Ground Plane Rectification from Crowd Motion

by

Ian John Hales

Submitted in accordance with the requirements for the degree of Doctor of Philosophy.

UNIVERSITY OF LEEDS

The University of Leeds
School of Computing

January 2014

Declarations

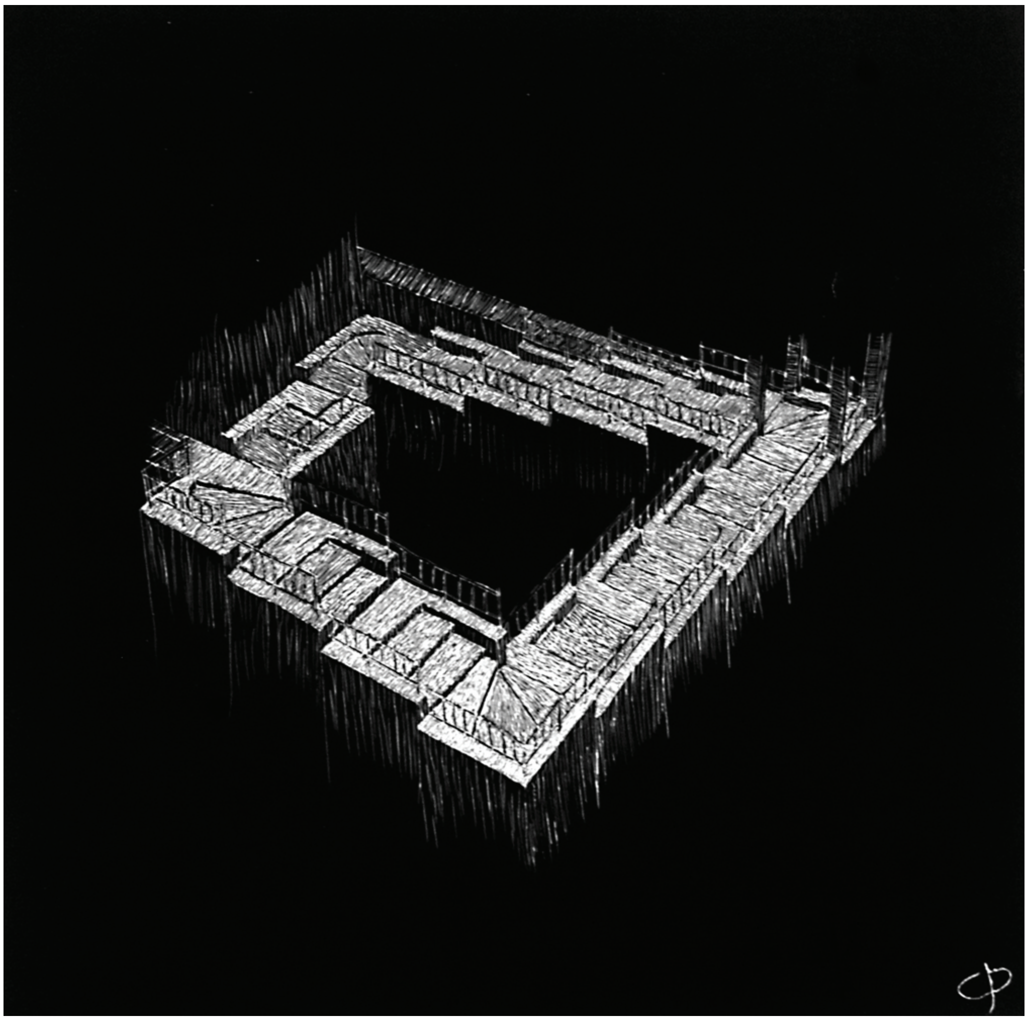
The candidate confirms that the work submitted is his/her own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Some parts of the work presented in this thesis have been published in the following article and book, for both of which the author of the thesis was the lead author:

Hales, I., Hogg, D., Ng, K. and Boyle, R. (2013) "Automated Ground-Plane Estimation for Trajectory Rectification", In R. Wilson, E. Hancock, A. Bors, and W. Smith (eds.), *Computer Analysis of Images and Patterns*, vol. 8048 of *Lecture Notes in Computer Science*, pp. 378-385. Springer Berlin Heidelberg.

Portions of chapter 3 are briefly discussed in:

Sonka, M., Hlavac, V., Boyle, R. (2014) "Reconstructing Scene Geometry" in "Image Processing, Analysis, and Machine Vision". Cengage Learning, 4 edn. pp 675-677.



"*Escher Stair*" by Chris Detmer, an interpretation of Escher's original "Ascending and Descending". Reproduced with permission of the artist.

Acknowledgements

First of all I would like to express my utmost gratitude towards my current supervisors, Prof. David Hogg and Dr. Kia Ng. Their support, creativity and wisdom over the last four years has been a great inspiration to me and they have taught me much about research methodology, critical thinking and the world of academia throughout my PhD. Their enthusiasm and encouragement has proved to be most contagious and the discussions in our meetings were always both fruitful and enjoyable. I would also like to convey my appreciation to my examiners, Dr. Andy Bulpitt and Prof. Tim Ellis for taking the time to read my thesis, examine me and give me useful and insightful feedback.

I also wish to thank my former supervisor Prof. Roger Boyle, who not only supported me through my first research experience at undergraduate level, but inspired me to start a PhD. Meetings were invariably an experience, whether it be working through new concepts on the whiteboard or discussing his latest bee-related mishaps, I always left with a smile on my face, feeling motivated for the week ahead.

Without the help and support of my colleagues and friends, I could not have made it through the last four years. I offer my utmost thanks to Nicci Kerrison, Dave Harrison, Dave Stott, Duane Carey, Sam Johnson and so many others who have helped me through the hard times and celebrated with me through the good. I would also like to thank the staff in the School of Computing at Leeds, particularly Judi and Teresa, without whom I would still be searching for my desk, let alone submitting a thesis, and of course Jill Duggleby, whose supply of healthy lunches made it possible for me to survive the final push of my write-up. My gratitude also goes to Dr. Derek Magee and Dr. Matthew Hubbard whose advice over a beer often gave a new perspective on my work. There are many others, too many to list here, but my gratitude goes out to everyone over the years who has helped me, discussed ideas with me or just been a friend to me. I thank you all.

I would like to express sincere thanks to Bethan Watkins, who has supported me at the toughest of times, understanding when I was too caught up in my work for even the most basic communication and accepting graciously those periods when I was unable to see her for days on end.

Most of all, I wish to thank my family. Without the unending support of my parents, both emotional and financial, I would not be here now. They were always at the end of the phone when I needed them and they were always willing to listen to my rants when things went wrong. Without their encouragement – the subtle pushes when I lapsed and the less subtle pushes when I continued to lapse – I would never have survived this experience.

Abstract

This work focuses on the estimation of the ground-plane parameters needed to rectify and reconstruct crowded pedestrian scenes, projected into 2D by an uncalibrated, monocular camera. Deformities introduced during the imaging process affect metrics such as size, velocity and distance, which are often useful when examining the behaviour of agents within the scene.

A framework is presented to reverse "perspective distortion" by calculating the "ground-plane", upon which motion within the scene occurs. Existing methods use geometric features, such as parallel lines, or objects of known size, such as the height of individuals in the scene; however these features are often unavailable in densely crowded scenes due to occlusions.

By measuring only the imaged velocity of tracked features, assumed to be constant in the world, the issue of occlusion can be largely overcome. A novel framework is presented for estimation of the ground-plane and camera focal-length for scenes modelled with a single plane. The above assumption is validated against simulations, outperforming an existing technique [12] against real-world benchmark data.

This framework is extended into a two-plane world and the additional challenge of determining the respective topology of the planes is introduced. Several methods for locating the intersection-line between the two planes are evaluated on simulations, with the effect of variation in velocity and the height of tracked features on reconstruction accuracy being investigated, with the results indicating this technique is suitable in real-world conditions.

This framework is generalised, removing the need for prior knowledge of the number of planes. The problem is reformulated as a linear-series of planes, each connected by a single hinge, allowing the calculation of a single rotation for each new plane. Again, results are shown against simulations on scenes of varying complexity, as well as real-world datasets validating the success of this method given realistic variations in velocity.

Contents

List of Figures

List of Tables

Notation

General Notation

Geometric entities will be represented by column vectors and denoted by bold-face, lowercase symbols, such as $\mathbf{x}$. Row vectors are denoted as the transpose of a column vector, $\mathbf{x}^\top$. Using this convention, we define the following notation:

Image-space coordinates	$\mathbf{u} = (u, v, 1)^\top$
World-space coordinates	$\mathbf{x} = (x, y, z)^\top$
Matrices	Uppercase, bold-face symbol. E.g. $\mathbf{M}$
Dot-Product	$\mathbf{x} \circ \mathbf{y}$ for vectors $\mathbf{x}$ and $\mathbf{y}$
Cross-Product	$\mathbf{x} \times \mathbf{y}$ for vectors $\mathbf{x}$ and $\mathbf{y}$

Camera and Plane Parameters

We define the plane and camera parameters as follows:

Unit Normal to Plane	$\mathbf{n} = (a, b, c)^\top$
Camera-Plane Distance	The scalar, d
Plane Equation	$\mathbf{n} \circ \mathbf{x} = d$
Camera's Angle of Elevation	Rotation in camera's x-axis, θ (see figure 1)

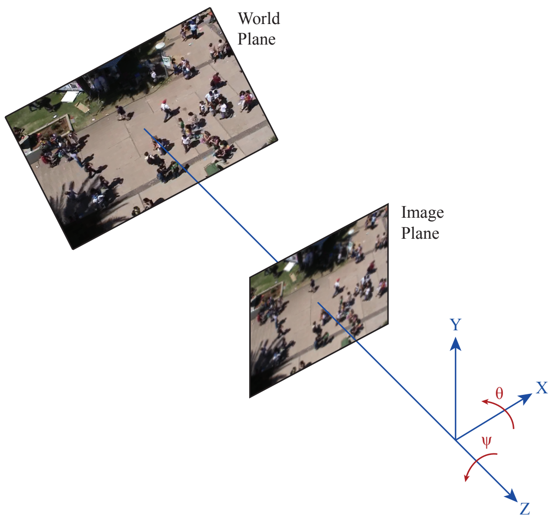


Figure 1: This illustrates the orientation of the ground-plane with respect to the camera axis (x, y, z), showing the ground-plane in world coordinates and the same transformed into image coordinates. The rotation parameters of the camera, in terms of elevation in the camera x-axis and yaw in the z-axis (θ and ψ respectively) are also given.

Camera's Angle of Yaw	Rotation in camera's z-axis, ψ (see figure 1)

Motion Vectors

We define the notion of a *motion vector* (discussed further in section 3.3.2) as a pair of points representing the position of a tracked feature at consecutive time intervals. These are depicted thusly in the text:

Image-space motion vectors	$\Upsilon = \{(\mathbf{u}_i, \mathbf{u}'_i)_{i=1\ldots N_\Upsilon}\}$
World-space motion Vectors	$\Xi = \{(\mathbf{x}_i, \mathbf{x}'_i)_{i=1\ldots N_\Xi}\}$
Assumed world speed	The scalar, $\hat{l}$

Trajectories

We define a trajectory as a time series of points upon on a plane recorded at equally spaced time-steps. We observe these in the image-coordinate system and wish to obtain their respective points within the camera coordinate system through perspective back-projection.

Image-space Trajectory	$\tau = (\mathbf{u}_i^\tau)_{i=1 \ldots N_\tau}, \forall_{\tau \in \mathbf{T}}$
World-space Trajectory	$\tau = (\mathbf{x}_i^\tau)_{i=1 \ldots N_\tau}, \forall_{\tau \in \mathbf{T}}$
Set of all Trajectories	$\mathbf{T} = \{\tau_i\}_{i=1 \ldots N_\mathbf{T}}$
Trajectory Speeds	The displacements at each time-step for trajectory τ. $\{L_i^\tau\}_{i=2}^{i=N_\tau}$ Note that $i = 2$ is chosen as the lower bound since L_i^τ examines i and $i-1$.
Mean Speed	For some trajectory, τ: $\mu(L^\tau)$
Std. Dev. of Speeds	For some trajectory, τ: $\sigma(L^\tau)$
Set of all speeds	$\mathbf{L} = \{L^\tau\}_{\tau \in \mathbf{T}}$

Chapter 1

Introduction

When a camera captures a real-world scene, it transforms the 3-dimensional (3D) coordinates of the world to 2-dimensional (2D) image coordinates. This transformation results in several obvious distortions, notably the effect of perspective distortion.

Analysis of perspective for the creation of images has been performed throughout history, from its discussion in Euclid's "Optics" to the work of Viator in the 16th century [2] and the artists of the renaissance period. In the 20th and 21st century, knowledge of perspective geometry has allowed the reconstruction of 3D scenes from their 2D image projections.

In computer vision, it is not uncommon to take measurements of certain metrics relating to objects in an image, such as speed or height. It is desirable to take these measurements as they exist in the 3D coordinate system; however, during projection in the image, that information is lost. Whilst transformation of a 3D system into 2D is a simple task, reproducing the 3D coordinates from the 2D image coordinates is impossible without additional information or constraints upon the environment.

By capturing the equation of the "ground-plane" (that is, the plane upon which the tracked objects are assumed to move), we can negate the effect of perspective distortion and provide a vastly improved measurement of the objects' motion.

1.1 Motivation

The world around us is monitored by millions of CCTV cameras; as of 2002, in the UK alone there were an estimated 4.2 million of them watching over us [88], a number that is expected to have grown since. Due to the vast numbers, many are not watched constantly, indeed an operator may have tens of cameras to watch at once. Because of this, the need has arisen for automated monitoring of video streams so as to minimise the requirement for human interaction.

This is the basis for several fields in computer vision, including activity analysis for the purpose of "event" or "anomaly" detection where an automated system uses some model of "normal" behaviour and identifies when observed behaviour diverges from that model [7,1,10,11]. In these situations, measurements such as size, speed and shape often provide important cues for the system to establish what is happening. As an illustrative example, consider a crowd of people in a busy square. If that crowd suddenly begins to move at high speed away from a certain scene position, it is not unreasonable to assume something untoward has occurred to cause such behaviour.

Event detection is often coupled with "density estimation", the problem of automatically establishing how busy a scene is in terms of the number of people in the space [86,43,78, 102]. Clearly in this case, a notion of size is important, either in terms of the people in the scene or the dimensions of the scene itself, so as to ascertain how dense the crowd is. Both of event detection and density estimation have particular importance in crowd safety monitoring, where detection of potentially dangerous situations can be used to prevent them escalating.

As mentioned above, the measurement of these quantities is affected by the process of projection from the world into an image when it is viewed by a monocular camera. Without rectification of the distortion undertaken by the scene, an example of which is shown in figure 1.1, one cannot reliably use these metrics for further analysis. In many cases the information to reconstruct the scene is not available for a piece of video footage and so it must be calculated after the fact. This is particularly relevant given the explosion of online video hosting services such as YouTube [127]. By using a non-intrusive calibration method (i.e. one does not require a known calibration object such as a chessboard pattern) a vast wealth of video information can be capitalised upon.

Different approaches exist to correct for the distortion, from a simple linear normalisation to full camera calibration. Some methods require manual intervention by a human, such

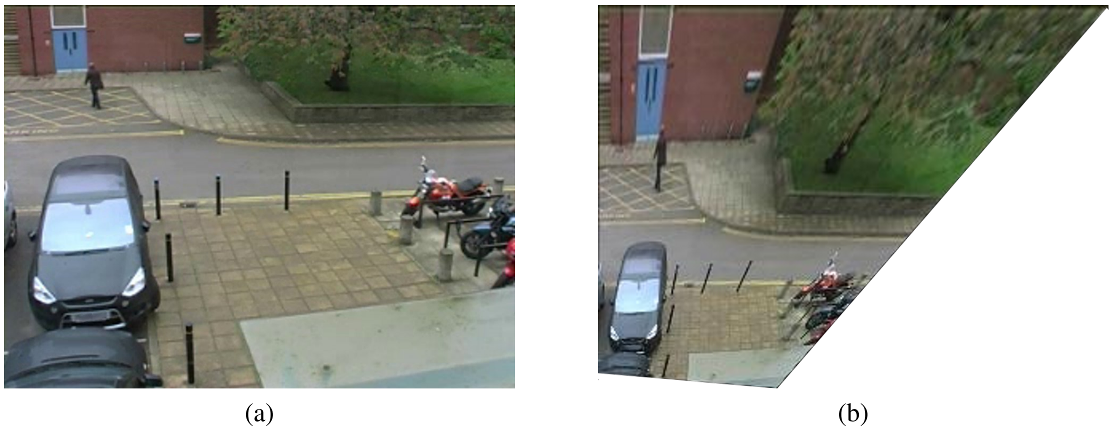


Figure 1.1: The effect of perspective distortion is particularly clear in image (a) (taken at the University of Leeds). If measurements of size or speed are required, it will be necessary first to correct for perspective distortion, as has been performed on image (b).

as labelling known points of reference, others focus on automated self-calibration. These related methods are discussed in detail in chapter 2. The aim of this work is to provide a reconstruction of the planes upon which motion is observed within a scene, through the use of ground-plane rectification.

1.2 The Problem of Densely Crowded Scenes

Many CCTV cameras overlook scenes that are likely to be crowded, some very densely, such as those outside stadiums or arenas. Often at such venues the majority of the crowd will leave at once, during which time its density will be extremely high.

Much of the existing literature (covered in detail in chapter 2) relies on geometric cues such as parallelism or the orthogonality of lines in the scene or on the ability to view an object of constant or known size in several positions of the image. However, in very densely crowded scenes it is unlikely that reliable detections of such lines or objects will be possible. Scene geometry is likely to be partially or entirely obscured and extreme levels of inter-occlusion will inhibit the use of the commonly chosen foot-head measurement. This inter-occlusion within a crowd is also likely to impede the ability to track any one feature for a prolonged interval. Whilst it is likely that in longer videos the crowd density will fall, in shorter sequences this is not the case and geometric data cannot be relied on in these situations.

Additionally, blob trackers are likely to fail to track individuals with much success due to the complexity of the view and those that rely on identifying individuals will fail as a large proportion of bodily features will be obscured. This provides a challenging problem to any trajectory based techniques and the solution presented in this work is to use a sparse feature tracker to obtain trajectories that are as long as possible. The methods described in this thesis need only relatively short trajectories in order to calculate an estimate of the plane orientation and are shown to apply in scenes of varying crowd density in section 3.6.

1.3 Aims and Objectives

Given the issues outlined above, it is the intention of this work to provide a framework for the estimation of the parameters describing the ground-plane or planes in pedestrian scenes, with a focus on those containing dense crowds. Having estimated the ground-plane or planes, it is trivial to back-project the image-coordinates of the tracked features onto the camera plane, thus correcting for perspective distortion.

The work shown in this thesis aims to tackle the issues outlined in the previous section using only the imaged-speed of moving objects as the source of information for rectification. Within a crowd, the motion of each individual is restricted such that they are likely to move together, as a group. It is not therefore, unreasonable to assume that tracked features within that scene will move at constant (or near-constant) velocity. Under this assumption, we can intuitively assume that the distance travelled by a tracked feature in 3D coordinates over a fixed time interval remains the same. Given enough of these measurements, we can recapture the distortions undergone by the scene during projection thus allowing for the removal of perspective distortion.

1.4 Novelty and Significance

This thesis presents three main contributions for the estimation of the parameters required for the rectification of imaged scenes from an uncalibrated camera. The single-plane and generalised multi-plane frameworks are unsupervised, and it is not necessary to make assumptions of scene geometry beyond the expectation in the former method that the world can be represented by a single, flat plane. Described below are the novel and significant contributions of the work presented herein:

1. A framework is presented for reconstructing the ground-plane in crowded scenes using the imaged speed of pedestrians alone.
2. The single-plane method is shown to extend into a world known to consist of two adjacent planes. Hitherto no techniques have been applied to such a problem using speed alone.
3. The two-plane framework is generalised to apply to any number of planes without prior knowledge of how many are present in the scene. No foresight of topology is required, with the layout of the planes with respect to each other being established within the framework.

1.5 Thesis Outline

This thesis begins in chapter 2 with a discussion of relevant research, commencing with an introduction to the underlying theory in the realm of camera calibration and reconstruction both in terms of single and multiple view geometry. It then continues to discuss applications of this theory with a focus on single-view reconstruction in pedestrian scenes before finishing with a detailed explanation of a conceptually similar method to the work in this thesis.

In chapter 3 the first novel contribution of this work is introduced – a speed-based reconstruction method for crowded pedestrian scenes in which the world is assumed to be realistically modelled by a single plane. First the elemental mathematical basis is introduced, before the incremental development of the method is outlined. Next feasibility studies on controlled, simulated data are discussed, after which experimental results on real-world data-sets are given, including a comparison to a similar method.

Chapter 4 explains the extension of this single-plane method into a two-plane domain and a framework for estimating the orientation and topology of the scene is introduced. Various methods for determining the boundary between planes are shown, along with an analysis of their robustness when the constant speed assumption is violated. Finally, improvements are suggested to improve the quality of the reconstruction.

Chapter 5 first proposes a technique for ensuring proper alignment of reconstructed planes before further generalising the framework such that it no longer requires prior knowledge of the number of planes within the scene. Results are presented on a range of simulated views of varying complexity.

Chapter 6 concludes the thesis with a summary of the presented work and its novel aspects. An explanation of the applicability of the work in the pedestrian domain is provided, before a discussion of other potentially applicable domains. Finally consideration is given to possible extensions to the work and future research directions.

Chapter 2

Related Work

When imaging a 3D scene using a standard monocular camera, the world coordinate system is projected onto the 2D image plane. This projection introduces distortions intrinsic to the camera, for example perspective distortion, the strength of which is affected by the focal length of the camera, and radial distortion due to the curvature of the lens. In various domains within the computer vision field; particularly the field of pedestrian behaviour analysis, within which this work is principally directed; it is necessary to examine properties of objects within the imaged scene. Some of these, such as colour or intensity gradients, may be less affected by the distortions described above, but many of the metrics used are affected including object size, location and speed. If one wishes to take measurements of these, the 2D to 3D transformation must first be reversed. This work focuses on the problem of calculating and reversing the perspective transformation, assuming that others such as the aforementioned radial distortion are negligible.

Whilst projection from three to two dimensions is trivial, any points lying upon the principal axis (i.e. the axis intersecting the optical centre and the principle point as shown in figure 2.1) project onto the same point in the image [129]. As such, information pertaining to depth within the image is no longer available. Without additional information, such as the camera's calibration parameters one cannot reconstruct the 3D point, hence the need for the techniques discussed in the remainder of this chapter.

This chapter will examine the existing literature for image rectification and 3D reconstruction, beginning with an introduction to the underlying model and mathematics used throughout much of the literature in section 2.1.1, then wider research into reconstruction from multiple views in section 2.1.2. The concept of plane to plane homographies is presented in section 2.1.3, before the idea of "Vanishing Points" is explained along with a description of how they can be used to rectify an image in section 2.1.4. Work into recovering structure from motion is discussed in section 2.2, before texture-based techniques are demonstrated in section 2.3.

Section 2.4 discusses methods for rectification of a scene using the geometry within it such as parallel lines, then section 2.5 discusses the use of objects of known size, with a particular emphasis on pedestrian scenes as that is the focus of this work. Finally, section 2.6 examines, in detail, a pedestrian motion-based approach that is conceptually similar to that presented in chapter 3 of this thesis, before an analysis of the applicability of existing methods is given in section 2.7. The chapter is concluded in section 2.8.

2.1 Common Theory in Scene Reconstruction and Image Rectification

Before discussing specific techniques for reconstruction, the common theory behind many of the techniques should be specified. This section will explain some of the more ubiquitous approaches to the reconstruction problem, before specific techniques are elaborated upon in the subsequent sections.

2.1.1 The Pin-Hole Camera Model and the Projection Matrix

Before one can compensate for the distortions encountered as a result of 3D to 2D projection, the underlying mathematical model must be identified. The assumption of a perspective-only transformation leads us to the choice of this model – hereafter this work assumes the projection undergone by the coordinate system follows the "*pin-hole camera*" model [53]. This embodies the way traditional cameras work and is illustrated in figure 2.1. This section will briefly introduce the pin-hole camera model before discussing the calibration parameters in more detail.

The aperture of the camera is referred to as the "optical centre" and the distance between this and the image plane is the focal length of the camera, denoted as f. Note that in

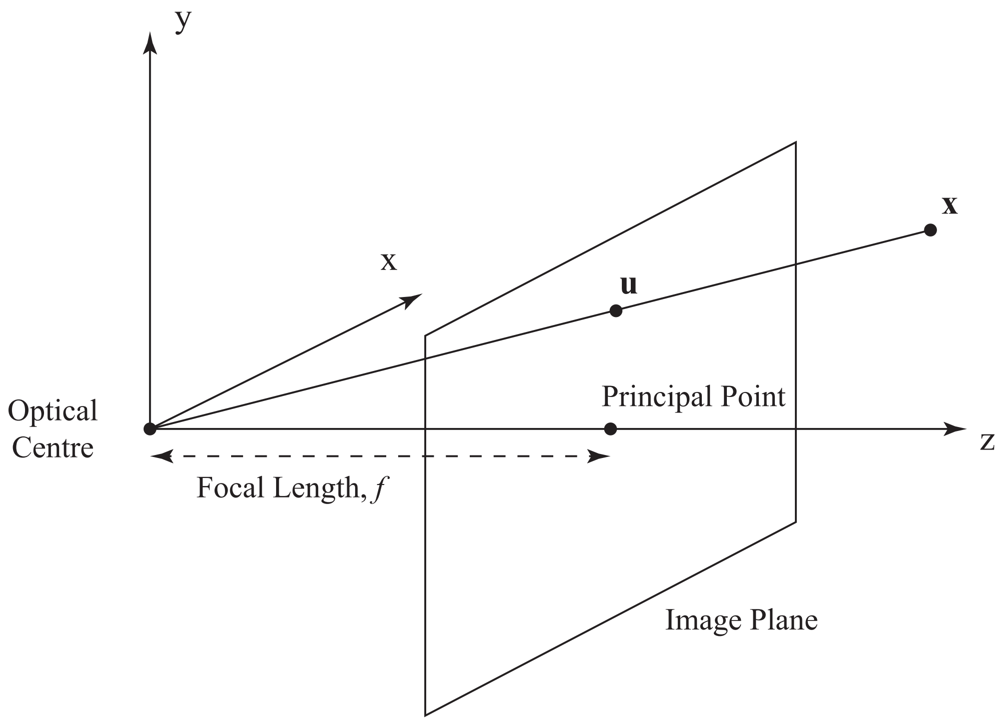


Figure 2.1: This work assumes the "*Pin-Hole Camera*" model as the basis for the transformation from 3D world coordinates to 2D image coordinates. Here we see the camera axes (x, y, z), with the principal point marked at the centre of the image plane and the focal length is defined as the distance from the optical centre to the image plane. The projection of the point **x** in the world onto the image plane is denoted as **u**.

real-world cameras the image plane is in fact behind the optical centre, rotated 180°, but it is common to simplify the model using a "virtual image plane" in front of the optical centre as shown here. Henceforth, references to the image plane in fact pertain to the virtual image plane.

The ray from the optical centre, perpendicular to the image plane is known as the "principal axis" and intersects the image plane at the "principal point". When a point **x** in the world is imaged by a camera, it is projected as the point **u** that intersects the image plane and the ray between **x** and the optical centre. By similar triangles, it is clear that the following mapping gives the projection of some homogenous point in the 3D camera coordinate system **x** into the image coordinate space [53]:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.1}$$

The pin-hole camera models the simple case where the principal point is in the centre of the image; however this may not be the case. Additional parameters can be added to the mapping to account for a translation in the more generalised case. Following the nomenclature of [53], the pair $(p_x, p_y)^\top$ represents the coordinates of the principal point. The previous mapping also assumes that pixels will be square, which may not be the case in all cameras. As such the scaling factor (m_x, m_y) is also introduced into the mapping. Finally, in rare cases the x and y axes of the camera's sensor (CCD) may not be perpendicular to the principal axis. In order to model this, a further "skew" parameter s, is introduced. These parameters are added to the previous mapping thusly:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} fm_x & s & p_x & 0 \\ & fm_y & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.2}$$

The matrix used in the mapping is known as the "camera calibration matrix" or "intrinsic matrix" and is commonly (though not always) denoted as **K**. In this work, we focus on reconstructing, up a scale factor, into the camera's own coordinate system, but in some applications it is useful to know the camera's position and orientation with respect to the world coordinate system – the "external parameters". Therefore, a further mapping is required, from the 3D world to the 3D camera coordinate systems. This is represented by the compound rotation matrix **R** and translation column vector **t**, both with respect to the world-coordinate origin. Let us denote a 3D point in world space as the homogenous vector $\mathbf{x}_w$ and the corresponding point in camera coordinates as $\mathbf{x}_c$. The mapping from world to camera coordinates is given as:

$$\mathbf{x}_c = [\mathbf{R}|\mathbf{t}]\mathbf{x}_w \tag{2.3}$$

It is now possible to map a world point to an image point using **K**, **R** and **t**. This combination of the internal and external camera parameters is known as the "projective matrix" and is commonly denoted:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{2.4}$$

By assigning correspondences between world and image points, one can solve for the parameters of the projection matrix and obtain complete calibration such that an image-point can be back-projected into world coordinates. There are a total of twelve unknowns within **P** – six from **K**, and three each from **R** and **t** – however, since it is only solved to scale, the system has only eleven degrees of freedom. As each correspondence gives an x and y constraint, a total of 6 correspondences is sufficient. Since it is very likely that correspondences will be subject to at least some noise, it is prudent to produce an over-determined system and solve using a direct linear transform (DLT) [53]. Given pure translation (that is, the principal axes of the two views are parallel, only differing by some displacement), with a minimum of 7 correspondences, a two-stage method involving first solving a linear system to obtain the external parameters, followed by a non-linear optimisation to find the internal and refine the external parameters allows for full calibration [124]. A similar framework can be used to calculate full calibration based on generalised motion of the calibration object [133].

The correspondence methods outlined above are sufficient in a supervised scenario, but often an automated or semi-automated solution is sought. Many of the methods in the forthcoming sections attempt to solve for either the internal parameters ("intrinsic calibration") or the full projective matrix ("full calibration") in an automated way, whilst others use a less formal approach. One example of this is the use of image-world homographies, which map points in image space to their world-space counterparts and are discussed in more detail in the section 2.1.3. First, however, the fundamentals of calibration and reconstruction from multiple views will be introduced in the following section.

2.1.2 Multiple View Geometry

When reconstructing the geometry of a scene or objects within it, it is often useful to image it from two or more views. This could involve multiple cameras imaging the same scene or a single moving camera. In static scenes, these two situations are geometrically equivalent [53], so the same methodology can be applied to both. In the following section, if reference is made to multiple cameras one can assume the same methodology can be employed given one camera in multiple positions.

Within this framework it is possible to reconstruct a 3D or corrected view of the scene without prior calibration of the camera or cameras producing the images as shown in [87, 122, 55] to name but a few. As the work in this thesis is primarily concerned with reconstruction from a single view, this section will offer only a brief overview of the core multiple view geometry approaches.

2.1.2.1 Epipolar Geometry

Consider a static scene, imaged in two overlapping views. Given the external calibration of the cameras, one can reconstruct the 3D position of imaged points [59]. A point in 3D coordinates $\mathbf{x}$ is imaged in the first camera at position $\mathbf{u}$ and in the second at position $\mathbf{u}'$. It is not difficult to imagine that some mapping exists between $\mathbf{u}$ and $\mathbf{u}'$. Rather than calculating the actual camera positions in the world, it is generally sufficient to calculate their positions with respect to one another. This knowledge, in combination with the internal parameters of the cameras, can be represented by the 3×3 "fundamental matrix", denoted in the literature as $\mathbf{F}$. If $\mathbf{x}$ is viewed in the cameras as $\mathbf{u}$ and $\mathbf{u}'$ respectively as illustrated in figure 2.2, the image points satisfy $\mathbf{u}'^{\top}\mathbf{F}\mathbf{u} = 0$ [53].

One can calculate $\mathbf{F}$ using the concept of "epipolar geometry". Denoting the two cameras as $\mathbf{C}$ and $\mathbf{C}'$, $\mathbf{C}$ projects onto the image plane of $\mathbf{C}'$ at the "epipole" $\mathbf{e}'$. Similarly $\mathbf{C}'$ projects onto the point $\mathbf{e}$ on the image plane of $\mathbf{C}$. The line between $\mathbf{u}$ and $\mathbf{e}$ is known as the "epipolar line" (again, see figure 2.2) and the line intersecting $\mathbf{e}$ and $\mathbf{e}'$ is called the "baseline".

The methods for calculation of $\mathbf{F}$ are well described in existing literature [53, 34, 114], so are only covered briefly here. The fundamental matrix has 9 degrees of freedom; however it is known to be singular, therefore its determinant is 0. This constrains the system to 7 degrees of freedom. Using 7 of the equations resulting from 4 point correspondences between the two images, one can solve for the fundamental matrix using a non-linear optimisation. With 8 or more constraints, (e.g. using additional correspondences) one can obtain a Maximum Likelihood estimate of it [55].

Each camera has an associated projective matrix and these can be calculated up to a projective ambiguity by decomposing $\mathbf{F}$ [114]. If $\mathbf{P}$ and $\mathbf{P}'$ are known, for example having been calculated using the methods discussed in section 2.1.1, one can calculate the projective reconstruction of a scene by triangulating the imaged points using the now known camera centres. Of course, this assumes zero-noise, which is unrealistic in practice. As such a minimisation of the sum-squared distance between the predicted and measured image positions of the triangulated point is employed [52].

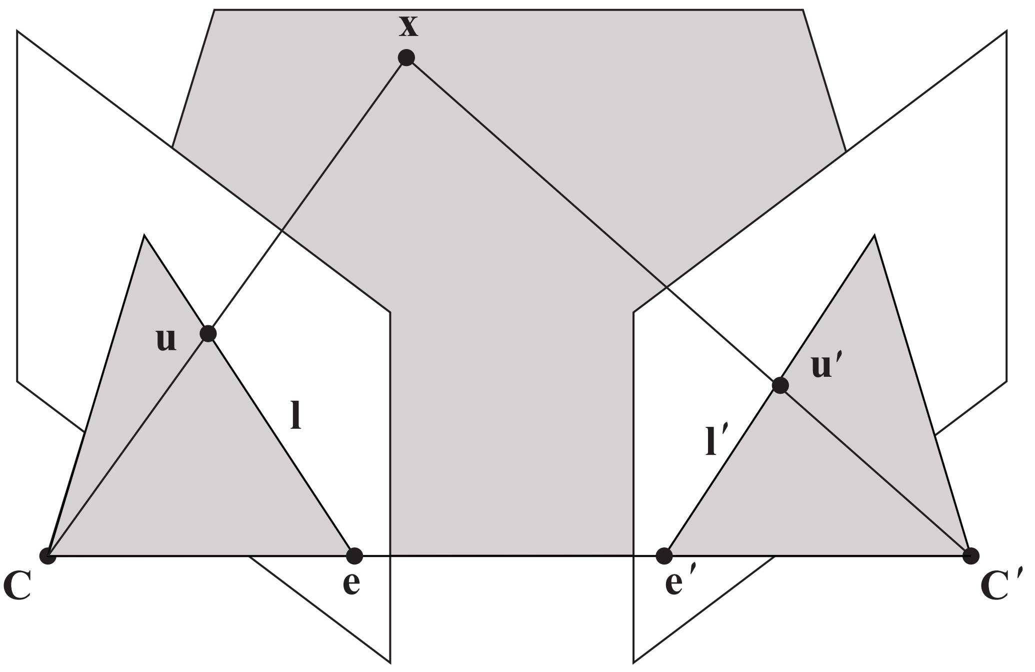


Figure 2.2: The point **x** is imaged by two cameras **C** and **C**′, producing a ray from **C** to **x** (similarly for **C**′). This produces the projections **u** and **u**′ respectively at the intersection between the ray and the image-plane. Each camera's centre projects onto a point on the other camera's image plane. These points are known as the "epipoles" and are denoted **e** and **e**′. The "epipolar lines" **l** and **l**′, are the lines intersecting **e** and **u** (respectively for **e**′ and **u**′). As an alternative definition for the epipolar line, consider the plane containing **x**, **C** and **C**′ pictured here in grey. This is known as the "epipolar plane" and intersects each camera's image plane on **l** and **l**′.

Provided the internal parameters of each camera, $\mathbf{K}$ and $\mathbf{K}'$ are known, the "essential matrix" can be found. This encapsulates the relative pose of the two cameras and is found thusly:

$$\mathbf{E} = \mathbf{K}'^{\top}\mathbf{F}\mathbf{K} \tag{2.5}$$

Obtaining the epipolar geometry in scenes with little overlap can be complex as the problem can become ill conditioned. Given certain scene assumptions, the infinite homography – the homography relating points at infinity between two image planes – can be used to supplement correspondences to allow for calculation of $\mathbf{F}$ [99]. These assumptions relate to the "vanishing points" of an image and are described in more detail in section 2.1.4.

2.1.3 Homographies

Commonly, scenes being viewed in surveillance applications involve a flat ground-surface which can clearly be modelled as a plane (the "ground-plane"). Let some point $\mathbf{v}$ be observed in this plane by a camera, creating the projection $\mathbf{u}$ in the image-plane of the camera. The mapping of $\mathbf{v}$ to $\mathbf{u}$ can be modelled as a 2D projective transformation from the ground-plane to the image-plane, given by the 3×3 non-singular matrix $\mathbf{H}$ known as a homography:

$$\mathbf{u}' \simeq \mathbf{H}\mathbf{u} \tag{2.6}$$

This mapping can apply equally to any two constructs with equal dimensionality. Another common use for homographies is mapping images from two views to each other such as those in figure 2.3. Such a technique is commonly used in image-stitching and mosaicing (stitching of multiple images) [62, 119, 15].

2.1.3.1 Calculating the Homography with Point Correspondences

Calculating the homography matrix is a well researched problem. A minimum of four point-correspondences between the two planes is necessary [53]. Given precise correspondences, one can use Singular Value Decomposition (SVD) [24]. Let $\mathbf{H}$, from equation (2.6), be represented by the vector $\mathbf{h} = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^{\top}$. For

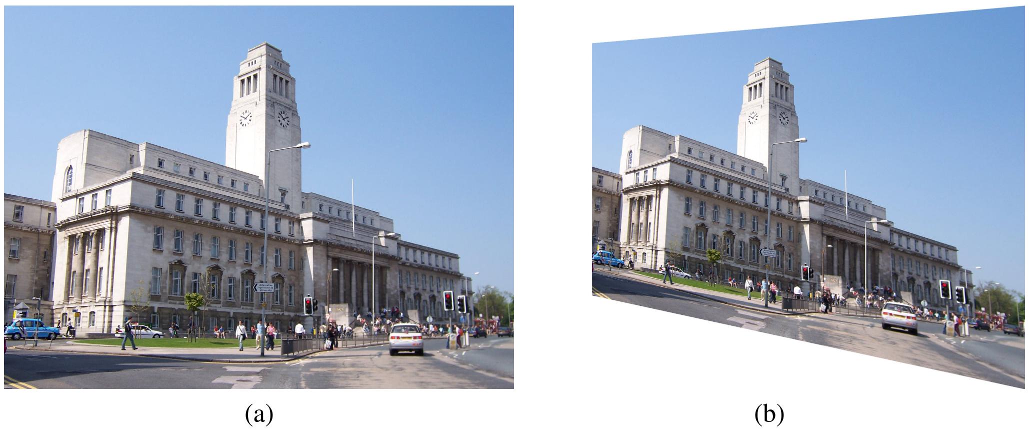


Figure 2.3: Image (b) is the result of applying some 2D projective distortion, modelled as a 3×3 matrix, to image (a).

some set of n points, we have the relationship $\mathbf{Ah} = \mathbf{0}$, where $\mathbf{A}$ is a $2n \times 9$ matrix. For each point correspondence $\mathbf{u}_i \leftrightarrow \mathbf{u'}_i$ we obtain two rows of $\mathbf{A}$:

$$\begin{pmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_ix'_i & -y_ix'_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_iy'_i & -y_iy'_i & -y'_i \end{pmatrix} \tag{2.7}$$

The vector $\mathbf{h}$ that minimises the residuals of $||\mathbf{Ah}||$ is given by the eigenvector with minimum eigenvalue of $\mathbf{A}^\top\mathbf{A}$, which can be found by decomposing $\mathbf{A}$ into three matrices by SVD, $\mathbf{A} = \mathbf{U\Sigma V}$. Where $n = 4$, $\mathbf{h}$ is simply the null-vector of $\mathbf{A}$.

This method can be problematic in real-world situations as inaccurate measurements can have a distinct effect on the result. Instead, consider the likely situation that there are many potentially noisy correspondences between the two planes, generated using some feature detector and some matching algorithm as in figure 2.4. The RANSAC algorithm [38] works by randomly sub-sampling sets of input data and selecting those that best fit some model, in this case a homography between the two planes. For each set of correspondences, the homography is calculated and the number of inliers determined. The homography with the largest set of inliers is chosen [53, 72, 75]. The result can then be refined using a Maximum Likelihood estimation over all inliers, before a "guided matching" is produced, by searching for more matches based on the homography estimate [53].

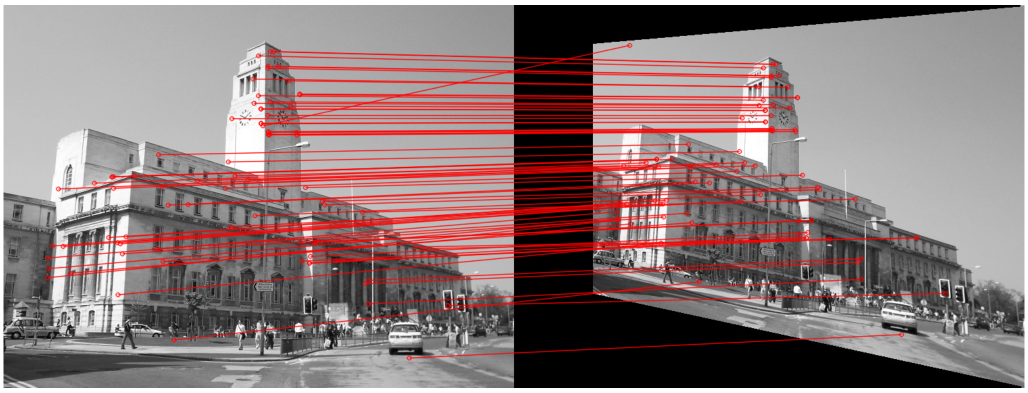


Figure 2.4: The homography between two images can be calculated using point correspondences. Here, SURF Features [8] are detected and matches calculated using FLANN [93] matching. Not all matches are correct so RANSAC is employed to maximise inlying matches that can be used in calculation of the homography, before a Maximum Likelihood estimation is used to refine the result.

