ABSTRACT

Title of dissertation:	STRUCTURE FROM MOTION ON TEXTURES: THEORY AND APPLICATION TO CALIBRATION
	Patrick Baker Doctor of Philosophy, 2005
Dissertation directed by:	Professor Yiannis Aloimonos Department of Computer Science

This dissertation introduces new mathematical constraints that enable us, for the first time, to investigate the correspondence problem using texture rather than point and lines. These three multilinear constraints are formulated on parallel equidistant lines embedded in a plane. We choose these sets of parallel lines as proxies for fourier harmonics embedded on a plane as a sort of “texture atom”. From these texture atoms we can build up arbitrarily textured surfaces in the world. If we decompose these textures in a Fourier sense rather than as points and lines, we use these new constraints rather than the standard multifocal constraints such as the epipolar or trifocal. We propose some mechanisms for a possible feedback solution to the correspondence problem.

As the major application of these constraints, we describe a multicamera calibration system written in C and MATLAB which will be made available to the public. We describe the operation of the program and give some preliminary results.

STRUCTURE FROM MOTION ON TEXTURES: THEORY AND APPLICATION TO CALIBRATION

by

Patrick Baker

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

For Lynn

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

This dissertation describes a theory of structure from motion which is based on the atom of the textured planar patches rather than the atoms of points and lines. While one can construct a beautiful theory based on points and lines [10, 24, 18, 14, 17, 4, 15], it has proved to be inadequate when used for some applications, especially natural scenes and scenes with repetitive texture. We believe that when the world is modeled by points and lines, attempts to make measurements can lead to errors and biases when the reality does not match the model assumptions. It is surprising that the theory of structure from motion on textures described here is just as beautiful as the current theory, even while it uses a model which can be thought of as more complicated. In fact, in the new theory there are linear reconstructions and constraints which operate on the textured planar patches, in analogy to the epipolar [18] and trilinear [24] constraints with which we are all familiar.

While we believe these ideas have general applicability, we have chosen to apply them first in the domain of calibration of multiple cameras as an initial calibration. We believe these texture constraints can provide a more flexible, stable, and accurate calibration system than those based on points [2, 3, 25]. We have implemented a calibration program in Matlab and C and are making it generally available to the vision community.

The basic framework in structure from motion has been pursued in a three step approach of:

1. finding correspondence

2. computing camera geometry

3. computing scene geometry

Correspondence consists of finding image features that have been generated by the same world feature. The world features are usually modeled by points and lines, although this can be a source of error. Tracking has had some success in domains where there are obvious corners to track [21], but other more natural scenes have proved impossible for current methods. The small baseline stereo problem is solved reliably in hardware when sufficient texture is present [9].

There has been some recent work in wide-baseline stereo [7, 27]. This work consists of programs which attempt to match in wide-baseline situations and they report some success in the situations which they attempted. Instead of trying to implement a system, in this dissertation we begin to address this first step of correspondence in a novel manner by combining geometry and signal processing. Assume that you are given two views and you are asked to match them. Whatever you do, you will have to silently assume that these are views of the same scene. To know that two views are views of the same scene, you have to know something about the camera geometry and the scene model. The correspondence problem can only be posed as a question when some information about camera and scene geometry is available. But the reason we want to solve the correspondence problem is to find the camera and scene geometry. This means, that correspondence, camera and scene geometry have to be solved at the same time. By using texture, we can possibly break these dependencies. The importance of texture has already been noted by several research groups [28, 22], but this work argues along a different line altogether.

In order obtain a deeper insight into the correspondence problem, we need to relax our idea of correspondence. For if we have corresponding points as input there is clearly no reason to develop constraints on textures. Often obtaining these correspondences is difficult, so we break up the problem.

If we have two images, we say that two image regions are in **patch correspondence** if they are images of the same planar patch with some uniform texture property. For instance, in figure 1.1, the amorphous patches of the two buildings are in patch correspondence, even though we have not corresponded individual points. We show how to use this idea as a basis for geometric constraints.

Somehow, the intuitive notion of the wavelength of a signal has to be considered. If you have a collection of textured planes, and these planes do not contain any wavelength greater than λ, then it is clear that if our camera positions are not known to accuracy at least less than λ, then it is impossible to compute any sort of correspondence. On the other

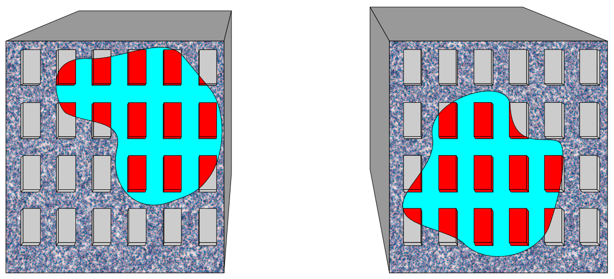
Figure 1.1: The amorphous regions are in **patch correspondence**

hand, if we know our camera positions to within $\alpha << \lambda$, and we have many textures with wavelengths greater than λ, then it should somehow be possible to match with a high degree of probability. The smaller α is, the higher the probability that we find a match.

The above intuitive notions strongly suggest that the next step in making 3D models is to formulate a feedback mechanism. Feedback has been tried to some degree in computer vision. RANSAC [11] and bundle adjustment [6] are forms of feedback, but they use as input the points and lines we find inadequate. Somehow, the feedback mechanism should work in a way that better measurements are introduced in the process.

The current state of the art implicitly assumes some conditions on the structure and camera geometry, which usually allow a reasonable answer. This answer will contain some errors, which we need to correct so that precision can be considerably increased. In that effort, features such as points and lines cannot be of much help. As a community, we have more or less completely understood the multiview geometry of features. But working with features is not the only approach in Vision. We can also work with frequencies, and this dissertation works out the geometry of frequencies projected into multiple cameras.

We start by constructing some new atoms for structure from motion. Consider a planar patch in space that contains one harmonic component (a sinusoid) and consider viewing this patch from different cameras. A sinusoid can be thought of as a set of parallel lines. Thus, we transform the problem into the analysis of a patch containing a set of parallel lines viewed from different cameras. We show that there exist new constraints, like the harmonic epipolar, the harmonic trifocal, the harmonic directionality constraint - a generalization of vanishing point constraint (also analyzed in a different way by [23]), and others. Previous studies of the relationship between a textured surface, its orientation, and its projected image focus on extracting depth cues from single images, by analyzing feature densities, areas, sizes, or variation [1, 13, 8].

There could be a large number of uses for these new constraints and we will point to some of them. One way to think of the new theory is as a new tool for improving existing solutions, i.e. a tool for performing feedback. We discuss this briefly in chapter 8.

The main application of the theory in this dissertation is the calibration of multiple cameras using textures rather than points [26, 29] and lines [16]. We have programmed a multiple camera calibration toolbox which is to be made freely available. We will show results which show that we get accurate subpixel calibration even in the presence of nonlinear distortion, and with very little user effort. Because we use textures, we are able to calibrate even cameras which do not overlap in their fields of view. In addition, the measurement of an extended textures can be made more accurate than the measurement of a single point, since we have a larger region of support.

In chapter 2, we go over some fundamental mathematics to set the notation. In chapter 3 we show how to project points, lines, and textures into cameras using Plücker coordinates [20, 5]. In chapter 4, we show how to reconstruct these points, lines, and textures from their projections. The reconstruction of the texture is the main mathematical contribution of this dissertation. In chapter 5, we use incidence relations to form the constraints on points, lines, and textures. In chapter 6, we discuss how we can make these constraints operate on fewer cameras than they were formulated for, and also make explicit the relation of geometry to texture. In chapter 7 we describe the calibration application, its use and operation, and give some preliminary results. We also show how to apply nonlinear calibration methods to lines rather than points, as is done in [12, 30]. In chapter 8 we go into more detail about how feedback relates to the correspondence problem, and give some possible theorem statements (unproved).

Chapter 2

Mathematical Fundamentals

We describe here some concepts which may not be familiar to the reader so that we may later feel free to use them.

2.1 Homogeneous Coordinates

If we have a certain set of objects S which we would like to coordinatize with vectors, it is often convenient to have more than one vector refer to the same object. More specifically,

Definition 1 *If for any object $s \in S$ with coordinate* $\mathbf{s}$*, the coordinate* $\lambda\mathbf{s}$ *also refers to s for any $\lambda \in \mathbb{R}$, then the coordinates are said to be* homogeneous. *The coordinate* $\mathbf{0}$ *does not represent any object $s \in S$.*

Homogeneous coordinates allows us to use linear algebra in projective spaces. We define three such projective spaces in this paper:

1. The projective plane $\mathbb{P}^2$

2. The Plücker lines

3. The space of the textured planes

2.2 Vector Algebra

We use vector algebra extensively in the derivations of this paper. First, let us note that, for consistency, *all* our vectors are column vectors, and we transpose them as necessary. We also use the following vector algebraic identities extensively in this paper:

$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) = \mathbf{b}\mathbf{c}^{\mathrm{T}}\mathbf{a} - \mathbf{c}\mathbf{a}^{\mathrm{T}}\mathbf{b} \tag{2.1}$$

and

$$\mathbf{a}^{\mathrm{T}}(\mathbf{b}\times\mathbf{c}) = \mathbf{c}^{\mathrm{T}}(\mathbf{a}\times\mathbf{b}) \tag{2.2}$$

$$|\mathbf{a}\ \mathbf{b}\ \mathbf{c}| = |\mathbf{c}\ \mathbf{a}\ \mathbf{b}| \tag{2.3}$$

where both $\cdot^{\mathrm{T}}(\cdot\times\cdot)$ and $|\cdot\ \cdot\ \cdot|$ are used to denote the triple product. The latter notation is used because the determinant of $[\mathbf{a}\ \mathbf{b}\ \mathbf{c}]$ is equal to the triple product of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$.

We also use the following identity for the inverse B^{-1} of a matrix B.

Fact 1 *If we have a matrix:*

$$B = \begin{bmatrix} \mathbf{b}_1^{\mathrm{T}} \\ \mathbf{b}_2^{\mathrm{T}} \\ \mathbf{b}_3^{\mathrm{T}} \end{bmatrix} \tag{2.4}$$

then its inverse is:

$$B^{-1} = \frac{1}{\det(B)} \begin{bmatrix} \mathbf{b}_2 \times \mathbf{b}_3 & \mathbf{b}_3 \times \mathbf{b}_1 & \mathbf{b}_1 \times \mathbf{b}_2 \end{bmatrix} \tag{2.5}$$

Proof. Note that

$$\frac{1}{\det(B)} \begin{bmatrix} \mathbf{b}_1^{\mathrm{T}} \\ \mathbf{b}_2^{\mathrm{T}} \\ \mathbf{b}_3^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{b}_2 \times \mathbf{b}_3 & \mathbf{b}_3 \times \mathbf{b}_1 & \mathbf{b}_1 \times \mathbf{b}_2 \end{bmatrix} \tag{2.6}$$

$$= \frac{1}{\det(B)} \begin{bmatrix} |\mathbf{b}_1\ \mathbf{b}_2\ \mathbf{b}_3| & |\mathbf{b}_1\ \mathbf{b}_3\ \mathbf{b}_1| & |\mathbf{b}_1\ \mathbf{b}_1\ \mathbf{b}_2| \\ |\mathbf{b}_2\ \mathbf{b}_2\ \mathbf{b}_3| & |\mathbf{b}_2\ \mathbf{b}_3\ \mathbf{b}_1| & |\mathbf{b}_2\ \mathbf{b}_1\ \mathbf{b}_2| \\ |\mathbf{b}_3\ \mathbf{b}_2\ \mathbf{b}_3| & |\mathbf{b}_3\ \mathbf{b}_3\ \mathbf{b}_1| & |\mathbf{b}_3\ \mathbf{b}_1\ \mathbf{b}_2| \end{bmatrix} \tag{2.7}$$

using equation (2.2), noting $|\mathbf{b}_i\ \mathbf{b}_i\ \mathbf{b}_j| = 0$, and dividing we obtain

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Since BB^{-1} multiplies to the identity, then B and B^{-1} must be inverses. $\square$

Chapter 3

Atoms, Cameras and Projection

In this section we discuss the atoms on which we base our new theory. The list of atoms includes points, lines, and textures in the world. In addition, we discuss what a camera is and how it projects our atoms into an image. We do not at this point discuss how to measure these projections, as this is better addressed as the signal processing problem which we discuss later.

3.1 Points and Cameras

We define world points in three dimensions as one would expect.

Definition 2 *A* **world point** *is an element $P \in \mathbb{R}^3$. We may represent such a point in a particular coordinate system as:*

$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.1}$$

Our image points exist in the projective plane $\mathbb{P}^2$. This space has the advantage of duality between points and lines, which makes it extremely easy to consider both and transform formulas which consider one into formulas which consider the other.

Definition 3 *An* **image point** *is an element $p \in \mathbb{P}^2$. We may represent such a point in a particular coordinate system as:*

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.2}$$

These coordinates are homogeneous.

Note that the coordinates of the world and image points are both 3-vectors, so our *representation* for both coordinates is identical. We can think of a camera as as device for considering the coordinates in $\mathbb{R}^3$ to be coordinates in $\mathbb{P}^2$. However, our world points exist in one fiducial coordinate system, while the image points exist in the particular camera coordinate systems. Therefore our camera is defined in relation to this fiducial coordinate system as follows.

Definition 4 *A* **camera** *C is a map $C : \mathbb{R}^3 \to \mathbb{P}^2$ from world points to image points. Given a fiducial coordinate system, we may represent this map with a pair $(B, \mathbf{T})$, where $B : \mathbb{R}^3 \to \mathbb{R}^3$ is a linear function (represented by a 3×3 matrix), and $\mathbf{T}$ is a 3-vector representing the* **camera center**. *The action of the map on a world point coordinate $\mathbf{P}$ is:*

$$C(\mathbf{P}) = B \cdot (\mathbf{P} - \mathbf{T}) \tag{3.3}$$

where $C(\mathbf{P})$ is considered as a member of $\mathbb{P}^2$.

Here we have defined a camera as taking a world point and mapping it to an image point through a general linear transformation on the world coordinates. Note that we have defined the transformation as first a translation and then a matrix multiplication on the world point. This allows us to easily undo the matrix multiplication on the image point by applying B^{-1}. Each camera will then be only a translation away from the fiducial coordinate system, which allows easier derivation of our constraints. Additionally, note that the matrix B is applied to the 3-vector resulting from the translations of the point $\mathbf{P}$. B may be considered to be a transformation on the world points $\mathbb{R}^3$ or of the image points $\mathbb{P}^2$. In fact, B is usually split apart using a QR decomposition, with the orthogonal matrix representing a rotation of the camera (a transformation on $\mathbb{R}^3$) and the residual matrix representing a linear transformation of the image (a transformation on $\mathbb{P}^2$). Since the coordinates are the same, we ignore such distinctions for the present.

This definition of a camera represents real world cameras reasonably well. For better accuracy, it advisable to consider B to be a parameterized nonlinear function $B : \mathbb{R}^3 \to \mathbb{R}^3$ to account for radial, tangential, and other sorts of distortion. Later in this paper we extend our theory to account for this sort of projection model.

3.2 Lines and Cameras

Now that we see how world points are projected to image points, we can derive similar formulas for lines. First, we define a convenient coordinate system for world lines, the Plücker coordinate system.

Definition 5 *A* **world line** *L is the set of all the points $P \in \mathbb{R}^3$ such that $\mathbf{P} = (1-\lambda)\mathbf{Q}_1 + \lambda\mathbf{Q}_2$ for two points $\mathbf{Q}_i$, and some scalar λ. The Plücker coordinates of this line are $\mathbf{L} = \left[\begin{smallmatrix}\mathbf{L}_d \\ \mathbf{L}_m\end{smallmatrix}\right]$, where:*

$$\mathbf{L}_d = \mathbf{Q}_2 - \mathbf{Q}_1 \qquad \textit{direction of } L \tag{3.4}$$

$$\mathbf{L}_m = \mathbf{L}_d \times \mathbf{P} \qquad \textit{moment of } L \tag{3.5}$$

$\mathbf{L}_m$ has many alternative definitions

Fact 2 *Given the variables defined in definition 5, the following definitions for $\mathbf{L}_m$ are equivalent.*

$$\mathbf{L}_m = \mathbf{L}_d \times \mathbf{P} \tag{3.6}$$

$$= \mathbf{L}_d \times ((1-\lambda)\mathbf{Q}_1 + \lambda\mathbf{Q}_2) \tag{3.7}$$

$$= \mathbf{L}_d \times \mathbf{Q}_1 \tag{3.8}$$

$$= \mathbf{L}_d \times \mathbf{Q}_2 \tag{3.9}$$

$$= \mathbf{Q}_2 \times \mathbf{Q}_1 \tag{3.10}$$

Proof. One can derive these equations using vector algebra. □

The meaning of the direction $\mathbf{L}_d$ of the line is clear. The moment of the line has magnitude equal to the minimum distance of the line from the origin, and is perpendicular to the plane containing the origin and the line.We will see shortly why these coordinates are particularly convenient for projection and reconstruction.

Fact 3 *The coordinates $\mathbf{L}$ of L are homogeneous.*

Proof. We calculate, using variables defined in definition 5

$$\lambda \mathbf{L} = \begin{bmatrix} \lambda \mathbf{L}_d \\ \\ \lambda \mathbf{L}_m \end{bmatrix} \tag{3.11}$$

$$= \begin{bmatrix} \lambda \mathbf{L}_d \\ \\ (\lambda \mathbf{L}_d) \times \mathbf{Q}_1 \end{bmatrix} \tag{3.12}$$

$$= \begin{bmatrix} \lambda \mathbf{L}_d \\ \\ (\lambda \mathbf{L}_d) \times \mathbf{Q}_2 \end{bmatrix} \tag{3.13}$$

so that, according to the definition, the line still passes through $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Since $\lambda \mathbf{L}$ and $\mathbf{L}$ represent the same line, we see that these coordinates are homogeneous. □

We can consider the lines to be members of the space $\mathbb{P}^5$. However, note that not every element of $\mathbb{P}^5$ is a line, since by our construction we must have that $\mathbf{L}_d$ and $\mathbf{L}_m$ are perpendicular, or $\mathbf{L}_m^{\mathrm{T}} \mathbf{L}_d = 0$. We thus have parameterized the four dimensional space of 3D lines as a subset of $\mathbb{P}^5$.

For any point P on the line L, the cross product defining $\mathbf{L}_m$ will be the same, so that the choice of point P does not matter. We formalize this

Fact 4 *A world line* $\mathbf{L}$ *is incident on a world point* $\mathbf{P}$ *if and only if* $\mathbf{L}_d \times \mathbf{P} = \mathbf{L}_m$.

Proof. From fact 2, it is clear that if $\mathbf{P}$ is on $\mathbf{L}$, then $\mathbf{L}_d \times \mathbf{P} = \mathbf{L}_m$. If $\mathbf{P}$ is not on $\mathbf{L}$, then there is no λ such that $\mathbf{P} = (1 - \lambda)\mathbf{Q}_1 + \lambda \mathbf{Q}_2$. We must therefore have that

$\mathbf{P} = (1-\lambda)\mathbf{Q}_1 + \lambda\mathbf{Q}_2 + \mathbf{v}$, where $\mathbf{v}$ is linearly independent of either of the $\mathbf{Q}_i$, or both.

$$\mathbf{L}_d \times \mathbf{P} = \mathbf{L}_d \times ((1-\lambda)\mathbf{Q}_1 + \lambda\mathbf{Q}_2 + \gamma\mathbf{v}) \tag{3.14}$$

$$= \mathbf{L}_m + \mathbf{L}_d \times \mathbf{v} \tag{3.15}$$

$$= \mathbf{L}_m + (\mathbf{Q}_2 - \mathbf{Q}_1) \times \mathbf{v} \tag{3.16}$$

And we know the last term must be nonzero because $\mathbf{v}$ is linearly independent of the difference $\mathbf{Q}_2 - \mathbf{Q}_1$, so we know $\mathbf{L}_d \times \mathbf{P} \neq \mathbf{L}_m$. $\square$

Now we show the condition for incidence of lines on lines.

Fact 5 *If we have two lines $\mathbf{L}_1$ and $\mathbf{L}_2$, they intersect if and only if*

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0 \tag{3.17}$$

If they do intersect, their point of intersection is

$$\mathbf{P} = \frac{\mathbf{L}_{m,1} \times \mathbf{L}_{m,2}}{\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2}} \tag{3.18}$$

$$= \frac{\mathbf{L}_{m,2} \times \mathbf{L}_{m,1}}{\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1}} \tag{3.19}$$

Proof. By construction, we know that the coordinates of all points on a line $\mathbf{L}_i$ are perpendicular to the $\mathbf{L}_{m,i}$. Since our point of intersection is perpendicular to both the $\mathbf{L}_{m,i}$, it must be equal to

$$\mathbf{P} = \lambda(\mathbf{L}_{m,1} \times \mathbf{L}_{m,2}) \tag{3.20}$$

for some λ. We also know from fact 4 that

$$\mathbf{L}_{d,i} \times \mathbf{P} = \mathbf{L}_{m,i} \tag{3.21}$$

so that, substituting equation (3.20) into (3.21)

$$\lambda(\mathbf{L}_{d,i} \times (\mathbf{L}_{m,1} \times \mathbf{L}_{m,2})) = \mathbf{L}_{m,i} \tag{3.22}$$

for $i = 1$ we get

$$\lambda(\mathbf{L}_{m,1}\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} - \mathbf{L}_{m,2}\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,1}) = \mathbf{L}_{m,1} \tag{3.23}$$

$$\lambda\mathbf{L}_{m,1}\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} = \mathbf{L}_{m,1} \tag{3.24}$$

for $i = 2$ we get

$$\lambda(\mathbf{L}_{m,1}\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,2} - \mathbf{L}_{m,2}\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1}) = \mathbf{L}_{m,2} \tag{3.25}$$

$$-\lambda\mathbf{L}_{m,2}\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = \mathbf{L}_{m,2} \tag{3.26}$$

We can only satisfy both these equations if $\lambda = \frac{1}{\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2}}$ in equation (3.24) and $\lambda = -\frac{1}{\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1}}$ in equation (3.26). We thus obtain

$$\frac{1}{\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2}} = -\frac{1}{\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1}} \tag{3.27}$$

so we obtain

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0 \tag{3.28}$$

We thus know that if there is a point of intersection, then our desired equation must hold. If our desired equation holds, then we can calculate our point of intersection as in the theorem statement. □

We wish to define image lines, so that we may project our world lines onto the image. It is well known that in the space $\mathbb{P}^2$, the lines are dual to the points, and can also be represented as homogeneous 3-vectors. One may think of the image line coordinate as a normal to the plane containing the line. The magnitude of our coordinates has no meaning in $\mathbb{P}^2$. One may also think of the image line as a great circle on a sphere, with the coordinates representing the axis perpendicular to all the points of the great circle.

