 1

[3] HONGSHENG HE, ZHENZHOU SHAO, AND JINDONG TAN. **Recognition of Car Makes and Models From a Single Traffic-Camera Image**. *IEEE Transactions on Intelligent Transportation Systems*, **16**[6]:3182–3192, 2015. 2

[4] CHRISTOS NIKOLAOS E ANAGNOSTOPOULOS, IOANNIS E ANAGNOSTOPOULOS, VASSILIS LOUMOS, AND ELEFTHERIOS KAYAFAS. **A License Plate-Recognition Algorithm for Intelligent Transportation System Applications**. *IEEE Transactions on Intelligent transportation systems*, **7**[3]:377–392, 2006. 2

[5] EDUARDO R. CORRAL-SOTO, RON TAL, LANGYUE WANG, RAVI PERSAD, LUO CHAO, CHAN SOLOMON, BOB HOU, GUNHO SOHN, AND JAMES H ELDER. **3D Town: The Automatic Urban Awareness Project**. In *Computer and Robot Vision (CRV), 2012 Ninth Conference on*, pages 433–440. IEEE, 2012. 4, 6, 87, 90, 91

[6] EDUARDO R. CORRAL-SOTO AND JAMES H ELDER. **Automatic Single-View Calibration and Rectification from Parallel Planar Curves**. In *European Conference on Computer Vision*, pages 813–827. Springer, 2014. 5, 26, 55

[7] EDUARDO R. CORRAL-SOTO AND JAMES H. ELDER. **Slot Cars: 3D Modelling for Improved Visual Traffic Analytics**. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 5, 87, 92

[8] T. KANADE, R. COLLINS, A. LIPTON, P. ANANDAN, AND P. BURT. **Cooperative Multisensor Video Surveillance**. *Proc. of DARPA Image Understanding Workshop*, pages 3–10, 1997. 9

[9] T. KANADE, R. COLLINS, A. LIPTON, P. BURT, AND L. WIXSON. **Advances in Cooperative Multi-Sensor Video Surveillance**. *Proc. of DARPA Image Understanding Workshop*, pages 3–24, 1998. 9, 21

[10] H.S. SAWHNEY, A. ARPA, R. KUMAR, S. SAMARASEKERA, M. AGGARWAL, S. HSU, D. NISTER, AND K. HANNA. **Video Flashlights - Real Time Rendering of Multiple Videos for Immersive Model Visualization**. *Thirteenth Eurographics Workshop on Rendering (2002)*, pages 157–168, 2002. 9, 10, 21

[11] U. NEUMANN, S. YOU, J. HU, B. JIANG, AND J. LEE. **Augmented Virtual Environments (AVE): Dynamic Fusion of Imagery and 3D Models**. *VR03, March 2003*, 2003. 9, 21

[12] I.O. SEBE, J. HU, S. YOU, AND U. NEUMANN. **3D Video Surveillance with Augmented Virtual Environments**. *IWVS03, November 7, 2003, Berkeley, California, USA*, pages 107–112, 2003. 9

[13] K. KIM, S. OH, J. LEE, AND I. ESSA. **Augmenting Aerial Earth Maps with Dynamic Information**. *IEEE International Symposium on Mixed and Aug-*

mented Reality 2009, Science and Technology Proceedings, pages 19–22, 2009. 9, 21

[14] J. LI-CHEE-MING, D. GUMEROV, T. CIOBANU, AND C. ARMENAKIS. **Generation of Three-Dimensional Photo-Realistic Models from LIDAR and Image Data**. *Proceedings 2009 IEEE Toronto International Conference - Science and Technology for Humanity*, pages 445–450, 2009. 13, 14

[15] P.J. BESL AND N.D. MCKAY. **A Method for Registration of 3-D Shapes**. *IEEE Trans. Pat. Anal. and Mach. Intel.*, **14**[2]:239–256, 1992. 13

[16] D.G.BAILEY. **Raster Based Region Growing**. *in Proceedings 6th New Zealand Image Processing Workshop, Lower Hutt, New. Zealand*, pages 21–26, 1991. 14

[17] F. H MOFFIT AND E. M. MIKHAIL. *Photogrammetry*. Harper & Row, Inc., 1980. 14

[18] P. DENIS, J.H. ELDER, AND F.J. ESTRADA. **Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery**. *European Conference on Computer Vision*, pages 197–210, 2008. 14, 16, 17, 23

[19] RICHARD HARTLEY AND ANDREW ZISSERMAN. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 15, 28, 30, 31, 34

[20] J.M. COUGHLAN AND A.L. YUILLE. **Manhattan World: Compass Direction from a Single Image by Bayesian Inference**. *International Conference on Computer Vision.*, **2**:941–947, 1999. 15