2.1.3.2 An Alternative to Point Correspondences

The homography can be decomposed uniquely into three matrices [76]. These are the "pure projective" matrix **P**, the 'affine" matrix **A** and the "similarity" matrix **S**:

$$\mathbf{H} = \mathbf{SAP} \tag{2.8}$$

These matrices, and therefore the homography, can commonly be calculated in man-made scenes using pairs of parallel lines using the concept of "vanishing point" theory. The following section briefly introduces vanishing point theory, whilst applications of its use in image rectification and reconstruction are described in sections 2.4 to 2.6.

2.1.4 Vanishing Points and the Vanishing Line

When parallel straight lines in the world are imaged, they become distorted such that they converge (provided they are not parallel with the x-axis of the image plane), a phenomenon observed in the art world for hundreds of years. The point at which they converge is known as the "vanishing point". Vanishing points can be used to obtain a full camera calibration [16, 126] or a homography between the ground-plane and the image-plane [76]. The position of vanishing points is clearly dependant on the direction of the parallel lines in the world – lines of one direction will form one vanishing point, lines in

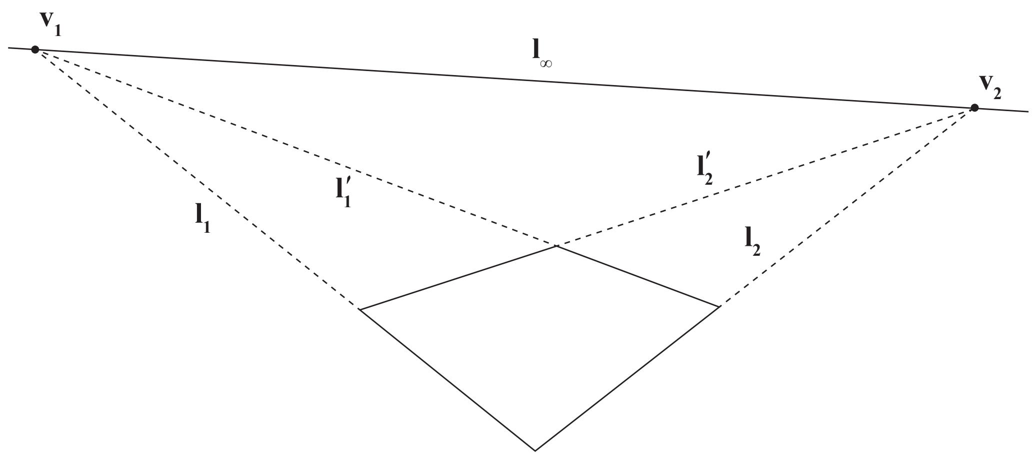


Figure 2.5: When imaged under perspective projection, lines that are parallel in the world become non-parallel. The point at which they converge is known as the "vanishing point". Given two such vanishing points, one can calculate the line at infinity or "vanishing line", denoted $\mathbf{l}_\infty$. In the case above, lines $\mathbf{l}_1$ and $\mathbf{l}'_1$ were parallel in world-space, as were $\mathbf{l}_2$ and $\mathbf{l}'_2$ but in the image they converge at the points $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively.

another form a second. Exploiting this fact, one can obtain the equation for the "vanishing line", otherwise known as the line at infinity, as show in figure 2.5.

Let us first assume that a number of parallel lines have been found in an image. It is entirely feasible that more than two lines of the same direction will be obtained. Given the presence of realistic detection noise, there is unlikely to be a closed-form solution to obtain the vanishing-point. This can be addressed by simply taking a weighted mean of the proposed vanishing point locations [16]; however a Maximum Likelihood estimation based on reprojection error is likely to yield better results [76] a technique which is prevalent throughout much of the vanishing point literature. Alternatively, one can cluster lines based on their potential vanishing point locations [46] or use RANSAC to iteratively filter sets of lines onto vanishing points [98, 105].

Thus far this section has focused only on finding two vanishing points which lie on the vanishing line. By finding a third vanishing point, orthogonal to the other two, it is possible to achieve full metric rectification [26, 46, 105, 95], or indeed full calibration [73, 67]. Vanishing points are particularly applicable in man-made scenes due to the prevalence of parallel lines in three orthogonal directions on scene features such as buildings – commonly referred to as the "Manhattan Assumption" [23].

2.1.4.1 Decomposition of the Homography Matrix

The vanishing line has a particularly useful property in scene reconstruction – it allows the reduction of the projection transformation to an affine one [21, 35, 76]. This gives a pseudo-frontal view of the scene, meaning that world-parallel lines are now parallel in the transformed image.

Given two pairs of lines which are parallel in the world, each pair pointing in a different direction. Denote these $(\mathbf{l_1}, \mathbf{l_1}')$ and $(\mathbf{l}_2, \mathbf{l}_2')$ respectively. $(\mathbf{l_1}$ and $\mathbf{l_1}')$ converge in image-space at the vanishing point $\mathbf{v}_1$, whilst $(\mathbf{l_2}$ and $\mathbf{l_2}')$ converge at $\mathbf{v}_2$. The two convergence points form the vanishing line, $\mathbf{l}_\infty = (l_1, l_2, l_3)^\top$, where $l_i, i \in (1,2,3)$ represents the three coefficients of the vanishing line. Using the formulation of [76], the pure projective matrix, $\mathbf{P}$ is now:

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{pmatrix} \tag{2.9}$$

In order to perform metric rectification, the affine matrix must be found, which takes the form [76]:

$$\mathbf{A} = \begin{pmatrix} \frac{1}{\beta} & -\frac{\alpha}{\beta} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

The parameters α and β give the image of the "circular points", which lie upon $\mathbf{l}_\infty$. These are a pair of fixed, complex conjugate points at infinity, at which every circle intersects $\mathbf{l}_\infty$, hence their name. A particularly useful property of the circular points is that they are invariant to Euclidean transformations [53], meaning they can be used to estimate the projective transformation undergone by the scene. Given any two of the following three constraints, one can establish the location of α and β and calculate the affine matrix [76]:

1. A known angle between lines
2. Equality of two (unknown) angles
3. A known length ratio

Commonly, right-angles are used in (1) and (2) as they frequently occur in man-made

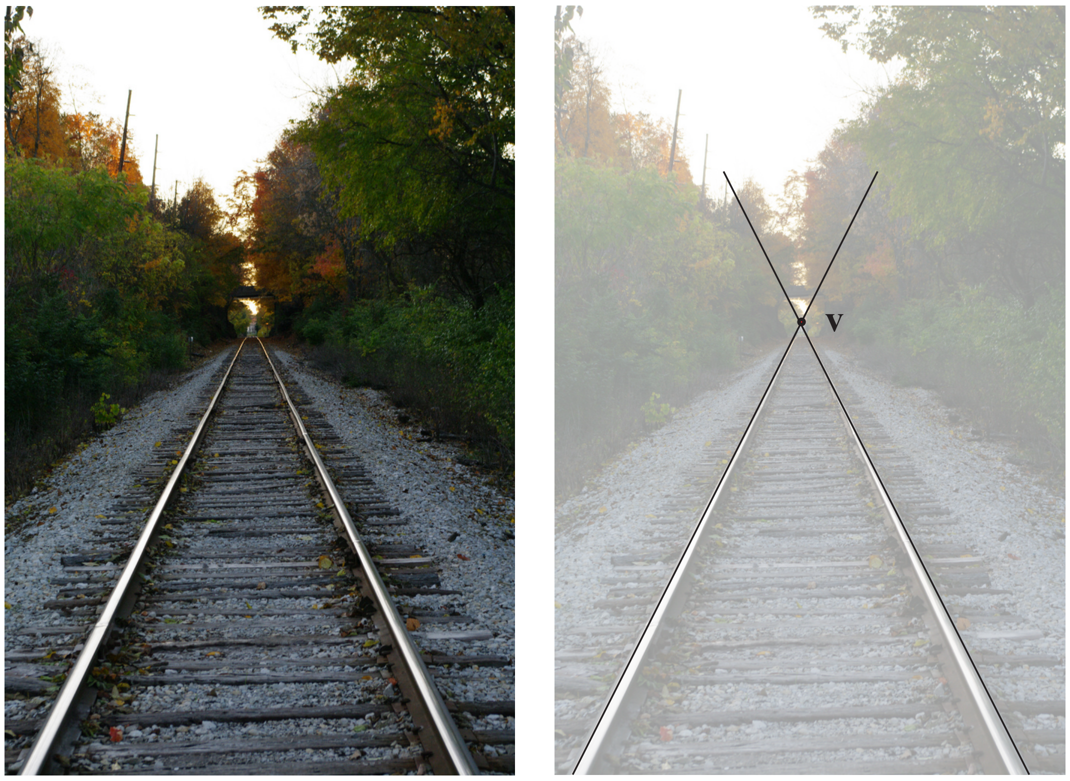


Figure 2.6: In some scenes only one vanishing point can be found reliably such, as the example above. Such a scene is referred to as having “one-point perspective”. In such cases additional information such as scale ratios must be used to obtain the vanishing line.

scenes [23]. Alternatively a measuring object of known length on the ground-plane could be utilised to satisfy (3).

Finally, the similarity matrix represents rotation $\mathbf{R}$, a translation $\mathbf{t}$ and an isotropic scaling s [76]:

$$\mathbf{S} = \begin{pmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tag{2.11}$$

It is sufficient for most applications to solve up to metric rectification, and as such, existing methods commonly only solve up to $\mathbf{A}$, disregarding the final similarity matrix.

In certain degenerate situations, there may only be one vanishing point in the scene, an example of which is shown in figure 2.6. This is what is known as “one-point perspective”. In these cases, one may still obtain the vanishing point as above, or alternatively a model of human attention can be used to robustly discover the location [117]. Whilst typically the determination of the vanishing line requires two vanishing points, one can use a scale ratio in combination with the vertical vanishing point to obtain the vanishing line [70].

In this section, the theory common amongst many approaches in the literature has been discussed. The following sections will examine specific techniques in more detail.

2.2 Structure from Motion

The human interpretation of the world relies on several cues to understand its structure. In particular, the motion of our viewpoint allows us to infer much about the relative depth and position of objects within the world. Structure from Motion (*SfM*) could be considered an interpretation of this in that the techniques within the field rely on motion information to derive 3D geometry from 2D views of a scene. Rigid structure from motion assumes that any observed movement occurs without deformation, an example of which would be a camera moving across a scene in front of a building. Consider such a situation and suppose the internal parameters of the camera are known. At each frame a new view of the scene is produced, allowing for calculation of the essential matrix, which can then be decomposed to give the position and orientation of the new view with respect to the last. Now the 3D structure of the building as imaged thus far can be determined simply by triangulating the imaged points from the two views [53]. This can be performed equivalently for three views using the trifocal tensor [53]. It is common to perform this calculation each time a new frame is added, using the latest two or three views.

Due to the noisy nature of the imaging process, this initial alignment may not be sufficient. Therefore an optimisation step known as bundle adjustment is necessary, which simultaneously optimises both the 3D positions of the reconstruction and the camera parameters [123]. This approach will result in an point-cloud in 3D space, from which surface orientations can be roughly estimated, allowing the creation of a dense depth map [94].

The above is a generic technique for sequential SfM; many others exist in the literature. Using epipolar geometry to calculate the relation between views is a prevalent theme [101, 54]. Once the camera calibration has been found in the first image-pair, the reconstructed points can be augmented and updated, for example using the Kalman filter [9] as new ones are observed, with correspondences between the previous frame and the new one being used to calculate the updated relative pose and orientation. In some cases assumptions on camera motion may be used to improve the quality of the reconstruction [39]. In many

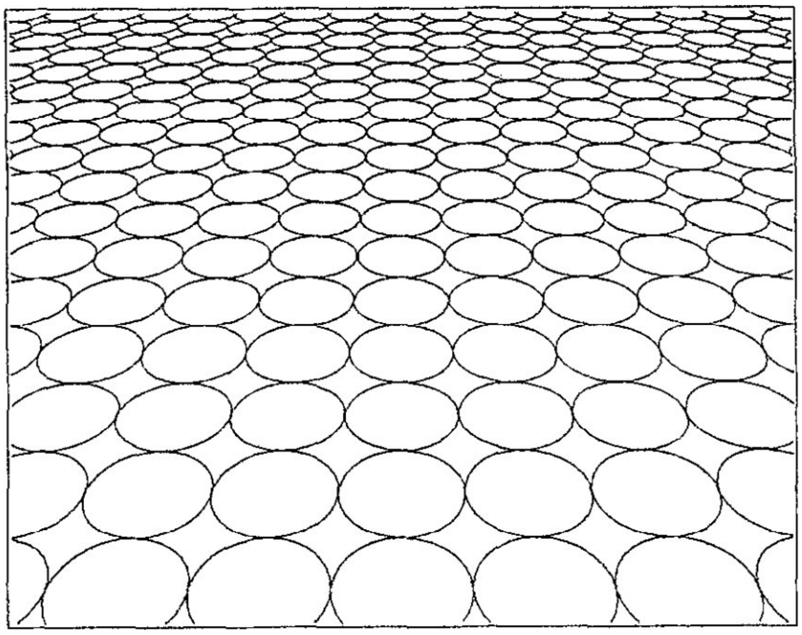


Figure 2.7: Just as the outline of an object deforms with perspective, so does the texture upon it. Here a simulated circular texture undergoes perspective projection [42], causing ellipses to become increasingly non-circular as the distance from the camera increases.

applications the scene may not be static, with objects moving separately to the camera. By assuming the camera moves at constant speed, it is possible account for motions such as these and capture the path of camera motion and of the moving objects within it under orthographic projection or weak perspective projection [51].

Alternatively, by assuming the scene to be modelled in a piecewise-planar fashion, one can use image-image homographies given projective rectification of the cameras and an assumption that surfaces can be modelled as hinged planes of each other [5].

2.3 Structure from Texture

It has long been observed that texture can provide cues towards the transformation undergone by an object during the imaging process – a concept known as the "texture gradient" [44], an example of which can be seen for a simple circular texture in figure 2.7. Indeed, this is thought to be another of the key indicators used by human vision to establish the orientation of an object in the world [118]. The imaging transformation may be simplified to a pure rotation of a scene object and can be solved given the assumption of an "isotropic" texture, that is edges in the texture are uniformly spaced [128]. Texture gradients can instead be used to generate a texture density map enabling the scene to be modelled as a planar surface [63]. This information can be supplemented with the effect of haze upon a distant texture and texture energies [114] to obtain a more accurate ground-plane estimation [20]. A more complete rectification can be obtained by calculating the homography between a planar surface and the image using “low-rank textures” [92] – textures of low-rank when the projective transformation is rectified [134].

In lieu of a global transformation approach, the surface architecture of an object may be obtained in terms of its surface normals [42, 79, 85, 120].

If the texture of the desired object is assumed to fit some lattice, one can search for the optimal distortion of that lattice over the objects in the scene using Markov Random fields [96], thus allowing for the modelling of more complex, non-planar objects. Lattice-fitting can be extended to allow for full-scene reconstruction from a single image. Commonly a super-pixel segmentation of the scene is found representing salient regions of an image [58], which can then be used to fit a mesh constructed of many small polygons [110, 109].

Textures need not be static in the scene to provide information for affine rectification – it is possible to use dynamic textures, i.e. textures which change over time, using optical flow in the case of translational or homogenous dynamic textures [111].

2.4 Reconstruction from Geometric Features

Given some model of the shape of an object in the world, it is possible to infer the transformation it has undergone during the production of the image. This may be a loose assumption, such as the Manhattan Assumption, or a more restrictive knowledge of the exact shape of an object. This section discusses recent approaches to image rectification and camera calibration under such constraints.

As previously discussed in section 2.1.4, the use of the Manhattan Assumption allows for the calculation of vanishing points, thus permitting rectification of an image up to at least a similarity, or indeed full calibration.

This is a technique frequently used in indoor scenes, where one can expect to see chains of walls, connected to ceilings and floors, the edges of which allow for the determination of three orthogonal vanishing points followed by semantic segmentation of walls, floors and ceilings [71, 30, 40]. This can either be done using a single image [71, 30] or vanishing point information augmented with motion and stereo data in a Bayesian framework, to allow a more accurate segmentation [40].

Traffic scenes also provide strong features for perspective correction, although they often only exhibit 1-point perspective [45]. If the camera is known to be parallel to the ground, generating the line at infinity and thus a second vanishing point is trivial, otherwise the width of the road lane can instead be used to constrain the system [45]. Alternatively, known distances-ratios between road lines can be used to extract the second vanishing point, allowing for the lane edges to be identified [56].

Alternatively, if curvature of a constant-width road section is observed, one can solve for rotation about the x-axis of the camera, providing the ability to rectify for perspective alone [84, 83].

Vehicle wheel bases form ideal vanishing point constraints [48,49] – one can easily detect the 2 primary directions of the wheel base using a Histogram of Oriented Gradients, giving two vanishing points. These are also known to be orthogonal in the world, meaning that for each vehicle observed an extra constraint is placed using the rule of known-angles [76] allowing for the calculation of the affine matrix [135].

When a circle in the world is imaged by a camera, the distortion it undergoes produces an ellipse in the image. Using such ellipses it is possible to calculate the orientation and position of circular objects in the scene [32] or indeed the homography between the world and image planes [80, 19, 47].

2.5 Reconstruction Using Known Sizes or Ratios in Pedestrian Scenes

In many scenes, the size of an object in the world may be known or can be assumed, such as the height of a pedestrian. By observing this object in many positions, one can obtain an estimate of the perspective transformation in terms of a rough ratio [17]. Alternatively, one may manually calculate the expected height distribution at a given pixel [18] or the model of the ground-plane [103, 116, 14] to normalise object size.

If a full view of a pedestrian is available, the line between the feet and the head is often assumed to be approximately vertical and can be used in calculation of a homography. It is possible to obtain three vanishing points from foot-head measurements taken at various locations across the scene [82]. Supplementing this with the knowledge of the object's height allows for full metric rectification. However, at low viewing angles the vertical vanishing point tends towards infinity making estimation of it extremely sensitive to noise. To help minimise the effect of this problem, the additional assumption that pedestrians will move at constant speed can be used to augment foot-head measurements in a Bayesian framework [68]. This assumption can be used alongside an extended model including shadow information to allow computation of not only the camera parameters (both internal and external) but also the lighting position [106].

Alternatively, provided the scene is imaged from two views with known internal parameters, it is possible to calculate the relative pose of the views avoiding the use of vanishing points altogether [33]. This foot-head approach can be extended to calculating the homology between the feet and head to perform full intrinsic and extrinsic calibration [90, 107] or "harmonic homologies" across multiple frames [66, 64]. The latter technique can similarly be modelled using epipolar geometry [65].

A somewhat distinctive approach to those discussed in this section is to use the average size of faces imaged in the scene to generate a normalisation map [28], thus providing a method for correcting observed sizes to scale.

2.6 Using Speed to Rectify Images

Given a consistent time interval in video footage, the projected speed of an object varies just as its size does in the image [116]. This allows for the rectification of a scene using an assumption of constant velocity.

One method warrants particular examination due to its conceptual similarities to the work in this thesis. Bose and Grimson [12], use vanishing point geometry in conjunction with constant-speed trajectories (detected using the Stauffer-Grimson tracker [115]) in order to obtain affine and metric rectification of the ground plane. Constant speed paths are determined by first assuming that any given trajectory is of a constant speed. The 1D homography between hypothesised trajectories of constant speed and the observed trajectories is then calculated and when the trajectory is back-projected, a small error indicates constant speed. The inverse of this homography can then be applied to the world vanishing point to obtain its imaged counterpart. With two such paths the vanishing line can be calculated, providing projective rectification. Affine rectification is then estimated using the known length ratio rule of [76].

The paper offers good results in what appear to be reasonably simple scenes, with sparse motion. The use of the Stauffer-Grimson tracker would hinder the effectiveness of this system in a dense-crowd situation, as individual blobs would be hard to detect, and particularly difficult to track for any length of time, which would impede this method due to its need for long paths taken over 25 frames. In a crowded scene, motion will most likely be slow, as such the length of the path may not be sufficient since tracking is necessary across a large section of the scene. It is also difficult to maintain a tracker identity due to the high inter-occlusion of crowd members.

2.7 Applicability of Existing Methods

This work deals largely with crowded scenes. Introduced in the preceding sections were the concepts of structure from shape, motion and texture, amongst others. Other reconstruction techniques exist, but are not closely related to the work presented in this thesis. Shape from shading is somewhat related to shape from texture and is based on the use of gradual shadowing on an object's surface to determine its orientation [132, 61]. Depth information can also be obtained from focus using specialised machinery [114], or defocus by imaging the scene at various focal lengths and using blurring to infer depth [114]. Photometric stereo uses multiple lighting conditions when imaging an object to infer surface data [125, 131]. Shape from shadow uses the outlines of shadows and the directionality of light to supplement information from a single camera [129, 29].

Many of the methods discussed in the previous sections, particularly those requiring foot-head measurements will be unable to function in such scenes due to the high levels of inter-occlusion between agents in the scene. Consequently these are largely inapplicable in this domain. Whilst the use of face-sizes alone bypasses the need for full views of agents, it does require frontal views throughout the scene which may not always be available. Additionally, the formulation given in [28] is a coarse normalisation across the entire image and insufficient to model scenes with any more than one plane of interest.

Scene geometry-based methods suffer from similar drawbacks – when a dense crowd is present in the scene, it is unlikely that reliable line-estimates will be accessible due to occlusion. This is highly problematic for any methods based on parallel line projection for the calculation of vanishing points. Similarly, whilst road lines may be present in pedestrian images, one cannot guarantee this and many of the techniques found in the literature are intended to be used from a vehicle's perspective.

The method of Bose and Grimson shows promise in our domain, but its requirement for prolonged tracking and use of a blob tracker are likely to cause accuracy to diminish when dense crowds are present within a scene. However, given its similarity to the method outlined in this work, comparative experiments have been performed to examine its performance against this work and can be seen in section 3.6.1 of chapter 3.

2.8 Conclusion

This chapter has discussed several aspects of the theory underpinning image rectification and 3D reconstruction relevant to the work in this thesis, and applications thereof.

The chapter began by discussing single-view projective geometry, specifically the pinhole camera model assumed throughout this work. Then the fundamentals of multiple view geometry were discussed, followed by an explanation of homographies and vanishing point theory. Applications of these areas were then illustrated in sections on Structure from Motion and Structure from Texture. The following three sections discussed applications more closely aligned to the work within this thesis, largely in the pedestrian domain (although some applicable methods from the traffic monitoring literature were included). The final section discussed the use of speed to rectify images, which to our knowledge is a relatively unexplored area given a static camera, and an approach similar in concept to our own was critiqued.

Chapter 3

Single Plane Reconstruction from Motion

In chapter 2, it was shown that it is possible to construct the 3D scene using information gleaned from measuring objects of known real-world size at various positions within the image. The work in this chapter operates on the basis that it is equally possible to use measurements of object velocity in order to reconstruct the 3D plane upon which observed agents are moving.

The work presented here makes the assumption of an uncalibrated, stationary "natural camera" [77], namely zero-skew and square pixels. Additionally we assume the principal point falls at the image centre, thus in the intrinsic calibration matrix, $(p_x, p_y) = \mathbf{0}$ (see chapter 2, section 2.1.1 for further details). Such realistic restrictions are not uncommon in the domain of reconstruction and rectification [25].

Throughout this chapter it is assumed that all objects move on a single, linear plane and that each observed object is moving at constant (or near-constant) velocity. First, early work using only motion vectors is discussed in section 3.1, in which all objects are assumed to be moving at the same velocity. Section 3.3 details a development of this idea, in which the necessity for all agents to move at the same speed is removed. Finally, section 3.4 describes the final iteration of this work, in which the error function used to represent the constant speed assumption is reconsidered. Rather than measuring the direct difference from a fixed speed in the world, the spread of speeds along a trajectory is minimised, offering improved robustness on larger sets of input data and particularly those with higher variation in speed between agents.

3.1 Relating Speed to the Ground-Plane

A set of velocity measurements are taken either using the simulation methods discussed in sections 3.2.2 and 3.3.2 or using the *KLT* tracker [81, 121, 112]. Further analysis of tracker choice and post-processing is given in section 3.5.

As discussed in the notation section (see front-matter), a plane is defined by its normal $\mathbf{n}$ and its distance from the camera d. Additionally, the plane-normal can be expressed in terms of "elevation" θ and "yaw" ψ rotations, about the x-axis and z-axis respectively of the camera coordinate system:

$$\mathbf{n} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} = -\begin{pmatrix} \sin(\psi)\sin(\theta) \\ \cos(\psi)\sin(\theta) \\ \cos(\theta) \end{pmatrix} \tag{3.1}$$

A full derivation of (3.1) is given in appendix A.

Given a ground-plane $\mathbf{n} \circ \mathbf{x} = d$ in world-space, a point upon it, $\mathbf{x} = (x, y, z)^\top$ and some α, given by equation (3.2) for focal length f; equations (3.3) to (3.5) show the back-projection of an imaged point, $\mathbf{u} = (u, v, 1)^\top$ onto $\mathbf{x}$. A visual representation of this model is given in figure 3.1.

$$\alpha = -\frac{1}{f} \tag{3.2}$$

$$x = \alpha u z \tag{3.3}$$

$$y = \alpha v z \tag{3.4}$$

$$z = \frac{d}{\alpha a u + \alpha b v + c} \tag{3.5}$$

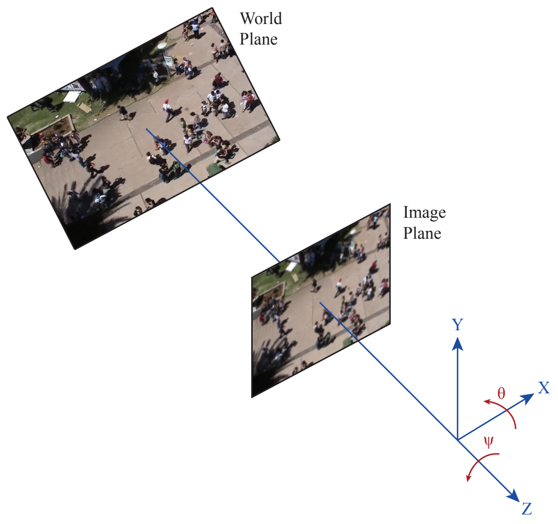


Figure 3.1: This illustrates the orientation of the ground-plane with respect to the camera axis (x,y,z), showing the ground-plane in world coordinates and the same transformed into image coordinates. The rotation parameters of the camera, in terms of elevation in the camera x-axis and yaw in the z-axis (θ and ψ respectively) are also given.

3.1.1 Speed as a Measuring Stick

An object's observed speed varies with perspective as do its other properties such as height [116]. Assuming the real-world speed of a tracked object is constant over some fixed time interval, measurements taken at multiple intervals can be used as the "measuring-stick" from which to gain an estimation of the ground-plane (to scale). Trivially, the speed of the motion of some tracked point in 3D camera coordinates can be found using the euclidean distance equation, shown for completeness below. Clearly, when the camera is directly above the plane, $(z-z')=0$.

$$D(\mathbf{x},\mathbf{x}') = \sqrt{(x-x')^2+(y-y')^2+(z-z')^2} \tag{3.6}$$

The system must transform 2D image coordinates onto a 3D plane given by $\mathbf{n} \circ \mathbf{x} = d$. From equations (3.3) to (3.5), it is clear there are four parameters in the system: the two orientation parameters defined by $(a,b,c)^\top$ as per equation (3.1), the distance of the optical centre from the plane d and α. As this work assumes no knowledge of the size of objects, rectification can only be performed up to scale. Therefore, the observed world-velocity $\hat{l}$ is fixed in order to set the scale of the solution and an estimate must be found for the plane parameters $\mathbf{n} = (a,b,c)^\top$ and d in addition to α. Camera-height, d is encoded into the orientation parameters as per equation (3.8).

Given a pair of points in the image, $(\mathbf{u},\mathbf{u}')$, the intention is to back-project them to the world-coordinates $(\mathbf{x},\mathbf{x}')$ and obtain the distance between them in the real-world $L_1(\mathbf{u},\mathbf{u}')^2$. Substituting the projection equations (3.3) to (3.5) into the Euclidean distance equation (3.6), the direct relationship between the velocity of tracked points in the world and its projection into image coordinates can be obtained, given some set of parameters $(\alpha,\hat{a},\hat{b},\hat{c})$, where $\hat{a}$ is an estimate of the true a (similarly for $\hat{b}$ and $\hat{c}$).

$$L_1(\mathbf{u},\mathbf{u}')^2 = \alpha^2\left(\frac{u}{\gamma}-\frac{u'}{\gamma'}\right)+\alpha^2\left(\frac{v}{\gamma}-\frac{v'}{\gamma'}\right)+\alpha^2\left(\frac{1}{\gamma}-\frac{1}{\gamma'}\right) \tag{3.7}$$

where,

$$\gamma = \alpha u\hat{a}+\alpha v\hat{b}+\hat{c} \quad \text{and} \quad \gamma' = \alpha u'\hat{a}+\alpha v'\hat{b}+\hat{c}$$

and

$$(\hat{a},\hat{b},\hat{c})^\top = \left(\frac{a}{d},\frac{b}{d},\frac{c}{d}\right)^\top \tag{3.8}$$

3.2 Motion Vectors and Constant Speeds

First, consider the situation where the strict assumption is made that the only moving agents within a scene are pedestrians. Furthermore, suppose that they all move at the same velocity (1m/s is observed to be approximately correct) and that all motion occurs upon the ground-plane. From this it can be surmised that any motion vectors observed within the scene have equal length in world coordinates, thus providing the “measuring stick” needed to rectify the scene.

In the manner of Dee [27], short linear (or near linear) motion vectors of moving points in the image are obtained over short time intervals, characteristically about one second. The KLT tracker [112] is used on video data taken from an uncalibrated, stationary camera. The aim is to derive the ground-plane parameters from these motion vectors with no prior scene knowledge other than an assumption of a real-world plane. The use of such motion vectors mitigates the problem of tracking loss in very dense crowds as long intervals of sustained tracking are not required. An example of the motion vector generation output (using the “students003” dataset, described in section 3.5) is shown in figure 3.2. In some cases, degenerate movements are observed, such as very slow motion or a rapid change of direction during the time-interval over which we track a feature. Whilst the former issue can be easily overcome by thresholding on minimum velocity, the latter may pose a problem during estimation as the change in direction will not be captured by the tracker resulting in motion that appears slower than it actually was.

Using the constant-velocity assumption, it is now possible to assess the fit of a set of plane parameters to the motion-vector data. As previously mentioned, the expected world-space velocity $\hat{l}$, must be fixed so the unit assumption is made: $\hat{l} = 1$. Therefore, any deviation from 1 can be classed as error. With this in mind, the error measure is defined as the sum-squared difference between the back-projected velocity and $\hat{l}$, over the set of input motion vectors, Υ.

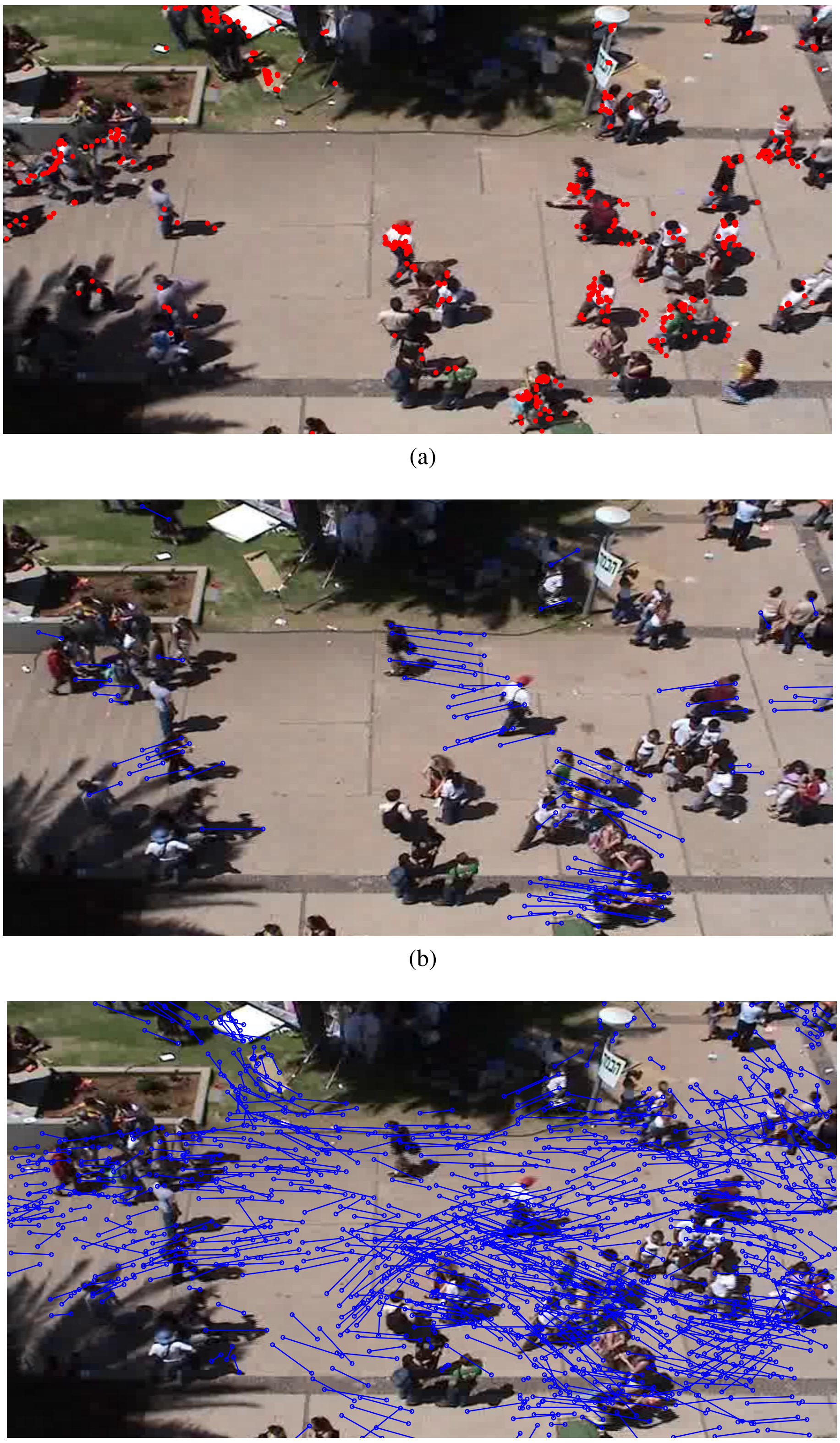
$$E_\Upsilon = \sum_{i=1}^{N_\Upsilon} \left(L_1(\mathbf{u}_i, \mathbf{u}'_i) - 1 \right)^2 \quad \text{where,} \quad (\mathbf{u}_i, \mathbf{u}'_i) \in \Upsilon \tag{3.9}$$

It is important to note the assumption that all motion vectors lie upon the ground-plane. In the vast majority of cases this will not be the case – feature points may be detected at various positions on an agent's body, be it the head or the feet. This is a source of error and the scale of this error is examined in detail in section 3.2.2.

In a real-world scenario, the actual distribution of velocities is likely to vary. In order to determine the likely levels of variation, the distribution of velocities for motion vectors back-projected into the camera-coordinate system (using ground-truth camera parameters) is given. These motion vectors are produced from manually ground-truthed trajectories taken from bounding box centroids [3], so give an indication as to the motion of individual people in the scene. Whilst this differs from the tracking that can be expected from the KLT feature tracker, it does give a reliable indication as to the variation in speed. From figure 3.3, it can be seen that the distributions of velocities in the scenes presented are roughly Gaussian, although the spread within them varies. In those scenes in which the spread is low, the system presented here should function well; however if the spread is high, as in figure 3.3d, the likelihood of choosing enough motion vectors of equal velocity is low, hindering the approximation process.

3.2.1 Sensible Motion Vector Selection

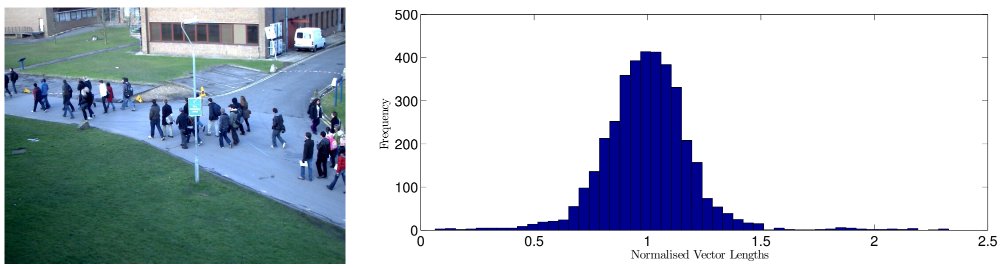
After running the tracker on a video, the size of Υ will generally be very large, typically of the order of 10,000 motion vectors per minute of 640x480, 25 FPS video for a moderately


(a)

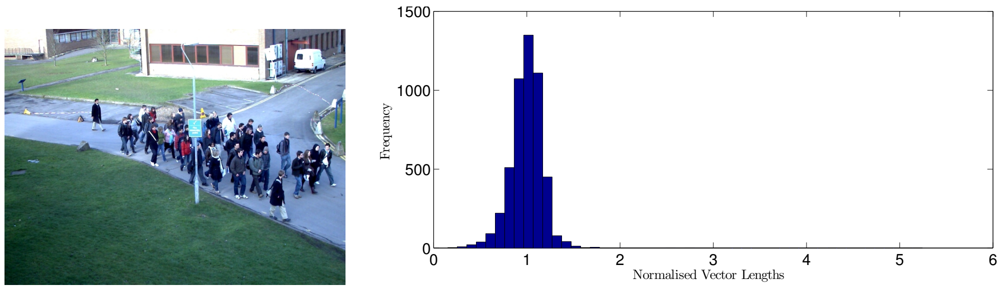
(b)

(c)

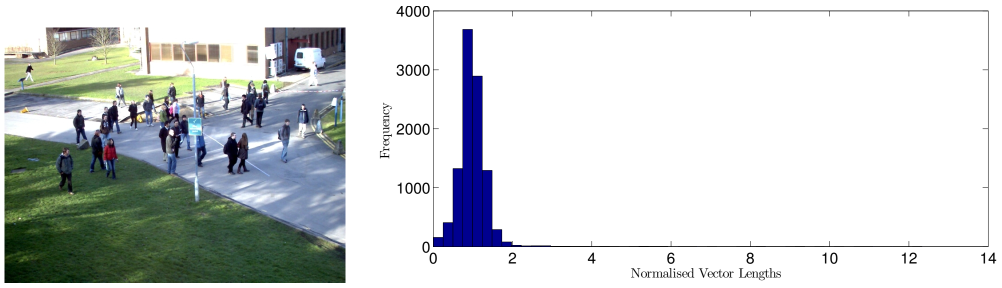
Figure 3.2: The KLT Tracker [112] is used to collect short motion vectors representing the motion of tracked features in the scene over some fixed time interval. (a) shows one frame of video with the detected salient points highlighted in red. (b) then offers a set of motion vectors resulting from the motion of tracked features and (c) the final set of vectors by the end of the video.



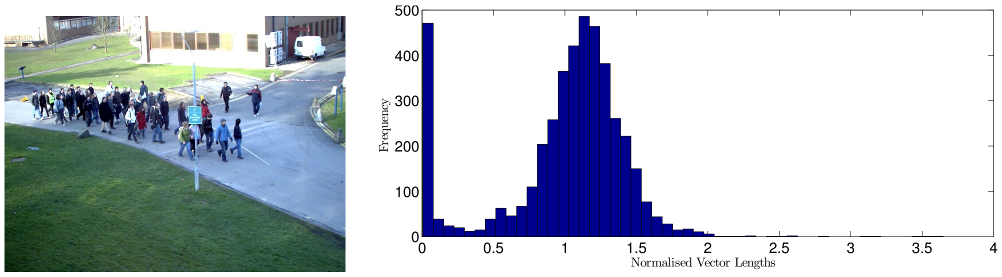
(a)



(b)



(c)



(d)

Figure 3.3: In order to assess the likelihood of selecting motion vectors of approximately equal velocity in the real world the distribution of velocities was calculated using manually labelled trajectories back-projected using ground-truth camera parameters. We see that in many cases, the speed has a somewhat Gaussian distribution although in cases such as (b) and (c) this has a strong frequency peak indicating that it should be possible to find combinations of motion vectors with approximately equal speed. In (d), there is an additional peak at near-zero speed - these motion vectors can be readily discarded using a minimum velocity threshold.

crowded scene (it is common for many features to be tracked on each person). Additionally, in scenes with strong directional lighting, it is likely that the movement of shadows will create additional tracked features for each person.

As the system contains only four unknowns, there is a risk of drastically over-constraining the system should the entire set be used. This work assumes that all tracked points move at constant velocity in world-space. Realistically this is unlikely to hold at all times, meaning that using the whole of Υ is likely to result in the system fitting to invalid data. Therefore, an approach akin to Random Sampling and Consensus (RANSAC) is taken, in which the set of all motion vectors Υ is sampled, creating the new sub-set of four vectors $\bar{\Upsilon}$, which produces an inlying result during estimation. The remainder of this section will discuss this approach in terms of the selection of a sub-set of four vectors that offer an accurate estimation of plane parameters.