Definition 6 *An* **image line** *is a line $\ell \in \mathbb{P}^2$. The line may be given coordinates*

$$\boldsymbol{\ell} = \begin{bmatrix} \ell_1 \\ \ell_2 \\ \ell_3 \end{bmatrix} \tag{3.29}$$

A point p is incident on a line ℓ if and only if

$$\mathbf{p}^{\mathrm{T}}\boldsymbol{\ell} = 0 \tag{3.30}$$

Since a line and point are incident if and only if their coordinates are perpendicular to each other, this provides a simple means for finding the line incident on two points

Fact 6 *The line ℓ which is connects two points p_1 and p_2 has coordinates $\boldsymbol{\ell} = \mathbf{p}_1 \times \mathbf{p}_2$*

and the point at the intersection of two lines

Fact 7 *The point p which is on both lines ℓ_1 and ℓ_2 has coordinates $\mathbf{p} = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$*

We show how to find the linear transformation on image lines given a linear transformation on image points.

Fact 8 *If we transform the image points $\hat{\mathbf{p}} = B\mathbf{p}$, then the image lines are transformed by $\hat{\boldsymbol{\ell}} = B^{-\mathrm{T}}\boldsymbol{\ell}$.*

Proof. We note that point/line incidences must be preserved by the linear transformation. In particular, if $\mathbf{p}^{\mathrm{T}}\boldsymbol{\ell} = 0$, then $\hat{\mathbf{p}}^{\mathrm{T}}\hat{\boldsymbol{\ell}} = 0$ still holds after the transformation of $\mathbf{p}$ and $\boldsymbol{\ell}$. We want D so that

$$\hat{\mathbf{p}}^{\mathrm{T}}\hat{\boldsymbol{\ell}} = 0 \tag{3.31}$$

$$(B\mathbf{p})^{\mathrm{T}} D\boldsymbol{\ell} = 0 \tag{3.32}$$

$$\mathbf{p}^{\mathrm{T}} B^{\mathrm{T}} D\boldsymbol{\ell} = 0 \tag{3.33}$$

and if $B^{\mathrm{T}}D = I_3$

$$\mathbf{p}^{\mathrm{T}}\boldsymbol{\ell} = 0 \tag{3.34}$$

So therefore we get $D = B^{-\mathrm{T}}$. $\square$

Fact 9 *If we have a line L and a camera $(B, \mathbf{T})$, then the image line associated with L is*

$$\hat{\boldsymbol{\ell}} = B^{-\mathrm{T}}(\mathbf{L}_m - \mathbf{T}\times\mathbf{L}_d) \tag{3.35}$$

Proof. If we have two points on the world line P_1 and P_2, the coordinates of their image points on a camera $(B, \mathbf{T})$ are just $B(\mathbf{P}_1 - \mathbf{T})$ and $B(\mathbf{P}_2 - \mathbf{T})$, considered as members of $\mathbb{P}^2$. Then the image line containing them must be just

$$\hat{\boldsymbol{\ell}} = (B(\mathbf{P}_1 - \mathbf{T}))\times(B(\mathbf{P}_2 - \mathbf{T})) \tag{3.36}$$

$$= B^{-\mathrm{T}}(\mathbf{P}_1\times\mathbf{P}_2 - (\mathbf{T}\times(\mathbf{P}_2 - \mathbf{P}_1))) \tag{3.37}$$

$$= B^{-\mathrm{T}}(\mathbf{L}_m - \mathbf{T}\times\mathbf{L}_d) \tag{3.38}$$

$\square$

Here is an interesting parallel between the projection of points and lines. The coordinates of the projection of a linearly transformed world point are the same as $\mathbf{P}$, just considered in $\mathbb{P}^2$. The coordinates of the projection of a linearly transformed world line are equal to $\mathbf{L}_m$, considered in $\mathbb{P}^2$.

3.3 Textures and Cameras

The third object that we would like to project is the singly textured plane. This is a new object in computer vision and it will allow us to incorporate signal processing directly into our framework, rather than adding it on as an afterthought. The idea is to define mathematical objects consisting of equally spaced lines on a plane in 3D. With this we can represent a simple texture on a plane. By putting many of these objects together with appropriate constraints, we may represent any arbitrary textured plane, but in a fashion which allows for geometric reasoning about correspondence and reconstruction.

Let us assume that we have a periodic texture in space which is embedded in a 3D plane. Since this is a periodic texture, we may think of it as a set of lines, equally spaced, embedded in the plane. We can represent the plane and the texture embedded in the plane together in a geometrical way as follows.

We motivate our definition as follows. Consider one line from the set of equally spaced lines in the plane, call it $\mathbf{L}_0$. We may represent this line using Plücker coordinates as $\mathbf{L}_0 = \left[\begin{smallmatrix} \mathbf{L}_d \\ \mathbf{L}_m \end{smallmatrix} \right]$. We take $\mathbf{Q}_0$ to be a point on $\mathbf{L}_0$, and the point $\mathbf{Q}_n = \mathbf{Q}_0 + n\mathbf{d}$ to be on the n^{th} line in the texture for some direction $\mathbf{d}$. From fact 4 we see

$$\mathbf{L}_d \times \mathbf{Q}_0 = \mathbf{L}_{m,0} \tag{3.39}$$

so that to get $\mathbf{L}_{m,n}$

$$\begin{aligned}
\mathbf{L}_{m,n} &= \mathbf{L}_d \times \mathbf{Q}_n && (3.40)\\
&= \mathbf{L}_d \times \mathbf{Q}_0 + n\mathbf{L}_d \times \mathbf{d} && (3.41)\\
&= \mathbf{L}_m + n\mathbf{L}_\lambda && (3.42)
\end{aligned}$$

where $\mathbf{L}_\lambda = \mathbf{L}_d \times \mathbf{d}$. Note that since both $\mathbf{L}_d$ and $\mathbf{d}$ are vectors which lie inside the plane, we must have that $\mathbf{L}_\lambda$ is normal to the textured plane. This leads us to the following definition, as shown in figure 3.1

Definition 7 *A* **singly textured plane** *H is a set of lines, equally spaced, embedded in a world plane. We give the textured plane coordinates*

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix} \tag{3.43}$$

with $\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_m = 0$ and $\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_\lambda = 0$. The coordinates of each line in the plane, indexed by n are:

$$\mathbf{L}_n = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m + n\mathbf{L}_\lambda \end{bmatrix} \tag{3.44}$$

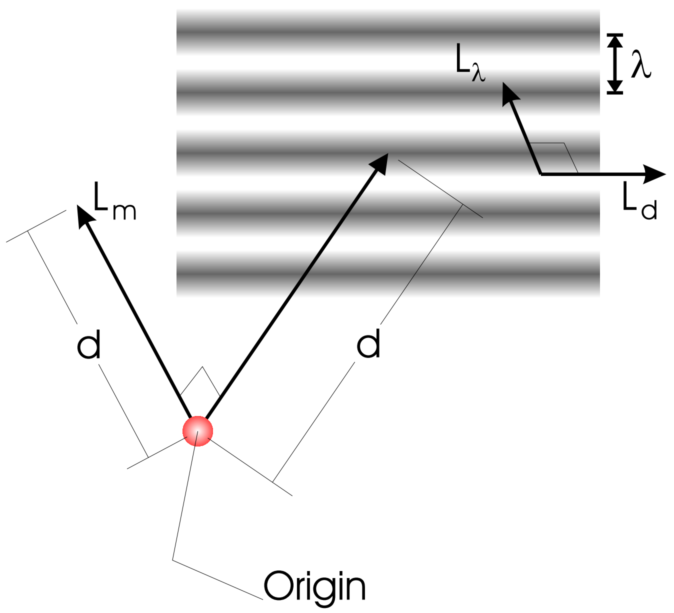

Figure 3.1: The Parameters of a Textured Plane

Fact 10 *If we have two textured planes* $\mathbf{H}_1$ *and* $\mathbf{H}_2$*, then they lie on the same world plane if and only if:*

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0 \tag{3.45}$$

and

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{\lambda,2} = 0 \qquad \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{\lambda,1} = 0 \tag{3.46}$$

Proof. Let us assume that we have two textured planes $\mathbf{H}_1$ and $\mathbf{H}_2$ which are coincident. Because of this, we know that line $\left[\begin{smallmatrix}\mathbf{L}_{d,1}\\ \mathbf{L}_{m,1}\end{smallmatrix}\right]$ intersects with all lines $\left[\begin{smallmatrix}\mathbf{L}_{d,2}\\ \mathbf{L}_{m,2}+i_2\mathbf{L}_{\lambda,2}\end{smallmatrix}\right]$, and $\left[\begin{smallmatrix}\mathbf{L}_{d,2}\\ \mathbf{L}_{m,2}\end{smallmatrix}\right]$ intersects with all lines $\left[\begin{smallmatrix}\mathbf{L}_{d,1}\\ \mathbf{L}_{m,1}+i_1\mathbf{L}_{\lambda,1}\end{smallmatrix}\right]$. With fact 5, and with $i_2 = 0$, we obtain

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0 \tag{3.47}$$

With $i_2 = 1$, we obtain

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{\lambda,2} = 0 \tag{3.48}$$

Similarly, with $i_1 = 1$, we obtain

$$\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{\lambda,1} = 0 \tag{3.49}$$

□

The projected image of a textured plane is easy to calculate from fact 9 and equation (3.44), and consists of a set of image lines.

Fact 11 *The set of image lines $\{\hat{\boldsymbol{\ell}}_n\}$ of a singly textured plane H in a camera with parameters $(B, \mathbf{T})$ is*

$$\{\hat{\boldsymbol{\ell}}_n : \hat{\boldsymbol{\ell}}_n = B^{-\mathrm{T}}(\mathbf{L}_m + n\mathbf{L}_\lambda - \mathbf{L}_d \times \mathbf{T})\} \tag{3.50}$$

where $n \in \mathbb{Z}$.

Note that the image lines have homogeneous coordinates so that the image lines are not equally spaced, as would appear from first glance at the equation. To the contrary, as n goes to infinity, the image lines will approach the image line $B^{-\mathrm{T}}\mathbf{L}_\lambda$, since that term will dominate.

Now if we would like to formulate a parameterization for a doubly textured plane. In this case there are two textured contained in the plane and we would like to have a good parameterization for this structure as a whole. We do this by specifying an $\mathbf{L}_\lambda$, a $\mathbf{P}$, and an $\mathbf{L}_{d,1}^{\perp}$ and $\mathbf{L}_{d,2}^{\perp}$. We may form the parameters for our textured planes with $\mathbf{L}_{d,i} = \mathbf{L}_{d,i}^{\perp} \times \mathbf{L}_\lambda$ and $\mathbf{L}_{m,i} = \mathbf{L}_{d,i} \times \mathbf{P}$. We also set $\mathbf{L}_{\lambda,i} = \mathbf{L}_\lambda$.

We may think of this as defining a 2D lattice of points in 3D. Let us derive the projections of these points into our cameras. Since our lines project to $B^{-\mathrm{T}}(\mathbf{L}_{m,i} + n_i\mathbf{L}_{\lambda,i} - \mathbf{L}_{d,i} \times \mathbf{T})$, we may form our image points as

$$(B^{-\mathrm{T}}(\mathbf{L}_{m,1} + n_1\mathbf{L}_\lambda - \mathbf{L}_{d,1} \times \mathbf{T})) \times (B^{-\mathrm{T}}(\mathbf{L}_{m,2} + n_2\mathbf{L}_\lambda - \mathbf{L}_{d,2} \times \mathbf{T})) \quad (3.51)$$

then we may derive

$$B[((\mathbf{L}_{d,1}^\perp \times \mathbf{L}_\lambda) \times \mathbf{P} + n_1\mathbf{L}_\lambda - (\mathbf{L}_{d,1}^\perp \times \mathbf{L}_\lambda) \times \mathbf{T}) \times ((\mathbf{L}_{d,2}^\perp \times \mathbf{L}_\lambda) \times \mathbf{P} + n_1\mathbf{L}_\lambda - (\mathbf{L}_{d,2}^\perp \times \mathbf{L}_\lambda) \times \mathbf{T})] \quad (3.52)$$

we may then derive

$$B[(\mathbf{L}_\lambda\mathbf{P}^{\mathrm{T}}\mathbf{L}_{d,1}^\perp - \mathbf{L}_{d,1}^\perp\mathbf{P}^{\mathrm{T}}\mathbf{L}_\lambda + n_1\mathbf{L}_\lambda - \mathbf{L}_\lambda\mathbf{T}^{\mathrm{T}}\mathbf{L}_{d,1}^\perp + \mathbf{L}_{d,1}^\perp\mathbf{T}^{\mathrm{T}}\mathbf{L}_\lambda) \times (\mathbf{L}_\lambda\mathbf{P}^{\mathrm{T}}\mathbf{L}_{d,2}^\perp - \mathbf{L}_{d,2}^\perp\mathbf{P}^{\mathrm{T}}\mathbf{L}_\lambda + n_1\mathbf{L}_\lambda - \mathbf{L}_\lambda\mathbf{T}^{\mathrm{T}}\mathbf{L}_{d,2}^\perp + \mathbf{L}_{d,2}^\perp\mathbf{T}^{\mathrm{T}}\mathbf{L}_\lambda)] \quad (3.53)$$

we may then derive

$$B[(\mathbf{L}_\lambda((\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_{d,1}^\perp + n_1) - \mathbf{L}_{d,1}^\perp(\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_\lambda) \times (\mathbf{L}_\lambda((\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_{d,2}^\perp + n_1) - \mathbf{L}_{d,2}^\perp(\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_\lambda)] \quad (3.54)$$

from this we may derive

$$B[(\mathbf{L}_{d,2}^\perp \times \mathbf{L}_\lambda)((\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_{d,1}^\perp(\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_\lambda + n_1(\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_\lambda) (\mathbf{L}_{d,1}^\perp \times \mathbf{L}_\lambda)((\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_{d,2}^\perp(\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_\lambda + n_2(\mathbf{P} - \mathbf{T})^{\mathrm{T}}\mathbf{L}_\lambda) + (\mathbf{L}_{d,1}^\perp \times \mathbf{L}_{d,2}^\perp)((\mathbf{P} - \mathbf{T})\mathbf{L}_\lambda)^2] \quad (3.55)$$

These are the projected points of a doubly textured plane, indexed by n_1 and n_2.

3.4 Normalization Notation

In this paper we make a distinction between the measured image lines/points and the calibrated/derotated image lines/points. If we have a world coordinate system, and a camera in that coordinate system with parameters $(B, \mathbf{T})$, then we consider the $\hat{\mathbf{p}}$ and $\hat{\boldsymbol{\ell}}$ to be the actual point and line measured in the image. For most of the derivations, we use the calibrated/derotated image lines/points

$$\mathbf{p} = B^{-1}\hat{\mathbf{p}} \tag{3.56}$$

$$\boldsymbol{\ell} = (B^{-\mathrm{T}})^{-1}\hat{\boldsymbol{\ell}} \tag{3.57}$$

$$= B^{\mathrm{T}}\hat{\boldsymbol{\ell}} \tag{3.58}$$

Whenever we assume that we already have the B, we will use the calibrated/derotated image lines/points for the calculations. We multiply the appropriate B or $B^{-\mathrm{T}}$ back later. We show that by considering the prismatic line constraint, we are able, when using three cameras, to find the B matrices independently, thus giving this notational convenience a theoretical basis.

Chapter 4

Reconstruction

Now that we have our fundamental objects and the methods we use to project them, we move on to reconstructing the objects from their projections. In the following, we consider that we have one, two, three, or four cameras, and that we know the camera parameters $(B_i, \mathbf{T}_i)$.

Given two arbitrary cameras, it is not in general possible to reconstruct a world point from two image points. A moment's thought will confirm this, since the two world lines formed by the image points with their respective centers of projection do not in general intersect. It is possible to form a joint reconstruction/constraint, but this complicates matters, and yields no benefit.

4.1 Reconstructing a World Line

Fact 12 *If we have a line L in space which projects to two image lines $\hat{\boldsymbol{\ell}}_1$ and $\hat{\boldsymbol{\ell}}_2$ in cameras $(B_1, \mathbf{T}_1)$, and $(B_2, \mathbf{T}_2)$, then we can calculate the coordinates for $\mathbf{L}$ if $|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2| \neq$*

$\mathbf{0}$ *as:*

$$\mathbf{L} = \begin{bmatrix} \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \\ \\ \boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \end{bmatrix} \tag{4.1}$$

Proof. Using fact 9, we know that our projected lines are

$$\boldsymbol{\ell}_i \simeq \mathbf{L}_m - \mathbf{T}_i \times \mathbf{L}_d \qquad\qquad i \in \{1, 2\} \tag{4.2}$$

We assume that $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ are not parallel. Since $\mathbf{L}_m$ is perpendicular to $\mathbf{L}_d$, then we know that the $\boldsymbol{\ell}_i$ are both perpendicular to $\mathbf{L}_d$. Using this, and the fact that the $\boldsymbol{\ell}_i$ are distinct, we can calculate the $\mathbf{L}_d$ as:

$$\mathbf{L}_d = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \tag{4.3}$$

Note that we have used an equals sign here. Since $\mathbf{L}$ has homogeneous coordinates, we need to set a particular scale factor, and this scale factor is a particularly good one, since the magnitude of $\mathbf{L}_d$ will be smaller in cases where the $\boldsymbol{\ell}_i$ are in almost the same direction, which are just the cases where our reconstruction will not be accurate.

Let us now calculate $\mathbf{L}_m$. Since the $\boldsymbol{\ell}_i$ do not have the same direction (since their cross product is not zero) and they are both perpendicular to $\mathbf{L}_d$. Also, since $\mathbf{L}_m$ is perpendicular to $\mathbf{L}_d$, we may write:

$$\mathbf{L}_m = \gamma \boldsymbol{\ell}_1 + \delta \boldsymbol{\ell}_2 \tag{4.4}$$

We can substitute this into equations (4.2), and multiply by scale factors α and β to make the equations equalities, and obtain:

$$\alpha\boldsymbol{\ell}_1 = \gamma\boldsymbol{\ell}_1 + \delta\boldsymbol{\ell}_2 - \mathbf{T}_1\times(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_2) \tag{4.5}$$

$$= \gamma\boldsymbol{\ell}_1 + \delta\boldsymbol{\ell}_2 + \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \boldsymbol{\ell}_1\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_2 \tag{4.6}$$

$$\beta\boldsymbol{\ell}_2 = \gamma\boldsymbol{\ell}_1 + \delta\boldsymbol{\ell}_2 - \mathbf{T}_2\times(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_2) \tag{4.7}$$

$$= \gamma\boldsymbol{\ell}_1 + \delta\boldsymbol{\ell}_2 + \boldsymbol{\ell}_2\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_1 - \boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 \tag{4.8}$$

By equating the coefficients of the $\boldsymbol{\ell}_i$ in these equations, we obtain

$$\delta = \mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 \qquad\qquad \gamma = -\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 \tag{4.9}$$

From equations (4.4) and (4.9) we obtain

$$\mathbf{L}_m = \boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 \tag{4.10}$$

□

We have made the condition that the $\boldsymbol{\ell}_i$ not point in the same direction. The condition is equivalent to:

$$\mathbf{L}_m - \mathbf{L}_d\times\mathbf{T}_1 \simeq \mathbf{L}_m - \mathbf{L}_d\times\mathbf{T}_2 \tag{4.11}$$

which is equivalent to

$$\mathbf{L}_m\times(\mathbf{L}_d\times\mathbf{T}_1) = \mathbf{L}_m\times(\mathbf{L}_d\times\mathbf{T}_2) \tag{4.12}$$

$$\mathbf{L}_d\mathbf{L}_m^{\mathrm{T}}\mathbf{T}_1 - \mathbf{T}_1\mathbf{L}_m^{\mathrm{T}}\mathbf{L}_d = \mathbf{L}_d\mathbf{L}_m^{\mathrm{T}}\mathbf{T}_2 - \mathbf{T}_2\mathbf{L}_m^{\mathrm{T}}\mathbf{L}_d \tag{4.13}$$

using the fact that $\mathbf{L}_m^{\mathrm{T}}\mathbf{L}_d = 0$

$$\mathbf{L}_m^{\mathrm{T}}\mathbf{T}_1 = \mathbf{L}_m^{\mathrm{T}}\mathbf{T}_2 \tag{4.14}$$

$$\mathbf{L}_m^{\mathrm{T}}(\mathbf{T}_2 - \mathbf{T}_1) = 0 \tag{4.15}$$

which means that the translation between the cameras cannot be perpendicular to the moment vector of the world line. Translating in this plane leaves the image line the same in both cameras, so that there is no depth information in the images, and no reconstruction is possible.

We may calculate the line reconstruction even if the $\boldsymbol{\ell}_i$ are parallel, and the result will just be the zero vector, which indicates that we have zero confidence in the location, which is just what we would expect. The arbitrary magnitude of our homogeneous Plücker line can be used as a confidence measure when we actually make calculations.

4.2 Reconstructing a World Point

While it is not in general possible to reconstruct a world point from two arbitrary cameras and image points, it is possible to reconstruct a world point from three arbitrary cameras using image lines which are incident on the world point's image in each of the three cameras. We show the following:

Fact 13 *If we project a world point P into three cameras with parameters $(B_i, \mathbf{T}_i)$, and we have measured image lines $\hat{\boldsymbol{\ell}}_i$ which go through the image points $\hat{\mathbf{p}}_i$, then the coordi-*

nates of P are

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}} \\ \boldsymbol{\ell}_2^{\mathrm{T}} \\ \boldsymbol{\ell}_3^{\mathrm{T}} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1 \\ \boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 \\ \boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3 \end{bmatrix} \tag{4.16}$$

$$= \frac{(\boldsymbol{\ell}_2 \times \boldsymbol{\ell}_3)\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1 + (\boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1)\boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 + (\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2)\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3}{|\boldsymbol{\ell}_1\ \boldsymbol{\ell}_2\ \boldsymbol{\ell}_3|} \tag{4.17}$$

If we have a point in one camera and a line in the other, we can consider $\mathbf{T}_3 = \mathbf{T}_2$, $B_3 = B_2$ and $\mathbf{p} = \boldsymbol{\ell}_2 \times \boldsymbol{\ell}_3$, and obtain

$$\mathbf{P} = \frac{\mathbf{p}_2\boldsymbol{\ell}_1^{\mathrm{T}}(\mathbf{T}_1 - \mathbf{T}_2)}{\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2} + \mathbf{T}_2 \tag{4.18}$$

Proof. Because of definition 6, we know that the image line is perpendicular to the image of the point:

$$\boldsymbol{\ell}_i^{\mathrm{T}}(P - \mathbf{T}_1) = 0 \tag{4.19}$$

Casting this as a matrix equation, we obtain:

$$\begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}} \\ \boldsymbol{\ell}_2^{\mathrm{T}} \\ \boldsymbol{\ell}_3^{\mathrm{T}} \end{bmatrix} \mathbf{P} = \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1 \\ \boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 \\ \boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3 \end{bmatrix} \tag{4.20}$$

The first result follows.