[21] J. H. ELDER AND S. W. ZUCKER. **Local Scale Control for Edge Detection and Blur Estimation**. *Transactions on Pattern Analysis and Machine Intelligence*, **20**[7]:699–716, 1999. 16

[22] R.O. DUDA AND P.E. HART. **Use of the Hough Transformation to Detect Lines and Curves in Pictures**. *Communications of the ACM*, **1**[15]:11–15, 1972. 16

[23] O. BARINOVA, V. LEMPITSKY, E. TRETIAK, AND P. KOHLI. **Geometric Image Parsing in Man-Made Environments**. *European Conference on Computer Vision*, pages 57–70, 2010. 16, 27, 47, 48

[24] S. T. BARNARD. **Interpreting Perspective Images**. *Artificial Intelligence*, **21**[4]:435–462, 1983. 16

[25] M. AVRIEL. *Nonlinear Programming: Analysis and Methods*. Prentice Hall, 1976. 17

[26] C. GRAMKOW. **On Averaging Rotations**. *Int. J. Comput. Vision*, **42**[1-2]:7–16. 18

[27] COMANICIU DORIN AND PETER MEER. **Mean Shift: A Robust Approach Toward Feature Space Analysis**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**[5]:603–619, 2002. 20

[28] JAKUB SOCHOR, ROMAN JURÁNEK, JAKUB ŠPAŇHEL, LUKÁŠ MARŠÍK, ADAM ŠIROKỲ, ADAM HEROUT, AND PAVEL ZEMČÍK. **BrnoCompSpeed: Review of Traffic Camera Calibration and Comprehensive Dataset for Monocular Speed Measurement**. *arXiv preprint arXiv:1702.06441*, 2017. 27

[29] KEVIN MATZEN AND NOAH SNAVELY. **NYC3DCars: A Dataset Of 3D Vehicles in Geographic Context**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 761–768, 2013. 27, 54

[30] YOUNG-WOO SEO AND RAJ RAJKUMAR. **Use of a Monocular Camera to Analyze a Ground Vehicles Lateral Movements for Reliable Autonomous City**

Driving. In *Proceedings of IEEE IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, pages 197–203, 2013. 27

[31] KENJI OKUMA, JAMES J LITTLE, AND DAVID G LOWE. **Automatic Rectification of Long Image Sequences**. In *Asian Conference on Computer Vision*, pages 9–14, 2004. 27

[32] NEERAJ K. KANHERE AND STANLEY T. BIRCHFIELD. **A Taxonomy and Analysis of Camera Calibration Methods for Trafic Monitoring Applications**. *IEEE Transactions on Intelligent Transportation Systems*, **11**[2]:441–452, 2010. 27, 28

[33] GEORGE SK FUNG, NELSON HC YUNG, AND GRANTHAM KH PANG. **Camera Calibration from Road Lane Markings**. *Optical Engineering*, **42**[10]:2967–2977, 2003. 27

[34] XIAO CHEN HE AND NELSON HC YUNG. **New Method for Overcoming Ill-Conditioning in Vanishing-Point-Based Camera Calibration**. *Optical Engineering*, **46**[3]:037202–037202, 2007. 27, 28

[35] ANDREW HS LAI AND NELSON HC YUNG. **Lane Detection by Orientation and Length Discrimination**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **30**[4]:539–548, 2000. 27

[36] KAI-TAI SONG AND JEN-CHAO TAI. **Dynamic Calibration of Pan-Tilt-Zoom Cameras for Traffic Monitoring**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **36**[5]:1091–1103, 2006. 28

[37] TODD N. SCHOEPFLIN AND DANIEL J. DAILEY. **Dynamic Camera Calibration of Roadside Traffic Management Cameras for Vehicle Speed Estimation**. *IEEE Transactions on Intelligent Transportation Systems*, **4**[2]:90–98, 2003. 28

[38] ZHAOXIANG ZHANG, MIN LI, KAIQI HUANG, AND TIENIU TAN. **Robust Automated Ground Plane Rectification Based on Moving Vehicles for Traffic Scene Surveillance**. *ICIP 2008. 15th IEEE International Conference on Image Processing, 2008.*, pages 1364–1367, 2008. 28

[39] FW CATHEY AND DJ DAILEY. **A Novel Technique to Dynamically Measure Vehicle Speed Using Uncalibrated Roadway Cameras**. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 777–782. IEEE, 2005. 28

[40] PATRYK FILIPIAK, BARTLOMIEJ GOLENKO, AND CEZARY DOLEGA. **NSGA-II Based Auto-Calibration of Automatic Number Plate Recognition Camera for Vehicle Speed Measurement**. In *European Conference on the Applications of Evolutionary Computation*, pages 803–818. Springer, 2016. 28