In the previous section, it was observed that the distribution of velocities in real-world scenes are approximately Gaussian, it stands to reason (by the Central Limit Theorem [100]) that the distribution of mean velocities for all subsets will also be roughly Gaussian. By taking a number of subsets, it is assumed that some of these will contain motion vectors of similar (or ideally identical) velocities. Those subsets with constant velocity are expected to provide a better estimation than those without and inliers are determined by the error function given in equation (3.9).

As the system contains four unknowns, clearly $||\bar{\Upsilon}|| \geq 4$ must hold. The number of potential subsets of four motion vectors from Υ makes it computationally intractable to calculate for them all (for example, after 3 minutes of the "students003" video, there were roughly 30,000 motion vectors – approximately 4×10^{16} possible combinations). As such it is important to choose sensibly from Υ.

If the motion vectors in $\bar{\Upsilon}$ are not well distributed across the scene, the problem becomes ill-conditioned. Therefore, those subsets of four highly-separated motion vectors are likely to offer greater stability during the estimation step than those containing badly separated motion vectors. To this end, potential subsets $\bar{\Upsilon}$ are chosen from a sample of motion vectors $\Upsilon' \subset \Upsilon$, taken from the outer edges of the image-space.

In order to build Υ', the convex hull K_1^{Υ} (hereafter referred to as the "primary convex hull") of the set of all motion vectors is calculated, as this will include those around the edges of the observed motion:

$$K_1^{\Upsilon} = \text{convhull}(\Upsilon) \tag{3.10}$$

This is likely to only lead to a very small set of motion vectors, in comparison with the size of Υ. In order to improve the likelihood of finding subsets of near-constant velocity, additional motion vectors are included by taking the "secondary convex hull" K_2^Υ. That is, the convex hull of the remaining motion vectors once those in K_1^Υ are removed:

$$K_2^\Upsilon = \text{convhull}(\Upsilon \setminus K_1^\Upsilon) \tag{3.11}$$

This leads us to the final set of motion vectors to use in the sampling process:

$$\Upsilon' = K_1^\Upsilon \cup K_2^\Upsilon \tag{3.12}$$

From Υ', sets of four motion vectors are sampled in a Monte-Carlo fashion, with the constraint that the area of the polygon formed by the four motion vectors must exceed a threshold of half the area of the image. Figure 3.4 shows the example output of this method on motion vectors taken from 30 seconds of the "students003" dataset. The set of all motion vectors is given in figure 3.4a, with b–g showing example subsets of motion vectors, denoted by the corners of each blue quadrilateral. Also highlighted are the primary and secondary convex hulls (coloured magenta and green respectively), from which Υ' was formed. It can be seen that the points are well separated, appearing towards the edge of the image.

The non-linear system shown in equation (3.9) must now be solved. The Levenberg-Marquadt algorithm is used to iteratively optimise for some initial set of parameters, chosen as it is particularly robust, combining the benefits of both the gradient descent and Gauss-Newton optimisation methods. This It should be noted that the problem-space is highly non-convex, particularly when far from the true values, therefore selection of the initial condition for the optimisation is important. Adopting a Monte-Carlo approach, initial points in the problem space are randomly chosen from within a feasible range of values for each parameter. To define this feasible range it is assumed that for most views in the real-world, θ will fall within the $0°$ to $90°$ range, with ψ being contained in the range $-45°$ to $45°$. Using equation (3.1), a feasible range of values for $(\hat{a}, \hat{b}, \hat{c})$ can be constructed. The remaining parameter, α is chosen according to the likely range of focal lengths, giving the range 10^{-3} to 10^0.

The algorithm, should it converge, will deliver a parameter set $(\alpha', \hat{a}^*, \hat{b}^*, \hat{c}^*)$. The feasibility of such a parameter set is assessed by back-projecting all imaged motion vectors onto the ground-plane, then checking the normality of the speed distribution using

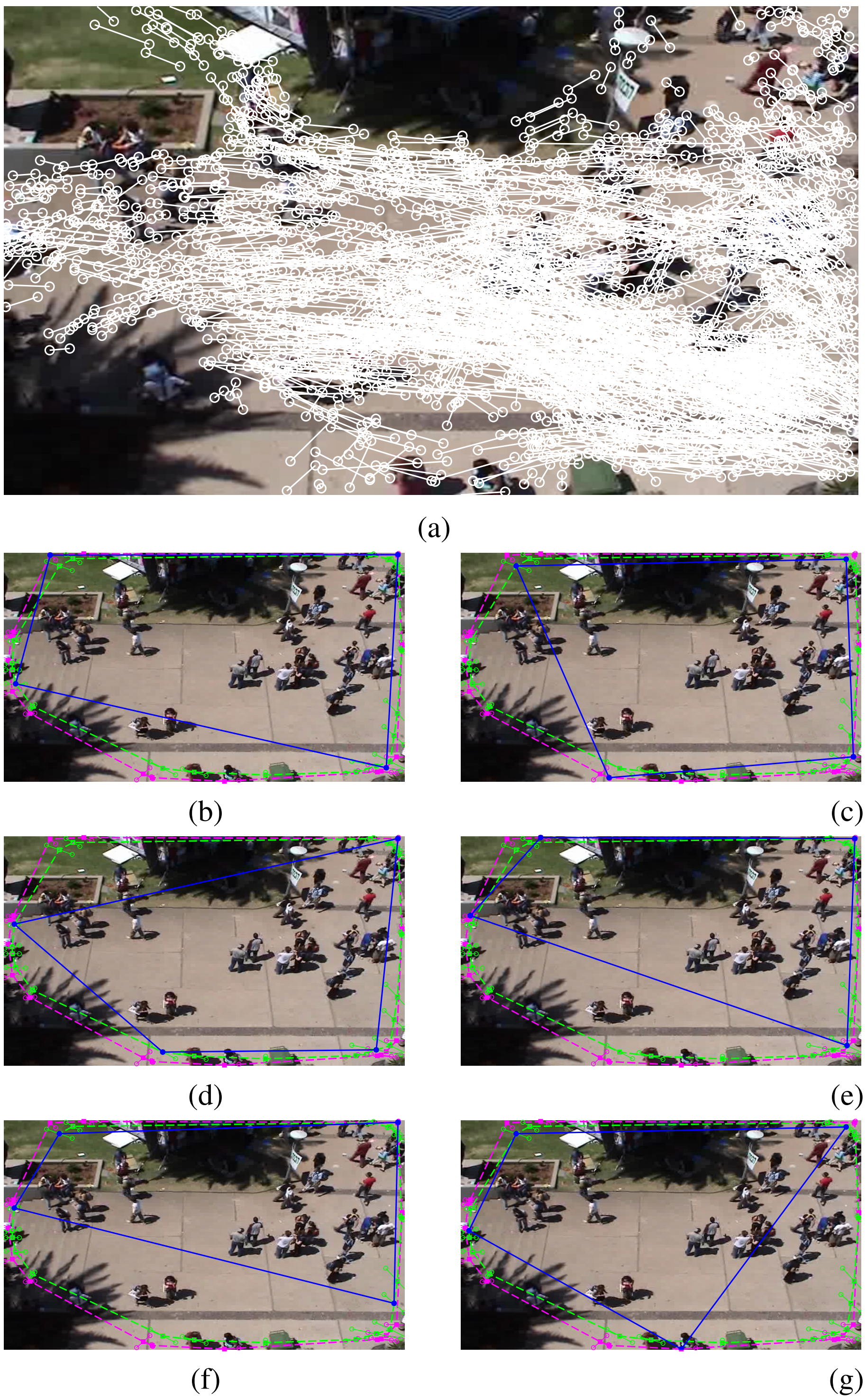


Figure 3.4: Subsets of four motion vectors are chosen from the outer edges of the motion so as to maximise the separation between them. This is performed by calculating the primary K_1^{Υ} (magenta) and secondary K_2^{Υ} (green) convex hulls of the motion vector set (the latter being the convex hull of $\Upsilon \setminus K_1^{\Upsilon}$). These subsets are then selected in a Monte-Carlo fashion provided the area of the polygon they form (blue) is greater than 50% of the area of the image. (a) Shows the initial input set of motion vectors, while (b)–(g) show examples of the output.

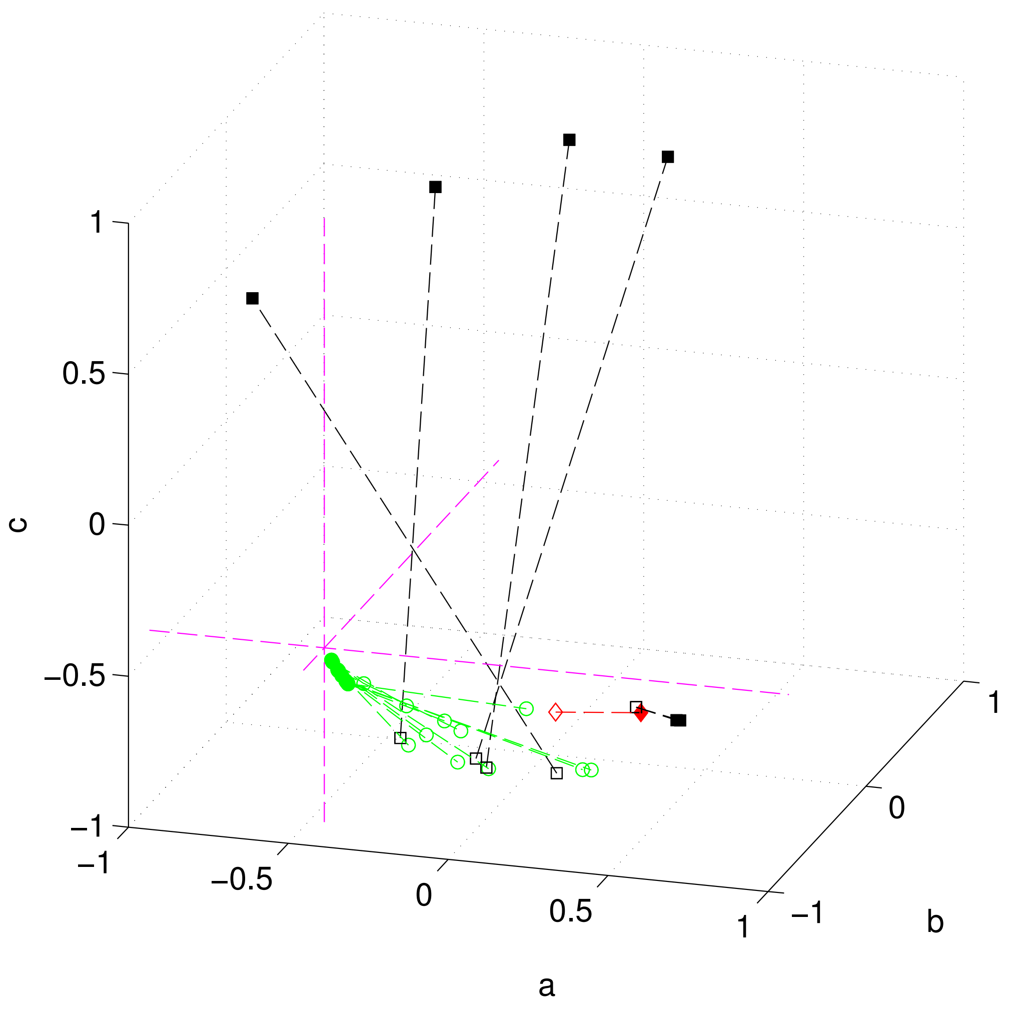


Figure 3.5: Initial values (hollow markers) and converged results (solid markers) using the Levenberg-Marquadt algorithm on noise-free input motion vectors in terms of unit $(a,b,c)^\top$ space. For clarity only a representative sub-set of values is shown. The ground-truth is represented by the intersection of the magenta axes. The majority of initial positions result in a value that is either reasonably close to it (green circles) or immediately discountable as incorrect (black, filled squares), either based on the distribution of velocities, in the case of the point with negative c, or by falling outside of the feasible range in the case of the remaining black points. Very few initial values result in an identifiably incorrect answer (red).

the Kolmogorov-Smirnov test. If the distribution is highly non-normal, we class this result as infeasible. Large numbers of results can thus be excluded, based on either non-convergence, non-normality of the speed distribution and if they do not fall within the ranges mentioned above. The final hypothesis for the ground-plane is found by taking the mean of the surviving results. Empirically, removing extreme outliers in computation of the mean offers little effect (positive or negative) as to the final estimation accuracy.

3.2.2 Viability Testing on Controlled Simulations

The assumption of uniform, constant speed throughout the scene implies that all motion vectors should have a speed of 1, given the correct parametrisation of a perfect world. To test the feasibility of this method, a "perfect world" is simulated by generating vectors of length 1 across a plane of known orientation. In each experiment 500 motion vectors were generated and placed randomly throughout the scene. Therefore, it is possible to measure the accuracy of the estimations by examining the squared-length error, summed across all motion vectors and the orientation error in the plane estimation, given by the angle between the true and estimated planes.

Figure 3.5 shows a representative set of initial conditions for the parameters of the unit normal $(a,b,c)^\top$ (hollow markers) and convergence values (solid markers) for noise-free simulated data, in which all motion vectors were of unit length in the world coordinate system. Note that the majority of solutions were either correct (defined as close to the true value and shown in green) or recognisably infeasible (shown in black). Very few erroneous points survived the filtering process.

Given that this method relies on assumptions that are unlikely to hold precisely true in the real-world, it was then necessary to assess the effect of adding realistic variation to the input motion vector data. There are two types of error here:

1. Motion vectors lying on parallel planes above the true ground-plane
2. Motion vectors being of unequal length.

The latter is expected to have a particularly pronounced effect as the assumption of constant speed is particularly strict in this formulation. Clearly, in the real-world other potential sources of noise exist. but in these experiments they are assumed to be negligible or non-existent – namely poor tracker quality, as is common in low resolution or badly lit scenes, and non-pedestrian features being tracked, such as swaying tree branches.

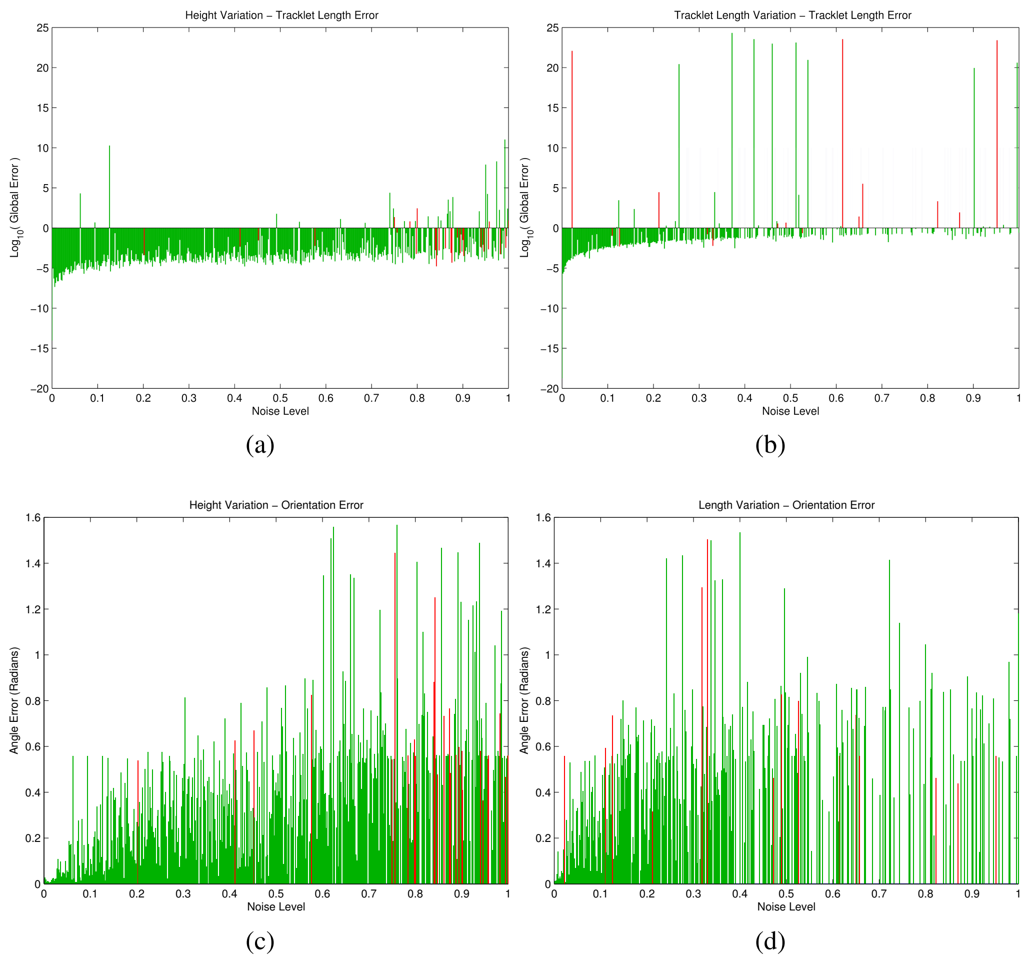


Figure 3.6: Two probable sources of error are examined using controlled, simulated data. The left-hand column shows the effect of motion vectors being displaced above or below the plane whilst the right-hand column demonstrates the effect of variations in vector length. Error is described in two terms. The top row is in terms of the global $\log_{10}$ sum-squared error in vector length (with a $\log_{10}(1) = 0$ as a point of reference), whilst the bottom row gives the angle error between the ground-truth and estimated normals. Red bars indicate results identified as infeasible, whilst green shows results the algorithm accepts as correct. Only those estimations that reached a convergence are shown.

Height variations are additive and are modelled using a Gaussian distribution with mean 0. The level of height variation is controlled by the standard deviation of the distribution in the range 0-1. As walking speed is assumed to be approximately 1m/s, represented by unit length in the world, this gives a feasible range of height deviation. At each noise level, motion vectors are generated with heights randomly selected from the distribution. The motion vector still lies parallel to the plane and is merely translated above it.

Variation in speed is also parametrised as a Gaussian distribution, but is multiplicative. The mean of the distribution is 1 and the level of variation is again controlled using the standard deviation between 0 and 1, giving a reasonable approximation of the range of speeds one is likely to observe in the real world. The length of generated motion vectors is extracted randomly from this distribution, and depending on the standard deviation could involve stationary points (although these would be filtered out by the estimation procedure) up to motion vectors of more than twice the average speed.

Figure 3.6 shows the effect of these types of error on estimation accuracy in terms of two error metrics. Firstly, the angle between the ground-truth unit normal, $\mathbf{n}$ its estimated counterpart $\hat{\mathbf{n}}$, shown in equation (3.13) is examined. Secondly, comparison is made between the length of motion vectors rectified with the ground-truth and estimated parameters as shown in equation (3.14). Here the true length of a motion vector is described as l_υ and the estimated length as $\hat{l_\upsilon}$, where υ is some vector in the set of all motion vectors Υ.

$$E_a = \cos^{-1}(\mathbf{n} \circ \hat{\mathbf{n}}) \tag{3.13}$$

$$E_l = \frac{1}{n} \sum_{\upsilon \in \Upsilon} (l_\upsilon - \hat{l_\upsilon})^2 \tag{3.14}$$

Figure 3.6 shows the results of experiments taken on 1000 different plane orientation configurations. It can be seen from figure 3.6a that global vector length error (displayed on a logarithmic scale for clarity) stays relatively low despite high noise level, but in the occasions that it does rise, the error is extreme. However, when considering the orientation error in figure 3.6c, a considerably more pronounced increase in error can be observed as noise increases. This is due to the enforcement of equal length for all motion vectors, but those closer to the camera (i.e. raised above the plane) will appear larger in the image.

A more pronounced effect can be observed when considering vector length variation. In terms of the length error shown in figure 3.6b, not only do more examples of high error get classified as correct, but towards higher levels of noise, a much lower density of completed experiments occurs (that is most ,if not all, plane configurations could not obtain a result due to the non-linear optimiser not converging). Examining the orientation error also shows that higher error occurs more quickly than for height, as well as the fact that considerably fewer experiments were solvable at all.

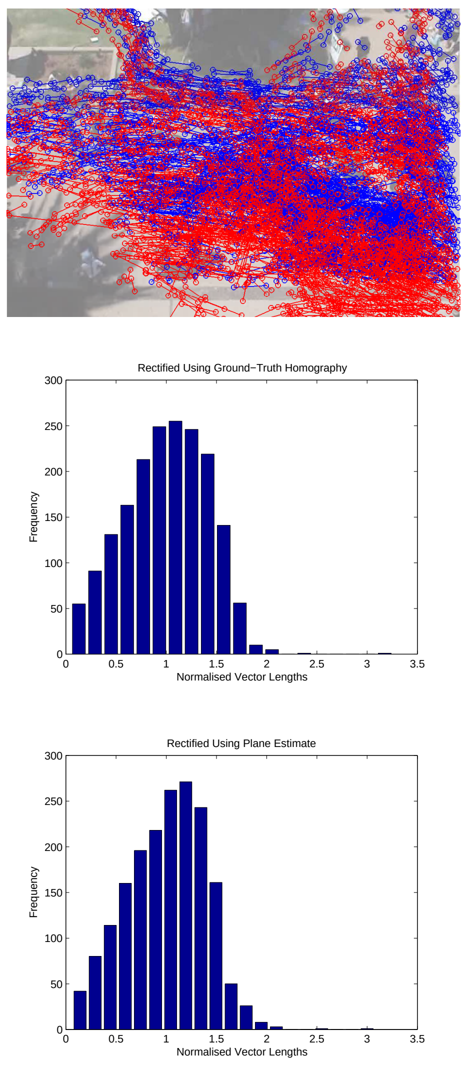
The error levels discussed above are unacceptable for many applications and are due, in large part, to the strictness of the constant speed assumption not allowing the system to function in anything but the most perfect of worlds. We see further evidence of this in the reconstructions of real-world scenes given in figure 3.7, where there is a noticeable difference in shape between the ground-truth reconstruction (blue) and the estimated reconstruction (red). In order to improve this situation the constant speed assumption is loosened still retaining the idea of using only pedestrian motion to solve for the ground-plane parameters.

3.3 Moving from Motion Vectors to Trajectories

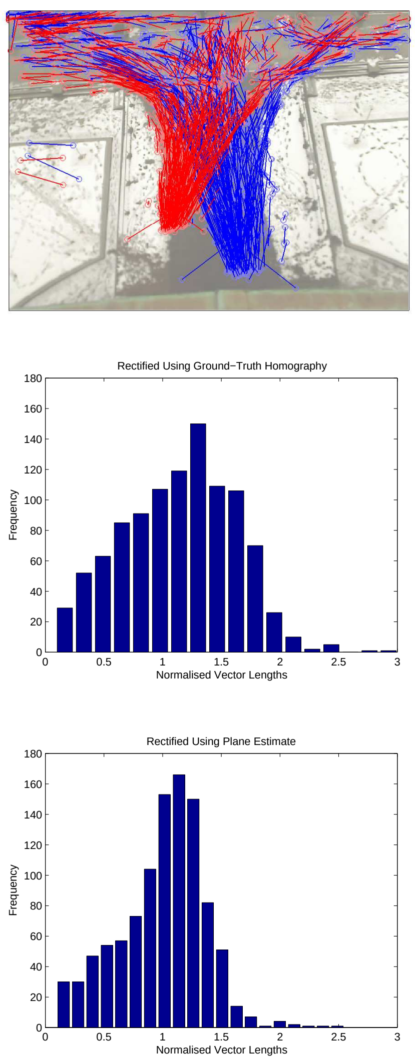
In the previous section it was assumed that all agents in the scene move at the same speed. Clearly this is not the case and violations of this assumption were a core source of error in the work presented in the previous section. Instead assume that different individuals will move at different speeds, but each individual moves at their own constant-speed. Since all agents are of the same class, pedestrians, sensible restrictions can be placed on the difference in speeds between people – observations in which one individual moves at 1m/s whilst another moves at 4m/s are unlikely, so these solutions can be disregarded – otherwise the system freely determines the ratio of speeds between each trajectory.

To this end, the system is enhanced to record not motion vectors but trajectories, discretised into linear segments over some time-interval. Features are tracked as long as is feasible using the KLT tracker (using the goodness measure associated with the tracker), giving a set of trajectories to use as input. Clearly the formulation can no longer rely on the agent's speed being 1; however by fixing the speed of one trajectory, the scale of the reconstruction is fixed as in the previous method. The speeds of the remaining trajectories are described as being some multiple of the speed of the first. That is, for a set of trajectories **T**, there exists a set of speeds:

$$\{l_\tau\}_{\tau \in \mathbf{T}} \tag{3.15}$$



(a)



(b)

Figure 3.7: Top row: Two reconstructions of motion vectors for real-world scenes using ground-truth parameters (blue) and their estimated counterparts (red). The reconstructions include considerable error, with (a) showing too little angle of elevation, whilst (b) shows considerable skew. The most likely cause of error is different agents moving at relatively different speeds (some run, some walk). The distributions of vector length across all motion vectors, back-projected with ground-truth are provided (middle row) as are those rectified using the estimated parameters (bottom row), further illustrating the evident skew.

Scale is set by fixing $l_1 = 1$, with all further lengths being denoted as $l_\tau = \omega_\tau l_1 = \omega_\tau$. Using the notation introduced previously, the set of parameters for the system is $(\alpha, \hat{a}, \hat{b}, \hat{c}, \omega_2, \omega_3, \ldots, \omega_{N_\mathbf{T}})$ – $\omega_1 = 1$ so is not included in estimation – giving the error measure:

$$E(\mathbf{T}) = \sum_{\tau \in \mathbf{T}} \sum_{i=2}^{N_\tau} \left(L_1(\mathbf{u}_{i-1}^\tau, \mathbf{u}_i^\tau) - \omega_\tau \right)^2 \tag{3.16}$$

Again the Levenberg-Marquadt (LM) non-linear solver is used to optimise a series of initial points; however given the system now requires a solution for a large number of parameters, choice of initial condition must be taken more carefully than the earlier Monte-Carlo algorithm. Therefore a multi-resolution global search approach is taken with a final optimisation step to find the correct solution. Sensible bounds must be placed upon the search-space and in particular feasible orientation values should be given. Whilst the parameter sets $(\hat{a}, \hat{b}, \hat{c})$ and (θ, ψ, d) encode the same information, the latter lends itself more easily to uniform sampling of the space of plane orientations. Therefore, the parameter set being searched is modified to $(\alpha, \theta, \psi, d, \omega_2, \omega_3, \ldots, \omega_{N_\mathbf{T}})$.

At the first level all combinations of α, θ and ψ are considered over a coarse mesh – increments of 15° in both the θ and ψ with feasible ranges 0° to 90° and -45° to 45° respectively. For α, an exponential search is used to find its initial scale, in the same range as for the motion-vector experiments, 10^{-3} to 10^0.

Taking the point with minimum error from this search, a finer grid is produced around it; now searching α linearly and reducing the step size for θ and ψ to 10% of their previous value, 1.5° at the second level. At each node of the mesh an optimisation is performed, with the intention of moving closer to the true answer quickly. This procedure is repeated until either the lowest error point is below a given tolerance (empirically 10^{-5} is sufficient) or the maximum level allowed for search is reached. Empirically, 3 levels is observed to be sufficient for an accurate estimation on simulated data with some speed variation.

As it is impossible to know the values for $(\omega_2, \ldots, \omega_{N_\tau})$ in advance, they are simply initialised unit speed. Since the speed of all agents' motion is expected to be within the same order of magnitude, this should be sufficient (an assumption which is explored in more detail in section 3.3.2).

3.3.1 Generating Trajectory Simulations

Again, the effect of controlled variations on input data must be assessed. The previous simulation method did not allow for the creation of continuous trajectories and more realistic output is desired than the randomly placed motion vectors of the previous method. This section will briefly introduce the simulation method, which is used for the remainder of this chapter.

First, a surface is defined, in terms of four points on the x-y plane in world coordinates. These can then be rotated by some given values in any of the three axes to produce a plane of any orientation with respect to the camera. Once the plane section has been generated, trajectories must be created upon it. Starting points are randomly generated along the scene boundaries. At each time-step the next position is generated by rotating the previous direction vector about the normal to the plane by a value taken at random from a zero-mean Gaussian distribution.

This allows the simulation of direction changes for agents moving across the plane. By adjusting the spread of this distribution the erraticism of the simulated motion can be increased. Similarly, the speed of the agent is chosen at random from a Gaussian distribution. Here the mean represents the mean speed of the agent, whilst varying the standard deviation allows intra-trajectory speed variations to be added (an agent can travel at different speeds during their time in the scene). Each agent is assigned a speed distribution with mean taken from a third Gaussian, allowing the introduction of inter-agent speed variation (agents can move at different speeds). The final parameter in these simulations is the height of the tracked point within the scene. Again this is parametrised as a Gaussian distribution with the spread representing the distance of the tracked feature from the ground-plane.

By manipulating the means and standard deviations of these parameters, the noise levels and violations of the constant-speed assumption can be assessed. Hereafter, the spread of these distributions is described as the “noise-level” for each parameter.

In addition to varying the above parameters, the camera position and orientation can be varied before transforming the scene into image coordinates, allowing observation of the system in both likely real-world situations and extreme cases (such as a camera close to the ground-plane with little rotation). A number of example scenes containing simulated trajectories are shown in figure 3.8.

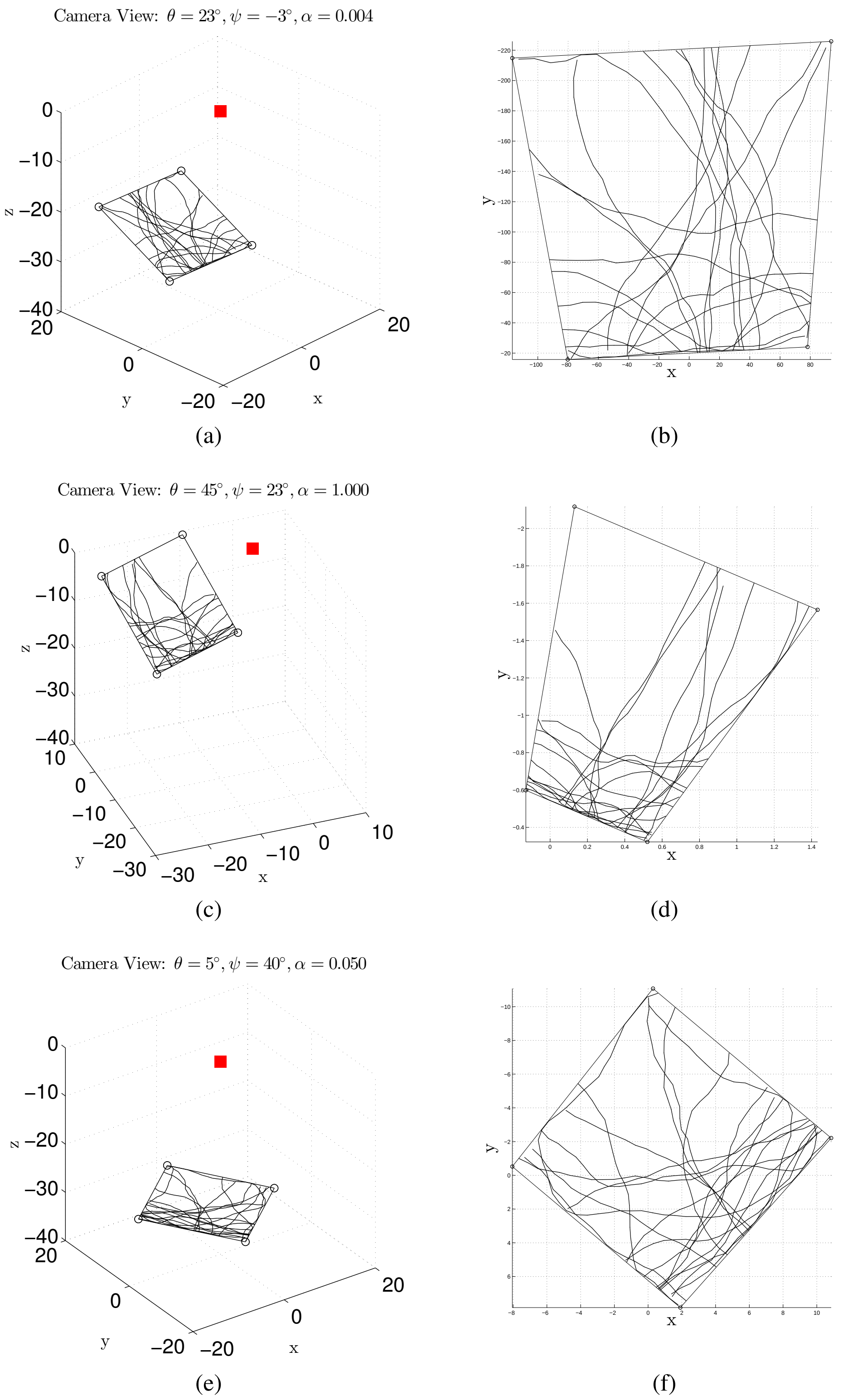


Figure 3.8: A series of simulated scenes at varying orientations and focal lengths using the method described in section 3.3.1. (a) to (e) show the scenes in camera coordinates with the camera shown as a red square, whilst (b) to (f) show each scene transformed into image coordinates.

Level (%)	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Height (°)	0.00	0.01	0.00	0.02	0.00	0.00	0.00	0.04	0.00	0.00	0.01
Intra (°)	0.00	13.1	31.9	41.1	34.7	39.8	32.6	34.6	31.8	33.1	28.8
Inter (°)	0.00	0.00	0.00	0.00	0.00	0.71	0.43	2.37	0.00	10.03	2.47

Table 3.1: Controlled, simulated data is used as input to the system outlined here to examine how the algorithm deals with noise or assumption violations. Three sources of error are examined: height variation, *intra-trajectory* speed variation and *inter-trajectory* speed variation. Of particular importance is the observation that height variation no longer has a pronounced effect given the new formulation; however intra-trajectory speed variation is still a strong factor in estimation accuracy.

3.3.2 Viability Testing on Controlled Simulations

Previously, the effect of two parameters on estimation accuracy was examined:

1. Height variation (assumed to vary between trajectories but remain constant throughout any particular trajectory) and
2. Speed variation within a trajectory.

The term "speed variation" must now be redefined as the system now has two potential sources of this. It is now assumed that individuals can move at different speeds. It would be useful to examine how those speed differences effect estimation of the plane parameters, particularly as $(\omega_2, \dots, \omega_{N_\mathrm{T}})$ is initialised at unit speed, which may be an over simplification. This is defined as "inter-trajectory speed variation". The violation of the constant-speed assumption is retained; however this is redefined to be "intra-trajectory speed variation", that is, the variation in speed in one particular trajectory.

In the previous section it was clear that measuring speeds between ground-truth and estimated reconstructions did not provide a reliable indication of reconstruction error. Therefore analysis will now focus on the orientation error between the estimated and ground-truth planes. Table 3.1 shows the resulting error profiles for varying these three types of noise:

Type 1 Height Variation: Tracking points above the ground-plane

Type 2 Intra-Trajectory Speed Variation: Variation in speed within a trajectory
e.g. person changes speed.

Type 3 Inter-Trajectory Speed Variation: Variation in speed between trajectories
e.g. different people walk at different speeds.

Let us first examine the effect of height variation on the algorithm. With the previous formulation, where unit speed was enforced for all motion vectors, this was observed to be a high source of error on the estimations. From table 3.1 it can be seen that it no longer has a pronounced effect on error. Instead the estimation accounts for the differences in observed speed by manipulating the speed ratio ω_t for that trajectory.

Type 2 variation is expected to have the most pronounced effect on the quality of the estimation as it violates the core assumption in equally as strong a fashion as in the previous formulation. Again, the orientation error rises steeply as the noise level increases. However, the levels of variation given here provide the most extreme of cases – it is unlikely that one person will change their speed to such a great extent in the real-world. With that in mind, the strictness of the constant-speed assumption will still limit the ability of the system to solve for these more difficult cases.

Finally, it can be seen that type 3 noise does have some effect on the quality of estimations, but only towards the extremes. At a standard-deviation of 0.8, it is possible to have one person moving many multiples faster than another, providing an extreme situation for the solver to surmount. Also note, that as the level of intra-trajectory noise increases, the algorithm sometimes uses $(\omega_2, \ldots, \omega_{N_\mathrm{T}})$ to compensate for the more difficult data. It would be prudent to limit these edge cases to feasible ranges.

Whilst changing from a somewhat naïve system using vectors to a more considered trajectory-based method has improved matters, the system is prone to error when trajectories of different speed are observed. With larger sets of input data, the number of degrees of freedom of the system in turn becomes very large, making finding the correct solution much more sensitive to initialisation. In the next section, an approach to overcome this difficulty is presented, along with quantitative analysis of its accuracy.

3.4 Reformulating the Constant Speed Assumption

It has become clear that the primary source of error within the system is the complexity of the problem space when a large number of trajectories is given as input. Additionally, the parameters for $\omega_2, \ldots, \omega_{N_\mathrm{T}}$ are initialised uniformly at one, which is unlikely to truly represent the variation in trajectory speeds. As such, this chapter introduces a new formulation of the constant speed assumption, such that variation in speeds within a trajectory can be analysed independently of the mean speed of the trajectory.

One consequence of this change is that pedestrian speed can no longer be used to set the scale of the scene. Instead, note that the height of the camera with respect to the ground-plane, d, acts primarily as a scaling parameter. As this work only aims to reconstruct to scale, this is fixed such that $d = 1$ in (3.5), thus simplifying further calculation.

In the majority of scenes, the camera height is observed to be substantial compared to the height variations of the tracked feature points. Section 3.4.1 offers results showing that tracked feature height variation does not significantly affect the estimate of ground-plane orientation.

As d is now fixed, $(\hat{a}, \hat{b}, \hat{c})$ can be exchanged for (a, b, c) in the back-projection function given in equation (3.7), giving rise to the variation below:

$$L_2(\mathbf{u}, \mathbf{u}')^2 = \alpha^2 \left(\frac{u}{\gamma} - \frac{u'}{\gamma'} \right) + \alpha^2 \left(\frac{v}{\gamma} - \frac{v'}{\gamma'} \right) + \alpha^2 \left(\frac{1}{\gamma} - \frac{1}{\gamma'} \right) \tag{3.17}$$

where,

$$\gamma = \alpha u a + \alpha v b + c \quad \text{and} \quad \gamma' = \alpha u' a + \alpha v' b + c$$

For a single trajectory τ, the set of distances at all time intervals is denoted as:

$$\{L_t^\tau\}_{t=2}^{t=N_\tau} \tag{3.18}$$

The mean and standard deviation of the above set are denoted as $\mu(L^\tau)$ and $\sigma(L^\tau)$ respectively. Examining $\sigma(L^\tau)$ gives a measure for how well a given set of parameters, α, θ and ψ fit the observed data. If a feature point is moving at constant speed and a good set of parameters has been found, $\sigma(L^\tau)$ should be close to zero. Conversely, a poor parametrisation is expected to give a high spread in L^τ.

Since it is undesirable to impose the constraint that all objects must move at constant speed, $\sigma(L^\tau)$ is normalised by $\mu(L^\tau)$ giving a speed-invariant measure of the spread, which is posed in terms of a minimisation over the sum of squared errors for each trajectory as shown in equation (3.19).

$$E_1^{\mathbf{T}} = \sum_{\tau \in \mathbf{T}} \left(\frac{\sigma(L^\tau)}{\mu(L^\tau)} \right)^2 \tag{3.19}$$

As tracked feature points are expected to have originated from agents of the same class (e.g. all pedestrians, no vehicles), it can reasonably be assumed *a priori* that they should all move with similar (although not identical) speed. A set is created consisting of the mean speeds of all trajectories in **T**:

$$\{\mu(L^{\tau})\}_{\tau\in\mathbf{T}} \tag{3.20}$$

If agents move at similar speeds, the spread of the above set should be low, whereas if different agents move at very different speeds the spread will be large. This concept forms the basis of a prior:

$$E_2^{\mathbf{T}} = \sigma(\{\mu(L^{\tau})\}_{\tau\in\mathbf{T}}) \tag{3.21}$$

where $\sigma(\{\dots\})$ is used to denote the standard deviation of the set. Applying the above prior to the error measure in (3.19) penalises a high spread of speeds, thereby preventing some of the least plausible configurations. A weighted sum of these two terms gives rise to the error measure $E^{\mathbf{T}}$ as given in equation (3.22). Minimising $E^{\mathbf{T}}$ over all input trajectories offers an estimation of the plane parameters (α, a, b, c).

$$E^{\mathbf{T}} = E_1^{\mathbf{T}} + \lambda E_2^{\mathbf{T}} \tag{3.22}$$

3.4.1 Experiments on Simulated Data

In this section, the robustness of the improved error measure given in equation (3.22) when faced with deviations from the perfect world is examined. As in section 3.3.2, three sets of experiments are performed on a number of different plane configurations across the feasible range of orientations and focal lengths. In each case, the potential source of error is varied in terms of the standard deviation of its Gaussian distribution. Examining the issues in order, let us first consider the effect of the different agents in the scene moving at different speeds. Agents' initial speeds are chosen randomly from the distribution and remain constant throughout the simulation.

Table 3.2 shows that the variation of the height of tracked features again exhibits negligible effect of the accuracy of the final estimation. Data was generated using a camera height of 10m, a feasible figure in the real-world; meaning the variation in the height of tracked features that is likely to be observed is small relative to their distance from the

Level (%)	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Height (°)	0.01	0.02	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.03	0.03
Intra (°)	0.01	2.57	3.91	5.53	7.23	8.09	7.00	5.41	6.66	6.81	5.98
Inter (°)	0.01	2.02	3.36	4.34	4.61	4.50	4.36	4.56	5.18	4.44	4.55

Table 3.2: The results of input data variation experiments for inter-trajectory speed variation, intra-trajectory speed variation and height variation, in terms of orientation error from the true ground-plane (in degrees). In all cases, error is low enough so as to produce a usable reconstruction of the 3D data.

camera.

In consideration of variation in speed between trajectories, it can be observed that even at high levels of variation, the average error stays low – below 10% of the mean speed. The effect of intra-trajectory speed variation is expected to be more pronounced as it is the defining metric used to recover the parameters. The speed of a feature at each frame is taken from the distribution. We observe that although more error is seen than with inter-trajectory variation, it is not so substantially pronounced as to seriously damage the quality of the result.