The proof of the second result uses the duality between points and lines, and the cross product to go between them. We start with the first result and merely identify cameras 3 and 2. We calculate by setting $\mathbf{T}_3 = \mathbf{T}_2$ and $B_3 = B_2$.

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}} \\ \boldsymbol{\ell}_2^{\mathrm{T}} \\ \boldsymbol{\ell}_3^{\mathrm{T}} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1 \\ \boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 \\ \boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3 \end{bmatrix} \tag{4.21}$$

Setting $\mathbf{T}_3 = \mathbf{T}_2$ and separating terms

$$= \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}} \\ \boldsymbol{\ell}_2^{\mathrm{T}} \\ \boldsymbol{\ell}_3^{\mathrm{T}} \end{bmatrix}^{-1} \left(\begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}}(\mathbf{T}_1 - \mathbf{T}_2) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_2 \\ \boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 \\ \boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_2 \end{bmatrix} \right) \tag{4.22}$$

Multiplying through and making the inverse explicit

$$= \frac{1}{|\boldsymbol{\ell}_1 \; \boldsymbol{\ell}_2 \; \boldsymbol{\ell}_3|} \begin{bmatrix} \boldsymbol{\ell}_2 \times \boldsymbol{\ell}_3 & \boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1 & \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\ell}_1^{\mathrm{T}}(\mathbf{T}_1 - \mathbf{T}_2) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} + \mathbf{T}_2 \tag{4.23}$$

Using $\mathbf{p}_2 = \boldsymbol{\ell}_2 \times \boldsymbol{\ell}_3$

$$= \frac{1}{\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2} \mathbf{p}_2 \boldsymbol{\ell}_1^{\mathrm{T}}(\mathbf{T}_1 - \mathbf{T}_2) + \mathbf{T}_2 \tag{4.24}$$

□

The second result is essentially the same as the first, except with cameras 2 and 3 considered as identical. Because we can consider a point as the cross product of two lines, we get an equation in terms of a point (which is can be considered the intersection of two lines) and another line. We use this principle throughout this paper by really only proving the constraint for lines, and then collapsing pairs of cameras in order to obtain constraints on points.

This result is not an essential result, and has been shown only to demonstrate the principle of camera collapse, which we elaborate on later in this paper. Let us point out that in most cases, we will not have isolated points which we must reconstruct, since most points are located as the intersection of lines. Even if we have isolated points, if there are at least three, then we may as well consider the lines joining them rather than the points themselves. We choose to operate with the lines rather than their intersection, so we will not use the point reconstruction later in the paper.

4.3 Reconstructing a Textured Plane

This exhausts the possibilities of reconstruction from arbitrary cameras using arbitrary points and lines, but we still have yet to reconstruct the textured planes. We now turn to the reconstruction of a textured plane from image lines in four cameras. This reconstruction is non-intuitive in a sense because we do not require that the cameras be looking at the same lines. Each of the four cameras can look at a different line. We only require that we know which line has been imaged, that is, its index n. Given these four lines we can reconstruct a textured plane, as in figure 4.1, with the following *multilinear equation*.

Fact 14 *If we have a textured plane* $\mathbf{H}$ *which is imaged by four cameras into image lines* $\hat{\boldsymbol{\ell}}_i$*, and we know that our cameras have parameters* $(B_i, \mathbf{T}_i)$*, and further, we know that*

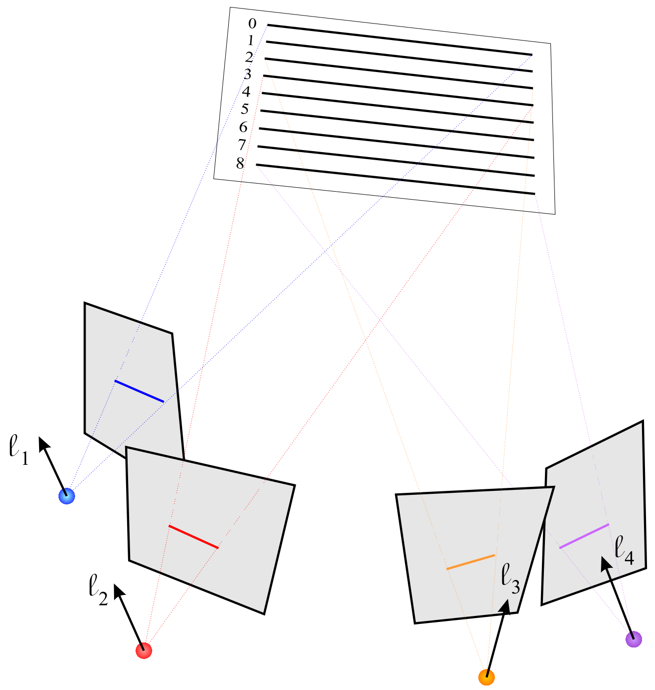

Figure 4.1: Reconstructing a Textured Plane

the image lines have indices n_i, then we may reconstruct the textured plane as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix} = \sum_{[i_1\ i_2\ i_3\ i_4] \in \text{perm}^+(1\ 2\ 3\ 4)} \begin{bmatrix} n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}| (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}) \\ 2 n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}| \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \\ 2 n_{i_1} |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \boldsymbol{\ell}_{i_1} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \end{bmatrix} \tag{4.25}$$

Note that $|\cdot|$ is the signed *magnitude, and since the coordinates are homogeneous, it does not matter which sign is chosen. The same result could be obtained by defining*

$$|\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j| = |\boldsymbol{\ell}_j\ \mathbf{v}\ \boldsymbol{\ell}_i| \tag{4.26}$$

where $\mathbf{v}$ is any arbitrary vector not in the plane of $\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j$.

Proof. We start by finding an expression for $\mathbf{L}_d$. Since $\mathbf{H}$ is homogeneous, we may treat the following as the actual $\mathbf{L}_d$, and multiply through later to obtain the expression in the above fact. In order to reconstruct the $\mathbf{L}_d$, we choose any pair of image lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ which have a non-zero cross product. Note that the cross product between any two image lines will have the same direction, but different magnitude.

$$\mathbf{L}_d = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \tag{4.27}$$

Now let us find the $\mathbf{L}_m$ and $\mathbf{L}_\lambda$ associated with our H. We can form four equations from fact 11:

$$\delta_i \boldsymbol{\ell}_i = \mathbf{L}_m - \mathbf{T}_i \times \mathbf{L}_d + \mathbf{L}_\lambda n_i \qquad 1 \leq i \leq 4 \tag{4.28}$$

In order to convert these equations to more manageable scalar equations, we define

$$\mathbf{r}_i = \mathbf{L}_d \times \boldsymbol{\ell}_i \tag{4.29}$$

so that $\mathbf{r}_i$ is perpendicular to $\boldsymbol{\ell}_i$, yielding

$$\mathbf{r}_i^{\mathrm{T}} \boldsymbol{\ell}_i = 0 \tag{4.30}$$

We multiply the four equations (4.28) through by the $\mathbf{r}_i$ to obtain the scalar equations:

$$0 = \mathbf{r}_i^{\mathrm{T}} \mathbf{L}_m - \mathbf{r}_i^{\mathrm{T}} (\mathbf{T}_i \times \mathbf{L}_d) + \mathbf{L}_\lambda n_i \qquad 1 \leq i \leq 4 \tag{4.31}$$

We wish to solve for the $\mathbf{L}_\lambda$ and $\mathbf{L}_m$. It would seem that we do not have enough information, since the $\mathbf{L}_\lambda$ and $\mathbf{L}_m$ have six coordinates, but we have only four equations. However, we know that both $\mathbf{L}_\lambda$ and $\mathbf{L}_m$ must be perpendicular to $\mathbf{L}_d$, which we already know, so that we really only have four coordinates to find.

We know that the $\boldsymbol{\ell}_i$, $\mathbf{L}_m$, and $\mathbf{L}_\lambda$ are all perpendicular to $\mathbf{L}_d$. For this reason, we may express any of these vectors in terms of two of the other ones. Let us derive the formula for expressing $\boldsymbol{\ell}_i$ in terms of a linear combination of $\boldsymbol{\ell}_j$ and $\boldsymbol{\ell}_k$.

We would like α and β so that:

$$\begin{bmatrix} \boldsymbol{\ell}_j & \boldsymbol{\ell}_k & \mathbf{L}_d \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \boldsymbol{\ell}_i \tag{4.32}$$

We thus have that

$$\begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_j & \boldsymbol{\ell}_k & \mathbf{L}_d \end{bmatrix}^{-1} \boldsymbol{\ell}_i \tag{4.33}$$

Using fact 8, we find that

$$\alpha = \frac{|\boldsymbol{\ell}_k \; \mathbf{L}_d \; \boldsymbol{\ell}_i|}{|\boldsymbol{\ell}_k \; \mathbf{L}_d \; \boldsymbol{\ell}_j|} \tag{4.34}$$

$$\beta = \frac{|\boldsymbol{\ell}_i \; \mathbf{L}_d \; \boldsymbol{\ell}_j|}{|\boldsymbol{\ell}_k \; \mathbf{L}_d \; \boldsymbol{\ell}_j|} \tag{4.35}$$

For convenience, we define the notation

$$\sigma_n^m = |\boldsymbol{\ell}_m \; \mathbf{L}_d \; \boldsymbol{\ell}_n| \tag{4.36}$$

And we get the following expression for $\boldsymbol{\ell}_i$:

$$\boldsymbol{\ell}_i = \boldsymbol{\ell}_j \frac{\sigma_i^k}{\sigma_j^k} + \boldsymbol{\ell}_k \frac{\sigma_j^i}{\sigma_j^k} \tag{4.37}$$

It is important to note that this equation only holds because these vectors are all in the plane perpendicular to $\mathbf{L}_d$. Another identity that we will use is that

$$\mathbf{r}_j^{\mathrm{T}} \boldsymbol{\ell}_k = \sigma_j^k \tag{4.38}$$

which follows from the definitions of $\mathbf{r}_i$ and σ_j^k. Also, it is easy to see that:

$$\sigma_j^i = -\sigma_i^j \tag{4.39}$$

And we may also derive that

$$\sigma_j^i \sigma_k^h - \sigma_k^i \sigma_j^h = \sigma_h^i \sigma_k^j \tag{4.40}$$

We also note that since $\mathbf{r}_i = \mathbf{L}_d \times \boldsymbol{\ell}_i$, we can derive that

$$\mathbf{r}_i^{\mathrm{T}}(\mathbf{T}_i \times \mathbf{L}_d) = -|\mathbf{L}_d|^2 \mathbf{T}_i \boldsymbol{\ell}_i \tag{4.41}$$

Since $\mathbf{L}_m$ and $\mathbf{L}_\lambda$ lie in the same plane as the $\boldsymbol{\ell}_i$, we can form for the sake of derivation that:

$$\mathbf{L}_m = a\boldsymbol{\ell}_1 + b\boldsymbol{\ell}_2 \tag{4.42}$$

$$\mathbf{L}_\lambda = c\boldsymbol{\ell}_1 + d\boldsymbol{\ell}_2 \tag{4.43}$$

Using these expressions, we may substitute into (4.31) to obtain:

$$0 = b\mathbf{r}_1^{\mathrm{T}}\boldsymbol{\ell}_2 - \mathbf{r}_1^{\mathrm{T}}(\mathbf{T}_1 \times \mathbf{L}_d) + d\mathbf{r}_1^{\mathrm{T}}\boldsymbol{\ell}_2 n_1 \tag{4.44}$$

$$0 = a\mathbf{r}_2^{\mathrm{T}}\boldsymbol{\ell}_1 - \mathbf{r}_2^{\mathrm{T}}(\mathbf{T}_2 \times \mathbf{L}_d) + c\mathbf{r}_2^{\mathrm{T}}\boldsymbol{\ell}_1 n_2 \tag{4.45}$$

$$0 = \mathbf{r}_3^{\mathrm{T}}\boldsymbol{\ell}_4 \frac{(a\sigma_1^3 + b\sigma_2^3)}{\sigma_4^3} - \mathbf{r}_3^{\mathrm{T}}(\mathbf{T}_3 \times \mathbf{L}_d) + \mathbf{r}_3^{\mathrm{T}}\boldsymbol{\ell}_4 n_3 \frac{(c\sigma_1^3 + d\sigma_2^3)}{\sigma_4^3} \tag{4.46}$$

$$0 = \mathbf{r}_4^{\mathrm{T}}\boldsymbol{\ell}_3 \frac{(a\sigma_4^1 + b\sigma_4^2)}{\sigma_4^3} - \mathbf{r}_4^{\mathrm{T}}(\mathbf{T}_4 \times \mathbf{L}_d) + \mathbf{r}_4^{\mathrm{T}}\boldsymbol{\ell}_3 n_4 \frac{(c\sigma_4^1 + d\sigma_4^2)}{\sigma_4^3} \tag{4.47}$$

We may rewrite the above equation, then, as

$$0 = b\sigma_1^2 + |\mathbf{L}_d|^2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 + d\sigma_1^2 n_1 \tag{4.48}$$

$$0 = a\sigma_2^1 + |\mathbf{L}_d|^2 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 + c\sigma_2^1 n_2 \tag{4.49}$$

$$0 = a\sigma_3^1 + b\sigma_3^2 + |\mathbf{L}_d|^2 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 + n_3(c\sigma_3^1 + d\sigma_3^2) \tag{4.50}$$

$$0 = a\sigma_4^1 + b\sigma_4^2 + |\mathbf{L}_d|^2 \mathbf{T}_4^{\mathrm{T}} \boldsymbol{\ell}_4 + n_4(c\sigma_4^1 + d\sigma_4^2) \tag{4.51}$$

And since $|\mathbf{L}_d|^2 = \sigma_1^2$, we may further rewrite these equations as:

$$0 = b + \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 + dn_1 \tag{4.52}$$

$$0 = a - \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 + cn_2 \tag{4.53}$$

$$0 = a\sigma_3^1 + b\sigma_3^2 + \sigma_1^2 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 + n_3(c\sigma_3^1 + d\sigma_3^2) \tag{4.54}$$

$$0 = a\sigma_4^1 + b\sigma_4^2 + \sigma_1^2 \mathbf{T}_4^{\mathrm{T}} \boldsymbol{\ell}_4 + n_4(c\sigma_4^1 + d\sigma_4^2) \tag{4.55}$$

We would like to solve for the c and d so that we can find $\mathbf{L}_\lambda$. We solve for a and b in terms of c and d as follows:

$$a = \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - cn_2 \tag{4.56}$$

$$b = -\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 - dn_1 \tag{4.57}$$

Substituting equations (4.56) and (4.57) into equation (4.54) we obtain:

$$0 = (\mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - cn_2)\sigma_3^1 + (-\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 - dn_1)\sigma_3^2 + \sigma_1^2 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 + n_3(c\sigma_3^1 + d\sigma_3^2) \tag{4.58}$$

and into equation (4.55) we obtain:

$$0 = (\mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - cn_2)\sigma_4^1 + (-\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 - dn_1)\sigma_4^2 + \sigma_1^2 \mathbf{T}_4^{\mathrm{T}} \boldsymbol{\ell}_4 + n_4(c\sigma_4^1 + d\sigma_4^2) \tag{4.59}$$

Collecting the c and d, we obtain:

$$0 \;=\; \sigma_3^1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 \;-\; \sigma_3^2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \;+\; \sigma_1^2 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 \;+\; c(n_3 \;-\; n_2)\sigma_3^1 \;+\; d(n_3 \;-\; n_1)\sigma_3^2 \tag{4.60}$$

and

$$0 \;=\; \sigma_4^1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 \;-\; \sigma_4^2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \;+\; \sigma_1^2 \mathbf{T}_4^{\mathrm{T}} \boldsymbol{\ell}_4 \;+\; c(n_4 \;-\; n_2)\sigma_4^1 \;+\; d(n_4 \;-\; n_1)\sigma_4^2 \tag{4.61}$$

We eliminate the d to obtain:

$$\begin{aligned}(n_4 - n_1)\sigma_4^2(\sigma_3^1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \sigma_3^2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 + \sigma_1^2 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3) + c(n_3 - n_2)(n_4 - n_1)\sigma_4^2\sigma_3^1 \\ = (n_3 - n_1)\sigma_3^2(\sigma_4^1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \sigma_4^2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 + \sigma_1^2 \mathbf{T}_4^{\mathrm{T}} \boldsymbol{\ell}_4) + c(n_4 - n_2)(n_3 - n_1)\sigma_3^2\sigma_4^1\end{aligned} \tag{4.62}$$

We eliminate the c to obtain:

$$\begin{aligned}(n_4 - n_2)\sigma_4^1(\sigma_3^1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \sigma_3^2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 + \sigma_1^2 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3) + d(n_3 - n_1)(n_4 - n_2)\sigma_4^1\sigma_3^2 \\ = (n_3 - n_2)\sigma_3^1(\sigma_4^1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \sigma_4^2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 + \sigma_1^2 \mathbf{T}_4^{\mathrm{T}} \boldsymbol{\ell}_4) + d(n_4 - n_1)(n_3 - n_2)\sigma_3^1\sigma_4^2\end{aligned} \tag{4.63}$$

If we set

$$\begin{aligned}k =& (n_4 - n_2)(n_3 - n_1)\sigma_3^2\sigma_4^1 - (n_4 - n_1)(n_3 - n_2)\sigma_3^1\sigma_4^2 && (4.64)\\ =& (n_1 n_2 + n_3 n_4)\sigma_2^1\sigma_4^3 && (4.65)\\ &+ (n_1 n_3 + n_4 n_2)\sigma_3^1\sigma_2^4 && (4.66)\\ &+ (n_1 n_4 + n_2 n_3)\sigma_4^1\sigma_2^3 && (4.67)\end{aligned}$$

And by doubling the terms and permuting the sigmas, we get

$$=\frac{1}{2}[(n_1n_2+n_3n_4)\sigma_2^1\sigma_4^3+(n_2n_1+n_4n_3)\sigma_1^2\sigma_3^4 \tag{4.68}$$

$$+(n_1n_3+n_4n_2)\sigma_3^1\sigma_2^4+(n_3n_1+n_2n_4)\sigma_1^3\sigma_4^2 \tag{4.69}$$

$$+(n_1n_4+n_2n_3)\sigma_4^1\sigma_2^3+(n_4n_1+n_3n_2)\sigma_1^4\sigma_3^2] \tag{4.70}$$

$$=\frac{1}{2}\sum_{[h\ i\ j\ k]\in \text{perm}^+(1234)} n_hn_i\sigma_i^h\sigma_k^j \tag{4.71}$$

With this definition of k, and noting that $\sigma_2^1=-|\mathbf{L}_d|^2$, we may now solve for c and d.

$$c=\frac{1}{k}[(n_4-n_1)\sigma_4^2(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{4.72}$$

$$-(n_3-n_1)\sigma_3^2(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)] \tag{4.73}$$

and

$$d=\frac{1}{k}[(n_3-n_2)\sigma_3^1(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.74}$$

$$-(n_4-n_2)\sigma_4^1(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)] \tag{4.75}$$

We know that $\mathbf{L}_\lambda=c\boldsymbol{\ell}_1+d\boldsymbol{\ell}_2$. Let us split up our equation as follows

$$\mathbf{L}_\lambda=n_1\mathbf{v}_1+n_2\mathbf{v}_2+n_3\mathbf{v}_3+n_4\mathbf{v}_4 \tag{4.76}$$

We calculate from the c and d that

$$\mathbf{v}_1=\frac{1}{k}\boldsymbol{\ell}_1(-\sigma_4^2\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2+\sigma_4^2\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1-\sigma_4^2\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 \tag{4.77}$$

$$+\sigma_3^2\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_3^2\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_3^2\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.78}$$

which, by equation (4.40), we can simplify to:

$$\mathbf{v}_1=\frac{\sigma_1^2}{k}\boldsymbol{\ell}_1(\sigma_4^3\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_4^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3+\sigma_3^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.79}$$

We then calculate that

$$\mathbf{v}_2 = \frac{1}{k}\boldsymbol{\ell}_2(-\sigma_3^1\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_3^1\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_3^1\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 \tag{4.80}$$

$$+\sigma_4^1\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^1\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_4^1\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{4.81}$$

we can simplify to:

$$\mathbf{v}_2 = \frac{\sigma_1^2}{k}\boldsymbol{\ell}_2(-\sigma_4^3\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_4^1\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 - \sigma_3^1\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.82}$$

We then calculate that

$$\mathbf{v}_3 = \frac{1}{k}[\boldsymbol{\ell}_1\sigma_3^2(-\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.83}$$

$$-\boldsymbol{\ell}_2\sigma_3^1(-\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4))] \tag{4.84}$$

Since $\boldsymbol{\ell}_1\sigma_3^2 - \boldsymbol{\ell}_2\sigma_3^1 = \sigma_1^2\boldsymbol{\ell}_3$, we can simplify this to

$$\mathbf{v}_3 = \frac{\sigma_1^2}{k}\boldsymbol{\ell}_3(+\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^1\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.85}$$

We then calculate that

$$\mathbf{v}_4 = \frac{1}{k}(\boldsymbol{\ell}_1\sigma_4^2(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{4.86}$$

$$-\boldsymbol{\ell}_2\sigma_4^1(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{4.87}$$

Since $\boldsymbol{\ell}_1\sigma_4^2 - \boldsymbol{\ell}_2\sigma_4^1 = \boldsymbol{\ell}_4$, we can simplify this to:

$$\mathbf{v}_4 = \frac{\sigma_1^2}{k}\boldsymbol{\ell}_4(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{4.88}$$

We thus have the following form for $\mathbf{L}_\lambda$

$$\mathbf{L}_\lambda = \frac{\sigma_1^2}{k}[n_1\boldsymbol{\ell}_1(\sigma_4^3\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^4\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + \sigma_3^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.89}$$

$$+n_2\boldsymbol{\ell}_2(\sigma_3^4\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_4^1\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + \sigma_1^3\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.90}$$

$$+n_3\boldsymbol{\ell}_3(\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^4\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^1\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{4.91}$$

$$+n_4\boldsymbol{\ell}_4(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^3\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)] \tag{4.92}$$

