Sven Utcke

Error Propagation in Geometry-Based Grouping

Dissertation zur Erlangung des Doktorgrades
der Fakultät für Angewandte Wissenschaften
der Albert-Ludwigs-Universität Freiburg im Breisgau

ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG

25. April 2006

Acknowledgements

The work described in this thesis, and its description itself, spans over nearly a decade. During this time many people accompanied me and my work: supervisors and colleagues, friends and relatives, acquaintances and strangers have left their impact, be it stimulating or restricting, inspiring or correcting, aiding or abetting, encouraging or forbearing, loving or caring. Everyone of them my heartfelt thanks!

Prof. Burkhardt and Prof. Neumann I have to thank for giving me the possibilities and wide support for the research which has led to this thesis. I have always enjoyed working for Prof. Burkhardt, who invariably supported my independence, and I'm indebted to Prof. Neumann, who offered me a position among his staff when Prof. Burkhardt left Hamburg to follow a call from the University of Freiburg.

I also wish to thank the numerous colleagues and fellow researchers who helped along my thesis through many fruitful discussions and who, in many cases, took on themselves the considerable burden of proofreading various parts of this thesis at various stages. My particular thanks go to Andrew Zisserman, who in many respects shaped my research interests and methodology and whose inimitable people skills will always serve as an unreachable example; others who went far beyond the call of duty include Simon Julier, Nic Pillow, Jeff Uhlmann, and Michael Nölle. Andreas Bieniek, Marc Schael, Sven Siggelkow, and Gerald Schreiber made my work at Ti-I and, in the case of the first three, LMB, most pleasing, and Ullrich Köthe and Hans Meine did the same for me at KOGS; the last two often also had to serve as a sounding board for new ideas — as Ullrich often used me to sound out his ideas. Thank you very much for your comradeship!

But most of all my thanks belong to my family: to my mother, Christl Utcke-Hamann, whose unshakeable trust in me and my abilities, as well as her constant and devotional support made me into the person I am today. Vielen, vielen Dank Mama, Du bist die beste! And to my wife, Gabi Beutner, for the countless hours I was excused from household and other chores in order to work on my thesis (and sometimes did), and for the longanimity and patience with which she bore my sulkiness and fretfulness during the bleaker moments of my work; and finally to my son, Moritz Utcke, whose birth was the final impetus which ultimately had me finish this thesis.

Without you, this thesis would not be.

Zusammenfassung

In dieser Arbeit beschreibe ich meinen Ansatz zur Kombination von Methoden der Fehlerfortpflanzung mit mehreren Algorithmen, die das Geometrie-basierte *grouping* von Strukturen erlauben. Von der bekannten Literatur unterscheidet sich meine Arbeit vor allem durch die Schwerpunktsetzung auf Anwendbarkeit: die tatsächliche praktische Anwendung zeigt deutlich, welche zusätzlichen Möglichkeiten man durch Fehlerfortpflanzung gewinnt; andererseits habe ich, statt starr an der exakten Lösung festzuhalten (die, wo möglich, natürlich gegeben wird) auch untersucht, welche Auswirkungen die Verwendung von Näherungslösungen haben kann — und in welchen, in der Literatur teilweise recht häufig anzutreffenden, Fällen solche Näherungslösungen verheerende Auswirkungen auf die Korrektheit (oder sogar Existenz) des Ergebnisses haben können.

Warum glaube ich, dass solch eine Arbeit nötig oder auch nur nützlich sein kann? Zumal doch die Grundlagen der Fehlerfortpflanzung (wenn auch nicht in der projektiven Geometrie) seit vielen Jahrzehnten bekannt sind und oft genug bereits in der Schule unterrichtet werden? Einer der Gründe für die geringe Verbreitung der Fehlerfortpflanzung unter Bildverarbeitern liegt meiner Meinung nach in der vorhandenen Literatur, deren Interesse stets der *korrekten* Lösung gilt, ohne Blick auf die *praktische* Anwendbarkeit.

Im Gegensatz hierzu ist die vorliegende Arbeit aus der Praxis für die Praxis entstanden: ich zeige anhand von Beispielen, dass sich viele Probleme tatsächlich *einfacher* lösen lassen, wenn man Grundlagen der Fehlerfortpflanzung berücksichtigt — oder sogar nur dann; ich denke die Anwendung auf Zebrastreifen in Kapitel 5 meiner Dissertation ist so ein Beispiel. Dabei behalte ich jedoch stets die algebraische und algorithmische Komplexität der verwendeten Verfahren sowie die *Notwendigkeit* zu ihrer Verwendung (oder, auch das kann passieren, die mangelnde Notwendigkeit) im Auge. Aus diesem Grund beschreibe ich nicht nur die Kombination von Fehlerfortpflanzung und projektiver Geometrie (die für den uneingeweihten einige Schwierigkeiten bereithält) sondern demonstriere die Anwendung dieser Prinzipien anhand von 3 sehr verschiedenen Beispielen. Im Folgenden beschreibe ich den Aufbau meiner Arbeit.

Nach Einleitung und einführenden Erläuterungen zu projektiver Geometrie und Fehlerfortpflanzung in den Kapiteln 1–3 beginnt der Hauptteil meiner Arbeit in Kapitel 4, in dem die Verbindung zwischen Fehlerfortpflanzung und projektiver Geometrie herausgearbeitet wird. Die zugrundeliegende Idee ist nicht neu und geht auf Kanatanis N-Vektoren zurück; darüber hinausgehend beschreibe ich aber auch die Anwendung der gleichen Grundsätze auf andere Parametrierungen und leite eine Reihe neuer Ergebnisse her, wie zum Beispiel eine hervorragende Ap-

proximation der Kovarianz eines an einige Edgel angepassten Linienstücks, eine Abbruchbedingung für inkrementelle line-fits und einen neuen Algorithmus für die Berechnung des Doppelverhältnisses von 4 Linien, welcher Aufgrund der Verwendung von Fehlerfortpflanzung tatsächlich sogar *schneller* ist als bisherige Verfahren. Desweiteren gebe ich eine Erklärung, warum die von vielen Autoren verwendete sphärische Normalisierung von Koordinaten tatsächlich einer Euklidischen Normalisierung überlegen ist; und schließlich gebe ich eine Übersicht darüber, wie viele der häufigsten Messgrößen am sinnvollsten verglichen werden können — allein dieser letzte Abschnitt könnte bereits viele der in der Bildverarbeitung so häufig anzutreffenden, fein eingestellten Parameter überflüssig machen.

In den daran anschließenden drei Kapiteln beschreibe ich verschiedene Anwendungsszenarien. Die erste Anwendung in Kapitel 5 ist die Erkennung von Zebrastreifen (und anderer periodischer Strukturen). Es handelt sich hier um eine Anwendung von der ich glaube, dass sie so ohne Fehlerfortpflanzung nicht möglich gewesen wäre; besonders interessant an dieser Anwendung ist, wie einige wenige Konfidenz-Tests eine Vielzahl manuell zu wählender Parameter ersetzen können, wodurch ein extrem stabiles System entstanden ist.

Die Algorithmen, die in Kapitel 6 beschrieben werden, beschäftigen sich mit der Segmentierung von Häuserfronten (orthogonalen und parallelen Strukturen) in Einzelbildern. Es wird kein fertiger Algorithmus präsentiert, stattdessen wird dieses Szenario genutzt, um eine Anzahl unterschiedlicher und auf unterschiedlichen Skalen operierender Techniken zu vergleichen. Der Schwerpunkt liegt auf der Bestimmung kollinearer Liniensegmente und von Fluchtpunkten.

Das letzte Anwendungskapitel, Kapitel 7, beschreibt schließlich Teile der Segmentierungsroutinen, die meinen ältesten Publikationen über die Erkennung rotationssymmetrischer Objekte zugrundeliegen. Ein wesentliches Merkmal ist dabei das Bild der Rotationsachse. Dieses lässt sich theoretisch als eine Linie durch die Schnittpunkte von Bitangenten berechnen. Da diese jedoch erheblich in ihrer Genauigkeit variieren können, haben wir hier ein exzellentes Beispiel, um verschiedene Algorithmen zu vergleichen; ich zeige, wie selbst ein bekannter und häufig genutzter Algorithmus wie die kleinste Summe der Fehlerquadrate zu unbrauchbaren Ergebnissen führen kann, wenn die zugrundeliegende Annahme unabhängiger, isotroper und gleichverteilter Fehler nicht zutrifft, und stelle bessere Alternativen vor.

Contents

Symbols

x, X	: scalars.
$\mathbf{x}$, $\mathbf{X}$	: vectors. In a transformation, capital letters usually indicate the source of a transformation, small letters indicate the target.
P	: matrix.
$\boldsymbol{\Sigma}$	: covariance matrix.
$\mathsf{J}_{\mathbf{yx}}$	: Jacobian; matrix of first derivatives of $\mathbf{y}$ with respect to $\mathbf{x}$. This is a matrix proper if $\mathbf{x}$ and $\mathbf{y}$ are both vectors, a vector (either row or column) if one of the two is a scalar variable, and a scalar if both x and y are scalar variables.
$\propto$	: proportional to.
∞	: infinity.
$\mathbb{R}$	: set of real numbers.
$(\,\cdot\,)^{-}$	: pseudoinverse.
$(\,\cdot\,)^{-}_{n}$	: pseudoinverse computed by setting all eigenvalues except the first n to zero.
$\lvert\cdot\rvert$	: determinant.
$\lvert\cdot\rvert_{n\times n}$	: determinant of the upper left $n \times n$ matrix.
$\lVert\cdot\rVert$	: norm.
$(\,\cdot\,)^{\mathsf{T}}$	: transpose.
$(\,\cdot\,)^{-\mathsf{T}}$	: inverse of the transpose (or, of course, transpose of the inverse).

Chapter 1

Introduction

The last thing we decide in writing a book is what to put first.

Blaise Pascal, 1623–1662

1.1 Grouping and Error Propagation

This thesis describes the approach used for, and the improvements possible by, the use of error propagation in conjunction with several algorithms for the grouping of structures based on geometric entities. But rather than rigidly favouring the *exact* solution each and every time[1] I have put particular weight on practicability, demonstrating the relative gain for many approaches and giving shortcuts where the results are not marred by their use; but also demonstrating how common shortcuts used by many authors can lead to disaster if the underlying assumptions are violated.

1.1.1 Why Error Propagation?

Why do I believe that such a thesis is necessary and indeed valuable? The principles of linear error propagation, which I will use in this thesis, have been known for a long time, often enough they are even taught in school; they are the staple of photogrammetrists, geodesists, physicists, as well as many other scientists. But — they are rarely enough used in computer vision. True, a number of publications exist, starting with Kanatani's work [70, 75] more than 13 years ago, and with Förstner's contribution to the "Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neurocomputing and Robotics" [49] as the latest, very nice, example[2]; but by and large error propagation has been all but ignored by the computer vision community.

I believe that the reason for this disregard is twofold: for one thing error propagation is simply unknown in computer vision circles, and if Kanatani didn't manage to change this then surely this thesis won't be able to either. But I also believe that error propagation is seen as an unnecessary complication: "*Let me solve this really complicated and important problem first, and then I can worry about details like error propagation*" seems to be the attitude of many a researcher, or even "*Sorry, but error propagation is much too slow for any real(-time) application*". And such a mind-set is unfortunately fostered by authors like Kanatani, who are more interested in *correct* than in *practicable* solutions. And it is here that I hope this thesis could have a small impact: demonstrating that many problems are indeed much *easier* solved using error propagation, or indeed *only* solvable using error propagation — I believe that the application described in Section 5 is such an example — but all the time with a firm eye on computational complexity as well

[1] Exact in its derivation, that is.
[2] Chapter 4.1 lists more literature on the subject

as the *necessity* for error propagation (or, as it sometimes happens, the lack of it). It is to this end that I not only describe the combination of error propagation with projective geometry, which for the unwary keeps a number of stumbling blocks at hand, but also demonstrate 3 very different application domains. In the following I'll describe the outline of this thesis in more detail.

1.2 The Outline of this Thesis

The flow of this thesis goes from the theoretical foundations (projective geometry, error propagation, and their combination) to practical applications showcasing one or more of the previously described theoretical principles; within the application chapters I go from the 2D case of a single planar homography to the case of several homographies all within one image and from there to the case of an even less restricted class of objects, surfaces of revolution.

In more detail, I'm starting this thesis with an overview of the state of the art in projective geometry (Chapter 2) and error propagation (Chapter 3) respectively. These chapters do not contain anything new and are for a huge part lifted straight out of [103] and a couple of other books, in spirit if not in words. If you know your way around projective geometry or error propagation I would recommend to simply skip the respective chapter, they are here for completeness, and as a handy reference for later work.

The actual thesis starts with Chapter 4, which combines projective geometry and error propagation. The underlying idea is not new, and as far as the application to homogeneous coordinates is concerned can be found in [75]; however, in this chapter I also consider the application of these principles to other parameterisations than homogeneous coordinates and, starting from first principles, derive a number of new results such as an excellent approximation to the covariance of a line segment fitted to edgels, a new stopping-criterion for incremental fits based on a χ^2-test, and a new algorithm for the calculation of the cross-ratio of 4 lines which due to the use of error propagation in fact performs *faster* than current algorithms. I will also give an intuitive explanation why the spherical normalisation used by many authors is indeed superior to an Euclidean normalisation; and finally I will give an overview on how to compare a number of common stochastic entities. Just this last section alone could already put away with many of the numerous, finely tuned parameters so common to computer vision algorithms.

The next three chapters describe different application scenarios. In Chapter 5 I describe the application of error-propagation principles to the grouping and recognition of zebra crossings and other repeated structure. This application was first described by me in [6], and is a nice example of an implementation which I believe would have been impossible without the use of error propagation due to the high variations of a zebra-crossing's size and quality even within a single image; of particular interest here is how only a few confidence-tests can replace a host of manually chosen parameters, resulting in a uniquely stable algorithm. It describes the groundbreaking work on which later publications such as [135] build.

In Chapter 6 I outline an algorithm for the grouping of houses (or, indeed, any structure consisting of orthogonal and parallel elements). Over the years we have seen a few algorithms for the reconstruction of buildings from monocular images [36, 87, 97], however, in contrast to multi-view approaches these nearly always require manual segmentation of image regions. The algorithm outlined in this chapter could be seen as an attempt to remedy this situation. It is, however, included in this thesis for a different reason: buildings show a number of diverse features at different scales, and I will in particular have a closer look at collinear line segments of only a few pixels to several hundreds of pixels in length and distance as well as vanishing points, the image of intersection of parallel lines at infinity, which can be anywhere from literally in the image to literally at infinity. What is more, these features come with differing accuracies, and even one and the same feature can have different accuracies attached to it depending on context. This application is therefore well suited as a showcase for several different ideas and approaches such as a new algorithm for the iterative improvement of vanishing-point position and one for the automatic grouping of vanishing points; a new objective function for the (partial) calibration of a camera from vanishing-points which takes the different uncertainties in the positions of the vanishing points into account and extends the usual Legoland assumption to more general setups; an extension on previous work which takes the vanishing-point information into account when merging line-segments; and finally a comparison of the performance of several different error-measures, both new ones first introduced in this thesis as well as established ones from the literature, for the identification of collinear line segments.

Chapter 7 finally describes part of the grouping algorithm underlying some of my older publications on the recognition of surfaces of revolution such as [3–5, 9], but also newer publications on their reconstruction, such as [8]. An important feature for both recognition as well as reconstruction of SORs is the object's axis. The axis can be calculated, e. g., based on the intersections of bitangents, which can vary considerably in their accuracy; it is therefore an excellent example to compare the performance of a number of established algorithms on a number of different features and to demonstrate how even a well-known and often-used algorithm like total least squares will fail if the underlying assumptions (iiid-data) are violated; much better alternatives are introduced and an extensive comparison and discussion shows the merit of error propagation for a problem which, in similar form, one can see tackled with unsuitable tools at nearly any computer-vision conference, even today. The comparisons are done on real contour-data derived from real images which previously appeared in publications about the grouping and recognition of SORs.

This thesis ends, as all theses do, with a conclusion and outlook in Chapter 8.

Due to the diverse nature of the underlying problems, ranging from projective geometry to error propagation, from intrinsically two-dimensional problems like the recognition of repeated structure to intrinsically three-dimensional problems like the grouping of box-like and even (partly) free-form objects (surfaces of revolution), there is no separate chapter entitled “literature survey”. Instead you can find a small overview over the then relevant literature in each chapter’s introduction, and then again whenever a direct reference can help to set the work described in context. The bibliography itself comes in two parts, starting with a list of my own relevant work on page 217 and the bibliography proper on page 219.

Chapter 2

Projective Geometry

. . . experience proves that anyone who has studied geometry is infinitely quicker to grasp difficult subjects than one who has not.

Plato, The Republic, Book 7, 375 B. C.

2.1 Introduction

When working in computer vision and image understanding, one of the first things one often seeks to describe is the image formation process, i. e. how are the real world and any specific image of this world related to each other. This connection can be made elegantly by projective geometry.

Projective geometry is much older than computer vision. According to [138] the first systematic treatise on projective geometry was published 1822 by Poncelet in his *Traité des propriétés projectives des figures*. Prompted by Felix Klein's Erlangen programme of 1872 [79] as well as a general interest in invariant theories, projective geometry became rather fashionable among the mathematicians of the late 19th and early 20th century (e. g. [39]). The book that by many in the vision community is considered the standard reference on projective geometry, *Algebraic Projective Geometry* by J. G. Semple and G. T. Kneebone [138], dates back to 1952. Only comparatively recent trends in computer vision require a somewhat more involved algebra; mostly tensor algebra as it is used in shape from multiple view approaches [59]. However, since this thesis concentrates on single view geometry, only standard projective geometry is used here.

This chapter describes the theory and principles of projective geometry as they apply to this thesis. Starting from 2D projective transformations, the notion of homogeneous coordinates is introduced and several subgroups of the projective group are presented (Section 2.2). This leads naturally to the discussion of different camera models in Section 2.3. Points, lines and conics are introduced (Sections 2.4 and 2.5) as well as the crossratio of four collinear points or four coincident lines respectively (Section 2.6). Finally some special transformations (canonical frames in Section 2.7 and "projective symmetry" in Section 2.8) are presented, and an alternative representation of the projective plane is introduced: the Gaussian sphere (Section 2.9), which has proven useful for error-propagation purposes or algorithms like the grouping by vanishing points discussed in Section 6. This introduction is naturally a rather brief and incomplete one, the interested reader can find additional information in, e. g., [43, 69, 103, 138, 146].

2.2 Projective Transformations

Projective geometry describes a group based on central (conic) projections. Confining ourselves to an image's two dimensions, each projection can be visualised as a central projection from an arbitrary plane Π' onto a second plane π, compare

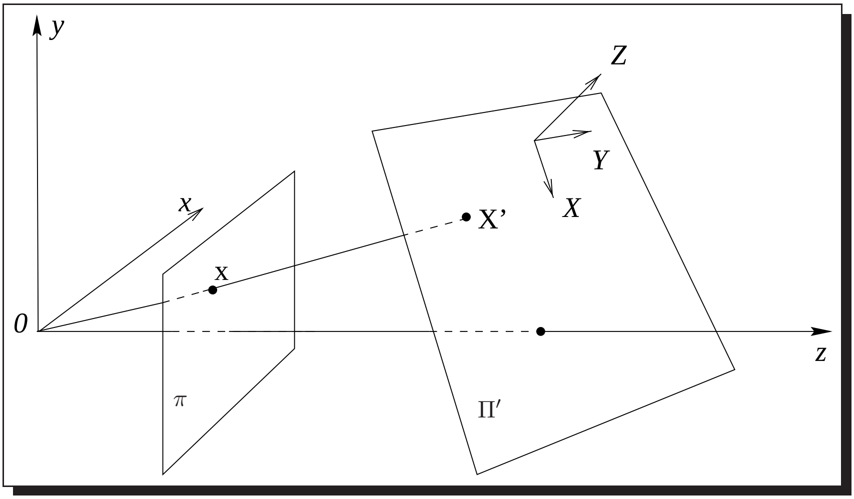


Figure 2.1: *A central projection from one plane onto another.*

Figure 2.1. The totality of all those projections from one plane onto another forms the projective group [138].

Since any two-dimensional plane in 3D can be transferred into any other two-dimensional plane by rotation and translation[1], we can think of any plane Π' as a rotated and translated version of the special plane Π formed by the points $\mathbf{X} = (X, Y, 0)^{\mathsf{T}}$. Any point $\mathbf{X}$ on Π is transformed into a new point $\mathbf{X}'$ on an arbitrary plane Π' with

$$\mathbf{X}' = \mathsf{R}\,\mathbf{X} + \mathbf{t}, \tag{2.1}$$

where $\mathsf{R} \in \mathbb{R}^{3\times 3}$ is the matrix of rotation and $\mathbf{t} \in \mathbb{R}^3$ the vector of translation.

Since the third coordinate of $\mathbf{X}$ was chosen to be 0, the rigid transformations between Π and Π' (translation and rotation) can be combined into a single 3×3 transformation matrix, namely

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}. \tag{2.2}$$

Here r_{ij} denotes the element in the i^{th} row and j^{th} column of R.

[1]Possibly by an infinite amount.

The central projection from a point $\mathbf{X}' \in I\!R^3$ on the plane Π' on to a point $\mathbf{x} = (x, y, 1)^\mathsf{T} \in I\!R^2$ on the plane π is given by

$$\begin{aligned} x &= \frac{X'}{Z'} = \frac{r_{11}X + r_{12}Y + t_1}{r_{31}X + r_{32}Y + t_3} \\ y &= \frac{Y'}{Z'} = \frac{r_{21}X + r_{22}Y + t_2}{r_{31}X + r_{32}Y + t_3}. \end{aligned} \tag{2.3}$$

This makes the nonlinear nature of projection in Euclidean coordinates apparent.

Equation (2.3) does not yet describe the *group* of 2D projective transformations; in particular the r_{ij} are not general, since they are columns of a rotation matrix with only 3 degrees of freedom [103]. Repeated application of Equations (2.2) and (2.3) leads to the form of a general projective transformation:

$$\begin{aligned} x &= \frac{X'}{Z'} = \frac{p_{11}X + p_{12}Y + p_{13}}{p_{31}X + p_{32}Y + p_{33}} \\ y &= \frac{Y'}{Z'} = \frac{p_{21}X + p_{22}Y + p_{23}}{p_{31}X + p_{32}Y + p_{33}}. \end{aligned} \tag{2.4}$$

This transformation has 8 degrees of freedom (DOF), despite having 9 parameters p_{ij} — any one parameter $p_{ij} \neq 0$ can arbitrarily be set to $p_{ij} = 1$ by multiplying both numerator and denominator with $1/p_{ij}$. Such a transformation, and equally any projective transformation from a space of dimensionality n into a space of the same dimensionality n, is sometimes called a *homography*.

2.2.1 Homogeneous Coordinates

Equation (2.4) can be expressed by a single, linear matrix transformation such that

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \tag{2.5}$$

or

$$\mathbf{x} = \mathsf{\mathbf{P}}\,\mathbf{X}, \tag{2.6}$$

if the convention is adopted that

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_1/x_3 \\ x_2/x_3 \end{pmatrix}. \tag{2.7}$$

This 3-vector representation of a point is known as *homogeneous coordinates*. Its main advantage is the fact that, using homogeneous coordinates, a projection can be expressed by a single matrix multiplication, which hides the nonlinearity inherent in projection and is therefore handy for computational purposes. For this reason homogeneous coordinates will be used throughout the remainder of this thesis, unless otherwise stated.

In homogeneous coordinates any finite two-dimensional point $\mathbf{x} = (x, y)^\mathsf{T}$ can be expressed as the triplet $\mathbf{X} = (X, Y, Z)^\mathsf{T}$ with $Z \neq 0$. The conversion between the two is

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = k \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.8}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix}. \tag{2.9}$$

From Equations (2.8) and (2.9) it is clear that the homogeneous representation $\mathbf{X}$ is only defined up to an arbitrary scale factor $k \neq 0$; only the ratio of homogeneous coordinates is significant. We also see from Equation (2.9) that in the limit $Z \to 0$ a point at infinity can be expressed quite naturally as $\mathbf{X} = (X, Y, 0)^\mathsf{T}$, compare also Section 2.4.2. Any non-singular matrix $\mathsf{P} \in \mathbb{R}^{3\times3}$ forms a valid projective transformation with eight degrees of freedom (see above).

The group of projective transformations discussed above contains several subgroups. These are discussed in the next sections, going from the more special to the more general.

2.2.2 The Euclidean Group

Equation (2.6) describes a Euclidean transform if

$$\mathsf{P}_{\text{eucl}} = k \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} = k \left(\begin{array}{cc|c} \multicolumn{2}{c|}{\mathsf{R}} & \mathsf{t} \\ \hline 0 & 0 & 1 \end{array} \right), \tag{2.10}$$

where $\mathsf{R} \in \mathbb{R}^{2\times2}$ is an orthogonal matrix, i. e.

$$\mathsf{R}\mathsf{R}^\mathsf{T} = \mathsf{R}^\mathsf{T}\mathsf{R} = \mathsf{I}_2. \tag{2.11}$$

It is easy to show that all orthogonal matrices describe either rotations ($\det(\mathsf{R}) = 1$) or reflections ($\det(\mathsf{R}) = -1$). The usual parameterisations for a rotation or reflection are

$$\mathbf{R}_{\mathrm{rot}} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{2.12}$$

$$\mathbf{R}_{\mathrm{refl}} = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ \sin(\alpha) & -\cos(\alpha) \end{pmatrix}. \tag{2.13}$$

The Euclidean transformation therefore has 3 degrees of freedom (the angle of rotation α and the vector of translation $\mathbf{t} = (t_x, t_y)^\mathsf{T}$), and it is easy to see that all transformations of this type form a group. Compare Figure 2.2(a) on Page 24 for examples of all possible Euclidean transformations.

2.2.3 The Similarity Group

The similarity group is a generalisation of the Euclidean group through the addition of a uniform scale-factor s to the matrix of rotation or reflection $\mathbf{R}$. Equation (2.10) becomes

$$\mathbf{P}_{\mathrm{sim}} = k \begin{pmatrix} s \cdot r_{11} & s \cdot r_{12} & t_x \\ s \cdot r_{21} & s \cdot r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} = k \left(\begin{array}{cc|c} \multicolumn{2}{c|}{s \cdot \mathbf{R}} & \mathbf{t} \\ \hline 0 & 0 & 1 \end{array} \right). \tag{2.14}$$

Consequently, a similarity transformation has 4 degrees of freedom. It is again easy to see that all similarity transformations form a group. Figure 2.2(b) on Page 24 gives examples of similarity transformations.

2.2.4 The Affine Group

The affine group is derived from the similarity group through the inclusion of anisotropic scaling and skew. This introduces two additional degrees of freedom, resulting in 6 degrees of freedom altogether. An affine transformation has the matrix

$$\mathbf{P}_{\mathrm{aff}} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} \tag{2.15}$$

where $\det(\mathbf{P}_{\mathrm{aff}}) \neq 0$. Skew alone can be described by a matrix

$$\mathbf{P}_{\mathrm{skew}} = \begin{pmatrix} 1 & a_x & 0 \\ a_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.16}$$

where a_x and a_y describe skew in x-direction (i. e. parallel to the x-axis) and y-direction respectively. For $a_x = -a_y$ this also describes a rotation around the origin and isotropic scaling; the effect of skew can conversely be created by a suitable combination of rotations and anisotropic scaling. Figure 2.2(c) on Page 24 gives examples of affine transformations, in particular skew in y-direction.

2.2.5 The Projective Group

The projective group finally can be derived from the affine group by introducing so-called perspective skew in the x- and y-direction. This has also been called projective shear or chirp and keystoning. This is simply the full matrix in Equation (2.5), or

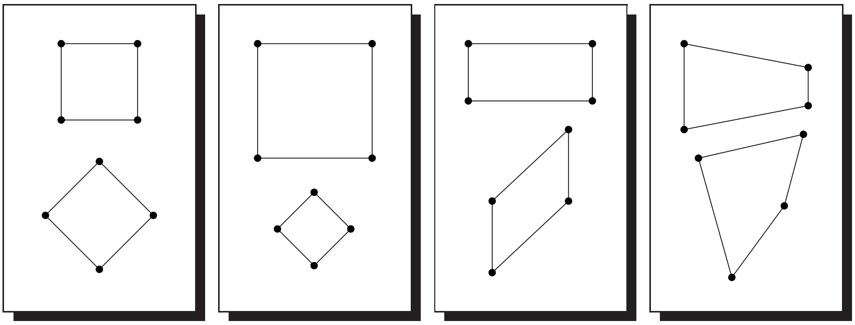
$$\mathbf{P}_{\text{proj}} = \mathbf{P}_{\text{aff}}\mathbf{P}_{\text{proj skew}} \tag{2.17}$$

where the projective skew alone can be parametrised as

$$\mathbf{P}_{\text{proj skew}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_x & b_y & 1 \end{pmatrix} \tag{2.18}$$

if b_x and b_y describe projective skew in x-direction (i. e. symmetric around the x-axis) and y-direction respectively. An example of projective shear in one or both directions can be seen in Figure 2.2(d).

Figure 2.2 and Table 2.1 give an overview over the projective group and its subgroups as well as some invariant features.



(a) Rotation and translation. **(b)** Isotropic scaling. **(c)** Anisotropic scaling and skew. **(d)** Projective skew.

Figure 2.2: *Visual effects of different group-actions: (a) Euclidean, (b) Similarity, (c) Affine, (d) Projective.*

Group	DOF	Matrix	Invariant properties
Projective Group	8	$\begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}$	• cross-ratio (ratio of ratios of collinear lengths) • concurrency and collinearity • order of contact • tangent discontinuities and cusps
Affine Group	6	$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix}$	• ratio of lengths of collinear or parallel segments (e. g. midpoints) • ratio of areas • linear combinations of vectors • parallelism
Similarity Group	4	$k\begin{pmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$	• ratio of lengths • angles
Euclidean Group	3	$k\begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$	• lengths • areas

Table 2.1: *Common subgroups of the projective group and their geometric properties. Groups lower in the table inherit from groups higher in the table (but the converse is of course not true). See also [103, introduction].*

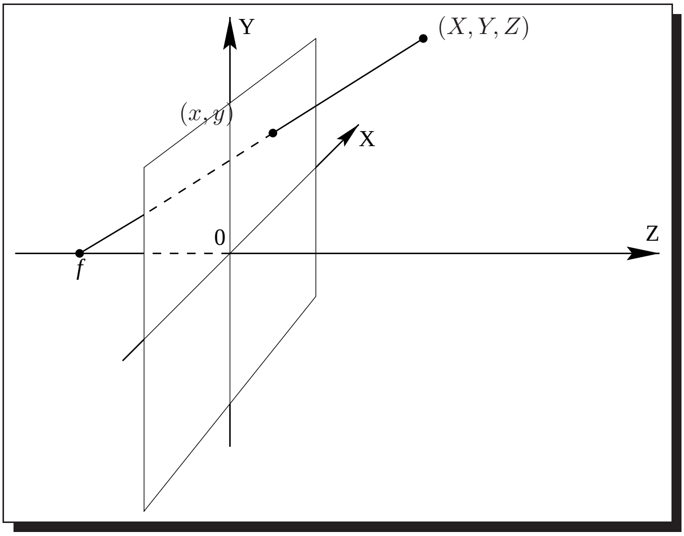


Figure 2.3: *Generic camera Model.*

2.3 Camera Models

This section describes the four camera models used in this thesis, namely the weak perspective camera in Section 2.3.1, the affine camera in Section 2.3.2, the projective camera in Section 2.3.4 (preceded by a short description of the perspective and constrained perspective camera models in Sections 2.3.3 and 2.3.5), and what I call the quasi-calibrated camera in Section 2.3.6 — the most realistic and therefore the preferred model for most applications discussed later. These models are all useful approximations of real cameras for certain applications, and each section gives examples of such applications. Section 2.3.7 finally discusses the limits of all these linear models when compared to real, nonlinear cameras. This section is in its approach complementary to a good discussion of camera models in the Appendix of [103].

The discussion is based on the simple model of a pinhole-camera depicted in Figure 2.3. Note the small difference in the placement of the origin between Figure 2.1 on Page 19 and Figure 2.3. The former is called a *viewer-centred* coordinate system, while the latter is called *image-centred* [68]. It is easy to see from Figure 2.3 that the projection from arbitrary homogeneous world-coordinates $\mathbf{X} = (X, Y, Z, 1)^\mathsf{T}$

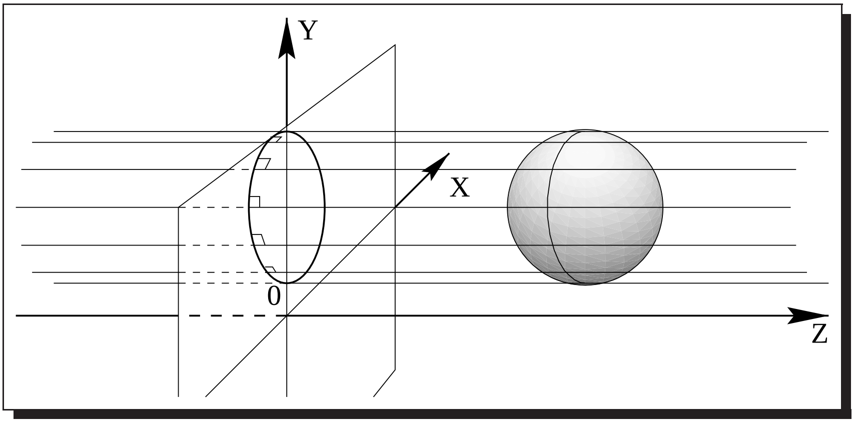


Figure 2.4: *The weak perspective camera.*

onto homogeneous image coordinates $\mathbf{x} = (kx, ky, k)^\mathsf{T}$ is given by

$$\mathbf{x} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{pmatrix} \mathbf{X}. \tag{2.19}$$

Different values for f (which is often taken to be the focal length, from which it takes its name) lead to different camera models; we distinguish the two cases $f = \infty$ described in Sections 2.3.1 and 2.3.2 and $f \neq \infty$ described in Sections 2.3.3 and 2.3.4.

2.3.1 The Weak Perspective Camera

The weak perspective camera is derived from Equation (2.19) in the limit $f \to \infty$. This means that all rays are parallel to each other and orthogonal to the image plane, as illustrated in Figure 2.4. In addition to this projection, the image plane can undergo an arbitrary Euclidean transformation (see Equation (2.10)).

This model describes the case of a calibrated camera viewing a planar object in a plane parallel to the image plane, and at a known distance. Only the object's position and orientation within that plane is assumed unknown. This setup is sometimes found in inspection tasks, where a calibrated camera is installed at a known distance above a conveyor-belt which carries flat objects with a fixed

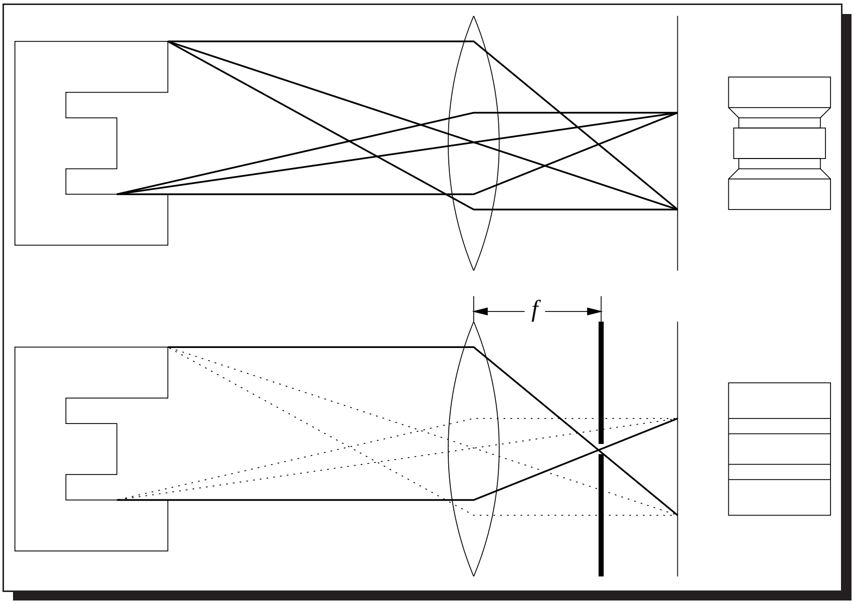


Figure 2.5: *Normal lens (top) and telecentric lens (bottom).*

orientation towards the camera (namely lying on the belt). If the distance between the camera and the planar object (and therefore the object's size in the image) is not known, it is customary to replace the Euclidean transformation of the image plane used above by a similarity transformation according to Equation (2.14). The resulting model is often called *scaled orthographic* projection.