Whilst it is true that points tracked at different heights (figure 3.2) with a low-positioned camera will be affected more strongly, in most real-world scenes the distances between the lowest and highest tracked points are negligible with respect to the camera height. These experiments on simulated data show that over realistic ranges, the three potential sources of error identified above have negligible detrimental effect on the accuracy of estimation. Of particular importance is the intra-trajectory speed variation, which is a violation of assumptions and yet still does not have an especially pronounced effect on the quality of the estimation.

3.5 Obtaining Real-World Motion Data

Having confirmed the viability of the method presented in section 3.4 on simulated data, its accuracy will now be assessed when applied to real-world video footage. In order to observe imaged-object speed, identifiable features on objects must first be tracked for some period across a scene. To this end the OpenCV 2.3 implementation of the *KLT* tracker [81,121,112] was used, which tracks “corner” features – that is, points which offer identifiable motion information in two dimensions allowing non-ambiguous measurement of motion in both directions. Each feature is tracked frame–by–frame until such a time as tracking fails.

Clearly, one object may have multiple features assigned to it, just as some objects may be completely neglected by the feature point detector; however this is not a major concern provided that information is gathered from various positions in the scene. Also, since this work is based in the domain of high-density pedestrian crowds, one cannot guarantee that long-term tracking will be possible. High levels of inter-occlusion between pedestrians mean that retaining a tracking ID across the entire scene is likely to be infeasible. Due to this, only relatively short term trajectory segments may be recorded for each tracked object. Realistically, an object need only be tracked over 20-30 frames to gain useful information from the trajectory.

As discussed previously, it is assumed that each agent moves at constant speed. Whilst the formulation discussed thus far is somewhat robust to smaller speed variations that occur in the real-world, it would be prudent, where possible, to split trajectories where strong variations in speed are evident within a trajectory. In the original 3D scene, this would be simple; however due to the nature of the problem discussed in this work, it is difficult to tell whether a change in observed speed is actually related to a change in the plane direction. Nonetheless, if given the assumption of a single plane, as is the case in the work presented in this chapter, it is possible to account for severe changes of speed by recursively splitting trajectories at positions where the difference in speed in consecutive intervals is more than some threshold from the mean for that trajectory. A median filter is used to smooth the difference data prior to performing the splitting since in some cases there are spikes due to image noise.

Figure 3.9 shows one such trajectory with figure 3.9c indicating definite spikes in observed speed, particularly as the agent slows to turn around. As a result of this procedure three sub-trajectories are obtained, each with smoother speed profiles in the real-world.

3.5.1 Trajectory Grouping

Over the course of a video, many trajectories are likely to be obtained – far more than is necessary or in fact, feasible to solve for. In fact, all that is needed is a representative sub-sample of the set of trajectories to solve the non-linear system described above. A Monte-Carlo approach is unlikely to yield good results, as only short trajectories or trajectories representing merely a small part of the scene may be chosen. In this case the resulting non-linear system is likely to be ill-conditioned. Many trajectories may be ob-

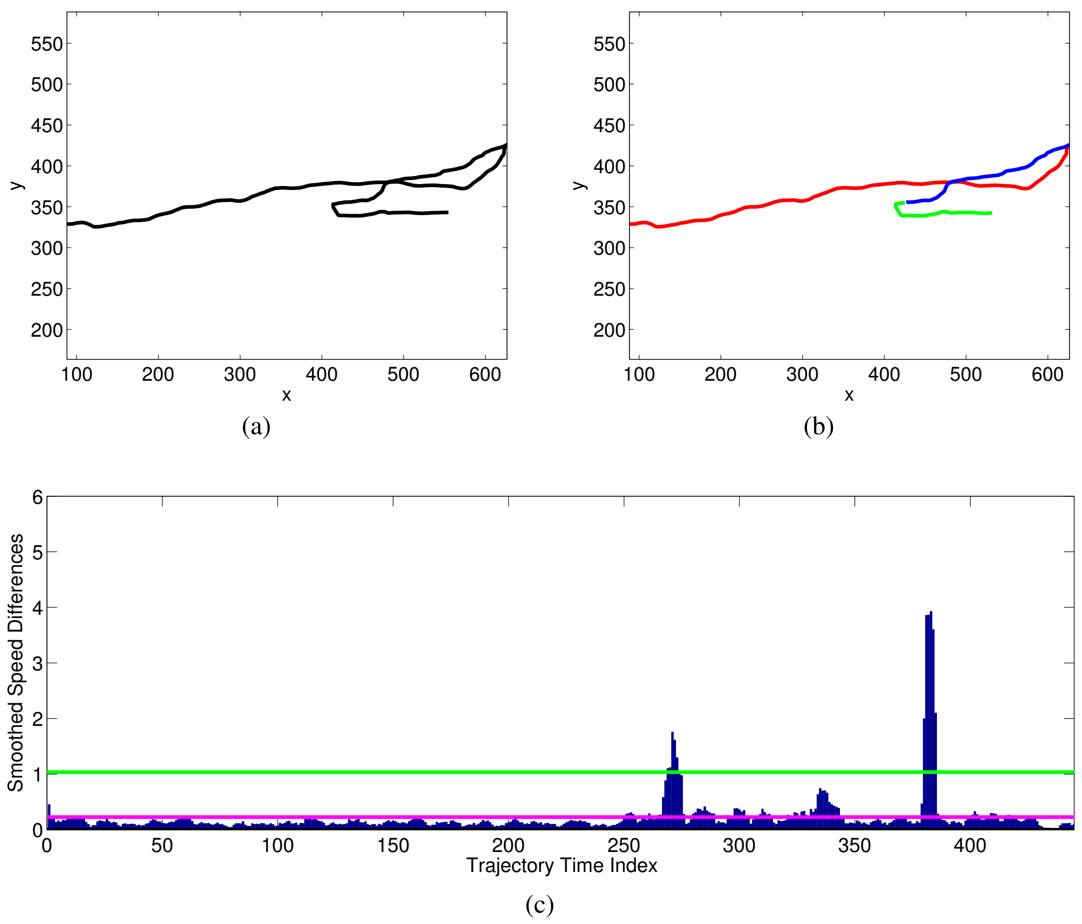


Figure 3.9: Trajectories are split at rapid changes in acceleration as these are likely to represent the agent changing speed in the real-world scene. In this example, (a) shows the original trajectory, (c) shows the smoothed speed-difference plot for the trajectory (mean value shown by a magenta line and the threshold in green) and (b) shows the result of the split, where the three coloured sections show the resulting sub-trajectories.

tained from a single person, or indeed a group of people moving together, giving rise to a number of extremely similar trajectories. Rather than using the entire set of trajectories, the trajectories are clustered relative to their position and shape in the image, in order to minimise the number of trajectories required to represent motion across the scene.

Since no prior knowledge is available as to the number of clusters needed to suitably represent the set, a simple K-Means approach does not apply. Instead Affinity Propagation [41] is implemented to group similar trajectories.

The first step is to find the optimal alignment for every pair of image trajectories in $\mathbf{T}$. This is based on two metrics - a measure of distance within the image and the similarity of their shape. As trajectories of different lengths will be compared, the Hungarian algorithm [69] is first used to obtain the optimal alignment between each pair of trajectories, taking the cost matrix generated using the function (3.25) as input. The constituent elements, distance and shape, will now be introduced in the following sections.

3.5.1.1 Trajectory Distance

For each trajectory pair, $\{\tau, \tau'\}$ the cost of matching each point in τ to each point in τ' is calculated. Given a pair of points $\{\mathbf{u}_i^\tau, \mathbf{u}_j^{\tau'}\}$ for time indices i and j in trajectories τ and τ' respectively, the normalised distance between the two points in the image is found as follows:

$$\frac{||\mathbf{u}_i^\tau - \mathbf{u}_j^{\tau'}||}{I_{diag}} \tag{3.23}$$

where I_{diag} is the diagonal size of the image frame.

3.5.1.2 Trajectory Shape Difference

The shape ϑ_i^τ at the point $\mathbf{u}_i^\tau$ is measured in radians as follows (illustrated in figure 3.10):

1. Find the line vector between homogenous points $\mathbf{u}_{i-1}^\tau$ and $\mathbf{u}_i^\tau$ using the vector cross product [53]:
$$\mathbf{l}_{\{i-1,i\}} = \mathbf{u}_{i-1}^\tau \times \mathbf{u}_i^\tau$$
2. Repeat for $\mathbf{u}_i^\tau$ and $\mathbf{u}_{i+1}^\tau$:
$$\mathbf{l}_{\{i,i+1\}} = \mathbf{u}_i^\tau \times \mathbf{u}_{i+1}^\tau$$

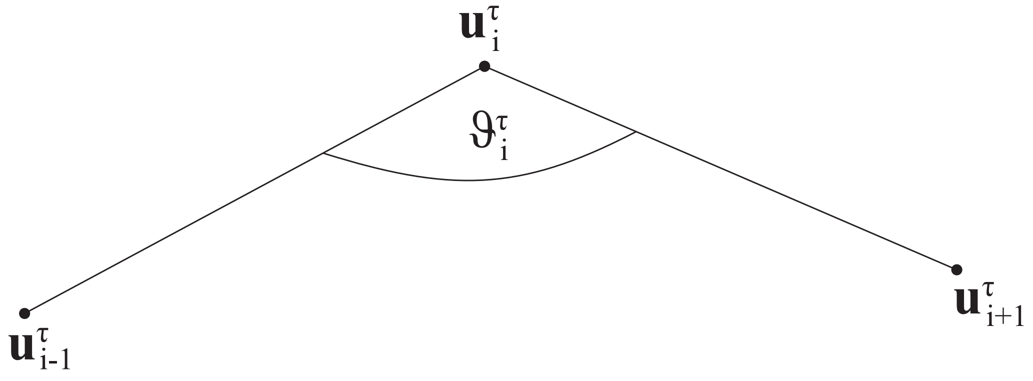


Figure 3.10: The shape of a trajectory τ is defined by the changes in direction ϑ_i^τ measured at each consecutive point $\mathbf{u}_i^\tau, i \in \{2, \ldots, N_\tau - 1\}$.

3. Calculate the angle ϑ_i^τ between the two lines using the dot-product:

$$\vartheta_i^\tau = \arccos\left(\frac{\mathbf{l}_{\{i-1,i\}} \circ \mathbf{l}_{\{i,i+1\}}}{||\mathbf{l}_{\{i-1,i\}}||\,||\mathbf{l}_{\{i,i+1\}}||}\right)$$

Having calculated the shape for both $\mathbf{u}_i^\tau$ and $\mathbf{u}_{i'}^{\tau'}$ in trajectories τ and τ' respectively, finding the difference, normalised over π is trivial:

$$\frac{|\vartheta_i^\tau - \vartheta_{i'}^{\tau'}|}{\pi} \tag{3.24}$$

3.5.1.3 Trajectory Alignment Cost

These two cost metrics are combined as a weighted sum to define the cost of matching a point in τ to one in τ':

$$C(\mathbf{u}_i^\tau, \mathbf{u}_j^{\tau'}) = \frac{||\mathbf{u}_i^\tau - \mathbf{u}_{i'}^{\tau'}||}{I_{diag}} + \lambda \frac{|\vartheta_i^\tau - \vartheta_{i'}^{\tau'}|}{\pi} \tag{3.25}$$

The function above is used to generate a cost matrix of assigning every point in τ to every point in τ'. This is then passed as input to the Hungarian algorithm, the result of which is an optimal alignment between τ and τ' and its associated minimal cost. This minimal cost is denoted $C^+(\tau, \tau')$:

$$C^{+}(\tau,\tau') = \underset{f}{\operatorname{argmin}} \sum C(\mathbf{u}_i^{\tau}, \mathbf{u}_{f(i)}^{\tau'}) \tag{3.26}$$

where $f(i)$ represents some mapping of index i onto another index.

3.5.1.4 Affinity-Based Trajectory Clustering

Having found the optimal alignment of each trajectory onto all others, a $N_\tau \times N_\tau$ matrix of affinities between trajectories can be built. From the previous section, the cost of matching two trajectories to each other is known as a result of the Hungarian algorithm's alignment. Given a set of trajectories $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_{N_\mathbf{T}}\}$, a matrix of trajectory matching costs can be constructed as follows:

$$\mathbf{C}_{match} = \begin{pmatrix} C^{+}(\tau_1,\tau_1) & \ldots & C^{+}(\tau_1,\tau_{N_\mathbf{T}}) \\ \vdots & \ddots & \vdots \\ C^{+}(\tau_{N_\mathbf{T}},\tau_1) & \ldots & C^{+}(\tau_{N_\mathbf{T}},\tau_{N_\mathbf{T}}) \end{pmatrix} \tag{3.27}$$

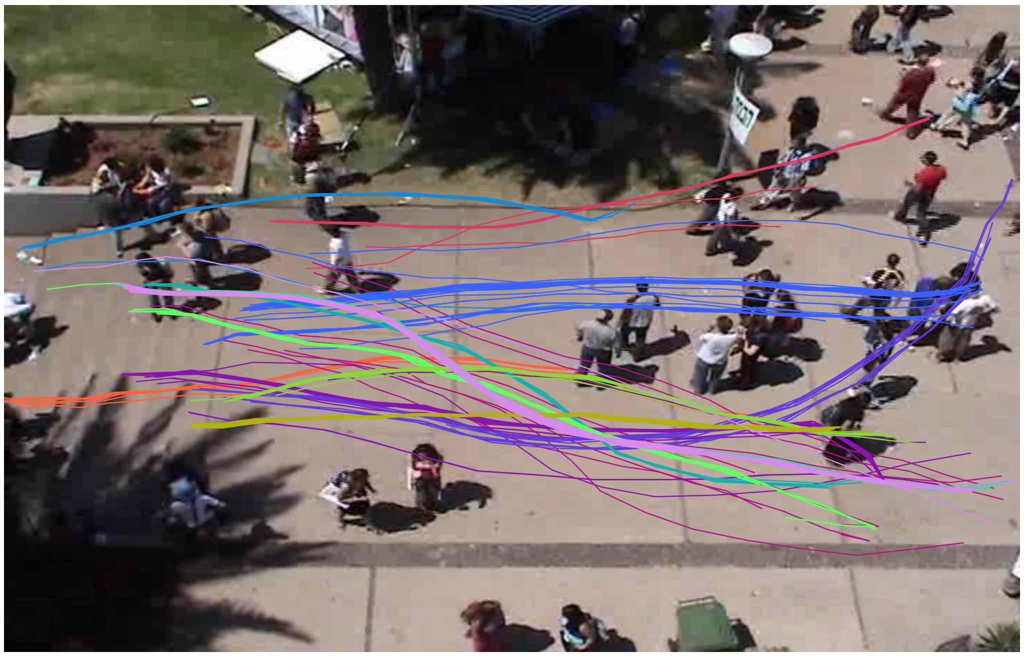
This is then normalised over the maximum value of the matrix to give values in the range $0 \ldots 1$, denoted $\mathbf{C}'_{match}$. To use this data as input to the affinity propagation algorithm, it must first be converted into an affinity, $\mathbf{A}_{match}$. This involves first subtracting the values of $\mathbf{C}_{match}$ from 1, and then preventing a trajectory from matching itself by setting the diagonal of $\mathbf{A}_{match}$ to zero:

$$\mathbf{A}_{match} = \mathbf{1} - \mathbf{C}'_{match} \tag{3.28}$$

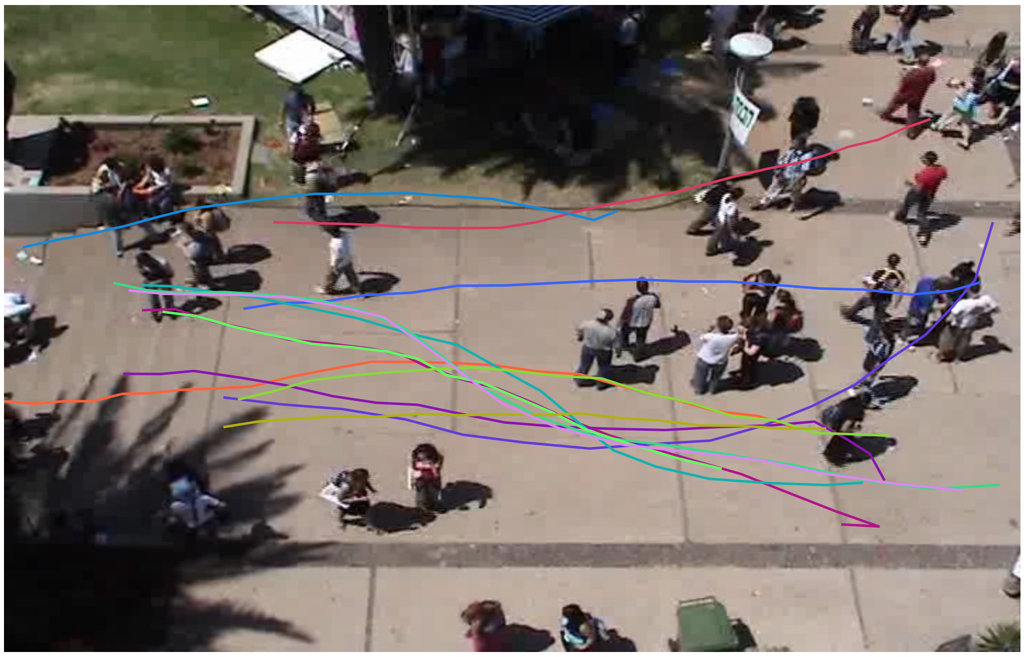
$$diag(\mathbf{A}_{match}) = \mathbf{0} \tag{3.29}$$

The result of running Affinity propagation on the input data is sets of trajectories, clustered according to their shape similarity and proximity. From each cluster, the most representative trajectory must be selected. This is a simple case of selecting the trajectory which maximises the sum affinity across all trajectories in the cluster. Figure 3.11 shows the result of one such clustering on the "students003" footage from the "Crowds by Example" dataset [74]. Only relatively long trajectories are shown for clarity.

Whilst some trajectories within certain clusters appear to be outliers in terms of image distance, it can be noted the shapes are often similar, suggesting that these trajectories



(a)



(b)

Figure 3.11: Trajectories are optimally aligned with one-another using the Hungarian algorithm [69], before being clustered using Affinity Propagation [41]. Finally, the most representative trajectory from each set is chosen such that the affinity with all others within the set is maximised. Here we see one such clustering on the “students003” dataset, with (a) showing the assignment of trajectories to clusters and (b) showing the most representative trajectories for each cluster.

came from different parts of the agent's body. Others are simply outliers for which there were no sensible clusters to be formed. These are automatically discarded when choosing the most representative trajectory for each cluster.

3.6 Results in the Real World

The majority of evaluation in this chapter is given against the PETS2009 dataset [37], specifically videos taken from View001 shown in 3.12a - 3.12g. Since these come with full intrinsic and extrinsic calibration, direct comparison of rotation angles and focal length is possible. This consists of several scenarios, named "S0" to "S3" and within those several sub-datasets "RF" (regular flow) in the case of S0 and "L1" to "L3" for the others.

All scenes contain pedestrian motion containing medium to dense crowds. Only those sequences within the dataset that are applicable to this work are chosen, including some challenging sequences containing sudden direction and speed changes and those containing difficult tracking conditions. Table 3.3 shows results on the selected datasets. The system converges on the same focal length for all sequences, within 0.05mm of the true solution of 5mm.

Dataset S0 is described as regular flow, with agents walking along a curved road with consistent motion from the left to right-hand sides of the scene. The exception is time-index 14-06, where agents start in the middle of the scene and remain stationary for a period of time, before walking together to the right. Higher error is observed on this particular sequence as the motion is somewhat uni-directional, with the lack of directional data in one direction meaning that one of the rotation axes is ill-conditioned when solving.

S1 is again regular crowd motion, with L1 containing medium crowds and L2 containing more densely packed individuals. L3 is a particularly challenging sequence for this system as it contains a change from walking to running at a point within the scene; however, a good result is still obtained since the rapid speed change is detected by the aforementioned trajectory splitting algorithm.

In S2, we consider only L2 and L3 as L1 contains a very sparse crowd (a maximum of 8 people at any one time, distributed across the scene). These subsets contain medium and dense crowds respectively, with unstructured motion throughout the scene. Accuracy drops in the former sub-set, due to people changing speed and direction rapidly to avoid

(a) PETS 2009 S0 Regular Flow

(b) PETS 2009 S1 L1

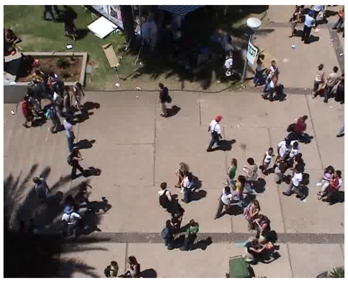
(c) PETS 2009 S1 L2

(d) PETS 2009 S1 L3

(e) PETS 2009 S2 L2

(f) PETS 2009 S2 L3

(g) PETS 2009 S3 L1


(h) students003

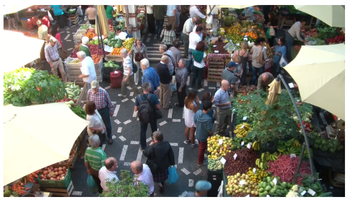

(i) Madeira Marketplace

Figure 3.12: Example stills from PETS2009 (a)-(g), students003 (h) and Madeira Marketplace (i) video datasets.

Table 3.3: Results for plane estimation on videos from the PETS2009 dataset, using View 001. “Speed error” is the mean back-projection error over all trajectories, calculated as a percentage relative to the reconstruction scale for which the mean speed is 1.

Dataset	**Time Index**	**Traj. Clusters**	**θ Error (deg.)**	**ψ Error (deg.)**	**Focal Length Error (mm)**	**Speed Error (%)**
S0 RF	13-57	5	-2.1	+3.8	+0.0481	3.87%
S0 RF	13-59	5	-5.2	-1.8	+0.0522	3.17%
S0 RF	14-03	8	-6.7	-0.3	+0.0467	4.68%
S0 RF	14-06	9	-5.3	+1.1	+0.0511	3.26%
S0 RF	14-29	7	-6.1	-19.7	+0.0546	9.65%
S1 L1	13-57	4	-8.4	+4.9	+0.0524	8.38%
S1 L1	13-59	4	+1.1	+11.7	+0.0452	3.55%
S1 L2	14-06	7	+7.5	-0.5	+0.0490	3.64%
S1 L3	14-17	72	-4.8	-7.2	+0.0532	3.34%
S2 L2	14-55	19	-8.7	+13.8	+0.0467	2.36%
S2 L3	14-55	8	-4.8	-7.0	+0.0498	4.85%
S3 L1	14-13	29	-4.8	+3.1	+0.0522	3.89%
S3 L1	14-37	7	-4.8	+0.1	+0.0513	5.28%

others, meaning that the constant speed assumption is violated on a regular basis.

S3 contains some particularly challenging subsets and those not applicable to this work are disregarded, such as the entire L2 sub-set, designed for event detection, in which motion is extremely irregular with rapid speed changes. Of those studied in L1, good results are achieved – within 5° for θ and approximately 3° for ψ.

One other video dataset, “students003” from the University of Cyprus is examined, a frame of which is given in figure 3.12h. This is provided with an image to ground-plane homography; however without knowledge of the intrinsic calibration of the camera it is not possible to uniquely decompose the homography to obtain the planar orientation [6]. Therefore the homography is used to determine the 2D feature coordinates on the ground-plane and compare the speed ratios for each trajectory. In section 3.6.1 rectified trajectories produced using the method in [12] (discussed in detail in chapter 2) are compared with those generated using this method.

Table 3.3 shows the orientation error for several scenes in terms of direct difference for the two rotation components, θ and ψ. Also given in the table are the mean reconstruction errors (“Speed Error”). These are calculated by first normalising both the ground-truth and estimated reconstructions such that the mean speed is 1, then calculating the mean of the differences in speeds at each time interval, summed over all trajectories. Our recon-

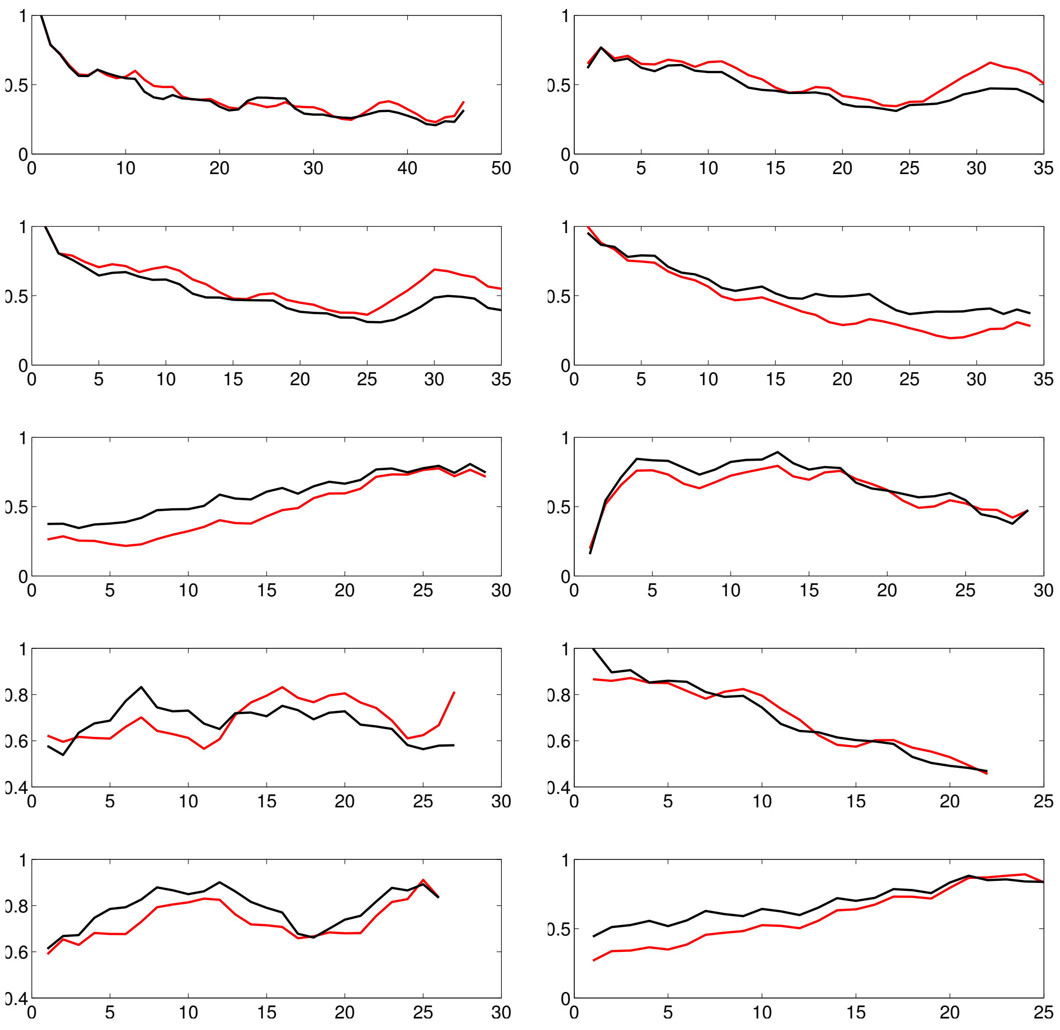


Figure 3.13: Comparison of trajectory speeds rectified using the ground-truth (*black, dashed*) and estimated parameters (*red, solid*). Examples are the 10 longest trajectories from the Madeira Marketplace dataset.

structions are very close to the true solutions in all cases.

A further dataset for comparison was captured above a busy marketplace in Funchal, Madeira, a frame of which can be seen in figure 3.12i. This footage contained extremely dense crowds with limited movement both due to the number of pedestrians in view and the layout of the market itself. The system performed well, obtaining a good estimate of the plane parameters, as evidenced by a close reconstruction of trajectory speeds, as shown in figure 3.13. No comparison could be drawn against the Bose and Grimson algorithm demonstrated in the following section as the system was unable to produce an estimate of the plane parameters due to the high levels of variation within the input data.

3.6.1 Comparison with Bose and Grimson [12]

In this section, a comparison with the method of Bose and Grimson [12], discussed in detail in chapter 2, is offered. As no public implementation of the method was available, this was developed as part of this work, based on the details within the paper. The use of a blob-tracker in the original work was substituted for the aforementioned KLT approach as the former provided tracks of insufficient quality (compared to ground-truth trajectories) for reliable estimation when applied to our data.

Figures 3.14 and 3.15 show some example comparisons of normalised trajectory speeds from the PETS 2009 S1L1 and “students003” datasets respectively. For each result the speeds are presented as rectified using the provided calibration data or ground-truth (black), the estimation from the system presented here (red) and the method of [12] (blue). The reconstruction based on the system in this work is generally very close, even on trajectories with some tracking error, whereas the method of Bose and Grimson performs poorly. This is likely due to the flexibility of this approach in minimising spread rather than the strict constant speed assumption of their method.

3.7 Conclusions

The work described in this chapter has considered the problem of reconstructing 3D geometry from 2D observations taken from videos of pedestrian data taken using a single uncalibrated camera. This method differs from previous techniques as it requires no knowledge of scene geometry or a fixed size object; needing only motion of individuals. Evidence was provided on simulations for the validity of the method and the assumptions held within. Results were then given on the PETS2009 dataset which illustrate the success of the method in a number of cases and have given a qualitative comparison for another. A limitation of this work is that it assumes the scene to contain only one ground-plane, an assumption that may not be valid in real-world scenes as ramps and similar deviations from a single plane can often be observed. Therefore, the following chapter will consider the problem of a scene containing two planes, before generalising this to an unknown number of planes in chapter 5.

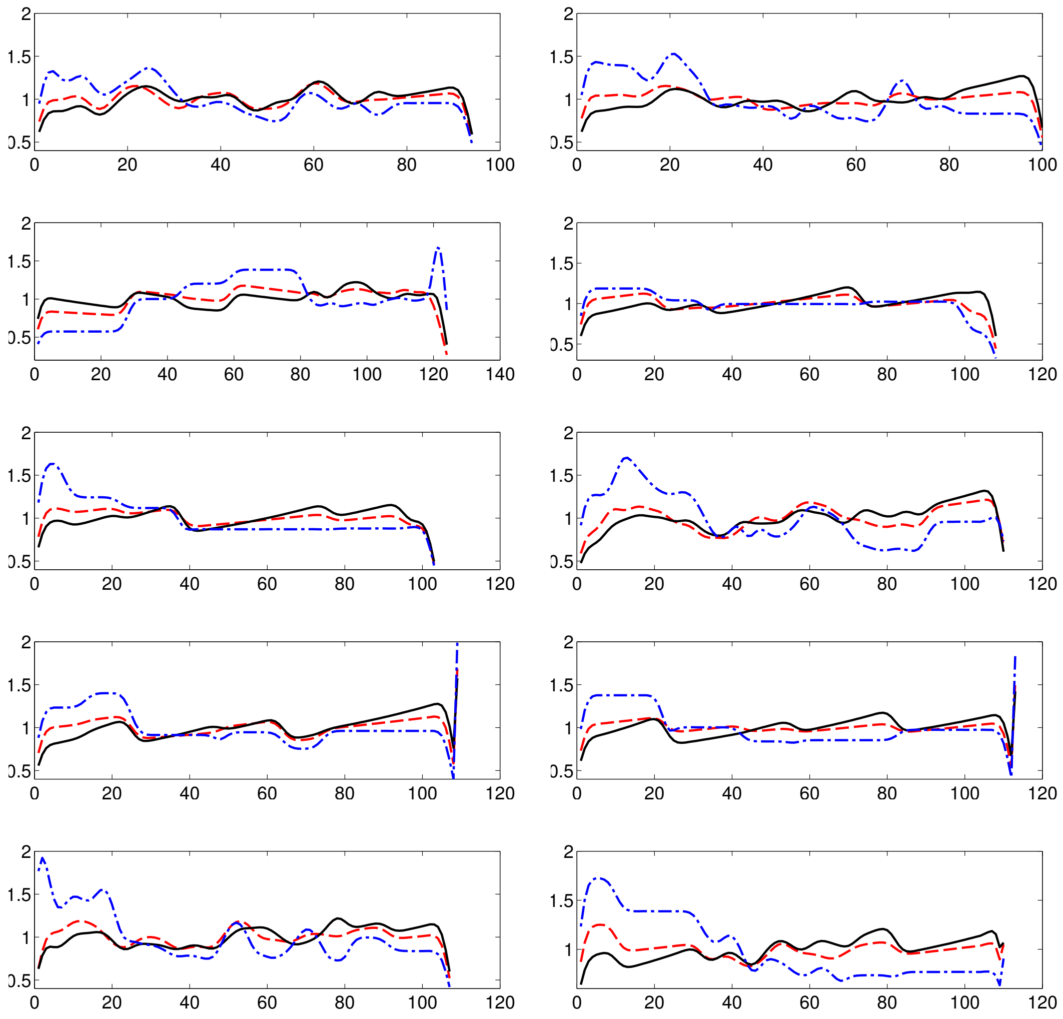


Figure 3.14: Comparison of trajectory speeds rectified using the ground-truth (*black, dashed*) and estimated parameters (*red, solid*) and Bose (*blue, dot-dashed*). Examples are the 10 longest trajectories from PETS 2009 S1 L1, 13-59. In general, the system presented here outperforms Bose in terms of reconstruction quality.

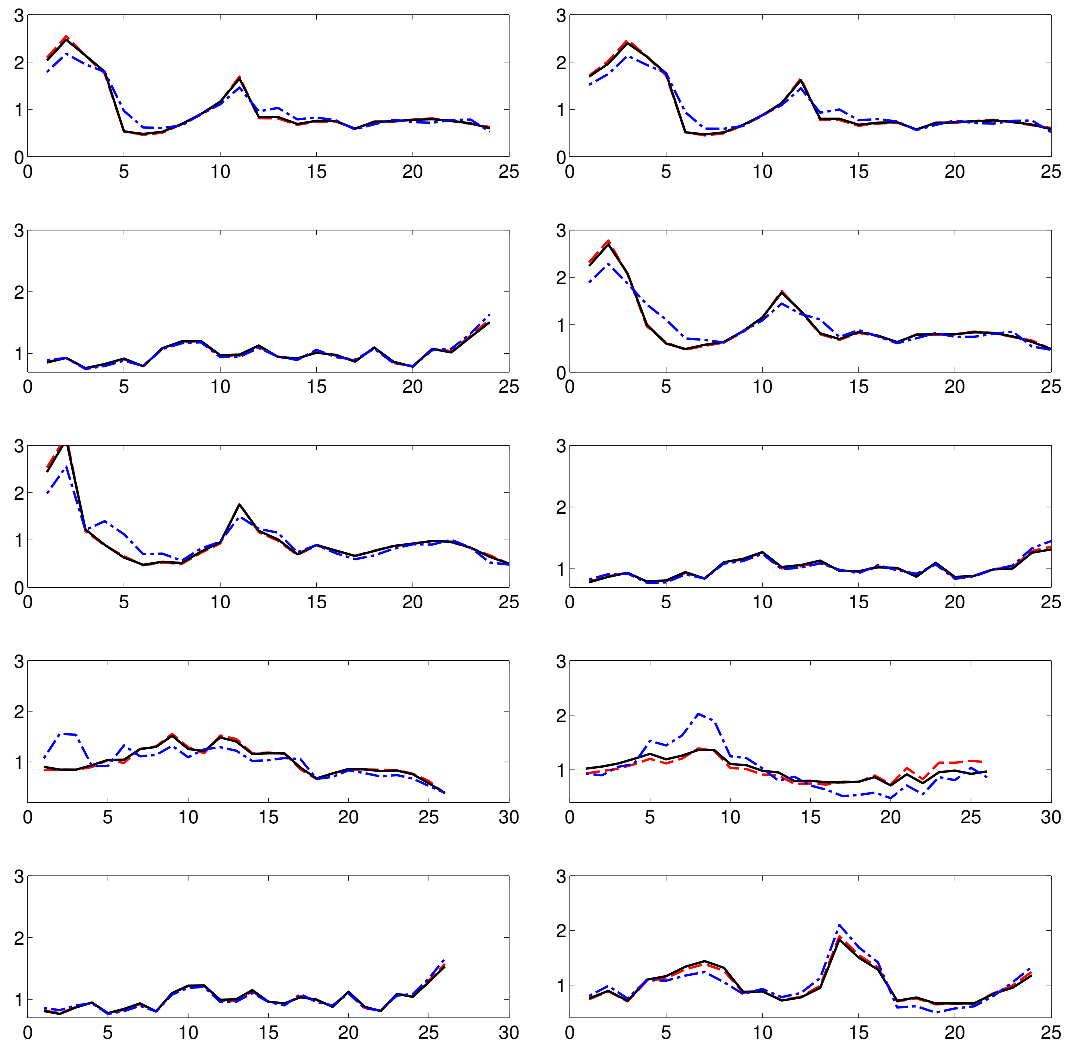


Figure 3.15: Comparison of trajectory speeds rectified using the ground-truth (*black, dashed*) and estimated parameters (*red, solid*) and Bose (*blue, dot-dashed*). Again, the 10 longest trajectories from the students 003 dataset are shown. In general, the system presented here outperforms Bose in terms of reconstruction quality.

Chapter 4

Two Plane Reconstruction

The previous chapter describes a system to estimate the parameters of a single ground-plane. In this chapter an extension of the system is presented, allowing it to work in a scene consisting of two planes; establishing the parameters for each and the boundary line between them. Whilst in principle this seems like a relatively simple extension, in practice the accuracy of the plane and the parameters of the line intersecting the planes are dependant upon one another, meaning that a poor estimate of one leads to a poor estimate of the other. This chapter discusses the various approaches we have taken to overcome this problem and the resulting system, which can determine both the plane parameters and boundary sufficiently well to allow for generalisation to a multi-planar world described in chapter 5.

4.1 Testing Multi-Planar Methods

Controlled, simulated data, generated using a multi-plane extension of the simulation technique discussed in section 3.3.1, is used to test the method. In order to allow the generation of multiple scenes, two different world-creation algorithms were implemented. The first takes input of Wavefront "obj" files, meaning realistic world scenes can be generated in 3D rendering software such as Blender and imported into the system. The second method creates a linear chain of planes, with the rotation of each plane being taken from a Gaussian distribution centred at 25°. The spread of this distribution can be altered to create more or less extremely slanted scenes.

4.2 Estimating the Plane Parameters

The previous chapter illustrated that accurate estimates for a single plane can be obtained using only the motion of pedestrians in the scene; however the work presented therein uses trajectory data from across the entire scene to obtain those estimates. Assuming there are two distinct planes within a scene, such a global approach no longer applies. Instead this work proposes the use of a sliding window across the scene, using only the sections of trajectories within the set radius of the window's centre to estimate the plane parameters. Since α is related to the intrinsic parameters of the camera rather than the planes, this is solved globally across all regions, as opposed to finding a value of α for each.

Clearly, the dimensions of the sliding window will affect the accuracy of the localised estimates; too small and the problem will be ill-posed as there will be too little information, too large and there is a high chance of the window overlapping multiple planes. In order to assess a sensible window size, a number of simulated scenes were generated (details of the two-planar extension of the simulation method are given in section 4.1), and the accuracy of the estimates produced was assessed. All scenes used in this experiment had image-dimensions 352x288 pixels – a standard size for much CCTV footage in the real world – and a focal length of 720mm. Each scene had randomly generated orientation parameters for each plane, whereby the θ and ψ of each plane was chosen randomly from within their respective feasible ranges.

As the number of planes for this experiment is known to be 2, k-means, with $k = 2$ is used to cluster the set of all estimates and obtain the correct number of hypotheses to planes. Accuracy is measured in terms of the *orientation error* obtained using the dot product of the true normal and the estimated one and the α*-error* - the difference between the true alpha and the estimated one. The results are given in figure 4.1, which shows a window size of 25-30 pixels generates minimum error. Windows above 80 pixels did not successfully generate hypotheses in these experiments.

Having obtained a number of estimates, the next task is to assign them to the relevant pixels. The initial approach was to perform a k-means clustering of all estimates,

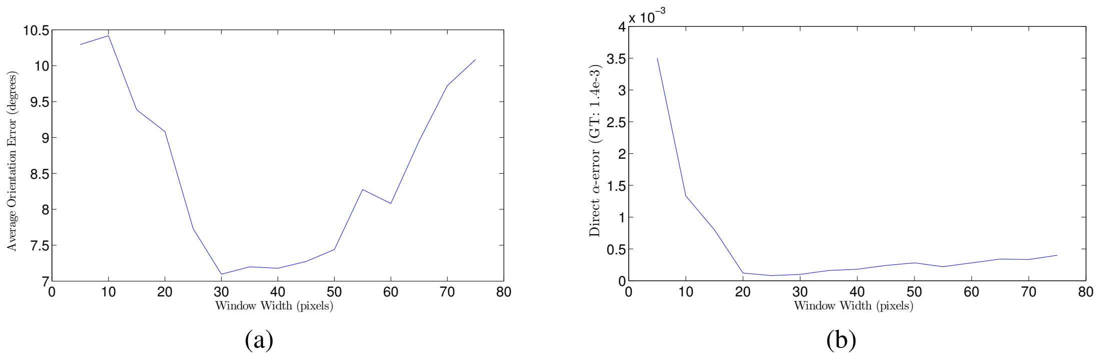


Figure 4.1: The size of the sliding window affects the accuracy of the localised estimates. Here, a number of noise-free scenes were generated and the window size was adjusted and estimation accuracy was calculated in terms of (a) the summed orientation error across both planes and (b) the error in α compared to a ground truth of 1.4×10^{-3}.

where $k = 2$, then assign the hypothesis with minimum error to each pixel given the sub-trajectories in its window. To calculate the fit of a hypothesis to a set of sub-trajectories, the error measure listed in equation (3.17) is used, as in the single-plane system.

In theory, this should be sufficient; however, in practice there are often poor estimates for some regions, which skew the cluster centres away from the true values. Many of these are a result of a region's window containing too little trajectory information, as is often the case towards the edge of a scene, which leads to an ill-conditioned system when an attempt is made to estimate the parameters. Therefore, those regions containing no trajectories with more than 4 constituent motion vectors are excluded from calculation. Estimation error also increases noticeably in regions where a region-window crosses a plane boundary. In such cases the system clearly attempts to average out the two planes, often with little success.