We may collect the $\mathbf{T}_i^{\mathrm{T}}\boldsymbol{\ell}_i$ to get another form for $\mathbf{L}_\lambda$.

$$\mathbf{L}_\lambda = \frac{\sigma_1^2}{k}[(n_2\sigma_3^4\boldsymbol{\ell}_2 + n_3\sigma_4^2\boldsymbol{\ell}_3 + n_4\sigma_2^3\boldsymbol{\ell}_4)\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 \tag{4.93}$$

$$(n_1\sigma_4^3\boldsymbol{\ell}_1 + n_3\sigma_1^4\boldsymbol{\ell}_3 + n_4\sigma_3^1\boldsymbol{\ell}_4)\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 \tag{4.94}$$

$$(n_1\sigma_2^4\boldsymbol{\ell}_1 + n_2\sigma_4^1\boldsymbol{\ell}_2 + n_4\sigma_1^2\boldsymbol{\ell}_4)\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 \tag{4.95}$$

$$(n_1\sigma_3^2\boldsymbol{\ell}_1 + n_2\sigma_1^3\boldsymbol{\ell}_2 + n_3\sigma_2^1\boldsymbol{\ell}_3)\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 \tag{4.96}$$

$$] \tag{4.97}$$

We can write this more compactly as a sum over all the positive permutations of $[1, 2, 3, 4]$:

$$\mathbf{L}_\lambda = \frac{\sigma_1^2}{k} \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1}\sigma_{i_3}^{i_2}\boldsymbol{\ell}_{i_1}\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \tag{4.98}$$

Now that we have $\mathbf{L}_\lambda$, we may solve for $\mathbf{L}_m$. Using equations (4.52), (4.53), and (4.42), we obtain the following formula for $\mathbf{L}_m$ in terms of c and d, which we know:

$$\mathbf{L}_m = \boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - n_2c\boldsymbol{\ell}_1 - n_1d\boldsymbol{\ell}_2 \tag{4.99}$$

We may substitute c and d into this equation to form:

$$\mathbf{L}_m = \frac{1}{k}[(n_4 - n_2)(n_3 - n_1)\sigma_3^2\sigma_4^1(\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1) \tag{4.100}$$

$$-(n_4 - n_1)(n_3 - n_2)\sigma_3^1\sigma_4^2(\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1) \tag{4.101}$$

$$-n_2(n_4 - n_1)\sigma_4^2(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)\boldsymbol{\ell}_1 \tag{4.102}$$

$$+n_2(n_3 - n_1)\sigma_3^2(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)\boldsymbol{\ell}_1 \tag{4.103}$$

$$-n_1(n_3 - n_2)\sigma_3^1(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)\boldsymbol{\ell}_2 \tag{4.104}$$

$$+n_1(n_4 - n_2)\sigma_4^1(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)\boldsymbol{\ell}_2] \tag{4.105}$$

We collect terms based on $n_i n_j \mathbf{T}_k^{\mathrm{T}} \boldsymbol{\ell}_k$ to obtain

$$\mathbf{L}_m = \frac{1}{k}\big[\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1[n_1 n_2(+\sigma_4^1\sigma_3^2\boldsymbol{\ell}_2 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 + \sigma_3^2\sigma_4^2\boldsymbol{\ell}_1 \tag{4.106}$$

$$-\sigma_4^2\sigma_3^2\boldsymbol{\ell}_1 + \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2) \tag{4.107}$$

$$+n_1 n_3(+\sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2) \tag{4.108}$$

$$+n_1 n_4(-\sigma_4^1\sigma_3^2\boldsymbol{\ell}_2 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2) \tag{4.109}$$

$$+n_2 n_3(-\sigma_3^2\sigma_4^2\boldsymbol{\ell}_1 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2) \tag{4.110}$$

$$+n_2 n_4(+\sigma_4^2\sigma_3^2\boldsymbol{\ell}_1 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2) \tag{4.111}$$

$$+n_3 n_4(+\sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2)] \tag{4.112}$$

$$+\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2[n_1 n_2(-\sigma_4^1\sigma_3^1\boldsymbol{\ell}_2 + \sigma_3^1\sigma_4^1\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1 \tag{4.113}$$

$$+\sigma_4^2\sigma_3^1\boldsymbol{\ell}_1 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_1 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1) \tag{4.114}$$

$$+n_1 n_3(-\sigma_3^1\sigma_4^1\boldsymbol{\ell}_2 + \sigma_3^1\sigma_4^2\boldsymbol{\ell}_1) \tag{4.115}$$

$$+n_1 n_4(+\sigma_4^1\sigma_3^1\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1) \tag{4.116}$$

$$+n_2 n_3(+\sigma_3^2\sigma_4^1\boldsymbol{\ell}_1 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1) \tag{4.117}$$

$$+n_2 n_4(-\sigma_4^2\sigma_3^1\boldsymbol{\ell}_1 + \sigma_3^1\sigma_4^2\boldsymbol{\ell}_1) \tag{4.118}$$

$$+n_3 n_4(-\sigma_3^1\sigma_4^2\boldsymbol{\ell}_1 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1)] \tag{4.119}$$

$$+\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3[n_1 n_2(-\sigma_4^1\sigma_1^2\boldsymbol{\ell}_2 + \sigma_4^2\sigma_1^2\boldsymbol{\ell}_1) \tag{4.120}$$

$$+n_1 n_4(+\sigma_4^1\sigma_1^2\boldsymbol{\ell}_2) + n_2 n_4(-\sigma_4^2\sigma_1^2\boldsymbol{\ell}_1)] \tag{4.121}$$

$$+\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4[n_1 n_2(+\sigma_3^1\sigma_1^2\boldsymbol{\ell}_2 - \sigma_3^2\sigma_1^2\boldsymbol{\ell}_1) \tag{4.122}$$

$$+n_1 n_3(-\sigma_3^1\sigma_1^2\boldsymbol{\ell}_2) + n_2 n_3(+\sigma_3^2\sigma_1^2\boldsymbol{\ell}_1)]\big] \tag{4.123}$$

We may use the identities in equations (4.40) and (4.37) to simplify this to:

$$\mathbf{L}_m = \frac{1}{k}[\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1(n_3 n_2 \sigma_2^3 \sigma_1^2 \boldsymbol{\ell}_4 + n_2 n_4 \sigma_4^2 \sigma_1^2 \boldsymbol{\ell}_3 + n_4 n_3 \sigma_3^4 \sigma_1^2 \boldsymbol{\ell}_2) \tag{4.124}$$

$$+\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2(n_1 n_3 \sigma_3^1 \sigma_1^2 \boldsymbol{\ell}_4 + n_4 n_1 \sigma_1^4 \sigma_1^2 \boldsymbol{\ell}_3 + n_3 n_4 \sigma_4^3 \sigma_1^2 \boldsymbol{\ell}_1) \tag{4.125}$$

$$+\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3(n_2 n_1 \sigma_1^2 \sigma_1^2 \boldsymbol{\ell}_4 + n_1 n_4 \sigma_4^1 \sigma_1^2 \boldsymbol{\ell}_2 + n_4 n_2 \sigma_2^4 \sigma_1^2 \boldsymbol{\ell}_1) \tag{4.126}$$

$$+\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4[n_1 n_2 \sigma_2^1 \sigma_1^2 \boldsymbol{\ell}_3 + n_3 n_1 \sigma_1^3 \sigma_1^2 \boldsymbol{\ell}_2 + n_2 n_3 \sigma_3^2 \sigma_1^2 \boldsymbol{\ell}_1)] \tag{4.127}$$

We may factor out the σ_1^2 to obtain

$$\mathbf{L}_m = \frac{\sigma_1^2}{k}[(n_3 n_2 \sigma_2^3 \boldsymbol{\ell}_4 + n_2 n_4 \sigma_4^2 \boldsymbol{\ell}_3 + n_4 n_3 \sigma_3^4 \boldsymbol{\ell}_2)\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 \tag{4.128}$$

$$+(n_1 n_3 \sigma_3^1 \boldsymbol{\ell}_4 + n_4 n_1 \sigma_1^4 \boldsymbol{\ell}_3 + n_3 n_4 \sigma_4^3 \boldsymbol{\ell}_1)\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 \tag{4.129}$$

$$+(n_2 n_1 \sigma_1^2 \boldsymbol{\ell}_4 + n_1 n_4 \sigma_4^1 \boldsymbol{\ell}_2 + n_4 n_2 \sigma_2^4 \boldsymbol{\ell}_1)\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 \tag{4.130}$$

$$+(n_1 n_2 \sigma_2^1 \boldsymbol{\ell}_3 + n_3 n_1 \sigma_1^3 \boldsymbol{\ell}_2 + n_2 n_3 \sigma_3^2 \boldsymbol{\ell}_1)\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 \tag{4.131}$$

We can write this more compactly as a sum over all the positive permutations of $[1, 2, 3, 4]$:

$$\mathbf{L}_m = \frac{\sigma_1^2}{k} \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} n_{i_2} \sigma_{i_2}^{i_1} \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \tag{4.132}$$

We have to this point used $\mathbf{L}_d = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$. Since our singly textured plane representation is homogeneous, we may multiply through by $\frac{k}{\sigma_1^2}$ to obtain:

$$\mathbf{L}_d = \frac{\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2}{2\sigma_1^2} \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} n_{i_2} \sigma_{i_2}^{i_1} \sigma_{i_4}^{i_3} \tag{4.133}$$

$$\mathbf{L}_m = \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} n_{i_2} \sigma_{i_2}^{i_1} \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \tag{4.134}$$

$$\mathbf{L}_\lambda = \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} \sigma_{i_3}^{i_2} \boldsymbol{\ell}_{i_1} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \tag{4.135}$$

Since the $\boldsymbol{\ell}_i$ are all in the same direction, we may see that

$$\frac{\sigma_{i_2}^{i_1}\sigma_{i_4}^{i_3}}{\sigma_1^2} = (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}) \tag{4.136}$$

We may also define our signed magnitude

$$|\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j| = \frac{\sigma_j^i}{|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2|} \tag{4.137}$$

Note that this is the *signed* magnitude, whose absolute value is equal to the magnitude of the cross product. By using the above relations and dividing through by $\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$, we obtain the desiderata. $\square$

Chapter 5

Constraints

At this point we know how to project lines and reconstruct them, and project singly textured planes and reconstruct them. We also have constraints on intersections of lines and coincidences of textured planes. We use the world constraints and the reconstructions to form a number of constraints *on the images*. Note, however, that the constraints fall out naturally from the constraints on the world.

5.1 Quadrilinear and Related Constraints

We start with the quadrilinear constraint, so named because it constrains the images of four lines in four cameras in a multilinear way in all of the cameras. Our quadrilinear constraint is based on the reconstruction and intersection of lines in space. If we have two lines $\mathbf{L}_1$ and $\mathbf{L}_2$, then the condition for them to intersect is shown in fact 5.

Fact 15 *If we have a world point* $\mathbf{P}$ *which projects to cameras* 1 *through* 4 *with parameters* $(B_i, \mathbf{T}_i)$*, and the image points are intersected by lines* $\hat{\boldsymbol{\ell}}_i$*, then if we set* $\boldsymbol{\ell} = {B_i}^{-\mathrm{T}}\boldsymbol{\ell}_i$*,* *the following holds:*

$$|\boldsymbol{\ell}_4\ \boldsymbol{\ell}_3\ \boldsymbol{\ell}_2|\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1 + |\boldsymbol{\ell}_3\ \boldsymbol{\ell}_4\ \boldsymbol{\ell}_1|\boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 + |\boldsymbol{\ell}_2\ \boldsymbol{\ell}_1\ \boldsymbol{\ell}_4|\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3 + |\boldsymbol{\ell}_1\ \boldsymbol{\ell}_2\ \boldsymbol{\ell}_3|\boldsymbol{\ell}_4^{\mathrm{T}}\mathbf{T}_4 = 0 \tag{5.1}$$

This is known as the **quadrilinear constraint***.*

Proof. Let us reconstruct the world lines from the images lines in the camera pairs $(1, 2)$ and $(3, 4)$. These are virtual world lines, not necessarily having any reality, but even so they can be expressed as:

$$\mathbf{L}_{1,2} = \begin{bmatrix} \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \\ \boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 \end{bmatrix} \tag{5.2}$$

$$\mathbf{L}_{3,4} = \begin{bmatrix} \boldsymbol{\ell}_3 \times \boldsymbol{\ell}_4 \\ \boldsymbol{\ell}_3\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 - \boldsymbol{\ell}_4\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 \end{bmatrix} \tag{5.3}$$

Both these lines must contain the world point $\mathbf{P}$. Therefore the lines must intersect. We may use the line intersection property from fact 5 to obtain immediately our constraint. □

We use the principle of camera collapse used earlier when deriving the point reconstruction formula in fact 13 for the following two constraint:

Fact 16 *If we have three cameras with parameters $(B_i, \mathbf{T}_i)$, and a world point P which projects to $\hat{\mathbf{p}}_i$, then if we have image lines $\hat{\boldsymbol{\ell}}_1$ and $\hat{\boldsymbol{\ell}}_3$ which intersect their respective image points, then if we make the usual calibration:*

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{p}_2 - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2 - (\mathbf{T}_2 \times \mathbf{p}_2)^{\mathrm{T}}(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0 \tag{5.4}$$

This is known as the **point trilinear constraint***.*

Proof. Let us derive the trilinear constraint. If we set $\mathbf{T}_4 = \mathbf{T}_2$ in (5.1), then we get

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_2^{\mathrm{T}}(\boldsymbol{\ell}_4\times\boldsymbol{\ell}_3) + \mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2\boldsymbol{\ell}_1^{\mathrm{T}}(\boldsymbol{\ell}_3\times\boldsymbol{\ell}_4) + \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_4^{\mathrm{T}}(\boldsymbol{\ell}_2\times\boldsymbol{\ell}_1) + \mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_4\boldsymbol{\ell}_3^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_2) = 0 \quad (5.5)$$

or by using equations (2.1) and (2.2)

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_3^{\mathrm{T}}(\boldsymbol{\ell}_2\times\boldsymbol{\ell}_4) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_1^{\mathrm{T}}(\boldsymbol{\ell}_2\times\boldsymbol{\ell}_4) + (\mathbf{T}_2\times(\boldsymbol{\ell}_4\times\boldsymbol{\ell}_2))^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3) = 0 \quad (5.6)$$

Note that whatever $\boldsymbol{\ell}_2$ and $\boldsymbol{\ell}_4$ we choose, we will end up with the same $\mathbf{p}_2 = \boldsymbol{\ell}_2\times\boldsymbol{\ell}_4$. We have shown our result. $\square$

Fact 17 *If we have a world point P projected onto two cameras with parameters $(B_i, \mathbf{T}_i)$ at image points $\hat{\mathbf{p}}_i$, then if we make the usual calibration assumption, we have the* **epipolar constraint***:*

$$(\mathbf{T}_1\times\mathbf{p}_1)^{\mathrm{T}}\mathbf{p}_2 + (\mathbf{T}_2\times\mathbf{p}_2)^{\mathrm{T}}\mathbf{p}_1 = 0 \quad (5.7)$$

Proof. If we set $\mathbf{T}_3 = \mathbf{T}_1$ in equation (5.4), we obtain:

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{p}_2 - \mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2 - (\mathbf{T}_2\times\mathbf{p}_2)^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3) = 0 \quad (5.8)$$

or by again using equation (2.1)

$$-(\mathbf{T}_1\times(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3))^{\mathrm{T}}\mathbf{p}_2 - (\mathbf{T}_2\times\mathbf{p}_2^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3) = 0 \quad (5.9)$$

Again, it does not matter which $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_3$ we choose, because we will end up with the same $\mathbf{p}_1 = \boldsymbol{\ell}_1\times\boldsymbol{\ell}_3$. $\square$

All these constraints are merely a consequence of intersecting two Plücker lines. However, we will prove later that the only Plücker intersection constraint that matters is the epipolar constraint.

5.2 Line Constraint

The above constraints were a result of considering lines which intersect in space. Now let us consider lines which *coincide* in space. For this, we need three cameras, so here is another trilinear constraint if we consider lines.

Fact 18 *If we have three cameras with parameters $(B_i, \mathbf{T}_i)$, and a world line L which projects to $\hat{\boldsymbol{\ell}}_i$, then if we have an image point $\hat{\mathbf{p}}_2$ which is on $\hat{\boldsymbol{\ell}}_2$ and we make the usual calibration/derotation:*

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{p}_2 - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2 - (\mathbf{T}_2 \times \mathbf{p}_2)^{\mathrm{T}}(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0 \tag{5.10}$$

which is the same equation, but with different measurements, and is known as the **line trilinear constraint**.

Proof. Using fact 12, we may reconstruct our line as:

$$\mathbf{L}_d = \boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1 \tag{5.11}$$

$$\mathbf{L}_m = \boldsymbol{\ell}_3\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \boldsymbol{\ell}_1\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 \tag{5.12}$$

If we project this line into the second camera, we obtain, from fact 9

$$\boldsymbol{\ell}_2 = \boldsymbol{\ell}_3\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \boldsymbol{\ell}_1\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 - \mathbf{T}_2 \times (\boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1) \tag{5.13}$$

Therefore, if we consider any point $\mathbf{p}_2$ on $\boldsymbol{\ell}_2$, we must have the constraint that:

$$\mathbf{p}_2^{\mathrm{T}}\boldsymbol{\ell}_3\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \mathbf{p}_2^{\mathrm{T}}\boldsymbol{\ell}_1\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 - (\mathbf{p}_2 \times \mathbf{T}_2)^{\mathrm{T}}(\boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1) = 0 \tag{5.14}$$

□

Note that, while this equation has the same equation as the point trilinear constraint, it operates on different objects. Indeed, I show that the only constraint that is relevant for points is the epipolar constraint, while the only constraint that matters for lines is the above line trilinear constraint. The quadrilinear and point trilinear are redundant with these two constraints if we consider three or more points.

5.3 Prismatic Line Constraint

There is, however, another constraint which does not depend on the intersection or coincidence of lines. This is the *prismatic line constraint*. Although this equation is a consequence of the line trilinear constraint, its full meaning is better realized when based on equation (4.1). Instead of trying to reconstruct the whole line, we just try to reconstruct $\mathbf{L}_d$. Given two image lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$, we may find $\mathbf{L}_d$ using equation (4.1). Note that we have made no reference to the position of either the line or the cameras. The constraint operates independently of any translation, whether of the cameras or of the line. In particular, the cameras may be separate or identical. Also, the image lines can have been created from a single line *or two parallel lines*. In the latter case, whether there is one or two cameras, we may reconstruct the direction. This is called shape from texture in the computer vision literature. So while the full reconstruction equation can be used to find particular lines, equation (4.3) can be used to find the direction of a set of parallel world lines, regardless of whether exact correspondence is known, as in figure 5.1.

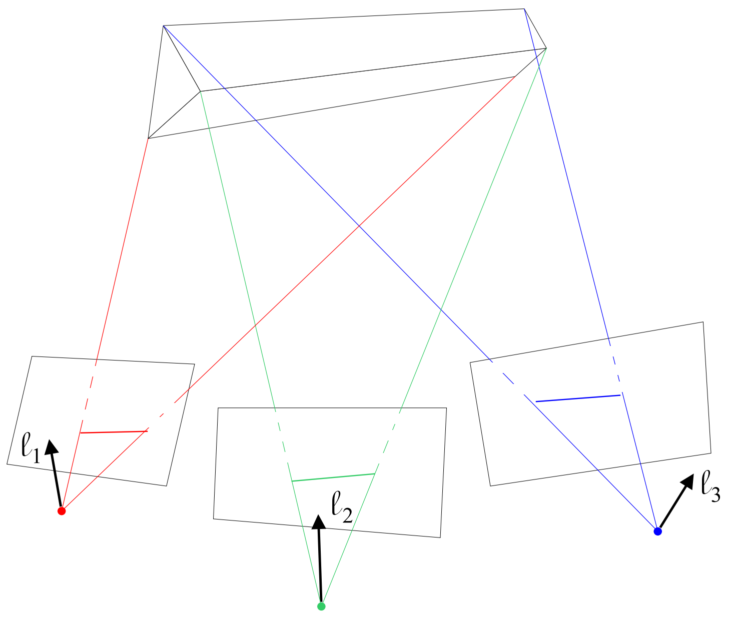

Figure 5.1: The Prismatic Line Constraint operates on *parallel* lines

In a similar fashion to the quadrilinear constraint, we may use the shape from texture equation (4.3) to constrain the motion. In particular, if we have three image lines, each of which are images of one of a set of parallel world lines, then we may reconstruct the direction $\mathbf{L}_d$ of the world lines with (4.3). From the construction of the Plücker lines, we know that any line with direction $\mathbf{L}_d$ must have moment vector perpendicular to $\mathbf{L}_d$. Putting these two facts together, we obtain

Fact 19 *If we have one, two, or three parallel world lines, and three cameras with rotation/calibration matrices B_i, then if these three cameras view images of one of our world lines as $\hat{\boldsymbol{\ell}}_i$, with the lines not necessarily the same in all cameras, then if we make the usual derotation/calibration, we obtain* **the prismatic line constraint**.

$$\boldsymbol{\ell}_2^{\mathrm{T}}(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0 \tag{5.15}$$

or in the uncalibrated sense we obtain a constraint on our rotation and calibration as:

$$\hat{\boldsymbol{\ell}}_2^{\mathrm{T}} B_2^{-1} (B_1{}^{-\mathrm{T}} \hat{\boldsymbol{\ell}}_1 \times B_3{}^{-\mathrm{T}} \hat{\boldsymbol{\ell}}_3) = 0 \tag{5.16}$$

We may use the same trick as we used earlier in making $B_3 = B_1$, which allows the case where both $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ are taken from the same camera. If these are different parallel lines, then we obtain the *vanishing point constraint*

$$(\hat{\boldsymbol{\ell}}_2)^{\mathrm{T}} B_3^{-1} B_1 (\hat{\boldsymbol{\ell}}_1 \times \hat{\boldsymbol{\ell}}_3) = 0 \tag{5.17}$$

The quantity $\boldsymbol{\ell}_1' \times \boldsymbol{\ell}_3'$ is called a vanishing point, and it is the point through which all images of world lines of direction $\mathbf{L}_d$ will pass.

Note that for the prismatic line constraint to constrain, all the $\boldsymbol{\ell}_i$ must be linearly independent, which means that the cameras may not all lie in one plane perpendicular to $\mathbf{L}_d$.

5.4 Equivalence of Constraints

While these derivations are geometrically correct, let us step back for a moment and look at the measurements of the images which we are able to obtain. From this, we discover that instead of considering the epipolar, trilinear, quadrilinear, and prismatic constraints, it is sufficient to consider the epipolar, prismatic, and a modified 2D trilinear constraint which only considers translation.