Special precautions have to be taken when applying this model to objects that are neither planar nor parallel to the image plane. Telecentric lenses (as seen in Figure 2.5) can be used and give a very good approximation of this model. The size of the object is, however, limited by the diameter of the front lens, which has to be bigger than the object. In practice the model of a weak perspective camera is often used whenever the change in depth within the object is small compared to the object's distance from the camera. Since *small* is often taken to mean a difference in size of an order of magnitude or more, this can usually only be achieved with telephoto-lenses; an extreme example might be images of (stellar) constellations taken through a telescope.

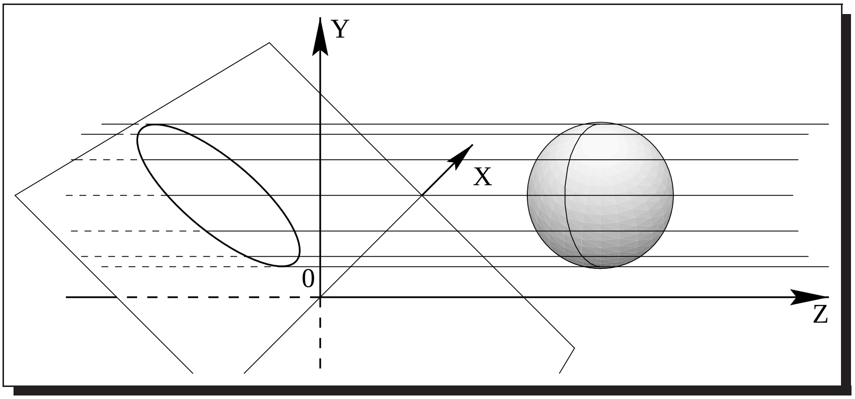


Figure 2.6: *The affine camera. Movement of the image plane around the origin (plus scaling) is equivalent to an affine transformation of the image plane.*

Allowing an arbitrary object to freely change its orientation in 3D will usually result in changes in the object's appearance which cannot be modelled by a Euclidean or similarity transformation. For planar objects, these changes can be modelled by an affine transformation of the image plane (compare Equation (2.15) and the affine camera described in the next section). For arbitrary, non-planar, fully 3-dimensional objects this can become arbitrarily complex, and cannot normally be described by a transformation of the image plane. Note, however, that in both cases the resulting effect is entirely due to changes in the object's orientation relative to the camera; it is often possible to recover completely the object's orientation from its weak perspective image, which is not possible for any of the other models discussed below (with the exception of the quasi-calibrated camera under certain restrictions).

2.3.2 The Affine Camera

The affine camera, like the weak perspective camera in Section 2.3.1, assumes $f \to \infty$. However, the image plane can now undergo an arbitrary, unknown 2D affine transformation. This is illustrated in Figure 2.6 by a movement of the image plane around the origin, which together with scaling is equivalent to an affine transformation. For planar objects in front of an affine camera, the result

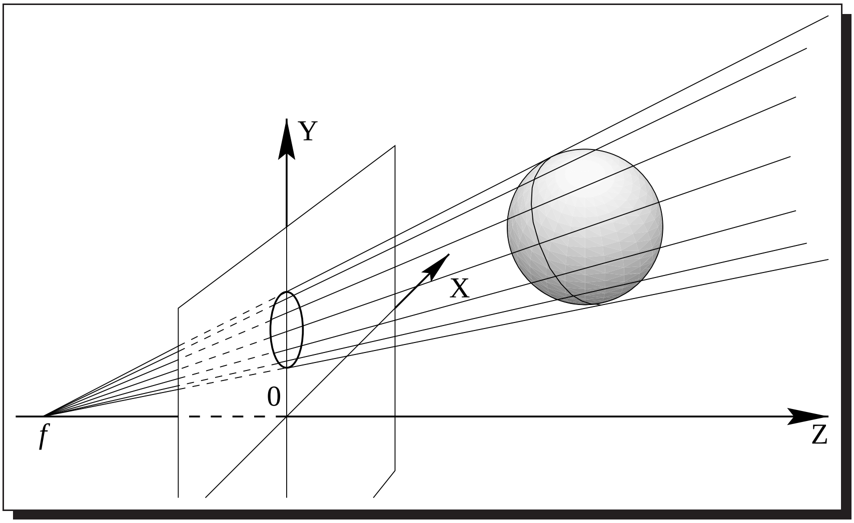


Figure 2.7: *The perspective camera.*

of rotating the image plane is equivalent to rotating the object. This means that it is not possible anymore to infer the object's orientation from its image (as was possible with the weak perspective camera), since it is not clear whether any distortions are due to rotations of the object or of the image plane.

The affine camera can be used to model an uncalibrated CCD-camera under restrictions which are otherwise unchanged from that of a weak perspective camera (i. e. the change in depth within the object is small compared to the object's distance from the camera); the additional degrees of freedom introduced by the use of an affine are used to approximate the unknown camera parameters, in particular if the camera's sensor is not orthogonal to the camera's optical axis.

2.3.3 The Perspective Camera

The perspective camera or pin-hole camera depicted in Figure 2.7 is the linear camera model which most closely resembles the real cameras used in computer vision. Here f is the distance between the pin-hole and the image plane; this corresponds to the distance between a camera's lens and the image plane for real cameras. This distance is therefore also called the focus-setting. For a camera focused at infinity this is equivalent to the camera's focal length. More generally, for a camera focused at a distance p and with focal length F this is

$$f = \frac{pF}{p - F}. \tag{2.20}$$

In addition to the conic projection onto the image plane, the image plane itself can be subject to an arbitrary affine transformation. Since this is equivalent to a movement of the image plane around the origin (and subsequent scaling operation), it corresponds well to the usual sources of mal-calibration in real cameras: a sensor-array which is slightly tilted or displaced, a lens which is not exactly centred, an unknown aspect-ratio and an unknown focus-setting f (and therefore overall scale).

The model's main drawback, and the reason it is not often used in (uncalibrated) computer vision, is its comparative complexity due to the fact that perspective projections do not form a group — a perspective projection of a perspective projection is not necessarily a perspective projection. This can be avoided when using the projective camera model described next.

2.3.4 The Projective Camera

The projective camera is similar to the perspective camera described above. The only difference is that the image can undergo an arbitrary projective transformation (instead of an affine transformation). This has the advantage of improved simplicity over the perspective camera (from a mathematicians point of view), since projective transformations form a group. It also models the process of taking images of images. This has e. g. been used to deal with shadows [89, 154], see Figure 2.8. The use of a projective camera model for this application is however only necessary if both an object and its shadow are considered valid representations of the object, and this ability is also one of the model's main disadvantages — its inherent inability to distinguish between the image of an object and the image of its shadow, at least from the outline alone.

A problem with all the models discussed so far is that a number of assumptions which are sensible for real cameras are not easily incorporated into any of the above camera models. This has given rise to what I call the constrained perspective and the quasi-calibrated or "sensible" camera model, described in the next two sections.

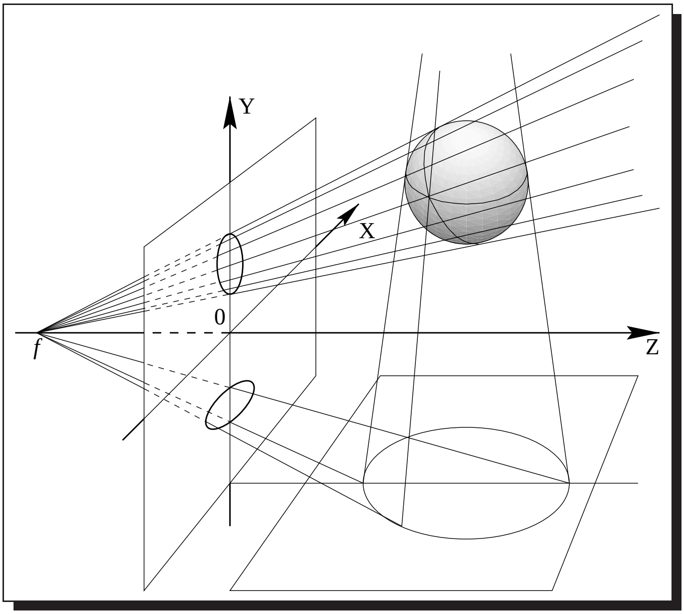


Figure 2.8: *The projective camera. It is not possible to distinguish between the object and its shadow from the images of their respective outlines.*

2.3.5 The Constrained Perspective Camera

This is essentially a perspective camera as described in the previous section, but with the added constraint that the image was taken by a human or otherwise known operator from an ordinary perspective, and at a roughly known orientation — i. e. we know which side of the image is up, and the horizontal and vertical direction within the image are roughly known. This is true for almost all images which we usually encounter and can provide rather strong constraints on possible solutions as we will see in Sections 5–7.

2.3.6 The Quasi-Calibrated Camera

The “sensible” or quasi-calibrated camera, my preferred camera model for most of the applications discussed later on, is also called natural camera in [87].

Using a calibrated camera means that all internal camera-parameters — the image coordinate scale factors $(s_x, s_y)^\mathsf{T}$, the principal point $(t_x, t_y)^\mathsf{T}$, and the focal length f — as well as all external parameters (position and orientation of the camera) — are known with high precision.

A quasi-calibrated camera, in this context, means a camera where only a rough approximation for these values exist: the focal length as printed on the lens (or simply an educated guess), the scale factors as found in the camera's manual, the image centre as principal point. While these values will not, as a rule, be very accurate, they will certainly be within sensible bounds. It is possible to collect all these parameters into a matrix of internal camera-parameters

$$\mathbf{P}_{\text{camera}} = \begin{pmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1/f \end{pmatrix} \tag{2.21}$$

This is basically the same matrix as given in [103].

In addition it is also often possible to make a few generic assumptions about the external camera parameters, in particular the height above ground (about head-high, some 1.6 m–1.8 m), roll-angle (usually accurate to within a few degree) and pitch-angle (horizon somewhere in the image) which can additionally constrain possible interpretations of the image scene. The effects of the choice of camera model will be discussed in detail in Sections 5–7.

2.3.7 Real Cameras

Of course all six models given above are only approximations of real cameras. They all have in common that they only attempt to model linear effects. However, real cameras suffer from several nonlinear effects. These range from comparatively simple nonlinear (barrel or pincushion) distortions [139] to complex effects dependent on the particular wavelength. Although in my experience good lenses will not suffer much from any of these problems up to a field of view of about $40\,^\circ$, it is none the less advisable to check for any of theses problems and correct for them, if necessary. Algorithms can be found e. g. in [139]; [14] uses a very nice approach in keeping with this thesis (minimising vanishing point dispersion), although the actual implementation is in my opinion flawed. The need to correct for nonlinear distortions can make the notion of uncalibrated cameras, which have become quite fashionable since Faugeras published his landmark article in 1992 [44], seem less appealing.

2.4 Points and Lines

We saw in Section 2.2 that the two-dimensional point $\mathbf{p} = (x, y)^\mathsf{T}$ can be expressed in homogeneous coordinates as a triplet $\mathbf{P} = (X, Y, Z)^\mathsf{T} = k(x, y, 1)^\mathsf{T}$ (compare Equation (2.8)). If we define a line as the set of all points for which the equation

$$aX + bY + cZ = k(ax + by + c) = 0 \tag{2.22}$$

holds, we can write this line as a 3-vector $\boldsymbol{\ell}$ with

$$\boldsymbol{\ell} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{2.23}$$

and the equation that specifies all points $\mathbf{P}$ on the line as

$$\boldsymbol{\ell}^\mathsf{T}\mathbf{P} = \mathbf{P}^\mathsf{T}\boldsymbol{\ell} = 0. \tag{2.24}$$

A line $\boldsymbol{\ell}$ which passes through two points $\mathbf{P}_1$ and $\mathbf{P}_2$ satisfies $\boldsymbol{\ell}^\mathsf{T}\mathbf{P}_1 = 0$ and $\boldsymbol{\ell}^\mathsf{T}\mathbf{P}_2 = 0$. Therefore $\boldsymbol{\ell}$ can be calculated as

$$\boldsymbol{\ell} = \mathbf{P}_1 \times \mathbf{P}_2 \tag{2.25}$$

where $\times$ denotes the cross-product.

2.4.1 Duality

Writing the line $\boldsymbol{\ell}$ as an homogeneous 3-vector makes apparent the duality between points and lines in plane projective geometry — points and lines cannot be distinguished from Equation (2.24). It is in fact possible for any result derived for points to be applied to lines and vice versa; this will for example be used in Section 2.6 when introducing the crossratio.

Another example is the calculation of the intersection $\mathbf{P}$ of two lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$. This is the dual problem to finding the line through two points in Equation (2.25), and the intersection of the two lines is therefore given by

$$\mathbf{P} = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2. \tag{2.26}$$

It should, however, be noted that, although the structure is the same for both points and lines, this is not necessarily the case for the individual parameters of a transformation. If A describes the transformation from one plane Π onto a second plane Π', i. e. a point $\mathbf{x}$ is transformed into another point $\mathbf{x}'$ as

$$\mathbf{x}' = \mathsf{A}\mathbf{x} \tag{2.27}$$

so is the transformation from a line $\boldsymbol{\ell}$ on Π onto a line $\boldsymbol{\ell}'$ on Pi' given by the inverse of its transpose $\mathsf{A}^{-\mathsf{T}}$, it is

$$\boldsymbol{\ell}' = \mathsf{A}^{-\mathsf{T}}\boldsymbol{\ell} \tag{2.28}$$

as can be seen from

$$\boldsymbol{\ell}'^{\mathsf{T}}\mathbf{x}' = (\mathsf{A}^{-\mathsf{T}}\boldsymbol{\ell})^{\mathsf{T}}(\mathsf{A}\mathbf{x}) = \boldsymbol{\ell}^{\mathsf{T}}\mathsf{A}^{-1}\mathsf{A}\mathbf{x} = \boldsymbol{\ell}^{\mathsf{T}}\mathbf{x} = 0 \tag{2.29}$$

2.4.2 Special Points and Lines

We will now discuss several points and lines of particular interest. We can see from Equations (2.8) and (2.9) on Page 21, which described the conversion between Euclidean (image) and homogeneous coordinates, that not every homogeneous coordinate corresponds to an image coordinate. We have already mentioned that the set of points $(X, Y, 0)^{\mathsf{T}}$ with $X^2 + Y^2 > 0$, which cannot be mapped onto (finite) image coordinates using Equation (2.9); these points are customarily treated as points at infinity (in the direction indicated by X and Y). This makes the point $(0, 0, 0)^{\mathsf{T}}$ the only point in homogeneous coordinates without a well-defined counterpart in image coordinates; it is customary to exclude $(0, 0, 0)^{\mathsf{T}}$ from the set of homogeneous coordinates.

Conversely, for lines in homogeneous coordinates the special case is the line specified as $(0, 0, c)^{\mathsf{T}}$. It is easy to see that this has to be the line at infinity, since all points at infinity $(X, Y, 0)^{\mathsf{T}}$ lie on this line, it is $(0, 0, c)\,(X, Y, 0)^{\mathsf{T}} = 0$. Note that there is only one line at infinity, since homogeneous coordinates are invariant to uniform scaling; it is again customary to exclude the line $(0, 0, 0)^{\mathsf{T}}$ from the set of homogeneous coordinates. The line $(a, b, 0)^{\mathsf{T}}$, on the other hand, is simply the line through the origin whose normal-vector is given by $(a, b, k)^{\mathsf{T}}$.

2.4.3 Vanishing Points and Lines

Additional distinguished points and lines are vanishing points and vanishing lines respectively; these can be interpreted as projective transformations of points and lines at infinity (in 3D). Lines that are parallel in the world (and could therefore be said to intersect at a point at infinity) will not, in general, appear parallel

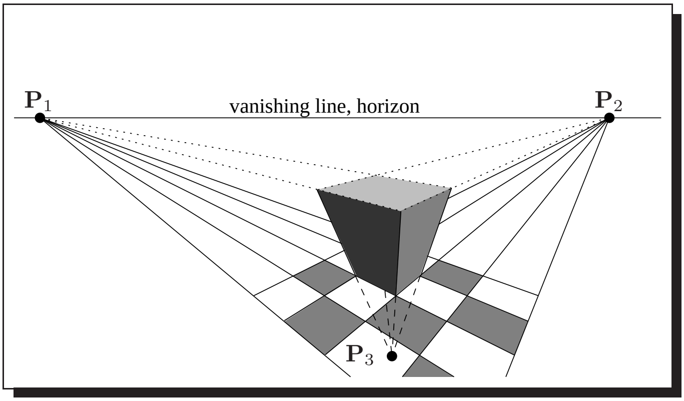


Figure 2.9: *Vanishing points and vanishing line.*

under a projective transformation. Since order of contact is a projective invariant (compare Table 2.1) this means that the lines' original intersection at infinity will be projected to a new location generally not at infinity. This point is called the lines' vanishing point since it is the point where infinitely long lines seem to vanish when viewed in an image. Figure 2.9 shows examples of vanishing points.

Two such sets of lines, both parallel to the same plane, but not parallel to each other, define two separate vanishing points which in turn define a line. This line is called the plane's vanishing line, since in an image it is the locus where the plane seems to vanish. An example for a vanishing line is given in Figure 2.9. An image can contain several vanishing lines, each associated with a different plane in 3D. Additional vanishing lines (not shown in the figure) go through the point-pairs $(\mathbf{P}_1, \mathbf{P}_3)$ and $(\mathbf{P}_2, \mathbf{P}_3)$.

2.4.4 The Horizon

One vanishing line of particular interest is the line customarily termed the horizon. It is formed by two vanishing points corresponding to different directions parallel to the ground-plane (compare Figure 2.9). The name horizon is adopted here although it is somewhat misleading, since the horizon encountered in the real world is not a line, but rather part of a conic (a hyperbola, to be precise). And although the difference between the two is quite small in most images (usually within a pixel), one should keep this difference in mind, since it can become arbitrarily large under certain conditions.

2.5 Conics

While the last section described important point and line-based features, such as the duality between points and lines and special points and lines such as the vanishing point and vanishing line, these are by no means the only geometric entities that are easily integrated into projective geometry. Another important geometric structure are conics, which are self-similar under projective transformations. This section describes their embedding into homogeneous coordinates and projective geometry.

A conic curve in the plane, i. e. an ellipse, parabola, or hyperbola, is defined by the quadratic homogeneous expression

$$AX^2 + BXY + CY^2 + DXZ + EYZ + FZ^2 = 0. \tag{2.30}$$

Note that this homogeneous equation has 6 parameters, but only 5 degrees of freedom, as only the ratio of parameters in Equation (2.30) is significant. It can be written as

$$\mathbf{P}^{\mathsf{T}}\mathbf{C}\mathbf{P} = 0 \tag{2.31}$$

with a symmetric matrix $\mathbf{C} \in I\!R^{3\times3}$ and vector $\mathbf{P} \in I\!R^3$ as follows

$$\mathbf{C} = \begin{pmatrix} A & \frac{B}{2} & \frac{D}{2} \\ \frac{B}{2} & C & \frac{E}{2} \\ \frac{D}{2} & \frac{E}{2} & F \end{pmatrix} \tag{2.32}$$

$$\mathbf{P} = (X, Y, Z)^{\mathsf{T}}.$$

If a point $\mathbf{P}$ transforms as $\mathbf{p} = \mathbf{AP}$ under the action of a matrix of transformation $\mathbf{A} \in I\!R^{3\times3}$, so is the corresponding conic $\mathbf{C}$ transformed as

$$\mathbf{c} = \mathbf{A}^{-\mathsf{T}}\mathbf{C}\mathbf{A}^{-1}. \tag{2.33}$$

The resulting matrix $\mathbf{c} \in I\!R^{3\times3}$ is again a symmetric matrix of the form given in Equation (2.32) and therefore a conic, it is $\mathbf{c}^{\mathsf{T}} = (\mathbf{A}^{-\mathsf{T}}\mathbf{C}\mathbf{A}^{-1})^{\mathsf{T}} = \mathbf{A}^{-\mathsf{T}}\mathbf{C}\mathbf{A}^{-1} = \mathbf{c}$. It can indeed be shown that all conics are projectively equivalent, compare for example the appendix of [103].

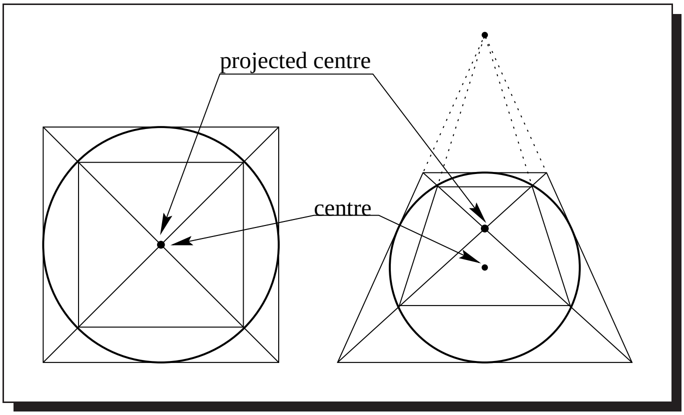


Figure 2.10: *A conic's midpoint is not invariant to projective transformation.*

It should be noted that a conic's midpoint is of course *not* invariant under general projective transformations, Figure 2.10 illustrates this effect. This is due to the fact that the ratio of lengths is not an invariant under projective transformations, as stated in Table 2.1 on Page 24.

2.5.1 Duality

Conics are so called self-dual figures. This means that they can be considered to be both the locus of points as well as the envelope of tangent-lines. The latter view is commonly referred to as a line-conic. The line-conic's equation is

$$\mathsf{L} = |\mathsf{C}|\mathsf{C}^{-1} \tag{2.34}$$

where $|\mathsf{C}|$ is the determinant of C and $\boldsymbol{\ell}^{\mathsf{T}}\mathsf{L}\boldsymbol{\ell} = 0$ for all tangent-lines $\boldsymbol{\ell}$; it transforms as

$$\mathsf{l} = \mathsf{ALA}^{\mathsf{T}}. \tag{2.35}$$

2.5.2 Pole and Polar of a Conic

For any point $\mathbf{P}$ outside a conic there are two tangents from $\mathbf{P}$ to the conic C as illustrated in Figure 2.11. The two points of tangency define a line $\boldsymbol{\ell}_{\mathbf{P}}$ which is called the *polar* of point $\mathbf{P}$ with respect to the conic C. Conversely, the point $\mathbf{P}$ is

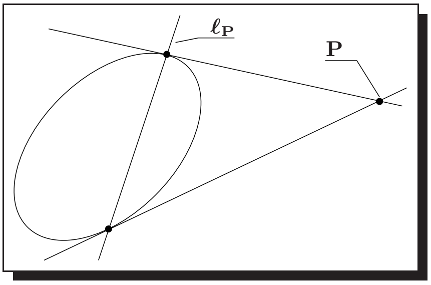


Figure 2.11: *Pole and polar of a conic.*

called the *pole* of line $\boldsymbol{\ell}_{\mathbf{P}}$ with respect to the conic C. The interrelation between pole, polar, and conic is given by

$$\boldsymbol{\ell}_{\mathbf{P}} = \mathsf{C}\mathbf{P}. \tag{2.36}$$

Note that using Equation (2.36) it is also possible to calculate the polar which corresponds to a pole inside the conic, although the notion of tangents is not defined for these points (the polar corresponding to a point on the conic is the tangent to the conic at that point).

2.6 The Crossratio

We can see from Table 2.1 on Page 24 that neither length nor the ratio of length is preserved under projective transformation. Luckily there is one feature which is preserved and this is the crossratio, or ratio of ratios of collinear lengths. The crossratio is indeed by far the most important projective invariant, and Mundy and Zisserman ventured in [103] that likely all invariant properties of a geometric configuration can ultimately be interpreted in terms of some number of crossratio constructions.

2.6.1 Definition

The crossratio of four collinear points $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ is defined with respect to Figure 2.12, usually [72, 103, 138] as

$$\mathrm{cr}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}) = \frac{\overline{\mathbf{AC}}}{\overline{\mathbf{BC}}} \cdot \frac{\overline{\mathbf{BD}}}{\overline{\mathbf{AD}}} = \frac{C-A}{C-B} \cdot \frac{D-B}{D-A}, \tag{2.37}$$

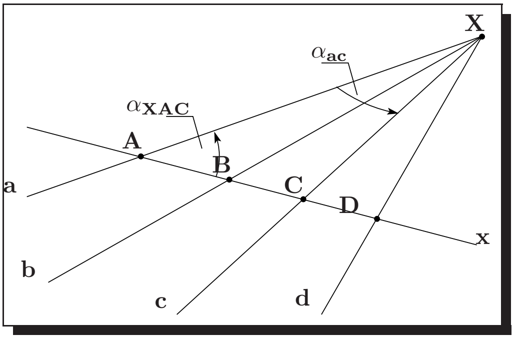


Figure 2.12: *The crossratio. Capital letters denote points, and small letters denote lines.*

where $\overline{\mathbf{AC}}$ is the directed Euclidean distance between point **A** and point **C**, and $\{A, B, C, D\}$ are scalars representing the corresponding Euclidean position of each point along the line relative to an arbitrarily chosen origin. That the crossratio is indeed a projective invariant can easily be proven by direct substitution and cancellation of the resulting non-zero factor in each term [103, 138, 146].

2.6.2 The Six Crossratios of Four Points

The form of Equation (2.37) suggests that the value of the crossratio of four collinear points depends on the order of these points. There are $4! = 24$ possible permutations, suggesting the existence of 24 different values for the crossratio cr. In fact there are at most 6 distinct values of the crossratio within these 24 permutations, as can easily be shown [103, 138, 146]. These are

$$\left\{ \mathrm{cr}, 1-\mathrm{cr}, \frac{1}{\mathrm{cr}}, 1-\frac{1}{\mathrm{cr}}, \frac{1}{1-\mathrm{cr}}, \frac{\mathrm{cr}}{1-\mathrm{cr}} \right\}. \tag{2.38}$$

For a general set of four points $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ these 6 functions of cr will indeed produce six distinct values. However, if the four points are related in a suitable way, some of the six crossratios formed from Equation (2.38) may be equal. A complete catalogue of these special cases can be calculated by equating cr with each of the other expressions and solving for cr; following [138] the three special cases are:

1. $\{1, 1, 0, 0, \infty, \infty\}$: two of the four points coincide.

2. $\{-1, -1, 1/2, 1/2, 2, 2\}$: this case is called harmonic separation, see Section 2.8.
3. $\{-\omega, -\omega, -\omega, -\omega^2, -\omega^2, -\omega^2\}$ with $\omega = e^{2\pi i/3}$: The four points, not all of which can have real parameters, form an equianharmonic tetrad [138, Page 48].

It might be interesting to note that in every case all the values of the crossratio occur the same number of times in the full set of 24: 4 times in the general case, 8 in case 1 and 2, and 12 in case 3.

The existence of 6 distinct values for the crossratio dependent on the order of points could possibly cause problems for some applications where the order is not known, especially since projective transformations do preserve order only up to a cyclic permutation. A possible invariant which does not depend on the order can be calculated as [103, 138]

$$I(\mathrm{cr}) = \frac{(\mathrm{cr}^2 - \mathrm{cr} + 1)^3}{\mathrm{cr}^2(\mathrm{cr} - 1)^2}. \tag{2.39}$$

The application of this equation allows one to use the crossratio without the need to determine the order of points beforehand, as well as in cases where a cyclic permutation of the points due to some projective transformation occurred.

2.6.3 The Crossratio of Four Lines

Since points and lines are dual, there must also be a crossratio of four coincident lines (the dual of collinearity is incidence at a point). Such a set of coincident lines is called a pencil. Where in the case of four points on a line the points could be described by a single parameter position on the line, in the case of four coincident lines it is possible to uniquely describe each line by its gradient. One possible formulation for the crossratio of four lines $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$ is in terms of the angles between the lines [103] (see also Figure 2.12):

$$\mathrm{cr}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \frac{\sin(\alpha_{\mathbf{ac}})}{\sin(\alpha_{\mathbf{bc}})} \cdot \frac{\sin(\alpha_{\mathbf{bd}})}{\sin(\alpha_{\mathbf{ad}})}. \tag{2.40}$$

Any fifth line $\mathbf{x}$ not coincident with the other four will intersect the pencil at four points of intersection $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$. These intersections form in turn a crossratio on the line, as illustrated in Figure 2.12. It is easy to prove that the two crossratios are identical, $\mathrm{cr}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \mathrm{cr}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, using only the law of sines, that is

$$\frac{\sin(\alpha_{\mathbf{ac}})}{\overline{\mathbf{AC}}} = \frac{\sin(\alpha_{\mathbf{XAC}})}{\overline{\mathbf{XC}}}. \tag{2.41}$$

and similarly for the other angles, compare Figure 2.12. Substituting these terms in Equation (2.40) and cancelling out some of the terms one immediately gets (2.37).

2.6.4 Alternative Formulations of the Crossratio

Equations (2.37) and (2.40) are not particularly convenient for the actual computation of the crossratio, since it is always possible that one of the points $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ is an ideal point at infinity, requiring the introduction of special cases when computing the Euclidean distance used in Equation (2.37). Similar problems exist for the calculation of the angles in Equation (2.40) if the pencil's intersection is a point at infinity, in which case all the lines are parallel.

Therefore the crossratio is often calculated using the equation

$$\mathrm{cr}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}) = \frac{|\mathbf{ACX}|}{|\mathbf{BCX}|} \cdot \frac{|\mathbf{BDX}|}{|\mathbf{ADX}|} = \frac{|\mathbf{acx}|}{|\mathbf{bcx}|} \cdot \frac{|\mathbf{bdx}|}{|\mathbf{adx}|} = \mathrm{cr}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}), \tag{2.42}$$

where $|\mathbf{ACX}|$ is the determinant of a matrix formed by the three column-vectors $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{X}$. The point $\mathbf{X}$ as well as the line $\mathbf{x}$ can be chosen arbitrarily as long as none of the matrices in Equation (2.42) become singular[2]. This means in particular that the point $\mathbf{X}$ must not be collinear with the points $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$, and the line $\mathbf{x}$ must not be coincident with the lines $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$. A proof that Equation (2.42) is indeed equivalent to Equations (2.37) and (2.40) can e. g. be found in [72]. An alternative proof is outlined below:

The determinant $|\mathbf{ACX}|$ can be written as

$$|\mathbf{ACX}| = (\mathbf{A} \times \mathbf{C})^{\mathsf{T}}\mathbf{X}. \tag{2.43}$$

We have seen in Section 2.4 that $\mathbf{A} \times \mathbf{C} = k\mathbf{x}$ if $\mathbf{x}$ is the line through $\mathbf{A}$ and $\mathbf{C}$. Furthermore

$$\mathbf{A} \times \mathbf{C} = \|\mathbf{A}\|_2 \|\mathbf{C}\|_2 \sin(\alpha_{\mathbf{AC}}) \frac{\mathbf{x}}{\|\mathbf{x}\|_2} \tag{2.44}$$

and consequently

$$|\mathbf{ACX}| = (\mathbf{A} \times \mathbf{C})^{\mathsf{T}}\mathbf{X} = \|\mathbf{A}\|_2 \|\mathbf{C}\|_2 \sin(\alpha_{\mathbf{AC}}) \|\mathbf{X}\| \cos(\alpha_{\mathbf{xX}}). \tag{2.45}$$

Since the homogeneous points $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ are collinear, the line $\mathbf{x}$ goes through all of them, resulting in similar equations for the other three combinations. From there it is easy to see that Equation (2.42) is indeed equivalent to Equation (2.40). Equation (2.45) suggests that $\mathbf{X} \propto \mathbf{x} \propto (\mathbf{A} \times \mathbf{C})$ is a reasonable choice for $\mathbf{X}$ — we will see in Section 4.5 that this is in fact not so.

[2]Note that any of the matrices will of course become singular if the two points (lines) used are identical. However, if the four points (lines) are distinct from each other then either all the matrices will be singular, or none, depending solely on $\mathbf{X}$.

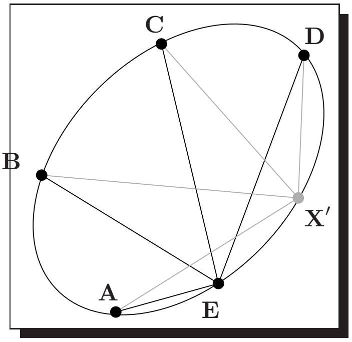


Figure 2.13: *A conic can be defined using the crossratio.*

2.6.5 Conics and the Crossratio

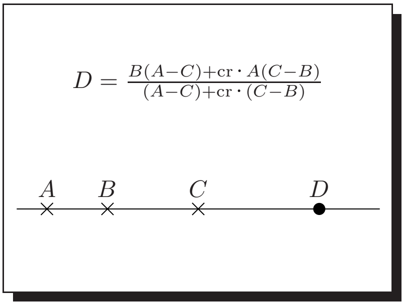
Conics can be defined with respect to the crossratio: take four points **A**, **B**, **C**, **D**, no three of which are collinear. Draw a pencil of lines from an arbitrary point **E** to all four fixed points. The locus of the vertices $\mathbf{X}'$ of all pencils with constant crossratio is a conic, compare [103, p. 490] and Figure 2.13.

2.6.6 Projective Coordinates

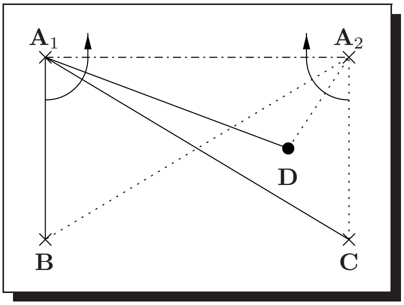
The cross-ratio can be used to define projective coordinates. This is easy to see in the case of projective coordinates on the line as in Figure 2.14(a). In the Euclidean case two points on a line define a coordinate system where one point is the origin and the second point's position relative to the first determines the scale factor. However, scale (or, more precisely, length) is not a projective invariant. We therefore need to know a third point's position along the line. Only then is it possibly to describe every other point's position on the line uniquely by its crossratio with the three base-points. Conversely, it is also possible, given three base-points and the crossratio, to compute the Euclidean position of the forth point on the line by solving Equation (2.37) for this position; it is without loss of generality (w. l. o. g.):

$$D = \frac{B(A-C) + \mathrm{cr}\cdot A(C-B)}{(A-C) + \mathrm{cr}\cdot(C-B)}. \tag{2.46}$$

The same is possible in the plane. Euclidean coordinates in the plane consist of


$D = \frac{B(A-C)+\mathrm{cr}\cdot A(C-B)}{(A-C)+\mathrm{cr}\cdot(C-B)}$
A
B
C
D

(a) Projective coordinates on the line.



(b) Projective coordinates on the plane.

Figure 2.14: *Projective coordinates on the line and plane.*

an origin and two orthonormal vectors which define two independent directions as well as a scale-factor. Again, scale (length) is not a projective invariant, nor is orthogonality. In order to construct a projective coordinate system of the plane 4 points are needed, no three of which are collinear. Several approaches have been used to define projective coordinates on the plane using 4 reference-points; however, they are all equivalent since 5 points only have two functionally independent invariants, corresponding to the planes 2 degrees of freedom.

One often used approach singles out one reference-point and draws lines from there to the other 3 reference points, resulting in 3 coincident lines. Any fifth point would add a forth line, and the crossratio of four lines would uniquely determine the *ray* on which the forth point is located. Selecting a different base-point we end up with a similar construction, giving a second ray. The point where the two rays intersect is the fifth point (compare Figure 2.14(b)).

Another way to uniquely describe a points' position on the plane is to solve for the transformation that projects the base-points into a fixed position and determine any other point's position within this frame. This approach is discussed in the next section.