It should also be noted that the assignment is often not smooth – in an imperfect world the trajectory lengths in some windows will not be exactly equal so the minimum-error hypothesis is actually incorrect compared to the ground truth causing an uneven labelling. This is illustrated in figure 4.2. Therefore these raw assignments are insufficient to segment the two planes. Instead consider the problem modelled as a bipartite graph, with each pixel being a node within the graph and edges inter-connecting each pixel. The segmentation can now be generated using the well-known minimum graph-cut algorithm to find the smoothed division between the planes. In the simple case of two planes, graph-cut is sufficient; however this will not extend to more than two planes as the algorithm only considers binary labels. Since this work will later generalise the problem to that of

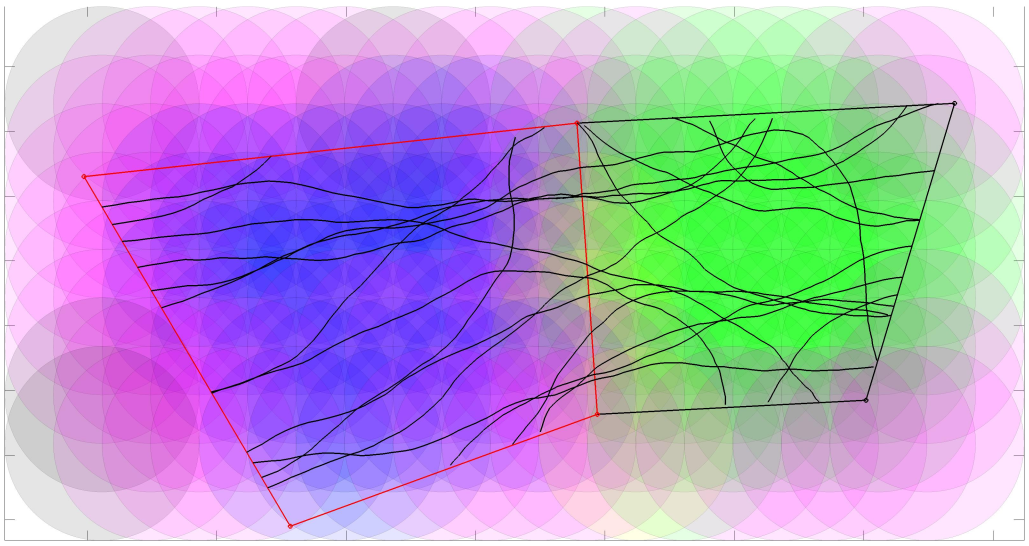


Figure 4.2: A circular sliding window is passed across the image of the scene and hypotheses are generated at each window using the single-plane method described in chapter 3. The resulting hypotheses are then clustered and the regions labelled in such a way as to minimise the error at that window (irrespective of its neighbours). Windows with too little information or for which reliable estimates could not be obtained are discounted from the process. In the example above, discounted windows are shown in pink, with the remainder of the regions labelled with one of the two hypotheses (green and blue). We see that the segmentation is fragmented and not smooth, particularly around the boundary.

more than two planes (discussed in the next chapter), the alpha-expansion [13], a generalised approximation of minimum graph-cut for more than two labels, is employed to approximate the division between the planes.

The alpha-expansion requires two terms:

The data term The cost of assigning some hypothesis, $\mathbf{f} = (\hat{\alpha}, \hat{a}, \hat{b}, \hat{c})$ to some input data

The smoothness term The cost of assigning differing labels to neighbouring pixels.

For some window w, the set of sub-trajectories contained within it is denoted $\mathbf{T}_w$. The data term is taken to be the spread of speeds across $\mathbf{T}_w$, rectified using the hypothesis assigned to w, denoted as $\mathbf{f}_w$:

$$E_w = \sum_{\tau \in \mathbf{T}_w} \left(\frac{\sigma(L^\tau)}{\mu(L^\tau)} \right)^2 + \lambda \underset{\tau \in \mathbf{T}_w}{\sigma} (\mu(L^\tau)) \tag{4.1}$$

where $\underset{\tau \in \mathbf{T}_w}{\sigma} (\mu(L^\tau))$ denotes the standard deviation of the set of mean speeds for all trajectories $\mathbf{T}_w$ in window w – this is equivalent to the prior described in equation (3.21).

For the smoothness term, the angle between the normals of plane-hypotheses is used to define whether two plane assignments are similar to each other. This is shown in equation 4.2 for the normals $\mathbf{n}$ and $\mathbf{n}'$ resulting from two hypotheses, $\mathbf{f}$ and $\mathbf{f}'$ respectively as per equation (3.1).

$$ang(\mathbf{n},\mathbf{n}') = \frac{1}{\pi}\arccos\left(\frac{\mathbf{n}\circ\mathbf{n}'}{||\mathbf{n}||\,||\mathbf{n}'||}\right) \tag{4.2}$$

This angle measurement is not sufficient on its own. In the graph being cut, all pixels are attached to all others, but when smoothing only proximal pixels need be considered. Therefore the smoothness cost is normalised over the image-space distance between the pixels being examined, meaning that distant pixels have little influence over the labelling decision. This normalisation function is shown in equation 4.3 for two imaged points $\mathbf{u}$ and $\mathbf{u}'$ and with I_{diag} representing the diagonal size of the imaged scene (i.e. the distance, in pixels, from the top-left corner to the bottom-right):

$$dist(\mathbf{u},\mathbf{u}') = 1 - \frac{\sqrt{(u-u')^2+(v-v')^2}}{I_{diag}} \tag{4.3}$$

The energy of some labelling $\mathbf{F}$ over the set of all windows $\mathbf{W}$ is calculated as below. Using the notation outlined in the Alpha-Expansion paper [13], $\mathbf{f}_w$ refers to the hypothesis assigned to window $w \in \mathbf{W}$. Again $\mathbf{T}_w$ is the set of sub-trajectories present within w.

$$E_{\mathbf{F}} = \sum_{w\in\mathbf{W}} E_w + \sum_{\{w,w'\}\in\mathbf{W}} \frac{ang(\mathbf{n}_w,\mathbf{n}_{w'})}{dist(w,w')} \tag{4.4}$$

The result of minimising the energy function $E_{\mathbf{F}}$ is a smoothed segmentation of window labels with the plane-hypotheses generated in the first step. This gives a set of regions, which, given perfect data, should be closely associated with the plane segmentation. Realistically this is unlikely to be the case, as noise within the data will cause some degree of misassignment, exacerbated by the relatively small window size, meaning some error will be observed within the hypotheses. What we are likely to have is a set of contiguous regions which should fall close to the actual plane boundaries.

Using these new, larger regions as input to the estimator, much as was done initially with the localised regions, more accurate hypotheses for the plane orientations can be obtained. Iterating the hypothesis generation and alpha-expansion stages should lead to a smooth segmentation with accurate hypotheses. There is one key problem that prevents this being the case in reality – the sensitivity of the algorithm to the smoothing term. If this is weighted too highly, the algorithm simply assigns one label to all regions, too low and the benefits of using graph-cuts are lost. Therefore, this method requires manual supervision to find a sensible weighting.

4.3 A Framework For Plane Parameter and Boundary Line Estimation

Generally speaking, if the boundary line between two planes is accurately estimated, the parameters for those planes are also accurately discovered and vice-versa. As these parameters are coupled it makes sense to solve them together.

In a bi-planar scene, there are many possible methods to feasibly solve for the boundary line. Whilst the implementation of the estimation method used may differ between each, the general framework remains the same. This section will describe the framework and the various estimation methods used within.

The previous section showed that a number of rough hypotheses for the plane parameters can be obtained using localised estimates, based on data from a sliding window across the scene. This technique forms the initialisation step of the framework. Hypotheses are generated as above and are then clustered using k-means with the number of clusters set to two; one for the estimate of each plane. Regions that have too little information to successfully obtain a result, hereafter referred to as *“empty regions”*, are excluded. The two reliable estimates are used in the boundary line estimation algorithm of choice. It is important to note that it is assumed here that each plane can be separated from another by a single boundary line. The realism of this assumption and therefore the applicability of this method in complex scenes are discussed further in section 6.3.

Having obtained an estimate of the boundary line, a segmentation of the scene into two distinct planar regions can be produced. Each of these regions now has a set of sub-trajectories associated with it. These new regions can be used as input to re-estimate both the plane hypotheses and the boundary. As each new region contains considerably more trajectory information than is available in the localised sliding windows, the estimation is generally much more robust (given a reasonable estimate of the boundary line). The increase in error around the boundary line is also notably less influential on the estimations

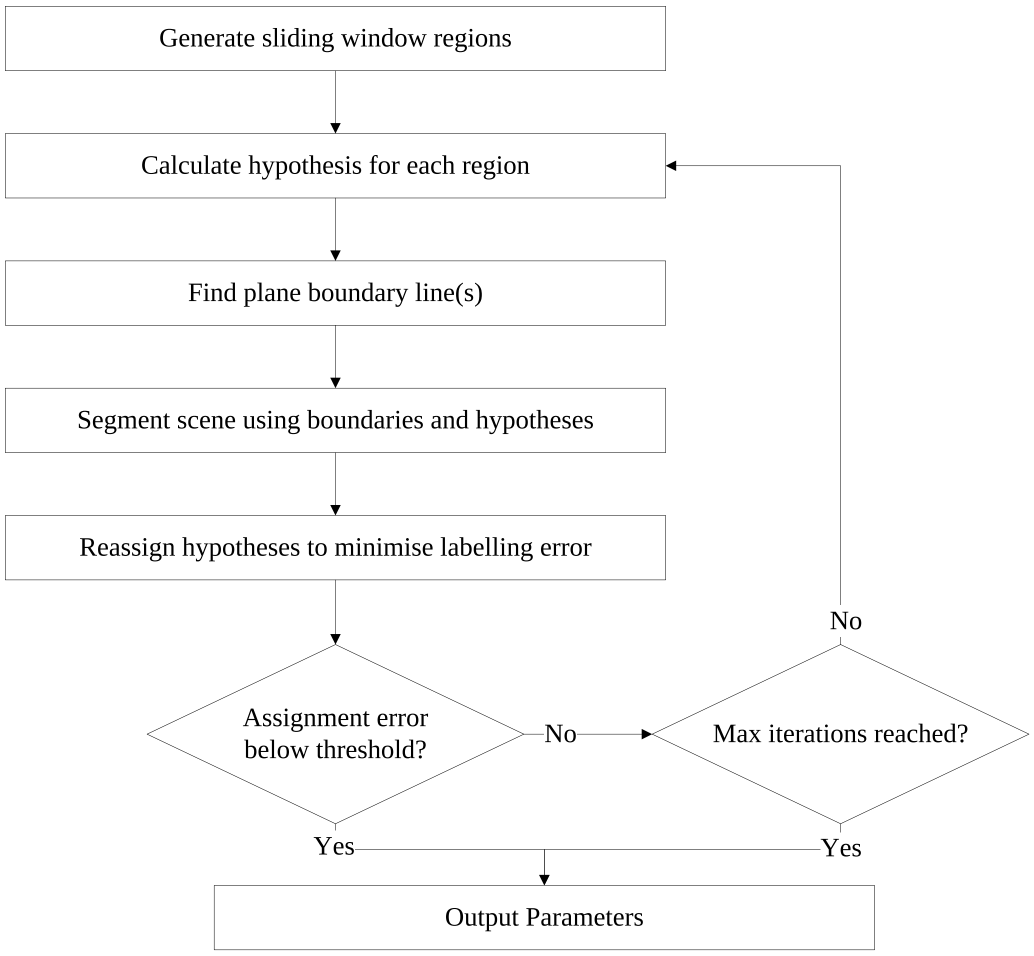

Generate sliding window regions
Calculate hypothesis for each region
Find plane boundary line(s)
Segment scene using boundaries and hypotheses
Reassign hypotheses to minimise labelling error
Assignment error below threshold?
No
Max iterations reached?
No
Yes
Yes
Output Parameters

Figure 4.3: By iterating the location of boundary lines and plane parameter estimation, more accurate results can be obtained than by the initial labelling. This is because the local windows are generally much smaller than the segmented regions causing the non-linear optimisation to be less well conditioned.

as the number of trajectories within the region that are not near the boundary tends to be much larger. By iteratively repeating this procedure as per the flowchart in figure 4.3, considerable accuracy improvements can be obtained over simply using the localised estimates used to initialise the framework.

This section has thus far introduced the general estimation framework, but not the boundary line estimation techniques vital to obtain an accurate estimation of the plane parameters. The following sections will address the various techniques used within this work to estimate the boundary line.

Table 4.1: Quantitative comparison of true and estimated parameters for the two-plane scene pictured in figure 4.4 at various levels of speed variation.

	Plane 1		**Plane 2**		
	θ	ψ	θ	ψ	α
Ground Truth	30.0	15.0	40.8	-32.9	0.0014
0% Speed Variation	29.9	15.2	40.8	-33.0	0.0014
10% Speed Variation	31.1	14.3	40.6	-33.2	0.0015
20% Speed Variation	31.1	16.3	40.0	-31.6	0.0017

4.3.1 Exhaustive Search

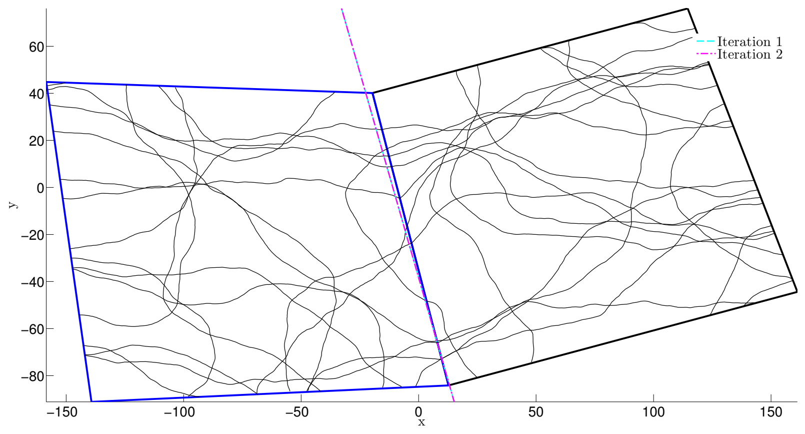
We consider the problem of finding the boundary line in terms of its direction vector and a point upon it. This could be posed as an exhaustive search across the scene, moving and rotating a line across the scene to find the location that minimises the cost of assigning the hypotheses from the previous iteration to the segmented regions. This search is slow so only lines that pass through the central horizontal and vertical scan-lines are considered as it can be reasonably assumed that in most cases the boundary line will pass through at least one of these. In addition, the rotation of the line is initially discretised into 5° increments, before refining the search around the global minimum.

Qualitative results for the accuracy of the boundary line estimation on simulated data are shown in figure 4.4 with the qualitative measurement of plane orientation accuracy shown in table 4.1. The method is shown to work well on clean data and is robust to variations in speed provided the resulting hypotheses are reasonable; however, this method is inefficient and the simplifying assumption that the boundary passes through either the horizontal or vertical scan-lines means it is not robust to all scene configurations.

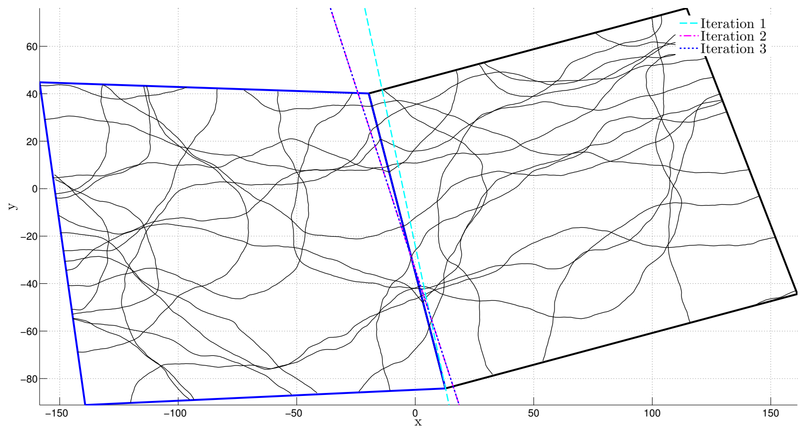
4.3.2 Combining Boundary and Orientation Parameters

Let the boundary line be represented as a homogenous image point upon the line $\mathbf{u} = (u, v, 1)^\top$ and its direction in terms of a rotation φ from the vertical. Clearly in a "perfect" world, where there is no variation in the speed of individual tracked features, the global error minimum should fall where the plane parameters and boundary line are correct. As a result, it should be possible to combine the boundary line parameters with the plane parameters and solve the entire system at once. This gives the new parameter set $(\alpha, \theta_0, \psi_0, \theta_1, \psi_1, d_1, u, v, \varphi)$ to be estimated by the non-linear solver.

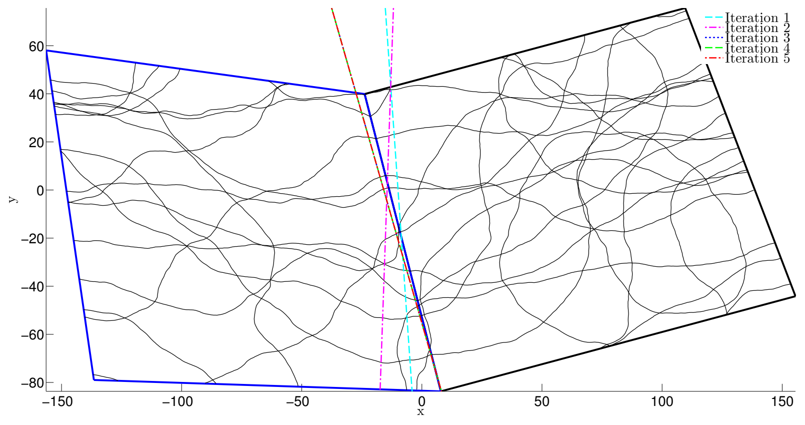
To initialise the plane parameters, hypotheses from the first step of the framework are



(a)



(b)



(c)

Figure 4.4: The most robust method for finding the boundary between two planes is an exhaustive search across the scene, rotating the line at each position and assigning one hypothesis to either side of the line. The configuration of position, angle and hypothesis labelling with minimum error is assumed to be at the boundary. When used in an iterative framework this process converges quickly given perfect data as seen in 4.4a. At higher levels of speed variation (10% and 20% in (b) and (c) respectively), more iterations are required but the algorithm still converges on a very close approximation. In all experiments, a window size of 30 pixels is used. The quantitative results for the three experiments pictured here are given in table 4.1.

taken. Initialising the boundary position and direction is more difficult as no prior knowledge is given as to where it may lie within the scene. To find a potential initial site and orientation for the line a fast, coarse version of the aforementioned exhaustive search method was used, with the point of lowest error being chosen to initialise the boundary line.

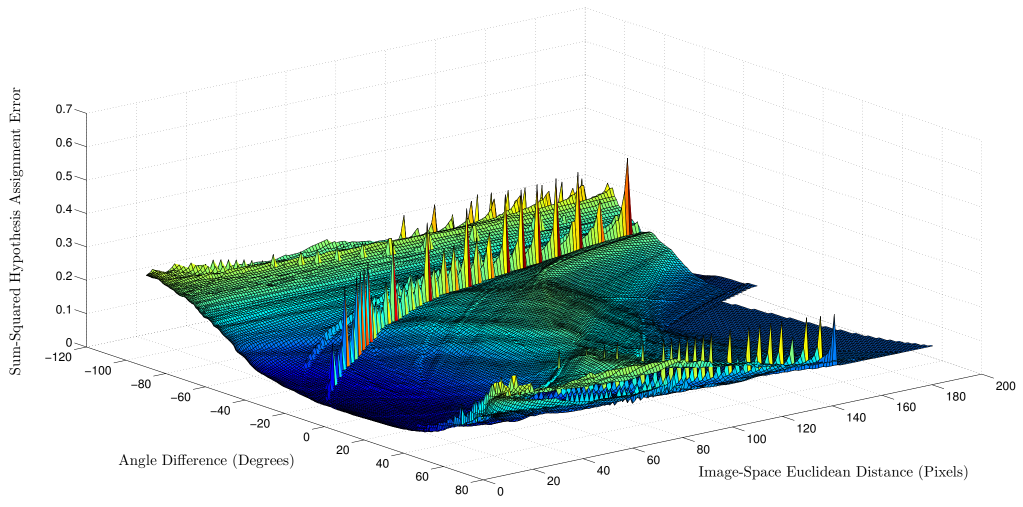
To examine the feasibility of this estimation, the error-volume of the boundary-line parameters alone was calculated, first with all other parameters fixed at their true value (figure 4.5a), then with hypotheses generated with 20% speed variation (figure 4.5b). Experimental results show that the problem space is smooth in places, but very rough in others, meaning that this technique is particularly sensitive to the initial conditions, even with the plane orientations fixed. On perfect data, the result is often still not exact even when the initial estimate is close to the true answer, as can be seen in figure 4.6. Once variation in speed is applied to the trajectories used in the non-linear optimisation, it rarely converges before reaching a local minimum, even at relatively low levels. This makes this method infeasible in scenarios with anything but perfect data and therefore, it will not be used in the forthcoming experiments. Instead a new technique will be introduced, which considers the problem of the boundary line as though finding the separating hyperplane in supervised classification data.

4.3.3 Training an SVM to Segment the Plane

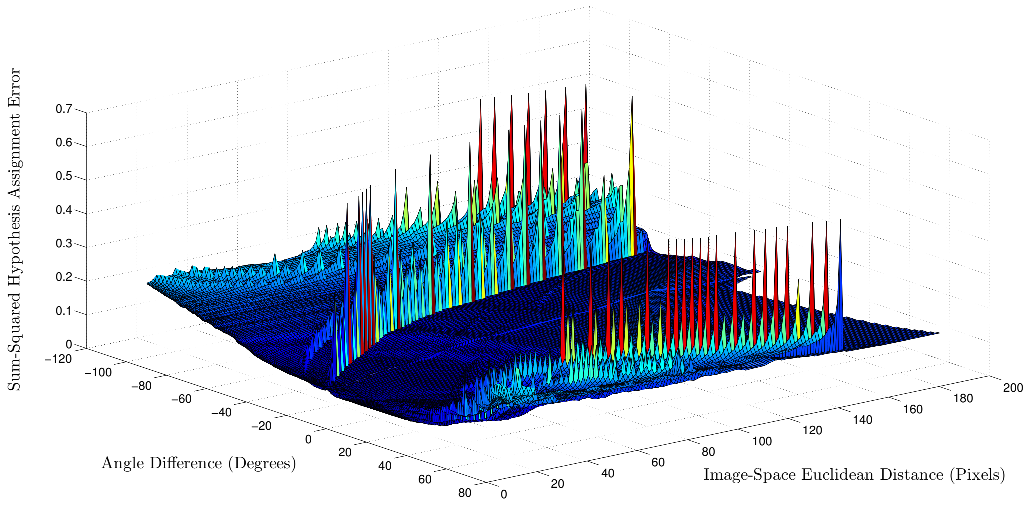
Consider the problem of identifying the boundary between planes as a binary segmentation, with the separation between segmented regions defined by a linear function. The intention of this method is to label pixels to one side of the boundary with one label, and those on the other with the other label. As the parameters are not exact and the input data is not perfect, some degree of mislabelling is expected, particularly around the boundary.

Support Vector Machines (SVMs) [22] have proven popular in recent years for learning the boundary hyper-plane between two classes, particularly in cases where the initial labelling is imperfect. By introducing slack variables to account for some error in the supervised labelling, the system obtains a “soft-margin” between the two classes, balancing the margin between the groups and the label assignment error.

This property of SVMs can be exploited in order to find the line between the pixels labelled with each plane hypothesis. The SVM is trained in the 2D image domain and the resulting separation line between the labelled pixels in the image represents the line, again



(a)



(b)

Figure 4.5: To examine the feasibility of combining the boundary line parameters with the orientation estimation, the orientation parameters were fixed and the error-space of the boundary line parameters examined. For representation, the euclidean distance of $\mathbf{u} = (u, v)$ and the difference in φ in degrees from their true values are used as the metric. The figures above show error in reconstructed trajectory speeds, in comparison to the distance of the parameter set from the true answer. The error-space was examined for two scenarios: (a) used precise orientation parameters and (b) used hypotheses generated at 20% intra-speed variation. It can be observed that the error space is largely smooth, although there are deep local minima and sharp maxima, particularly in the latter case, requiring careful consideration of the initial parameters for the optimisation.

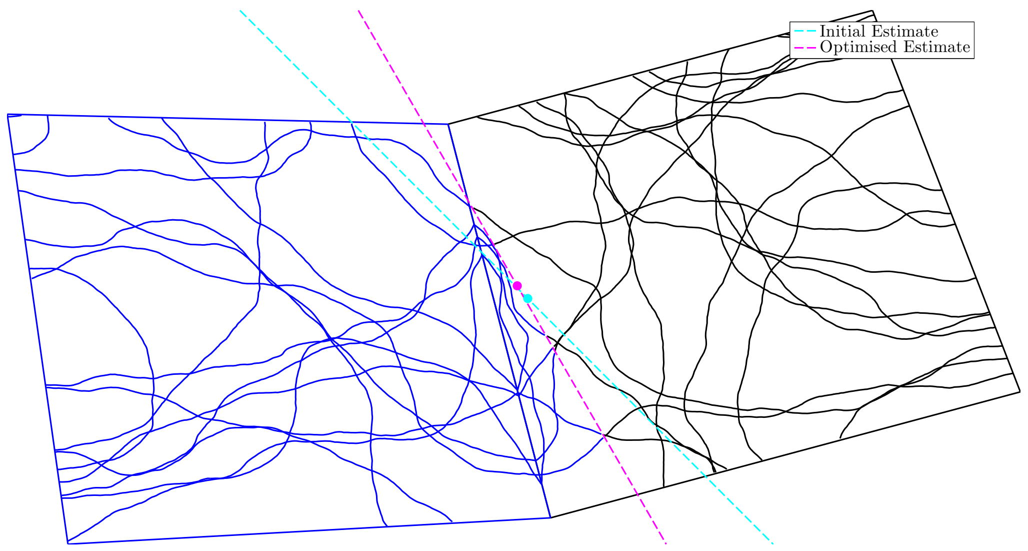


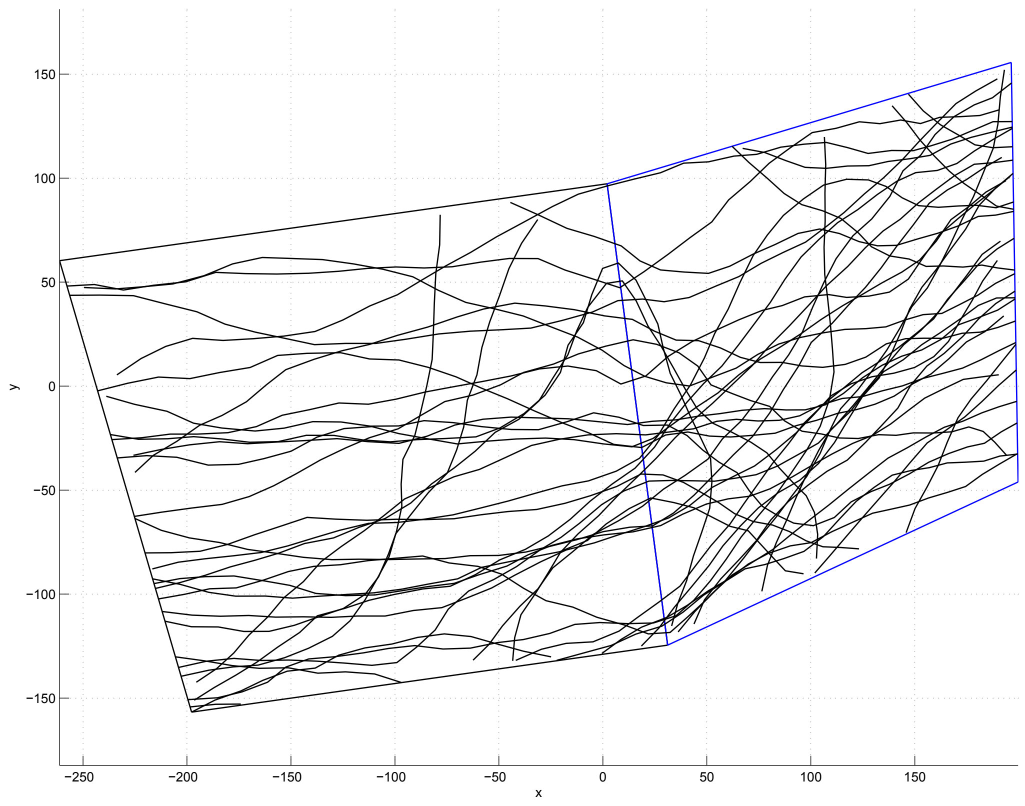
Figure 4.6: An initial estimate for the boundary line is produced using a coarse version of the exhaustive search algorithm described in section 4.3.1. The boundary line parameters in terms of a point on the line and its rotation in the image are then appended to the parameter set for the plane orientations before a solution is sought using a non-linear optimisation. Even on perfect data, this method fails to produce a reliable estimate of the boundary line due to the irregular nature of the error space.

in image-space, best separating the pixels assigned those two labels.

Given two hypotheses obtained as discussed in the previous section, a raw labelling is produced by assigning each pixel the minimum-error hypothesis. That is, input to the SVM training step is a set of (u,v) image coordinates and their estimated best-fit label. Now, the SVM is trained on this labelling to obtain the soft margin between those pixels labelled with the first hypothesis, and those labelled with the second. This soft margin is assumed to represent the boundary between those two planes in image space, which clearly can then be rectified using the estimates of the plane parameters to back-project it into camera coordinates. An example of this process is shown in figure 4.7, in which the two labels are indicated by red and green data points, with the support vectors obtained during training highlighted by black circles around each relevant data point.

As the initial labelling is likely to be erroneous as the localised windows are relatively ill-posed in terms of the overall scene, this process can be iterated to improve accuracy, with the vast majority of experiments showing improved accuracy with iteration.

With perfect data, a very accurate estimate of the plane parameters is produced and therefore the boundary parameters are also closely approximated, meaning that the planes line up well and almost precisely match the true reconstruction as can be seen in figure 4.8,



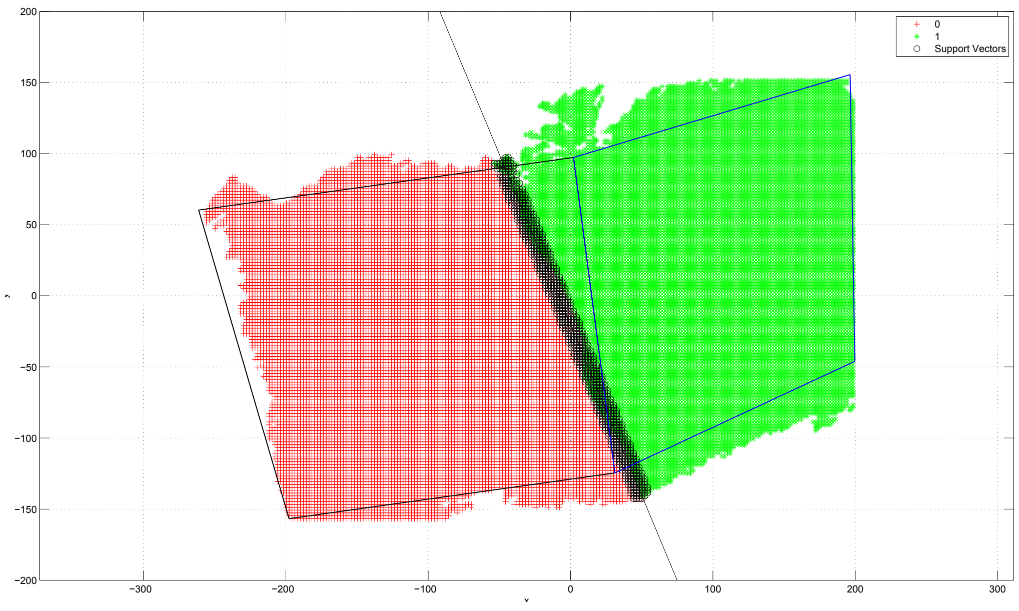


Figure 4.7: An example of the first iteration of the SVM based boundary location method on input data under 10% speed variation. The image project of the planes and trajectories is shown (top), with the labelling indicated by red and green data points at each pixel(bottom). The support vectors obtained by training the SVM on this labelling are indicated by black circles around the relevant pixels. The boundary line is approximated reasonably, although further iterations are required to refine it.

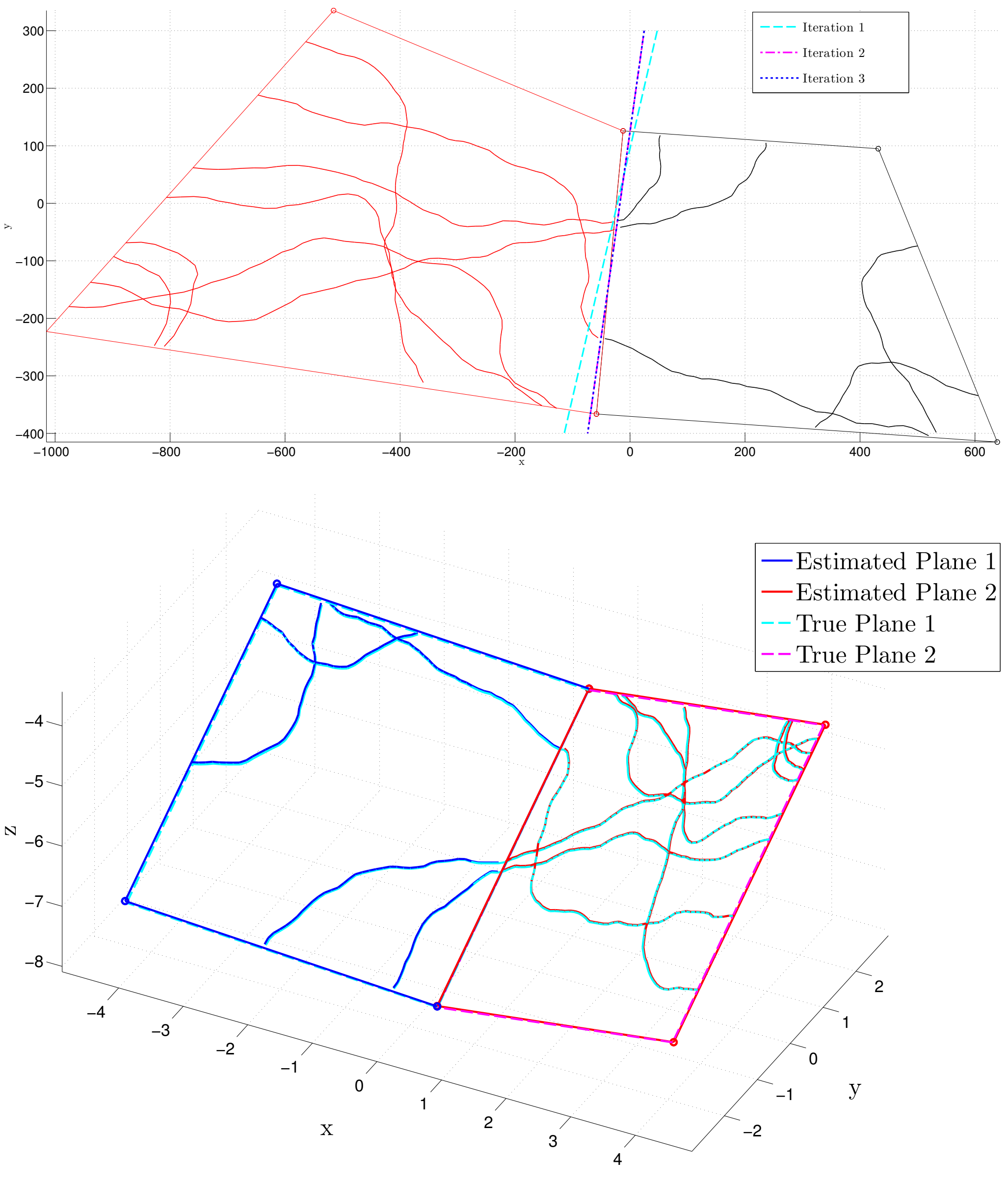


Figure 4.8: Given "perfect" data (that is, data with zero intra-trajectory speed variation), the boundary can be precisely located using a Support Vector Machine. Under these conditions, the system performs a perfect reconstruction of the true planes and their trajectories.

Table 4.2: Quantitative comparison of true and estimated parameters for a two-plane "perfect" scene, pictured in figure 4.8.

	Plane 1		**Plane 2**		
	θ	ψ	θ	ψ	α
Ground Truth	36.0	-4.0	-32.2	-42.6.	0.0014
Estimated	35.8	-3.9	-31.9	-42.4	0.0014

Table 4.3: Quantitative comparison of true and estimated parameters for a two-plane scene at 20% noise, pictured in figure 4.9.

	Plane 1		**Plane 2**		
	θ	ψ	θ	ψ	α
Ground Truth	30.0	-3.0	38.3	-46.0	0.0014
Estimated	23.0	2.7	38.1	-43.9	0.0015

the numerical results of which are given in table 4.2.

As variation in speed increases, the plane estimations clearly become less precise; however in a bi-planar world, this is coupled with a new issue – the planes no longer line-up along the plane boundary as illustrated in figure 4.9, the quantitative results of which are given in table 4.3.

4.4 Enforcing Plane Alignment at the Boundary

Previously when estimating planes, the value assigned to d was arbitrary, only providing scale. When estimating multiple planes in the same scene, this condition only applies to one of them, hereafter referred to as the "reference plane". All additional planes introduce a further parameter to the estimation procedure – their scale relative to d. Therefore the set of parameters to be solved within a bi-planar system is now $(\alpha, \theta_0, \psi_0, \theta_1, \psi_1, d_1)$.

Including d_1 in the estimation does not in itself prevent drift between the planes; it merely allows the second plane to vary in scale. To this end, a further prior is included in the error function for estimation. Let $\mathbf{u}_1$ and $\mathbf{u}_2$ be two image points on the line of intersection between the two planes. In addition, let $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ be those points back-projected with the estimated parameters for the reference plane. The prior dictates that the second plane, when reconstructed using the estimated parameters, passes through these two points. In the following, let $\mathbf{n}_1$ be the normal for the second plane, and d be its distance from the camera:

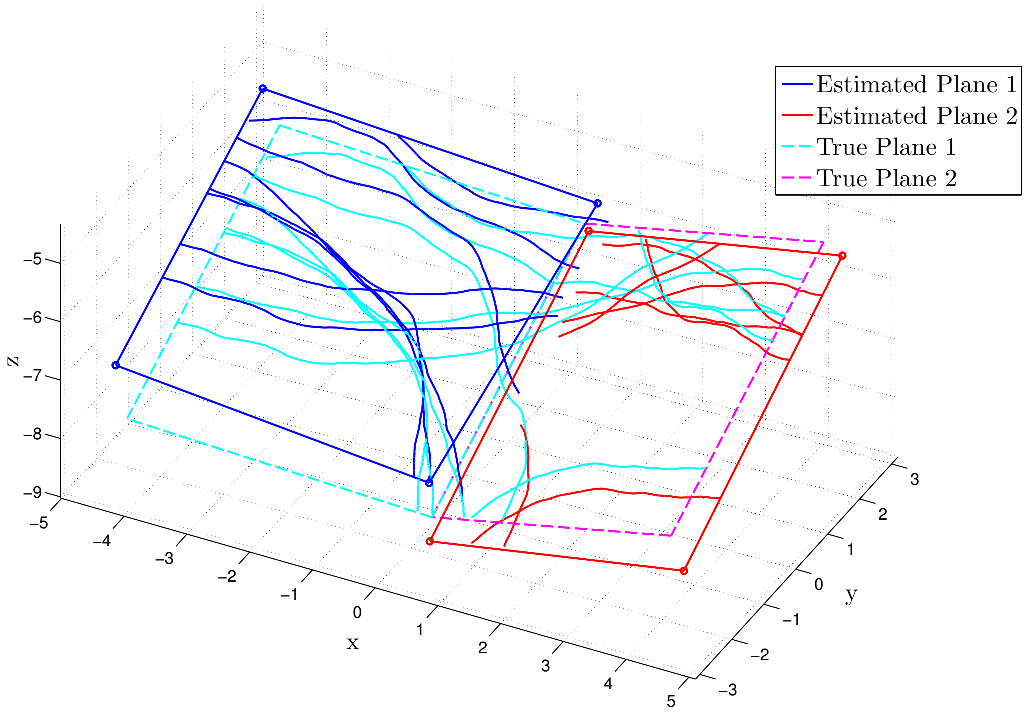


Figure 4.9: Given clean data and using the SVM based line estimation method a perfect estimation of the true planes can be calculated (see figure 4.8). However; introducing variation in intra-trajectory speed skews the estimation causing an imperfect estimation. Under these conditions the two planes drift apart from each other, indicating some criterion is required to enforce alignment at the plane boundary.