As we have formulated it, the line trilinear constraint operates on three image lines $\boldsymbol{\ell}_i$ in three cameras. If we choose *any* image point $\mathbf{p}$ which is incident on $\boldsymbol{\ell}_2$ ($\mathbf{p}^{\mathrm{T}} \boldsymbol{\ell}_2 = 0$), the equations hold. Note that although we may choose any incident point, we can only obtain two linearly independent equations, since the equation (5.4) is linear in all the lines and the space of all points incident on $\boldsymbol{\ell}_2$ is of rank 2. Further, if we choose two points which are linearly independent, then this accounts for all the constraints possible on these three lines.

The point trilinear constraint operates on three corresponding image points $\mathbf{p}_i$. The equation will still hold if we choose any two lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_3$ which are incident on $\mathbf{p}_1$ and $\mathbf{p}_3$, respectively, so that $\mathbf{p}_i^{\mathrm{T}}\boldsymbol{\ell}_i = 0$, for $i \in \{1, 3\}$. We see here that we can create at most four linearly independent equations, since we may choose two lines for each of two points. Again, if we choose two sets of two linearly independent lines, then these four equations account for all possible constraints on three points.

The corresponding point formulations of the trilinear constraint is equivalent to the corresponding line formulation plus the epipolar constraint. The line correspondence formulation has the advantage of being able to be split into the prismatic line constraint plus a 2D trilinear constraint. Therefore it is desirable to use the line trilinear constraint plus the epipolar constraint rather than using the point trilinear constraint.

Let us show that where there are three image points, the point trilinear constraint is equivalent to the line trilinear constraint plus the epipolar constraint. See figure 5.2 for a diagram.

Fact 20 *Given three world points $\mathbf{P}_i$ projected into three cameras j, with parameters $(B_j, \mathbf{T}_j)$, at $\hat{\mathbf{p}}_{j,i}$. The constraints on the positions of the cameras using the point trilin-*

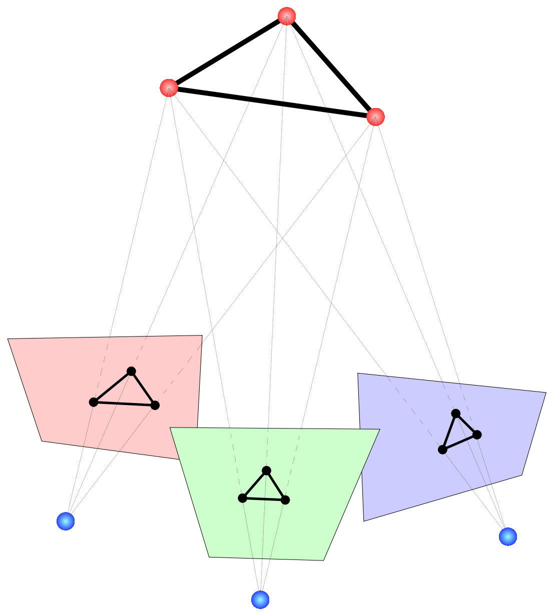

Figure 5.2: With three world points, an equivalence of points and lines is obtained

ear constraint are equivalent to the constraints on the cameras using the line trilinear constraint plus the epipolar constraint.

Proof. We work with the calibrated coordinates $\mathbf{p}_{j,i}$. Note that we may define image lines:

$$\boldsymbol{\ell}_{j,1} = \mathbf{p}_{j,2} \times \mathbf{p}_{j,3} \tag{5.18}$$

$$\boldsymbol{\ell}_{j,2} = \mathbf{p}_{j,3} \times \mathbf{p}_{j,1} \tag{5.19}$$

$$\boldsymbol{\ell}_{j,3} = \mathbf{p}_{j,1} \times \mathbf{p}_{j,2} \tag{5.20}$$

A consequence of this is that we also have

$$\mathbf{p}_{j,1} = \frac{\boldsymbol{\ell}_{j,2} \times \boldsymbol{\ell}_{j,3}}{|\mathbf{p}_{j,1}\ \mathbf{p}_{j,2}\ \mathbf{p}_{j,3}|} \tag{5.21}$$

$$\mathbf{p}_{j,2} = \frac{\boldsymbol{\ell}_{j,3} \times \boldsymbol{\ell}_{j,1}}{|\mathbf{p}_{j,1}\ \mathbf{p}_{j,2}\ \mathbf{p}_{j,3}|} \tag{5.22}$$

$$\mathbf{p}_{j,3} = \frac{\boldsymbol{\ell}_{j,1} \times \boldsymbol{\ell}_{j,2}}{|\mathbf{p}_{j,1}\ \mathbf{p}_{j,2}\ \mathbf{p}_{j,3}|} \tag{5.23}$$

We assume without loss of generality that $\mathbf{T}_2 = \mathbf{0}$ We may form our twelve trilinear equations by writing four for each point. For point 1 we get (by choosing lines 2 and 3 through point 1):

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) = 0 \tag{5.24}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) = 0 \tag{5.25}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) = 0 \tag{5.26}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{2,3}) = 0 \tag{5.27}$$

for point 2 (choosing lines 3 and 1), we get similarly

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) = 0 \tag{5.28}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) = 0 \tag{5.29}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) = 0 \tag{5.30}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{2,1}) = 0 \tag{5.31}$$

for point 3 (choosing lines 1 and 2), we get

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{5.32}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{5.33}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{5.34}$$

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{5.35}$$

Let us look closely at equations (5.31) and (5.32). We see that $(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1})$ and $(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2})$ are just points on line 1 in camera 2. So these two equations are equivalent to the line trilinear constraint using the $\boldsymbol{\ell}_{j,1}$. Similarly, equations (5.24) and (5.35) can be derived using $\boldsymbol{\ell}_{j,2}$ and equations (5.27) and (5.28) can be derived using $\boldsymbol{\ell}_{j,3}$. The remaining equations are equivalent to the epipolar constraints between pairs of points, in the following manner.

We may equate the $\mathbf{T}_3$ terms in equations (5.25) and (5.34) to obtain (switching some triple product orders also):

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{2,3}^{\mathrm{T}}(\boldsymbol{\ell}_{3,2}\times\boldsymbol{\ell}_{2,2})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{2,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{3,2})}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}}\mathbf{p}_{2,3}} \tag{5.36}$$

similarly, using equations (5.26) and (5.29):

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{2,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{3,3})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{2,1}^{\mathrm{T}}(\boldsymbol{\ell}_{3,3}\times\boldsymbol{\ell}_{2,3})}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}}\mathbf{p}_{2,2}} \tag{5.37}$$

finally, using equations (5.30) and (5.33):

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{2,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{3,1})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,2}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{2,2}^{\mathrm{T}}(\boldsymbol{\ell}_{3,1}\times\boldsymbol{\ell}_{2,1})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,3}} \tag{5.38}$$

We now note that since $\boldsymbol{\ell}_{3,i} \times \boldsymbol{\ell}_{2,i}$ gives the direction of line i, and so does $\boldsymbol{\ell}_{1,i} \times \boldsymbol{\ell}_{2,i}$, we know that these vectors have the same direction, but have different magnitudes. We may therefore substitute and divide out the magnitudes. We want equations in only cameras 1 and 2, so we thus may derive the following from the above equations:

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{2,3}^{\mathrm{T}}(\boldsymbol{\ell}_{1,2} \times \boldsymbol{\ell}_{2,2})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{2,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{1,2})}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}}\mathbf{p}_{2,3}} \tag{5.39}$$

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{2,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{1,3})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{2,1}^{\mathrm{T}}(\boldsymbol{\ell}_{1,3} \times \boldsymbol{\ell}_{2,3})}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}}\mathbf{p}_{2,2}} \tag{5.40}$$

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{2,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1} \times \boldsymbol{\ell}_{1,1})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,2}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{2,2}^{\mathrm{T}}(\boldsymbol{\ell}_{1,1} \times \boldsymbol{\ell}_{2,1})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,3}} \tag{5.41}$$

By substituting points for cross products of lines, we may change equations (5.39) and (5.40) to:

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,1}}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,3}}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}}\mathbf{p}_{2,3}} \tag{5.42}$$

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,1}}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,2}}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}}\mathbf{p}_{2,2}} \tag{5.43}$$

By multiplying through the left denominators, subtracting the above equations, and using equation 2.1 we may obtain:

$$(\mathbf{T}_1 \times (\boldsymbol{\ell}_{1,3} \times \boldsymbol{\ell}_{1,2}))^{\mathrm{T}}\mathbf{p}_{2,1} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,3}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,1}}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}}\mathbf{p}_{2,3}} - \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,2}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,1}}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}}\mathbf{p}_{2,2}} \tag{5.44}$$

$$= \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}}\mathbf{p}_{2,3}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}\mathbf{p}_{2,2}}(\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,3}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,1}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}\mathbf{p}_{2,2} - \boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,2}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,1}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}\mathbf{p}_{2,3}) \tag{5.45}$$

Now let us do a simple derivation. First, we know that we may write

$$\boldsymbol{\ell}_{1,1} = \lambda_1((\mathbf{P}_2 - \mathbf{T}_1)\times(\mathbf{P}_3 - \mathbf{T}_1)) \tag{5.46}$$

$$= \lambda_1(\mathbf{P}_2\times\mathbf{P}_3 + \mathbf{T}_1\times\mathbf{P}_2 + \mathbf{P}_3\times\mathbf{T}_1) \tag{5.47}$$

similarly, we get

$$\boldsymbol{\ell}_{1,2} = \lambda_2(\mathbf{P}_3\times\mathbf{P}_1 + \mathbf{T}_1\times\mathbf{P}_3 + \mathbf{P}_1\times\mathbf{T}_1) \tag{5.48}$$

and

$$\boldsymbol{\ell}_{1,3} = \lambda_3(\mathbf{P}_1\times\mathbf{P}_2 + \mathbf{T}_1\times\mathbf{P}_1 + \mathbf{P}_2\times\mathbf{T}_1) \tag{5.49}$$

where the λ_i are scale factors. Since $\mathbf{T}_2 = \mathbf{0}$, we may also write

$$\mathbf{p}_{2,i} = \gamma_i\mathbf{P}_i \tag{5.50}$$

Using these substitutions, we see that the parenthesized term in the RHS of equation (5.45) is:

$$\lambda_1\lambda_2\lambda_3\gamma_1\gamma_2\gamma_3(|\mathbf{P}_1\ \mathbf{T}_1\ \mathbf{P}_3||\mathbf{P}_2\ \mathbf{T}_1\ \mathbf{P}_1||\mathbf{P}_3\ \mathbf{T}_1\ \mathbf{P}_2| - |\mathbf{T}_1\ \mathbf{P}_1\ \mathbf{P}_2||\mathbf{T}_1\ \mathbf{P}_3\ \mathbf{P}_1||\mathbf{T}_1\ \mathbf{P}_2\ \mathbf{P}_3|) \tag{5.51}$$

and this is zero. So that we have derived that

$$(\mathbf{T}_1\times(\boldsymbol{\ell}_{1,3}\times\boldsymbol{\ell}_{1,2}))^{\mathrm{T}}\mathbf{p}_{2,1} = 0 \tag{5.52}$$

which is just the epipolar constraint for point 1 with cameras 1 and 2. If we subtract other pairs of equations (5.39) through (5.41), we may obtain the epipolar constraints for points 2 and 3. We find the epipolar equations between cameras 2 and 3 by equating the $\mathbf{T}_1$ terms instead of the $\mathbf{T}_3$ terms. $\square$

We now split the line trilinear constraint into the prismatic line constraint and a new 2D trilinear constraint.

Fact 21 *If we have three cameras with parameters* $(B_i, \mathbf{T}_i)$*, and a world line which projects to* $\hat{\boldsymbol{\ell}}_i$*, then the prismatic line constraint holds. There is only one other independent constraint, and it is:*

$$0 = \sum_{[i_1..i_3]\in \mathrm{P}^+[1..3]} (\mathbf{T}_{i_1})^{\mathrm{T}} \boldsymbol{\ell}_{i_1} |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \tag{5.53}$$

where $|\cdot|$ *is the signed magnitude.*

Proof. Recall the line trilinear constraint from fact 18:

$$\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \boldsymbol{\ell}_3^{\mathrm{T}} \mathbf{p}_2 - \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 \boldsymbol{\ell}_1^{\mathrm{T}} \mathbf{p}_2 0 (\mathbf{T}_2 \times \mathbf{p}_2)^{\mathrm{T}} (\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0 \tag{5.54}$$

Let us introduce the notation $\mathbf{Q} = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3$. We know that the prismatic line constraint holds, that is $|\boldsymbol{\ell}_1\ \boldsymbol{\ell}_2\ \boldsymbol{\ell}_3| = 0$. This is the same as $\mathbf{Q}^{\mathrm{T}} \boldsymbol{\ell}_2 = 0$. Since $\mathbf{Q}$ and $\boldsymbol{\ell}_2$ are perpendicular, we know that we may choose $\mathbf{p}_2 = \mathbf{Q} \times \boldsymbol{\ell}_2$. Using this definition, we may derive from equation (5.10) that

$$\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 |\boldsymbol{\ell}_3\ \mathbf{Q}\ \boldsymbol{\ell}_2| - \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 |\boldsymbol{\ell}_1\ \mathbf{Q}\ \boldsymbol{\ell}_2| - (\mathbf{T}_2 \times (\mathbf{Q} \times \boldsymbol{\ell}_2))^{\mathrm{T}} \mathbf{Q} = 0 \tag{5.55}$$

$$\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 |\boldsymbol{\ell}_3\ \mathbf{Q}\ \boldsymbol{\ell}_2| - \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 |\boldsymbol{\ell}_1\ \mathbf{Q}\ \boldsymbol{\ell}_2| - (\mathbf{Q} \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2)^{\mathrm{T}} \mathbf{Q} + (\boldsymbol{\ell}_2 \mathbf{T}_2^{\mathrm{T}} \mathbf{Q})^{\mathrm{T}} \mathbf{Q} = 0 \tag{5.56}$$

but since $\boldsymbol{\ell}_2^{\mathrm{T}} \mathbf{Q} = 0$, the last term is zero and we can derive

$$\mathbf{T}_1 \boldsymbol{\ell}_1 |\boldsymbol{\ell}_2\ \boldsymbol{\ell}_3\ \mathbf{Q}| + \mathbf{T}_2 \boldsymbol{\ell}_2 |\boldsymbol{\ell}_3\ \boldsymbol{\ell}_1\ \mathbf{Q}| + \mathbf{T}_3 \boldsymbol{\ell}_3 |\boldsymbol{\ell}_1\ \boldsymbol{\ell}_3\ \mathbf{Q}| \tag{5.57}$$

Since the prismatic line constraint holds, it is clear that $\mathbf{Q}$ is arbitrary, as long as it does not lie in the same plane as the $\boldsymbol{\ell}_i$. We therefore remove it and replace it with the signed magnitude $|\cdot|$ to obtain our result. $\square$

It has already been proved elsewhere [19] that the quadrilinear constraint is redundant with the trilinear constraints and the epipolar constraint. What we are left with is three constraints:

1. the epipolar constraint
2. the prismatic line constraint
3. the 2D trilinear constraint

The constraints all have different properties which can be used for different portions of structure from motion. The first constraint ensures that two points correspond. The second constraint is only on rotation and ensures that lines are properly aligned. The third constraint ensures that depths are consistent when calculated from different pairs of cameras.

5.5 Textured Plane Constraints

At this point we must discuss further the general framework. We propose to represent all textured planes by means of a linear combination of singly textured planes through the use of the Fourier transform, with a singly textured plane representing a single exponential. Therefore we will be concerned not only with the reconstruction of a singly textured

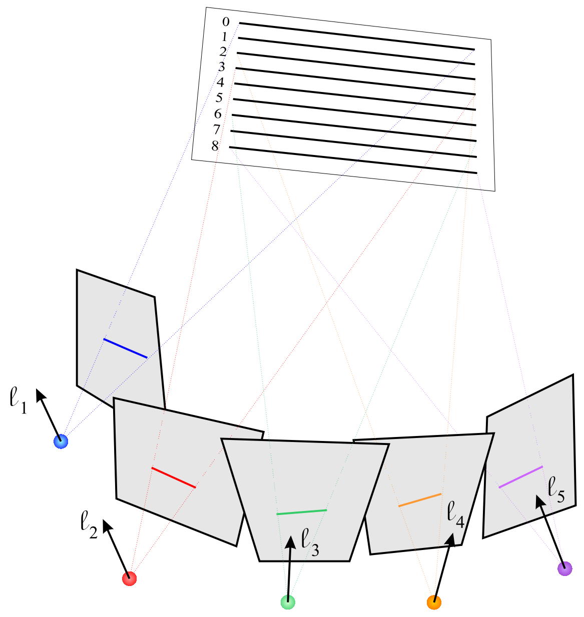

Figure 5.3: The Harmonic Trifocal Constraint operates on five image lines

plane, but with the constraint that two singly textured planes must be coincident, which condition is stated by fact 10.

It is as simple to form the texture constraints as it was to form the previous constraint. There is a line texture constraint a point texture constraint, and a mixed constraint. Keep in mind that all these constraints can be applied to fewer cameras by the principle of collapse used earlier to derive the point trilinear and epipolar constraint from the quadrilinear constraint.

The first is the five camera constraint, as shown in figure 5.3

Fact 22 *If we have five cameras* $(B_i, \mathbf{T}_i)$, *and measure five lines* $\hat{\boldsymbol{\ell}}_i$, *which have indices* n_i *from a textured plane* $\mathbf{H}$. *We may form the* $\boldsymbol{\ell}_i$ *using the* $\hat{\boldsymbol{\ell}}_i$ *and the* B_i *and have the* *following constraint:*

$$0 = \sum_{[i_1..i_5] \in \mathbf{P}^+[1..5]} n_{i_1} n_{i_2} (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}} (\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}) \mathbf{T}_{i_5}^{\mathrm{T}} \boldsymbol{\ell}_{i_5} \tag{5.58}$$

Where perm$^+$ denote the positive permutations.

Proof. Using fact 14, we may reconstruct the textured plane to obtain the parameters of the textured plane $\mathbf{H}$ using the lines one through four. Using this reconstruction, we can find the fifth image line as:

$$\boldsymbol{\ell}_5 = \mathbf{L}_m + n_5 \mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d \tag{5.59}$$

If $\mathbf{p}_5$ is a point on $\boldsymbol{\ell}_5$, we know that $\mathbf{p}_5$ is perpendicular to $\boldsymbol{\ell}_5$, so that $\mathbf{p}_5^{\mathrm{T}} \boldsymbol{\ell}_5 = 0$. We can use this with the above equation to formulate the constraint. Note that since $\mathbf{L}_d$ is perpendicular to $\boldsymbol{\ell}_5$ that $\mathbf{L}_d$ is a point on the line $\boldsymbol{\ell}_5$, but if we set $\mathbf{p} = \mathbf{L}_d$, all of the right hand side terms disappear and we have no constraint. Therefore we know that there is only one equation in our constraint, and we use $\mathbf{p}_5 = \mathbf{L}_d \times \boldsymbol{\ell}_5$. We can derive

$$0 = (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}} (\mathbf{L}_m + n_5 \mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d) \tag{5.60}$$

$$= |\mathbf{L}_d \ \boldsymbol{\ell}_5 \ \mathbf{L}_m| + n_5 |\mathbf{L}_d \ \boldsymbol{\ell}_5 \ \mathbf{L}_\lambda| - (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}} (\mathbf{T}_5 \times \mathbf{L}_d) \tag{5.61}$$

we use vector algebra and the fact that $\mathbf{L}_d^{\mathrm{T}} \boldsymbol{\ell}_5 = 0$ to obtain $-\mathbf{L}_d^{\mathrm{T}} \mathbf{L}_d \mathbf{T}^{\mathrm{T}} \boldsymbol{\ell}_5$ for the last term

$$= \sum_{[j_1..j_4] \in \mathbf{P}^+[1..4]} [2 n_{j_1} n_{j_2} (\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2})^{\mathrm{T}} (\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_3}) \mathbf{T}_{j_4}^{\mathrm{T}} \boldsymbol{\ell}_{j_4} \tag{5.62}$$

$$+ 2 n_{j_1} n_5 (\boldsymbol{\ell}_{j_2} \times \boldsymbol{\ell}_{j_3})^{\mathrm{T}} (\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_1}) \mathbf{T}_{j_4}^{\mathrm{T}} \boldsymbol{\ell}_{j_4} \tag{5.63}$$

$$+ n_{j_1} n_{j_2} (\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_{j_4})^{\mathrm{T}} (\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2}) \mathbf{T}_5^{\mathrm{T}} \boldsymbol{\ell}_5 \tag{5.64}$$

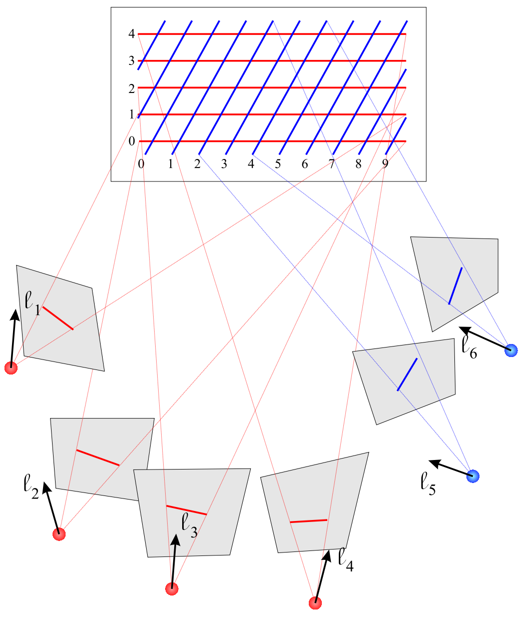

Figure 5.4: The Mixed Harmonic Constraint operates on six image lines

which we can expand to

$$= \sum_{[j_1..j_4]\in \mathrm{P}^+[1..4]} [n_{j_1} n_{j_2} (\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2})^{\mathrm{T}} (\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_3}) \mathbf{T}_{j_4}^{\mathrm{T}} \boldsymbol{\ell}_{j_4} \tag{5.65}$$

$$+ n_{j_2} n_{j_1} (\boldsymbol{\ell}_{j_2} \times \boldsymbol{\ell}_{j_1})^{\mathrm{T}} (\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_5) \mathbf{T}_{j_4}^{\mathrm{T}} \boldsymbol{\ell}_{j_4} \tag{5.66}$$

$$+ n_5 n_{j_1} (\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_1})^{\mathrm{T}} (\boldsymbol{\ell}_{j_2} \times \boldsymbol{\ell}_{j_3}) \mathbf{T}_{j_4}^{\mathrm{T}} \boldsymbol{\ell}_{j_4} \tag{5.67}$$

$$+ n_{j_1} n_5 (\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_5)^{\mathrm{T}} (\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_{j_2}) \mathbf{T}_{j_4}^{\mathrm{T}} \boldsymbol{\ell}_{j_4} \tag{5.68}$$

$$+ n_{j_1} n_{j_2} (\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2})^{\mathrm{T}} (\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_{j_4}) \mathbf{T}_5^{\mathrm{T}} \boldsymbol{\ell}_5] \tag{5.69}$$

and this is equal to the desiderata. □

Next is the mixed constraint, which operates on six cameras, as in figure 5.4 figure 5.4