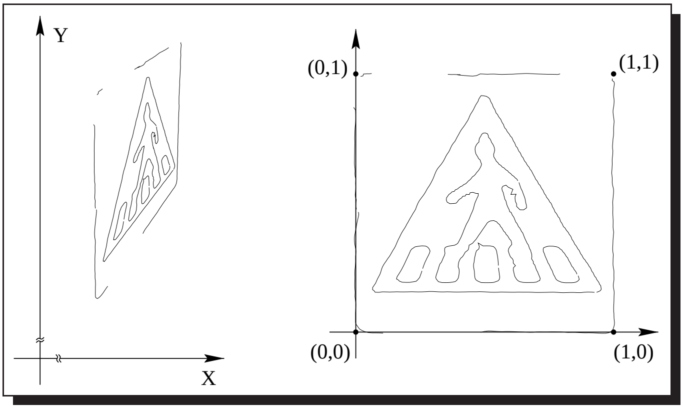


Figure 2.15: *The outline of a traffic-sign as seen in the image (left) and after transformation into a canonical frame (right).*

2.7 Canonical Frames

We have seen in the last section that the position of 4 points on a plane allows us to uniquely specify the position of each additional point on that plane independently of any projective transformation applied to that plane. These projective coordinates are, however, not a particularly intuitive way to describe most image features, and it is therefore often desirable to find some quasi-Euclidean representation instead. This can easily be done if the entire object plane is transformed in such a way that the projective coordinates' four reference-points (or lines) are transformed onto four points (or lines) in a fixed position, the so called canonical frame.

2.7.1 Motivation

Three possible uses for canonical frames are described below; all have been used within this thesis.

Verification and Recognition: Canonical frames have traditionally been used for verification and recognition purposes [127, 129][9], as they allow for the direct comparison of features within a quasi-Euclidean framework. Possible comparisons range from direct comparison of pixel-positions to the calculation of higher order features of non-algebraic curves, where they considerably reduce the number of derivatives required (from up to seventh order to only first or second order [158]). It is noteworthy that all frames are mathematically equivalent in the absence of errors. This is however not the case for practical applications, as we will see in, e. g., Section 4.4.2.

Recognition within a canonical frame can be implemented as simple as a comparison with different models, and as complicated as the extraction of invariants or the application of an index-function. Examples for both uses are given in Section 7.

Backprojection: Normally, image pixels or low-level features like edgels or lines are projected into the canonical frame in order to test a hypothesis. If instead the known contour (or other features) of a hypothesis are projected from the frame back into the image we talk about backprojection. This is often done for verification directly in the image, recognising the fact that in practice, and in the presence of errors, all canonical frames are *not* equal.

Another use is the prediction of additional image features from a hypothesis, which, if found, would lend additional credibility to the particular hypothesis. This is used in Section 5.

Fitting: Using the canonical frame to fit higher order structures to low-level features (mainly edgels) allows us to enforce additional constraints not easily enforced within the image. The basic idea here is to find the transformation from the image into the canonical frame which minimises the error between the transformed image features and a structure in the canonical frame. It is then possible to invert the transformation in order to calculate the structure's position within the image.

This approach is of course only useful in the presence of errors (fitting would not be required otherwise) and therefore discussed in more detail in Section 4.4.2, where it is used.

2.7.2 Commonly used Frames

In the case of 4 points, commonly used frames include e. g. the unit square

$$\left\{(0,0,1)^{\mathsf{T}},(1,0,1)^{\mathsf{T}},(0,1,1)^{\mathsf{T}},(1,1,1)^{\mathsf{T}}\right\} \tag{2.47}$$

and the triangle of reference and unit point

$$\left\{(1,0,0)^{\mathsf{T}},(0,1,0)^{\mathsf{T}},(0,0,1)^{\mathsf{T}},(1,1,1)^{\mathsf{T}}\right\}, \tag{2.48}$$

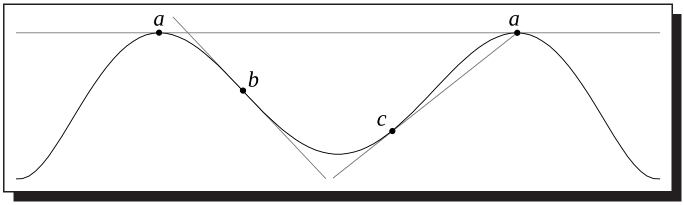


Figure 2.16: *Some distinguished points: a. Bitangent-points, b. inflection, c. casttangent-point.*

where the first three points are called the vertices of the triangle of reference, while the last point is called the unit point [103, 138]. Other canonical frames are often based on the object's appearance in the Euclidean world. Figure 2.15 shows an example where the unit square is also the object's natural frame. In the absence of measurement errors, all canonical frames are of course mathematically equivalent.

2.7.3 Commonly used Image Features

A canonical frame describes a particular instance of a projectively transformed plane (or, more general, space). In order to define this particular instance, it is necessary to determine the projective transformation between the original space and its representation within the canonical frame. In the case of planar structures as discussed here, this transformation has 8 degrees of freedom, and it is clear that the position within the image and frame of any structure which fixes at least 8 degrees of freedoms can be used to describe the transformation between the two planes. In practice, however, this structure will nearly exclusively be made up of points and lines. The reason for this is that the use of points and lines leads to a set of linear equations (compare Section 2.7.4), while higher order algebraic structures generally do not. Also, points (edgels) and lines (straight edgel chains) are the most basic image features found.

Three different types of points are commonly used in computer vision:

1. Corners of grey-level discontinuities as found by corner detectors.
2. Intersections of higher-order algebraic features, usually lines fit to grey-level discontinuities.
3. Distinguished points. These are points on a curve which are easily distinguishable from all the other points on the curve by order of contact, which

is a projective invariant (compare Table 2.1). Examples are points of bitangency or inflections (see Figure 2.16), which are easily identified using only up to first or second order derivatives.

Once a point and a curve are identified it is often easy to create a number of additional distinguished points. Examples are rays cast from one distinguished point and tangent to the curve at a second point, so called casttangents. This second point, the casttangent-point, is another distinguished point. Another example is the intersection of a line through two distinguished points and the curve (compare Item 2), again generating extra distinguished points (although of course collinear with the first two).

2.7.4 Calculation of Canonical Frames

Finding the transformation $\mathsf{A} \in I\!R^{3\times 3}$ such that $N \geq 4$ image points $\mathbf{X}_i$ are mapped onto the corresponding frame-points $\mathbf{x}_i$ is easily done by solving the equation

$$\mathsf{AX} = \mathsf{xk} \tag{2.49}$$

for A. $\mathsf{X}, \mathsf{x} \in I\!R^{3\times N}$ are two matrices, where each column represents one image or frame point respectively, and $\mathsf{k} \in I\!R^{N\times N}$ is a diagonal matrix of scale-factors accounting for the fact that the overall scale of each homogeneous coordinate can be chosen arbitrarily. The resulting equations are of the form:

$$\begin{array}{rcl} a_{11}X_i + a_{12}Y_i + a_{13}Z_i & = & k_i x_i \\ a_{21}X_i + a_{22}Y_i + a_{23}Z_i & = & k_i y_i \\ a_{31}X_i + a_{32}Y_i + a_{33}Z_i & = & k_i z_i \end{array} \tag{2.50}$$

and it is always possible to eliminate k_i. We assume that w. l. o. g. $z_i = 1$ (this is obviously not the case for the triangle of reference (2.48), but the underlying principle is the same) and get

$$\begin{array}{rcl} a_{11}X_i + a_{12}Y_i + a_{13}Z_i & = & a_{31}X_i x_i + a_{32}Y_i x_i + a_{33}Z_i x_i \\ a_{21}X_i + a_{22}Y_i + a_{23}Z_i & = & a_{31}X_i y_i + a_{32}Y_i y_i + a_{33}Z_i y_i. \end{array} \tag{2.51}$$

Furthermore, since the overall scale of **A** is arbitrary we can choose w. l. o. g. $a_{33} = 1$ and get, for $N = 4$

$$\begin{pmatrix} X_1 & Y_1 & Z_1 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & Z_1 & -X_1y_1 & -Y_1y_1 \\ X_2 & Y_2 & Z_2 & 0 & 0 & 0 & -X_2x_2 & -Y_2x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & Z_2 & -X_2y_2 & -Y_2y_2 \\ X_3 & Y_3 & Z_3 & 0 & 0 & 0 & -X_3x_3 & -Y_3x_3 \\ 0 & 0 & 0 & X_3 & Y_3 & Z_3 & -X_3y_3 & -Y_3y_3 \\ X_4 & Y_4 & Z_4 & 0 & 0 & 0 & -X_4x_4 & -Y_4x_4 \\ 0 & 0 & 0 & X_4 & Y_4 & Z_4 & -X_4y_4 & -Y_4y_4 \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{pmatrix} = \begin{pmatrix} Z_1x_1 \\ Z_1y_1 \\ Z_2x_2 \\ Z_2y_2 \\ Z_3x_3 \\ Z_3y_3 \\ Z_4x_4 \\ Z_4y_4 \end{pmatrix} . \qquad (2.52)$$

The existence of this linear system ensures the existence and uniqueness of a solution for **A** given four point correspondences, provided that no three of the points are collinear[103].

A more elegant implementation would use a singular value decomposition (SVD) approach to calculate the eigenvector to the smallest (zero!) eigenvalue of the system

$$\begin{pmatrix} & & & & \vdots & & & & \\ X_i & Y_i & Z_i & 0 & 0 & 0 & -X_i\frac{x_i}{z_i} & -Y_i\frac{x_i}{z_i} & -Z_i\frac{x_i}{z_i} \\ 0 & 0 & 0 & X_i & Y_i & Z_i & -X_i\frac{x_i}{z_i} & -Y_i\frac{x_i}{z_i} & -Z_i\frac{x_i}{z_i} \\ & & & & \vdots & & & & \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{pmatrix} = \mathbf{0}. \qquad (2.53)$$

This way it is not necessary to single out any particular $a_{ij} = 1$. In addition, such an approach will also work in the presence of errors, and given $N \neq 4$ point-pairs. SVD is, however, computationally more expensive.

The same basic approach can be used when 4 lines are given, no three of which are coincident, solving for $\mathbf{A}^{-\mathsf{T}}$ instead (compare Equation (2.28)). Rearranging Equation (2.28) to read $\mathbf{L} = \mathbf{A}^{\mathsf{T}}\boldsymbol{\ell}\mathbf{k}$ it is even possible to combine the equations for points and lines into one system of equations; the equation for a line-pair corresponding to Equation (2.52) and w. l. o. g. $C_i \neq 0$ would read

$$\begin{pmatrix} & & & & \vdots & & & \\ a_i & 0 & a_iA_i & b_i & 0 & b_iA_i & c_i & 0 \\ 0 & a_i & a_iB_i & 0 & b_i & b_iB_i & 0 & c_i \\ & & & & \vdots & & & \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{pmatrix} = \begin{pmatrix} \vdots \\ c_iA_i \\ c_iB_i \\ \vdots \end{pmatrix} . \tag{2.54}$$

Closed form solutions for combinations with higher-order algebraic forms, e. g. conics, are unfortunately not as easy to find.

2.7.5 Semi-Frames

So far, we have always assumed that a canonical frame fixes all of a planar transformation's 8 degrees of freedom. For many applications, however, this is not necessary. Imagine a frame which solely consists of a set of horizontal lines with fixed distance from the origin. Neither an anisotropic scaling factor in x direction, nor any skew nor translation in that direction would change the appearance of the frame. It would therefore be sufficient to solve for a 5 degrees of freedom transformation and arbitrarily fix the remaining 3 degrees. This could, e. g., look like

$$\mathbf{A} = k \begin{pmatrix} 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} . \tag{2.55}$$

2.8 Symmetry under Projective Transformations

Symmetry plays a crucial role in everyday life; many man-made objects possess symmetry, and this has been exploited in vision systems [55, 102, 120, 128], [5, 9]. It is therefore reasonable to ask what happens to symmetry under a general projective transformation.

Two points are said to possess symmetry with respect to a line, the axis of symmetry, if the line is the perpendicular bisector of the line segment joining the two points. They are said to be symmetric with respect to a third point, the centre of symmetry, if the third point bisects the line joining the points [65]. Symmetry is therefore an inherently non-projective quality, since it depends on the invariance of midpoints and, in the case of axial symmetry, angles[3]. It is nonetheless possible to identify some properties of symmetry that do not rely on the invariance of angles or the ratio of lengths and this will be done in Section 2.8.1. We will then see in Section 2.8.2 that these properties describe a particular type of projective transformation, namely a plane harmonic homology. Section 2.8.3 finally shows that a plane harmonic homology maintains its structure under arbitrary projective transformations, and therefore is the most appropriate description of symmetry under projective transformations.

2.8.1 Properties of Symmetry

Symmetry can be described in terms of the transformation $\mathsf{H} \in I\!R^{3\times 3}$ which transforms one side $\mathbf{x}$ into its symmetric complement $\mathbf{x}'$, we get the necessary condition

$$\begin{aligned} & \mathsf{H}\mathbf{x} = k'\mathbf{x}' \\ & \mathsf{H}\mathbf{x}' = k\mathbf{x} \\ \Longrightarrow \quad & \mathsf{HH}\mathbf{x}' = kk'\mathbf{x}' \\ \Longrightarrow \quad & \mathsf{HH} = kk'\mathsf{I}_3. \end{aligned} \tag{2.56}$$

It is always possible to scale H so that $kk' = 1$. H is called an involution or automorphism.

Equation (2.56) is only the necessary condition for symmetry; additional restrictions are needed in order to ensure that H represents a symmetry transformation. In the case of axial symmetry the transformation H obviously leaves the axis itself unchanged; in other words, the axis forms a set of fixed points or united points. In the case of point symmetry, the centre of symmetry is left unchanged. It turns out on closer inspection that axial symmetry has another fixed point at infinity, in the direction perpendicular to the axis, while point-symmetry has a fixed line at infinity.

In the projective case, the condition above reduces to that of a fixed point and a line of fixed points in arbitrary position, as long as the point is not located on the line. Interestingly, this means that there is no intrinsic difference between axial symmetry and point symmetry in a projective space.

Finally, symmetry is characterised by the fact that the line segment joining $\mathbf{x}$ and its symmetric complement $\mathbf{x}'$ is bisected by the axis of symmetry (in the case of axial symmetry) or the point of symmetry (in the case of point symmetry).

[3]For an affine transformation the concept of skewed symmetry can be defined.

The ratio of collinear lengths is, however, not a projective invariant; the closest approximation within a projective space would be a constraint on the crossratio which any pair of symmetric points forms with its midpoint and the point at infinity: the crossratio is always $\text{cr} = -1$.This is called harmonic separation (see Section 2.6.2).

To identify a transformation that could take the role of symmetry within a projective space, we are looking for a transformation with a line of fixed points and an additional fixed point not on that line which fulfils Equation (2.56) and with the required crossratio of $\text{cr} = -1$. We will see in the next section that a plane harmonic homology has all these attributes.

2.8.2 Homologies

One condition on a transformation that could take the role of symmetry within a projective space was the existence of a line of fixed points and an additional fixed point not on that line. We can see from Equation (2.6) on Page 20 that any 2D projective transformation of a homogeneous vector $\mathbf{x} \in I\!R^3$ can be expressed as its multiplication with a matrix $\mathsf{P} \in I\!R^{3\times 3}$. This matrix will in general have 3 eigenvectors $\mathbf{x}_i \in I\!R^3$, $\mathbf{x}_i \neq \mathbf{0}$ and corresponding eigenvalues λ_i, such that

$$\mathsf{P}\mathbf{x}_i = \lambda_i \mathbf{x}_i. \tag{2.57}$$

Since homogeneous coordinates are invariant to overall scale this means that a general projective transformation will have at least 3 points which remain fixed under this particular transformation[4]. Depending on the multiplicity of the λ_i there are 6 distinctive cases. These are discussed in more detail in [138].

Here, we are only interested in cases that produce a line of united points, that is an eigenvalue of geometric multiplicity 2. There are two and only two such cases, the first case with one degenerate eigenvalue λ_0 of algebraic and geometric multiplicity 2 and one simple eigenvalue λ_2 and the second case of one degenerate eigenvalue λ_0 of algebraic multiplicity 3 and geometric multiplicity 2. These cases are customarily called the *plane homology* and the *special plane homology* respectively, and the corresponding set of united points is formed by a line $\boldsymbol{\ell} \in I\!R^3$ of united points and a single united point $\mathbf{v} \in I\!R^3$, also called the vertex. Of these two only the plane homology is of interest to us, since in the case of the special plane homology the line of united points and the single united point coincide, $\mathbf{v}^\mathsf{T}\boldsymbol{\ell} = 0$.

[4]It is possible that two of these points — or even all three — coincide. It is also possible that more than 3 such points exist. A simple example for the latter is $\mathsf{P} = \mathsf{I}_3$, the identical transformation.

According to [138] any plane homology $\mathsf{H} \in I\!R^{3\times 3}$ can always be parameterised as

$$\mathsf{H} = \mathsf{I}_3 + \frac{1-\mathrm{cr}}{\mathrm{cr}} \cdot \frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} \tag{2.58}$$

as long as $\mathbf{v}^\mathsf{T}\boldsymbol{\ell} \neq 0$, that is the homology is not a special plane homology. Accordingly, any plane homology with crossratio $\mathrm{cr} = -1$ can always be parametrised as

$$\mathsf{H} = \mathsf{I}_3 - 2\,\frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}}. \tag{2.59}$$

This is called a *plane harmonic homology*. By construction, any plane harmonic homology has a line of united points $\boldsymbol{\ell}$ and a single united point $\mathbf{v}$ as well as $\mathrm{cr} = -1$. It also satisfies the necessary condition for symmetry (2.56), it is

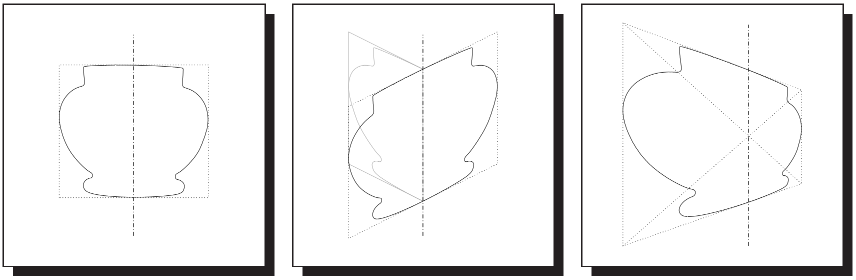
$$\begin{aligned}
\mathsf{HH} &= \mathsf{I}_3 - 4\,\frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} + 4\,\frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} \\
&= \mathsf{I}_3 - 4\,\frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} + 4\,\frac{\mathbf{v}(\boldsymbol{\ell}^\mathsf{T}\mathbf{v})\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}(\boldsymbol{\ell}^\mathsf{T}\mathbf{v})} \\
&= \mathsf{I}_3 - 4\,\frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} + 4\,\frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} \\
&= \mathsf{I}_3.
\end{aligned} \tag{2.60}$$

This shows that Equation (2.59) really describes a transformation as outlined in Section 2.8.1. It also describes Euclidean symmetry: $\boldsymbol{\ell} = (a,b,c)^\mathsf{T}$ and $\mathbf{v} = (a,b,0)^\mathsf{T}$ describe axial symmetry; $\mathbf{v} = (x,y,z)^\mathsf{T}$ and $\boldsymbol{\ell} = (0,0,1)^\mathsf{T}$ describe point symmetry.

2.8.3 Symmetry under Projection

Under an arbitrary projective transformation $\mathsf{P} \in I\!R^{3\times 3}$ with $\mathsf{P}\mathbf{v} = \widetilde{\mathbf{v}}$ and $\mathsf{P}^{-\mathsf{T}}\boldsymbol{\ell} = \widetilde{\boldsymbol{\ell}}$ the plane harmonic homology H transforms as $\widetilde{\mathsf{H}} = \mathsf{PHP}^{-1}$. That $\widetilde{\mathsf{H}}$ is again of the form (2.59) can be seen from:

$$\begin{aligned}
\widetilde{\mathsf{H}} &= \mathsf{PHP}^{-1} \\
&= \mathsf{PI}_3\mathsf{P}^{-1} + \frac{1-\mathrm{cr}}{\mathrm{cr}} \cdot \frac{\mathsf{P}\mathbf{v}\boldsymbol{\ell}^\mathsf{T}\mathsf{P}^{-1}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} \\
&= \mathsf{PP}^{-1}\mathsf{I}_3 + \frac{1-\mathrm{cr}}{\mathrm{cr}} \cdot \frac{\mathsf{P}\mathbf{v}(\mathsf{P}^{-\mathsf{T}}\boldsymbol{\ell})^\mathsf{T}}{\mathbf{v}^\mathsf{T}\mathsf{P}^\mathsf{T}\mathsf{P}^{-\mathsf{T}}\boldsymbol{\ell}} \\
&= \mathsf{I}_3 + \frac{1-\mathrm{cr}}{\mathrm{cr}} \cdot \frac{\widetilde{\mathbf{v}}\widetilde{\boldsymbol{\ell}}^\mathsf{T}}{\widetilde{\mathbf{v}}^\mathsf{T}\widetilde{\boldsymbol{\ell}}}.
\end{aligned} \tag{2.61}$$



(a) Symmetry with respect to a line. **(b)** Symmetry under affine transformation. **(c)** Symmetry under projective transformation.

Figure 2.17: *Symmetry under transformations.*

It follows (from the previous section) that any symmetry-transformation is of the form given in Equation (2.59), and (from Equation (2.61)) that all symmetry-transformations keep this form under an arbitrary projective transformation. Conversely, it is always possible to find a projection matrix $\mathbf{P}$ such that $\mathbf{PHP}^{-1}$ describes a symmetry. A plane harmonic homology therefore describes the form of a symmetry-transformation under an arbitrary projective transformation. Figure 2.17 gives examples for symmetry under different transformations.

2.9 The Gaussian Sphere

Persons used to the Euclidean plane generally find it difficult to envisage the projective plane with its ideal points at infinity, but without the usual invariance of angles and length (or at least ratio of length). So it is only understandable that other models have been proposed.

2.9.1 The Ray-Space Model

Perhaps the most widespread model is that of a ray space, a space of coincident rays embedded into a three-dimensional space $\mathbb{R}^3$, as described in e. g. [103]. In this space each ray — all rays emanate from a common origin — represents a projective point. Only the direction of the ray matters in this model. In projective space the crossproduct of two points defines a line, see Section 2.4. Consequently in ray space a line is represented by the crossproduct of two rays — a plane through the

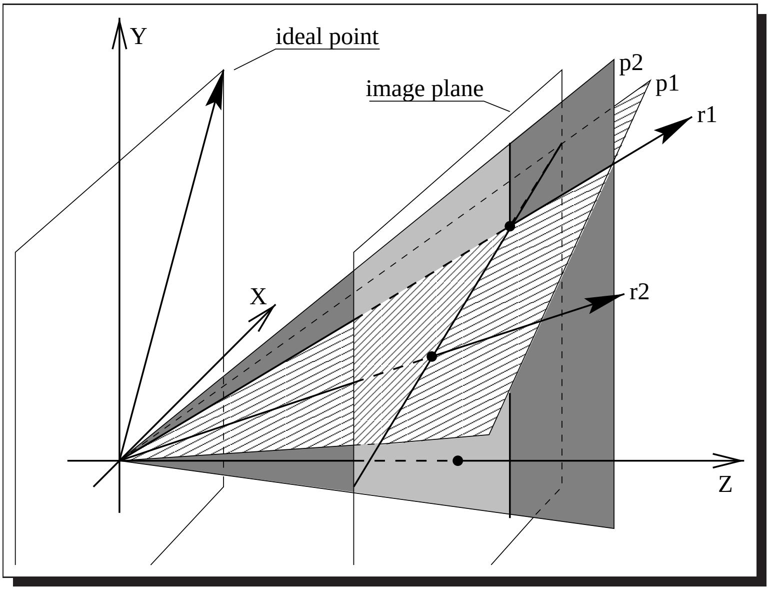


Figure 2.18: *Ray-space model. Each ray corresponds to a point in the image plane. Two rays span a plane, two planes define a ray. Ideal points correspond to a ray parallel to the image plane.*

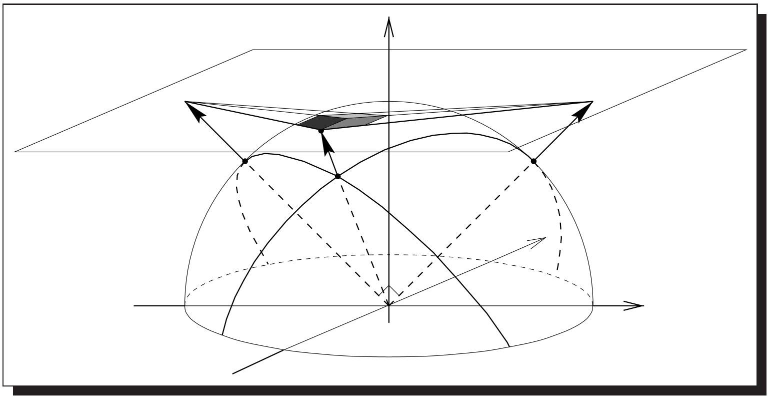


Figure 2.19: *Gaussian sphere model.*

origin spanned by the two rays. Conversely, the crossproduct of two planes is the ray common to both planes, representing a projective point. This is illustrated in Figure 2.18.

The process of image formation is modelled as the intersection of all theses rays and planes with a plane *not* through the origin (note the similarity between this model and the viewer centred camera model in Section 2.3). Consequently, ideal points with respect to this image plane are represented by rays parallel to the image plane, while a plane through the origin and parallel to the image plane represents the ideal line. It is easy to see from this model that the distinction between ideal points and other points is really quite arbitrary, since the image plane can be chosen randomly. The same model can be used to describe the mapping from one plane onto a second plane, where the origin of the rays is the centre of projection, and it is possible to model arbitrary relationships between two planes by a composition of rotations and anisotropic scaling in $\mathbb{R}^3$, compare [103].

2.9.2 The Gaussian Sphere Model

A slightly different model, but based on the one above, is the model of a Gaussian or unit sphere. A projective point corresponds to the point where a line from the sphere's origin to the projective point intersects the sphere, and a projective line corresponds to the great circle that is the intersection between the sphere and a plane through the sphere's origin and the projective line (compare Figure 2.19). Note that any line through the origin will intersect the sphere at 2 points on opposite sides of the sphere. It is therefore customary to avoid this ambiguity by considering only a semi-sphere.

It is obvious that the ray-space model can easily be converted into the Gaussian sphere model by calculating the intersections between rays and planes on the one side and the sphere on the other. Easier still, if a ray is expressed as $k(x, y, z)^\mathsf{T}$, the corresponding point on the Gaussian sphere is simply its normalisation into a unit-vector, $\frac{1}{\sqrt{x^2+y^2+z^2}}(x, y, z)^\mathsf{T}$.

2.9.3 Calibrated Cameras and Gaussian Sphere

The Gaussian sphere model has some particularly convenient features when dealing with calibrated cameras. It is then possible to calculate coordinates on the

Gaussian sphere $(x, y, z)^\mathsf{T}$ from image coordinates $(X, Y)^\mathsf{T}$ as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{\sqrt{X^2 + Y^2 + f^2}} \begin{pmatrix} X \\ Y \\ f \end{pmatrix} \tag{2.62}$$

where f is the distance between the centre of projection and the image plane. It is often called the camera's focal length, although this is strictly only true for a camera focused at a point at infinity, compare the discussion on Page 30. The beauty of this construction is that directions which are perpendicular in reality will also be perpendicular on the Gaussian sphere (compare the rays pointing to the vanishing points in Figure 2.19), although they are *not* perpendicular in the image[5]. Conversely, assuming that a sufficient number of directions in the image are known to be perpendicular in reality, this can then be used to calibrate an unknown camera, compare Section 6.3.2. In addition, Kanatani [69] showed that using the model in Equation (2.62), which he called N-vectors[6], has several advantages with respect to numerical computations as well as error distribution, compare also Section 4.

[5]The same is of course also true for the ray-space model, into which the Gaussian sphere model can be transformed.

[6]N standing for normalised.

Chapter 3

Probability and Statistics

The most may err as grossly as the few.

John Dryden, Absalom and Achitophel, 1631–1700

3.1 Introduction

Measurements in any discipline are generally encumbered with measurement errors. This is particularly true for image measurements, where a less than ideal imaging-process is followed by a discretisation of the image. It is, on the other hand, a reasonable assumption that knowledge about the accuracy of our measurements is essential when decisions based on these measurements are required. Systems which ignore this are at best cumbersome, requiring the user to fine-tune a generally high number of sometimes obscure parameters; at worst they will often simply fail.

Although this chapter covers the general aspects of statistical properties and error propagation, I will use examples from computer vision throughout this chapter. Virtually all image measurements boil down to measuring edgel positions, possibly with subpixel accuracy. Section 3.2 gives a short introduction into the kinds of errors customarily encountered when dealing with any measurements, as well as some basic concepts used in statistics. The edgel positions are then used to construct higher order structures — contours, line segments, conics, and ever more complex configurations. Section 3.3 describes how the measurement error in image coordinates — or any random variable — is propagated into derived quantities. One of the standard tasks in computer vision is to decide whether some structure derived from image measurements conforms to a given model. Section 3.4 explains how confidence tests, and in particular the χ^2-test can be used as a decision making tool. Section 3.5 finally describes some common probability distributions on the sphere; this is applicable to angles and other measurements with only finite support.

Much of what is said in this chapter can be found e. g. in [29, 100], or [43, pp. 151–164]. Books on photogrammetry [144][1] or [49] can be another rich source of information and inspiration for someone working in computer vision, in particular where error propagation is concerned. An introduction into confidence testing can be found in any textbook on statistics, examples are [81, 145]. Books concerned with statistics on directional data in contrast are much harder to find, the reader is referred to [95, 156].

[1] Note, however, that at least the 4th edition of the Manual of Photogrammetry contains several gross errors.

3.2 Basic Concepts in Statistics

The following gives a short introduction into the basic concepts of statistics, see [81, 100, 145] for more information.

3.2.1 Error Types

This entire chapter would not have been necessary if it were not for the fact that any observation will always contain errors. These errors are traditionally grouped into three categories: random errors, systematic errors, and blunders.

Blunders — or *outliers*, as they are customarily called in computer vision — are gross errors generally due not to the observed process or variable, but to the observer. If at all possible, they should be removed from the set of observations. How to reliably classify outliers is unfortunately still an open question, the reader is referred to [46, 67, 140, 151] for examples from nearly 20 years of outlier removal in computer vision. Particularly en vogue is currently once more a method called RANSAC — Random Sample Consensus — which was introduced in 1981 by Fischler and Bolles [46].

Outliers are ignored in the following unless otherwise stated.

Systematic errors — or *systematic effects*, as they are commonly named in the recent literature — are not really errors in the observations, but rather in the underlying model. It is therefore usually possible to remove or avoid systematic effects if an appropriate model is chosen, and part of Section 4 is dedicated to the process of model-selection. An example of systematic effects often encountered in computer vision are radial distortions of the image due to an imperfect lens, see Section 4.2.1. It is well known how to model this effect (usually by an odd polynomial of the distance to the principal point, compare e. g. [139]), and therefore easy to account for it. This is usually *not* done by incorporating the model of the radial distortion into that of (perspective) projection — which would lead to rather intractable equations — but by correcting the observations for this particular effect. In computer vision, such corrections are often part of (partial) camera calibration.

Random errors are the only kind of effects with which traditional statistics is concerned, although the use of the term “error” is deprecated in modern literature, and the term *statistical properties* used instead. This captures the fact that from a statistical standpoint, observations can be considered samples of an unknown probability distribution of a random variable. Discrepancies between several observations are therefore not due to errors, but simply serve to describe the particular probability distribution. It is statistics' task to gain as much information as possible about this distribution from the observations.

So how can we describe the properties of our unknown probability distribution? A very concise description can often be given by the use of moments, as described in the next section.

3.2.2 Mean and Central Moments

Probability distributions can often be described in terms of their mean and central moments. The population mean of a random variable x, also called first moment or expectation, is denoted by $E(x)$ and is defined (if it exists) as the average value μ_x of the variable over all possible values, weighted by their respective probabilities $P_x \in I\!R$ or probability density function[2] $p_x \in I\!R$, it is

$$E(x) = \mu_x = \sum_{i=1}^{n} x_i P_x(x_i) \tag{3.1}$$

or

$$E(x) = \mu_x = \int_{-\infty}^{\infty} x\, p_x(x)\, \mathrm{d}x \tag{3.2}$$

in the continuous case. Given two random variables x and y and three constants a, b, c, the following rules hold [100]:

$$E(E(x)) = E(x) \tag{3.3}$$

$$E(x+y) = E(x) + E(y) \tag{3.4}$$

$$E(c) = c \tag{3.5}$$

$$E(c \cdot x) = c \cdot E(x) \tag{3.6}$$

$$\Longrightarrow \qquad E(a \cdot x + b) = a \cdot E(x) + b. \tag{3.7}$$

If x and y are independent random variables, it is also true that

$$E(x \cdot y) = E(x) \cdot E(y). \tag{3.8}$$

Note, however, that, in general, $E(x^2) \neq (E(x))^2$.

[2]In the following denoted by pdf.

Central moments, which can be used to describe most pdfs, are expectations with respect to the mean, where the k$^{\text{th}}$ central moment is defined as

$$m_k = E\left((x - E(x))^k\right). \tag{3.9}$$

One particularly important central moment is the second moment or variance σ_x^2. It is:

$$\begin{aligned}
\sigma_x^2 = m_2 = E\left((x - E(x))^2\right) &= E\left((x - \mu_x)^2\right) && (3.10)\\
&= E\left(x^2 - 2x\mu_x + \mu_x^2\right) && (3.11)\\
&= E\left(x^2\right) - 2\mu_x E(x) + \mu_x^2 && (3.12)\\
&= E\left(x^2\right) - \mu_x^2. && (3.13)
\end{aligned}$$

The variance's positive square root $\sigma > 0$ is called standard deviation. Note that Equation (3.13) can easily lead to numeric problems for big values of μ_x^2. The equations corresponding to Equations (3.4)–(3.8) are:

$$\begin{aligned}
\sigma_{x+y}^2 &= \sigma_x^2 + \sigma_y^2 && (3.14)\\
\sigma_c^2 &= 0 && (3.15)\\
\sigma_{c\,\cdot\,x}^2 &= c^2 \cdot \sigma_x^2 && (3.16)\\
\Longrightarrow \qquad \sigma_{a\,\cdot\,x+b}^2 &= a^2 \cdot \sigma_x^2. && (3.17)
\end{aligned}$$

Related to the concept of the variance is that of the cofactor q, which could be viewed as a relative variance. It is

$$q_x^2 = \frac{\sigma_x^2}{\sigma_0^2} \tag{3.18}$$

for a possibly unknown value of σ_0^2, the reference variance.

The Equations (3.2), (3.10), and (3.13) can only be used if the pdf p_x is already known. In order to estimate the pdf from observations alone, we have to approximate the population mean and population variance (and possibly higher order moments) by the sample mean and sample variance. Given N measurements x_i, $(i = 1 \ldots N)$ the sample or empirical mean $\overline{x}$ is defined as the arithmetic mean:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{3.19}$$

It is $E(\overline{x}) = \mu_x$. The sample variance is defined as

$$s_x^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2 \tag{3.20}$$

and it is $E(s_x^2) = \sigma_x^2$. Higher order moments can be approximated similarly.

3.2.3 Normal Distribution

The most important probability distribution, and one uniquely defined by mean and variance, is the normal or Gaussian distribution

$$N(\mu_x, \sigma_x^2) = \frac{1}{\sqrt{2\pi\sigma_x^2}}\, e^{-\frac{1}{2}\frac{(x-\mu_x)^2}{\sigma_x^2}}. \tag{3.21}$$

3.2.4 Multidimensional Extension

The above can easily be extended for multi-dimensional random variables. If $\mathbf{x} \in \mathbb{R}^n$ is a vector of N (not necessarily independent) random variables, so is the expectation simply

$$E(\mathbf{x}) = \boldsymbol{\mu}_\mathbf{x} = \sum_{i=1}^{N} \mathbf{x}_i\, P_\mathbf{x}(\mathbf{x}_i) \tag{3.22}$$

or

$$E(\mathbf{x}) = \boldsymbol{\mu}_\mathbf{x} = \int_{-\infty}^{\infty} \mathbf{x}\, p_\mathbf{x}(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{3.23}$$

in the continuous case, where the summation (or integration) can be performed separately for each vector-element. Note that $P_\mathbf{x}, p_\mathbf{x} \in \mathbb{R}$.