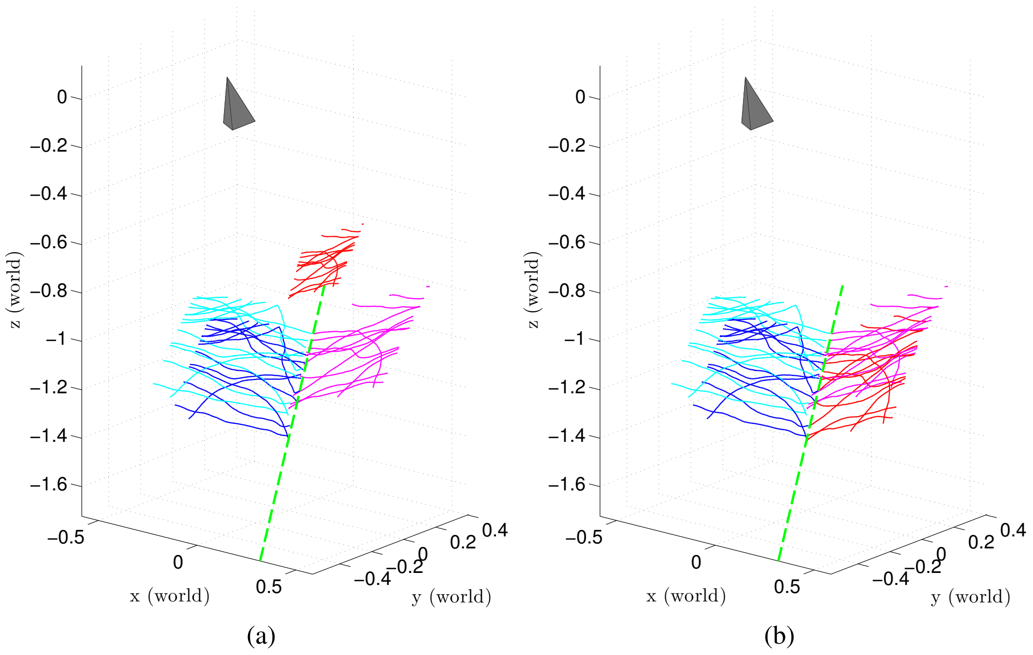


Figure 4.10: By applying a prior to the optimisation function, dictating that the plane should pass through a point on the boundary line (rectified using the hypothesis for the reference plane), the reference and second planes now intersect at the boundary. (a) shows the estimation without the prior and (b) shows the same data but with the second plane estimated with the prior applied. The camera is illustrated by the dark-grey pyramid.

$$dist_{boundary} = \sum_{i \in (1,2)} \frac{|\mathbf{n}_1 \mathbf{x}_i^* - d|}{||\mathbf{n}_1||} \tag{4.5}$$

Figure 4.10 illustrates the effectiveness of this prior on an estimation of parameters using input data under 20% speed variation. The second plane is estimated such that it correctly intersects the back-projected boundary line and the two planes are suitably aligned. Whilst this method proves to be effective, it is inefficient. The system to be solved is being constrained by the prior, but the same number of parameters must still be solved to estimate the orientation of the second plane.

Instead reconsider the situation. Given an approximation of the reference plane and the boundary between it and a connected plane, the orientation of the second plane is simply a rotation of the unit-normal of the reference plane about the back-projected boundary line. In this manner, intersection at the boundary can be enforced and the number of unknowns to be solved can be minimised. The following chapter will detail how this was achieved and how accurate the solution was.

4.5 Conclusions

The contribution of this chapter has been an iterative framework for estimating the relative position and orientation of planes in the world, given the prior knowledge that there are exactly two in the view of the camera. The presence of multiple planes introduces problems that do not arise in a single-plane world – namely one can no longer use input trajectories from across the entire scene and that finding the location of the intersection line between the two planes relies on reasonable estimates for the plane orientation parameters and the camera focal length.

Several methods were discussed for locating the intersection line. Exhaustive search is a possibility, but it is computationally expensive, even when simplifying assumptions are introduced to minimise the search space. Including the boundary parameters into the non-linear optimisation works to some extent with perfect data; however as the boundary line and plane parameters are dependant on one-another, noisy data such as that likely to arise from real-world situations badly affects the quality of the result. The most promising method involves training a Support Vector Machine, using a minimal-error independently labelling assigned to each region. The SVM approach was the most robust to noise as shown in results on controlled simulated data.

The limitation of this framework as currently described is the necessity for there to be exactly two planes in the scene. Instead it would be more useful in a real-world scenario to automatically determine the number of planes prior to segmentation. This problem is the focus of the following chapter, in which the existing framework is generalised to account for any number of flat planes in the scene.

Chapter 5

Reconstruction of Hinged Planes

The previous chapter discussed various methods to determine the parameters of two planes and the boundary line between them. It also highlighted the issues associated with multi-planar estimation, namely that given error in the input data, the two planes rectified with estimated parameters may not align correctly. The end of the chapter alluded to an alternative approach to solving the multi-planar estimation problem whereby the parameters of the second plane are calculated as a rotation of the reference plane's unit-normal about the boundary between the two planes. This chapter will discuss this concept in the domain of a bi-planar world, before extending the technique to apply to a more generalised world where the number of planes is not known in advance.

5.1 Hinging in a Bi-Planar World

In section, 4.3, a framework was introduced for determining the intersection line between two planes using a variety of methods. Hereafter, the SVM-based method shown in section 4.3.3 is used to find the boundary line. Once an estimate for the reference plane and the boundary line are obtained, the estimation of the second plane is reposed using the hinged formulation discussed below.

Given the estimated parameters for the reference plane θ and ψ, it is trivial to obtain the

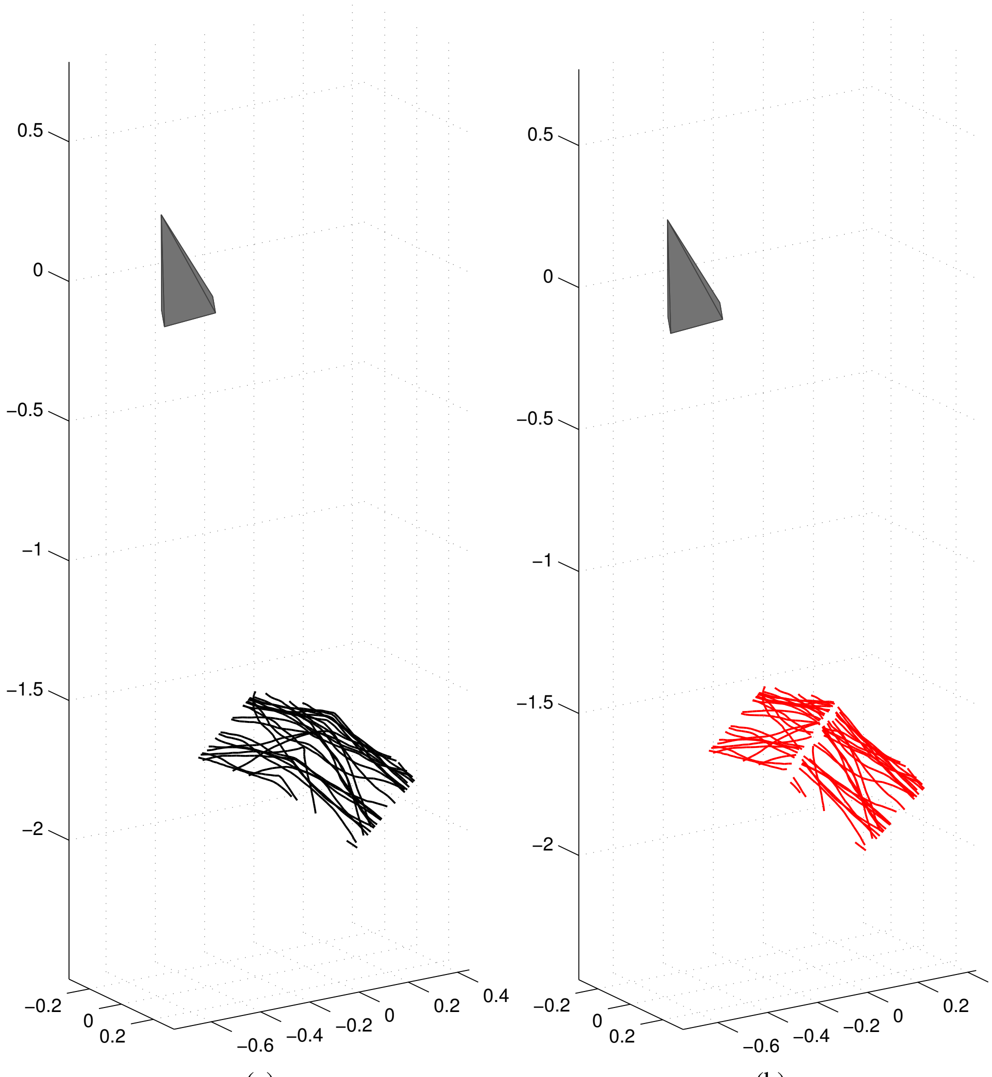
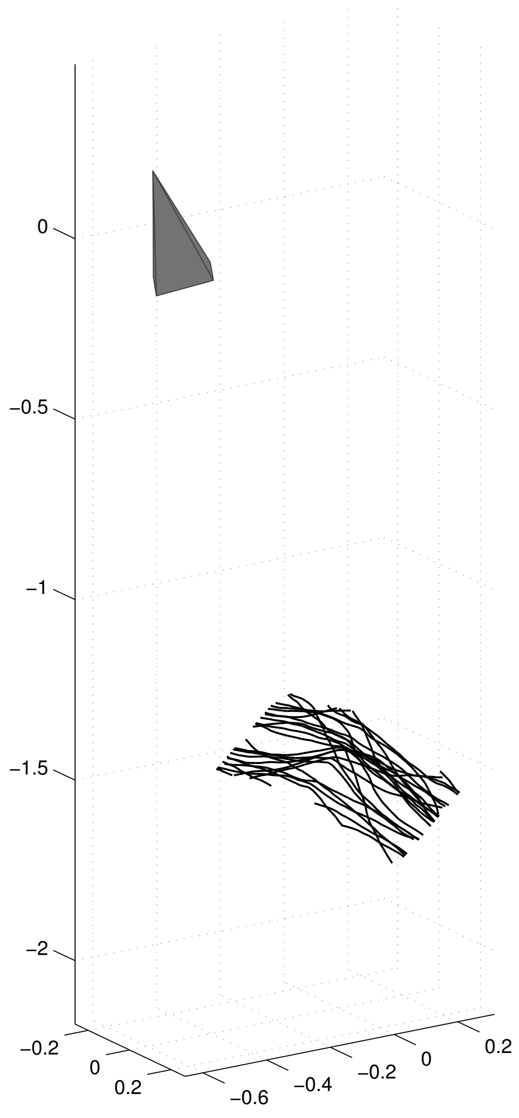
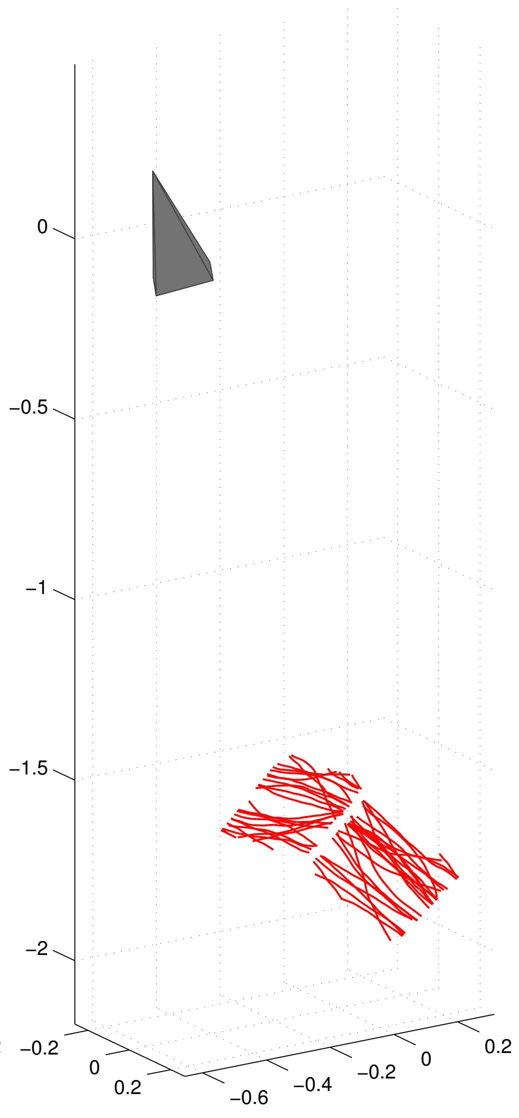


Figure 5.1: Ground-truth (left) and estimated (right) reconstruction of a bi-planar scene using the hinged estimation framework. Input data here was “perfect”, i.e. there was no speed variation. Quantitative analysis of the estimated parameters is given in table 5.1.



(a)



(b)

Figure 5.2: Ground-truth (left) and estimated (right) reconstruction of a bi-planar scene using the hinged estimation framework. Input data was under the influence of 20% intra-trajectory speed variation. The result is still very close to the ground-truth although there is some orientation error. Quantitative analysis of the estimated parameters is given in table 5.1.

Table 5.1: Quantitative comparison of true and estimated parameters for a two-plane scene at 20% speed variation, pictured in figures 5.1 and 5.2.

	Plane 1		**Plane 2**		
	θ	ψ	θ	ψ	α
Ground Truth	30.0	15.0	49.2	75.1	0.0014
0% Speed Variation	32.2	12.1	50.0	78.7	0.0014
20% Speed Variation	39.4	16.0	42.7	69.6	0.0017

estimated normal to the reference plane, $\mathbf{n}_0$ using the mapping presented in chapter 3, equation 3.1. The boundary line rectified with the estimated parameters is denoted by the direction vector $\mathbf{b}$ and a point upon it $\mathbf{x_b}^*$.

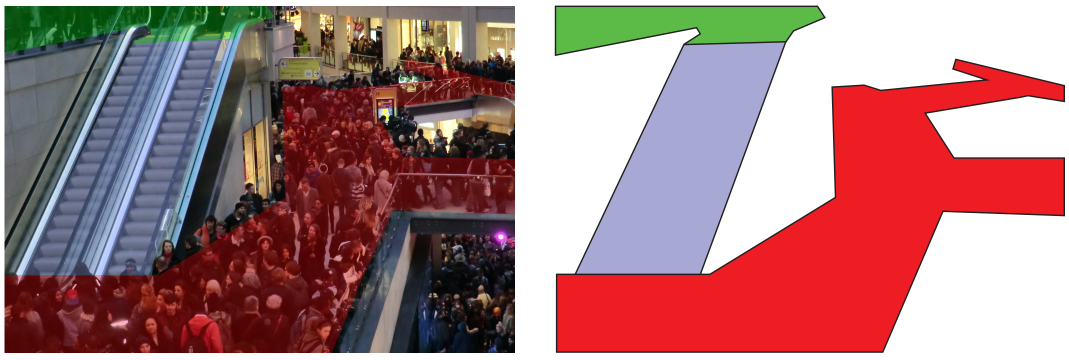
Assume the angle of rotation ϕ about the boundary line, between reference plane and the second is known. Finding $\mathbf{n}_1$, the normal of the second plane, is simply a case of rotating $\mathbf{n}_0$ about the boundary. In order to solve for d_1, a point known to lie on the rotated plane is needed. As the boundary line is known to lie upon the second plane, any point upon it $\mathbf{x_b}^*$ can be used. Therefore, given $\mathbf{n}_1$ and $\mathbf{x_b}^*$, d_1 can be found by the plane equation $\mathbf{n}_1 \circ \mathbf{x_b}^* = d_1$.

Finding the optimum rotation is a simple extension of this method. Searching values of ϕ between 0 and 180° allows us to find the angle that produces a plane orientation minimising the trajectory error function given in the previous chapters. Not only does this method of search improve upon the previous iterative method discussed in chapter 4, it is less computationally expensive as there is only one additional unknown to search instead of three.

Qualitative results are given on perfect data in figure 5.1 with figure 5.2 showing the result at 20% speed variation. The results are quantified in table 5.1. Note that whilst the estimations obtained by this algorithm under the influence of speed variation are lower (since the system has to skew both planes to allow them to align correctly), the actual reconstruction is closer to the true solution due to the hinging constraint.

5.2 Beyond the Two-Plane World

The system mentioned so far can solve for a bi-planar world with a reasonable degree of accuracy, yet many real-world scenes, such as the one in figure 5.3 contain more than two planes and it would be useful to have generalised system that does not need to know



(a) (b)

Figure 5.3: Some scenes cannot be modelled by a single plane and require a more complex topology. This example, taken at the Trinity Shopping Centre in Leeds shows such a scene.

in advance the number of planes. This section focuses on the development of a system with these stipulations in mind. The work is again based on the framework discussed in the previous chapter; however it must now be generalised such that any number of planes and the boundaries between them can be detected. Hereafter, it is assumed the world can be modelled as a tree of connected planes in which each node is considered to be a plane, with edges representing the boundary lines between them. The reference plane forms the root-node of the tree. Some examples of such graphs are given in figure 5.6. Implicit in this assumption is the further constraint that no combination of planes connect in a cyclic manner. The simulations presented here provide examples of relatively simple scenes – all planes are of approximately equal size in the world and the layout allows the reconstruction algorithm to be applied sequentially along a linear boundaries.

5.2.1 Hypothesis Clustering For Multiple Planes

In the aforementioned framework, the number of clusters is fixed as the number of planes is known *a priori*. In the generalised system, this knowledge is no longer present and therefore the clustering approach must be adapted accordingly. Various iterative extensions to the traditional k-means approach exist to hierarchically determine a sensible number of clusters to fit the data. One can find the point where increasing the number of clusters by splitting them, no longer increases the distance between clusters in the parameter space [108]. Many start with one cluster and split until some fitting criterion is met, such as each cluster fitting a Spherical Gaussian distribution. This could be done through direct computation [50, 36] or using a statistical model such as the Bayesian Information

Criterion [97] to assess the quality of a split. In this work we use the G-Means method described in [50], which begins with $k = 1$, and increases k until the clusters created fit with the Gaussian assumption.

5.2.2 Differentiating One Plane from Another

In the earlier case where the scene was known to have two planes, separating them is relatively easy as simple K-Means clustering can be used to enforce the detection of two different hypotheses. However, when taking a generalised view, one cannot know in advance how many hypotheses should be generated, hence the use of the G-Means algorithm discussed in the previous section.

The ability of this method to differentiate between two planes is affected by the angular difference between them (in terms of the rotation about the hinge axis). In cases where the two planes are almost parallel, one would expect the algorithm to smooth these hypotheses together, essentially merging the planes. Clearly this is undesirable, as over several hinges, error will propagate, causing poor final reconstructions. In order to examine the effect of hinging angle on the system's ability to differentiate planes, an experiment was run for bi-planar scenes with different angular differences. In this test, 100 two-plane configurations were simulated at each required angle. The number of resulting hypotheses was calculated using G-Means and the mean number of estimated planes plotted against the angle between the planes in 3D space. One would expect that the accuracy of the estimation would decrease as the amount of speed variation in the input data increases. As such this experiment was carried out on data with zero, 10% and 20% speed variation.

From figure 5.4, it can be observed that the system requires between 15° and 20° between the planes to reliably obtain estimates. As one would expect, the zero-variation results are show the least variation in the number of estimated planes, providing reliable estimates at around 15° difference. At 10% variation, more difference is required as the hypotheses generated prior to clustering are exhibit larger spread. An example of this can be observed in figure 5.5, in which 3D histograms of the estimated parameters (in terms of θ and ψ) are given, for zero and 10% variations. This is even more prevalent in the 20% variation experiment.

As the angle between planes nears 90°, the ability of the algorithm to correctly estimate the planes diminishes as the angular difference is on the limit of the search-space for the hinged rotation. However, in real-world scenarios such large differences are unlikely to

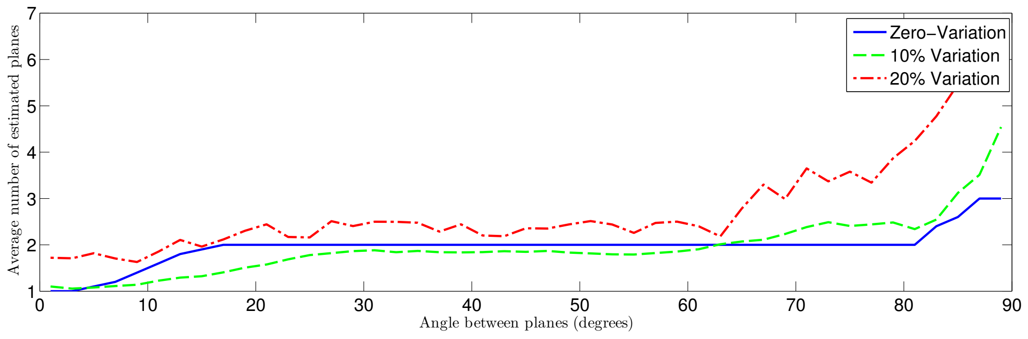


Figure 5.4: The angle between the planes in the world affects the ability of the system to differentiate them. Here, the number of planes estimated is compared against the angle of the planes in simulated scenes, averaged over 100 configurations at each angular interval. Three levels of speed variation were examined, zero variation (solid, blue), 10% (dashed, green) and 20% (dot-dashed, red). Depending on input speed variation, the algorithm tends to require between 15° and 20° degrees difference before it reliably estimates the correct number of planes. As the angle reaches 90%, the system struggles to obtain quality estimations as this is at the limit of the search space.

be part of the motion plane.

5.2.3 Hinged Rotations for Plane Estimation

Suppose that the location and orientation of the boundary lines is known. In order to perform the hinged rotation estimation, one must first obtain the location of the reference plane and the plane connection tree. As the graphs are assumed to be acyclic, for N_P planes, there are $N_P - 1$ boundary lines, each with their own rotation. This leads to the set

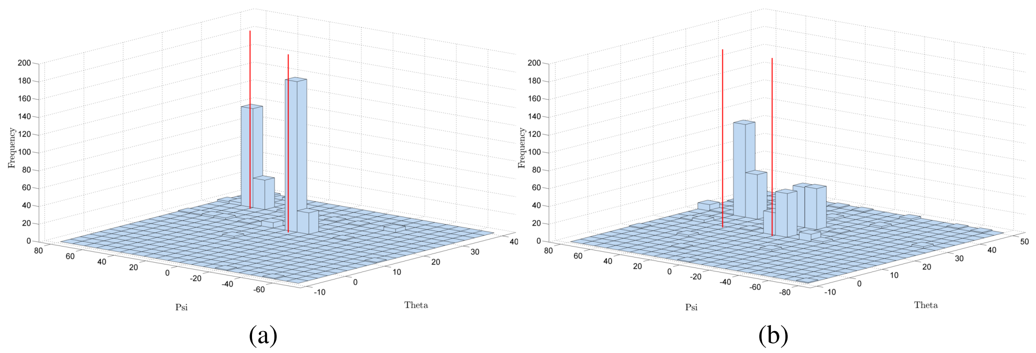


Figure 5.5: At higher levels of input speed variation, the full set estimated parameters across all localised windows show more spread. Above are illustrative 3D histograms of the approximated values for θ and ψ under (a) zero variation and (b) 10% speed variation.

of unknowns $\Phi = (\phi_0 \dots \phi_{N_P-1})$ to be calculated.

Assume that all ϕ_i are known and reliable estimates for the boundary lines and reference plane parameters have been determined. A reconstruction is generated by first rectifying all planes by the hypothesised parameters of the reference plane. Then each set of children of that plane is rotated in turn by the edge connecting it to its parent. As an illustrative example, consider the plane configuration given in figure 5.6b. Let P_i represent the plane at node i, where $i \in (0, ..., N_P)$, and $E_{i,i'}$ give the edge between P_i and $P_{i'}$ in reconstructed 3D space. First all planes are rectified by the parameters for the root plane P_0. Next P_1 and P_2 are rotated about $E_{0,1}$, before finally P_2 is rotated about $E_{1,2}$.

The quality of such a reconstruction is assessed by examining the spread of trajectories within each plane. The resulting spread values are then squared and summed to give a final metric for the fit. An exact reconstruction on perfect input trajectories would result in zero sum-squared-spread, with an increasing value showing a worse reconstruction quality.

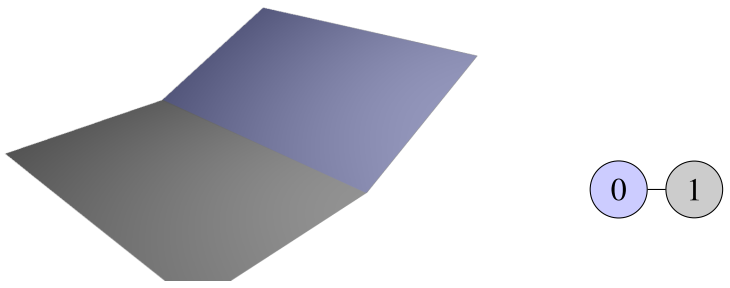
Using this error metric, one can perform a non-linear optimisation over Φ using the Levenberg-Marquadt algorithm as discussed in previous chapters. This introduces another problem - how can such a parameter set be initialised? The rough plane hypotheses generated while estimating the boundaries are assigned to each plane, in order to minimise the assignment cost (the case where the same hypothesis is assigned to more than one plane is discussed further in section 5.2.4.2). The unit-normal of each plane is obtained, allowing an approximation of the angle between connected planes to be calculated. The set of these approximations is then used to initialise the optimisation over Φ.

5.2.4 Generalised Boundary Location Algorithm

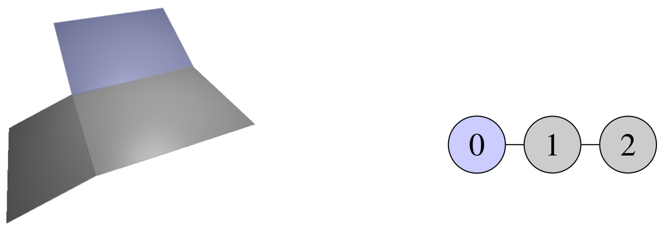
Having found some number of hypotheses, the boundaries between the planes must be found. This section discusses the various approaches trialled to discover the locations of the plane intersection lines without foreknowledge of the number of planes or their locations prior to calculation.

5.2.4.1 Capitalising on the Error Around Boundaries

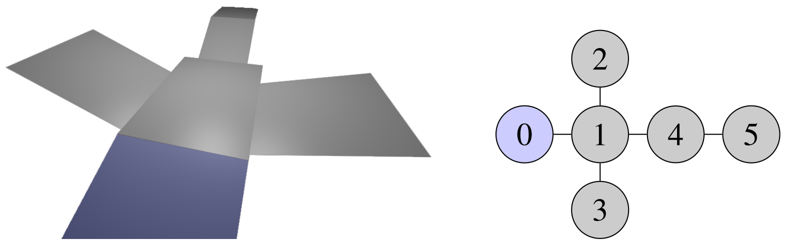
When the localised regions overlap the boundary, the assignment error tends to be high, relative to the rest of the scene. Figure 5.7 illustrates an example scene for which hypothe-



(a)



(b)



(c)

Figure 5.6: Example scenes of varying complexity generated using Blender and their respective plane-graphs used in the hinging process.

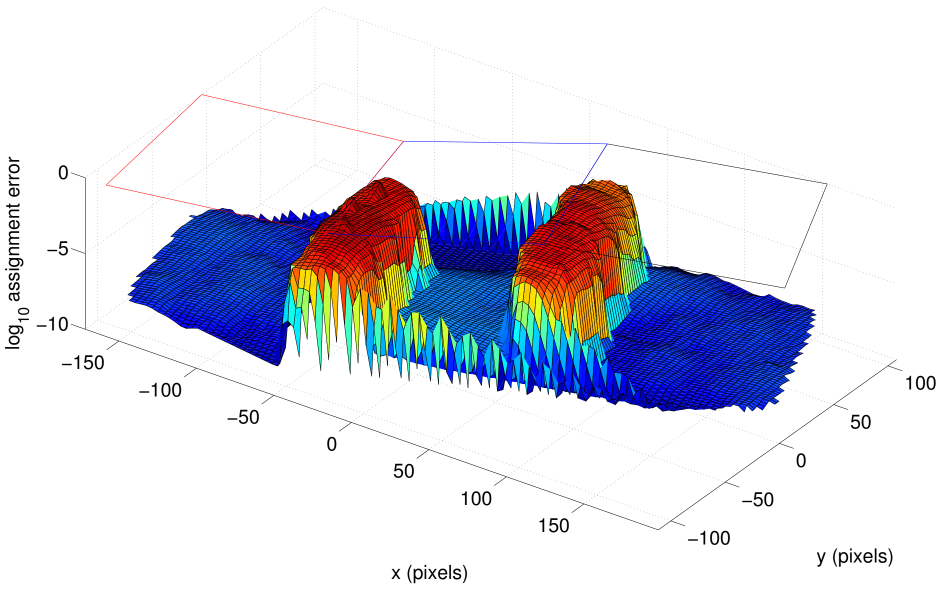
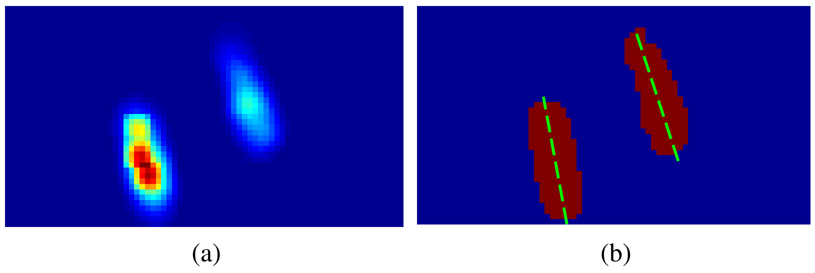


Figure 5.7: The error in regions proximal to the boundary lines between planes is generally noticeably higher than in regions encapsulating only one plane. Here, each pixel in the image is manually assigned the correct plane and the error of that assignment within the region surrounding the pixel is calculated. We observe that the raw assignment error about the boundary lines follows this observation, allowing the rough location of the area containing the boundary region to be established. The actual plane layout in image-space is projected onto the x-y axis.



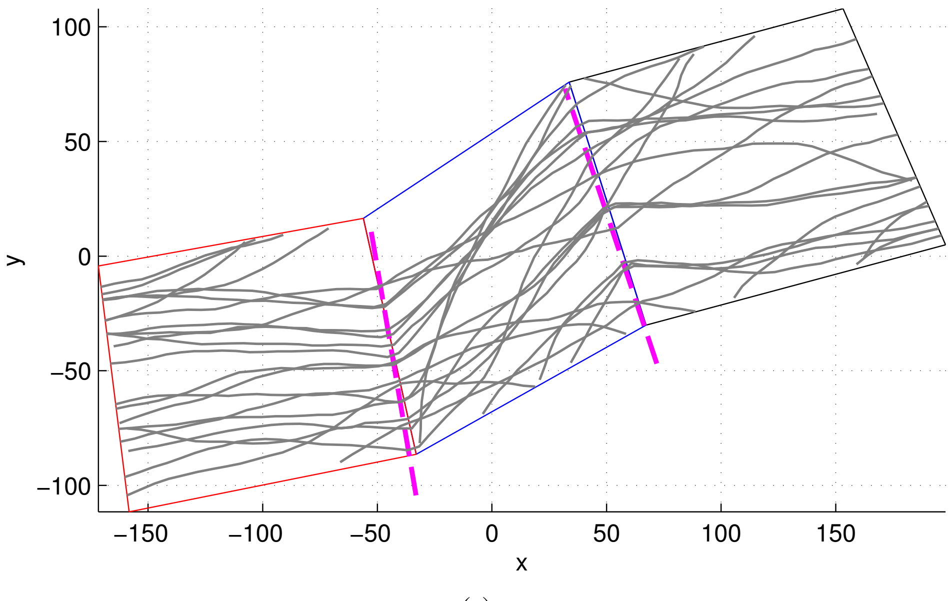
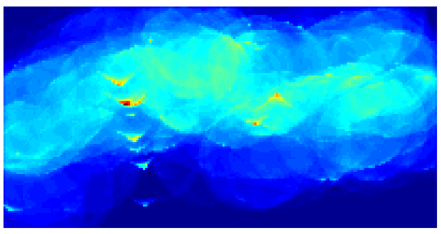
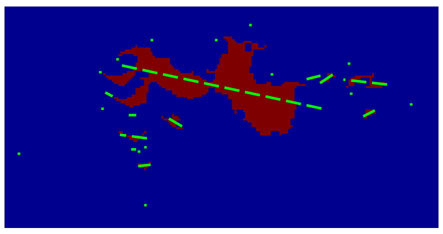
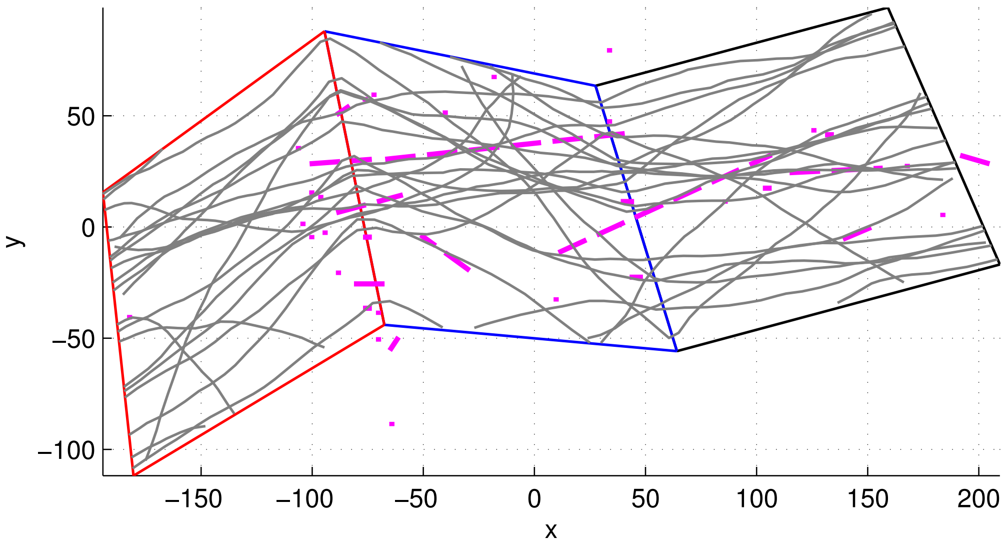


Figure 5.8: At the boundary between planes the local-region assignment error rises due to trajectories within such regions spanning multiple planes as shown in (a). By taking the line through the centroid of the blob parallel with its primary direction as per (b), an estimate for the boundary lines between planes can be automatically obtained without prior knowledge of the number of planes, displayed as dashed magenta lines in (c).


(a)


(b)



(c)

Figure 5.9: As the variation in speed increases, the error landscape becomes decreasingly usable as a means of finding the boundary points as can be seen in the example above, generated with 30% variation. As a result this method is infeasible in real-world scenarios. Whilst connected component analysis could be used to remove small erroneous regions, the clear error peaks observed about the boundary with perfect data are not evident here.

ses have been estimated on perfect data and assigned based on minimising error for each local region (regions containing too little trajectory data are excluded). It should be noted that the error is also high at the non-connected extremities of the centre plane. These regions can be excluded simply by increasing the minimum amount of data required for a region to be considered in calculation. We note that even though the hypotheses are extremely close to the true values, there is still a sharp increase in error at the boundary lines.

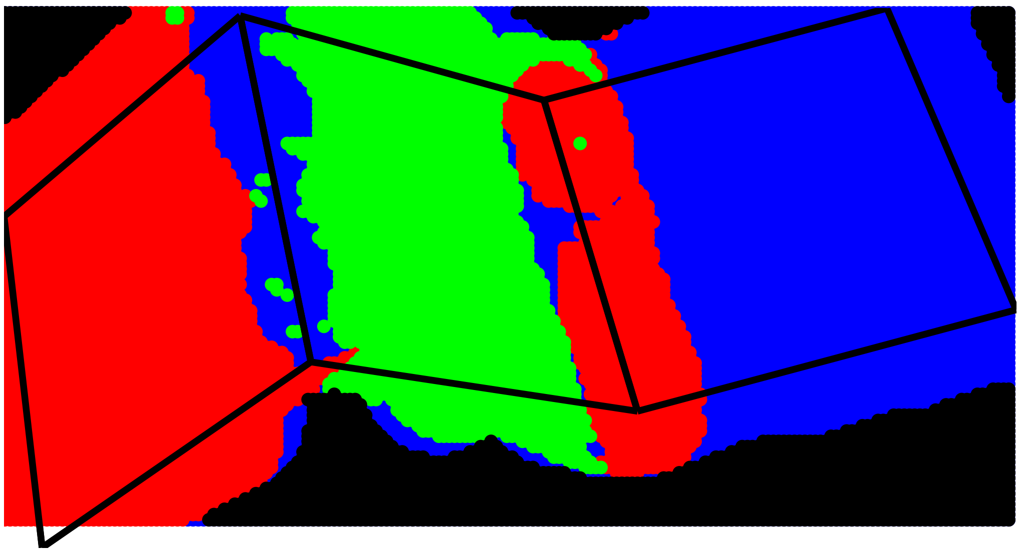
In order to use this error profile regions of higher error must first be segmented– thresholding at the mean error value is sufficient to produce clear blobs about the boundary lines. Given the heat map of error shown in figure 5.8a, thresholding at the mean gives rise to two clear regions around the boundary lines, shown in 5.8b. In order to find the boundary lines from the regions, Principal Component Analysis (PCA) is performed on the blobs to find their principal directions in the image. The boundary line is then taken to be the line passing through the centroid of each blob, parallel to its principal direction. Figure 5.8c gives the result of overlaying these vectors over the true plane configuration. In this case, the estimated boundary lines are extremely close to the true positions and orientations (centroids are ten and nine pixels apart with an orientation difference of two and one degrees respectively compared to true values); however this is to be expected as the input data was free of speed variation.

Upon examining the effect of speed variation on the method it becomes clear that thresholding the error space at the correct level is no longer a trivial matter. By examining the error space in figure 5.9a, the sharp peaks seen at the boundary lines when using perfect input data are no longer present, and instead much of the area covered by the planes is of a somewhat uniform error level. This makes thresholding extremely unreliable at best. We also note that many small regions are detected. Whilst these could be filtered out based on their area in comparison with other detected blobs, the issue regarding the lack of clear boundaries prevents the use of this method in the real-world.

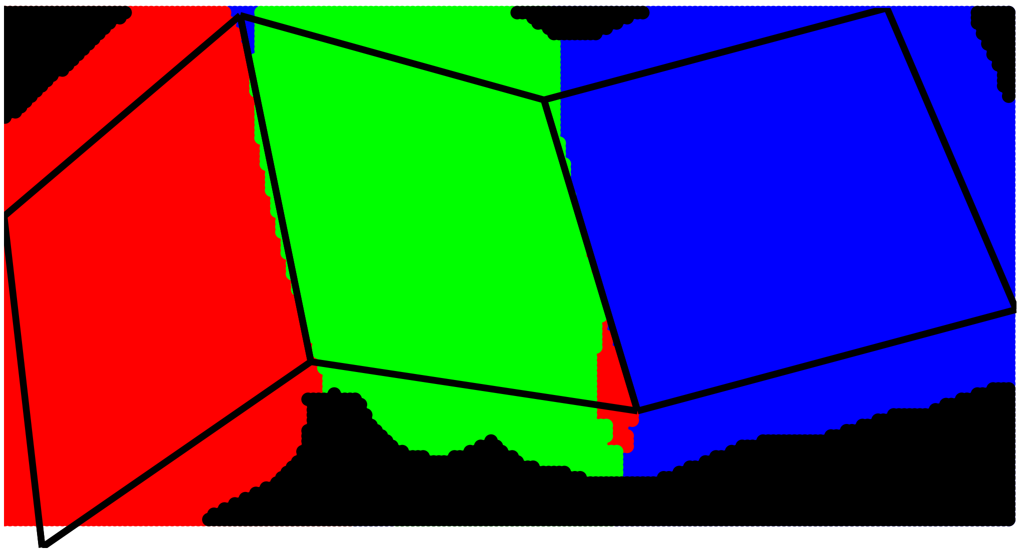
5.2.4.2 Smoothing the Raw Labelling

An initial labelling is produced using the rough hypotheses from the initialisation step of the framework described in the previous chapter, see figure 5.10a. The raw labelling contains many artifacts, particularly around the boundaries, which disrupt attempts to segment the planes.

In order to reduce this occurrence of such labelling errors prior to segmentation, the la-



(a)



(b)

Figure 5.10: The result of minimally assigning labels based on the labelling cost contains many irregularities, particularly around the boundaries of two planes where the window overlaps multiple planes. An example of this on a three-plane scene is shown in figure (a). To minimise these, a kernel is passed across scene. Each element in the kernel represents a pixel in the window surrounding the central point and each pixel has a measure of labelling error assigned to it. This error is used to create the measure of "importance" placed upon the label assigned to that pixel. The label of maximum sum-importance is then chosen.

belling must be smoothed. The “importance” of a label at a given pixel is based on the label assignment error. If the error is low, it is not unreasonable to assume the labelling is reliable as the data fits the hypothesis well. Conversely if the best-fitting label does not represent the observed data well, the error will be high and the labelling is assumed to be incorrect.

Using this definition of importance, smoothing is achieved by passing a sliding window of size $w \times w$ pixels across the scene labels. The aim is to calculate the best label for pixel $\mathbf{u}$ given all pixels in the window surrounding it, $\mathcal{N}_{\mathbf{u}} \equiv \mathbf{u}'_{i,j}, \{i, j\} \in \{1 \dots w\}$. Each pixel $\mathbf{u}'_{i,j}$ in the window has a minimally assigned label and the cost of assigning that label, denoted $e(\mathbf{u}'_{i,j})$.

An “importance matrix” $\mathbf{I_u}$ is then calculated, the same size as the window, in which the error at each pixel is normalised by the maximum error within the window, e_{max} to give a value in the range $0 \dots 1$, which is then subtracted from 1.

$$\mathbf{I_u} = \mathbf{1} - \begin{pmatrix} \frac{e(\mathbf{u}'_{1,1})}{e_{max}} & \cdots & \frac{e(\mathbf{u}'_{1,w})}{e_{max}} \\ \vdots & \ddots & \vdots \\ \frac{e(\mathbf{u}'_{1,w})}{e_{max}} & \cdots & \frac{e(\mathbf{u}'_{w,w})}{e_{max}} \end{pmatrix} \tag{5.1}$$

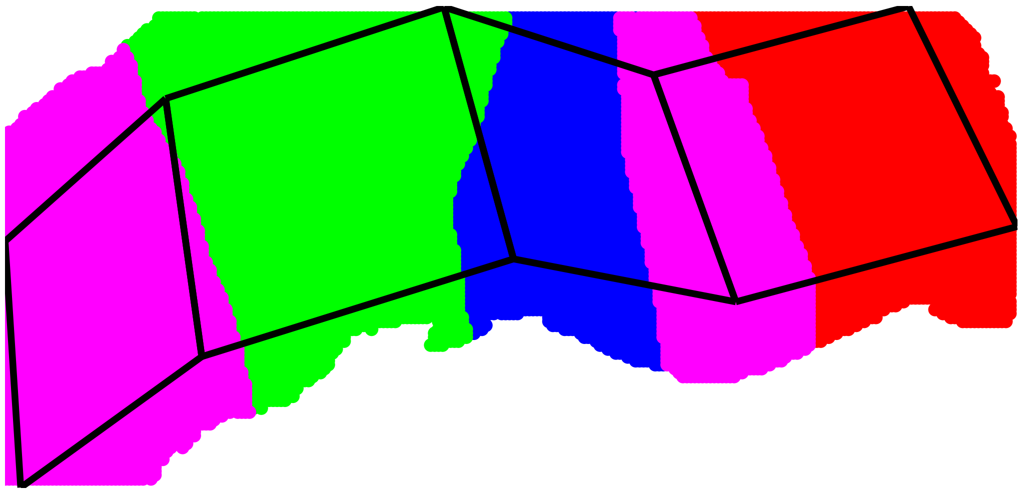
The total importance for each unique label within the window is then calculated as the sum of the importances of all pixels assigned that label. The label with highest importance is chosen for $\mathbf{u}$.