[41] E. RIBEIRO AND E. R. HANCOCK. **Estimating the 3D Orientation of Texture Planes Using Local Spectral Analysis**. *Image and Vision Computing*, pages 619–631, 2000. 28

[42] XINHUA YOU AND YUAN ZHENG. **An Accurate and Practical Calibration Method for Roadside Camera Using Two Vanishing Points**. *Neurocomputing*, **204**:222–230, 2016. 28

[43] LAZAROS GRAMMATIKOPOULOS, GEORGE KARRAS, AND ELLI PETSA. **Automatic Estimation of Vehicle Speed from Uncalibrated Video Sequences**. In *Proceedings of International Symposium on Modern Technologies, Education and Profeesional Practice in Geodesy and Related Fields*, pages 332–338, 2005. 28

[44] J.S. KIM, P. GURDJOS, AND I.S. KWEON. **Geometric and Algebraic Constraints of Projected Concentric Circles and Their Applications to Camera**

Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 637–642, 2005. 28

[45] GUANG JIANG AND LONG QUAN. **Detection of Concentric Circles for Camera Calibration**. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, **1**, pages 333–340. IEEE, 2005. 28

[46] MASOUD O. AND N.P. PAPANIKOLOPOULOS. **Using Geometric Primitives to Calibrate Traffic Scenes**. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, **2**:1878–1883, 2004. 28

[47] F. SCHAFFALITZKY AND A. ZISSERMAN. **Geometric Grouping of Repeated Elements within Images**. In D. A. FORSYTH, J. L. MUNDY, V. DI GESU, AND R. CIPOLLA, editors, *Shape, Contour and Grouping in Computer Vision*, LNCS 1681, pages 165–181. Springer-Verlag, 1999. 28, 34

[48] F. SCHAFFALITZKY AND A. ZISSERMAN. **Planar Grouping for Automatic Detection of Vanishing Lines and Points**. *Image and Vision Computing*, **18**:647–658, 2000. 28, 34

[49] R. C. YATES. *A Handbook on Curves and their Properties*. National Council of Teachers of Mathematics, 1974. 34

[50] A. GRAY, E. ABBENA, AND S. SALAMON. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, Boca Raton, FL, 3 edition, 2006. 34

[51] RICHARD I. HARTLEY. **Self-Calibration from Multiple Views with a Rotating Camera**. *Computer Vision (ECCV'94)*, pages 471–478, 1994. 35

[52] ZHENGYOU ZHANG. **Flexible Camera Calibration by Viewing a Plane from Unknown Orientations**. *IEEE International Conference on Computer Vision*, **1**:666–673, 1999. 38, 45, 58

[53] C. HARRIS AND M. STEPHENS. **A Combined Corner and Edge Detector**. *Proceedings of the 4th Alvey Vision Conference.*, pages 147–151, 1988. 39

[54] PAUL J. BESL AND NEIL D. MCKAY. **A Method for Registration of 3-D Shapes**. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 239–256, 1992. 43

[55] ROBERT B. FISHER. **Projective ICP and Stabilizing Architectural Augmented Reality Overlays**. *Virtual and Augmented Architecture (VAA'01)*, pages 69–80, 2001. 43

[56] ROBERT V HOGG AND ALLEN T CRAIG. *Introduction to Mathematical Statistics.(5"" Edition)*. Upper Saddle River, New Jersey: Prentice Hall, 1995. 48

[57] NADA ELASSAL AND JAMES H ELDER. **Estimating Camera Tilt from Motion without Tracking**. In *Computer and Robot Vision (CRV), 2017 14th Conference on*, pages 72–79. IEEE, 2017. 51

[58] MARTIN A FISCHLER AND ROBERT C BOLLES. **Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography**. In *Readings in computer vision*, pages 726–740. Elsevier, 1987. 51

[59] CHRISTIANO BOUVIE, JACOB SCHARCANSKI, PABLO BARCELLOS, AND FABIANO LOPES ESCOUTO. **Tracking and Counting Vehicles in Traffic Video Sequences using Particle Filtering**. *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 812–815, 2013. 54

[60] PABLO BARCELLOS, CHRISTIANO BOUVIÉ, FABIANO LOPES ESCOUTO, AND JACOB SCHARCANSKI. **A Novel Video Based System for Detecting and Counting Vehicles at User-Defined Virtual Loops**. *Expert Systems with Applications*, **42**[4]:1845–1856, 2015. 54, 74, 75

[61] J QUESADA AND P RODRIGUEZ. **Automatic Vehicle Counting Method Based on Principal Component Pursuit Background Modeling**. *IEEE International Conference on Image Processing*, pages 3822–3826, 2016. 54

[62] PINGKUN YAN, SAAD M KHAN, AND MUBARAK SHAH. **3D Model Based Object Class Detection in an Arbitrary View**. *IEEE International Conference on Computer Vision*, pages 1–6, 2007. 54