The central moments of order $(k_1 + \cdots + k_n)$ can be calculated as

$$E\left((x_1 - \mu_{x_1})^{k_1} \cdot \;\cdots\; \cdot (x_n - \mu_{x_n})^{k_n}\right). \tag{3.24}$$

Of particular importance are again the second order central moments of $\mathbf{x} \in \mathbb{R}^n$, which can be set up as all combinations between each two elements of the vector. The result can be arranged as a matrix $\mathbf{M}_{\mathbf{xx}} \in \mathbb{R}^{n\times n}$. This matrix is customarily called the matrix of second central moments, the variance-covariance matrix or simply the covariance matrix. It is

$$\mathsf{M}_{\mathbf{xx}} = E((\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}}) = \begin{pmatrix} m_{x_1x_1} & m_{x_1x_2} & \cdots & m_{x_1x_n} \\ m_{x_2x_1} & m_{x_2x_2} & \cdots & m_{x_2x_n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{x_nx_1} & m_{x_nx_2} & \cdots & m_{x_nx_n} \end{pmatrix}$$

$$= \begin{pmatrix} \sigma^2_{x_1} & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_n} \\ \sigma_{x_2x_1} & \sigma^2_{x_2} & \cdots & \sigma_{x_2x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{x_nx_1} & \sigma_{x_nx_2} & \cdots & \sigma^2_{x_n} \end{pmatrix} = \boldsymbol{\Sigma}_{\mathbf{x}}. \qquad (3.25)$$

Note that this is a square symmetric matrix since

$$\sigma_{x_ix_j} = E((x_i-\mu_{x_i})\cdot(x_j-\mu_{x_j})) = E((x_j-\mu_{x_j})\cdot(x_i-\mu_{x_i})) = \sigma_{x_jx_i}. \qquad (3.26)$$

Using Equation (3.24) it is always possible to construct higher order central moments. The equations corresponding to Equations (3.3)– (3.7) and (3.14)– (3.17) are (with random variables $\mathbf{x},\mathbf{y} \in I\!R^n$, constant vectors $\mathbf{b},\mathbf{c} \in I\!R^m$ and a constant matrix $\mathsf{A} \in I\!R^{m\times n}$):

$$E(E(\mathbf{x})) = E(\mathbf{x}) \qquad (3.27)$$
$$E(\mathbf{x}+\mathbf{y}) = E(\mathbf{x}) + E(\mathbf{y}) \qquad (3.28)$$
$$E(\mathbf{c}) = \mathbf{c} \qquad (3.29)$$
$$E(\mathsf{A}\mathbf{x}) = \mathsf{A}E(\mathbf{x}) \qquad (3.30)$$
$$\Longrightarrow \qquad E(\mathsf{A}\mathbf{x}+\mathbf{b}) = \mathsf{A}E(\mathbf{x}) + \mathbf{b} \qquad (3.31)$$

$$\boldsymbol{\Sigma}_{\mathbf{x}+\mathbf{y}} = \boldsymbol{\Sigma}_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{y}} \qquad (3.32)$$
$$\boldsymbol{\Sigma}_{\mathbf{c}} = 0 \qquad (3.33)$$
$$\boldsymbol{\Sigma}_{\mathsf{A}\mathbf{x}} = \mathsf{A}\boldsymbol{\Sigma}_{\mathbf{x}}\mathsf{A}^{\mathsf{T}} \qquad (3.34)$$
$$\Longrightarrow \qquad \boldsymbol{\Sigma}_{\mathsf{A}\mathbf{x}+\mathbf{b}} = \mathsf{A}\boldsymbol{\Sigma}_{\mathbf{x}}\mathsf{A}^{\mathsf{T}}. \qquad (3.35)$$

A cofactor matrix can be defined analogous to Equation (3.18), it is

$$\mathsf{Q}_{\mathbf{x}} = \frac{1}{\sigma_0^2}\boldsymbol{\Sigma}_{\mathbf{x}}. \qquad (3.36)$$

The sample mean $\overline{\mathbf{x}}$ and sample covariance matrix $\mathsf{S}_{\mathbf{x}}$ are defined in analogy to

Equations (3.19) and (3.20) as

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{3.37}$$

$$\mathsf{S}_{\mathbf{x}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^{\mathsf{T}}. \tag{3.38}$$

The n-dimensional normal distribution is given by

$$N(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}_{\mathbf{x}}|}} \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{x}}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})), \tag{3.39}$$

where $|\boldsymbol{\Sigma}_{\mathbf{x}}|$ is the determinant of $\boldsymbol{\Sigma}_{\mathbf{x}} \in I\!R^{n\times n}$. Additional distributions are given in Section 3.4 (the χ^2-distribution, used for testing) and 3.5 (an adaption of the normal distribution to cyclic data), but the normal distribution is by far the most important distribution used in this thesis. Its theoretical and practical importance is due to the *central limit theorem* which states that the sum $\sum_{i=1}^{n} x_i$ of n independent random variables $x_1, \ldots, x_n$ will be asymptotically normally distributed as $n \to \infty$. Normal distributions are encountered very often in practical applications; in particular, random variables that represent *independent* measurements in photogrammetry, geodesy, or surveying are often nearly normally distributed [100]. Another reason for the normal distribution's prominence is its simple form which is completely described by mean and variance. This makes it particularly well suited for the propagation of statistical properties as described in the next section.

3.3 Error Propagation

The idea of error propagation — or propagation of statistical properties — is the following: given a vector of measurements $\mathbf{x} \in I\!R^n$ with pdf $p_{\mathbf{x}}$, and a derived vector $\mathbf{y} \in I\!R^m$ such that

$$\mathbf{g} : \mathbf{x} \to \mathbf{y} = \mathbf{g}(\mathbf{x}) \tag{3.40}$$

find the pdf $p_{\mathbf{y}}$ for $\mathbf{y}$ depending on $\mathbf{x}$ and $p_{\mathbf{x}}$.

An example might make this more transparent: calculating the line $\boldsymbol{\ell}$ passing through the two points $\mathbf{p}_1$ and $\mathbf{p}_2$. Displacing one or both of the points will generally change the position of the line. So if we know that the two points (which could, e. g., be measured image coordinates) are random variables with probability distributions $p_{\mathbf{p}_1}$ and $p_{\mathbf{p}_2}$, it suggests itself to ask what the resulting line's probability distribution $p_{\boldsymbol{\ell}}$ will be.

3.3.1 Principle

For simplicity, let us consider the one-dimensional case first, i. e. random variables $\mathbf{x} = x \in I\!R$ and $\mathbf{y} = y \in I\!R$. We would expect that the probability for any event x' to fall into a small region $\mathrm{d}x$ around x should be equal to the probability of an event y' falling into the corresponding region $\mathrm{d}y$ around y in the limit $\mathrm{d}x \to 0$. This can be written as

$$p_x(x)|\mathrm{d}x| = p_y(y)|\mathrm{d}y|. \tag{3.41}$$

If we assume that the inverse function

$$g^{-1} : y \to x = g^{-1}(y) \tag{3.42}$$

is defined, we can write

$$p_x(x)|\mathrm{d}x| = p_x\left(g^{-1}(y)\right)\left|\frac{\partial g^{-1}(y)}{\partial y}\mathrm{d}y\right| = p_y(y)|\mathrm{d}y| \tag{3.43}$$

or

$$p_y(y) = p_x\left(g^{-1}(y)\right)\left|\frac{\partial g^{-1}(y)}{\partial y}\right|. \tag{3.44}$$

Taking the absolute value in Equations (3.43) and (3.44) ensures the correct sign of $p_y(y)$. This is not a problem, since both $g(x)$ and $g^{-1}(y)$ are monotonic (otherwise they would not be invertible).

The extension to the multidimensional case with $\mathbf{x}, \mathbf{y} \in I\!R^n$ is straightforward [100], it is

$$p_{\mathbf{y}}(\mathbf{y}) = p_{\mathbf{x}}\left(\mathbf{g}^{-1}(\mathbf{y})\right)|\mathbf{J}_{xy}|, \tag{3.45}$$

where $|\mathbf{J}_{xy}| = |\partial\mathbf{x}/\partial\mathbf{y}|$ is the determinant of the Jacobian of the inverse transformation $\mathbf{x} = \mathbf{g}^{-1}(\mathbf{y})$ with respect to $\mathbf{y}$.

The difficulty with the propagation of distributions is that we have to assume the existence of the inverse function. This means in particular that usually no solution is possible for $\mathbf{x} \in I\!R^n$, $\mathbf{y} \in I\!R^m$, $m \neq n$, where, in general, no inverse transformation exists. This is, for example, the case with our example of the line through two points, since it is not possible to infer from the parameters of the line the positions of the points. The necessity to use the inverse function severely limits the usefulness of propagation of statistical properties. And even if Equations (3.44) or (3.45) can be applied, so will the resulting pdf, in general, become arbitrarily complex.

3.3.2 Linear Case

All these problems do not exist if we concentrate on linear functions only, i. e. functions of the form $\mathbf{y} = \mathsf{A}\mathbf{x} + \mathbf{y}_0$ with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y}, \mathbf{y}_0 \in \mathbb{R}^m$, and $\mathsf{A} \in \mathbb{R}^{m\times n}$. Linear functions will, in general, not change the particular type of distribution [100]. What is more, we have already seen from Equations 3.31 and 3.35 that

$$E(\mathbf{y}) = E(\mathsf{A}\mathbf{x} + \mathbf{b}) = \mathsf{A}E(\mathbf{x}) + \mathbf{b} \tag{3.46}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}} = \boldsymbol{\Sigma}_{\mathsf{A}\mathbf{x}+\mathbf{b}} = \mathsf{A}\boldsymbol{\Sigma}_{\mathbf{x}}\mathsf{A}^{\mathsf{T}} \tag{3.47}$$

and in the case of a multidimensional normal distribution, we can directly write down the new distribution[3].

Unfortunately, most interesting functions are not linear. It is therefore necessary to find a way to apply the above equations to any arbitrary, nonlinear function $\mathbf{g}(\mathbf{x})$. This is usually done by approximating $\mathbf{g}(\mathbf{x})$ by a linear function $\mathbf{f}(\mathbf{x})$. Series expansion, and Taylor series expansion[4] in particular, is generally used for this purpose. The next section gives a short introduction into Taylor series expansion and the resulting laws for the propagation of statistical properties.

3.3.3 Explicit Functions

Any C^1 function $\mathbf{y} = \mathbf{g}(\mathbf{x})$ can be written as

$$\mathbf{y} = \mathbf{g}(\mathbf{x}_0 + \Delta\mathbf{x}) = \mathbf{g}(\mathbf{x}_0) + \mathsf{J}_{\mathbf{y}\mathbf{x}_0}\Delta\mathbf{x} + \mathcal{O}_2(\|\Delta\mathbf{x}\|^2). \tag{3.48}$$

In the vicinity of $\mathbf{x}_0$ this can usually be approximated by a linear function $\mathbf{f}(\mathbf{x})$ with

$$\mathbf{y} = \mathbf{g}(\mathbf{x}_0 + \Delta\mathbf{x}) \approx \mathbf{y}' = \mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}) = \mathbf{g}(\mathbf{x}_0) + \mathsf{J}_{\mathbf{y}\mathbf{x}_0}\Delta\mathbf{x} = \mathbf{y}_0 + \mathsf{J}_{\mathbf{y}\mathbf{x}_0}\Delta\mathbf{x}, \tag{3.49}$$

where $\mathsf{J}_{\mathbf{y}\mathbf{x}_0} = \left.\frac{\partial\mathbf{y}}{\partial\mathbf{x}}\right|_{\mathbf{x}_0}$ is the Jacobian of $\mathbf{g}(\mathbf{x})$ with respect to $\mathbf{x}$ at the point $\mathbf{x}_0$.

Note that the statistical properties are now associated with $\Delta\mathbf{x}$ instead of $\mathbf{x}$, it is

$$E(\Delta\mathbf{x}) = E(\mathbf{x} - \mathbf{x}_0) = E(\mathbf{x}) - E(\mathbf{x}_0) = \boldsymbol{\mu}_{\mathbf{x}} - \mathbf{x}_0 \tag{3.50}$$

$$\boldsymbol{\Sigma}_{\Delta\mathbf{x}} = \boldsymbol{\Sigma}_{\mathbf{x}-\mathbf{x}_0} = \boldsymbol{\Sigma}_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{x}_0} = \boldsymbol{\Sigma}_{\mathbf{x}}. \tag{3.51}$$

[3] It is possible to derive similar equations for higher-order central moments, independent of the distribution [100].

[4] It should be noted that a Taylor series will not necessarily converge, and even if it does, it will not necessarily converge towards the function $\mathbf{g}(\mathbf{x})$ it is meant to represent — although it normally does.

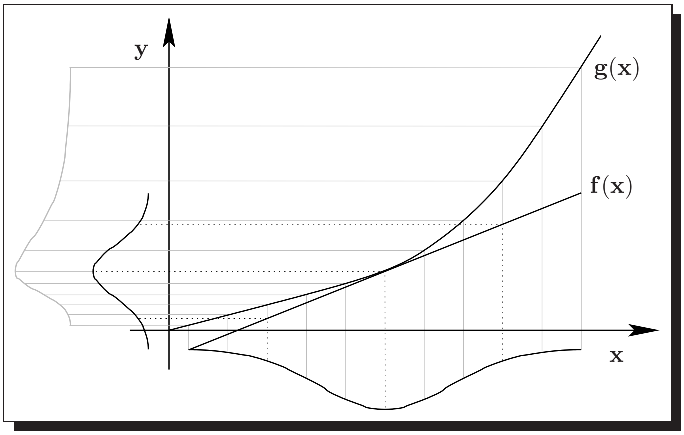


Figure 3.1: *Bias and deformation of the pdf due to nonlinearities. The true transformed pdf (grey) is deformed and also slightly offset with respect to the linear approximation (black).*

The linearisation will usually introduce an error. In order to keep this error small, it is customary to set $\mathbf{x}_0 = \boldsymbol{\mu}_\mathbf{x}$ and therefore $E(\Delta\mathbf{x}) = \mathbf{0}$. From there and Equation (3.49) we can calculate the mean and variance for the linearised function $\mathbf{y}' = \mathbf{f}(\mathbf{x})$ as

$$E(\mathbf{y}') = E(\mathbf{y}_0 + \mathsf{J}_{\mathbf{y}\mathbf{x}_0}\Delta\mathbf{x}) = E(\mathbf{y}_0) + \mathsf{J}_{\mathbf{y}\mathbf{x}_0}E(\Delta\mathbf{x})\mathbf{y}_0 = \boldsymbol{\mu}_{\mathbf{y}'} \tag{3.52}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}'} = \boldsymbol{\Sigma}_{\mathbf{y}_0+\mathsf{J}_{\mathbf{y}\mathbf{x}_0}\Delta\mathbf{x}} = \boldsymbol{\Sigma}_{\mathbf{y}_0} + \mathsf{J}_{\mathbf{y}\mathbf{x}_0}\boldsymbol{\Sigma}_{\Delta\mathbf{x}}\mathsf{J}^{\mathsf{T}}_{\mathbf{y}\mathbf{x}_0} = \mathsf{J}_{\mathbf{y}\mathbf{x}_0}\boldsymbol{\Sigma}_{\mathbf{x}}\mathsf{J}^{\mathsf{T}}_{\mathbf{y}\mathbf{x}_0} \tag{3.53}$$

with $\mathbf{y}_0 = \mathbf{g}(\mathbf{x}_0)$. Note that, in general, $\boldsymbol{\mu}_{\mathbf{y}'} \neq \boldsymbol{\mu}_\mathbf{y} = \mathbf{g}(\boldsymbol{\mu}_\mathbf{x})$, that is the linearisation introduces a bias. This is always the case if $\mathbf{g}(\mathbf{x})$ is not well approximated by its tangent within the region of dispersion of its random variables, as seen in Figure 3.1. This bias will always inflate the estimated covariance matrix, while the truncation of the Taylor series could either increase or decrease the result [29].

Just how good or bad this approximation is within a given region $\|\mathbf{x} - \mathbf{x}_0\|_2 \leq \Delta\mathbf{x}$ can be computed by calculating an upper bound on the remainder $\mathcal{O}_2(\|\Delta\mathbf{x}\|^2)$. Four different forms for $\mathbf{x}, \mathbf{g}(\mathbf{x}) \in \mathbb{R}$ are given in [65], and these are easily extended to $\mathbf{x} \in \mathbb{R}^n$, compare e. g. [22, p. 279]; in this thesis I will however only deal with first order approximations.

It is worth noting that, although the application of Equation (3.52) and in particular Equation (3.53) can result in extremely complex expressions, generating and using these expressions is a purely mechanical task which can in theory[5] be done by any computer algebra program. For our example of a line $\boldsymbol{\ell} \in \mathbb{R}^3$ though two points $\mathbf{p}_1, \mathbf{p}_2 \in \mathbb{R}^3$ this can be done as follows.

We know from Equation (2.25) that $\boldsymbol{\ell}$ can be calculated as $\boldsymbol{\ell} = \mathbf{p}_1 \times \mathbf{p}_2$, or alternatively as $\boldsymbol{\ell} = \mathsf{p}_{1_\times} \mathbf{p}_2 = \mathsf{p}_{2_\times} \mathbf{p}_1$ with

$$\mathsf{p}_{i\times} = \begin{pmatrix} 0 & -z_i & y_i \\ z_i & 0 & -x_i \\ -y_i & x_i & 0 \end{pmatrix}. \tag{3.54}$$

The Jacobian is therefore $\mathsf{J}_{\boldsymbol{\ell}\mathbf{p}_i} = (\mathsf{p}_{2_\times}, \mathsf{p}_{1_\times})$. If we know that the two points are normally distributed with covariance matrices $\boldsymbol{\Sigma}_{\mathbf{p}_1}, \boldsymbol{\Sigma}_{\mathbf{p}_2} \in \mathbb{R}^{3\times3}$ we can calculate the line's covariance matrix as

$$\boldsymbol{\Sigma}_{\boldsymbol{\ell}} = (\mathsf{p}_{2_\times}, \mathsf{p}_{1_\times}) \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{p}_1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{p}_2} \end{pmatrix} \begin{pmatrix} \mathsf{p}_{2_\times}^{\mathsf{T}} \\ \mathsf{p}_{1_\times}^{\mathsf{T}} \end{pmatrix} = \mathsf{p}_{2_\times} \boldsymbol{\Sigma}_{\mathbf{p}_1} \mathsf{p}_{2_\times}^{\mathsf{T}} + \mathsf{p}_{1_\times} \boldsymbol{\Sigma}_{\mathbf{p}_2} \mathsf{p}_{1_\times}^{\mathsf{T}} \tag{3.55}$$

using Equation (3.53). The mean is the line $\boldsymbol{\ell}$ itself. Note that the covariance matrices are of course all singular, since both points and lines only have 2 DOF each. This will be discussed in more detail in Section 4.

3.3.4 Implicit Functions

In practice, a result $\mathbf{y}_0 \in \mathbb{R}^m$ is often not calculated by an explicit function, but rather found as the set of parameters which extremises some function of the original data, i. e. the function $\mathbf{g}(\mathbf{x})$ is not explicitly known (and its Jacobian therefore cannot be calculated as usual). What is known instead is a cost-function $C(\mathbf{x}, \mathbf{y}) \in \mathbb{R}$ which we are trying to minimise. The necessary condition for an extremum is $\frac{\partial C(\mathbf{x},\mathbf{y})}{\partial \mathbf{y}}|_{\mathbf{x}_0} = 0$, and the *implicit function theorem* (compare [29, 43]) gives us the Jacobian $\mathsf{J}_{\mathbf{y}_0\mathbf{x}}$ for an *unknown* function $\mathbf{y} = \mathbf{g}(\mathbf{x})$ as

$$\mathsf{J}_{\mathbf{y}_0\mathbf{x}} = -\left(\frac{\partial^2 C}{\partial \mathbf{y}^2}\right)^{-1} \left(\frac{\partial^2 C}{\partial \mathbf{x} \partial \mathbf{y}}\right)^{\mathsf{T}} \Bigg|_{\mathbf{y}_0} \tag{3.56}$$

if the Hessian $\mathsf{H} = \left(\frac{\partial^2 C}{\partial \mathbf{y}^2}\right) \in \mathbb{R}^{m\times m}$ is invertible at the minimum $\mathbf{y}_0$. Using the idea of Lagrange multipliers, we can extend the above even further to constrained minimisation, see [29, 43] for more details.

[5] I still have to meet a computer algebra program which produces C-code that will not happily divide by zero or take the square-root of a negative argument.

Note that using Equation (3.53) a result's uncertainty can be calculated even if the result itself has *not* been obtained as the solution to some optimal algorithm minimising both the error and uncertainty of a particular calculation (a problem which often has no closed form solution), but rather by some faster but less accurate algorithm. However, using a faster, closed form solution might introduce a considerable bias and blow up $\mathbf{\Sigma}_{\mathbf{y}'}$, and the selection of a function $\mathbf{g}(\mathbf{x})$ that is both fast and accurate enough can become somewhat of an artform. Section 4 is practising this art for a number of common computer-vision constructs, mainly the ones introduced in Section 2.

3.3.5 Monte-Carlo Simulations

A second method for the propagation of statistical properties which is completely different from the analytical method given above is the Monte Carlo simulation. The basic idea is simple: given a function $\mathbf{y} = \mathbf{g}(\mathbf{x})$ and a vector $\mathbf{x}$ (assumed perfectly known), a large number of corrupted vectors $\mathbf{x}_i = \mathbf{x} + \mathbf{r}_i$ is created, where the $\mathbf{r}_i$ are distributed according to the measurement error's pdf $p_{\Delta\mathbf{x}}$ (assumed known). This large population is then used to estimate the pdf $p_{\mathbf{y}}$ of the samples $\mathbf{y}_i = \mathbf{g}(\mathbf{x}_i)$. In particular, the mean $\boldsymbol{\mu}_{\mathbf{y}}$ and covariance matrix $\mathbf{\Sigma}_{\mathbf{y}}$ can be approximated by the sample mean $\overline{\mathbf{y}}$ and sample covariance matrix $\mathsf{S}_{\mathbf{y}}$ using Equations (3.37) and (3.38). Note that $\mathbf{g}(\mathbf{x})$ does not need to be given explicitly; the $\mathbf{y}_i$ could just as well have been found by minimisation or any other technique.

The quality of Monte Carlo simulation relies only on the number of samples — between 10,000 and 1,000,000 samples are not unusual — and the quality of the pseudo random number generator, which should have a period at least 10 times greater than the number of samples required [29].

The two methods — analytical first order error propagation and Monte Carlo simulation — complement each other. First order error propagation is a fast and — for sufficiently small errors — reliable method which gives an analytical, closed form solution for $p_{\mathbf{y}}$. However, the equations used can become unwieldy, and it relies on the goodness of the linear approximation (which could in theory be assessed by calculating an upper bound for the remainder $\mathcal{O}_2(\|\Delta\mathbf{x}\|^2)$, compare Section 3.3.3).

Monte Carlo simulation, on the other hand, makes no assumptions on the function $\mathbf{g}(\mathbf{x})$ or resulting pdf $p_{\mathbf{y}}$ and is easy to program; these advantages are offset by an extremely long execution time (several minutes or even hours). Also, the result of Monte Carlo simulation is just a high number of points $\mathbf{y}_i$ instead of a closed-form probability density function.

In practice, Monte Carlo simulation is therefore often used to assess the goodness of the analytical solution found by first order error propagation; either visually, plotting both the points found by Monte Carlo simulation as well as the confidence regions found by first order error propagation, or analytically, where for example the χ^2 distribution described in the next section is used to compute the probability that the points $\mathbf{y}_i$ follow the distribution $p_{\mathbf{y}}$.

3.4 χ^2 Testing

The following gives a very short introduction into confidence testing, namely the χ^2 confidence test. More detailed information could be found in any textbook on statistics.

When using thresholds in a (computer-vision) algorithm, the underlying question is very often: "Am I satisfied that the observed value is the one that I'm looking for?". In statistics, questions like this[6] can be answered by confidence tests. The test of choice if the underlying distribution is Gaussian, as is frequently the case for measured values, is often the χ^2-test. Its application is simple and will be explained below for the example of two lines. The task is to decide whether a line measured in the image should be considered identical to an ideal line or not.

Assume that the lines have been parametrised by their angle φ with the x-axis and their distance d from the origin, $\boldsymbol{\ell} = (\varphi, d)^{\mathsf{T}}$, and that the measured line's covariance matrix is given by $\boldsymbol{\Sigma}_{\boldsymbol{\ell}} \in \mathbb{R}^{2\times 2}$. Simply speaking, the χ^2-test answers the question: "Am I $p\,\%$ satisfied that the difference between $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ is due to random fluctuations consistent with $\boldsymbol{\Sigma}_{\boldsymbol{\ell}}$?" by evaluating $(\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2)^{\mathsf{T}} \boldsymbol{\Sigma}_{\boldsymbol{\ell}}^{-1} (\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2) \le \chi^2_{p,2}$, where the subscript 2 denotes the number of degrees of freedom (φ and d in this case).

More generally we get (possibly after a suitable coordinate transform such that a covariance matrix becomes diagonal)

$$\sum_{i=1}^{N} \frac{d_i^2}{\sigma_{d_i}^2} \le \chi^2_{p,\nu} \tag{3.57}$$

where ν is the number of degrees of freedom (which need not be N, as we will see when fitting a line to edgels in Section 4.3.3) and p is the amount of required certainty in percent, called the significance level, and traditionally also often denoted by α or, confusingly, $1 - \alpha$ in some textbooks.

[6] Or, more accurately, the question "Am I not dissatisfied...?".

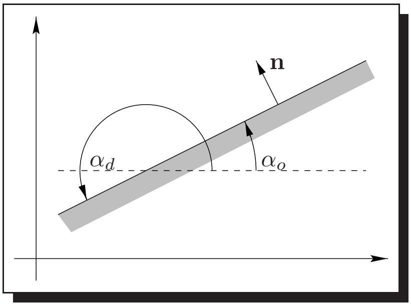


Figure 3.2: *Difference between the orientation $\alpha_o \in [-\pi/2, \pi/2)$ and the direction $\alpha_d \in [-\pi, \pi)$ for a greyscale-discontinuity.*
The discontinuity's normal vector $\mathbf{n} = (\sin(\alpha_d), -\cos(\alpha_d))^\mathsf{T}$ is pointing from dark to light.

3.5 Directional Statistics

Statistics commonly deals with distributions in $I\!R^n$, most commonly with distributions on a line ($n = 1$) or plane ($n = 2$). Many measurements, however, are concerned with quantities of a cyclic nature, in computer vision usually angles. Indeed, when Gauss developed the theory of errors he did so primarily to analyse certain directional measurements in astronomy. It is one of the ironies of statistics that the measurements under consideration were sufficiently accurate to allow him to develop the theory in relation to an infinite linear continuum rather than the actual topology, a sphere [95]. The subject of directional statistics received increased interest only after the 1953 landmark-paper by R. A. Fisher [47] and is thus a comparatively new branch of statistics.

The aim of this section is to introduce some very basic concepts of directional statistics needed later on; the reader is referred to [95, 156] for good introductory texts on the subjects.

3.5.1 Directions and Orientations

Directions in computer vision are usually associated with contours fitted to discontinuities in luminance, where one side will be lighter than the other. We can then differentiate between a contour's *orientation*, which can take values in the interval $\alpha_o \in [-\pi/2, \pi/2)$, and its *direction*, taking values in the interval $\alpha_d \in [-\pi, \pi)$. The orientation describes a geometric entity; turning this entity by 180° does not change it's representation, and the possible range of orientations is therefore limited to a semi-circle (semi-hypersphere) — the directions α_o and α_o+180° represent the same orientation α_o. Data like this is called *axial data*. The direction, on the other hand, is uniquely defined by a normal vector $\mathbf{n} = (\sin(\alpha_d), -\cos(\alpha_d))^\mathsf{T}$ pointing from the dark to the light side of a grey-level discontinuity, as shown in Figure 3.2, making each direction unique within one complete turn of a circle (hypersphere).

3.5.2 Mean and Variance

Directional data requires a notion of mean and variance different from usual statistics. Assume that two directions $\alpha_1 = 1^\circ$ and $\alpha_2 = 359^\circ$ are given. Naively applying Equation (3.19) to calculate the mean would give a value of $\overline{\alpha} = 180^\circ$, while intuition tells us that $\overline{\alpha} = 0^\circ$. If, however, directions are instead understood as points on the unit-circle[7] $\mathbf{x}_i = (\cos(\alpha_i), \sin(\alpha_i)^\mathsf{T})$ (or, alternatively, vectors of unit-length), we can calculate[8]:

$$\begin{pmatrix} C \\ S \end{pmatrix} = \sum_{i=1}^{N} \begin{pmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{pmatrix} \tag{3.58}$$

$$\overline{\alpha} = \begin{cases} \arctan(S/C) - \pi & C < 0, S < 0 \\ -\pi/2 & C = 0, S < 0 \\ \arctan(S/C) & C > 0 \\ \pi/2 & C = 0, S > 0 \\ \arctan(S/C) + \pi & C < 0, S > 0 \end{cases} . \tag{3.59}$$

It suggests itself to also calculate the length $R = \sqrt{C^2 + S^2}$ of the resulting vector, the *mean resultant length*. It is easy to see that R will be close to its maximum value $R = N$ if the α_i are very concentrated, while it will be close to its minimum value $R = 0$ if the α_i are very dispersed. Thus $N - R$ is a sensible measure of the dispersion of the whole sample about its estimated centre — analogous to the variance on a straight line — and indeed

$$S_0 = \frac{N - R}{N - 1} \tag{3.60}$$

is called the (sample) *spherical variance*. Note that $0 \leq S_0 \leq 1$, while of course for the variance on a line $0 \leq \sigma^2 \leq \infty$. A value which is more similar in magnitude to the usual variance on the line is given by

$$s_0^2 = -2\ln(1 - S_0). \tag{3.61}$$

[7] Or hypersphere in the general case $\mathbf{x}_i \in \mathbb{R}^n$.

[8] Many programming-languages provide a function `atan2(x,y)` for this purpose.

For axial data (i. e. $-\frac{\pi}{2} \leq \alpha_i < \frac{\pi}{2}$) the corresponding equations are [95]:

$$\begin{pmatrix} C' \\ S' \end{pmatrix} = \sum_{i=1}^{N} \begin{pmatrix} \cos(2\alpha_i) \\ \sin(2\alpha_i) \end{pmatrix} \tag{3.62}$$

$$\overline{\alpha} = \frac{1}{2}\overline{\alpha}' \tag{3.63}$$

$$S_0 = 1 - (1 - S_0')^{1/4}. \tag{3.64}$$

There is no known distribution on the circle which has all the properties of the normal distribution. It is most closely approximated by the von Mises distribution or the wrapped normal distribution, see e. g. [95, 156] for details.However, as was the case for Gauss' original use of his distribution on strictly speaking cyclic data, I too found that for the applications discussed in this thesis the Gaussian distribution is sufficient.

Chapter 4

Combining Projective Geometry and Error Propagation

. . . fügte ich rittlings zusammen, was zusammengehörte.
. . . astraddle I joined together what belonged together

Felix Salten, Josefine Mutzenbacher, 1869–1945

4.1 Introduction

This chapter, which lies at the heart of this thesis, combines the projective geometry constructs described in Chapter 2 with the statistical principles of Chapter 3, and in particular error propagation.

Starting from first principles, with an error model for single edgels in Section 4.2, I revisit the line-fitting problem in Section 4.3. There the covariance of a line is computed, based on the covariances of the single edgels; for the case of independently, identically, and isotropically distributed (iiid) edgels (which is the usual assumption when fitting a line to edgels) I also present, in Section 4.3.2, an excellent but previously unpublished approximation to that covariance based mainly on line-length; and in Section 4.3.3 I introduce a new stopping-criterion for incremental fits which is based on a χ^2-test. Section 4.4 compares several algorithms for vanishing-point calculation, clearly demonstrating that algorithms based on Euclidean distance, which are unfortunately still all too common in the literature, are inadequate for intersections far away from the image. In Section 4.5 I introduce a new algorithm for the calculation of the cross-ratio of 4 lines, which performs nearly as well as the best possible algorithms, but without knowledge about the lines' intersection — which makes the algorithm about an order of magnitude faster than other algorithms with comparable performance. Extensive Monte Carlo simulations are used throughout this section to evaluate and compare the relative performance of several competing algorithms both with regard to accuracy as well as speed. Section 4.6 finally demonstrates how to compare stochastic projective entities, and how to account for additional uncertainty in the model, e. g. due to an imperfect world.

The use of error propagation, while being a staple of photogrammetrists, geodesists, and many other scientists, has always been somewhat neglected in computer vision. Most notable is probably the influence of Kanatani [71–75, 77], who can be said to have pioneered this particular field. The main difference which sets this work apart from Kanatani's is its focus on applicability — whereas Kanatani concentrates on the *correct* solution, I mostly concentrate on the most *adequate* solution, weighing computational cost and implementational complexity against the gain in accuracy. Also related to the work described here is the work by Brillault-O'Mahony [20, 21], who used statistical considerations for vanishing-point detection, and grouping and recognition of high-level 3D-structures (see Section 6). More recent work includes [11, 52, 66, 115–117, 130, 141], of which the work by Pennec [116, 117] is closest to the work presented here. A very recent addition is Förstner's [49] contribution to the "Handbook of Computational Geometry", which collects a number of simple to use tools for uncertain geometric reasoning, and in particular

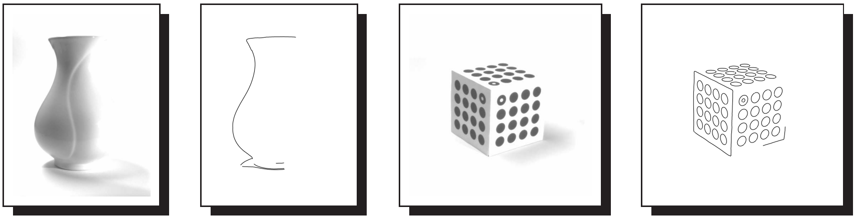


Figure 4.1: *Surface discontinuities do not always lead to visible edges.*

gives a number of explicitly calculated Jacobians which retain the elegance of projective algebra. It can serve as a nice and concise introduction into my work; however, as was the case with Kanatani's work, Förstner's focus is on elegant rather than computationally efficient solutions, which are at the heart of this thesis; his work differs also in the use of a less rigorous approach testing uncertain geometric relations — he directly tests observed entities against each other, rather than against the estimated true value, as I will recommend in e. g. Section 4.6.2. Finally I should point out that I have already published some of the results presented here in [6].

4.2 Edgels

The work described in this thesis is completely edge-based. All features described later on can ultimately be reduced to edgels (*edge-el*ements). We can distinguish two kinds of edges in 3D. The first one corresponds to a change in luminance, hue, saturation, or all three within one surface. These changes, which we will call surface markings, are always detectable using an appropriate setup. The second kind of edge corresponds to a surface discontinuity. This is not necessarily associated with any apparent change in visual properties, and the detectability of these kinds of edges within a certain image depends on the object's orientation towards the camera, lighting, and other external conditions. Figure 4.1 shows examples for surface markings as well as visible and invisible surface-discontinuities.

These edgels are located, often with subpixel accuracy, in the image using an edge detector [24]. The next two sections describe possible sources for error in the location of the edgels and how to model this error, as well as a possible parameterisation of edgel location and its probability distribution.

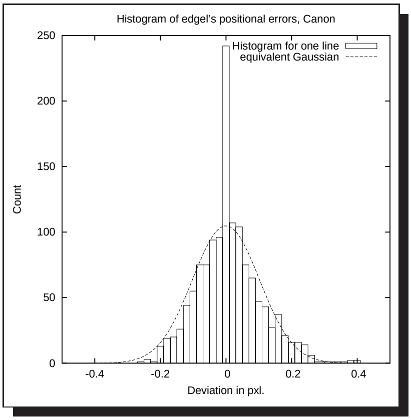
4.2.1 Error Sources

So what are the particular types of errors encountered in the imaging (and reconstruction) process, and which of those will I address in this thesis? Aberrations of the lens, which have been well documented in many publications [16, 18, 86, 144], include chromatic (axial and lateral) as well as monochromatic aberrations (also called Seidel aberrations after an 1857 paper by Ludwig von Seidel, i. e. spherical aberration, coma, astigmatism, field curvature and curvilinear — barrel and pincushion — distortions). In practical applications we also see vignetting, flares and diffraction — and of course simple defocus. Additional errors are being introduced by the CCD chip, foremost of course the discretisation itself, but we are also dealing with (thermal) pixel noise and with differences in the sensitivity of neighbouring (or further away) sensors which create a bias. In most of todays 1-chip colour cameras we get additional errors due to the interpolation of colour information from the mosaicing (usually Bayer) filter and possibly also from lossy compression (usually jpeg), both of which are particularly pronounced near edges. And finally, as a last source for error, we also have the effect of the edge detector itself.