It is not uncommon, particularly in scenes containing planes of similar orientation, for more than one region to be assigned the same label as can be seen in figure 5.11a. In such cases, connected components analysis is used to identify the two distinct regions and a new label is created for each additional region as shown in figure 5.11b.

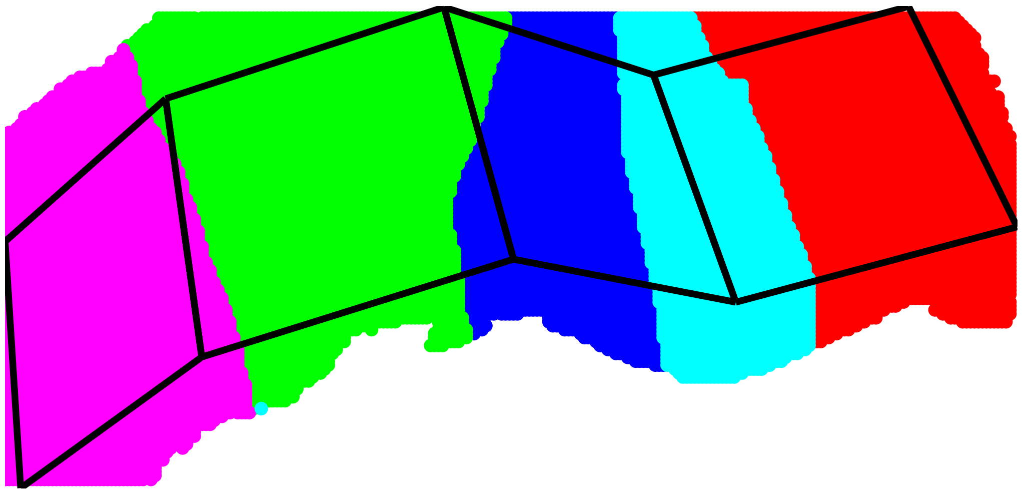
5.2.4.3 Extending the Two-Class SVM Approach

In the previous chapter several methods were introduced, with an SVM-based method best balancing efficiency and accuracy. Support Vector Machines in their original configuration can only separate two classes. The generalisation of SVM’s to train for multiple classes tends to involve one of two approaches:

One Versus All A classifier is trained for each class individually, finding the hyperplane separating it from the rest of the data



(a)



(b)

Figure 5.11: In circumstances where more than one plane of similar orientation exists, or when error is particularly high about the boundary, the same label may be assigned to multiple regions, as seen above. In such cases the additional regions are assigned new labels with the same parameters as the original. Here each colour is a different label. We observe the magenta label has been assigned to two separate regions in (a). After connected components analysis to identify the distinct regions, the second is assigned the a new cyan label in (b). Note that the labelling about the intersection is still incorrect - this will require a further iteration of estimation to compensate for the initial mislabelling.

One Versus One A classifier is trained for each pair of classes such that the hyperplane separating each is found

The argument as to which technique is more suitable under general circumstances is ongoing [104, 91, 60], and both have their merits in different applications. For the use intended here, “One vs All” is the more suitable approach. For each label, a binary segmentation is produced in which regions assigned that label are set as positive examples and all others as negative, as per figure 5.12c. Once a label has been used as the positive example, all regions labelled with it are removed from further calculation, as shown in figure 5.12d. Each SVM offers a boundary line for the plane used as its positive example. This method requires that the ordering is sensibly chosen such that no more than one plane boundary exists for each SVM during training (i.e. planes only have at most one connected plane remaining during training).

Having performed smoothing on the label assignment, the multi-class SVM approach can begin. It is reasonably assumed that a plane region can be found for which only one other hinged-plane is connected to it by along some intersection line (in the example in 5.10b, this could be either the large red or blue regions). There may be cases where adjacent planes are not connected by a hinge, meaning that the reference plane cannot be readily identified automatically; this is a limitation of the system and will be discussed further in chapter 6. In order to determine adjacency of planes, the following procedure is used:

1. For each label, generate a binary image with “on” pixels being those assigned the label and all others “off”.
2. Generate a list of connected components in the binary image.
3. Remove small components (those whose area is below the mean of component area minus two standard deviations) to discount some mislabellings.
4. In each binary image, grow the remaining components by 1 pixel.
5. For every pair of grown binary images, find the overlapping areas - if there is overlap, the regions are adjacent.
6. Pick, at random, any region with only one adjacency – the label assigned to this region is chosen as the positive example in the multi-SVM method.

Using the chosen label as a positive example and all other labels as negative, an SVM is trained on the (x, y) locations of each pixel, giving a boundary line. This label is then discounted and the above procedure is repeated on the remaining labels. By calculating

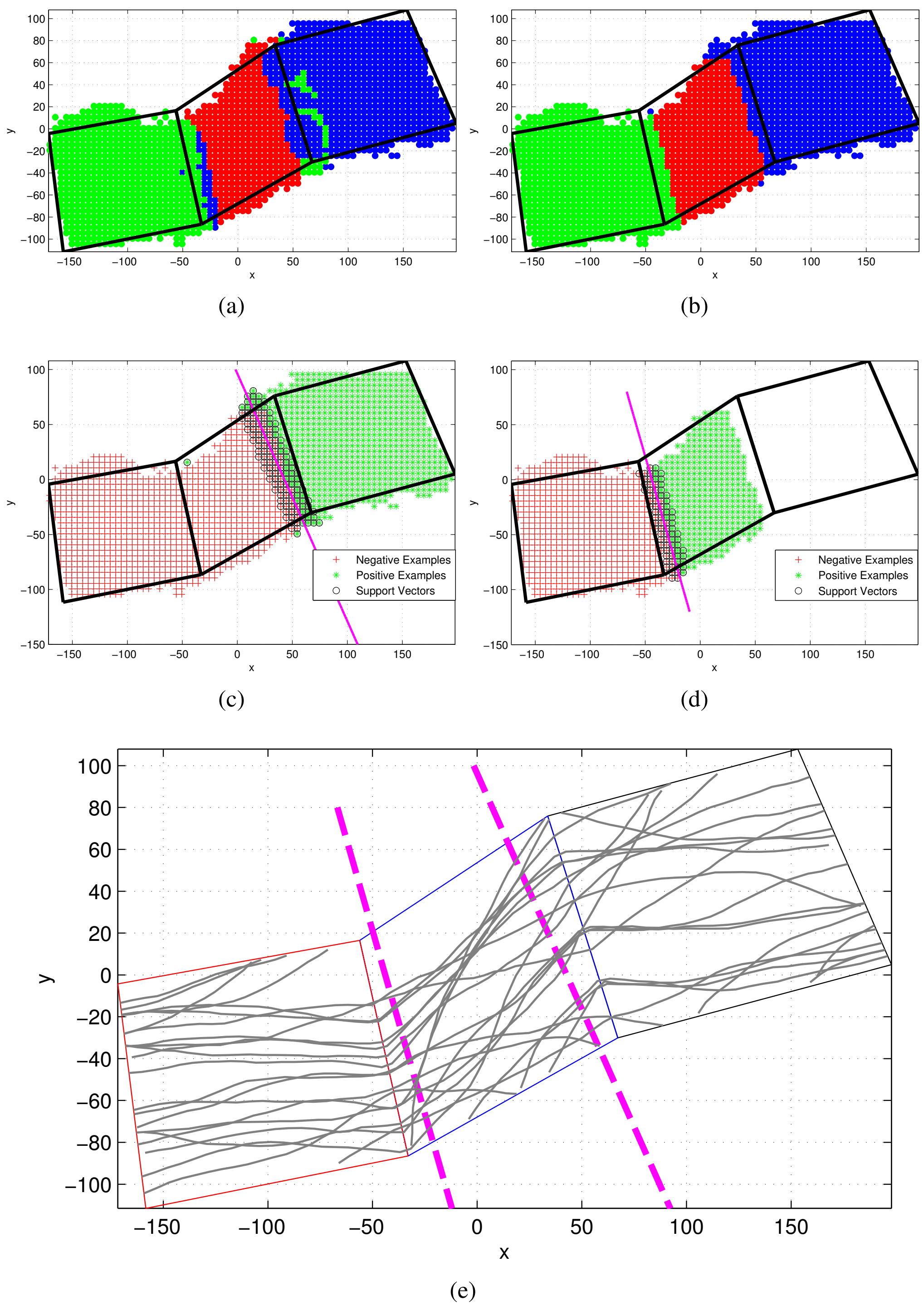


Figure 5.12: A multi-class SVM-based approach is used to obtain the boundary lines between planes in the image. An initial labelling of localised regions, as shown in (a), is generated using the framework described in chapter 4. This contains noisy observations that can skew the SVM classification so the labelling is smoothed using the importance based smoothing function described in section 5.2.4.2, giving the result in (b). SVM's are then trained for each plane in a "One Versus All" manner, with regions removed from further calculation once their SVM has been trained (c)-(d). The final result is a set of boundary lines in the image for each plane (e). The ground-truth plane separation is overlaid on all images as the black polygons in (a) to (d) and as coloured polygons in (e). The input data used here was subject to 10% speed variation.

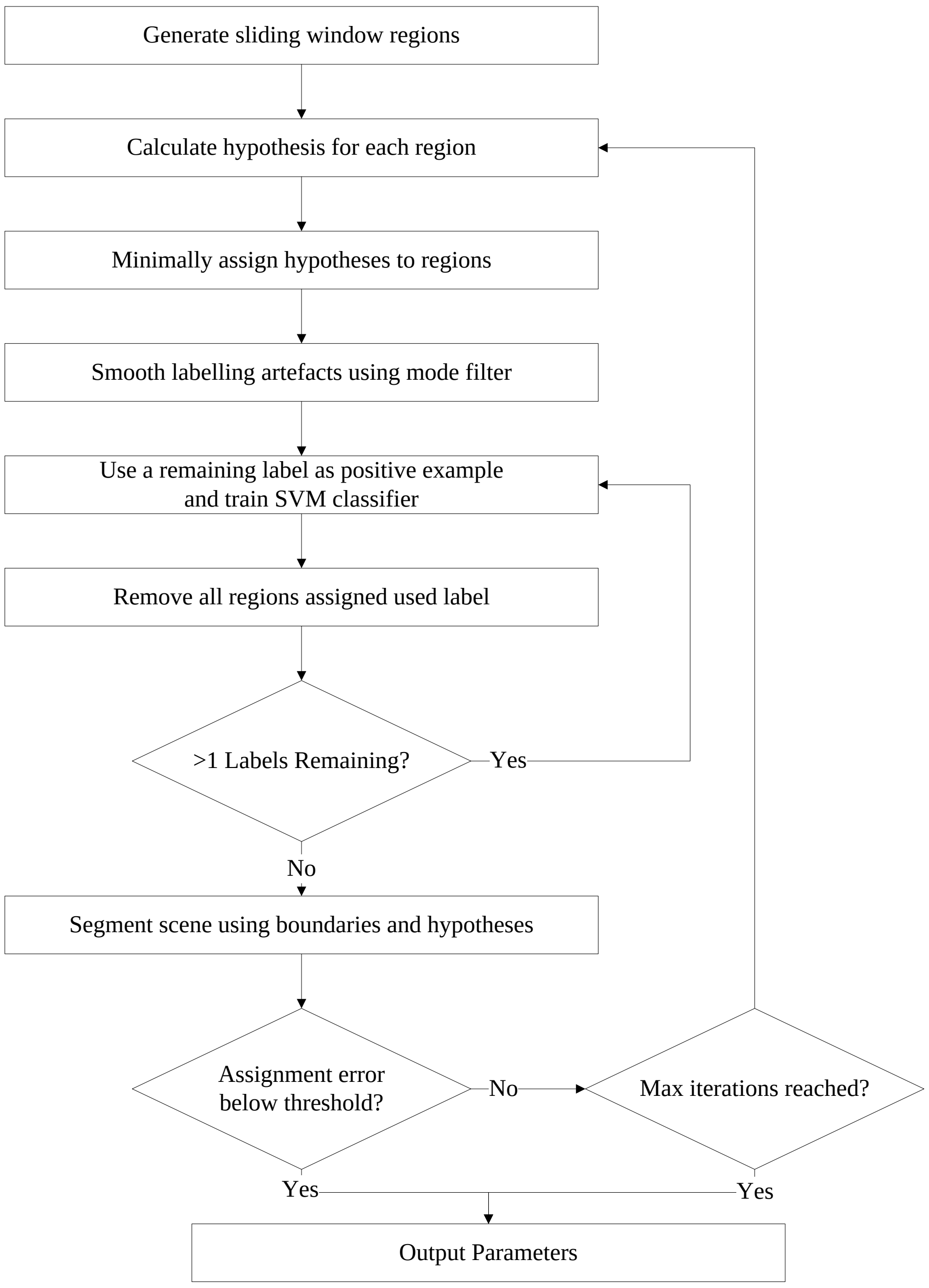

Generate sliding window regions
Calculate hypothesis for each region
Minimally assign hypotheses to regions
Smooth labelling artefacts using mode filter
Use a remaining label as positive example and train SVM classifier
Remove all regions assigned used label
>1 Labels Remaining?
Yes
No
Segment scene using boundaries and hypotheses
Assignment error below threshold?
No
Max iterations reached?
Yes
Yes
Output Parameters

Figure 5.13: The framework outlined in section 4.3 of chapter 4 is generalised to allow for an unlimited number of planes. The method is initialised as in the previous framework; however once raw assignments have been made, the data is smoothed to reduce erroneous assignments near plane boundaries. A labelled region with only adjacent region is chosen as the positive example for the first iteration of a multi-SVM to find a boundary line. All regions assigned this label are then removed, before the next plane is used. This process is repeated until there is only one unprocessed label, by which point an estimate has been found for all boundaries. The scene is then segmented and error estimated. This process can be iterated to improve the plane estimates and boundary line locations.

Table 5.2: Quantitative comparison of true and estimated parameters for a single-plane scene with 10% speed variation, pictured in figure 5.14a.

	θ	ψ	α
Ground Truth	67.0	20.0	0.0014
Estimated	67.7	12.6	0.0017

Table 5.3: Quantitative comparison of the true and estimated parameters obtained when applying the generalised framework to a two-plane scene containing 10% speed variation, pictured in figure 5.14b.

	Plane 1		**Plane 2**		
	θ	ψ	θ	ψ	α
Ground Truth	24.0	8.0	36.2	60.6	0.0014
Estimated	28.3	3.4	36.2	60.5	0.0012

the SVM at all but one labels as positive examples, the boundary lines corresponding to the estimated hypotheses are approximated. The example shown in figure 5.12 shows a typical set of boundary lines computed on a three-plane scene.

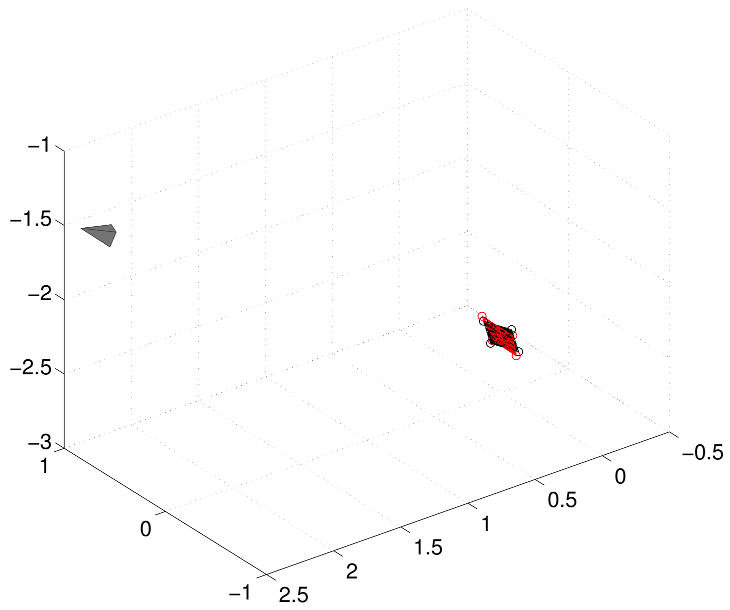
5.3 Results on Simulated Data

The method outlined above has been tested on a range of simulated datasets. In this section, examples of input data of varying complexity will be given as well as numerical analysis of how error rates vary with the complexity of the scene.

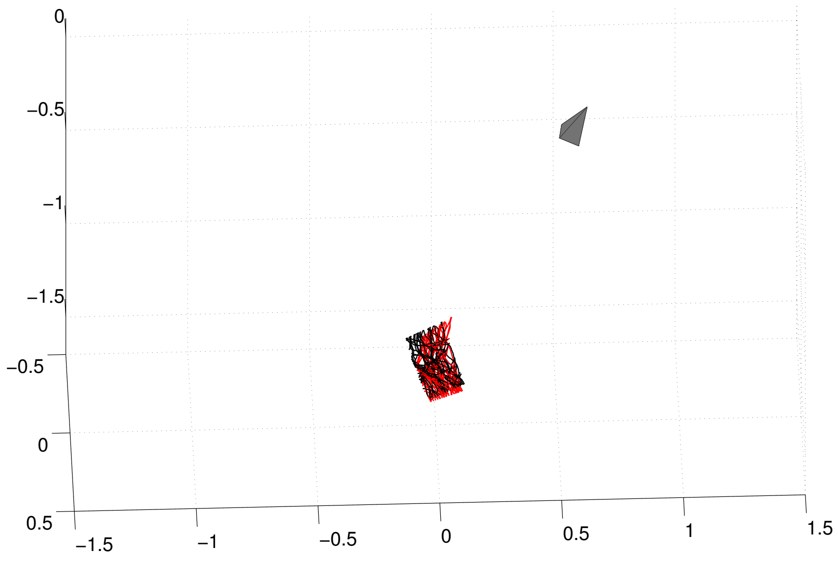
In order to fully assess the generalisation of this method, it must first be shown to work on data already considered by the previous two chapters, namely scenes containing one and two planes. The method correctly identifies a single plane, under 10% speed variation with reasonable accuracy as shown qualitatively in figure 5.14a and quantitatively in table 5.2. When considering scenes containing two planes, again with motion consisting of 10% speed variation, the parameters of the two planes are well approximated and the intersection between them found as shown in figure 5.14b and table 5.3.

The next type of scene considered here contains three planes. An example result on "perfect data" is given in figure 5.15. We see the system is able to identify both the positions of the plane intersections as well as the orientations of the plane itself. Quantitative results for the example given here are provided in table 5.4.

In order to assess the effect of speed variation on a scene of this complexity, a number of

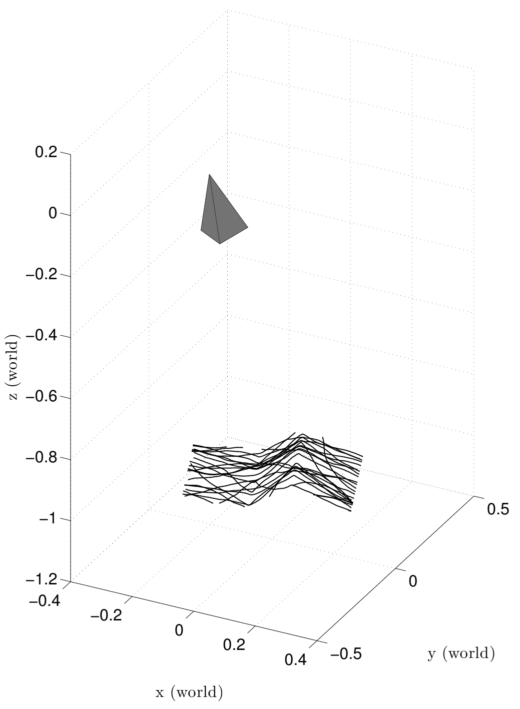


(a)

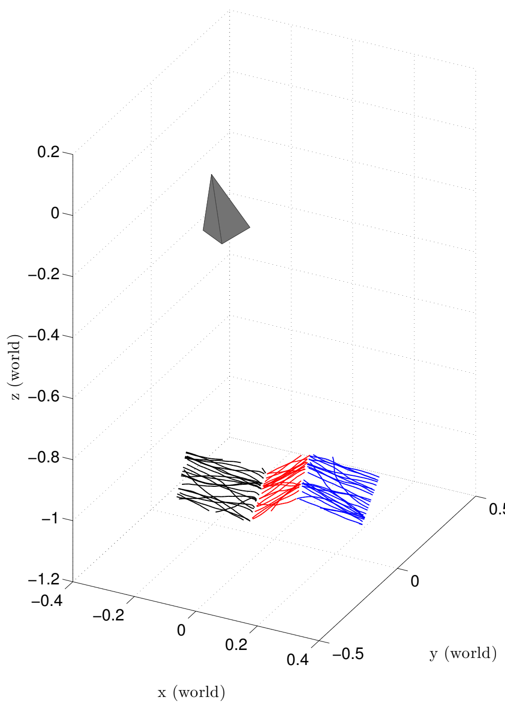


(b)

Figure 5.14: Single and two plane configurations were tested using the generalised estimation algorithm outlined in the latter part of this chapter. In both cases the system correctly identifies the number of planes and produces usable estimations of their parameters based on input trajectories exhibiting 10% variation in speed. The camera is indicated by the grey pyramid.



(a)

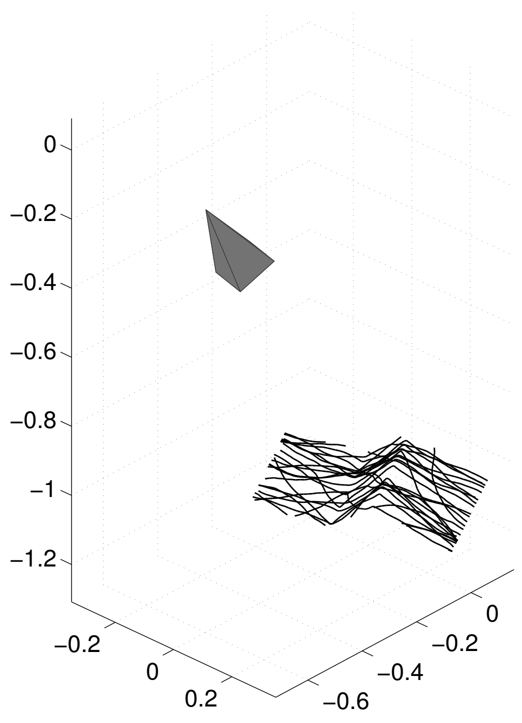


(b)

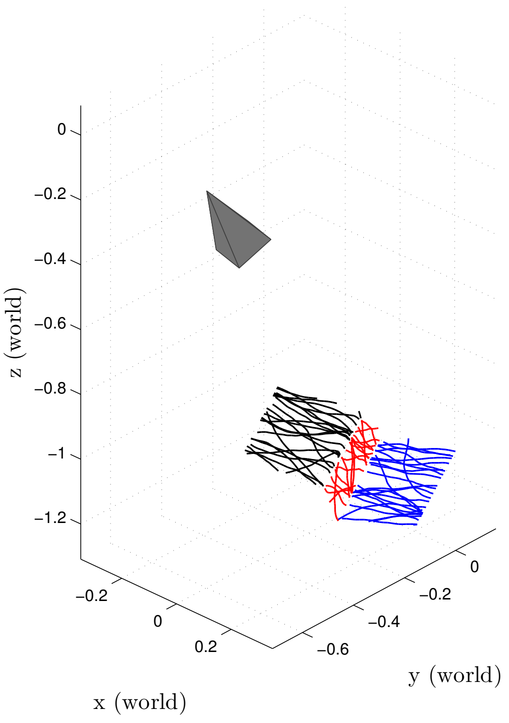
Figure 5.15: In this example of a three-plane world, generated using "perfect" data, the system correctly identifies the topology of the three planes, accurately estimates the intersections between the planes and their orientations. The final result is almost identical to the ground-truth reconstruction. Quantitative evaluation of this example is given in table 5.4.

Table 5.4: Quantitative comparison of true and estimated parameters for a three-plane scene with zero and 10% speed variation, pictured in figures 5.15 and 5.16 respectively.

	Plane 1		**Plane 2**		**Plane 3**		
	θ	ψ	θ	ψ	θ	ψ	α
Ground Truth	30.0	15.0	52.2	-48.4	44.8	69.4	0.0014
0% Speed Variation	27.3	17.2	51.6	-49.2	42.7	70.1	0.0015
10% Speed Variation	30.6	22.1	50.2	-42.5	45.3	71.5	0.0018



(a)



(b)

Figure 5.16: In this example of a three-plane world, generated using input data featuring 10% speed variation, the system again produces a very good approximation of the topology of the three planes and their orientations. Quantitative evaluation of this example is given in table 5.4.

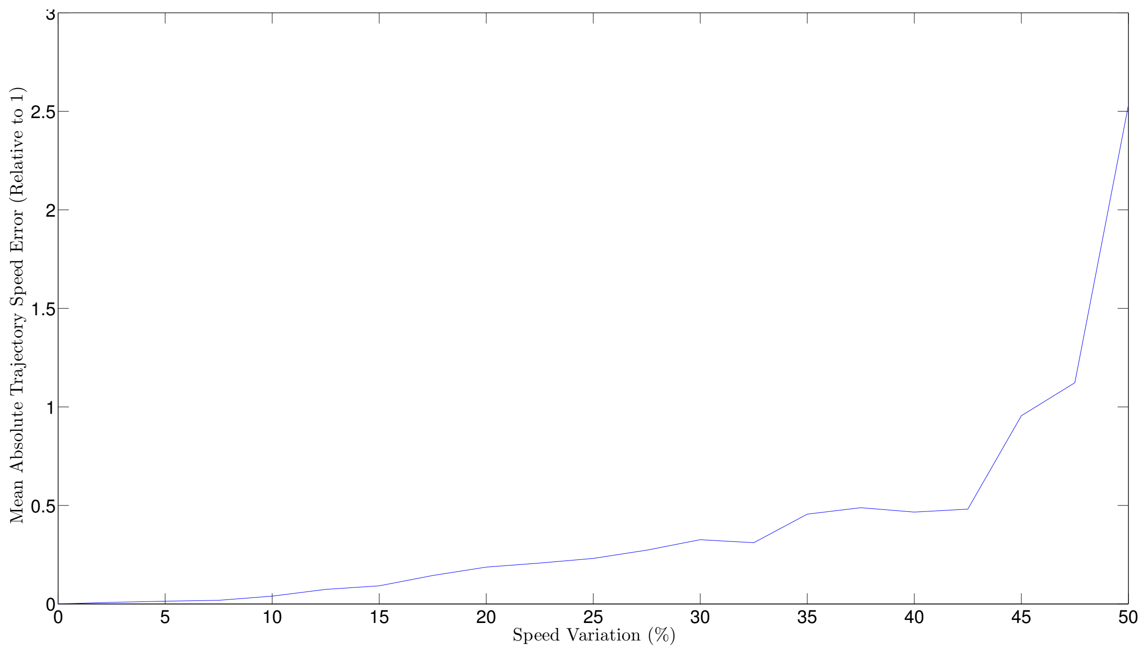
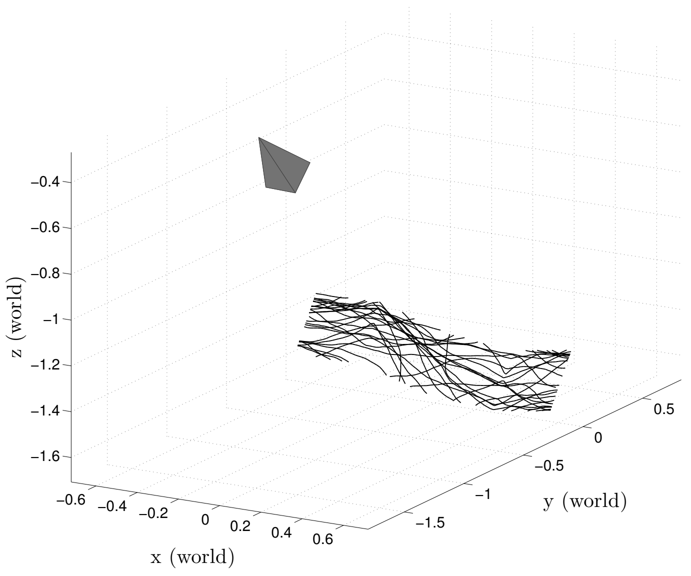


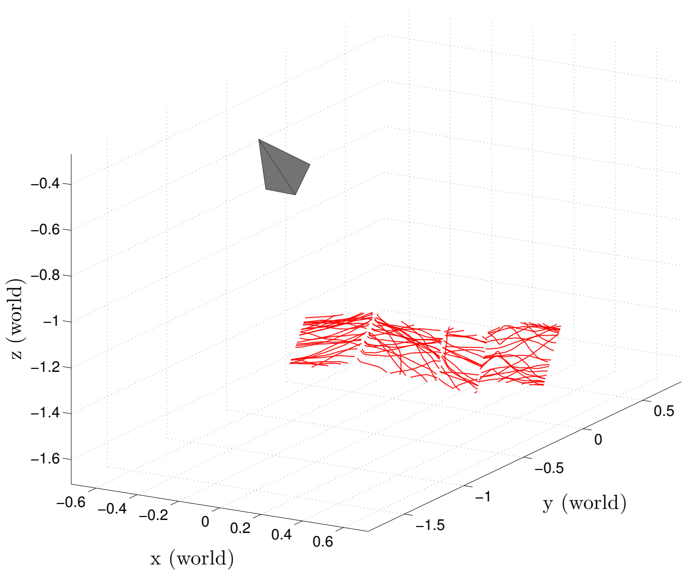
Figure 5.17: The effect of variation in speed in terms of the average trajectory speed error is an important factor in terms of the robustness of the method. Here we see that as variation increases, so the error starts to rise rapidly. However, at lower levels, more likely to observed in the real-world (e.g. 20-25% speed variation), a good reconstruction is still produced.

scenes were generated and trajectories placed upon them with increasing degrees of speed variation. Figure 5.17 offers insight into the increase of error in the orientation parameters as speed variation increases. As the standard variation of intra-trajectory speeds gets to a 0.5, in relation to a mean-speed of 1, the ability of the system to estimate sensible parameters is heavily impeded. However, this is a high level of variation and is not that likely to be observed in the real world. At more realistic levels (10-20%) usable, though not perfect sets of parameters are still obtained. An example of the quality of reconstruction at 10% input speed variation is given in figure 5.16.

An example of a four-plane scene with zero noise variation is given in figure 5.18, with the quantitative comparison given in table 5.5. The result is close to the true reconstruction; however there is more error in the orientations. This is likely to be due to the fact that each plane has less proportional area in the scene, meaning the solution to each is less well conditioned. Despite lower accuracy than the three plane method, the resulting parameters still provide a usable reconstruction of the scene.



(a)



(b)

Figure 5.18: In this example of a four-plane world, generated using "perfect" data, the system correctly identifies the topology of the four planes, estimates the intersections between the planes and their orientations with reasonable accuracy. (a) gives the true reconstruction, whilst (b) shows the scene reconstructed using estimated parameters. Quantitative evaluation of this example is given in table 5.5.

Table 5.5: Quantitative comparison of true and estimated parameters for a four-plane scene with zero speed variation, pictured in figure 5.18.

	Plane 1		Plane 2		Plane 3		Plane 4		
	θ	ψ	θ	ψ	θ	ψ	θ	ψ	α
Ground Truth	45.0	5.0	57.7	55.9	47.2	27.1	46.7	-14.4	0.0014
Estimated	39.2	-10.4	54.3	59.5	37.9	25.4	39.2	-10.4	0.0014

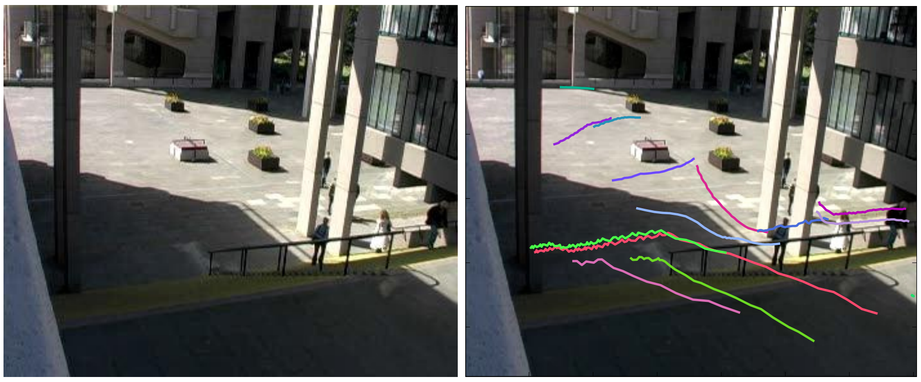


Figure 5.19: An example frame from the UoL steps dataset along with the most representative trajectories (coloured randomly) from the clusters obtained using the method described in chapter 3. The steps themselves are assumed to approximate a single linear plane. The extremely oblique angle of the camera makes this scene a challenge for the system presented here.

5.4 Application to the Real-World

The experiments presented thus-far describe the estimation of multi-planar scenes generated by a controlled simulation method. Of course, this is insufficient to indicate the systems applicability in the real-world, particularly as the simulations used above contained plane regions of approximately equal size, a configuration that cannot be assumed to be present in a real-world scenario.

This section will present results on two multi-planar datasets, UoL Steps and the Kingston Hill Dataset [130]. In both cases the generalised algorithm presented in this chapter was applied to trajectories generated by the KLT tracker and a reconstruction of the scene was generated. Ground-truth orientations were calculated using geometric features of the scene to calculate the orientation of each plane.

The UoL Steps dataset, a frame of which is shown in figure 5.19 has two key issues that may prove problematic for the system presented here:

1. The camera angle is very low, offering minimal view of the steps themselves

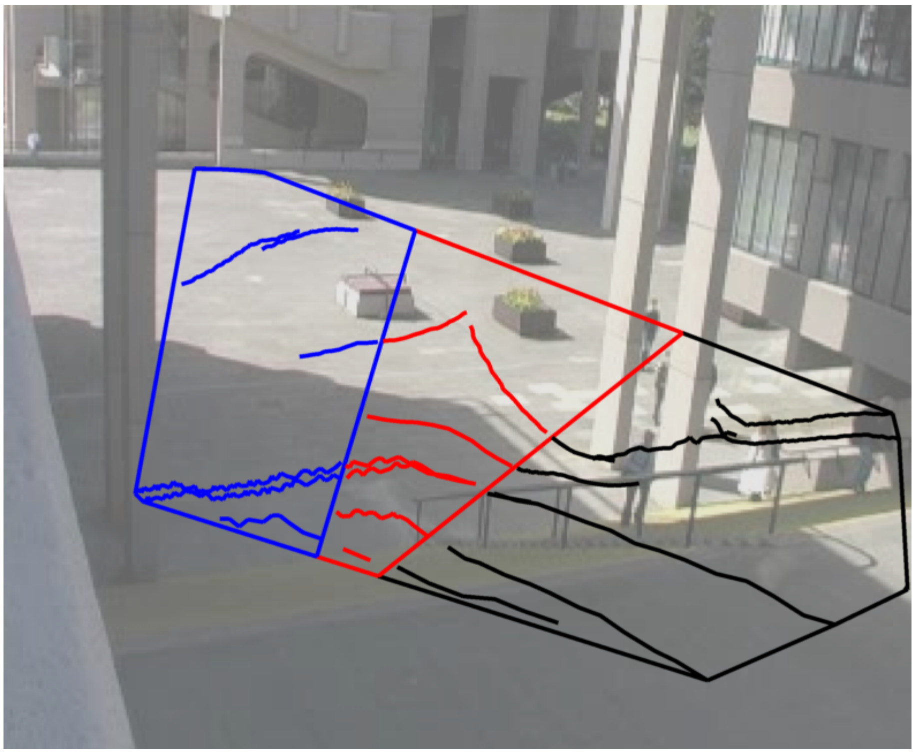


Figure 5.20: The segmentation of the scene into individual planes is given. Whilst the general area of segmentation is roughly correct, it is clear that the system has misidentified the plane boundaries. This is most likely due to the low camera angle, meaning that the variation in tracked feature height was extreme in comparison with the distance of the camera to the plane.

2. The scene contains a number of steps, which for the purposes of this experiment are assumed to roughly approximate a single plane.

The segmentation of the scene is given in figure 5.20, in which the approximation can be seen to roughly represent the true segmentation, although some miscalculation is evident, largely due to the low viewing angle. Despite this, the plane configuration is quite well estimated, with many of the trajectories being well reconstructed using the estimated parameters. Figure 5.21 shows a comparison of the internal speeds of trajectories estimated using the estimated parameters (red) and ground-truth ones (black), with both well and poorly reconstructed trajectories shown. The former are clearly very close to the true result, being those for which the segmentation has been correctly identified, whilst the poor ones are due to those regions of the scene being incorrectly labelled. It is also noticeable that reconstruction quality deteriorates from the top of the steps to the bottom. Here, the top plane was taken as the root node for the hinging mechanism, meaning that errors early in the segmentation propagate at each plane.

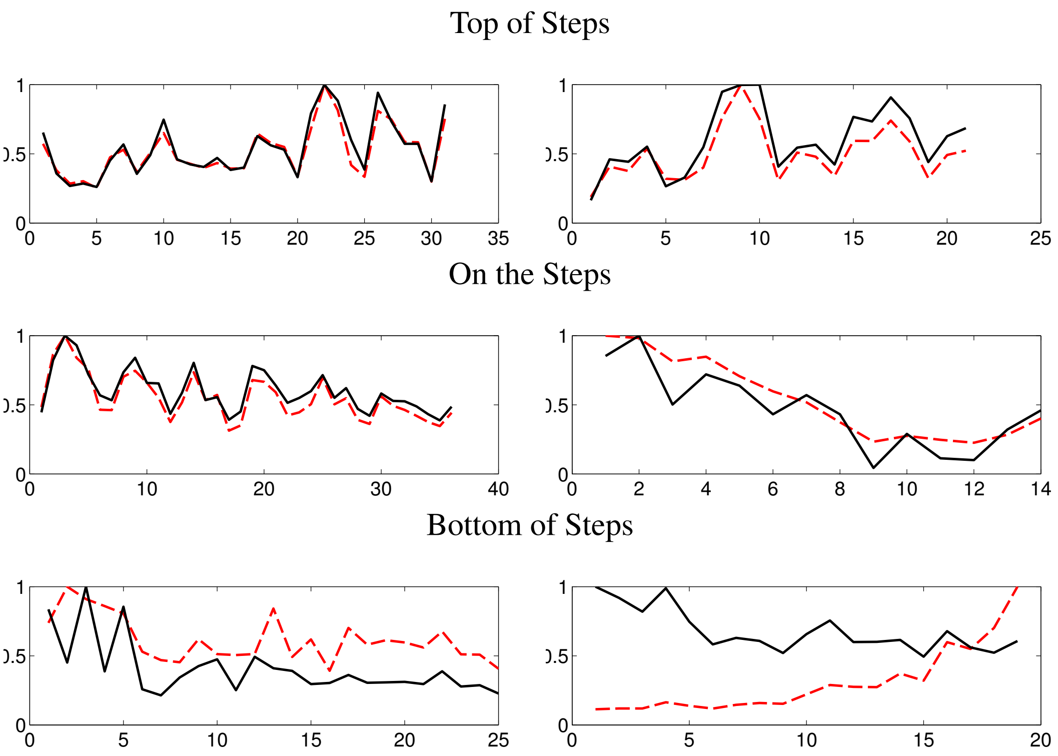


Figure 5.21: Comparison of trajectory speeds rectified using the ground-truth (*black, dashed*) and estimated parameters (*red, solid*). Examples are from the UoL steps dataset. Good quality (left) and poor (right) reconstructed trajectories are highlighted for each plane.

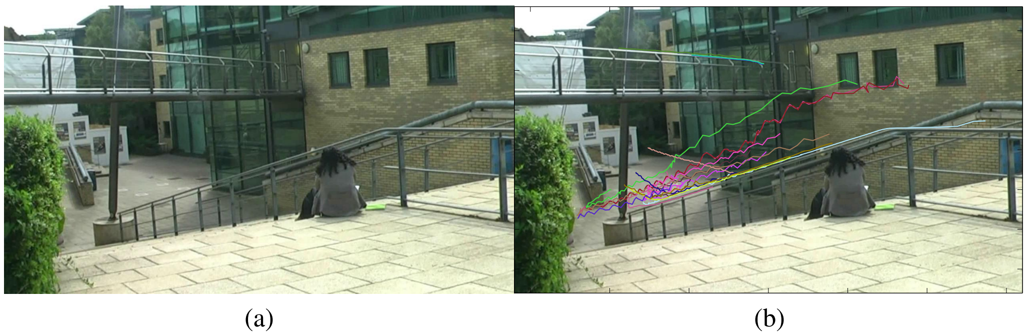


Figure 5.22: An example frame from the Kingston Hill dataset (camera 2) along with the most representative trajectories (coloured randomly) from the clusters obtained using the method described in chapter 3. The steps themselves are again assumed to approximate a single linear plane.