Fact 23 *If we have six cameras $(B_i, \mathbf{T}_i)$ which measure six lines $\hat{\boldsymbol{\ell}}_i$ on a doubly textured*

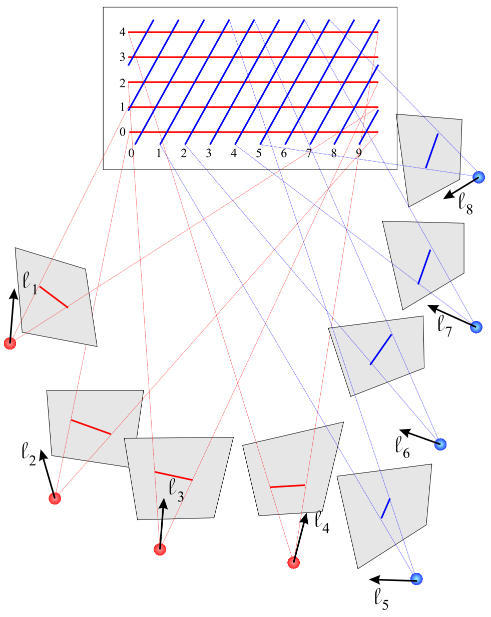

Figure 5.5: The Harmonic Epipolar Constraint operates on eight image lines

plane, and the first four lines measure one texture with indices n_i and the last four lines measure the other texture with unknown indices, then we may form the following constraint:

$$0 = \sum_{[i_1..i_4] \in \mathrm{P}^+[1..4]} n_{i_1} |\boldsymbol{\ell}_5 \; \boldsymbol{\ell}_6 \; \boldsymbol{\ell}_{i_1}| |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \mathbf{T}_{i_4} \boldsymbol{\ell}_{i_4} \tag{5.70}$$

Proof. We may reconstruct the $\mathbf{L}_{\lambda,1}$ of the singly textured plane from the first four cameras. We may reconstruct the $\mathbf{L}_{d,2}$ of the world line using the last two cameras. Using fact 10 and 14 we may easily obtain the equation. □

Last is the harmonic epipolar constraint, which operates on eight cameras, as in figure 5.5

Fact 24 *If we have eight cameras $(B_i, \mathbf{T}_i)$ which measure eight lines $\hat{\boldsymbol{\ell}}_i$ on a doubly textured plane, and the first four lines measure one texture with indices n_i $i \in [1..4]$ and* *the last four lines measure the other texture with indices* $n_i\ i \in [5..8]$*, then we may form the following constraint:*

$$0 = \sum_{[i_1..i_8] \in \mathrm{sP}^+} n_{i_1} n_{i_2} n_{i_5} n_{i_6} |\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}| \cdot |\boldsymbol{\ell}_{i_5} \times \boldsymbol{\ell}_{i_6}| \cdot |\boldsymbol{\ell}_{i_1}\ \boldsymbol{\ell}_{i_2}\ \boldsymbol{\ell}_{i_7}| \boldsymbol{\ell}_{i_8}^{\mathrm{T}} \mathbf{T}_{i_8} \tag{5.71}$$

where sP^+ *indicates the positive permutations among the first four and the last four indices, plus switching the first and last four sets of indices wholesale.*

Proof. This is a simple application of the first constraint of fact 10. □

This constitutes all the geometric constraints on the textured planes.

Chapter 6

Making the Constraints Useful

We have introduced various multilinear constraints without much explanation. As formulated, these constraints operate on five, six, and eight cameras. This is because this was the simplest way to construct the constraints. We show in this chapter how to actually apply these constraints to a real situation. In section 6.1 we show how to put an arbitrary number of any variety of constraints together into one equation. In section 6.2 we show how to put multiple measurements into one camera, extending the results from the section 6.1. Finally, in section 6.3, we make explicit the connection between these multilinear constraints and wavelengths as measured in the images.

6.1 Constraints on Arbitrary Numbers of Cameras

The constraints in the previous section operate on a fixed number of cameras. In this section we show how to integrate any number of these constraints with any number of cameras. In the following we assume that we have the rotational calibration for all the cameras because the prismatic line constraint gives us such a result. However, even if we did not use this constraint, we could still find the rotation as an eigenvalue minimization.

Let us assume that we have M cameras and that we want to extend our constraints to find the $\mathbf{T}_i$ considering all of the cameras and not just the fixed number which happen to appear in the equation. Note that all our constraints (except for the prismatic) are of the form

$$\sum_{i=1}^{n} c_i \boldsymbol{\ell}_i^{\mathrm{T}} \mathbf{T}_i = 0 \tag{6.1}$$

where n is the number of cameras. Let us also assume that $n < M$. We may form an equation for every choice $\{k_1..k_n\}$ of n numbers from the set $\{1..M\}$. Let us now set

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_i \\ \vdots \\ \mathbf{T}_M \end{bmatrix} \qquad \mathbf{T}' = \begin{bmatrix} \mathbf{T}_2 \\ \vdots \\ \mathbf{T}_i \\ \vdots \\ \mathbf{T}_M \end{bmatrix} \tag{6.2}$$

Each of our constraints is a linear equation over the $\boldsymbol{\ell}_{k_i}$, $\mathbf{T}_{k_i}$, and possibly the n_{k_i}. We can form this into a linear equation:

$$\sum_{i=1}^{n} f_i(\boldsymbol{\ell}_{k_1}, .., \boldsymbol{\ell}_{k_n}, n_{k_1}, .., n_{k_n}) \boldsymbol{\ell}_{k_i}^{\mathrm{T}} \mathbf{T}_{k_i} \tag{6.3}$$

With each choice of n lines from the M lines, we obtain another constraint, so we get $\binom{M}{n}$ equations, which we put into matrix form:

$$\begin{bmatrix} & & \vdots & & \\ \cdots & f_1(\boldsymbol{\ell}_{k_1},..,\boldsymbol{\ell}_{k_n},n_{k_1},..,n_{k_n})\boldsymbol{\ell}_{k_1}^{\mathrm{T}} & \cdots & f_n(\boldsymbol{\ell}_{k_1},..,\boldsymbol{\ell}_{k_n},n_{k_1},..,n_{k_n})\boldsymbol{\ell}_{k_n}^{\mathrm{T}} & \cdots \\ & & \vdots & & \end{bmatrix} \mathbf{T} = 0 \tag{6.4}$$

where the terms k_i are always in the k_i column. Let us call this matrix A. A has $\binom{M}{n}$ rows and n columns. Let us form the matrix $C = A^{\mathrm{T}}A$. It is easy to see that $\mathbf{T}$ must be contained in the subspace spanned by the eigenvectors corresponding to the zero eigenvalues (we call these the zero eigenvectors). For every corresponded object j (point, line, singly and doubly textured planes), we get a matrix C_j from the A_j matrix. If we add up all the matrices C_j

$$Q' = \sum_j C_j \tag{6.5}$$

then if there is no noise, then the zero eigenvectors will span a subspace which contains $\mathbf{T}$. Because the problem is translation invariant, we have three extra degrees of freedom, so that our matrix Q' will have rank $M-4$ in the general case. If we set $\mathbf{T}_1 = \mathbf{0}$, then we fix the location of the first camera to be at the origin of the fiducial coordinate system. If we form Q to be the bottom right $3(M-1)$ square matrix of Q', then we know that

$$(\mathbf{T}')^{\mathrm{T}} Q \mathbf{T}' = 0 \tag{6.6}$$

That is, the zero eigenvectors of Q give possible solutions for our $\mathbf{T}_i$. We thus have a method for integrating any of our constraints over any number of cameras to obtain the translation.

6.2 Camera Collapse

We can extend this treatment of arbitrary numbers of cameras to account for circumstances where there are multiple image lines in a single camera. Thus we could use the harmonic epipolar constraint with two cameras. When we form our matrix Q', let us consider each image line to be formed from a different camera. Let us consider that we have M cameras and N image lines. Let us form the $3N \times 3M$ matrix C, which we think of as composed of 3 by 3 blocks. Each block contains either zeros or an identity matrix I_3. An identity matrix is contained in the i^{th} block down and j^{th} block across if image line i belongs to camera j. Now we can form a new matrix S' which is

$$S' = C^{\mathrm{T}} Q' C \tag{6.7}$$

From this, as before, we can take the bottom right $3(M-1)$ square matrix to form S. We thus have again that:

$$(\mathbf{T}')^{\mathrm{T}} S \mathbf{T}' = 0 \tag{6.8}$$

but now our image lines need not correspond to our camera positions.

6.3 Harmonic Constraints

We have shown the multilinear constraints in the most general sense, which is the simplest and most symmetric. Unfortunately, when the equations are in this form the link to textures is not very apparent. By substituting into the equations, we can see how to make the textures integral to the equations, and thus worth of the name "harmonic". Remember

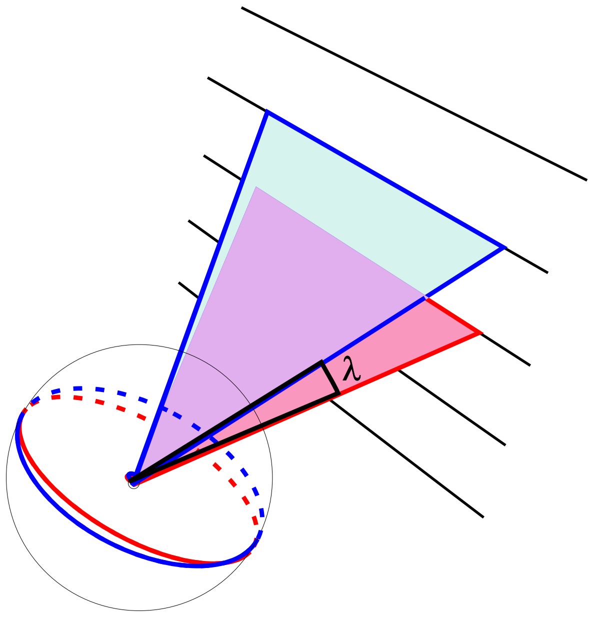

Figure 6.1: Wavelength as an Angle

that we speak of textures as being decomposed into their harmonic frequencies. We may think of the peaks in the wavefronts as being parallel lines in the world. The projection of a set of parallel lines is of course not a set of parallel lines, so it does not make sense to think of wavelengths in terms of perpendicular length between wavefronts. Instead we consider a wavelength as the angle between the great circles on two successive wavefronts, as in figure 6.1. This wavelength is a local measurement, in that the angle between lines i and $i+1$ will not in general equal that between j and $j+1$.

When we make a measurement of a texture in an image at a point $\mathbf{r}$, we obtain a direction, which we represent by $\boldsymbol{\ell}$, a line along the isophote. We also have two angles α and β, with $\lambda = \alpha + \beta$, if λ is the wavelength of the measured texture. α is the angle from the measurement $\mathbf{r}$ to the previous line (wave peak). β is the angle from the measurement

$\mathbf{r}$ to the next line (wave peak).

Since we are considering textures, we assume that this angle is small, so that $\sin(\lambda) \simeq \lambda$ and $\lambda_1\lambda_2 \simeq 0$. Using these coordinates and assumptions, we may write equations in which we specify the coordinates of a textured plane given a texture measurement in each of two cameras.

In fact 14, we determined the following equation for a textured plane:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_d \\ \mathbf{H}_m \\ \mathbf{H}_\lambda \end{bmatrix} = \sum_{[i_1\ i_2\ i_3\ i_4] \in \mathrm{perm}^+(1\ 2\ 3\ 4)} \begin{bmatrix} n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}| (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}) \\ 2 n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}| \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \\ 2 n_{i_1} |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \boldsymbol{\ell}_{i_1} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \end{bmatrix}$$

The equation in fact 14 is in terms of cross products, but since all these cross products lie in the same direction (that of the parallel world lines $\hat{\mathbf{d}}$), we can equally well think of them as sines of the angles between the lines. We may then rewrite the above equation as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_d \\ \mathbf{H}_m \\ \mathbf{H}_\lambda \end{bmatrix} = \sum_{[i_1\ i_2\ i_3\ i_4] \in \mathrm{perm}^+(1\ 2\ 3\ 4)} \begin{bmatrix} n_{i_1} n_{i_2} \sin_{i_3,i_4} \sin_{i_1,i_2} \hat{\mathbf{d}} \\ 2 n_{i_1} n_{i_2} \sin_{i_1,i_2} \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \\ 2 n_{i_1} \sin_{i_2,i_3} \boldsymbol{\ell}_{i_1} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \end{bmatrix}$$

where $\hat{\mathbf{d}}$ is the unit direction of the parallel lines. The subscript of the sines indicates which lines are being taken sines of.

Now the integer indices of the lines are restricted to consecutive lines. Since we are talking about the frequency of the texture, we can use n, and $n+1$ instead of i_1 and i_2, and m and $m+1$ instead of i_3 and i_4.

When we actually measure an image at a particular orientation and wavelength, we obtain a magnitude and a phase. We ignore the magnitude in this analysis and just consider the phase, which will be encapsulated in our ϕ and θ. We write our textured plane reconstruction equation in terms of texture measurements in different locations (and possibly from different cameras) on the same patch. We must parametrize our measurements so they can be useful n our projective coordinate system. In camera i, we represent the orientation by a line $\boldsymbol{\ell}_i$ along the wavefront. $\boldsymbol{\ell}_i$ is not necessarily at the zero phase position of the filter output, but instead is located at $\mathbf{r}_i$, the measurement position. There are two possible directions in which $\boldsymbol{\ell}_i$ may point. It is important that the directions chosen in each camera correspond to the same direction on the world texture, because otherwise our $m + 1$ referred to earlier would actually be $m - 1$. We assume that these are chosen correctly.

So if $\mathbf{d}_i$ is the direction of the wave, the $\boldsymbol{\ell}_i = \mathbf{r}_i \times \mathbf{d}_i$. The wavelength information we represent as an angle λ_i between the plane intersecting two successive zero phase lines, as in figure 6.1.

We also need the angle to the first zero phase line we cross as we go in the direction $-\boldsymbol{\ell}_i$ from $\mathbf{r}_i$. We call this angle α_i. For convenience, we set $\beta_i = \lambda_i - \alpha_i$, the angle to the first zero phase line in the direction $\boldsymbol{\ell}_i$ from $\mathbf{r}_i$. We also let θ be the angle between $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$, so $\theta = \arccos(\boldsymbol{\ell}_1^{\mathrm{T}} \boldsymbol{\ell}_2)$.

Now if we consider the angle between lines 1 and 2 to be θ, the angle between lines i

and the next line in the texture to be λ_i, then we may also have the following substitutions:

$$\begin{aligned}
\sin_{1,2} &= \sin\lambda_1 \simeq \lambda_1 \\
\sin_{1,3} &= \sin(\theta + \alpha_1 - \alpha_2) \\
&= \sin\theta\cos(\alpha_1 - \alpha_2) + \sin(\alpha_1 - \alpha_2)\cos\theta \\
&\simeq \sin\theta + (\alpha_1 - \alpha_2)\cos\theta \\
\sin_{1,4} &= \sin(\theta + \alpha_1 + \beta_2) = \sin\theta\cos\lambda_2 + \cos\theta\sin\lambda_2 \\
&\simeq \sin\theta + (\alpha_1 + \beta_2)\cos\theta \\
\sin_{2,4} &= \sin(\theta - \beta_1 + \beta_2) \\
&\simeq \sin\theta\cos + (\beta_2 - \beta_1)\cos\theta \\
\sin_{2,3} &= \sin(\theta - \beta_1 - \alpha_2) \\
&\simeq \sin\theta - (\beta_1 + \alpha_2)\cos\theta \\
\sin_{3,4} &= \sin\lambda_2 \simeq \lambda_2
\end{aligned}$$

where we have used the identity that the sine of a small angle is equal to the angle and the cosine of a small angle is 1.

Now we would like to determine what $\mathbf{H}$ is using these substitutions. The first entry

$\mathbf{H}_d$ in our vector becomes:

$$
\begin{aligned}
=&2\hat{\mathbf{d}}nm \sin_{2,4}\sin_{3,1} + 2(n+1)(m+1)\sin_{3,1}\sin_{2,4} + 2(n+1)m\sin_{4,1}\sin_{3,2}\\
&+ 2n(m+1)\sin_{2,3}\sin_{1,4} + 2m(m+1)\sin_{1,2}\sin_{3,4} + 2n(n+1)\sin_{3,4}\sin_{1,2}\\
=&2\hat{\mathbf{d}}[(2nm+n+m+1)\sin_{2,4}\sin_{3,1}\\
&+ (2nm+n+m)\sin_{2,3}\sin_{1,4} + (m^2+m+n^2+n)\sin_{1,2}\sin_{3,4}]\\
=&2\hat{\mathbf{d}}[(2nm+n+m+1)(\sin\theta + (\lambda_2-\lambda_1)\cos\theta)(-\sin\theta) + (2nm)\\
&+ n + m(\sin\theta - \lambda_1\cos\theta)(\sin\theta + \lambda_2\cos\theta) + (m^2+m+n^2+n)\lambda_1\lambda_2
\end{aligned}
$$

If we let the $\lambda_1\lambda_2$ terms go to zero, we get

$$
=2\hat{\mathbf{d}}[-\sin^2\theta - (\lambda_2-\lambda_1)\sin\theta\cos\theta]
$$

We can simplify this by putting the cross products back into the equation to obtain:

$$
\mathbf{H}_d = 2(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2)(|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2| - (\lambda_2-\lambda_1)\boldsymbol{\ell}_1^{\mathrm{T}}\boldsymbol{\ell}_2)
$$

We now calculate what $\mathbf{H}_m$ is using the substitutions.

$$
\begin{aligned}
\mathbf{H}_m = n(n+1)\sin_{1,2}(\boldsymbol{\ell}_2 - \alpha_2\mathbf{r}_2)\mathbf{T}_2^{\mathrm{T}}(\boldsymbol{\ell}_2 + \beta_2\mathbf{r}_2) \\
-n(n+1)\sin_{1,2}(\boldsymbol{\ell}_2 + \beta_2\mathbf{r}_2)\mathbf{T}_2^{\mathrm{T}}(\boldsymbol{\ell}_2 - \alpha_2\mathbf{r}_2) \\
+m(m+1)\sin_{3,4}(\boldsymbol{\ell}_1 - \alpha_1\mathbf{r}_1)\mathbf{T}_1^{\mathrm{T}}(\boldsymbol{\ell}_1 + \beta_1\mathbf{r}_1) \\
-m(m+1)\sin_{3,4}(\boldsymbol{\ell}_1 + \beta_1\mathbf{r}_1)\mathbf{T}_1^{\mathrm{T}}(\boldsymbol{\ell}_1 - \alpha_1\mathbf{r}_1) \\
+n(m+1)\sin_{1,4}(\boldsymbol{\ell}_1 + \beta_1\mathbf{r}_1)\mathbf{T}_2^{\mathrm{T}}(\boldsymbol{\ell}_2 - \alpha_2\mathbf{r}_2) \\
+n(m+1)\sin_{1,4}(\boldsymbol{\ell}_2 - \alpha_2\mathbf{r}_2)\mathbf{T}_1^{\mathrm{T}}(\boldsymbol{\ell}_1 + \beta_1\mathbf{r}_1) \\
+m(n+1)\sin_{2,3}(\boldsymbol{\ell}_1 - \alpha_1\mathbf{r}_1)\mathbf{T}_2^{\mathrm{T}}(\boldsymbol{\ell}_2 + \beta_2\mathbf{r}_2) \\
+m(n+1)\sin_{2,3}(\boldsymbol{\ell}_2 + \beta_2\mathbf{r}_2)\mathbf{T}_1^{\mathrm{T}}(\boldsymbol{\ell}_1 - \alpha_1\mathbf{r}_1) \\
+mn\sin_{1,3}(\boldsymbol{\ell}_2 + \beta_2\mathbf{r}_2)\mathbf{T}_1^{\mathrm{T}}(\boldsymbol{\ell}_1 + \beta_1\mathbf{r}_1) \\
+mn\sin_{1,3}(\boldsymbol{\ell}_1 + \beta_1\mathbf{r}_1)\mathbf{T}_2^{\mathrm{T}}(\boldsymbol{\ell}_2 + \beta_2\mathbf{r}_2) \\
+(m+1)(n+1)\sin_{2,4}(\boldsymbol{\ell}_2 - \alpha_2\mathbf{r}_2)\mathbf{T}_1^{\mathrm{T}}(\boldsymbol{\ell}_1 - \alpha_1\mathbf{r}_1) \\
+(m+1)(n+1)\sin_{2,4}(\boldsymbol{\ell}_1 - \alpha_1\mathbf{r}_1)\mathbf{T}_2^{\mathrm{T}}(\boldsymbol{\ell}_2 - \alpha_2\mathbf{r}_2)
\end{aligned}
$$

We combine and cancel terms to obtain

$$
\begin{aligned}
&= n(n+1)\sin_{1,2}\big(- \alpha_2\mathbf{r}_2\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \beta_2\boldsymbol{\ell}_2\mathbf{T}_2^{\mathrm{T}}\mathbf{r}_2 - \beta_2\mathbf{r}_2\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \alpha_2\boldsymbol{\ell}_2\mathbf{T}_2^{\mathrm{T}}\mathbf{r}_2\big)\\
&+m(m+1)\sin_{3,4}\big(- \alpha_1\mathbf{r}_1\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \beta_1\boldsymbol{\ell}_1\mathbf{T}_1^{\mathrm{T}}\mathbf{r}_1 - \beta_1\mathbf{r}_1\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \alpha_1\boldsymbol{\ell}_1\mathbf{T}_1^{\mathrm{T}}\mathbf{r}_1\big)\\
&+n(m+1)\sin_{1,4}\big(\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_1 - \boldsymbol{\ell}_1\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\\
&\qquad + \beta_1\mathbf{r}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \alpha_2\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\mathbf{r}_2 + \alpha_2\mathbf{r}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \beta_1\boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\mathbf{r}_1\big)\\
&+(n+1)m\sin_{2,3}\big(\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\\
&\qquad - \alpha_1\mathbf{r}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \beta_2\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\mathbf{r}_2 - \beta_2\mathbf{r}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \alpha_1\boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\mathbf{r}_1\big)\\
&+nm\sin_{1,3}\big(\boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2\\
&\qquad + \beta_2\mathbf{r}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \beta_1\boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\mathbf{r}_1 - \beta_1\mathbf{r}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \beta_2\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\mathbf{r}_2\big)\\
&+(n+1)(m+1)\sin_{2,4}\big(\boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2\\
&\qquad - \alpha_2\mathbf{r}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \alpha_1\boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\mathbf{r}_1 + \alpha_1\mathbf{r}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \alpha_2\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\mathbf{r}_2\big)
\end{aligned}
$$