The error sources given above can be grouped into two different categories, reversible and non-reversible effects. Curvilinear distortions and bias in the individual pixel values are easily removed by a simple calibration of the camera; as such they are really systematic errors and will be ignored in the following; the same is true for some of the errors introduced by the edge detector [106].

Most of the other lens effects, however, although quite systematic in their formation, are not easily reversible. In their sum total they will serve to make the image less sharp, and as such act as a low-pass filter blurring the image; approximating their influence by a convolution with a Gaussian is not uncommon [86]. If we then proceed to apply an edge filter like the well known Canny filter to the image this additional, non-uniform blurring will have a negative effect on the positional accuracy of the edgels found [24], at least in the neighbourhood of texture or additional edges. Lossy compression and demosaicing, on the other hand, tend to introduce random artefacts near edges, as will the pixel noise of the CCD chip, and these will directly influence the positional accuracy, since they violate the continuity assumptions which are the foundation of subpixel approximation.

Accurately modelling all these different error sources and their effect on the positional accuracy of edgels is well beyond the scope of this thesis. However, Figure 4.2 shows the histogram of the positional errors for typical edgels along a typical (but perfectly straight) line for images taken with two different cameras, as well as the equivalent Gaussian distribution (i. e. one with the same standard deviation). And



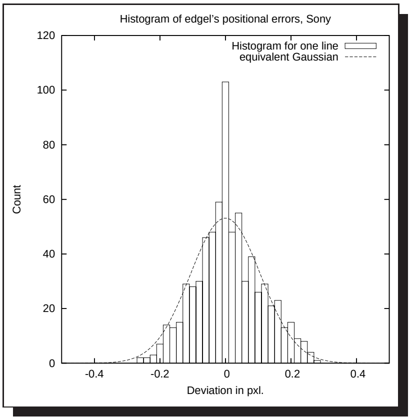


Figure 4.2: *Typical distributions of positional error for a perfect line. Plotted are the deviations of edgels from the true line for a Canon* 6 Mpxl *camera (left) and a Sony* 800 kpxl *camera (right). Overlayed are the equivalent Gaussian distributions.*

although the measured distributions are clearly not Gaussian (note in particular the high number of nearly accurate edgels, about 5 % to 10 % for this particular testcase), they are none the less reasonably well approximated by a Gaussian — and this is in fact what I will do for the remainder of this thesis.

It might be worth pointing out that both histograms in Figure 4.2 have approximately the same standard deviation (around $\sigma \approx 0.1$ pxl). This demonstrates nicely that the locational error for a perfect line is mostly a function of the sensor type used (a Bayer-type mosaicing filter in both cases). However, in reality most lines aren't quite perfect[1], and might suffer additional distortion depending on the lens used. So in addition to the one hand-selected (perfect) line above I also calculated the histograms over all lines, and those are given in Figure 4.3. And here the two variances are clearly different for the two cameras, with $\sigma \approx 0.27$ pxl for the Canon but only $\sigma \approx 0.15$ pxl for the Sony — here the higher resolution camera (the Canon) registers the higher standard deviation, since the same positional error in 3D will result in a bigger error (measured in pixel) in the image; the better lens of the Canon, suffering from fewer of the above-mentioned defects than the lens of the Sony, by comparison does not have as much of an effect due

[1]For this test I used lines printed on paper — and since the paper wasn't perfectly flat when the images were taken, not all lines are perfectly straight.

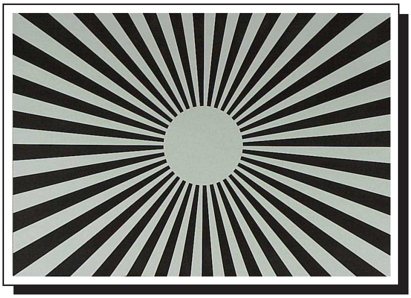
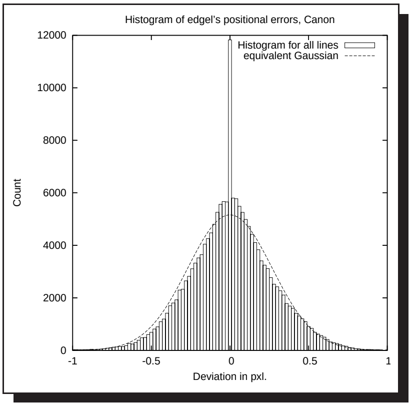
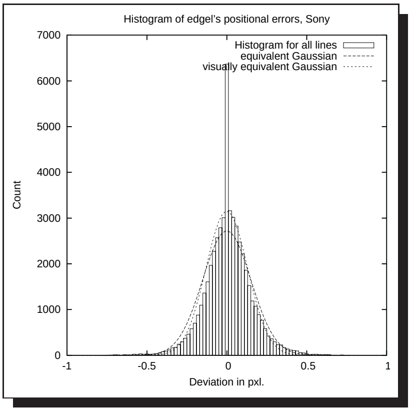


Figure 4.3: *Typical distributions of positional errors (below) for all edges within an image (to the left). Plotted are the deviations of edgels from the true edge for a Canon* 6 Mpxl *camera (left) and a Sony* 800 kpxl *camera (right). Overlayed are the equivalent Gaussian distributions.*





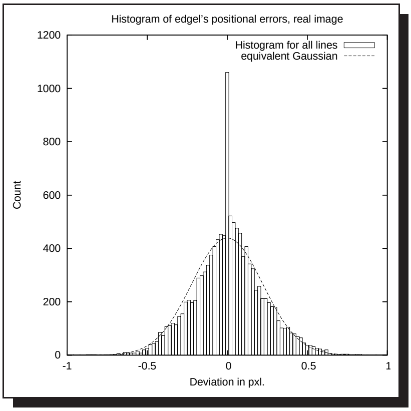


Figure 4.4: *Typical distribution of positional errors for all edges within a real-world image (Figure 5.5, $\sigma \approx 0.22\,\text{pxl}$, and equivalent Gaussian distribution.*

to the particular test-image chosen.

Figure 4.2.1 finally shows the distribution of positional errors for a real-world image (a street scene also used throughout most of Chapter 5, e. g. in Figure 5.5); here too we see the typical, Gauss-like distribution with its overpronounced peak for very small errors.

In this section we have seen that a Gaussian distribution is not altogether an unrealistic approximation for the particular distribution of location errors when fitting edgels. In the following I will describe how to represent the edgel coordinates and their distribution.

4.2.2 Geometric Representation

Edgels can be represented by their Euclidean coordinates within the image plane $(x, y)^{\mathsf{T}}$. Other possible representations include (pseudo) homogeneous coordinates,

$$\mathbf{x} = (x, y, 1)^{\mathsf{T}}, \tag{4.1}$$

which is the representations used throughout this chapter unless stated otherwise. Kanatani [69] and others suggested a parametrisation $(x, y, f)^{\mathsf{T}}$ with $x^2+y^2+f^2 = 1$, where f is of the same order of magnitude as x and y, often the focal length the image was taken with (if known, compare Section 2.9), the length of the image diagonal, or some other image dimension.

All coordinates computed in the image, independent of the parametrisation used, will contain errors due to the measurement process. These can be characterised by the edgels' covariance matrix, which for the pseudo homogeneous representation in Equation (4.1) would be structured as follows:

$$\boldsymbol{\Sigma}_{\mathbf{x}} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & 0 \\ \sigma_{xy} & \sigma_y^2 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.2}$$

If the error can be modelled by a Gaussian distribution, as is the case for many practical applications [100], this covariance matrix is sufficient to completely characterise the edgel's distribution. Using Equation (3.53) it is possible to calculate the covariance matrix for all other parametrisations from the covariance matrix in Equation (4.2). Note that $\boldsymbol{\Sigma}_{\mathbf{x}}$ is of course singular, since an edgel has only two degrees of freedom, independent of the parametrisation used. It is therefore not possible to directly compute its inverse. Instead the Moore-Penrose generalised or pseudo inverse should be used, or the problem should be reduced to the equivalent problem in fewer dimensions, again using Equations (3.52) and (3.53). However, in the context of projective geometry the latter is usually not desirable. Section 4.3.2.2 shows that (4.2) can be approximated by a diagonal matrix for many practical applications — the main thrust of the argument being that the covariance along the edge is usually of no consequence.

4.3 Lines

Detecting lines or, more generally, linear structures within an image is a classical problem in computer vision. Lines can convey valuable information about the 3D-structure depicted, particularly in our modern world, where parallel lines and orthogonal corners abound. Computer vision was concerned from the beginning with the construction of line drawings from natural scenes (and their subsequent interpretation), an approach still used today [110, 112–114, 148]. But lines convey valuable information about the structure of a scene even if no complete and realizable graph is available, as the applications described in Sections 5 and 6 will show.

Lines in computer vision are, in general, defined in terms of edgels, and it is therefore not too surprising that here too we distinguish lines due to surface markings on the one hand and due to surface discontinuities on the other. Common to both types of lines is the fact that the local neighbourhood of the line will be planar. Although other contours might appear as a line under special circumstances, this is

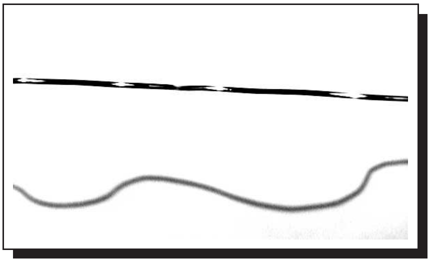


Figure 4.5: *Nonlinear structures might appear as lines under certain conditions. In the image to the left, the top structure looks line-like except for the occasional highlights, although the shadow below the structure reveals that it is clearly not a line.*

generally limited to one particular viewpoint and therefore of no interest to us (see Figure 4.5). However, this particular case makes apparent one of the differences between a line in the image and a line in 3D. We will in the following ignore the pathological case and assume that any 2D-line corresponds to a linear structure in 3D.

Finding the line $\boldsymbol{\ell}$ that best approximates a set of points (or edgels) $\mathbf{x}_i$ is a classical problem [43]. We can differentiate between three cases:

1. Fitting a line to a fixed number of points.
2. Incrementally fitting a line to an (unknown) number of points along a contour.
3. Fitting a line to an (unknown) subset of points.

Only the first two cases are discussed here, as I am concentrating on the effect of errors in valid measurements rather than the effect of outliers; the third case was in computer vision traditionally solved by a Hough transform [61], which can be used to identify possible lines and the corresponding points, followed by the approach used for Case 1. The Hough transform is however more and more replaced by an algorithm called RANSAC [46], a method of robust statistics. Although not discussed within this thesis, I would like to point out that RANSAC in particular needs a method to distinguish between inliers and outliers which can model both the noise of individual edgels as well as a line's uncertainty, and should therefore benefit especially from the methods described in Section 4.3.3.

The remainder of this section is structured as follows: Section 4.3.1 lists a number of possible line-parameterisations. The basic principles when either directly (1) or incrementally (2) fitting a line to points are the same and are therefore discussed for the first case only (fixed number of points, Section 4.3.2). An iterative solution first described by Kanatani [73, 77] is given in Section 4.3.2.1. In Section 4.3.2.2 I next discuss in which cases this can be simplified so that a closed form solution is possible and give an excellent but previously unpublished approximation to that covariance based mainly on line-length. In Section 4.3.3 finally I introduce a new stopping-criterion for incremental fits which is based on a χ^2-test; a summary is given in Section 4.3.4.

4.3.1 Parameterisations

Representations of a line in the image plane require a minimum of two parameters (a line on the plane has two degrees of freedom). Well known parametrisations include slope-intercept ($y = mx + b$), intercept-intercept ($x/a + y/b = 1$), angle-intercept, and angle and distance to the origin. It is well known that both slope as well as intercept become infinite for vertical lines (and in the case of intercept-intercept parametrisation also horizontal lines).

In addition there are a number of redundant representations for lines, using more than 2 parameters. An example already discussed in Section 2.4 is the homogeneous 3-vector $(a, b, c)^\mathsf{T}$. With $a^2 + b^2 = 1$ this becomes the normal form. Kanatani [69] and others suggested a parametrisation $(a, b, c/f)^\mathsf{T}$ with $a^2 + b^2 + c^2/f^2 = 1$, where f is a constant of the same order of magnitude as c, often the focal length the image was taken with (compare Section 2.9), the diagonal length of the image, or some other image dimension. Note that a covariance matrix for any of these redundant representations will of course be singular.

Parametrisations particularly appropriate to line segments as measured in the image have also been developed, e. g. using a line segment's endpoints (x_1, y_1, x_2, y_2) as cited in [140] or a line segment's centre (mean edgel position) and angle with the x-axis, $(\alpha, \bar{x}, \bar{y})^\mathsf{T}$ which I first presented in [6].

It is of course possible to convert any representation into any of the two-parameter or redundant representations, but, in general, not into the line segment representations. However, these conversions, while theoretically possible, can become numerically problematic; one example is the conversion from and to the slope-intercept representation for near vertical lines. We will later see that some parametrisations are better suited to a certain task than others, and that in particular the angle-centre representation is well suited for the line segments commonly fitted in computer vision. The next section re-visits the line-fit problem, starting from first principles.

4.3.2 Fixed Number of Points

The problem of fitting a line to a fixed number of points can be stated as follows: given n points $\mathbf{x}_i$, $i = 1 \dots n$, with distributions $p_{\mathbf{x}_i}(\mathbf{x}_i)$, find the line $\boldsymbol{\ell}$ with distribution $p_{\boldsymbol{\ell}}(\boldsymbol{\ell})$ that maximises the conditional probability

$$P(\boldsymbol{\ell}|\mathbf{x}_1 \dots \mathbf{x}_n) = \frac{P(\mathbf{x}_1 \dots \mathbf{x}_n|\boldsymbol{\ell})P(\boldsymbol{\ell})}{P(\mathbf{x}_1 \dots \mathbf{x}_n)} \longrightarrow \max_{\boldsymbol{\ell}} \tag{4.3}$$

where $P(\boldsymbol{\ell}|\mathbf{x}_1 \dots \mathbf{x}_n)$ is the probability of observing the line $\boldsymbol{\ell}$ given the points $\mathbf{x}_1 \dots \mathbf{x}_n$, $P(\mathbf{x}_1 \dots \mathbf{x}_n|\boldsymbol{\ell})$ is the probability to observe the set of points $\mathbf{x}_1 \dots \mathbf{x}_n$ given $\boldsymbol{\ell}$, $P(\boldsymbol{\ell})$ is the line's a-priori probability, and $P(\mathbf{x}_1 \dots \mathbf{x}_n)$ is the points' a-priori probability.

It is obvious that for any fixed set of points $\mathbf{x}_i$, $i = 1 \dots n$ the term $P(\mathbf{x}_1 \dots \mathbf{x}_n)$ can have no influence on which line $\boldsymbol{\ell}$ maximises Equation (4.3). Also, in nearly all applications no knowledge exists about the individual lines' a-priori probabilities $P(\boldsymbol{\ell})$, which are therefore usually assumed constant[2]. This reduces Equation (4.3) to

$$P(\boldsymbol{\ell}|\mathbf{x}_1 \dots \mathbf{x}_n) \propto P(\mathbf{x}_1 \dots \mathbf{x}_n|\boldsymbol{\ell}) \longrightarrow \max_{\boldsymbol{\ell}} . \tag{4.4}$$

If all the points $\mathbf{x}_i$, $i = 1 \dots n$ are independently distributed observations of a perfect line[3], we can multiply the individual probabilities

$$P(\mathbf{x}_1 \dots \mathbf{x}_n|\boldsymbol{\ell}) = \prod_{i=1}^{n} P(\mathbf{x}_i|\boldsymbol{\ell}) \longrightarrow \max_{\boldsymbol{\ell}} . \tag{4.5}$$

The individual point's probability depends on its distance from the line and its particular covariance matrix. If the point has the Euclidean coordinates $(x, y)^{\mathsf{T}}$ and the line is given by its normal form we can write

$$\mathbf{x}_i = (x_i, y_i, 1)^{\mathsf{T}} \tag{4.6}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}_i} = \begin{pmatrix} \sigma_{x_i}^2 & \sigma_{x_i y_i} & 0 \\ \sigma_{x_i y_i} & \sigma_{y_i}^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.7}$$

$$\boldsymbol{\ell} = (a, b, c)^{\mathsf{T}} = (\sin(\alpha), -\cos(\alpha), c)^{\mathsf{T}} . \tag{4.8}$$

[2]If a particular application does provide a-priori knowledge about, for example, a line's angle, this should of course be used. However, I'll assume that this is not the case here and, indeed, in virtually all computer vision applications that deal with projective geometry.

[3]This is approximately the case.

Using Equation (3.53) the individual point's probability is then given by

$$P(\mathbf{x}_i|\boldsymbol{\ell}) = \frac{1}{\sqrt{2\pi\boldsymbol{\ell}^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{x}_i}\boldsymbol{\ell}}} \exp\left(-\frac{1}{2}\frac{\boldsymbol{\ell}^\mathsf{T}\mathbf{x}_i\mathbf{x}_i^\mathsf{T}\boldsymbol{\ell}}{\boldsymbol{\ell}^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{x}_i}\boldsymbol{\ell}}\right). \tag{4.9}$$

Maximising Equation (4.5) is therefore equal to minimising the sum of weighted Euclidean distances

$$\min_{\boldsymbol{\ell}} \frac{1}{n}\sum_{i=1}^{n} \frac{\boldsymbol{\ell}^\mathsf{T}\mathbf{x}_i\mathbf{x}_i^\mathsf{T}\boldsymbol{\ell}}{\boldsymbol{\ell}^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{x}_i}\boldsymbol{\ell}} \tag{4.10}$$

under the condition $a^2 + b^2 = 1$. There is, in general, no closed form solution to Equation (4.10).

A slightly different approach was presented by Kanatani [73]. Instead of minimising Euclidean distance his approach minimises algebraic distance with vectors $\boldsymbol{\ell} = (a, b, c/f)^\mathsf{T}$ and $\mathbf{x}_i = (x, y, f)^\mathsf{T}$ under the condition $\|\boldsymbol{\ell}\|_2 = \|\mathbf{x}_i\|_2 = 1$, where f is again a constant in the order of magnitude of the focal length. This approach was later extended to $f = 1$ by Kanazawa and Kanatani [77]. They employ an elaborate scheme to avoid the bias which would be introduced by a naive iteration. A short description of their algorithm can be found in the following section.

4.3.2.1 Iterative Solution

As mentioned before, no closed form solution exists for Equation (4.10). It might seem reasonable though to rewrite Equation (4.10) as

$$\min_{\boldsymbol{\ell}} \boldsymbol{\ell}^\mathsf{T}\mathbf{M}\boldsymbol{\ell} \tag{4.11}$$

$$\text{with} \qquad \mathbf{M} = \frac{1}{n}\sum_{i=1}^{n} w_i\mathbf{x}_i\mathbf{x}_i^\mathsf{T} \tag{4.12}$$

$$\text{and} \qquad w_i = \frac{1}{\boldsymbol{\ell}'^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{x}_i}\boldsymbol{\ell}'} \tag{4.13}$$

and iteratively solve for $\boldsymbol{\ell}$, where $\boldsymbol{\ell}'$ is a previously found solution. It can, however, be shown that this approach will lead to a biased solution for $\boldsymbol{\ell}$, and Kanazawa and Kanatani [77] suggested the following approach instead:

1. Let $c = 0$ and $w_i = 1$, $i = 1 \dots n$.
2. Compute the matrices $\mathbf{M}$ (compare Equation (4.12)) and
$$\mathbf{N} = \frac{1}{n}\sum_{i=1}^{n} w_i\boldsymbol{\Sigma}_{\mathbf{x}_i}.$$

3. Compute the smallest eigenvalue λ and corresponding eigenvector $\boldsymbol{\ell}$ of
$$\hat{\mathbf{M}} = \mathbf{M} - c\mathbf{N}.$$

4. If the iteration reached a stationary state ($\Delta\lambda = 0$, [77] uses $\lambda \approx 0$) abort, else update c and w_i as follows and return to 2
$$\begin{aligned} c &\leftarrow c + \frac{\lambda}{\boldsymbol{\ell}^\mathsf{T}\mathbf{N}\boldsymbol{\ell}} \\ w_i &\leftarrow \frac{1}{\boldsymbol{\ell}^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{x}_i}\boldsymbol{\ell}}. \end{aligned}$$

An estimate for the line's covariance matrix is given by
$$\boldsymbol{\Sigma}_{\boldsymbol{\ell}} = \frac{c}{n-2}\left(\hat{\mathbf{M}}\right)_2^- \tag{4.14}$$
where $(\cdot)_2^-$ denotes the generalised inverse computed by ignoring the smallest eigenvalue, which might not be exactly zero due to numerical reasons or a premature termination of the above steps [77].

It should be noted that using Equation (4.14) the line's covariance can be calculated even if only the cofactor matrices are given for the individual points. This is based on a χ^2 distribution with $n-2$ DOF. It is in this case, however, impossible to make any statement about the quality of the fit.

The algebraic distance used above can be interpreted geometrically when using the ray-space or Gaussian sphere model (Section 2.9): imagine a unit sphere touching the image in the image centre. Each point in the image corresponds to the point on the sphere where a line from the sphere's origin to the image point intersects the sphere; each line in the image corresponds to a great circle where a plane through the origin and the original line in the image intersects the sphere. Minimising the algebraic distance finds the plane through the origin with minimum mean squared scaled orthographic distance from these points on the sphere. The intersection between this plane and the image plane is the line with minimum algebraic distance to the original points.

4.3.2.2 Direct Least Squares Solution

The iterative solution described above is comparatively slow. It would therefore be useful if some faster algorithm could be devised, preferably some closed form solution, or an approximation to such a solution. In order to use a closed form solution to Equation (4.10) we need to be able to approximate the denominator by a term independent of $\boldsymbol{\ell}$. For this we need to know what the individual edgels' covariance matrices look like. Most modern edge-finders will do a sub-pixel approximation orthogonal to the (perceived) edge direction, which might result in different variances orthogonal to ($\sigma_\perp^2$) or parallel with ($\sigma_\parallel^2$) the edge, but to a good approximation independent of the individual edgel itself[4]. The resulting covariance matrix is

$$\boldsymbol{\Sigma}_{\mathbf{x}_i} = \mathbf{R}^{-\mathsf{T}} \begin{pmatrix} \sigma_\parallel^2 & 0 & 0 \\ 0 & \sigma_\perp^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{R}^{-1}$$

$$\text{with} \qquad \mathbf{R} = \begin{pmatrix} \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where α_i is the estimated angle between the edge through $\mathbf{x}_i$ and the x-axis[5]. The denominator in Equation (4.10) then becomes

$$\boldsymbol{\ell}^\mathsf{T} \boldsymbol{\Sigma}_{\mathbf{x}_i} \boldsymbol{\ell} = \sigma_i^2 = \sigma_\perp^2 \cos^2(\alpha - \alpha_i) + \sigma_\parallel^2 \sin^2(\alpha - \alpha_i) \tag{4.15}$$

where α is the angle between the fitted line and the x-axis. Although, in general, $\alpha \neq \alpha_i$, the difference will nonetheless be small for any reasonable edge-finder. Figure 4.6 shows some typical histograms over the deviation from the true angle for 6 different angles $0 \le \alpha \le \frac{5}{12}\pi$.

Since the difference between the perceived angle α_i and true angle α is generally well below $2°$, we can, with $\sin^2(\alpha - \alpha_i) \le 0.00122$ and $\cos^2(\alpha - \alpha_i) \ge 0.9988$, approximate Equation (4.15) with

$$\boldsymbol{\ell}^\mathsf{T} \boldsymbol{\Sigma}_{\mathbf{x}_i} \boldsymbol{\ell} \approx \sigma_\perp^2 = \sigma^2 \tag{4.16}$$

which is independent of $\mathbf{x}_i$ and interestingly also $\sigma_\parallel^2$. This conforms with the intuition that in fitting a line to points the individual point's position along the direction of the line (and therefore also its covariance in that direction) is of small or no consequence for the fitting process. The particular instance of the Canny edge detector [24] used throughout this thesis results in a standard deviation of $0.1\,\text{pxl} \le \sigma \le 0.3\,\text{pxl}$, depending on the sensor type and image quality. This finally

[4] $\sigma_\perp^2$ and $\sigma_\parallel^2$ might depend on the perceived edge direction, but this effect is generally sufficiently small — and constant for any single line — to be safely ignored in this application.

[5] It is of course $\mathbf{R}^{-\mathsf{T}} = \mathbf{R}$.

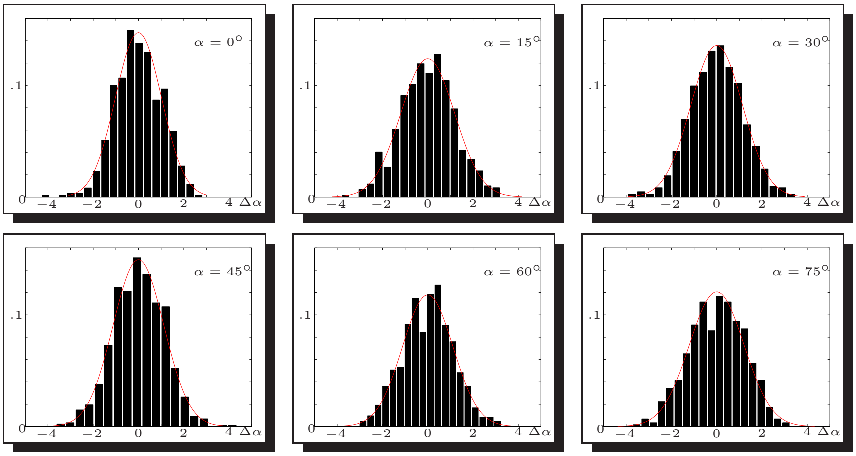


Figure 4.6: *Estimated angle versus true angle for different angles. Plotted are typical histograms over the deviation from the mean for lines at approximately* 0°, 15°, 30°, 45°, 60°, *and* 75°.

leads to the equation commonly minimised in orthogonal regression

$$\min_{\boldsymbol{\ell}} \frac{1}{n\sigma^2} \boldsymbol{\ell}^{\mathsf{T}} \left(\sum_i \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} \right) \boldsymbol{\ell} \tag{4.17}$$

under the constraint $a^2 + b^2 = 1$. This is well known [40, 43] to be minimised by the line $\boldsymbol{\ell}$ that passes through the point

$$\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \frac{1}{n} \sum_{i=1}^{n} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{4.18}$$

(we will call this point the line's centre point or simply centre) and whose normal-vector is the eigenvector to the matrix's

$$\mathbf{M}_{\mathbf{xx}} = \frac{1}{n\sigma^2} \sum_{i=1}^{n} \begin{pmatrix} (x_i - \bar{x})^2 & x_i y_i - \bar{x}\bar{y} \\ x_i y_i - \bar{x}\bar{y} & (y_i - \bar{y})^2 \end{pmatrix} \tag{4.19}$$

smaller eigenvalue $\lambda_{\min}$, where the mean squared weighted orthographic distance in Equation (4.17) corresponds to $\lambda_{\min}$.

Once the line $\boldsymbol{\ell}$ has been found, it is easy to calculate its covariance matrix using Equation (3.53) for any of the representations mentioned in Section 4.3 (although the actual equations involved can get somewhat lengthy, they are easily enough created, as discussed in Section 3.3.3). Using the α, $\bar{x}$, $\bar{y}$ parametrisation has the added advantage that the covariance matrix becomes to a very good approximation block-diagonal, i. e.

$$\boldsymbol{\Sigma}_{\boldsymbol{\ell}} = \begin{pmatrix} \sigma_\alpha^2 & 0 & 0 \\ 0 & \sigma_{\bar{x}}^2 & \sigma_{\bar{x}\bar{y}} \\ 0 & \sigma_{\bar{x}\bar{y}} & \sigma_{\bar{y}}^2 \end{pmatrix} \tag{4.20}$$

and as small deviations in the centre-point's position along the line are normally of little importance this can be further approximated by a diagonal matrix with $\sigma_{\bar{x}}^2 = \sigma_{\bar{y}}^2 = \frac{1}{n}\sigma^2$. This is in agreement with Equation (4.16) for the special case that $\boldsymbol{\Sigma}_{\mathbf{x}}$ describes a circular covariance region.

However, in many cases and for many applications it might not be necessary to calculate explicitly the line's covariance matrix using Equation (3.53). Since the edgels along a line are usually quite evenly distributed, where the distance between individual edgles depends mainly on the angle between the line and the x-axis, it is perfectly reasonable to give a rule of thumb for the covariance matrix based only on the length of the line l, the number of edgels n and the angle α. The relationship in (4.21)– (4.22) have been found experimentally, however, they are a nearly perfect representation of the true values, compare Figure 4.7.

$$\sigma_\alpha^2 \approx 12\frac{\sigma^2}{n'^3} \tag{4.21}$$

$$\text{with} \quad n'^3 = l^3 \left(1 + 12^{1/3} - 12^{1/3} \max(\|\cos(\alpha)\|, \|\cos(\pi/2-\alpha)\|)\right) \tag{4.22}$$

$$\sigma_{\bar{x},\bar{y}}^2 \approx \frac{\sigma^2}{n}. \tag{4.23}$$

Here n' is the equivalent number of edgels, an empirically found, purely arithmetic figure, it is usually $n \geq n' \geq l$ (unless the edgel-chain contained holes). Equation (4.23) can immediately be derived from Equation (4.18).

Figure 4.7, which plots the linearised variance $\left(12\sigma^2/\sigma_\alpha^2\right)^{1/3}$ over n', shows how well Equation (4.21) approximates the actual values for σ_α^2 measured for 5411 fitted lines from 10 different images of various real-life street scenes. Equations (4.21)–(4.23), which have never been published before, are similar in spirit to the dependence of σ_α^2 on line-length l derived by Brillault [20] — however, her theoretical approach, which considered only the endpoint-positions as a source for errors, gives a completely different qualitative behaviour not consistent with reality. They can

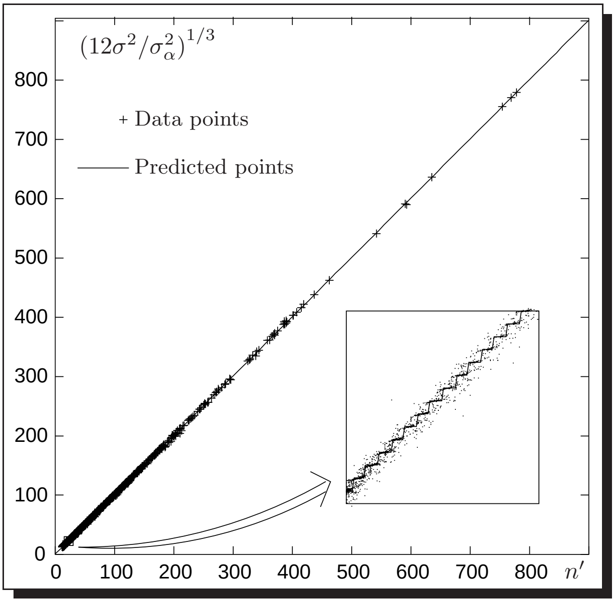


Figure 4.7: *σ_α^2 as a function of l and α. The points show the linearised variance $(12\sigma^2/\sigma_\alpha^2)^{1/3}$ over n' for 5411 fitted lines from 10 different images of various real-life street scenes; the steps represent the function $\lceil n' + 0.5\rceil$.*

be used to further speed up computations while retaining the full power of error propagation.

4.3.3 Incremental Fit

Equation (4.17) is especially advantageous when doing an incremental fit along a contour of linked edgels $\mathbf{x}_i$, adding one edgel at a time. Equation (4.18) and (4.19) are easily updated using

$$\bar{\mathbf{x}}' = \frac{n \cdot \bar{\mathbf{x}} + \mathbf{x}_{n+1}}{n+1} \tag{4.24}$$

$$\mathbf{M}_{\mathbf{x}'\mathbf{x}'} = \frac{\sigma^2 n \mathbf{M}_{\mathbf{xx}} + n\bar{\mathbf{x}}\bar{\mathbf{x}}^\mathsf{T} + \mathbf{x}_{n+1}\mathbf{x}_{n+1}^\mathsf{T} - (n+1)\bar{\mathbf{x}}'\bar{\mathbf{x}}'^\mathsf{T}}{\sigma^2(n+1)} \tag{4.25}$$

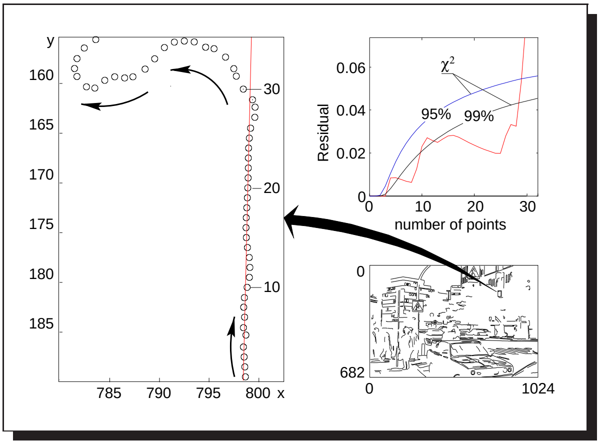


Figure 4.8: *Incrementally fitting a line and mean square error (residual). The circles on the left hand side correspond to a 1σ-region around the edgels.*

(compare (3.13) — these equations simplify considerably if we forego the normalisation in (4.19), which is only needed for the calculation of the error measure $\lambda_{\min}$). In addition to computing the line $\boldsymbol{\ell}$ with highest probability $P(\boldsymbol{\ell}|\mathbf{x}_1 \dots \mathbf{x}_n)$, the task is now also to decide which edgels actually form a line and when to stop fitting. This is commonly done (e. g. in [127]) by finding a seed-region of about a dozen edgels whose mean square error is below a given threshold (say $0.1\,\text{pxl}^2$). Once such a seed region has been found, further edgels will be added using Equations (4.24) and (4.25) until the mean square error (4.17) exceeds the fixed threshold.

There are three problems with this approach:

1. A comparatively large seed region is needed for the mean square error to be meaningful. This makes it impossible to fit small line segments of only a few pixels length.
2. Even then the seed-region is often fitted not to a real line segment but to the slightly curved corner segment which frequently leads up to a line segment.
3. Once a line has reached a certain length, it is easy to overshoot its end, since it takes several "bad" edgels to raise the mean square error above the threshold.

I will instead present in the following a new approach which remedies both Items 1 and 2 and helps to mitigate 3. Although fairly straightforward, this has to my knowledge not been described outside [6], namely that incrementally fitting a line to points can also be thought of as confidence testing, compare Section 3.4. Here the hypothesis is that the point belongs to the line, and we test whether there is sufficient reason for confidence in this hypothesis (or, rather, not enough reason to disbelieve that hypothesis). One possible test is the χ^2-test. Using a confidence test corresponds to using a variable threshold, whose value increases with line-length, instead of a fixed threshold. This is done by evaluating

$$\lambda_{\min} = \frac{1}{n\sigma^2}\boldsymbol{\ell}^\mathsf{T}\left(\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\mathsf{T}\right)\boldsymbol{\ell} \leq \chi^2_{p,n-2} \tag{4.26}$$

(compare Equations (4.17) and (4.19) as well as Figure 4.8), where p is the required probability (confidence level, really) of the outcome and $n-2$ is the number of degrees of freedom of fitting a line to n edgels. This approach allows us to overcome problems 1 and 2, as can be seen from Fig. 4.8: the threshold depends on the number of edgels to which we are currently fitting, so that we can easily discriminate between linear and nonlinear patches even for very short line segments. The third problem, the problem of overshooting, can unfortunately not be solved but only mitigated by the use of a χ^2 error measure — when fitting to some ten edgels it is often possible that two or three bad edgels are needed to drive the error measure (4.26) over the threshold. Other methods are therefore needed to overcome this particular problem, e. g. refitting from the end of the line towards the beginning, or a curvature analysis of the error-behaviour. In [127] the problem is worked around by dropping the last m edgels.