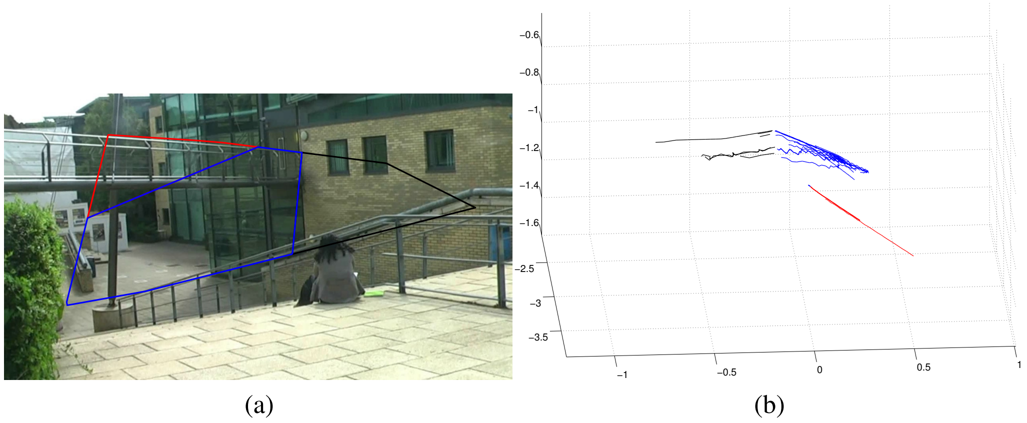


Figure 5.23: Results for camera 2 of the Kingston Hill dataset. (a) shows the segmentation of the scene into individual planes. The algorithm correctly identifies the walkway as a separate plane, although assumes a hinge attachment which is not present. Again, the segmentation at the top of the stairs is skewed by the low position of the camera. (b) gives the reconstructed exemplar trajectories using the estimated parameters. Whilst the top and lower planes have been well approximated, the disconnected walkway plane has not, due to the assumption of a hinged plane.

The Kingston Hill dataset, an example frame of which is shown in figure 5.22, also provides a challenging environment for estimation using this system. Again, the camera height is low, in some cases below the motion evident in the scene. Additionally, motion occurs on disconnected planes, meaning that the assumption of all planes being connected by a hinge is unlikely to apply. Indeed, from the results presented in figure 5.23, we see that although the angle calculated between the two connected planes is calculated reasonably well, the offset plane from the walkway is clearly incorrect. However, the segmentation of the planes shows promise, as the three planes exhibiting motion are all identified fairly well, notwithstanding a similar issue to that observed on the UoL Steps dataset, where the low camera angle relative to the top plane means that identification of regions is somewhat skewed around the top of the stairs.

To compare the rectifying trajectories using the parameters from our method, the homography of each plane to the image was calculated using the well-established vanishing point method. A comparison of rectified trajectory speeds is given in figure 5.24 for the top of the steps and the steps themselves (walkway results were entirely infeasible due to the limitations discussed above). The result from the method presented here is close to the solution calculated using the homographies, showning that on connected planes this framework recreates suitable parameters for rectification.

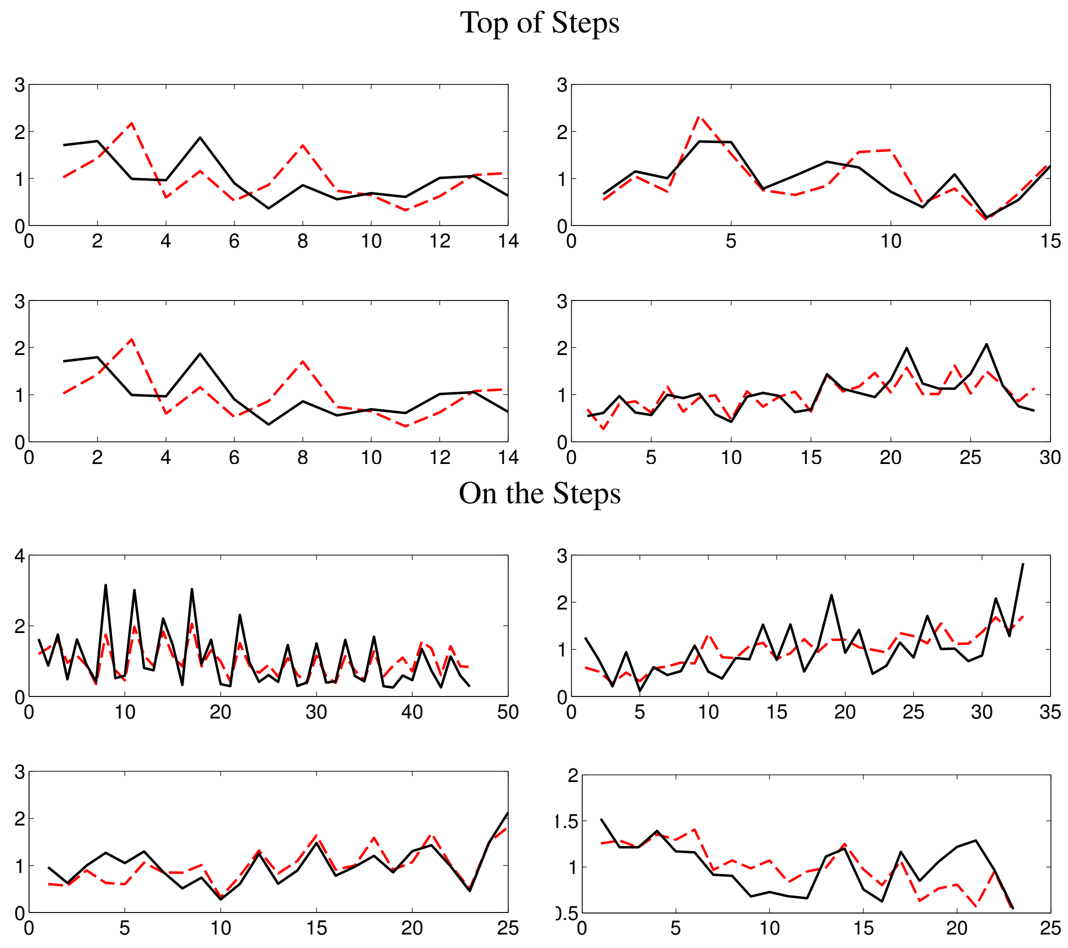


Figure 5.24: Comparison of trajectory speeds rectified using the ground-truth (*black, dashed*) and estimated parameters (*red, solid*). Examples are from the Kingston Hill dataset (camera 2). The system presented here recreates trajectories with speeds closely resembling the true values. Trajectory speeds appear to fluctuate, particularly on the steps. This is mostly likely a combination of the low camera angle meaning that limb articulation is particularly visible and the natural human motion on stair-cases. The walkway is omitted as the estimated plane parameters were entirely incorrect.

5.5 Conclusions

This chapter has sought to efficiently address the plane alignment issues encountered in the previous chapter. By considering the intersection of planes as a hinge, one can not only constrain them to realistic alignments when rectified, but also minimise the parameter-space. In the previous chapter, it was necessary to search for α, two orientation parameters (θ_i, ψ_i) for each plane and the distance from the camera d_i for all but the reference plane – a total of $3N_P$ parameters. With the multi-label SVM method presented in this chapter, the dimensionality of the parameter space has been decreased for all planes but the reference plane. The parameter space now contains $(\alpha, \theta_0, \psi_0)$ and a hinge-angle for each plane, $(\phi_0, \ldots, \phi_{N_P-1})$, giving a parameter space of dimensionality $3 + N_P - 1$.

By reducing the number of degrees of freedom in the system in this way, no flexibility has been lost; however robustness has improved. Because the hinge-rotation parameters of all planes are solved globally, the optimisation step can correct errors during the estimation much more reliably than if an iterative solution were used.

Assuming a linear-chain of planes, the system can obtain near-perfect reconstructions of three-planar scenes at varying levels of speed variation. Estimation on higher numbers of planes is less accurate as there is more influence from the error at plane boundaries and the relative amount of information each planar region offers lessens; however the resulting reconstructions still provide usable rectifications, with the correct topology and layout found, even with speed variation on the input data.

In more complex scenes, such as the Kingston Hill dataset, presented in the previous section the algorithm can identify planar regions with reasonable accuracy, given sufficient trajectory information. The system is reliant on being able to obtain sufficient information about each plane to produce an estimate, therefore in situations such as that shown in the photograph of the shopping centre, figure 5.3, where a relatively narrow plane is present, it is likely that some over-smoothing will occur, potentially missing small planes altogether. If disconnected planes are observed under the hinged plane formulation, the estimation of those planes is extremely poor. As further work, it would be worthwhile including a checking step, whereby if the error when attaching a region as a hinge is too high, that region is re-estimated without the hinging assumption. This is discussed in more detail in chapter 6

The experiments presented here, both on simulations and real-world data have proven the applicability of this framework to real-world multi-planar scenes in situations with relatively simple, although not uncommon, topology. Further experiments are required to fully analyse the usability of the system in more complex situations, with some enhancement needed within the system to allow for disconnected planes. The details of these extensions are discussed in the following chapter.

Chapter 6

Summary and Further Work

6.1 Overview

This thesis has proposed three techniques for reconstructing the world using only information gleaned from trajectories in crowded pedestrian scenes. The first assumes the scene can be modelled as a single plane and three incremental approaches are given along with examination of their success. The second method assumes the world contains two distinct planes and discusses the challenges that arise when the assumption of a single-plane no longer applies. An iterative framework is introduced whereby the coupled problem of solving the plane segmentation and orientation is refined after an initial estimate. The third builds on this by generalising the framework to apply to a piecewise-planar reconstruction of unknown order.

In this chapter, the main contributions of this work are presented in section 6.2. The applicability of the method in the domain is assessed and other possible domains are considered in section 6.3. Finally, section 6.4 will identify improvements and extensions to the techniques outlined herein and possible new research avenues will be discussed.

6.2 Contributions

In this section, the novel contributions in this thesis are discussed in the order they were presented. First, the single-plane estimation method will be recapped along with a brief discussion of its comparison to the similar method of [12]. Next, the two-plane method will be summarised before in the final section, the generalised hinged-plane model is reviewed.

6.2.1 Single-Plane Estimation Framework

This work contributes to the somewhat under-developed area of rectification from speed-information. A novel method of for ground-plane estimation was presented in chapter 3, using only the motion of tracked features to determine a single ground-plane. It was assumed in this technique that each agent moves across it at constant speed, though speeds may differ between agents. First, the mathematical basis was introduced and feasibility studies were conducted on simulated data with the method showing promise for application in the real-world. The majority of existing methods require clear geometric features, strong texture of objects of known size in order to estimate the parameters of the plane. The work presented here shows that using data from short imaged trajectories can provide a ground-plane estimation with sufficient accuracy to rectify the trajectories for perspective distortion. The system was evaluated on a number of video data-sets and was compared to the conceptually similar work in [12]. The method presented in this work achieved its intended aim for the estimation of the ground-plane in single-planar scenes, as evidenced by the superior results obtained on trajectories from a real-world data-set containing medium density crowds when compared to existing techniques.

6.2.2 Modelling a Bi-Planar World

Further enhancement to the field of scene reconstruction is evidenced in the second chapter, in which a method for solving the parameters for two planes is discussed. Introducing more than one plane presents a problem not experienced in a single-plane world – it is necessary to not only solve the parameters of the plane but also the topology of the plane-pair. To this end, an iterative framework was developed that first generates hypotheses for the plane-parameters based on localised estimates from a sliding window, then approximates the boundary line between the planes. The process can then be iterated until the improvement on rectification error drops below a threshold or a maximum number of iterations is reached. The primary contribution of this chapter lies in the novel use of a Support Vector Machine in the 2D image domain to estimate the location of the intersection line of the two planes, taking labels at each pixel as input. Results were given on simulated data showing that detection and reconstruction of the two planes can be performed accurately, even given input data containing speed variation. Again, the system achieved its intended objective for estimation in dual-plane scenes, shown by the levels of accuracy achieved on simulated trajectory data.

6.2.3 Generalised Piecewise-Planar Reconstruction

The final contribution of this work is to generalise the bi-planar estimation method to allow rectification of a piecewise-planar world in which the number of planes may not be known in advance. Without prior knowledge of the number of planes, simple k-means clustering no longer applies. As such a hierarchical clustering algorithm was used in its place. Again, an iterative framework was produced to generate the localised hypotheses before the boundary line is located using a one-vs-all hierarchical Support Vector Machine technique to learn the separation line between the assigned planes. Within this framework it was necessary to perform pre-processing such as filtering the rough labelling before input to the SVM. Promising results are presented on simulated data from a number of scenes exhibiting a range of levels of speed variation as well as two real-world scenes. Whilst the method did achieve reasonable segmentations and roughly accurate reconstructions on real-world data, further work is necessary to enhance the robustness of the method to more complex situations, such as disconnected planes and irregular configurations of planes.

6.3 Applicability of this Work

In this section, the applicability of the techniques in this work are discussed. The primary intention of this work is to allow accurate measurements of metrics such as object size, distance or speed from video taken using a stationary, uncalibrated monocular camera. The particular domain of focus is crowded pedestrian scenes as it is in those that existing methods fail to produce a satisfactory result. Good results were obtained on video sequences with a range of crowd densities, from sparse scenes with only a small number of people meandering within it to a large, dense group moving across the image. The reconstructed trajectories display very close correlation to the ground-truth ones, signifying that the system has indeed captured and reversed the perspective transformation undergone during the imaging process.

The multi-planar methods described in chapters 4 and 5 recreate the simulated scenes closely, showing that the method could be applied to real-world data of similar complexity. Results were presented on real-world datasets, with reasonable accuracy where planes were connected by a hinged intersection line as is assumed in this method. The necessity for such a line does limit the potential applications of this method to some degree; however many pedestrian scenes to exhibit configurations that meet this assumption. As such we believe this work to be a valid technique in many pedestrian scenes and could be incorporated into a density estimation or event detection method to improve the accuracy of observations.

Beyond the pedestrian domain, this work would also be highly applicable in traffic scenes. It is common for a camera to view roadways where traffic will be flowing freely and as such it is likely that vehicles will be moving at a constant speed for their duration in the imaged scene. Therefore the previous methods should apply to trajectories captured from moving vehicles. In scenes where both traffic and pedestrian motion is observed the system may be over-constrained by the similar speeds prior, but this could easily be modified to allow multi-modal distributions of speed, thus providing a further application area of this work.

6.4 Improvements and Further Work

Within this monograph, work has been presented to identify and reverse the perspective transformation undergone by a scene in the world as it is imaged by a camera. Potential extensions and further experiments consider various aspects of the domain and will be summarised within this section.

6.4.1 The Ability to Identify Disconnected Planes

As shown in the experiment on the Kingston Hill dataset, the current system cannot accurately estimate disconnected planes. It would be prudent to incorporate a detection system for such planes whereby disconnected planes can then be estimated as separate entities. Clearly this re-introduces the problem of establishing scale. A potential solution for this would be to use the observed speeds of the tracked features as a scaling measure, selecting values for the camera distance so as to approximately equalise observed speeds across all detected planes.

6.4.2 Allowing Cyclic Planes

The existing system makes the assumption that planes are organised in a tree-like manner, meaning there are no cycles in the adjacency graph. However, real-world scenarios do exist where planes are connected in a cyclic fashion, so including the ability to parse them would increase the applicability of this method in the real world.

Additionally, identification of cyclic planes would allow the system to further constrain the system of parameters through the necessity for loop closure. This is a common concept in the area of Simultaneous Localisation and Mapping [31, 4] and it is likely that inspiration can be gleaned from these methods.

6.4.3 Modelling the Height Difference and Limb Articulation in Tracked Features

Currently this work does not model the fact that tracked features are not observed upon the ground-plane itself, but are in fact almost without exception, tracked above the ground-plane at various heights. Some features may be tracked on the head of individuals, others on shoulders, others on feet. We currently hypothesise, on the basis of experiments on controlled simulations, that the algorithm presented in chapter 3 recovers gracefully from this deficiency, using the relative speed of the trajectory to normalise for the difference in proximity to the camera; however, allowing for variation in tracked feature height is likely to improve the quality of the resulting estimates.

Additionally, feature points tracked on articulate regions, such as limbs in the case of the pedestrian domain, do not move in line with the pedestrian's centre of mass. Rather they produce their own additional displacements relative to it. It is not unlikely that this is a source of error within our real-world estimations so modelling this, perhaps as a Gaussian centred at the pedestrian's centre of mass, may be advantageous. This would, however, rely on identifying individuals, something this work has attempted to avoid due to its complexity in densely crowded scenes.

6.4.4 Improved Models of Pedestrian Motion

The model of crowd motion used throughout this work is naïve, in the sense that it does not take into account human nature, merely working on the ideal of individuals walking at constant speed. Research has been performed into various models for crowd behaviour patters, including the use of the “social force” model first published by Helbing in [57], which places various attractive and repulsive forces on individuals and groups with respect to each other and the scene around them. This has been used with some success to model the behaviour of crowds [89, 113] and could be used here to augment our predictions of how tracked features will move.

6.4.5 Support for Multiple Classes of Moving Object

As was mentioned previously in the discussion regarding the applicability of the work contained in this thesis, the system can currently only support motion observed from a single class of object, such as only pedestrians. However, in the real-world, pedestrians often walk on pathways next to roads, upon which cars are likely to be moving. By enhancing the prior used to prevent infeasible reconstructions, a multi-modal approach could be implemented. This could further be extended to classify the moving objects themselves.

6.4.6 Applying the Method to Density Estimation Normalisation

The techniques in this work were developed with the aim of allowing measurements of object shape and speed to be determined more accurately than they can be from an unrectified image. By applying these techniques to normalise density estimation methods, the true utility of the work can be observed.

6.5 Concluding Statements

This thesis introduced techniques to obtain rectification of imaged data through the use of pedestrian trajectories, using only their speed to determine the parameters of the plane or planes upon which the pedestrians have moved. The single-plane method has been evaluated on both simulated and real-world data and shown to outperform the most conceptually similar algorithm from the literature. The multi-plane estimation framework has been analysed on realistic simulations to assess its accuracy and robustness to variations in world trajectory speed. Scenes containing two and three planes in relatively simple configurations can be reconstructed with high accuracy and whilst approximations of scenes containing four planes are less accurate, they still provide an informative approximation of the topology and orientation of the planes held therein. As discussed above, the system can roughly segment multi-planar scenes with acceptable accuracy; however the more complex configurations present in real-world scenarios diminish the ability of the system to produce accurate orientation estimates, particularly in the case of disconnected planes. Despite this, we believe the work presented here comprises a novel contribution to the field of image rectification and reconstruction providing a suitable framework for many real-world applications.

Bibliography

[1] Albiol, A., Silla, M. J., Albiol, A., and Mossi, J. M. (2009). Video analysis using corner motion statistics. In *Performance Evaluation of Tracking and Surveillance workshop at CVPR 2009*, pages 31–37. Miami, Florida.

[2] Andersen, K. (2006). *The Geometry of an Art: The History of the Mathematical Theory of Perspective from Alberti to Monge (Sources and Studies in the History of Mathematics and Physical Sciences)*. Springer, 2007 edition.

[3] Andriyenko, A. and Schindler, K. (2011). Multi-target tracking by continuous energy minimization. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1265–1272. IEEE.

[4] Aulinas, J., Petillot, Y., Salvi, J., and Lladó, X. (2008). The SLAM Problem: A Survey. In *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, pages 363–371. IOS Press, Amsterdam, The Netherlands, The Netherlands.

[5] Baillard, C. and Zisserman, A. (1999). Automatic reconstruction of piecewise planar models from multiple views. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, page 565 Vol. 2.

[6] Baker, S., Datta, A., and Kanade, T. (2006). Parameterizing homographies. Technical Report CMU-RI-TR-06-11, Robotics Institute, Pittsburgh, PA.

[7] Basharat, A., Gritai, A., and Shah, M. (2008). Learning object motion patterns for anomaly detection and improved object detection. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.

[8] Bay, H., Tuytelaars, T., and Gool, L. (2006). SURF: Speeded Up Robust Features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision ECCV 2006*,

volume 3951 of *Lecture Notes in Computer Science*, chapter 32, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg.

[9] Beardsley, P., Torr, P., and Zisserman, A. (1996). 3D model acquisition from extended image sequences. In Buxton, B. and Cipolla, R., editors, *Computer Vision ECCV 1996*, volume 1065 of *Lecture Notes in Computer Science*, chapter 59, pages 683–695. Springer Berlin Heidelberg, Berlin, Heidelberg.

[10] Benabbas, Y., Ihaddadene, N., and Djeraba, C. (2009). Global analysis of motion vectors for event detection in crowd scenes. In *Performance Evaluation of Tracking and Surveillance workshop at CVPR 2009*, pages 109–116. Miami, Florida.

[11] Boghossian, B. A. and Velastin, S. A. (2002). Motion-based machine vision techniques for the management of large crowds. *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, 2:pages 961–964 vol.2.

[12] Bose, B. and Grimson, E. (2004). Ground plane rectification by tracking moving objects. In *IEEE International Workshop on Visual Surveillance and PETS*.

[13] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 (11):pages 1222–1239.

[14] Brown, L. M. (2004). View independent vehicle/person classification. In *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance &amp; sensor networks*, pages 114–123. ACM, New York, NY, USA.

[15] Capel, D. and Zisserman, A. (1998). Automated mosaicing with super-resolution zoom. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 885–891. IEEE.

[16] Caprile, B. and Torre, V. (1990). Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4 (2):pages 127–139.

[17] Chan, A. B., Morrow, M., and Vasconcelos, N. (2009). Analysis of crowded scenes using holistic properties. In *Performance Evaluation of Tracking and Surveillance workshop at CVPR 2009*, pages 101–108. Miami, Florida.

[18] Chen, L., Feris, R., Zhai, Y., Brown, L., and Hampapur, A. (2008). An Integrated System for Moving Object Classification in Surveillance Videos. In *Advanced*

Video and Signal Based Surveillance, 2008. AVSS '08. IEEE Fifth International Conference on, pages 52–59. IEEE.

[19] Chen, Q., Wu, H., and Wada, T. (2004). Camera calibration with two arbitrary coplanar circles. In *Proc. European Conf. Computer Vision*, pages 521–532.

[20] Cherian, A., Morellas, V., and Papanikolopoulos, N. (2009). Accurate 3D ground plane estimation from a single image. In *2009 IEEE International Conference on Robotics and Automation*, pages 2243–2249. IEEE.

[21] Collins, R. T. and Beveridge, J. R. (1993). Matching perspective views of coplanar structures using projective unwarping and similarity matching. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pages 240–245. IEEE.

[22] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20 (3):pages 273–297.

[23] Coughlan, J. M. and Yuille, A. L. (1999). Manhattan world: Compass direction from a single image by bayesian inference. In *In Internation Conference on Computer Vision*, pages 941–947.

[24] Criminisi, A. (1999). A plane measuring device. *Image and Vision Computing*, 17 (8):pages 625–634.

[25] Criminisi, A. (2001). *Accurate Visual Metrology from Single and Multiple Uncalibrated Images*. Springer London, London.

[26] Criminisi, A., Reid, I. D., and Zisserman, A. (2000). Single view metrology. *International Journal of Computer Vision*, 40 (2):pages 123–148.

[27] Dee, H., Fraile, R., Hogg, D., and Cohn, A. (2008). Modelling scenes using the activity within them. In *Spatial Cognition VI. Learning, Reasoning, and Talking about Space*, pages 394–408.

[28] Dee, H. M. and Caplier, A. (2010). Crowd behaviour analysis using histograms of motion direction. In *IEEE International Conference on Image Processing (ICIP) 2010*, pages 1545–1548.

[29] Dee, H. M. and Santos, P. E. (2011). The Perception and Content of Cast Shadows: An Interdisciplinary Review. *Spatial Cognition & Computation*, 11 (3):pages 226–253.

[30] Delage, E., Lee, H., and Ng, A. (2007). Automatic Single-Image 3d Reconstructions of Indoor Manhattan World Scenes. In Thrun, S., Brooks, R., and Durrant-Whyte, H., editors, *Robotics Research*, volume 28 of *Springer Tracts in Advanced Robotics*, chapter 28, pages 305–321. Springer Berlin Heidelberg, Berlin, Heidelberg.

[31] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part I. *Robotics & Automation Magazine, IEEE*, 13 (2):pages 99–110.

[32] Ellis, T., Abbood, A., and Brillault, B. (1991). Ellipse Detection and Matching with Uncertainty. In *Procedings of the British Machine Vision Conference 1991*, pages 18.1–18.9. Springer-Verlag London Limited.

[33] Evans, M. and Ferryman, J. (2010). Surveillance Camera Calibration from Observations of a Pedestrian. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 64–71. IEEE.

[34] Faugeras, O. (1993). *Three-Dimensional Computer Vision (Artificial Intelligence)*. The MIT Press.

[35] Faugeras, O., Robert, L., Laveau, S., Csurka, G., Zeller, C., Gauclin, C., and Zoghlami, I. (1998). 3-D Reconstruction of Urban Scenes from Image Sequences. *Computer Vision and Image Understanding*, 69 (3):pages 292–309.

[36] Feng, Y. and Hamerly, G. (2007). G.: PG-means: learning the number of clusters in data. In *Advances in Neural Information Processing Systems 19*, pages 393–400.

[37] Ferryman, J. (2009). Pets 2009 benchmark data. Accessed: 2014-30-01.

[38] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24 (6):pages 381–395.

[39] Fitzgibbon, A. and Zisserman, A. (1998). Automatic camera recovery for closed or open image sequences. In Burkhardt, H. and Neumann, B., editors, *Computer Vision ECCV'98*, volume 1406 of *Lecture Notes in Computer Science*, chapter 20, pages 311–326. Springer Berlin Heidelberg, Berlin/Heidelberg.

[40] Flint, A., Murray, D., and Reid, I. (2011). Manhattan scene understanding using monocular, stereo, and 3D features. In *2011 International Conference on Computer Vision*, pages 2228–2235. IEEE.

[41] Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315:pages 972–976.

[42] GÃrding, J. (1992). Shape From Texture for Smooth Curved Surfaces in Perspective Projection. In *Journal of Mathematical Imaging and Vision*, pages 630–638.

[43] Ge, W. and Collins, R. T. (2009). Evaluation of sampling-based pedestrian detection for crowd counting. In *Performance Evaluation of Tracking and Surveillance (PETS-Winter)*, pages 1–7.

[44] Gibson, J. J. (1950). The perception of visual surfaces. *The American journal of psychology*, 63 (3):pages 367–384.

[45] Grammatikopoulos, L., Karras, G., and Petsa, E. (2002). Geometric Information From Single Uncalibrated Images of Roads. *International Archives of Photogrammetry & Remote Sensing*, 34 (5):pages 21–26.

[46] Grammatikopoulos, L., Karras, G., and Petsa, E. (2007). An automatic approach for camera calibration from vanishing points. In *ISPRS Journal of Photogrammetry and Remote Sensing*, pages 64–76.

[47] Guo, F. (2006). Plane Rectification Using a Circle and Points from a Single View. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 9–12. IEEE.

[48] Guo, F. and Chellappa, R. (2006). Video mensuration using a stationary camera. In Leonardis, A., Bischof, H., and Pinz, A., editors, *ECCV (3)*, volume 3953 of *Lecture Notes in Computer Science*, pages 164–176. Springer.

[49] Guo, F. and Chellappa, R. (2010). Video Metrology Using a Single Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (7):pages 1329–1335.

[50] Hamerly, G. and Elkan, C. (2003). Learning the K in K-Means. In *In Neural Information Processing Systems*, volume 17, pages 281–288.

[51] Han, M. and Kanade, T. (2000). Reconstruction of a scene with multiple linearly moving objects. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 542–549 vol.2. IEEE.

[52] Hartley, R. and Sturm, P. (1995). Triangulation. In Hlaváč, V. and Šára, R., editors, *Computer Analysis of Images and Patterns*, volume 970 of *Lecture Notes in Computer Science*, pages 190–197. Springer Berlin Heidelberg.

[53] Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition.

[54] Hartley, R. I. (1992). Estimation of relative camera positions for uncalibrated cameras. In *Computer Vision ECCV 1992*, volume 588, pages 579–587.

[55] Hartley, R. I. (1997). In Defense of the Eight-Point Algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19 (6):pages 580–593.

[56] He, Q. and Chu, C.-H. H. (2007). Lane detection and tracking through affine rectification. In *MVA*, pages 536–539.

[57] Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 51 (5):pages 4282–4286.

[58] Hoiem, D., Efros, A., and Hebert, M. (2007). Recovering Surface Layout from an Image. *Int. J. Comput. Vision*, 75 (1):pages 151–172.

[59] Horn, B. K. (1986). *Robot Vision*. McGraw-Hill Higher Education, 1st edition.

[60] Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13 (2):pages 415–425.

[61] Hurt, N. E. (1991). Mathematical methods in shape-from-shading: A review of recent results. *Acta Applicandae Mathematica*, 23 (2):pages 163–188.

[62] Iiyoshi, T. and Mitsuhashi, W. (2008). Homography-based image mosaicing for automatically removing partial foreground objects. In *Signal Processing and Communication Systems, 2008. ICSPCS 2008. 2nd International Conference on*, pages 1–9. IEEE.

[63] Jau, J. Y. and Chin, R. T. (1990). Shape from texture using the Wigner distribution. *Computer Vision, Graphics, and Image Processing*, 52 (2):pages 248–263.

[64] Junejo, I. and Foroosh, H. (2006). Robust Auto-Calibration from Pedestrians. In *Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on*, page 92. IEEE.

[65] Junejo, I. N. and Foroosh, H. (2007). Trajectory Rectification and Path Modeling for Video Surveillance. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE.

[66] Junejo, I. N. and Foroosh, H. (2008). Euclidean Path Modeling for Video Surveillance. *Image Vision Comput.*, 26 (4):pages 512–528.

[67] Keren, S., Shimshoni, I., and Tal, A. (2002). Placing Three-dimensional Models in an Uncalibrated Single Image of an Architectural Scene. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '02, pages 186–193. ACM, New York, NY, USA.

[68] Krahnstoever, N. O. and Mendonca, P. R. S. (2006). Autocalibration from Tracks of Walking People. In *BMVC06*, pages 107–116.

[69] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics*, 2 (1-2):pages 83–97.

[70] Lai, P.-L. and Yilmaz, A. (2009). A new approach for vanishing line estimation. *ASPRS Annual Conference, Baltimore, Maryland*, 2:pages 472–477.

[71] Lee, D. C., Hebert, M., and Kanade, T. (2009). Geometric reasoning for single image structure recovery. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2136–2143. IEEE.

[72] Lee, J. J. and Kim, G. (2007). Robust estimation of camera homography using fuzzy RANSAC. In *Proceedings of the 2007 international conference on Computational science and its applications - Volume Part I*, ICCSA'07, pages 992–1002. Springer-Verlag, Berlin, Heidelberg.

[73] Lefler, M., Hel-Or, H., and Hel-Or, Y. (2013). Metric plane rectification using symmetric vanishing points. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 300–304. IEEE.

[74] Lerner, Alon, Chrysanthou, Yiorgos, Lischinski, and Dani (2007). Crowds by Example. *Computer Graphics Forum*, 26 (3):pages 655–664.

[75] Li, X., Liu, Y., Wang, Y., and Yan, D. (2005). Computing homography with RANSAC algorithm: a novel method of registration. In *Electronic Imaging and Multimedia Technology IV*, volume 5637 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 109–112.

[76] Liebowitz, D. and Zisserman, A. (1998). Metric rectification for perspective images of planes. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pages 482–488. IEEE Comput. Soc.

[77] Liebowitz, D. and Zisserman, A. (1999). Combining scene and auto-calibration constraints. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 293–300 vol.1. IEEE.

[78] Lin, S.-F., Chen, J.-Y., and Chao, H.-X. (2001). Estimation of number of people in crowded scenes using perspective transformation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 31 (6):pages 645–654.

[79] Lindeberg, T. and Garding, J. (1993). Shape from texture from a multi-scale perspective. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 683–691. IEEE.

[80] Lourakis, M. I. A. (2009). Plane metric rectification from a single view of multiple coplanar circles. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 509–512. IEEE.

[81] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679.

[82] Lv, F., Zhao, T., and Nevatia, R. (2002). Self-Calibration of a Camera from Video of a Walking Human. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 562–567. IEEE Computer Society, Los Alamitos, CA, USA.

[83] Maduro, C., Batista, K., and Batista, J. (2009). Estimating Vehicle Velocity Using Image Profiles on Rectified Images. In Araujo, H., Mendonça, A., Pinho, A., and Torres, M., editors, *Pattern Recognition and Image Analysis*, volume 5524 of *Lecture Notes in Computer Science*, pages 64–71. Springer Berlin / Heidelberg.

[84] Magee, D. (2004). Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22 (2):pages 143–155.

[85] Malik, J. and Rosenholtz, R. (1997). Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 23 (2):pages 149–168.

[86] Marana, A. N., Velastin, S. A., Costa, L. F., and Lotufo, R. A. (1998). Automatic estimation of crowd density using texture. *Safety Science*, pages 165–175.

[87] Maybank, S. and Faugeras, O. (1992). A theory of self-calibration of a moving camera. *Int. J. Comput. Vision*, 8 (2):pages 123–151.

[88] McCahill, M. and Norris, C. (2002). CCTV in london. Working Paper 6, Report to the European Commission Fifth Framework RTD as part of UrbanEye: on the threshold of the urban panopticon.

[89] Mehran, R., Oyama, A., and Shah, M. (2009). Abnormal crowd behavior detection using social force model. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–942. IEEE.

[90] Micusik, B. and Pajdla, T. (2010). Simultaneous surveillance camera calibration and foot-head homology estimation from human detections. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1562–1569.

[91] Milgram, J., Cheriet, M., and Sabourin, R. (2006). one against one or one against all: Which one is better for handwriting recognition with svms? In *Tenth International Workshop on Frontiers in Handwriting Recognition*.

[92] Mobahi, H., Zhou, Z., Yang, A. Y., and Ma, Y. (2011). Holistic 3D reconstruction of urban structures from low-rank textures. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 593–600. IEEE.

[93] Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340.

[94] Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1498–1505. IEEE.

[95] Palaio, H., Maduro, C., Batista, K., and Batista, J. (2009). Ground plane velocity estimation embedding rectification on a particle filter multi-target tracking. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 825–830. IEEE.

[96] Park, M., Brocklehurst, K., Collins, R. T., and Liu, Y. (2009). Deformed Lattice Detection in Real-World Images Using Mean-Shift Belief Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31 (10):pages 1804–1816.

[97] Pelleg, D. and Moore, A. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. In *In Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734.

[98] Pflugfelder, R. and Bischof, H. (2005). Online auto-calibration in man-made worlds. In *DICTA ' 05. Proceedings Digital Image Computing: Technqiues and Applications, 2005.*, pages 519–526. IEEE.

[99] Pflugfelder, R. and Bischof, H. (2006). Computing of the epipolar geometry of slightly overlapping views. In Chum, O. and Franc, V., editors, *Proceedings of the Computer Vision Winter Workshop 2006*, pages 58–63.

[100] Pollard, D. (1982). A central limit theorem for empirical processes. *Journal of the Australian Mathematical Society (Series A)*, 33:pages 235–248.

[101] Pollefeys, M., Koch, R., Vergauwen, M., and Van Gool, L. (1999). Hand-held acquisition of 3D models with a video camera. In *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pages 14–23. IEEE.

[102] Rabaud, V. and Belongie, S. (2006). Counting crowded moving objects. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, volume 1, pages 705–711. IEEE.

[103] Renno, J., Orwell, J., and Jones, G. A. (2002). Learning Surveillance Tracking Models for the Self-Calibrated Ground Plane. *Zidonqhua Xuebao [Acta Automatica Sinica]*, 29 (3):pages 381–392.

[104] Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:pages 101–141.

[105] Rother, C. (2002). A new approach for vanishing point detection in architectural environments. In *In Proc. 11th British Machine Vision Conference*, pages 382–391.

[106] Rother, D., Patwardhan, K., Aganj, I., and Sapiro, G. (2008). 3D priors for scene learning from a single view. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8. IEEE.

[107] Rother, D., Patwardhan, K. A., and Sapiro, G. (2007). What Can Casual Walkers Tell Us About A 3D Scene? In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[108] Salvador, S. and Chan, P. (2004). Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Tools with Artificial Intelli-*

gence, 2004. ICTAI 2004. 16th IEEE International Conference on, pages 576–584. IEEE.

[109] Saxena, A., Sun, M., and Ng, A. (2007). Learning 3-D scene structure from a single still image. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8.

[110] Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3D: Learning 3D scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31 (5):pages 824–840.

[111] Sheikh, Y., Haering, N., and Shah, M. (2006). Shape from Dynamic Texture for Planes. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2285–2292. IEEE.

[112] Shi, J. and Tomasi, C. (1994). Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600. Seattle.

[113] Sochman, J. and Hogg, D. C. (2011). Who knows who-inverting the social force model for finding groups. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 830–837. IEEE.

[114] Sonka, M., Hlavac, V., and Boyle, R. (2014). *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 4th edition.

[115] Stauffer, C. and Grimson, W. E. L. (2002). Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 2:page 252 Vol. 2.

[116] Stauffer, C., Tieu, K., and Lee, L. (2003). Robust Automated Planar Normalization of Tracking Data. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*.

[117] Stentiford, F. (2006). Attention-Based Vanishing point detection. In *In Proc. ICIP 2006*, pages 8–11.

[118] Stevens, K. and of Technology, M. I. (1980). *Surface Perception from Local Analysis of Texture and Contour*. Ph.D. thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[119] Szeliski, R. (1996). Video mosaics for virtual environments. *Computer Graphics and Applications, IEEE*, 16 (2):pages 22–30.

[120] Terzopoulos, D. (1983). Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24 (1):pages 52–96.

[121] Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University.

[122] Triggs, B. (1998). Autocalibration from planar scenes. In Burkhardt, H. and Neumann, B., editors, *Computer Vision ECCV'98*, volume 1406 of *Lecture Notes in Computer Science*, pages 89–105. Springer Berlin Heidelberg.

[123] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle Adjustment A Modern Synthesis. In Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, chapter 21, pages 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg.

[124] Tsai, R. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3 (4):pages 323–344.

[125] Vogiatzis, G. and Hernández, C. (2010). Practical 3D Reconstruction Based on Photometric Stereo. In Cipolla, R., Battiato, S., and Farinella, G., editors, *Computer Vision*, volume 285 of *Studies in Computational Intelligence*, chapter 12, pages 313–345. Springer Berlin Heidelberg, Berlin, Heidelberg.

[126] Wang, L. L. and Tsai, W. H. (1991). Camera Calibration by Vanishing Lines for 3-D Computer Vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13 (4):pages 370–376.

[127] Wang, Z., Zhao, M., Song, Y., Kumar, S., and Li, B. (2010). YouTubeCat: Learning to categorize wild web videos. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 879–886. IEEE.

[128] Witkin, A. P. (1981). Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17 (1-3):pages 17–45.

[129] Wöhler, C. (2012). *3D Computer Vision: Efficient Methods and Applications*. X.media.publishing. Springer.

[130] Yin, F., Makris, D., Orwell, J., and Velastin, S. (2011). Learning Non-coplanar Scene Models by Exploring the Height Variation of Tracked Objects. In Kimmel,

R., Klette, R., and Sugimoto, A., editors, *Computer Vision ACCV 2010*, volume 6494 of *Lecture Notes in Computer Science*, pages 262–275. Springer Berlin Heidelberg.

[131] Yoon, J. J., Koch, C., and Ellis, T. J. (2004). Vision based occupant detection system by monocular 3D surface reconstruction. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 435–440. IEEE.

[132] Zhang, R., Tsai, P. S., Cryer, J. E., and Shah, M. (1999). Shape-from-shading: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21 (8):pages 690–706.

[133] Zhang, Z. (2000). A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22 (11):pages 1330–1334.

[134] Zhang, Z., Ganesh, A., Liang, X., and Ma, Y. (2012). TILT: Transform Invariant Low-Rank Textures. *International Journal of Computer Vision*, 99 (1):pages 1–24.

[135] Zhang, Z., Li, M., Huang, K., and Tan, T. (2008). Robust automated ground plane rectification based on moving vehicles for traffic scene surveillance. In *2008 15th IEEE International Conference on Image Processing*, pages 1364–1367. IEEE.

Appendices

Appendix A

Derivation of Elevation and Yaw from Plane Normal

In chapter 3, section 3.1 it is shown that the unit-normal of a plane in camera coordinates can be decomposed into two rotations of the camera, see equation 3.1. The aim of this appendix is to explicitly demonstrate the derivation of that mapping.

First, consider the case of a camera looking directly down at a plane, such that it has a bird's eye view of it. In this situation, the z-axis of the camera is towards the optical centre and the plane's normal in camera-space is:

$$\mathbf{n} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{A.1}$$

Now suppose that the camera is no longer looking directly down the normal of the plane and instead views it at an angle. In the world-coordinate system, the plane orientation remains fixed as above and the camera rotates about the world-origin. In this case, rotating the camera about the x-axis can be achieved using the standard 3D rotation matrix in the x-axis:

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{A.2}$$

However, the model used in this work dictates that the camera and its axes are fixed, rather the plane must rotate in camera-space. As a result, the directionality of the angle must change. As the cosine function is even, its terms do not change; however the sine function is uneven, so its terms must change sign. This gives the rotation matrix for the angle of elevation θ:

$$\mathbf{R}_{AoE} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{pmatrix} \tag{A.3}$$

Applying the above to the plane normal gives:

$$\mathbf{R}_{AoE}\mathbf{n} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{A.4}$$

$$= \begin{pmatrix} 0 \\ -\sin(\theta) \\ -\cos(\theta) \end{pmatrix} \tag{A.5}$$

Yaw is a rotation about the z-axis of the camera, again with directionality changed to reflect the conversion from world to camera coordinates:

$$\mathbf{R}_{yaw} = \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.6}$$

The final step is to apply the yaw rotation ψ to the elevation-corrected vector:

$$\mathbf{R}_{yaw}\mathbf{R}_{AoE}\mathbf{n} = \begin{pmatrix} -\sin(\psi)\sin(\theta) \\ -\cos(\psi)\sin(\theta) \\ -\cos(\theta) \end{pmatrix} \tag{A.7}$$

$$= -\begin{pmatrix} \sin(\psi)\sin(\theta) \\ \cos(\psi)\sin(\theta) \\ \cos(\theta) \end{pmatrix} \tag{A.8}$$