The first two terms cancel because of quadratic small angles. We are left with:

$$
\begin{aligned}
&= (\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1)\big(n(m+1)\sin_{1,4} + (n+1)m\sin_{2,3}\\
&\qquad - nm\sin_{1,3} - (n+1)(m+1)\sin_{2,4}\big)\\
&+\mathbf{r}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2\sin(\theta)\big(n(m+1)\beta_1 - (n+1)m\alpha_1 - nm\beta_1 + (n+1)m+1)\alpha_1\big)\\
&+\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\mathbf{r}_2\sin(\theta)\big(- n(m+1)\alpha_2 + (n+1)m\beta_2 - nm\beta_2 + (n+1)(m+1)\alpha_2\big)\\
&+\mathbf{r}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\sin(\theta)\big(n(m+1)\alpha_2 - (n+1)m\beta_2 + nm\beta_2 - (n+1)(m+1)\alpha_2\big)\\
&+\boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\mathbf{r}_1\sin(\theta)\big(- n(m+1)\beta_1 + (n+1)m\alpha_1 + nm\beta_1 - (n+1)(m+1)\alpha_1\big)
\end{aligned}
$$

$$= (\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1)(- \sin(\theta) +$$

$$\cos(\theta)(n(\alpha_1 + \beta_2) - m(\beta_1 + \alpha_2) - n(\beta_2 - \beta_1) - m(\beta_{-}\beta_1)))$$

$$+\mathbf{r}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 \sin(\theta)(n(m+1)\beta_1 - (n+1)m\alpha_1 - nm\beta_1 + (n+1)m+1)\alpha_1)$$

$$+\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \mathbf{r}_2 \sin(\theta)(- n(m+1)\alpha_2 + (n+1)m\beta_2 - nm\beta_2 + (n+1)(m+1)\alpha_2)$$

$$+\mathbf{r}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \sin(\theta)(n(m+1)\alpha_2 - (n+1)m\beta_2 + nm\beta_2 - (n+1)(m+1)\alpha_2)$$

$$+\boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \mathbf{r}_1 \sin(\theta)(- n(m+1)\beta_1 + (n+1)m\alpha_1 + nm\beta_1 - (n+1)(m+1)\alpha_1)$$

$$= (\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1)(-\sin(\theta) + \cos(\theta)(n\lambda_1 - m\lambda_2))$$

$$+\mathbf{r}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 \sin(\theta)(n\beta_1 + (n+1)\alpha_1)$$

$$+\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \mathbf{r}_2 \sin(\theta)((m+1)\alpha_2 + m\beta_2)$$

$$+\mathbf{r}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \sin(\theta)(-(m+1)\alpha_2 - m\beta_2)$$

$$+\boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \mathbf{r}_1 \sin(\theta)(-n\beta_1 - (n+1)\alpha_1)$$

$$= (\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1)(-\sin(\theta) + \cos(\theta)(n\lambda_1 - m\lambda_2))$$

$$+\mathbf{r}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 \sin(\theta)(n\lambda_1 + \alpha_1)$$

$$+\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \mathbf{r}_2 \sin(\theta)(m\lambda_2 + \alpha_2)$$

$$-\mathbf{r}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \sin(\theta)(m\lambda_2 + \alpha_2)$$

$$-\boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \mathbf{r}_1 \sin(\theta)(n\lambda_1 + \alpha_1)$$

$$= (\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1)(-\sin(\theta) + \cos(\theta)(n\lambda_1 - m\lambda_2))$$
$$+ \sin(\theta)((n\lambda_1 + \alpha_1)(\mathbf{r}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \mathbf{r}_1) +$$
$$(m\lambda_2 + \alpha_2)(\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \mathbf{r}_2 - \mathbf{r}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1))$$

To see what $\mathbf{H}_\lambda$ is,we merely see how $\mathbf{H}_m$ changes when we add 1 to n and m. We get

$$\mathbf{H}_\lambda = (\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1)(\cos(\theta)(\lambda_1 - \lambda_2))$$
$$+ \sin(\theta)(\lambda_1(\mathbf{r}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \mathbf{r}_1) + \lambda_2(\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \mathbf{r}_2 - \mathbf{r}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1))$$

We can see that there are some common elements. Let us make the following definitions.

$$\mathbf{L}_m = (\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1)$$
$$\mathbf{R}_{m,1} = (\mathbf{r}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \mathbf{r}_1)$$
$$\mathbf{R}_{m,2} = (\boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \mathbf{r}_2 - \mathbf{r}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1)$$

We can then simplify our $\mathbf{H}$ to be

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_d \\ \mathbf{H}_m \\ \mathbf{H}_\lambda \end{bmatrix}$$
$$= \begin{bmatrix} 2(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2)(|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2| - (\lambda_2 - \lambda_1)\boldsymbol{\ell}_1^{\mathrm{T}} \boldsymbol{\ell}_2) \\ \mathbf{L}_m \cos\theta(n\lambda_1 - m\lambda_2) + \sin\theta((n\lambda_1 + \alpha_1)\mathbf{R}_{m,1} + (m\lambda_2 + \alpha_2)\mathbf{R}_{m,2} - \mathbf{L}_m) \\ \mathbf{L}_m \cos\theta(\lambda_1 - \lambda_2) + \sin\theta(\lambda_1 \mathbf{R}_{m,2} + \lambda_2 \mathbf{R}_{m,2}) \end{bmatrix}$$

We thus have a textured plane reconstruction equation which is based purely on the location of our filter measurements, and the phase and wavelength of these filtered outputs. This, finally, makes the connection explicit between the multilinear geometry and the outputs of measurements on images.

Chapter 7

Multicamera Calibration Application

The multicamera calibration application uses the constraints and mathematics developed in the previous chapters to create a different kind of calibration application for multiple cameras. Since the inputs to the standard multilinear constraints are points and lines, most calibration algorithms have been based on these atoms.

We choose to calibrate using textures rather than points and lines for a host of reasons. Repetitive patterns can be measured more accurately than individual points. When you show an LED to a CCD, it can look like figure 7.1, for which there is no clear center. The chance of outliers is greatly reduced because there is much greater redundancy in the data. The use of textures allows the calibration of cameras which do not overlap in their fields of view. Nonlinear distortion is easier to measure when you have a single texture visible over a large portion of a camera's field of view. There may be others but these reasons were sufficient for us to develop a different sort of calibration system.

The multicamera camera calibration is written in MATLAB and C. The data mea-

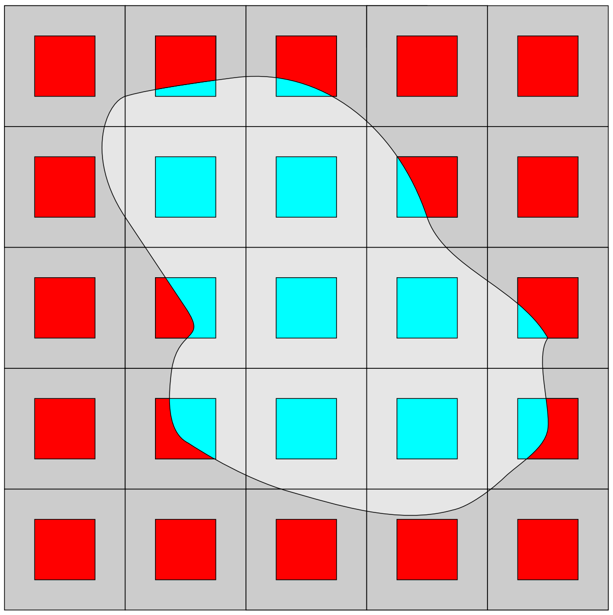

Figure 7.1: A set of parallel lines

surement routines are written in C using the Intel Image Processing Library, while all the geometric and matrix calculations are done in MATLAB for ease of programming and maintenance.

7.1 Method Overview

7.1.1 Initialization

When calibrating the cameras, we show two types of data. We show a set of parallel lines as in figure 7.2. We also show a pattern with orthogonal lines and boxes, as in figure 7.3.

The first thing the program does is to measure the parallel lines, which is diagrammed in figure 7.4. Since we know nothing about the orientation of the lines, we just measure in neighborhoods for the dominant line direction. We use the parallelism of

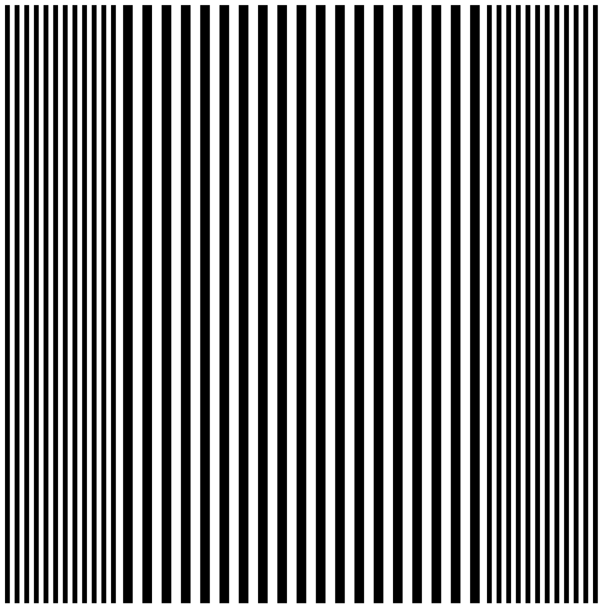

Figure 7.2: A set of parallel lines

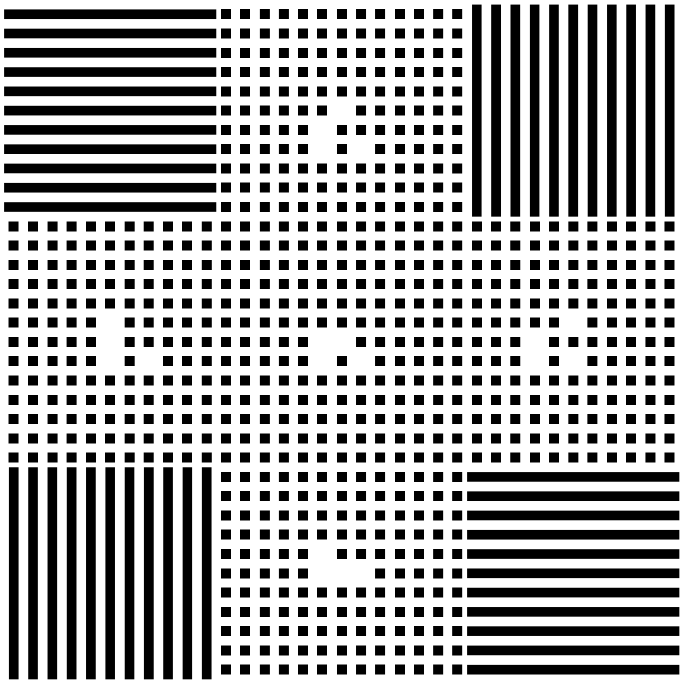

Figure 7.3: A calibration pattern with boxes

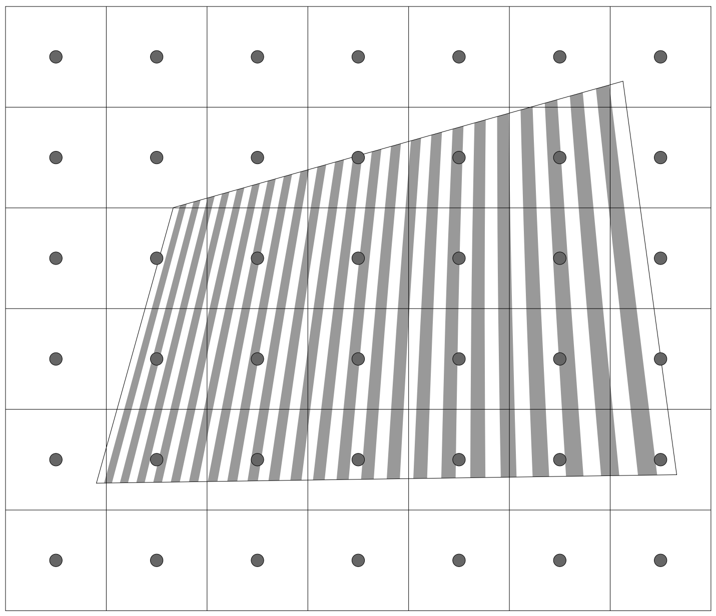

Figure 7.4: Measuring the image of a set of lines

the lines along with the assumption of square pixels to obtain an estimate of the nonlinear distortion's quadratic radial component.

We then measure the lines the the pattern with boxes and orthogonal lines. The orthogonal lines give us an estimate of the internal calibration for each of the cameras. We do this by finding the matrix A for which $\ell_{1,i}^T A \ell_{2,i} = 0$, for all images i from a single camera. We know that $A = KK^T$, so that we can take the Cholesky decomposition of A to obtain our K. It happens that for some cameras if the conditioning of the data is poor, then the matrix A will not be positive definite, and therefore the Cholesky decomposition does not exist. If there is at least one camera which has a positive definite A, which is usually the case, then this is not a problem for our method, since we can propagate this K estimate, since this has fixed an affine coordinate system. This estimate is rather poor, but is enough to provide initialization.

Once we have initial estimates for internal calibration, we use the parallel line measurements to find the external orientation of all the cameras. This is accomplished by adding one camera at a time to the complex of cameras, and then using the orthogonal lines to optimize over the external orientation and the internal calibration. Once all the external orientation and internal calibration parameters are measured, we obtain the orientation for all the textures.

7.1.2 Measuring the boxes

With the orientation of the cameras and the textures, we can “flatten” the texture by warping the image so that it may be measured in a flat manner. We convert something like that in figure 7.5 to something like figure 7.6, which is easily measured using a Fourier technique.

To measure the exact location of the boxes, we sum the pixel values in the x and y directions. We can unwarp so that we know to within a half a box width where the boxes area. If we merely convolve the x and y sums with an exponential e^{iax}, then the resulting phase of this exponential will tell us by how much to shift in order to obtain a precise location for the centers of the boxes. We can find these centers in the warped image and then warp these measurements back to the coordinate system of the original image.

From our synthetic tests, we estimate an accuracy of about a tenth of a pixel with these measurements, *given the correct warping parameters*.

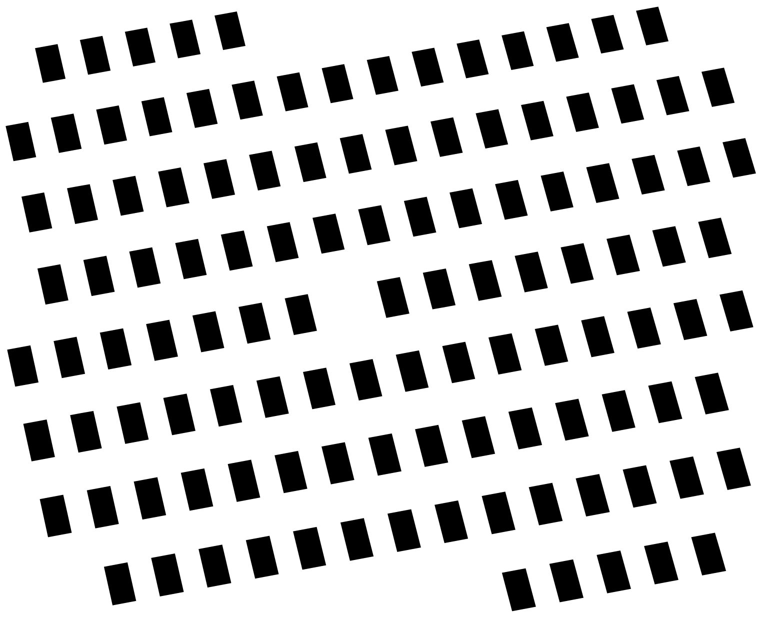
Figure 7.5: The image contains a projected area

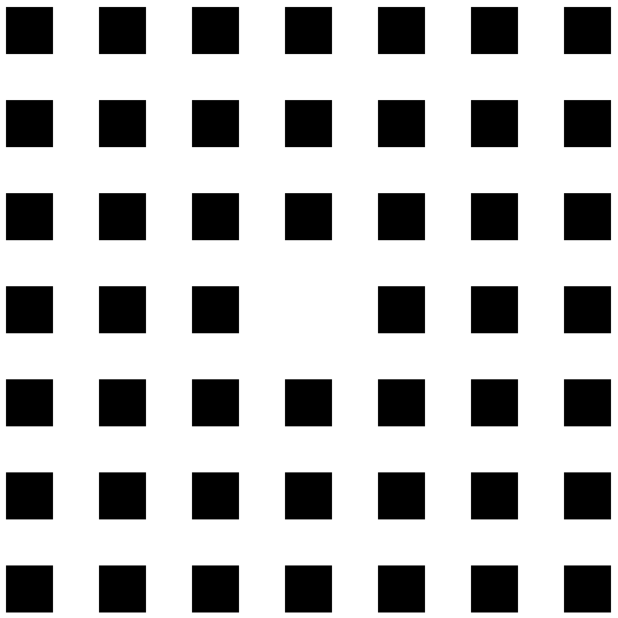
Figure 7.6: We measure on a warped image

7.1.3 Obtaining the translation

Note that there are particular patterns of missing boxes in the pattern, of which figure 7.3 is an example. We only make initial box measurements in the vicinity of these missing box patterns so we can determine the exact location of the measurements on our pattern. Using these measurements and the fact that we already have initial external orientation and internal calibration, we can find the camera positions using the techniques outlined in chapter 6.

The techniques in chapter 6 work not only for the standard epipolar and trilinear constraints, but also work for the octilinear constraints. Since our calibration pattern can extend over multiple fields of view, we may use all the constraints together to calibrate even cameras which do not share fields of view. This code has been implemented but not extensively tested, and it is possible it is somewhat unstable. I believe that the code can be improved with the addition of the hexalinear and quintilinear constraints, since it currently operates only with the octilinear constraint.

7.1.4 Optimizing the parameters

In any case, after these techniques are applied, we have initial estimates for all the camera parameters *and* all the locations for the patterns. What remains is an optimization problem over all these parameters. There are a couple of possibilities for optimization. One is to use bundle adjustment techniques to optimize over all the parameters at once. We have tried to implement this technique and it did not work. The optimization was programmed in a fast manner, with analytic computations of the sparse Hessian, but the optimization surface was not steep enough and standard optimization techniques could not find a minimum.

The technique that worked best was the simplest one. We merely estimate the textures locations using the current estimates for the camera parameters plus the measurements, then we estimate the camera parameters using those texture estimates plus the measurements. Doing this repeatedly converged in all the scenarios in which we tried.

7.1.5 Remeasuring the data

We alluded earlier to the fact that the measurements were depended on how good our estimates were for the camera parameters. Therefore for the best accuracy, we remeasure the textures once we have camera parameters which have been somewhat optimized. These measurements are more accurate and provide a better input for our optimization.

7.2 Nonlinear Factors and Lines

The prismatic line constraint operates just the same in the nonlinear case as it does in the linear case, except that we need to know where we measured our line. Since we work with image measurements, we will parameterize our functions over B^{-1}. We do this because it is very difficult to invert the function $\mathbf{r} = B^{-1}(\hat{\mathbf{r}})\hat{\mathbf{r}}$. It is easy to invert the matrix function $B(\hat{\mathbf{r}})$, and we do use this inverse. Note that this is the opposite of the way that most authors have defined it, but when the object is to use the images in order to reconstruct, it makes much more sense to parametrize the inverse of the projection than the forward direction.l

In the linear case, this inverse mapping is a matrix B^{-1} which is the same over the whole image. In this case the function inverse is easy and corresponds to the matrix inverse. In the nonlinear case, we have a matrix function $B^{-1}(\mathbf{r})$, which varies depending on the location of $\mathbf{r}$. So to put an image point into the translated fiducial coordinate system, we use

$$\mathbf{r} = B^{-1}(\hat{\mathbf{r}})\hat{\mathbf{r}} \tag{7.1}$$

The function is defined in terms of points $\hat{\mathbf{r}}$. If we have a measurement on a line which we would like to put in that same coordinate system, we can use the same technique as before and form:

$$\boldsymbol{\ell} = B^{\mathrm{T}}(\hat{\mathbf{r}})\hat{\boldsymbol{\ell}} \tag{7.2}$$

Note that this means that we need to know the location $\hat{\mathbf{r}}$ where we made our measurement of $\hat{\boldsymbol{\ell}}$. Because the location of our measurement is crucial our results, if we are to calibrate it is important to make a measurement of $\hat{\boldsymbol{\ell}}$ as locally as possible for maximum accuracy.