4.3.4 Summary

In Section 4.3 I started with a list of previously used line-parameterisations and introduced a new parameterisations, the $(\alpha, x, y)^\mathsf{T}$ parameterisation. Starting from first principles in Section 4.3.2 I then revisited the problem of fitting a line through a number of feature points. Two different methods were introduced in Section 4.3.2 — an iterative method developed by Kanatani and Kanazawa [73, 77], and a direct least squares approach. In Section 4.3.2.2 I gave a detailed derivation under which circumstances a direct least squares solution is reasonable and showed that in fitting a line through edgels these can be said to be to a good approximation independently, identically, and isotropically distributed (iiid). In this case virtually no difference in accuracy exists between the two methods, and the direct approach should be used, since it is always faster — Kanatani's method solves a problem equivalent to the direct approach in each iteration. Kanatani's method, on the other hand, is preferable whenever points are used which cannot be said to be identically and isotropically distributed, e. g. if points from other sources are used in addition to edgels. This was not explicitly demonstrated in this section, but we will see an example of this in Section 7.

I also gave a new approximate formula for the line's covariance, which can be used to further speed up calculations in the case where we are fitting to iiid edgels, and I demonstrated its excellent agreement with the true covariance values. In Section 4.3.3 I finally extended the direct least squares solution to incremental line fits, where in each step an additional edgel is added to (or removed from) the line. Here I described the use of a simple χ^2 error measure as a sliding threshold which allows to fit lines through fewer edgels than was previously possible.

4.4 Points

It has already been mentioned in Section 4.3 that lines can convey important information about man-made environments. In particular I will define points mainly as the intersection of lines. This is true both for corners, which can in general be defined as the point of intersection of two or more spatially close lines, but in particular also for parallel lines, which identify main directions in the world. These lines, which are parallel in 3D, will be coincident (concurrent) in the image, and it is therefore only logical to search for coincident lines — and their point of intersection — when looking for either corners or a structure that originally consisted of parallel lines. In the following I will mainly concentrate on originally parallel lines as this is generally the more complicated case. Corners, which are usually only of interest in or near the image, follow the same approach, but generally are easier to compute.

Calculating corners or vanishing points (the intersection of originally parallel lines) really consists of two problems. First we need to identify possible candidate lines which we believe to intersect all in the same point; only then can we compute the most likely intersection of all these candidate lines. Standard approaches for the former, which is not the subject of this section, are e. g. the Hough transform [13, 51, 94, 107, 142, 159] (see [20, 93] for more references), RANSAC [119, 134], or perceptual grouping algorithms [88, 89, 96]. The latter are usually based on the assumption that a normal camera was used to create the image rather than some arbitrary projective transformation. Section 5 gives an example of a perceptual grouping algorithm used for vanishing-point detection, where the individual lines bounding a pedestrian crossing are identified based on constraints on the camera-position; Section 6 gives another example where constraints on nearness and direction are used to group candidates for corners. Although not the subject of this thesis I would like to point out that of course Hough and RANSAC like algorithms could benefit from a statistical approach too; Hough through the use of covariance regions rather than lines when building the array [99], while the χ^2 measure suggests itself as a reliable criterion to determine membership in a class when using RANSAC.

Once a set of candidate lines has been identified we are left with the task of finding the point $\mathbf{x}$ which is the most likely intersection of n lines $\boldsymbol{\ell}_i$, $i = 1 \dots n$. This does sound like the dual problem to fitting a line to n points as discussed in Section 4.3 — this is in fact *not* the case, since this intersection can potentially lie at infinity, in which case the weighted Euclidean distance implicit in Equations (4.6) to (4.10) is not defined — the reason we could use this approach in Section 4.3 is the fact that both the edgels and a line fitted to these edgels are restrained to lie within the image, which is not necessarily the case here. The approach by Kanatani described in Section 4.3.2.1, which minimised algebraic distance (distances on the surface of a unit sphere) rather than Euclidean one, is of course still usable and will be discussed in Section 4.4.1.

Minimising algebraic distance (at least the particular one used by Kanatani) works very well, and is indeed nearly always the method of choice when calculating intersections. There exists, however, a very different interpretation to the problem of vanishing-point detection, and it is quite instructive to have a closer look at this approach. It is based on the assumption that the lines were indeed originally parallel in 3D. Instead of calculating the location of the intersection in the image, all lines are first transformed into a canonical frame of parallel lines. The homography connecting the two frames directly specifies the location of the intersection. This is described in Section 4.4.2. If instead of the lines the original edgels are transformed into this frame, we get a new set of lines for free which all pass through the calculated intersection. This approach is described in Section 4.4.2.2. A comparison of the individual approaches and their respective advantages and disadvantages can be found in Section 4.4.3.

4.4.1 Minimising Algebraic Distance

Finding the point $\mathbf{x}$ with $\|\mathbf{x}\| = 1$ which minimises the algebraic distance to n lines $\boldsymbol{\ell}_i$, $i = 1 \dots n$, is the dual problem of finding the line $\boldsymbol{\ell}$ with $\|\boldsymbol{\ell}\| = 1$ which minimises the algebraic distance to a set of points as described in Section 4.3.2.1. The algorithm given there can be applied here by interchanging all occurrences of

$\mathbf{x}$ and $\boldsymbol{\ell}$, the distance measure to minimise then becomes

$$\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbf{x}^{\mathsf{T}} \boldsymbol{\ell}_i \boldsymbol{\ell}_i^{\mathsf{T}} \mathbf{x}}{\mathbf{x}^{\mathsf{T}} \boldsymbol{\Sigma}_{\boldsymbol{\ell}_i} \mathbf{x}} \tag{4.27}$$

The geometric interpretation is similar to the one in Section 4.3.3, again using the Gaussian sphere (or ray-space) model from Section 2.9: each line in the image corresponds to a great circle on the unit sphere (a plane through the unit-sphere's origin), and minimising the algebraic distance finds the point $\mathbf{x}$ on the unit sphere with minimum mean squared scaled orthographic distance to the great circles (the planes). This is equivalent to representing each great circle (plane) by its normal-vector (a point on the unit sphere) and finding the great circle (plane through the sphere's origin) which best fits these points — the problem solved in Section 4.3.2.1. The great-circle's (plane's) normal vector corresponds to the intersection $\mathbf{x}$.

We will see in Section 4.4.3 that the nonlinear distortion of the covariance-regions inherent to the projection onto the unit sphere is in fact more adequate to the problem than the original formulation on the Euclidean plane (although it is still not perfect). This is one of the reasons why this approach works so well.

4.4.2 Canonical Frame Minimisation

The underlying assumption when looking for some lines' common intersection is often that the lines were originally parallel in 3D. So instead of looking for a point closest in (algebraic) distance to a given set of lines, this approach tries to identify a homography H into a canonical frame such that all lines become horizontal[6] again. The intersection is then the point given by $\mathbf{x} = \mathsf{H}^{-1}(1,0,0)^{\mathsf{T}}$ (assuming horizontal lines in this and the following equations).

A general homography H has 8 degrees of freedom. The condition that all lines should be parallel after the transformation $\boldsymbol{\ell}'_i = \mathsf{H}^{-\mathsf{T}} \boldsymbol{\ell}_i$ only fixes 2 DOF (one for an intersection at infinity, and one for the angle which the lines form with the x-axis). We therefore have to decide on a 2 degree of freedom parametrisation for

[6]Or any other predefined direction.

H which uniquely identifies H. A possible parametrisation is given by

$$\mathsf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.28}$$

$$\mathsf{H}^{-\mathsf{T}} = \begin{pmatrix} 1 & 0 & -p \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.29}$$

This parametrisation consists of a rotation around the image origin, followed by a (possibly negative) projective foreshortening along the x-axis.

The homography H can now be used to transform the lines $\boldsymbol{\ell}_i$ into a canonical frame of horizontal lines $\boldsymbol{\ell}'_i = \mathsf{H}^{-\mathsf{T}}\boldsymbol{\ell}_i$, or alternatively to directly transform the edgels $\mathbf{x}_{i,j}$ which originally gave rise to the lines $\boldsymbol{\ell}_i$, $\mathbf{x}'_{i,j} = \mathsf{H}\mathbf{x}_{i,j}$. The later would, in addition to the lines' intersection $\mathbf{x}$, also calculate a new set of lines $\hat{\boldsymbol{\ell}}_i$, which will all pass exactly through this intersection. Both approaches are described below.

4.4.2.1 Canonical Frame and Lines

We assume that the lines are given as a homogeneous 3-vector proportional to the normal form characterised by the angle α_i and the distance $c_i = -x_i \sin(\alpha_i) + y_i \cos(\alpha_i)$ from the origin as $\boldsymbol{\ell}_i = k_i(\sin(\alpha_i), -\cos(\alpha_i), c_i)^\mathsf{T}$. These will then transform as

$$\boldsymbol{\ell}'_i = \mathsf{H}^{-\mathsf{T}}\boldsymbol{\ell}_i = k_i \begin{pmatrix} \sin(\alpha_i + \beta) - c_i p \\ -\cos(\alpha_i + \beta) \\ c_i \end{pmatrix}^\mathsf{T} \tag{4.30}$$

Since we have no information about the transformed lines other than that they should all be horizontal, we can only use the transformed lines' angles to construct an error measure. The angle is

$$\alpha'_i = \arctan\left(\tan(\alpha_i + \beta) - \frac{c_i p}{\cos(\alpha_i + \beta)}\right) \tag{4.31}$$

Following an argument similar to the one in Section 4.3, the equation to minimise would be

$$\chi^2 = \min_{\beta,p} \sum_{i=1}^{n} \frac{{\alpha'_i}^2}{\sigma^2_{\alpha'_i}} \tag{4.32}$$

where $\sigma^2_{\alpha'_i}$ can be calculated using Equation (3.53) as

$$\sigma^2_{\alpha'_i} = \mathbf{J}_{\alpha'_i(\alpha_i,c_i)} \mathbf{J}_{(\alpha_i,c_i)\boldsymbol{\ell}_i} \boldsymbol{\Sigma}_{\boldsymbol{\ell}_i} \mathbf{J}^{\mathsf{T}}_{(\alpha_i,c_i)\boldsymbol{\ell}_i} \mathbf{J}^{\mathsf{T}}_{\alpha'_i(\alpha_i,c_i)} \tag{4.33}$$

$$\text{with} \qquad \mathbf{J}_{\alpha'_i(\alpha_i,c_i)} = \frac{(c_i p \sin(\alpha_i+\beta) - 1\,,\, p\cos(\alpha_i+\beta))}{2c_i p \sin(\alpha_i+\beta) - p^2 c_i^2 - 1} \tag{4.34}$$

$$\text{and} \qquad \mathbf{J}_{(\alpha_i,c_i)\boldsymbol{\ell}_i} = \begin{pmatrix} 1 & 0 & 0 \\ -x_i\cos(\alpha_i) - y_i \sin(\alpha_i) & -\sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{4.35}$$

if the line was given in $(\alpha, x, y)^{\mathsf{T}}$ notation — or similarly for any other notation.

If speed of computation is an issue, then the following approximation is viable: the angle in Equation (4.32) is well approximated by the tangent function under the assumption of small to medium sized errors (the relative error is only about 4 % for up to 20° of error in the orientation); we can therefore minimise the slightly simpler function

$$\chi^2 = \min_{\beta,p} \sum_{i=1}^{n} \frac{{t'_i}^2}{\sigma^2_{t'_i}} \tag{4.36}$$

$${t'_i}^2 = \tan(\alpha_i+\beta) - \frac{c_i p}{\cos(\alpha_i+\beta)} \tag{4.37}$$

$$\sigma^2_{t'_i} = \mathbf{J}_{t'_i(\alpha_i,c_i)} \mathbf{J}_{(\alpha_i,c_i)\boldsymbol{\ell}_i} \boldsymbol{\Sigma}_{\boldsymbol{\ell}_i} \mathbf{J}^{\mathsf{T}}_{(\alpha_i,c_i)\boldsymbol{\ell}_i} \mathbf{J}^{\mathsf{T}}_{t'_i(\alpha_i,c_i)} \tag{4.38}$$

$$\mathbf{J}_{t'_i(\alpha_i,c_i)} = \left(\frac{1 - c_i p \sin(\alpha_i+\beta)}{\cos^2(\alpha_i+\beta)}\,,\, -\frac{p}{\cos(\alpha_i+\beta)} \right) \tag{4.39}$$

and (4.35) as given above.

The most likely intersection in this frame is located at $(1, 0, 0)^{\mathsf{T}}$ (in homogeneous coordinates), and this is mapped onto the homogeneous image coordinate

$$\mathbf{x} = \begin{pmatrix} \cos(\beta) \\ -\sin(\beta) \\ -p \end{pmatrix} \tag{4.40}$$

Its covariance matrix can be calculated using Equations (3.53) and (3.56) (using the implicit function theorem).

Both Equation (4.32) and Equation (4.36) result in moderately complicated non-linear functions of the parameters β and p, for which no closed form solution exists. In order to locate the minimum, any of a number of numerical optimisation schemes [123] can be employed instead. A reasonable start-value is important to honour the small error assumption explicitly used for Equation (4.36), and implicitly present in the use of Gaussian distributions for angles in Equation (4.32). A reasonable start value could, for example, be found by one or two iterations of the algebraic distance algorithm in Section 4.4.1.

4.4.2.2 Canonical Frame and Edgels

If the original edgels are still available, rather than just the lines, it is tempting to directly transform the edgels into a canonical frame. Only the y-coordinates of each transformed edgel are of interest (assuming once more a frame of horizontal lines), and the function which is to be minimised becomes

$$\chi^2 = \min_{\beta,p} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \frac{\left(y'_{i,j} - \bar{y}'_i\right)^2}{\sigma^2_{y'_{i,j}}} \tag{4.41}$$

where

$$y'_{i,j} = \frac{x_{i,j}\sin(\beta) + y_{i,j}\cos(\beta)}{p x_{i,j}\cos(\beta) - p y_{i,j}\sin(\beta) + 1} \tag{4.42}$$

$$\bar{y}'_i = \sum_{j=1}^{m_i} \frac{y'_{i,j}}{\sigma^2_{y'_{i,j}}} \Big/ \sum_{j=1}^{m_i} \frac{1}{\sigma^2_{y'_{i,j}}} \tag{4.43}$$

$$\sigma^2_{y'_{i,j}} = \mathbf{J}_{y'_{i,j}\mathbf{x}_{i,j}} \boldsymbol{\Sigma}_{\mathbf{x}_{i,j}} \mathbf{J}^{\mathsf{T}}_{y'_{i,j}\mathbf{x}_{i,j}} \tag{4.44}$$

$$\mathbf{J}_{y'_{i,j}\mathbf{x}_{i,j}} = \frac{\left(p y_{i,j} - \sin(\beta)\,,\; p x_{i,j} + \cos(\beta)\right)}{\left(p x_{i,j}\cos(\beta) - p y_{i,j}\sin(\beta) + 1\right)^2}. \tag{4.45}$$

$\mathbf{x}_{i,j}$ is the j^{th} edgel of the i^{th} line. Again, Equation (4.41) can only be minimised numerically. In addition to the approach in Section 4.4.2.1, this strategy also calculates a new set of lines of the form $\boldsymbol{\ell}'_i = (0, -1, \bar{y}_i)^{\mathsf{T}}$ in the canonical frame or

$$\hat{\boldsymbol{\ell}}_i = \frac{1}{\sqrt{p^2\bar{y}_i^2 + 1}} \begin{pmatrix} p\bar{y}_i\cos(\beta) - \sin(\beta) \\ -p\bar{y}_i\sin(\beta) - \cos(\beta) \\ \bar{y}_i \end{pmatrix} \tag{4.46}$$

in the image. Such refitted lines can, e. g., be used for the calculation of the cross-ratio as described in Section 4.5, although in Section 4.5.2 I will present a new method which allows the calculation of the cross-ratio *without* prior calculation of the intersection and a new set of lines, at nearly the same accuracy.

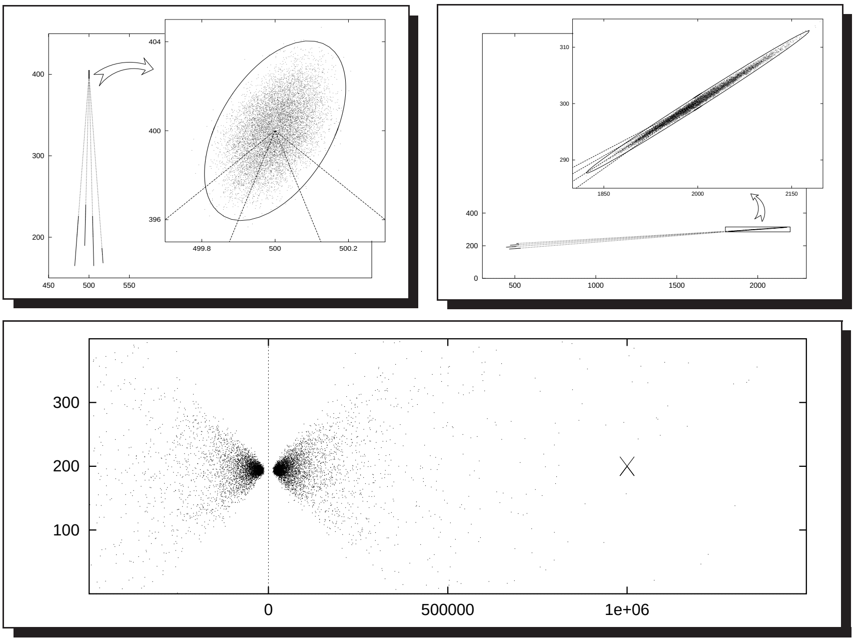


Figure 4.9: *Monte Carlo simulation of the calculation of line-intersections for three typical constellations. The two top figures demonstrate how well the theoretically derived covariance matrix (the ellipses, enclosing 99 % of all points) and the Monte-Carlo simulation agree for intersections close to the image. The bottom figure shows that for far away intersections (×) the covariance-region becomes hyperbolic in Euclidean coordinates, which is poorly modelled by Gaussian noise.*

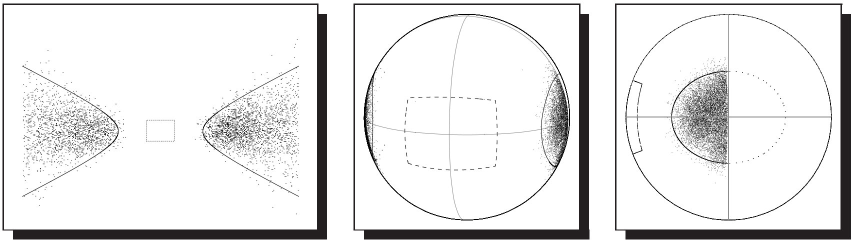


Figure 4.10: *The projection of a hyperbolic covariance region (left) onto a sphere (middle and right) becomes elliptical. The image region is marked by a dashed rectangle.*

4.4.3 Comparison and Summary

A Monte Carlo simulation shows that both the algebraic distance as well as the canonical frame algorithms work equally well under the condition of small errors. Figure 4.9 shows sample scatter-diagrams for three typical constellations, using the algorithm from [73] (both types of algorithms produce virtually identical diagrams). It becomes apparent that both types of algorithms work best when the intersection is located in or near the image (Figure 4.9, top row). The hyperbolic covariance-region in Figure 4.9, bottom row, shows clearly why an Euclidean model is not adequate for the calculation of these intersections, as was mentioned in the introduction to this section.

The spherical normalisation, and in particular Kanatani's N-vectors, are indeed much better adapted to the problem than are Euclidean coordinates. We can see in Figure 4.10 that the projection of a hyperbola onto the sphere will result in an "elliptic" region, in this case with the midpoint close to "infinity" and the covariance-region wrapping around both sides of the unit-(half)-sphere. The left figure shows a setup similar to the one in Figure 4.9 (bottom), with the image region marked by a dashed rectangle and the scatter-plot delimited by a hyperbola containing 99 % of the intersections. Figure 4.10 (middle and right) show the projection onto the unit sphere, using N-vectors, where the intersections are now well contained within an approximately elliptical region on the sphere, which would be well modelled by Gaussian error on the sphere.

The canonical-frame method is of course not affected by these considerations at all, as it only uses angular distances, which do not suffer from these effects.

I have said that both types of algorithms produce comparable results. However, Kanatani's method is up to an order of magnitude faster for small covariance matrices, where it converges rapidly. Processing times for larger uncertainties are about equal, but here Kanatani's method calculates slightly more accurate results, since the small-error condition needed for Equation (4.32) is not valid anymore, and his algorithm is therefore the method of choice for the fast calculation of line-intersections.

The decision whether to accept the hypothesis that the n lines tested were indeed coincident can as usual be based on a χ^2 test with $\nu = n - 2$ degrees of freedom.

4.5 The Crossratio

The cross-ratio of 4 collinear points — or, alternatively, 4 coincident lines — is one and probably the fundamental projective invariant. However, virtually no four measured points will be collinear, nor will four measured lines be coincident, even if we know that they should be. In the following we will mainly concentrate on the crossratio of four lines, since for the reasons discussed in Sections 4.3 and 4.4 (fitting lines is easier than calculating their intersection) this is usually the more error-prone case.

Two radically different approaches to the calculation of the crossratio are described below. The family of solutions described in Section 4.5.1 calculates the lines' intersection first and the crossratio only afterwards, thereby following a more conservative approach. In contrast, the approach described in Section 4.5.2 does not require any knowledge about the lines' intersection. This is a direct consequence of the application of error-propagation principles to the problem of the calculation of the crossratio, and has to my knowledge not been discussed outside of [6] and this thesis.

Section 4.5.3 compares the different approaches and shows clearly the crossratio's sensitivity to noise and to the particular method chosen to compute it — but also that with some care and the tools of error-analysis and error propagation it is possible to construct an algorithm which can calculate the cross-ratio as good as the best algorithms, but in a fraction of the time usually needed.

4.5.1 Refitting Lines

The approach for the calculation of the crossratio described in this section could be termed the classical approach. Given four lines $\boldsymbol{\ell}_i$, $i \in \{A, B, C, D\}$, which are not quite coincident, one first calculates the lines' most likely common intersection

— hopefully by one of the methods discussed in Section 4.4. From there a new set of lines $\hat{\boldsymbol{\ell}}_i$ is calculated, which passes through this intersection. Only then is the crossratio calculated, usually using Equation (2.42).

Three different representations of this approach are introduced in the following, corresponding to the three methods for the calculation of the lines' intersection in Section 4.4.

4.5.1.1 Refitting in the Image Plane

In this approach, which was suggested by Kanatani[74] and others, one first calculates the lines' intersection in the image plane, using the algebraic distance algorithm described in Section 4.4.1. From there a set of new lines $\hat{\boldsymbol{\ell}}_i \in \mathbb{R}^3$ is calculated which pass through this intersection and the original line segments' centres. One can then calculate the new lines' crossratio using Equation (2.42),

$$\operatorname{cr}(\hat{\boldsymbol{\ell}}_A\hat{\boldsymbol{\ell}}_B\hat{\boldsymbol{\ell}}_C\hat{\boldsymbol{\ell}}_D) = \frac{|\hat{\boldsymbol{\ell}}_A\hat{\boldsymbol{\ell}}_C\boldsymbol{\ell}_X|}{|\hat{\boldsymbol{\ell}}_B\hat{\boldsymbol{\ell}}_C\boldsymbol{\ell}_X|} \cdot \frac{|\hat{\boldsymbol{\ell}}_B\hat{\boldsymbol{\ell}}_D\boldsymbol{\ell}_X|}{|\hat{\boldsymbol{\ell}}_A\hat{\boldsymbol{\ell}}_D\boldsymbol{\ell}_X|}. \tag{2.42 a}$$

$\boldsymbol{\ell}_X$ can be any arbitrary 3-vector; the only formal criterion for $\boldsymbol{\ell}_X$ is that the scalar product with the vanishing point must not be 0 (the line $\boldsymbol{\ell}_X$ must not pass through the vanishing point). It is therefore often chosen to be the dual of the vanishing point itself [72].

4.5.1.2 Canonical Frame and Lines

In this approach all four lines are transformed into a canonical frame of horizontal lines, using the method described in Section 4.4.2.1. The crossratio is then calculated from the y-coordinates of the transformed lines' centre points alone, using e. g. Equation (2.37), i. e.

$$\operatorname{cr}(\hat{\boldsymbol{\ell}}_A\hat{\boldsymbol{\ell}}_B\hat{\boldsymbol{\ell}}_C\hat{\boldsymbol{\ell}}_D) = \frac{\hat{y}_C - \hat{y}_A}{\hat{y}_C - \hat{y}_B} \cdot \frac{\hat{y}_D - \hat{y}_B}{\hat{y}_D - \hat{y}_A}. \tag{2.37 a}$$

4.5.1.3 Canonical Frame and Edgels

Here the edgels are transformed into a canonical frame, instead of the lines, using the approach described in Section 4.4.2.2. Only the $\bar{y}'_i$ are then used for the calculation of the crossratio, again using e. g. Equation (2.37).

All three approaches described so far are basically equivalent in that they first calculate the lines' vanishing point (which is implicitly given by a canonical frame representation) and only then proceed to calculate the crossratio, differing only in the method used to calculate the vanishing point itself, and they will therefore all give similar results. However, the necessity to find a good approximation of the vanishing-point position means that all three algorithms are rather slow. While this is not a problem if only one or two crossratios need to be calculated, it can become a considerable burden if several thousand calculations are needed, e. g. if the crossratio is used for classification into two or more populations, where the biggest population would be due to random configurations of lines and could safely be ignored. It would then be convenient if a fast method for the calculation of the crossratio existed that would not require the calculation of the vanishing point. A possible approach is presented in the next section.

4.5.2 Direct calculation of the Crossratio

The main problem with any approach for the calculation of the crossratio that would not also calculate the vanishing point is of course the fact that a crossratio is computed for 4 non-coincident lines, although it is only defined for coincident ones (or equivalently 4 collinear points).

But what if we know that the original lines really were coincident? The formula usually used to compute the cross-ratio

$$\operatorname{cr}(\boldsymbol{\ell}_A\boldsymbol{\ell}_B\boldsymbol{\ell}_C\boldsymbol{\ell}_D) = \frac{|\boldsymbol{\ell}_A\boldsymbol{\ell}_C\boldsymbol{\ell}_X|}{|\boldsymbol{\ell}_B\boldsymbol{\ell}_C\boldsymbol{\ell}_X|} \cdot \frac{|\boldsymbol{\ell}_B\boldsymbol{\ell}_D\boldsymbol{\ell}_X|}{|\boldsymbol{\ell}_A\boldsymbol{\ell}_D\boldsymbol{\ell}_X|}. \tag{2.42 b}$$

which can e. g. be found in [72, 103], does not make explicit use of the condition of coincidence. The result might be meaningless if the lines are not coincident (and certainly highly dependent on $\boldsymbol{\ell}_X$, which can be arbitrarily chosen, as long as it does not pass through the vanishing point), but it can be calculated.

It is therefore interesting to analyse the quality of the result. The key here is the choice of $\boldsymbol{\ell}_X$ in dependence of the four lines $\boldsymbol{\ell}_i$ and their covariances. The optimal choice for $\boldsymbol{\ell}_X$ could in theory be computed by minimising the cross-ratio's variance — calculated using Equation (3.53) — with respect to $\boldsymbol{\ell}_X$. Unfortunately, no closed form solution exists, and numerical minimisation would be just as computationally expensive as the direct computation of the vanishing point in Section 4.5.1.

What can be said, however, is that $\boldsymbol{\ell}_X$ should intersect the $\boldsymbol{\ell}_i$ close to their centre points, and at right angles. This can be seen if we interpret $\boldsymbol{\ell}_X$ as a line intersecting the other four lines $\boldsymbol{\ell}_i$, $i \in \{A, B, C, D\}$, as seen in Figure 4.11. If the lines $\boldsymbol{\ell}_i$ are

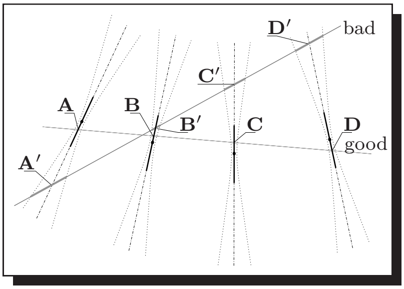


Figure 4.11: *The choice of $\boldsymbol{\ell}_X$ influences the accuracy of the fast algorithm.*

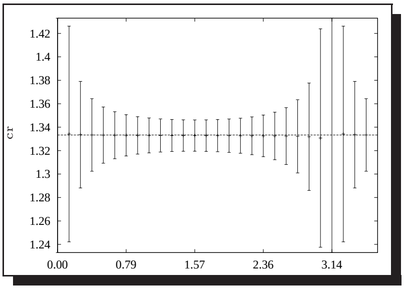


Figure 4.12: *The sample crossratio $\bar{\mathrm{cr}}(\alpha_{\boldsymbol{\ell}_X}) \pm s_{\bar{\mathrm{cr}}}(\alpha_{\boldsymbol{\ell}_X})$ for different angles $\alpha_{\boldsymbol{\ell}_X}$, Monte-Carlo simulation.*

not perfect lines but ones fitted to a number of edgels, it is clear that the fitted lines will become less and less accurate the further away from the line segments' centre points we go — the dotted hyperbolas to the left and right of each line segment in Figure 4.11 represent the border of an n-σ-interval. This means conversely that the resulting cross-ratio will be all the more reliable, the closer to their centre points the lines $\boldsymbol{\ell}_i$ will be intersected by $\boldsymbol{\ell}_X$, but also the closer to a right angle the angle between $\boldsymbol{\ell}_i$ and $\boldsymbol{\ell}_X$ has been chosen. Figure 4.12 gives an error-bar representation of the crossratio of 4 *horizontal* and equidistant lines ($\mathrm{cr} = 4/3$) as a function of that angle by plotting $\bar{\mathrm{cr}}(\alpha_{\boldsymbol{\ell}_X}) \pm s_{\bar{\mathrm{cr}}}(\alpha_{\boldsymbol{\ell}_X})$ (the sample variance) for different angles $\alpha_{\boldsymbol{\ell}_X}$. It can clearly be seen that the calculated crossratio becomes completely unreliable if the intersecting line $\boldsymbol{\ell}_X$ is approximately collinear with the line segments, but is else not overly sensitive to small changes in the orientation[7] — suggesting that it might indeed be possible to calculate a reasonable approximation of the crossratio without prior calculation of the vanishing point.

Even if the accuracy of a cross-ratio thus calculated were lower than possible, one could still use this as a first test using the χ^2-test described in Section 4.6.4 and only calculate the vanishing point and a better approximation[8] to the cross-ratio and its covariance if it did pass the test, using one of the approaches discussed in Section 4.5.1.

This would already considerably speed up processing compared with the straight-

[7]More complicated line-constellations can of course results in a less benign relationship.

[8]It is worth remembering that any algorithm will only compute an approximation unless the measured lines have been coincident to begin with.

forward approach of first calculating the vanishing point and then the cross-ratio, since the cross-ratio is both much faster to compute and usually has much higher discriminating power[9]. Note that even if the new fast algorithm were less accurate than the more conservative approach, no information will be lost. Only the number of false positives would increase. We will, however, see that with a carefully chosen $\boldsymbol{\ell}_X$ the accuracy of the fast algorithm can be virtually as good as that of the more traditional ones.

The only remaining question is how to best choose the $\boldsymbol{\ell}_X$. We have seen above that some weighted equilibrium needs to be found between a line $\boldsymbol{\ell}_X$ which intersects the $\boldsymbol{\ell}_i$ as close as possible to their centres, and a right-angle intersection between $\boldsymbol{\ell}_X$ and the $\boldsymbol{\ell}_i$s, since it is generally not possible to fulfil both requirements at the same time. In the following I will present two approaches which put more weight on either the first or second condition; the approach described in Section 4.5.2.1 tries to maintain as-close-as-possible right-angle intersections, while the approach described in Section 4.5.2.2 tries to pass a line close to the individual lines' centre points. Many more approaches can be conceived; the two approaches used here have mainly been chosen for their didactic properties — the first one works particularly bad, the second one particularly good, mostly due to the weighting chosen.

4.5.2.1 Right Angle Intersection

Keeping the intersections between $\boldsymbol{\ell}_X$ and the $\boldsymbol{\ell}_i$s as close as possible to a right angle is equivalent to finding the line orthogonal to a line through the vanishing point and with minimum average angular distance to the four lines $\boldsymbol{\ell}_i$. This determines a family of parallel lines, and fixing the remaining parameter requires us to specify a point through which $\boldsymbol{\ell}_X$ is expected to pass, the mean position of the four lines' centre points suggesting itself.

However, the above description requires us to know the position of the vanishing point. In order to avoid calculating the vanishing points position we will use a slightly modified approach; we will choose a line which passes through the point $(\bar{x}, \bar{y})^\mathsf{T}$ and whose homogeneous normal vector $\mathbf{n} \in I\!R^3$ is given as the vector pointing from that point into the direction of the intersection between $\boldsymbol{\ell}_A$ and $\boldsymbol{\ell}_D$,

[9]For most scenes taken from human environments we will usually have only $2n + 1$ (with n small, often $n = 1$) vanishing points through which most lines pass, compare Section 6. Coincidence alone therefore has only very little discriminating power.

i. e.

$$\mathbf{n} = (\boldsymbol{\ell}_A \times \boldsymbol{\ell}_D) \times \begin{pmatrix} \bar{x} \\ \bar{y} \\ 1 \end{pmatrix} \tag{4.47}$$

and the resulting line becomes

$$\boldsymbol{\ell}_X = \begin{pmatrix} n_x \\ n_y \\ -\bar{x}n_x - \bar{y}n_y \end{pmatrix}. \tag{4.48}$$

The point $(\bar{x}, \bar{y})^\mathsf{T}$ itself, rather than using the centre-points' mean position, will be calculated as

$$\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \left(\sum_{i \in \{A,B,C,D\}} \begin{pmatrix} \sigma^2_{x_i} & 0 \\ 0 & \sigma^2_{y_i} \end{pmatrix} \right)^{-1} \sum_{i \in \{A,B,C,D\}} \begin{pmatrix} \sigma^2_{x_i} & 0 \\ 0 & \sigma^2_{y_i} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{4.49}$$

This will be done for didactic reasons only, generating comparable scales in Figure 4.14ff.

Calculating the direction of $\boldsymbol{\ell}_X$ completely ignores the individual lines' covariance matrices, and it is therefore not overly surprising that the results achievable will remain well below that of the standard approaches, as can be seen in Section 4.5.3.

4.5.2.2 Line Fit

This approach tries to fit $\boldsymbol{\ell}_X$ to the $\boldsymbol{\ell}_i$'s centre points using a weighted fit. It is easy to see from Figure 4.11 that the bigger the variance $\sigma^2_{\alpha_i}$ in the angle α_i, the closer should $\boldsymbol{\ell}_X$ be to the line's centre point $\mathbf{x}_i$, and this makes minimising the following equation a good candidate

$$\min_{\boldsymbol{\ell}_\mathbf{X}} \sum_{i \in \{A,B,C,D\}} \sigma^2_{\alpha_i} \boldsymbol{\ell}_X^\mathsf{T} \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix} (\mathbf{x}_i, 1) \boldsymbol{\ell}_X \tag{4.50}$$

with $\boldsymbol{\ell}_\mathbf{X} \in I\!R^3$ and $\mathbf{x}_i \in I\!R^2$ — the $\mathbf{x}_i$ are centre points of measured line segments and therefore never at infinity[10]. The resulting line $\boldsymbol{\ell}_X$ passes through the point

$$\bar{\mathbf{x}} = \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \frac{\sum_{i \in \{A,B,C,D\}} \sigma^2_{\alpha_i} \mathbf{x}_i}{\sum_{i \in \{A,B,C,D\}} \sigma^2_{\alpha_i}} \tag{4.51}$$

[10]If one or more of the lines stem from a different source and therefore do have their centre points at infinity, then it is safe to ignore the lines as they must by necessity have a value of $\sigma_\alpha = 0$ or will be unusable.

and its normal-vector is the eigenvector to the smaller eigenvalue of

$$\sum_{i\in\{A,B,C,D\}} \sigma^2_{\alpha_i} \begin{pmatrix} (x_i-\bar{x})^2 & (x_i-\bar{x})(y_i-\bar{y}) \\ (x_i-\bar{x})(y_i-\bar{y}) & (y_i-\bar{y})^2 \end{pmatrix} . \tag{4.52}$$

This approach specifically calculates a solution based on the individual line's variance $\sigma^2_{\alpha_i}$ in the orientation α_i, and it is therefore not surprising that it will generally surpass the approach discussed before. Section 4.5.3 will, in fact, show that this method gives results comparable to the ones in Section 4.5.1, and that the result's pdf can be approximated well using Equations (3.52) and (3.53). However, we will see in the next section that the qualitative (and, near enough, quantitative) behaviour is the same for both the conservative, time consuming approach as well as the fast approach presented in Section 4.5.2.2, and this shows impressively the power of error propagation as a tool to devise both fast as well as accurate algorithms.