[63] JAN PROKAJ AND GÉRARD MEDIONI. **3-D Model Based Vehicle Recognition**. *Winter Conference on Applications of Computer Vision*, pages 1–7, 2009. 54

[64] KRISHNAN RAMNATH, SUDIPTA N SINHA, RICHARD SZELISKI, AND EDWARD HSIAO. **Car Make and Model Recognition Using 3D Curve Alignment**. *IEEE Winter Conference on Applications of Computer Vision*, pages 285–292, 2014. 54

[65] SAMEER AGARWAL, YASUTAKA FURUKAWA, NOAH SNAVELY, IAN SIMON, BRIAN CURLESS, STEVEN M SEITZ, AND RICHARD SZELISKI. **Building Rome in a Day**. *Communications of the ACM*, **54**[10]:105–112, 2011. 54

[66] RICHARD SZELISKI. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010. 54

[67] XUEFENG SONG AND RAMAKANT NEVATIA. **A Model-Based Vehicle Segmentation Method for Tracking**. *IEEE International Conference on Computer Vision*, **2**:1124–1131, 2005. 54, 72

[68] ZUWHAN KIM. **Robust lane detection and tracking in challenging scenarios**. *IEEE Transactions on Intelligent Transportation Systems*, **9**[1]:16–26, 2008. 55

[69] MARKÉTA DUBSKÁ, ADAM HEROUT, AND JAKUB SOCHOR. **Automatic Camera Calibration for Traffic Understanding.** *British Machine Vision Conference*, 2014. 57

[70] XIAOZHI CHEN, KAUSTAV KUNDU, ZIYU ZHANG, HUIMIN MA, SANJA FIDLER, AND RAQUEL URTASUN. **Monocular 3D Object Detection for Autonomous Driving**. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016. 57

[71] JAKUB SOCHOR, ADAM HEROUT, AND JIRI HAVEL. **BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition**. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3006–3015, 2016. 57, 80, 86

[72] S.J.D. PRINCE. *Computer Vision: Models, Learning and Inference*. Cambridge University Press, Cambridge, UK, 2012. 59

[73] J.H. ELDER, S.J.D. PRINCE, Y. HOU, M. SIZINTSEV, AND E. OLEVSKIY. **Pre-Attentive and Attentive Detection of Humans in Wide-Field Scenes**. *International Journal of Computer Vision*, pages 47–66, 2007. 62, 87, 88, 89, 90

[74] CHRISTOPHER M BISHOP. *Pattern Recognition And Machine Learning*. springer, 2006. 65, 72, 88

[75] RAFAEL C GONZALEZ, RICHARD E WOODS, ET AL. **Digital Image Processing**, 2002. 71

[76] TAO ZHAO AND RAMAKANT NEVATIA. **Bayesian Human Segmentation In Crowded Situations**. In *Computer Vision and Pattern Recognition, 2003. Pro-*

ceedings. 2003 IEEE Computer Society Conference on, **2**, pages II–459. IEEE, 2003. 72

[77] PAUL RODRIGUEZ AND BRENDT WOHLBERG. **A Matlab Implementation of a Fast Incremental Principal Component Pursuit Algorithm for Video Background Modeling**. *2014 IEEE International Conference on Image Processing (ICIP)*, pages 3414–3416, 2014. 75, 76, 77, 78

[78] **MIOvision Traffic Camera Dataset http://podoce.dinf.usherbrooke.ca/**. 80, 86

[79] JOHN CHENG, MAX GROSSMAN, AND TY MCKERCHER. *Professional Cuda C Programming*. John Wiley & Sons, 2014. 84

[80] PAUL VIOLA AND MICHAEL J JONES. **Robust Real-Time Face Detection**. *International journal of computer vision*, **57**[2]:137–154, 2004. 84

[81] WEI LIU, DRAGOMIR ANGUELOV, DUMITRU ERHAN, CHRISTIAN SZEGEDY, SCOTT REED, CHENG-YANG FU, AND ALEXANDER C BERG. **Ssd: Single shot multibox detector**. In *European conference on computer vision*, pages 21–37. Springer, 2016. 85, 86

[82] TOBIAS POHLEN, ALEXANDER HERMANS, MARKUS MATHIAS, AND BASTIAN LEIBE. **Full-resolution residual networks for semantic segmentation in street scenes**. *arXiv preprint*, 2017. 86

[83] JOSEPH REDMON, SANTOSH DIVVALA, ROSS GIRSHICK, AND ALI FARHADI. **You only look once: Unified, real-time object detection**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 86

[84] N. FRIEDMAN AND S. RUSSEL. **Image Segmentation in Video Sequences: A Probabilistic Approach**. *In Proc. UAI*, pages 175–181, 1997. 88