We consider in this section radial distortion, which in high quality lenses accounts for most of the distortion. If we express our matrix function as

$$B^{-1}(\hat{\mathbf{r}}) = R^{\mathrm{T}}S(K^{-1}\hat{\mathbf{r}})K^{-1} \tag{7.3}$$

where K is upper triangular, R is orthogonal, and

$$S(\hat{\mathbf{r}}) = I + (\hat{\mathbf{z}}\times(\hat{\mathbf{r}}\times\hat{\mathbf{z}}))(\rho_1 + \rho_2|\hat{\mathbf{z}}\times\hat{\mathbf{r}}|^2 + \rho_3|\hat{\mathbf{z}}\times\hat{\mathbf{r}}|^4)(\hat{\mathbf{z}}\times(\hat{\mathbf{r}}\times\hat{\mathbf{z}}))^{\mathrm{T}} \tag{7.4}$$

This is the reverse of the usual method of describing radial calibration, and while the parameters are not the same, the result encodes just the same information. To map lines, then, we use the equation

$$(B^{-1}(\hat{\mathbf{r}}))^{-\mathrm{T}} = B^{\mathrm{T}}(\hat{\mathbf{r}}) \tag{7.5}$$

$$= R^{\mathrm{T}} S^{-\mathrm{T}}(K^{-1}\hat{\mathbf{r}}) K^{\mathrm{T}} \tag{7.6}$$

But since S is symmetric

$$(B^{-1}(\hat{\mathbf{r}}))^{-\mathrm{T}} = R^{\mathrm{T}} S^{-1}(K^{-1}\hat{\mathbf{r}}) K^{\mathrm{T}} \tag{7.7}$$

Next note that if we can invert a matrix $I + \mathbf{v}\mathbf{v}^{\mathrm{T}}$ as

$$(I + \mathbf{v}\mathbf{v}^{\mathrm{T}})^{-1} = I - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{1 + \mathbf{v}^{\mathrm{T}}\mathbf{v}} \tag{7.8}$$

$$= I - \mathbf{v}\mathbf{v}^{\mathrm{T}}(1 - \mathbf{v}^{\mathrm{T}}\mathbf{v} + (\mathbf{v}^{\mathrm{T}}\mathbf{v})^2 - \dots) \tag{7.9}$$

So if we let

$$\mathbf{v} = (\hat{\mathbf{z}} \times (\hat{\mathbf{r}} \times \hat{\mathbf{z}}))\sqrt{\rho_1 + \rho_2|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^2 + \rho_3|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^4} \tag{7.10}$$

then we can see that

$$S(\hat{\mathbf{r}}) = I + \mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{7.11}$$

so that

$$S^{-1}(\hat{\mathbf{r}}) = I - (\hat{\mathbf{z}} \times (\hat{\mathbf{r}} \times \hat{\mathbf{z}}))(\hat{\mathbf{z}} \times (\hat{\mathbf{r}} \times \hat{\mathbf{z}}))^{\mathrm{T}}(\rho_1 + \rho_2|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^2 + \rho_3|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^4 \tag{7.12}$$

$$-|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^2(\rho_1 + \rho_2|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^2 + \rho_3|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^4)^2 \tag{7.13}$$

$$+|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^4(\rho_1 + \rho_2|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^2 + \rho_3|\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^4)^3 \dots) \tag{7.14}$$

We throw away all terms attached to $|\hat{\mathbf{z}}\times\hat{\mathbf{r}}|^6$ and higher to obtain

$$S^{-1}(\hat{\mathbf{r}}) = I - (\hat{\mathbf{z}}\times(\hat{\mathbf{r}}\times\hat{\mathbf{z}}))(\rho_1 + (\rho_2 - \rho_1^2)|\hat{\mathbf{z}}\times\hat{\mathbf{r}}|^2 + (\rho_3 - 2\rho_1\rho_2 + \rho_1^3)|\hat{\mathbf{z}}\times\hat{\mathbf{r}}|^4)(\hat{\mathbf{z}}\times(\hat{\mathbf{r}}\times\hat{\mathbf{z}}))^{\mathrm{T}} \quad (7.15)$$

Note that as this stands, it is impossible to use the same trick that we did before in premultiplying all the lines from one image, since the function depends on $\hat{\mathbf{r}}$, which changes from line to line. We may optimize for the radial calibration, but this must be a separate step from the internal calibration optimization.

7.3 Focal Length Changes on Nonlinear B

Let us look at the nonlinear equation for B more closely. We would like to solve for the radial calibration separately from the linear calibration, but we are thwarted by the structure of the equation. However, we may make reasonable assumptions about the cameras which allow us to separate the radial from the linear calibration. Because the linear method to find the initial linear calibration works best if we are working with linear cameras, we would like to have at least an initial radial calibration for the individual cameras before we do anything.

With modern cameras and lenses, we may assume both a zero skew, square pixels, and an optical center near the center of the image. If this is not the case, then if we know approximate values for skew, aspect ratio, and optical center, we still may radially calibrate first. If the radial distortion is small, then it is possible to linearly calibrate, ignoring these distortions, so that these parameters need not be known ahead of time. So we either assume approximate calibration matrix known, or small nonlinear distortion. In the case of unknown calibration and high nonlinear distortion, other methods must be used to calibrate.

Let us assume then, that we have a camera and we have an approximate calibration matrix K', where

$$\begin{aligned} K &= K' \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix} && (7.16) \\ &= K'A && (7.17) \end{aligned}$$

with K being an approximately correct camera calibration matrix. This equation corresponds to knowing everything about the internal parameters of the camera but the focal length.

Let us also note that

$$\begin{aligned} \hat{\mathbf{z}} \times (K^{-1}\hat{\mathbf{r}}) &= \hat{\mathbf{z}} \times (A^{-1}(K')^{-1}\hat{\mathbf{r}}) && (7.18) \\ &= \frac{1}{a}(\hat{\mathbf{z}} \times ((K')^{-1}\hat{\mathbf{r}})) && (7.19) \end{aligned}$$

We can also note that

$$(\hat{\mathbf{z}} \times \mathbf{v})A^{-1} = \frac{1}{a}(\hat{\mathbf{z}} \times \mathbf{v}) \qquad (7.20)$$

Using equation (7.18), we may rewrite our function

$$S(K^{-1}\hat{\mathbf{r}}) = I + (\hat{\mathbf{z}}\times((K')^{-1}\hat{\mathbf{r}}\times\hat{\mathbf{z}})) \cdot (\frac{\rho_1}{a^2} + \frac{\rho_2}{a^4}|\hat{\mathbf{z}}\times((K')^{-1}\hat{\mathbf{r}})|^2 + \frac{\rho_3}{a^6}|\hat{\mathbf{z}}\times((K')^{-1}\hat{\mathbf{r}})|^4) \quad (7.21)$$

Using equation (7.20), we may further rewrite our function $B^{-1}(\hat{\mathbf{r}})$ as:

$$B^{-1}(\hat{\mathbf{r}}) = R^{\mathrm{T}}A^{-1}S((K')^{-1}\hat{\mathbf{r}})(K')^{-1} \quad (7.22)$$

Using parameters ρ_i' instead of ρ_i, with

$$\rho_i' = \frac{\rho_i}{a^{2i}} \quad (7.23)$$

So if we use the wrong focal length, and then later find the correct one, we may adjust the ρ_i according to the above equation.

7.4 Using the Program to Calibrate

7.4.1 Obtaining Calibration Targets

Before you actually take data, you need patterns with which to take the data. We will store all the files in a single directory, which will be used for pattern generation *and* calibration. Load up Matlab to begin, make sure that you run `startup` in order to set all of the paths properly. Once you've done that, you can run

```
makeMultical('c:\myDirectory\')
```

This will create the following files:

patternTemplate This file specifies the calibration target with the boxes on it. Inside the file is a description of how to create a target of any shape you like. The pattern should have large fields of boxes, and also large fields of horizontal and vertical lines. The numbers `1-5` must be surrounded by at least 4 B's on all sides for the program to work.

calPattern.ps This is the Postscript file generated by `patternTemplate`. You can use this file to send to the printer.

pointLocs This important file generated from the pattern given in `patternTemplate` specifies the position of all the missing box patterns for the calibration program. This file must be present in the directory when the videos are used for calibration. You must show the pattern to the cameras which corresponds to this `pointLocs` file.

calPattern.tif This is a bitmapped version of `calPattern.ps`, for use if you want to generate synthetic images.

lineTemplate This file should just consist of lines going in one direction. You can use lines of three different widths, so the pattern can operate at different distances for more flexibility.

linePattern.ps This is the Postscript file generated by `lineTemplate`. You can use this file to send to the printer.

linePattern.tif This is a bitmapped version of `linePattern.ps`, for use if you want to generate synthetic images.

crossTemplate This files should contain orthogonal lines with no boxes. You can show this pattern to the cameras in order to make the rotation and internal calibration procedures more robust.

crossPattern.ps This is the Postscript file generated by `crossTemplate`. You can use this file to send to the printer.

crossPattern.tif This is a bitmapped version of `crossPattern.ps`, for use if you want to generate synthetic images.

multicalOptions This file contains options which control the operation of the program. The options are currently at settings which should work for a large variety of situations, but you may find need to change them. We discuss the options in greater depth next.

Once you have the files for the calibration targets, print them out and mount them as rigidly as possible. The calibration will be incorrect by the same amount which your target flexes. So if it flexes 1cm, then your calibration will be off by 1cm. Also confirm that the interbox/interline distance is at least 10 pixels when viewed from the middle of the viewing volume of the calibration setup.

7.4.2 Collecting the Data

The data collection is a crucial part of the process, and keep in mind that there is no need to work with a minimum amount of data. To the contrary, take as much data as you can. The main idea is to fill up the viewing volume with the patterns viewed from different orientations. In essence there is a six dimensional space consisting of 3D space plus the orientation of the target (not just the normal but the full orientation). I recommend capturing two sets of avi sequences. The first should contain just the parallel lines. The second should contain at least the box patterns with the orthogonal lines but preferable should contain some views of the target with just the orthogonal lines. So please make passes through the viewing volume in 60 degree increments (viewed from above) with the targets:

- vertical
- rotated 90 degrees but still vertical
- slanted facing down
- slanted facing down rotated
- slanted facing up
- slanted facing up rotated
- facing up (if you have cameras above)
- facing up rotated(if you have cameras above)

- facing down rotated (if you have cameras below)

In addition, show the box patterned target to each camera at a variety of distances and orientations.

In the end you should have two AVI files for each camera. One should have pictures of only parallel lines. The other should have pictures of orthogonal lines and boxes or just orthogonal lines. All the avi files should be synchronized. This is not an option. If the files are not synchronized the program will not work. Also, feel free to use some mild compression on the data. Since the measurement operators use a large amount of data, compression artifacts should not greatly affect the results.

7.4.3 Using the Data to Calibrate

Once the data is collected, then put the avi files in the same directory as the `pointLocs` files, which should be

```
 'c:\myDirectory\'
```

Once you have done this, start up Matlab again and remember to run `startup` so that all the paths are set properly. Then the calibration could just be a matter of running.

```
makeMultical('c:\myDirectory\lin*.avi', ...
             'c:\myDirectory\pat*.avi')
```

The program puts all its intermediate results into the same directory, and names the files by replacing the `.avi` extensions with various other extensions. The main ones for the results are:

- `.T` contains the position of its corresponding camera
- `.R` contains the orthogonal rotation matrix of its corresponding camera
- `.K` contains the upper triangular matrix of its corresponding camera
- `.kappa` contains the nonlinear parameters of its corresponding camera

The projection equation is as follows.

$$\mathbf{p} = K \cdot N(\kappa, R \cdot (\mathbf{P} - T)) \tag{7.24}$$

where $\mathbf{P}$ is the original 3D point and $\mathbf{p}$ is the homogeneous projection of the point. T, R, and K are the obvious translation, rotation, and internal calibration, respectively. N is a function which takes the five nonlinear parameters specified by κ and transforms the point by them. Of the five parameters, the first two and the last are the radial distortion coefficients of r^2, r^4, and r^6, respectively. The third and fourth parameter specify the tangential distortion. These parameters and the nonlinear model are the same as are used in the Caltech Camera Calibration Toolbox. You may look at the Matlab code to see how the parameters operate.

7.4.4 Options

The options are contained in the file `multicalOptions`. There are a few of major note:

numNonlin gives the number of nonlinear parameters which will be used in the optimization. Probably you should have this be 1 or 2, but if you know better, feel free to use 4 or 5. 3 wouldn't make sense here because then you would only be using one tangential parameter.

zeroSkew tells the program whether or not to force a zero skew. Most likely keeping this at 1 (true) should be fine for most cameras.

varThresh tells the program the minimum intensity difference between black and white. This is set to a fairly low value of 20, but feel free to set it higher if you're getting some spurious signals.

maxNoise tells the program the maximum variation within "black" or "white", before it rejects a particular measurement. This is set at 10, but feel free to change it if you need to.

The rest of the options have descriptions in the file for your reference.

7.5 Calibration Results

There are a few aspects to the success of a calibration program. They are:

1. accuracy
2. stability
3. performance with degraded data

7.5.1 Accuracy of Data

Accuracy tells us how small the reprojection error can be made when the parameters have been optimized. This is mainly a measure of the accuracy of the data measurements. In that respect the program described here does very well, since the measurements it takes can be made quite accurate. On synthetic data which we constructed and input into the program, we obtained average reprojection errors of approximately a quarter of a pixel, when using the correct nonlinear model so as to not overfit. When the true calibration parameters were input the error was of the same order, so this quarter pixel error is due to measurement error.

As far as the camera parameters go, we give the results of a synthetic test we did with 25 cameras. Our input K internal calibration matrices were:

$$K = \begin{bmatrix} 7.104 & 0.000 & 512.0 \\ 0.000 & 7.104 & 384.0 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{7.25}$$

Our input nonlinear parameter was just a radial distortion of 1.000. Our cameras were as equally spaced as we could manage them on a sphere. The images were 1024×768. We followed the procedures as laid out in this chapter for showing the synthetic images to all the cameras, and then ran the calibration program.

After calibration, our output K internal calibration matrices were

$$K = \begin{bmatrix} 7.12 \pm 0.01 & 0.000 & 512.0 \pm 0.3 \\ 0.000 & 7.12 \pm 0.01 & 384.0 \pm 0.3 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{7.26}$$

Our output nonlinear parameters were 1.005 ± 0.005. We cannot perform this same test on calibration parameters for actual cameras, for reasons that should be clear.

We took real data for the 3DPVT challenge, and that data is desribed in subsection 7.5.3. We are continuing work with calibration in order to

7.5.2 Stability

The stability of a calibration program measures the ability of the program to converge given a variety of data. On that score the program described here does reasonably well, but definitely it could improve. The trickiest part of the program is the calculation of the internal parameters from orthogonal lines and the construction of the external orientation for all the cameras. It is very important to have a good amount of data with orthogonal lines to make sure the program runs stably. If this is the case, then the program works quite well, but if not, there will probably be problems in the calculation of external orientation and internal calibration.

7.5.3 Performance with degraded data

When we calibrated the camera system for the 3DPVT challenge, we had a system of 31 cameras. Of course when you take data from such a large system it is practically

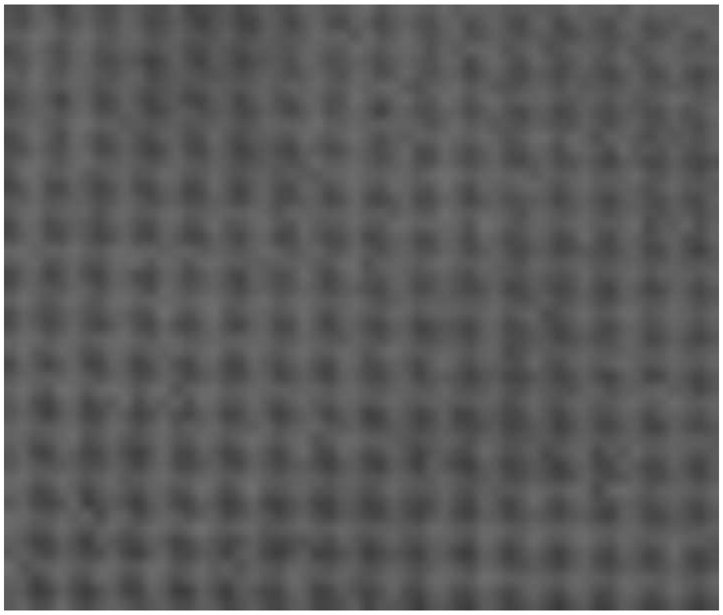

Figure 7.7: An example of poor data

impossible to make sure all the focus and exposure is correct. To show the advantages of the calibration method, we show some of the input data from one of the “bad” cameras in figure 7.7. Notice that the Bayer filter pattern dominates this data. If you squint you can just make out the real pattern. Even though most of the data in this camera was of this poor quality (out of focus, bad exposure), because our operators were taken over a large area, we were able to integrate this data into good calibration. Our reprojections errors were on average about a pixel and the worst reprojects were about four pixels for the very edges. Considering the quality of the data, these are quite good results.

Chapter 8

Final Thoughts on The Feedback Loop

We give a sketch how these constraints could be used for image processing. We first argue that one always has some initial calibration information. This is usually couched in terms of having multiple cameras "looking at the same object", but this is not a precise formulation. There is a relation between the depth of the viewed scene and the distance between the cameras which is the important factor for this initial calibration information. Indeed, if we were to have a camera on Earth and a camera on Venus looking at terrain, we would not expect to be able to correspond anything. However, if we turned the cameras toward the sky, we certainly could correspond, since our baseline is small related to the distance to the object. Thus any assumption which forms the basis for a correspondence method can always be expressed as a constraint on the error in calibration with respect to the distance to the scene.

Let us first look at an example of a scene for which it is impossible to compute correspondence. If we have a collection of textured planes, and these planes do not contain any wavelength greater than λ, then it is clear that if our camera positions are not known to accuracy at least less than λ, then it is impossible to compute any sort of correspondence. We may be able to compute the rotational calibration for the cameras from the patch correspondence, but after that we are stuck.

On the other hand, if we know our camera positions to within $\alpha << \lambda$, and we have many textures with wavelengths greater than λ, then it *is* possible to find our integer indices with a high degree of probability. The smaller α is, the higher the degree of probability that we can find the integer indices. Once we have the integer indices, we can turn the constraint around to improve the camera positions.

One can see the beginnings of the second part of the feedback mechanism taking shape. If we at first use patches with wavelengths much greater than our calibration error, we can then improve our camera position estimates. With these new estimates, we can then use patches with smaller wavelengths, which will be more accurate. Thus, a small number of these iterations should result in excellent positional estimates, depending on the frequency content of the scene.

Now how do these ideas integrate with the correspondence problem in a feedback sense? Let us assume that we have M cameras which are each viewing a line from a singly textured plane. Let us further assume that we know the positions $\mathbf{T}_i$ of the cameras. Let us write:

$$\mathbf{N} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_M \end{bmatrix} \tag{8.1}$$

Then if we form a matrix F with a zero diagonal, and the i row and j column equal to

$$(\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j)^{\mathrm{T}} \left[\sum_{k_1,k_2,k_3 \in [1..M]} (\boldsymbol{\ell}_{k_1} \times \boldsymbol{\ell}_{k_2}) \mathbf{T}_{k_3}^{\mathrm{T}} \boldsymbol{\ell}_{k_3} \operatorname{sign}(i, j, k_1, k_2, k_3) \right] \tag{8.2}$$

Using our line texture plane constraint, we may then know that

$$\mathbf{N}^{\mathrm{T}} F \mathbf{N} = 0 \tag{8.3}$$

We have a similar situation to the multiple camera situation above. First, let us note that the diagonal is zero, so that the matrix is singular. If we know that our lines are equally spaced, then we know that the zero eigenvector of F will be in the direction $\mathbf{N}$, the vector of integer indices. If our cameras have an error, then this $\mathbf{N}$ will have an error. But if we use the constraint that $\mathbf{N}$ must be a multiple of integers, this error can be erased in some situations.

However, the matrix F is not positive definite, so we can't use any sort of eigenvalue decomposition technique to find this direction $\mathbf{N}$. Also, it is quite difficult to calculate the error in $\mathbf{N}$ based on the errors in the translational vectors. The solution to these problems is beyond the scope of this dissertation, but hopefully could be solved with some further research.

Let us also point out how these feedback ideas degenerate when we talk about points and lines. One can think of a single line in space as a Dirac delta function in a direction on a plane. When you take the Fourier transform, it contains all the harmonics, from long to short wavelengths. You can see then, that a single line in space fits into the feedback theory since you have long and short wavelengths, and the feedback essentially collapses into the standard epipolar/trilinear constraints.

BIBLIOGRAPHY

[1] Y. Aloimonos and M.J. Swain. Shape from texture. *BioCyber*, 58(5):345–360, 1988.

[2] P. Baker and Y. Aloimonos. Complete calibration of a multi-camera network. In *Proc. IEEE Workshop on Omnidirectional Vision*, pages 134–141, Hilton Head Island, SC, 2000. IEEE Computer Society.

[3] Joao Barretto and Kostas Daniilidis. Wide area multiple camera calibration and estimation of radial distortion. In P. Strum, T. Svoboda, and S. Teller, editors, *Omnivis*, pages 5–16. IEEE, 2004.

[4] A. Blake and A. Zisserman. *Visual reconstruction*. MIT Press, Cambridge, MA, 1987.

[5] O. Bottema and B. Roth. *Theoretical Kinematics*. Dover, 1990.

[6] D. Brown. The bundle adjustment - progress and prospect. In *XIII Congress of the ISPRS*, Helsinki, 1976.

[7] L. Van Gool C. Strecha, R. Fransens. Wide-baseline stereo from multiple views: a probabilistic account. In *Computer Vision and Pattern Recognition*, volume 2, pages 552–559. IEEE, 2004.

[8] M. Clerc and S. Mallat. The texture gradient equation for recovering shape from texture. *PAMI*, 24(4):536–549, April 2002.

[9] W. Courington, L. Leroy, G. Gordon, J. Woodfill, M. Harville, H. Baker, and M. Foster. Real-time stereo vision for real-world object tracking. Technical report, Tyzx, April 2000.

[10] O. D. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, Cambridge, MA, 1992.

[11] M.A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[12] A.W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *CVPR01*, pages I:125–132, 2001.

[13] J. Garding. Direct estimation of shape from texture. *PAMI*, 15(11):1202–1208, November 1993.

[14] R. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *CVPR92*, pages 761–764, 1992.

[15] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[16] R.I. Hartley. Camera calibration using line correspondences. In *DARPA93*, pages 361–366, 1993.

[17] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America*, 8:377–385, 1991.

[18] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[19] Yi Ma, Kun Huang, Rene Vidal, Jana Kosecka, and Shankar Sastry. Rank conditions on the multiple view matrix. *International Journal of Computer Vision*, 59(2), September 2004.

[20] J. Plücker. On a new geometry of space. *Philisophical Transactions of the Royal Society of London*, 155:725–791, 1865.

[21] Marc Pollefeys, Reinhard Koch, Maarten Vergauwen, and Luc J. Van Gool. Metric 3d surface reconstruction from uncalibrated image sequences. In *Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, pages 139–154. Springer-Verlag, 1998.

[22] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Proc. 8th International Conference on Computer Vision, Vancouver, Canada*, July 2001.

[23] F. Shevlin. Analysis of orientation problems using plucker lines. In *ICPR98*, page CVP1, 1998.

[24] M. E. Spetsakis and J. Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:171–183, 1990.

[25] Tomás Svoboda, Daniel Martinec, and Tomás Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperator and Virtual Environments*, 14(4), August 2005.

[26] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:13–27, 1984.

[27] T. Tuytelaars and L.J. Van Gool. Matching widely separated views based on affine invariant regions. *IJCV*, 59(1):61–85, August 2004.

[28] A. Zalesny and L. Van Gool. Multiview texture models. In *IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition (CVPR'01)*, volume 1, pages 615–622. IEEE Computer Society, December 2001.

[29] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc International Conference on Computer Vision*, volume 1, pages 666–673, 1999.

[30] Z.Y. Zhang. A flexible new technique for camera calibration. *PAMI*, 22(11):1330–1334, November 2000.