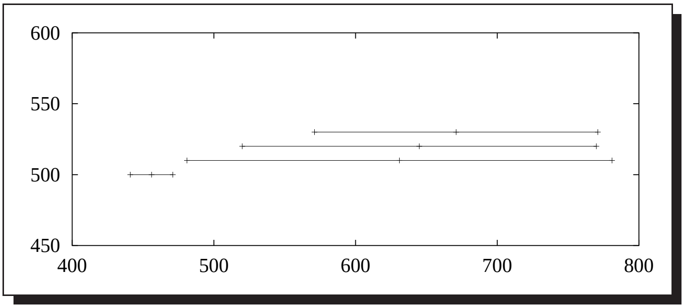
4.5.3 Comparison and Summary

In the following, a number of Monte Carlo simulations run on several different line-configurations are used to illustrate the respective merits of the two approaches — the refitting algorithm mentioned in Section 4.5.1.1 as an example of the group of algorithms mentioned in Section 4.5.1, and the direct calculations from Section 4.5.2. It can be seen that with a proper selection of $\boldsymbol{\ell}_X$, results for the fast algorithm are about as good as for the canonical-frame algorithm if the error is small (compare Figures 4.14 through 4.18), whereas the calculation is much faster. The individual line-sets used have all been calculated from one set of four equally spaced, parallel lines as given in Figure 4.13. These have been subjected to a projective transformation

$$\mathbf{P} = \begin{pmatrix} s\cos(\alpha) & -s\sin(\alpha) & st_x \\ s\sin(\alpha) & s\cos(\alpha) & st_y \\ p_x\cos(\alpha)+p_y\sin(\alpha) & -p_x\sin(\alpha)+p_y\cos(\alpha) & t_xp_x+t_yp_y+1 \end{pmatrix} . \tag{4.53}$$

This is equivalent to a rotation around the origin with angle α, translation by $(t_x, t_y)^{\mathsf{T}}$, projective distortion in x-direction with factor p_x — the horizontal vanishing point (the intersection of the four lines) is projected to $(1/p_x, 0, 1)$ — projective distortion in y-direction with factor p_y — the point at infinity $(0, 1, 0)$ is projected to $(0, 1/p_y, 1)$ — and a possible scaling of the entire set by s.

Note that the scaling operation is *not* equivalent to a change in resolution, as this would influence σ^2_α and $\sigma^2_{x,y}$ differently and in a nonlinear way (compare Figure 4.7



	α	x	y	n	σ_α^2	σ_x^2	σ_y^2
Line 1	π	456	500	30	$2.\bar{2} \cdot 10^{-4}$	$1.\bar{6} \cdot 10^{-2}$	$1.\bar{6} \cdot 10^{-2}$
Line 2	2π	631	510	300	$2.\bar{2} \cdot 10^{-7}$	$1.\bar{6} \cdot 10^{-3}$	$1.\bar{6} \cdot 10^{-3}$
Line 3	π	645	520	250	$3.84 \cdot 10^{-7}$	$2.0 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$
Line 4	2π	671	530	200	$7.5 \cdot 10^{-7}$	$2.5 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$

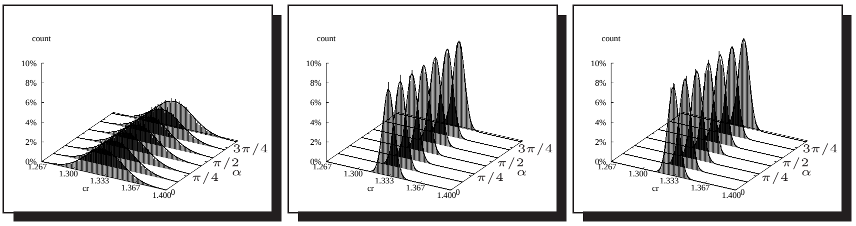
Figure 4.13: *Dataset used for Monte-Carlo simulations (cross-ratio).*

and Equations (4.21)–(4.23)). We can therefore additionally subject the variances in the table in Figure 4.13 to the following transformation:

$$\begin{aligned} {\sigma'_\alpha}^2 &= \sigma_\alpha^2/k^3 \\ {\sigma'_x}^2 &= \sigma_x^2/k \quad . \\ {\sigma'_y}^2 &= \sigma_y^2/k \end{aligned} \tag{4.54}$$

Changing only the scale s describes the case where, with constant line-length, the distance between the lines varies; varying only the factor k would correspondingly describe a setup in which the line-length varies, but the distance between the lines is kept constant. Alternatively this also describes the case where the image quality degrades or improves respectively. Varying both s and k by the same amount finally corresponds to a change in image resolution.

In order to create datasets for Monte-Carlo simulations, Gaussian noise of the appropriate variance is then added to the (α, x, y) values in Figure 4.13. The sample size is 10 000 unless otherwise stated. Each parameter of Equation (4.53) has been changed in turn, and a histogram of the values computed for the crossratio has been plotted together with the predicted distribution as given by the median of the crossratio (the expected value is 4/3) and median predicted variance. Each experiment will be discussed in turn below; it will be seen from Figures 4.14 through 4.18 that the predicted and actual distributions agree nicely. In all experiments the left and middle graphs refer to the two fast methods, while the right one refers to the traditional algorithm, compare e. g. Figure 4.14; the left one, labelled “Right angle” and (a), refers to Section 4.5.2.1 where we tried to get a right-angle intersection, while the middle one, labelled “Line fit” and (b), refers to Section 4.5.2.2 where $\boldsymbol{\ell}_X$ was chosen to pass as closely as possible through the centre points.



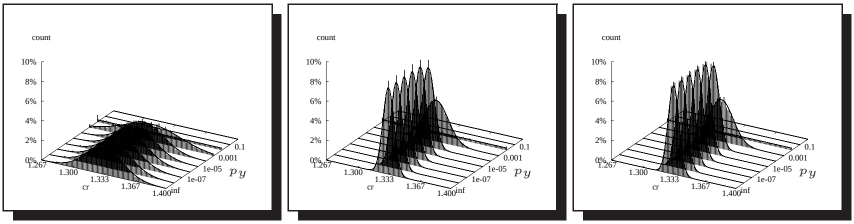
(a) Right angle: rotation by α. **(b)** Line fit: rotation by α. **(c)** Refit: rotation by α.

Figure 4.14: *Measured and predicted distribution of the crossratio for three different algorithms and varying angle of rotation α.*

Figure 4.14 shows histograms of the measured distributions for the crossratio under different angles of rotation α as well as the predicted distribution (median values) using the three methods discussed above; Figures 4.14(a) and 4.14(b) show typical results for the direct (fast) method using different lines $\boldsymbol{\ell}_X$ calculated according to Section 4.5.2.1 in Figure 4.14(a) and Section 4.5.2.2 in Figure 4.14(b) respectively; Figure 4.14(c) shows typical results for the slower, but more exact conservative approach described in Section 4.5.1.1. It can be seen that the distributions are independent of the rotation α, as could have been expected. Also, the measured distribution (histogram) obviously corresponds well with the predicted distribution (solid curve). Finally, it can be seen that for the direct method the distribution of the crossratio depends on the particular line $\boldsymbol{\ell}_X$ chosen, and can be nearly as good as the much slower conventional method for a well-chosen $\boldsymbol{\ell}_X$ — the corresponding values for $\alpha = 0$, e. g., are $\sigma_{4.14(\text{a})} = 0.01973$, $\sigma_{4.14(\text{b})} = 0.00599$, $\sigma_{4.14(\text{c})} = 0.00582$, i. e. only a 3 % difference between the last two.

Most of these observations — the very good correspondence between predicted and measured distribution, the high quality of the direct approach if the line $\boldsymbol{\ell}_X$ is chosen by an inversely weighted line fit according to Section 4.5.2.2, and the lower accuracy when choosing the line $\boldsymbol{\ell}_X$ according to Section 4.5.2.1 — will also be valid for variations of any of the other parameters; the main difference is how the distribution varies with variations of the individual parameters.

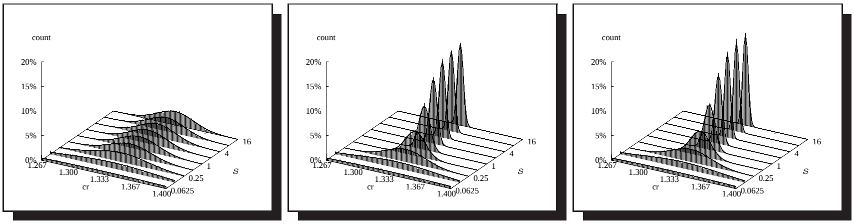
Figure 4.15 shows the measured and predicted distribution for varying projective



(a) Right angle: projective distortion p_y. **(b)** Line fit: projective distortion p_y. **(c)** Refit: projective distortion p_y.

Figure 4.15: *Measured and predicted distribution of the crossratio for three different algorithms and varying projective skew in y-direction p_y.*

distortions in y-direction, results are similar for distortions in x-direction. It can be seen that results remain constant for a wide range of distortions $0 < p_x, p_y < 10^{-4}$ (which was not obvious from the problem itself) but degrade sharply between $10^{-4} < p_x, p_y < 10^{-2}$. This is, however, not overly surprising, as we are then already dealing with rather severe distortions which seriously influence both the actual as well as relative length of the individual lines, a behaviour not modelled by Equations (4.53) and (4.54).

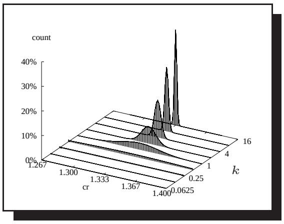


(a) Right angle: scale s. **(b)** Line fit: scale s. **(c)** Refit: scale s.

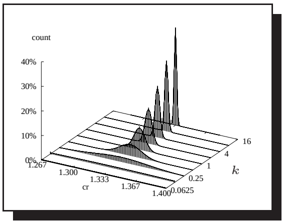
Figure 4.16: *Measured and predicted distribution of the crossratio for three different algorithms and varying scale s.*

Figure 4.16 shows the measured and predicted distributions for varying scales s, varying the distance between the lines while keeping the line-length constant. As was to be expected, the variance increases with decreasing scale (distance between lines) and decreases with increasing scale. It is, however, quite interesting to note that at least the latter is only the case within a relatively small interval around the original setup (approx. $0.1 < s < 10$).

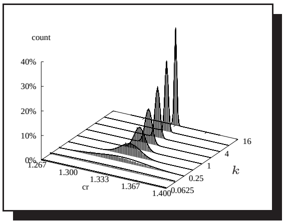
Figure 4.17 shows the measured and predicted distributions for varying (relative)



(a) Right angle: accuracy k.



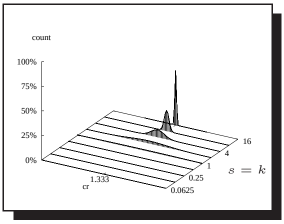
(b) Line fit: accuracy k.



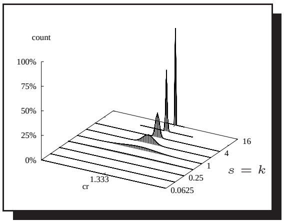
(c) Refit: accuracy k.

Figure 4.17: *Measured and predicted distribution of the crossratio for three different algorithms and varying accuracy k.*

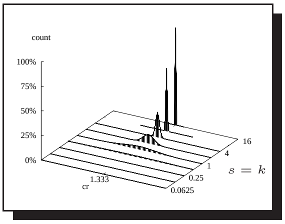
accuracy k, corresponding to a fixed distance between the lines and either varying line-length or varying quality of the original image (e. g., as is noticeable when comparing images taken with a 3-chip RGB camera to those taken with a 1-chip RGB camera). Here it is very clearly the case that the variance decreases with increasing accuracy.



(a) Right angle: resolution $s = k$.



(b) Line Fit: resolution $s = k$.



(c) Refit: resolution $s = k$.

Figure 4.18: *Measured and predicted distribution of the crossratio for three different algorithms and varying resolution $s = k$.*

Figure 4.17 shows the measured and predicted distributions for varying (relative) resolutions $s = k$, i. e. varying both s and k simultaneously. This corresponds directly to a change in resolution. Clearly the variance decreases with increasing accuracy.

Figure 4.19 shows the functional relation between the cross-ratio's variance σ^2_{cr} and the scale s (solid line), accuracies k (dashed line), and resolution $s = k$ (dotted line). Variance changed cubic with resolution, showing that in this case the angle's variance σ^2_α is the dominating factor (compare (4.21)); the same is true at low accuracies. For high accuracies (small σ^2 of the edgels) we have an

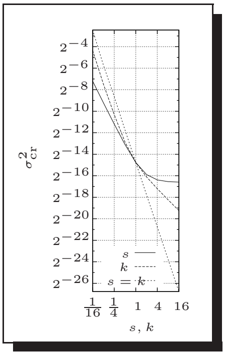


Figure 4.19: *Variance of the crossratio as a function of scale s (solid line), accuracy k (dashed line), and resolution $s = k$ (dotted line). Shown is the median predicted variance when using the fast method described in Section 4.5.2.2 (line fit weighted by σ^2_α). It is interesting to note that the variance changes cubic with the resolution (compare Equation (4.21)) and for low accuracies, quadratic for low scales, linear for high accuracies, and is constant for high scales.*

approximately linear dependency, which again is mirrored in (4.21) and (4.23). Finally, when varying the scale we see an approximately quadratic relationship at low scales, which becomes constant for higher scales — I'm not sure how to interpret this.

Collectively all these results show that for a reasonable choice of line $\boldsymbol{\ell}_X$ — based on error-propagation principles — it is possible to rival the best algorithms for the computation of the crossratio in accuracy at a fraction of their runtime.

4.6 Comparing Stochastic Entities

In comparing stochastic entities we are generally trying to answer the question whether two observations $\mathbf{x}_1, \mathbf{x}_2 \in I\!R^n$ with known covariance matrices $\boldsymbol{\Sigma}_{\mathbf{x}_1}, \boldsymbol{\Sigma}_{\mathbf{x}_2} \in I\!R^{n\times n}$ could both be valid observations of the unknown true value $\boldsymbol{\mu}_{\mathbf{x}} \in I\!R^n$. We assume that the observations $\mathbf{x}_i \in I\!R^n$ are Gaussian distributed with pdf

$$p_{\mathbf{x}_i}(\mathbf{x}_i) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}_{\mathbf{x}_i}|}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathbf{x}_i}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}})\right), \tag{4.55}$$

where $|\boldsymbol{\Sigma}_{\mathbf{x}_i}|$ is the determinant of $\boldsymbol{\Sigma}_{\mathbf{x}_i}$.

Assuming that the two observations $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent of each other we can simply multiply their probabilities; the joint probability that both observations are valid observations of the same feature is then proportional to the sum

$$R = (\mathbf{x}_1 - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathbf{x}_1}^{-1} (\mathbf{x}_1 - \boldsymbol{\mu}_{\mathbf{x}}) + (\mathbf{x}_2 - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathbf{x}_2}^{-1} (\mathbf{x}_2 - \boldsymbol{\mu}_{\mathbf{x}}). \tag{4.56}$$

The residuum R in (4.56) above is a function of the unknown original feature vector $\boldsymbol{\mu}_{\mathbf{x}}$; it is straightforward to show that it is minimised by the feature

$$\boldsymbol{\mu}_{\mathbf{x}} = \left(\boldsymbol{\Sigma}_{\mathbf{x}_1}^{-1} + \boldsymbol{\Sigma}_{\mathbf{x}_2}^{-1}\right)^{-1} \left(\boldsymbol{\Sigma}_{\mathbf{x}_1}^{-1}\mathbf{x}_1 + \boldsymbol{\Sigma}_{\mathbf{x}_2}^{-1}\mathbf{x}_2\right) \tag{4.57}$$

which has a (calculated) accuracy (covariance) of

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Sigma}_{\mathbf{x}_1}^{-1} + \boldsymbol{\Sigma}_{\mathbf{x}_2}^{-1}\right)^{-1} . \tag{4.58}$$

Once the residuum R has been found we can then use a simple χ^2-test to test the hypothesis that the two measurements $\mathbf{x}_1$ and $\mathbf{x}_2$ are observations of the same entity $\boldsymbol{\mu}_{\mathbf{x}}$:

$$R \overset{!}{<} \chi^2_{p,2} . \tag{4.59}$$

This basic approach is valid for all the examples given below. If the uncertainty of the model is explicitly given (by a covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\mu}_{\mathbf{x}}}$) then it is straightforward to add this matrix to all occurrences of $\boldsymbol{\Sigma}_{\mathbf{x}_i}$, it is always $\boldsymbol{\Sigma} > \boldsymbol{\Sigma}_{\boldsymbol{\mu}_{\mathbf{x}}}$ in (4.58).

In cases where the intrinsic dimensionality of the problem m is smaller than its algebraic dimension n — this is always the case when dealing with homogeneous coordinates, and in particular projectively transformed data — we would need to replace the inverse $\boldsymbol{\Sigma}_{\mathbf{x}_i}^{-1}$ by the pseudoinverse $(\boldsymbol{\Sigma}_{\mathbf{x}_i})^{-}_{m}$, and if necessary replace the determinant in (4.55) by the product of its m nonzero eigenvalues. Usually we will also need to normalise the measurements or otherwise make them comparable, examples of this are given below. This is also one of the reasons why the otherwise very elegant approach presented by Irani and Anandan in [64] can not be generalised beyond affine transformations. Their approach is based on transforming the raw-data into a covariance-weighted data space, where the components of noise are uncorrelated and identically distributed; however, so far the proof that the optimal solution in this space will also be optimal in the original space is missing. Still, the reader who is only dealing with affinely transformed data is urged to have a look at their work.

4.6.1 Edgels

Edgels are the output of an edge-finder like Canny [24]. No scenario exists where it would be required to compare two edgels. The simplest comparison would already be between derived features like lines or points as described in the next section.

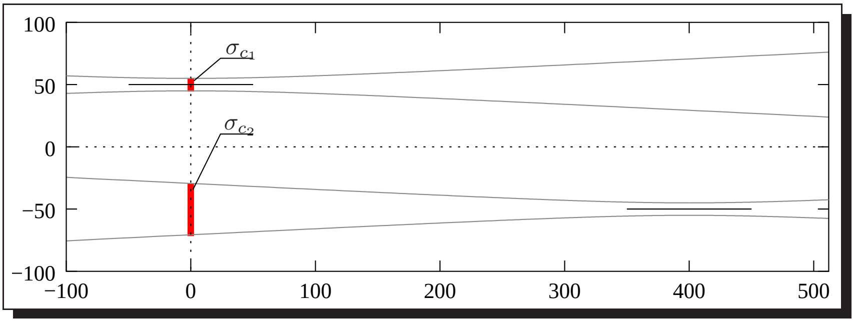


Figure 4.20: *The uncertainty in the distance from the origin c depends on the location of the line segment.*

4.6.2 Lines

When comparing “lines” it is important to realise that in images we are never really dealing with (infinite) lines, but always with line segments only. When comparing “lines” we are therefore in reality trying to answer the question whether two line segments $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ could be considered independent observations of the same line $\boldsymbol{\ell}$. This is the same as answering the question whether two line segments could be considered collinear, a question posed in Section 6.

As mentioned in Section 4.3 there exists a multitude of different line-parameterisations. Particularly prevalent is the normal form $(a, b, c)^\mathsf{T}$ with $a^2 + b^2 = 1$; this is however not well suited to comparisons as the uncertainty in c can be considerable, depending on the position of the line segment, compare Figure 4.20. This effect is greatly mitigated by the use of spherically normalised parameters $a^2 + b^2 + c^2 = 1$ and in particular Kanatani's N-vectors; I would however like to advocate the use of a new $(\alpha, \bar{x}, \bar{y})^\mathsf{T}$ parameterisation, which I believe to be the most convenient for this particular application. We are then dealing with the two measurements $\boldsymbol{\ell}_1 = (\alpha_1, x_1, y_1)^\mathsf{T}$ and $\boldsymbol{\ell}_2 = (\alpha_2, x_2, y_2)^\mathsf{T}$ with covariances $\boldsymbol{\Sigma}_{\boldsymbol{\ell}_1}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\ell}_2}$. Similar to Section 4.6 we then have to find the ideal line $\boldsymbol{\ell} = (\alpha, x, y)^\mathsf{T}$ with minimum weighted distance to the two lines.

It is however clear that the point $(x, y)^\mathsf{T}$ can be chosen arbitrarily along the lines, a direct comparison with $(x_1, y_1)^\mathsf{T}$ and $(x_2, y_2)^\mathsf{T}$ would not be meaningful. Instead we use the distance between the points $(x_i, y_i)^\mathsf{T}$ and their projection onto the ideal line, i. e. $(x_i - d_i \sin(\alpha), y_i + d_i \cos(\alpha))^\mathsf{T}$ with $d_i = x_i \sin(\alpha) - y_i \cos(\alpha) - x \sin(\alpha) +$

$y\cos(\alpha)$, which is the distance from the point $(x_i, y_i)^{\mathsf{T}}$ to the ideal line[11]. We therefore evaluate

$$\min_{\alpha,x,y} \quad \left[(\alpha_1 \ominus \alpha, -d_1\sin(\alpha), d_1\cos(\alpha))\, \boldsymbol{\Sigma}_{\boldsymbol{\ell}_1} \begin{pmatrix} \alpha_1 \ominus \alpha \\ -d_1\sin(\alpha) \\ d_1\cos(\alpha) \end{pmatrix} \right. \\ \left. + (\alpha_2 \ominus \alpha, -d_2\sin(\alpha), d_2\cos(\alpha))\, \boldsymbol{\Sigma}_{\boldsymbol{\ell}_2} \begin{pmatrix} \alpha_2 \ominus \alpha \\ -d_2\sin(\alpha) \\ d_2\cos(\alpha) \end{pmatrix} \right] \le \chi^2_{p,2} \quad (4.60)$$

in order to decide whether the two line segments should be considered collinear. The symbol $\ominus$ denotes a cyclic subtraction such that $-\pi \le \alpha \ominus \beta < \pi$. We find that the line $\boldsymbol{\ell}$ which minimises Equation (4.60) is given by

$$\boldsymbol{\ell} = \left(\boldsymbol{\Sigma}_{\boldsymbol{\ell}_1}^{-1} + \boldsymbol{\Sigma}_{\boldsymbol{\ell}_2}^{-1}\right)^{-1} \left(\boldsymbol{\Sigma}_{\boldsymbol{\ell}_1}^{-1}\boldsymbol{\ell}_1 + \boldsymbol{\Sigma}_{\boldsymbol{\ell}_2}^{-1}\boldsymbol{\ell}_2\right) \quad (4.61)$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\ell}} = \left(\boldsymbol{\Sigma}_{\boldsymbol{\ell}_1}^{-1} + \boldsymbol{\Sigma}_{\boldsymbol{\ell}_2}^{-1}\right)^{-1} \quad (4.62)$$

and the minimum in Equation (4.60) is given by

$$R = (\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2)^{\mathsf{T}} \mathbf{J}^{\mathsf{T}} \boldsymbol{\Sigma}_{\boldsymbol{\ell}_1}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{\ell}} \boldsymbol{\Sigma}_{\boldsymbol{\ell}_2}^{-1} \mathbf{J} (\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2) \quad (4.63)$$

with the Jacobian

$$\mathbf{J} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sin^2(\alpha) & -\sin(\alpha)\cos(\alpha) \\ 0 & -\sin(\alpha)\cos(\alpha) & \cos^2(\alpha) \end{pmatrix}. \quad (4.64)$$

Equations (4.61) and (4.62) can also be interpreted as the combination of two line segments $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_1$ into a new line segment $\boldsymbol{\ell}_3 = \boldsymbol{\ell}$ with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\ell}_3} = \boldsymbol{\Sigma}_{\boldsymbol{\ell}}$, and it is possible to substitute the two original line segments by the new segment formed this way in all future comparisons.

4.6.3 Points

In contrast to edgels, which are measurements directly on the image plane, points are features calculated directly or indirectly from edgels, e. g. as the intersection of two or more lines as described in Section 4.4 and used in Section 6. Points can be parameterised using any of the approaches described for edgels in Section 4.2.

[11] Of course I could just as well have defined a measure which is directly based on this distance, but the advantage of my approach is that the coordinates of the ideal line naturally come out in $(\alpha, x, y)^{\mathsf{T}}$ format too.

Common are an Euclidean parameterisation $(x, y)^\mathsf{T}$, to which (4.57)–(4.59) are directly applicable, and a parameterisation in homogeneous coordinates, $k(x, y, w)^\mathsf{T}$. The latter parameterisation does not usually allow a direct application of (4.57)–(4.59), as the difference of two measurements can be nonzero even if the two measurements are exactly identical, it is in general

$$k_1\mathbf{x} = k_2\mathbf{x} \tag{4.65}$$

$$k_1\mathbf{x} - k_2\mathbf{x} \neq \mathbf{0} \tag{4.66}$$

In order for (4.57)–(4.59) to be comparable we first need to normalise the measurements to assure the equal-sign in (4.66). A normalisation $w = 1$ is essentially an Euclidean parameterisation and shares with it the problem that points at infinity can not be represented; also very common is a spherical normalisation which enforces $\mathbf{x}^\mathsf{T}\mathbf{x} = 1$. This is essentially the Gaussian sphere or N-vector representation described in Section 2.9.2. Not only does this normalisation enforce the equal-sign in (4.66), but the covariance-region on a sphere is indeed a much more adequate model for points as intersections of lines than is the covariance-region on the plane (as we have seen in Section 4.4.3 — and will again see in Section 7). After normalisation, (4.57)–(4.59) can be applied if the pseudoinverse is used instead of the inverse.

4.6.4 Crossratios

The comparison of cross-ratios is straightforward; one can simply use (4.56)– (4.59) with $n = 1$ so that the covariance matrices become simple variances. We can then calculate the most likely true crossratio as

$$\overline{\mathrm{cr}} = \frac{\mathrm{cr}_1\sigma^2_{\mathrm{cr}_2} + \mathrm{cr}_2\sigma^2_{\mathrm{cr}_1}}{\sigma^2_{\mathrm{cr}_1} + \sigma^2_{\mathrm{cr}_2}} \tag{4.67}$$

which it was possible to calculate with accuracy (variance)

$$\sigma^2 = \frac{\sigma^2_{\mathrm{cr}_1}\sigma^2_{\mathrm{cr}_2}}{\sigma^2_{\mathrm{cr}_1} + \sigma^2_{\mathrm{cr}_2}}. \tag{4.68}$$

The hypothesis that the two measurements cr_1 and cr_2 are observations of the same entity $\overline{\mathrm{cr}}$ can then be tested using:

$$R = \frac{(\mathrm{cr}_1 - \mathrm{cr}_2)^2}{\sigma^2_{\mathrm{cr}_1} + \sigma^2_{\mathrm{cr}_2}} \overset{!}{<} \chi^2_{p,1}\,. \tag{4.69}$$

Chapter 5

Detecting Repeated Parallel Structure

. . . Stripe for Stripe.

The Bible: Hebrew Exodus 21:23

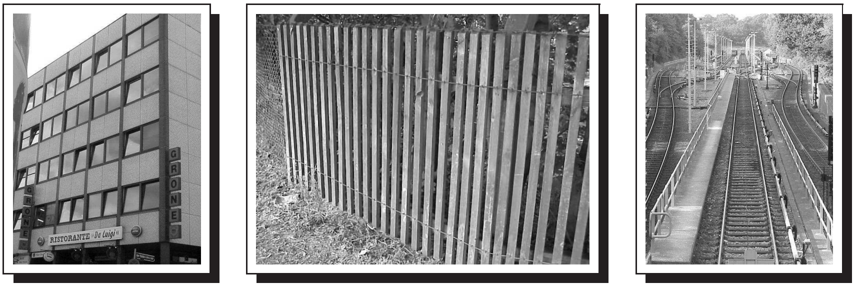


Figure 5.1: *Examples of repeated parallel structure in images.*

5.1 Introduction

This chapter, as well as Chapters 6 and 7, demonstrates the application of the theories discussed in Chapter 4 to a real-world example.

The algorithm described here deals with the detection of repetitive structures consisting of parallel line segments with a given crossratio. Some of the more obvious examples for this are railway-sleepers, fences, and windows (particularly in big office-buildings). Figure 5.1 shows some examples. The structure used throughout most of this chapter is that of a pedestrian or zebra crossing . This was originally implemented as part of the project MOVIS[1] — Mobile Optoelectronic Visual Interpretative System for the Blind and Visually Impaired — which took place from 1995 to 1997 [1, 6]. Within this project, a first prototype of a portable device for blind and visually impaired persons was created which was able to recognise a small number of useful objects and signs customarily found in street scenes. This prototype consisted of a spectacle-like device connected to an (at the time stationary) computer doing the image processing. Figure 5.17 on Page 143 shows images of the actual device used. The theory is, however, independent of this particular application and equally applicable to any other repeated structure of parallel line segments.

Detecting zebra crossings may sound like an easy task. After all, they're big, and they're designed to be fairly obvious. However, it isn't. Reasons include:

- The general amount of occlusion connected with street scenes, namely fellow pedestrians who get in the way, lamp and sign posts, cars, and basically everything that moves. Moreover, zebra crossings are particularly prone to

[1]MOVIS was funded by the BMBF, the German Ministry for Education and Research.

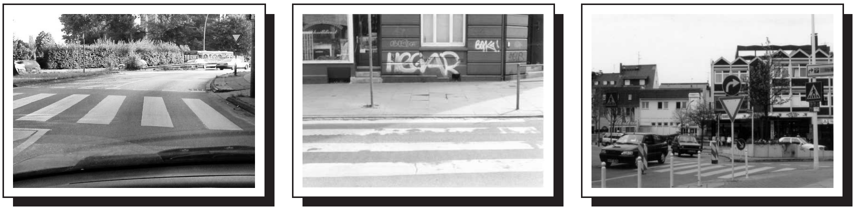


Figure 5.2: Different views of a zebra crossing as seen from a car (left) and a pedestrian (middle and right).

occlusion, since they are designed for people to walk on and for cars to drive across.

- Zebra crossings are often in bad repair, patches are missing, they have spots or holes.
- Due to varying viewing geometries the width of stripes in an image may vary from dozens of pixels to only 2 to 4 pixels — even within one stripe.

The recognition of repeated parallel structures under perspectivity has traditionally been dealt with in the context of texture analyses. In [84] an algorithm for the recognition of arbitrary repeated structures is presented. However, this approach requires a minimum amount of texture within each element of the structure and is therefore unsuitable or at best problematic for structures with little or no texture as they are presented here. Only after the work described here was first published [6] did a small number of papers appear based on this work [134, 135, 137].

The work specific to zebra crossings on the other hand has nearly exclusively assumed an autonomous vehicle's (car's) point of view [85, 108]. This way, the camera's orientation relative to the ground can be assumed known. Also, the street's left and right boundaries are generally well known, and these can be used to identify the road's (virtual) vanishing point [90], through which all lines bounding a zebra crossing have to pass [108]. Finally, from the viewpoint of a car a zebra crossing is always encountered head on, which means that all stripes will have approximately the same width on any row of the image, and that the zebra crossing will at most be occluded by objects directly on the road. Figure 5.2 (left) shows an example of a zebra crossing as seen from a car.

None of the constraints mentioned above apply when dealing with a camera carried by a pedestrian, as within MOVIS. Here, the camera's orientation relative to the ground is at best only approximately known (e. g. from motion sensors affixed to the camera), and no other constraints exist. Also, the zebra crossing will often be heavily occluded, fracturing the individual stripes into several "stubs". Figure 5.2 (middle, right) shows two examples of a zebra crossing as seen from a pedestrian's point of view. This means that it is generally necessary to group several separate patches into one zebra crossing. It is my experience that this is best achieved using a line-based approach, which will be described in Section 5.3. Error propagation plays a particularly important role here since a zebra-crossing's size and quality in the image can vary considerably from image to image — so much in fact that a first prototype based on static thresholds never worked on more than at most two images at the same time, while the approach presented here has proven it's stability on literally thousands of images. Work on the recognition of pedestrian crossings from the viewpoint of a pedestrian only appeared after the original publication of this work in [6], and building on it [135].

The remainder of this chapter is organised as follows: Section 5.2 describes the underlying model used to group and recognise repeated parallel structures. The example of a zebra crossing used is easily modified for other structures, possibly simply replacing "horizontal" with "vertical" where appropriate. Section 5.3 describes the actual process of grouping and recognition based on the theory and principles discussed in Chapter 4. This makes use of my new formulation for the calculation of the cross-ratio described in Section 4.5.2. In addition I present a new method for the transformation of lines into an only partly specified canonical frame, i. e. one where only some structural information is given, in Section 5.3.3. To my knowledge this was also the first application where the horizon was calculated from image structure alone (now a staple of projective geometry). A heuristic, but in my experience rather efficient method for merging hypotheses in the presence of unquantified errors in the object's geometry is given in Section 5.3.4. Although all sections discuss the relative merits of different camera models, I have found that it is the assumption of a quasi-calibrated ("sensible") camera which allowed me to implement an algorithm that is both fast and robust. Based on this camera model, Section 5.4 describes a simple but at the time of implementation new approach used for verification, which stands in the tradition of [97] and could well be seen as the forerunner of algorithms such as [33, 34, 87]. Finally, Section 5.5 presents some examples of successfully recognised zebra crossings and discusses the results.

5.2 Model

I will first discuss the underlying 3D-model in Section 5.2.1, followed by a discussion of the different camera models in Sections 5.2.2 ff.

5.2.1 3D Model

Zebra crossings can be found throughout Western Europe as well as in many other parts of the world. In most countries, the following would be considered a reasonable 3D-model of a zebra crossing:

1. Zebra crossings are planar.
2. They are located on the ground-plane.
3. They consist of light stripes on a darker surface.
4. All stripes are parallel.
5. All stripes are of equal width.
6. All gaps between stripes are of equal width[2].
7. The ratio of the width of the stripes to the width of the gaps is known in advance.

The main problem with the above description is that it is not, strictly speaking, true. Streets are not entirely plane, but will generally slope to the sides to enable rainwater to drain. Neither are the stripes entirely plane themselves (but will generally extend somewhat above the street-surface) nor are they exactly parallel nor exactly the same width or distance from each other[3]. Experiments show, however, that these deviations are small for most images compared with resolution related artifacts, and are well accounted for by simple error models and a stepwise refinement approach used for grouping (described in Section 5.3) that requires only three consecutive stripes at a time to conform to the above model.

The following sections describe how this 3D-model will be projected onto a 2D-plane under the assumption of different camera models.

5.2.2 Projective Camera Model

This model was discussed in Section 2.3.4. It is the most general linear camera model available, and can be parameterised by a general 8 DOF projective transformation[4]. Using this model, only very little can be said about a zebra-crossing's appearance after projection:

[2]They are usually also approximately the same width as the stripes, but this is not a sine qua non, but see also Item 7.

[3]Although German standards[23], for example, only allow for deviations in width or position of a maximum of ± 10 mm or 2 % on new zebra crossings.

[4]Since we are only dealing with a transformation from one plane onto another, a homography.

- All lines bounding the stripes are coincident (this follows from Items 1 and 4).
- The crossratio is known in advance (from Items 1, 5, 6, and 7).
- The lines bounding the stripes have alternating directions, i. e. cutting across all lines we see a change in luminance from darker to lighter to darker[5] (and so forth — this follows from 3).

That the crossratio is known means in particular that from two stripes (four line segments[6]) it is possible to predict the location of additional stripes within the image, as will be done in Section 5.3.3.

5.2.3 Constrained Perspective Camera Model

This model is based on the model of a perspective camera described in Section 2.3.3, but with the added constraint that the horizontal and vertical direction are approximately known within the image.

Assuming a level zebra crossing (i. e. the street does not go uphill), the stripes' vanishing line coincides directly with the horizon. Although the location of the horizon is of course arbitrary under a general projective transformations, it will none the less be close to horizontal in virtually all images taken by human operators. It is therefore possible to formulate some additional constraints on the appearance of the zebra crossing within the image, in particular:

- The vanishing line is located completely above the zebra crossing (from 2).
- The vanishing line is approximately horizontal (see Items 1 and 2).
- It is approximately within the image.

If, on the other hand, the image was taken by a robot, UGV or similar, additional knowledge about the image's orientation will often be available (as is, at least to some extent, the case within MOVIS). The above conditions would then reduce to

- The vanishing line approximately coincides with the (known) location of the horizon.

Figure 5.3 shows an example of a drawing where this condition is violated; the drawing does not represent a valid representation of a zebra crossing.

[5]For many images both the street's surface as well as the zebra crossing's have luminance values lighter than the image's mean luminance.

[6]Strictly speaking three line segments would be sufficient.

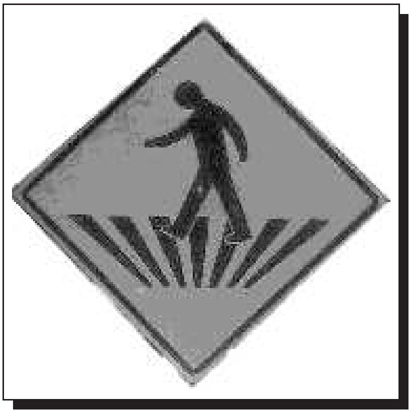


Figure 5.3: *Example of an impossible Geometry. The vanishing point in this street-sign from Uruguay is* below *the zebra crossing, corresponding to a scene viewed from* below *ground, which does not agree with the pedestrian's position. Clearly this would not be recognised as a valid representation of a zebra crossing under the assumption of a constrained perspective or quasi-calibrated camera model.*

5.2.4 Quasi-Calibrated Camera Model

This model, which is based on the constrained perspective model described above, further assumes that some approximate information about the imaging process is available, such as the camera's focal length and aspect ratio, and the camera's approximate height over ground. I will show in Section 5.4 that in conjunction with Item 2 this information can be used for verification purposes.

5.3 Grouping

The following describes a bottom-up approach for the grouping and recognition of partly occluded zebra crossings in natural images. The approach is completely line-based and assumes that suitable line segments and their covariance matrices have already been found using a sub-pixel accurate edge-finder [24] in connection with the approach for line fitting described in Section 4.3.

Starting with the individual line segments, sets of four line segments are identified (using perceptual grouping, two possible approaches are described in Section 5.3.1) and tested for coincidence and crossratio (Section 5.3.2). These line segments are then backprojected into the images, and additional stripes are identified (Section 5.3.3), creating several hypotheses (see also Figure 5.4). Finally, overlapping hypotheses are merged into a single hypothesis (Section 5.3.4).

Most of this approach is directly based on the ideas and principles described in

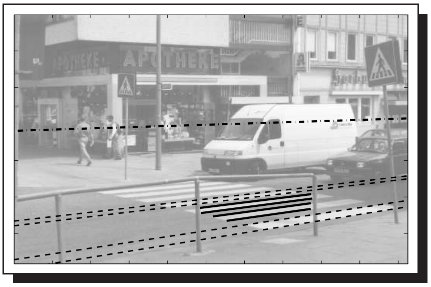


Figure 5.4: *Grouping zebra crossings. Starting with a set of 4 coincident line segments (continuous lines), a vanishing line (dot-dashed line) is calculated and additional lines corresponding to adjacent stripes are hypothesised (dashed lines).*

Chapter 4, in which case only a short reference to the corresponding section is given. Only where additional strategies were used is this explained in detail in the text below.

5.3.1 Sets of 4 Lines

To take advantage of the constraint of a fixed crossratio — the only constraint on the appearance of a zebra crossing after general projection — one first has to identify 4 coincident lines. A very simple approach would be to directly test all possible groups of 4 line segments for coincidence and crossratio. This has the advantage that no additional knowledge is needed to identify possible sets of lines, and is easily implemented. It will, however, lead to an algorithm of complexity $\mathcal{O}(N^4)$, where N is the number of line segments in the image. For cluttered images of real street scenes, which can easily contain several hundred line segments, this may lead to execution times of several hours or more even on today's computers.

It is therefore advisable to employ a scheme for the identification of sets of lines which makes use of structural information within the image. This kind of approach is called perceptual grouping and enjoyed growing popularity throughout the computer-vision community during the 90s, in particular where the evaluation of aerial images is concerned. We differentiate between a top-down approach, whereby a bigger set of lines is reduced to four line segments, and a bottom-up approach, which starts with the individual line segments. Both are described in the following.

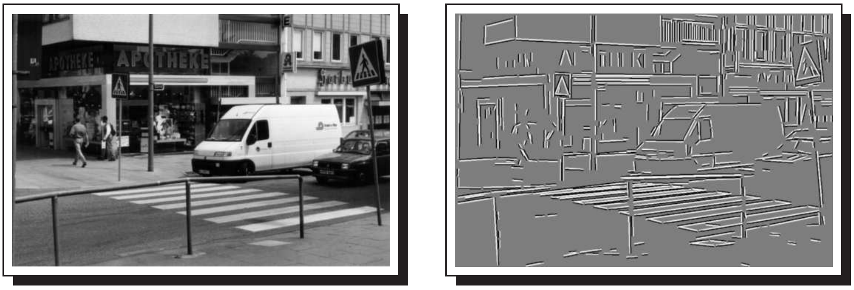


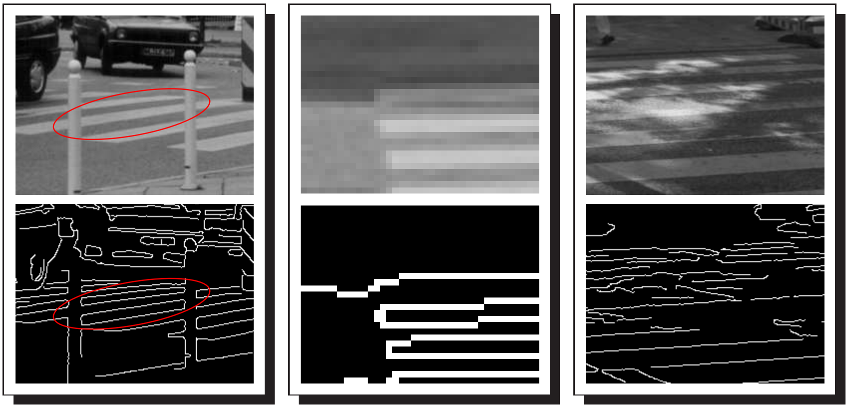
Figure 5.5: *A grey level image and the line segments fitted to grey-level discontinuities. Each line segment is displayed as a black and white double line, the black side corresponding to the darker side of the discontinuity, and the white side to the lighter one.*

5.3.1.1 Top-Down Approach

Finding sets of coincident line segments is equivalent to vanishing-point detection, i. e. identifying the common intersection of a set of lines as well as the corresponding set of lines. Once the vanishing point has been identified it is then possible to parameterise these lines solely by their angle of orientation $-\pi/2 \leq \alpha_o < \pi/2$ or direction $-\pi \leq \alpha_d < \pi$ (refer to Section 3.5.1 for the definition of *orientation* and *direction*). Finding all possible sets of lines that could form a zebra crossing is then equivalent to finding all paths from any line to at least 3 other lines such that the lines' orientations increase monotonically, while their directions alternate. Figure 5.5 shows an actual image and all line segments, displayed as black and white double lines, the black side corresponding to the darker side of the discontinuity, and the white side to the lighter one.

The success of this approach does entirely rely on the output of the algorithm used to identify the vanishing point and corresponding set of line segments. It has, however, been pointed out in the literature [149] that this is not a particularly reliable process if only a small percentage of the overall number of line segments in the image actually converges to this particular vanishing point[7], and especially in the presence of clutter. For this reason a bottom-up approach was used within the project MOVIS, which will be described in the following section.

[7]Vanishing point detection *is* used in Section 6, where the aim is to identify the main directions within an image rather than to identify a possible small (sub-) set of lines.



(a) Stripe occluded by a light object.

(b) Stripe is to small to fit a line to the narrow side.

(c) Stripe in bad repair (and badly illuminated), the connection between the two sides is lost.

Figure 5.6: *Some typical problems when recognising pedestrian crossings.*

5.3.1.2 Bottom-Up Approach

This approach tries to group two line segments into a stripe based on structural information within the image, or additional constraints known about the imaging process. It then proceeds to group two stripes into a set of four line segments, which can subsequently be tested for coincidence and crossratio (compare Section 5.3.2). Several such approaches are conceivable, and in the following a few of them are presented, together with their relative merits. Only the last one has been found suitable within MOVIS, but several might be useful when dealing with structures other than zebra crossings..

One approach is to identify all quadrangles that are lighter on the inside than on the outside. Although this approach is the only one of the perceptual grouping algorithms presented here that could work with arbitrary projections, it does have some serious downsides. The main problem is occlusion. Since these can be of an arbitrary shape, they can easily lead to nonlinear boundaries of a stripe. They might even be of a lighter colour than the stripe, in which case any algorithm looking for a light quadrangle on dark background will fail, as can be seen in Figure 5.6(a). Finally, the two line segments corresponding to the long sides of a stripe might only be separated by a few (e. g. 3) pixels, in which case no lines could conceivably be fitted to the two smaller sides, making the search for a quadrilateral rather pointless, see Figure 5.6(b) — note that the edgels, although drawn at pixel-position, have in fact been calculated with sub-pixel accuracy.

A second approach could be trying to identify a ⊔-like structure instead, where two longer line segments are joined by one smaller one. This approach would only work reasonably with at least the constrained perspective model, which would allow us to constrain the possible angle between the two longer line segments. It would then be able to cope with occlusions by nonlinear or lightly coloured objects on at most one side of the stripe, but would otherwise share all the disadvantages discussed above. In particular this approach too would not work for stripes which are too narrow to fit a line to one of the short sides, Figure 5.6(b) again shows an example.

Dropping the constraint that the two longer line segments should be connected by a shorter line segment, we reach a model where it would be sufficient that the two line segments are connected by any kind of edge. This model has actually proven quite reasonable and will only fail in the case of badly preserved zebra crossings or in cases where both ends of a stripe are occluded by an object similar in luminance to the stripe. Badly preserved zebra crossings will often contain holes or spots and might not allow fitting any consecutive edge from one side of the stripe to the other, Figure 5.6(c) shows an example. It is only to accommodate these kinds of zebra crossings that instead the approach below was used within MOVIS.

This approach does not rely in any way on connectivity between the two sides of a zebra crossing. While this initially results in many more false positives, it also avoids some of the false negatives which would otherwise be inevitable. The approach is based on a constrained perspective model. This allows us to limit the possible directions under which a zebra crossing can be seen, and as a consequence limit how it would appear in the image. In particular, it is now possible to calculate a maximum angle between the two line segments bounding a stripe, say $\Delta\alpha_o < 30°$. For each line, only lines are considered as a match which

1. are entirely on the first line's lighter side,
2. face that line (corresponding to a transition from dark to light to dark),
3. have a difference in orientation of at most $\Delta\alpha_o$,
4. *overlap* each other to at least $p\,\%$.

Where overlap, in this context, is defined as follows (compare Figure 5.7):
Project one line segment onto the other. The line segments are said to overlap to p percent if the shorter line segment shares $p\,\%$ of its length with the longer one. The overlap between the two line segments is the maximum overlap of projecting

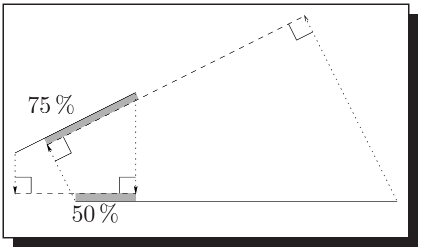


Figure 5.7: *Overlap between two line segments.*

each line segment onto the other.

Finally, stripes are grouped according to rules similar to the ones used above for grouping line segments, but without taking the overlap into consideration, forming the required sets of four line segments (two stripes).

5.3.2 Crossratio

Once four line segments have been identified, the first and only hard test of whether the line segments might actually be part of a zebra crossing (or any other repeated structure of given geometry) can be performed by calculating their crossratio and comparing it to the original structure's crossratio using Equation (4.69) from Section 4.6.4. This is straightforward if the four line segments were found by a top-down approach. In this case it is already known that the four lines share a common intersection, as well as the intersection's coordinates, and the crossratio can efficiently be calculated using any of the methods described in Section 4.5.1.

Things are slightly more complicated if the four lines were found by a bottom-up approach, as I will assume was done here. It is, in this case, not yet known whether the four lines will indeed share a common intersection, nor where this intersection could be found. Calculating the intersection using any of the methods described in Section 4.4 is, however, expensive, especially since the overwhelming majority of line-sets will not belong to any interesting structure, so that this computation would ultimately be in vain.

It is therefore advisable to use a two-stage approach instead, as described in Section 4.5.2, whereby in a first stage the fast algorithm described there is used to calculate the crossratio. Only if this initial result passes the χ^2-test in Equation (4.69) — possibly using a low value for p — is the lines' intersection calculated

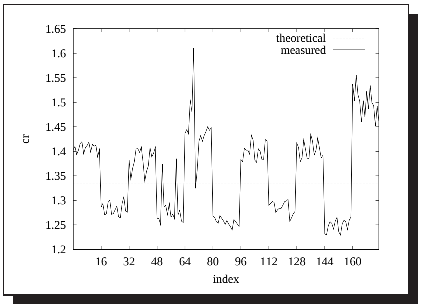


Figure 5.8: *Different crossratios for white stripes (mostly* cr $> 4/3$*) as opposed to black ones (mostly* cr $< 4/3$*). The effect is due to a difference in height between the white stripes and the surface of the road.*

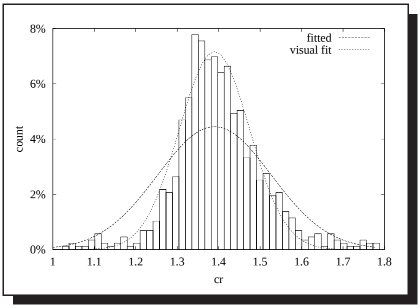


Figure 5.9: *Distribution of 900 crossratios (white stripes only) calculated from 10 images of real zebra crossings (median* cr $= 1.39$, $s_{\mathrm{cr}} = 0.136$, *visual* $\sigma_{\mathrm{cr}} \approx 0.085$*).*

and a second χ^2-test used to evaluate whether the lines are actually coincident. Only if this test too is passed successfully will a more accurate algorithm be used to recalculate the crossratio and once more apply the χ^2-test in Equation (4.69). Only very few sets of line segments will remain after these three tests.

When used on images of real-world zebra crossings, an interesting effect can be observed comparing the crossratio of two of the “white” stripes with the crossratio of two of the “black” gaps between the stripes. These both have the same width and should therefore result in a uniform crossratio of 4/3. In practice, however, this is not the case. Figure 5.8 shows this for the zebra crossing depicted in Figures 5.4, 5.5 and 5.15, second row, right. For this zebra crossing, each line bounding a stripe is naturally divided into two line segments by an occluding object. Calculating all possible crossratios of 4 consecutive line segments therefore results in a sequence of 16 crossratios for the first two stripes, 16 crossratios for the first two gaps, 16 crossratios for stripe two and three and so forth. It can be seen that instead of a common crossratio of cr $= 4/3$ we get crossratios around cr ≈ 1.4 (and growing) and cr ≈ 1.28 (and falling) respectively. For an observer at a distance of approximately 15 m and a height of approximately 1.8 m this is consistent with stripes that extend approximately 4 mm above the surface of the street (compare [23]).

It is not possible to account for this effect geometrically, since the effect can only be

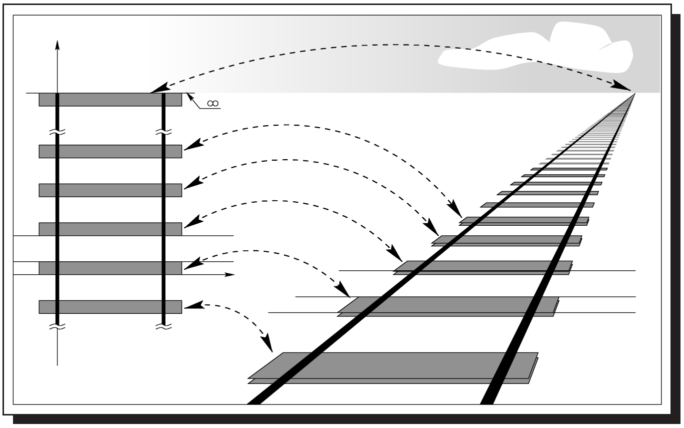


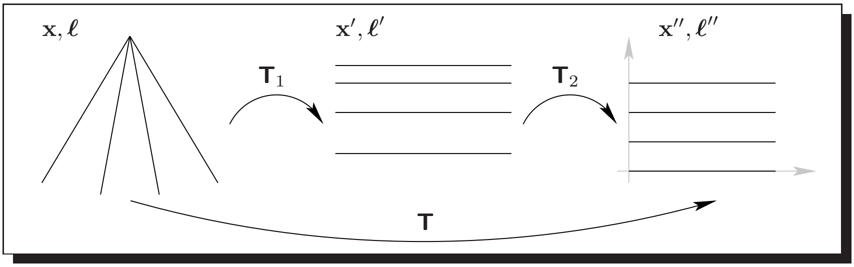
Figure 5.10: *Finding additional line segments by backprojection.*

corrected if a calibrated camera is used, and since the height of the stripes above the surface of the street can vary considerably (the stripe can even be slightly below the surface) and will generally be unknown. The effect has therefore to be accounted for by some other means, and the factor $\sigma^2_{\mathrm{cr}_2}$ in Equation (4.69) presents one possible approach, although the use of this term implies that the expected values for the crossratio will be Gaussian distributed, which is of course not the case. Figure 5.9, which shows a histogram of the distribution of 900 different crossratios calculated from 10 images of real zebra crossings as well as two fitted (both numerically and visually) Gaussian distribution, does however show that the actual distribution is sufficiently “Gauss-like” to expect reasonable results, and this is born out by the results described in Section 5.5.

5.3.3 Additional Lines

The condition on the crossratio used in the above section is only a necessary condition to identify the structure we are looking for. Usually a number of additional line-sets with similar crossratios exist in any given image. In MOVIS, I therefore decided that finding two stripes (four lines) with given crossratio is not sufficient evidence for a zebra crossing (the same argument could be made for any other repeating structure). Instead, a minimum of three stripes (six lines) is required.

Luckily it is relatively easy to identify additional line segments by using an adap-



$\mathbf{x}, \boldsymbol{\ell}$
$\mathbf{x}', \boldsymbol{\ell}'$
$\mathbf{x}'', \boldsymbol{\ell}''$
T_1
T_2
T

Figure 5.11: *Decomposition of* T *into two transformations* $\mathsf{T} = \mathsf{T}_2\mathsf{T}_1$*, using an intermediate canonical frame representation.*

tion of the canonical-frame approach described in Section 4.5.1.2. Within the canonical frame, the locations of all other lines potentially belonging to the structure in question are known. These can then be backprojected into the image to get the approximate position of additional stripes in the image. If corresponding stripes are found, these are then added to the set of four lines to form a hypothesis. In addition, this also means that the location of the stripes' vanishing line is known (the backprojection of a line at infinity), in Section 5.4 this will be used for verification — the backprojected vanishing line should coincide with the horizon of the image. Figure 5.10 shows an example where the position of a minimum of three lines within the image is sufficient to predict the position of an infinite number of additional lines. In the following an alternative approach to the ones described in Section 4.5.1.2 is given.

The most accurate way to achieve the backprojection is to find a (5 degrees of freedom) transformation T^T from a canonical frame (of, say, horizontal lines of known position) into the image that minimises the distance between the proposed and the measured lines. Once this transformation is found, it is then easy to predict other lines by calculating $\mathsf{T}^\mathsf{T}\boldsymbol{\ell}''_i$, where $\boldsymbol{\ell}''_i = (0, 1, c''_i)^\mathsf{T}$ is one of the lines in the canonical frame (see Figure 5.11). By the same idea, the vanishing line can be found by calculating $\mathsf{T}^\mathsf{T}(0, 0, 1)^\mathsf{T}$. As for many of the problems which we encountered in Section 4 there is again no closed-form solution to the problem of finding T^T.

A somewhat similar but much faster approach finds the inverse transformation $\mathsf{T}^{-\mathsf{T}}$ such that the distance between the proposed and measured lines becomes minimal within the canonical frame (instead of the image). A very efficient approximation for this transformation exists under the assumption of small errors. It is then possible to decompose T into two matrices T_1 and T_2 for which we can solve

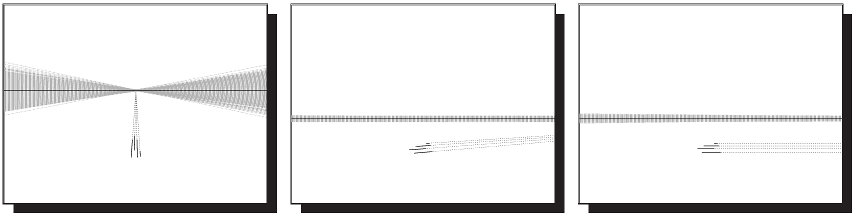


Figure 5.12: *Monte-Carlo simulation of vanishing-line calculation using the canonical-frame algorithm for three typical constellations. Notice that small errors in the vanishing-*point *coincide with big errors in the orientation of the vanishing-*line *and vice-versa; compare also Figure 4.9, where the same lines were used to calculate the vanishing-*point.

separately. $\mathsf{T}_1^{-\mathsf{T}}$ transforms the lines into an intermediate canonical frame in which all lines are (as near as possible) horizontal. $\mathsf{T}_2^{-\mathsf{T}}$ is the transformation into the final canonical frame, in which the individual lines will end up in definite positions (compare Figure 5.11). It is immediately clear from the above that the matrix in Equation (4.28) could be used as T_1, as could be any other transformation uniquely defined by the vanishing point; a nicer example is the matrix

$$\mathsf{T}_1 = \begin{pmatrix} x & y & 0 \\ -y & x & 0 \\ \frac{-xz}{\sqrt{x^2+y^2}} & \frac{-yz}{\sqrt{x^2+y^2}} & \sqrt{x^2+y^2} \end{pmatrix} \tag{5.1}$$

if the vanishing point is given as $(x, y, z)^{\mathsf{T}}$ and $x^2 + y^2 \neq 0$, or the matrix

$$\mathsf{T}_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{5.2}$$

if $x = y = 0$. All that remains is to find the 3 degrees of freedom transformation

$$\mathsf{T}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & t_y \\ 0 & p_y & s \end{pmatrix} \tag{5.3}$$

for which a closed form solution exists.

The decomposition of T into T_1 and T_2 is strictly speaking only possible if either all lines $\boldsymbol{\ell}'_i$ in the intermediate canonical frame are exactly horizontal, or if $p_y \equiv 0$, since $\mathbf{T}_2$ with $p_y \neq 0$ will change the angle of all non-horizontal lines. However, if the assumption that all lines were originally parallel is true, and if the vanishing point used to determine $\mathbf{T}_1$ was calculated using one of the methods described in Section 4.4, then we can also guarantee that the lines in the intermediate frame *are* as horizontal as possible — any deviation must be an error in the measurements, which *should* be corrected — and the change in the angle will be small (and can in fact be ignored). The results of a Monte-Carlo simulation in Figure 5.12 show that the above approximation works quite well, although it is clear that the small-error assumption is *not* valid anymore for the resulting lines.

It is quite instructive to have a closer look at the matrix $\mathbf{T} = \mathbf{T}_2\mathbf{T}_1$. It was already mentioned that it has 5 degrees of freedom. These determine uniquely (up to scale) the last two rows — the first row can be chosen arbitrarily as long as the matrix does not become singular. We also see, when backprojecting the line at infinity $\mathbf{T}^\mathsf{T}(0, 0, 1)^\mathsf{T}$, that the third row is nothing but the vanishing line in the image, fixing 2 degrees of freedom. By the same argument we see that the second row is the backprojection of the horizontal line through the origin $(0, 1, 0)^\mathsf{T}$, leaving 1 degree of freedom to be fixed. What is the remaining degree of freedom used for? It is easy to see that any line passing through the vanishing point in the image (and therefore horizontal in the canonical frame) can be constructed as a linear combination of the second and third row by calculating $\mathbf{T}^\mathsf{T}(0, b, 1 - b)^\mathsf{T}$. The last degree of freedom fixes where in the image a line with given b will be located; it corresponds to a relative scale-factor or weight between the two lines. It should be mentioned for completeness that the first row of $\mathbf{T}$ determines where the vertical line through the origin $(1, 0, 0)$ will be projected and its relative scale compared with the third row gives the position of all other vertical lines after backprojection.

An alternative method for the calculation of additional lines should be mentioned for completeness. This method uses three of the four lines as a projective base and calculates the fourth line $\boldsymbol{\ell}_i$ with given cross-ratio, compare [72, 138]. However, this method has the two problems that it only uses three out of four lines (and therefore discards one-fourth of the information) and in addition has to choose an auxiliary vector $\boldsymbol{\ell}_X$ as described in Section 4.5.2. Tests have shown that the results are too unstable to be used under any but the most restrictive circumstances.

5.3.4 Merging Hypotheses

The preceding steps usually generate a high number of possible hypotheses. In particular, for a completely flat zebra crossing with N uninterrupted stripes, $(N - 1)$ identical hypotheses will ideally have been created, one for each pair of two consecutive stripes. For real-world zebra crossings, this will not usually happen

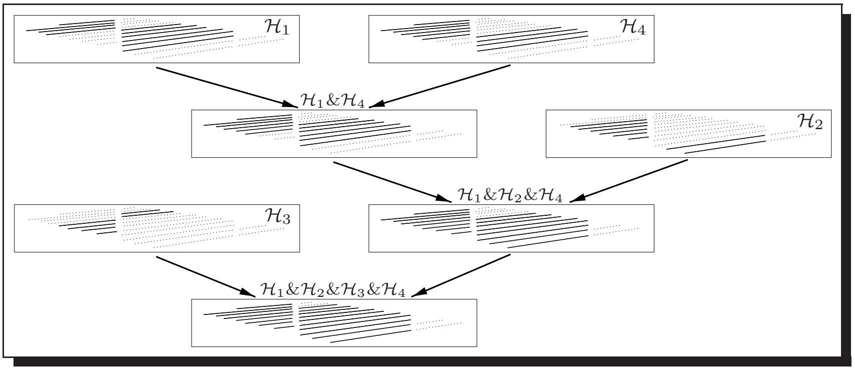

$\mathcal{H}_1$
$\mathcal{H}_4$
$\mathcal{H}_1\&\mathcal{H}_4$
$\mathcal{H}_2$
$\mathcal{H}_3$
$\mathcal{H}_1\&\mathcal{H}_2\&\mathcal{H}_4$
$\mathcal{H}_1\&\mathcal{H}_2\&\mathcal{H}_3\&\mathcal{H}_4$

Figure 5.13: *Merging the four hypotheses $\mathcal{H}_1$–$\mathcal{H}_4$. The hypotheses are ordered by number of stripes within the hypotheses.*

due to the street's slightly nonplanar surface, which usually only allows to group between 3 and 5 stripes into one hypothesis. But even these hypotheses will not all be unique and will, as a rule, overlap to a considerable extent.

Also, if the zebra crossing was partly occluded, a single stripe will often fracture into several smaller stripes, all of which will as a rule be part of some hypothesis sharing stripes with other hypotheses. Merging hypotheses not only reduces the number of hypotheses under consideration, but also connects the different parts of a partly occluded stripe, as seen in Figures 5.4 (on Page 126), 5.5 (on Page 127) and 5.15, second row, right image (on Page 141).

When merging hypotheses, it is in my experience advantageous to sort the individual hypotheses by number of stripes, to start with the hypothesis with the highest number of stripes, and cycle through all the other hypotheses (in order), adding hypotheses where appropriate, until no additional hypothesis can be added. The same step is repeated for the remaining hypotheses (using the second biggest hypothesis) and so forth, until all hypotheses have been merged where possible. Figure 5.13 show the individual steps and their results for the zebra crossing from Figure 5.4 on Page 126. This removes all duplicates and typically leaves only a small set of hypotheses. Which of these actually form zebra crossings, and which just share a similar structure, cannot be decided using the projective model described in Section 5.2.2. Only if one of the more constrained models is used are additional test possible, and these are described in Section 5.4.

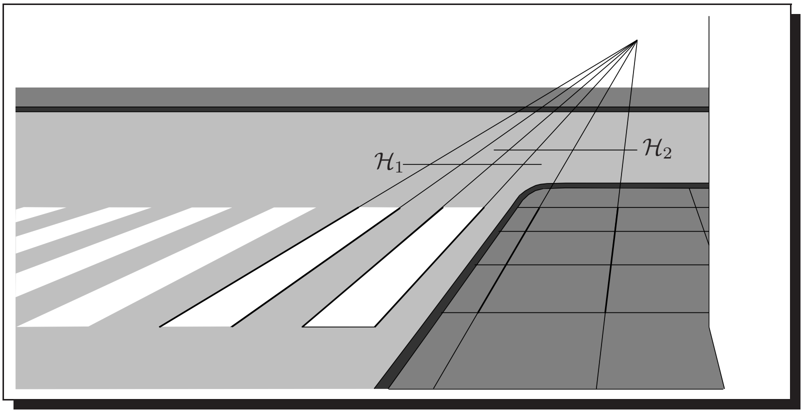


Figure 5.14: *Example of one stripe belonging to two unconnected hypotheses. $\mathcal{H}_1$ has correctly been formed by two stripes, while $\mathcal{H}_2$ has been formed by one stripe and one row of tiles from the pavement. Both hypotheses have the correct crossratio, but different vanishing lines (hypothesis $\mathcal{H}_2$ would normally have been filtered out during verification due to its necessarily non-horizontal vanishing line).*

The decision on whether a new hypothesis will be added to an existing one should in theory be based on a χ^2-test on the location of vanishing point, vanishing line, a reference line and the crossratio. This approach is, however, handicapped by the fact that the street's surface is generally nonplanar and of a three-dimensional form not easily modelled for error propagation. Within MOVIS I therefore used the much simpler criterion that two hypotheses have to share at least a certain number of stripes in order to be merged. A minimum of two stripes would be required for a theoretically sound solution; I have, however, found that at least for the detection of zebra crossings it is usually sufficient if the two hypotheses to be merged share just one stripe. This can fail in the rare case (only observed once so far) that due to an accidental constellation of additional line segments within the image, one stripe belongs to two different hypotheses, which differ only in the location of the vanishing line — the vanishing point should already be reasonably fixed by the common stripe's two line segments, one of which can double as a reference line. Figure 5.14 shows an example for such a constellation; Figure 5.15, bottom right (on Page 141), shows the only example of such a mismatch encountered so far. Here, the last stripe of the zebra crossing forms a hypothesis with some of the slabs on the pavement which does not match the other hypothesis generated.

Complementing the condition of at least one common stripe by a χ^2-test on the vanishing line reduces the number of false positives (one stripe and the vanishing line uniquely define the hypothesis' geometry) but at the same time increases the number of false negatives due to the street's usually convex surface. The χ^2-test itself is considerably simpler than the one described in Section 4.6.2; the two vanishing lines must, by necessity, not only share the angle and distance from the origin, but also the centre point — the vanishing point (assuming a representation by angle and centre, $\boldsymbol{\ell} = (\alpha, x, y)^\mathsf{T}$). We can therefore test

$$(\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2)^\mathsf{T} \boldsymbol{\Sigma}_1^{-1} \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2) \leq \chi^2_{p,3}. \tag{5.4}$$

Replacing $\sigma^2_{\alpha_i}$ in $\boldsymbol{\Sigma}_i$ with ${\sigma'_{\alpha_i}}^2 = \sigma^2_{\alpha_i} + \sigma^2_\alpha$ would allow for some variability in the vanishing-line's orientation; this is in accord with a street's cylindrical surface, with the cylinder's axis parallel to the stripes, but of otherwise unknown shape, which would affect only the vanishing-line's orientation[8].

5.4 Verification

Not many verifications can be done using the projective camera model described in Section 5.2.2, and these have already been incorporated into the grouping algorithm described above. However, the situation drastically improves once the constrained perspective camera model or the quasi calibrated camera model are used. The additional constraints these introduce are described below. While these tests could be applied to the final hypotheses, it should be noted that it is generally much more efficient to incorporate these constraints directly into the algorithm. It is only for the sake of discussion that here they are listed separately.

5.4.1 Constrained Perspective Camera Model

The verifications possible when using the constrained perspective camera model are a direct application of the constraints given in Section 5.2.3. We have seen in Section 5.3.3 that, based on a set of four line segments with given crossratio, it is possible to calculate the location of the horizon if the four lines were indeed part of a zebra crossing. Under the constrained perspective camera model the horizon has to be completely above the line segments, approximately horizontal and in or near the image; and these constraints on the position of the horizon will conversely

[8]Assuming a convex surface, more detailed predictions are in fact possible. So should α increase monotonically with increasing distance of the stripes from the observer.

also constrain the stripes themselves to plausible image positions. Within MOVIS, hard but very accommodating thresholds are used on the latter two conditions.

As mentioned above, it is prudent to include these checks directly into the above algorithm, just after the calculation of the horizon, and before additional lines are predicted. This will considerably reduce the number of hypotheses to be considered in later steps.

5.4.2 Quasi-Calibrated Camera Model

Using a quasi calibrated camera means that rough approximations exist for the camera's intrinsic parameters (compare Section 2.3.6): the focal length as printed on the lens (or simply an educated guess), the scale factors as found in the camera's manual, the image centre as principal point. In addition, it is very often possible to give a good guess for an external parameter — the height from which the image was taken. Based on these values, further verifications are possible as follows:

If we define the camera-position to be at the origin and use the coordinate system from Figure 2.1 on page 19, we can calculate the camera's orientation using the vanishing-line's position within the image and the internal camera parameters. If we define the three angles $(\varphi_x, \varphi_y, \varphi_z)$, where φ_x is a rotation around the x-axis and $\mathbf{R}_x$ the corresponding matrix of rotation, we can combine these matrices into a single matrix of projection $\mathbf{P} = \mathbf{P}_{\text{camera}}\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z$ such that the correspondence between the vanishing line $\boldsymbol{\ell} = (a, b, c)^\mathsf{T}$ measured in the image and the horizon $\boldsymbol{\ell}''' = (0, 1, 0)^\mathsf{T}$ becomes

$$\boldsymbol{\ell} = \mathbf{P}_{\text{camera}}^{-\mathsf{T}}\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z\boldsymbol{\ell}''' \tag{5.5}$$

$$\boldsymbol{\ell}' = \mathbf{P}_{\text{camera}}^{\mathsf{T}}\boldsymbol{\ell} = (a', b', c')^\mathsf{T} \tag{5.6}$$

$$\boldsymbol{\ell}'' = \mathbf{R}_x^{\mathsf{T}}\mathbf{P}_{\text{camera}}^{\mathsf{T}}\boldsymbol{\ell} = (a'', b'', 0)^\mathsf{T} \tag{5.7}$$

It is then easy to calculate the angles of rotations as

$$\varphi_x = \arctan(c'/b') = \arctan\left(\frac{at_x + bt_y + c/f}{bs_y}\right) \tag{5.8}$$

$$\varphi_y = \varphi_y \tag{5.9}$$

$$\begin{aligned}\varphi_z &= \arctan(-a''/b'') \\ &= \arctan\left(-\frac{as_x}{bs_y\cos(\varphi_x) + (at_x + bt_y + c/f)\sin(\varphi_x)}\right)\end{aligned} \tag{5.10}$$

Note that φ_y can not be determined from the above, but could be arbitrarily set to 0. Alternatively, a value could be calculated from the direction of the vanishing point.

Once the matrix $\mathbf{P}$ is completely known we can calculate, for each pixel $(x, y, 1)^\mathsf{T}$ in the image, the corresponding ray of possible positions in 3D

$$\lambda \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{P}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{5.11}$$

If we further assume that the camera is located at a distance h above the ground (i. e. the ground is parallel to the X-Y-plane at $-h$), the ray's intersection with the ground will be at

$$\left(-h\frac{X}{Z}, -h\frac{Y}{Z} \right)^\mathsf{T} \tag{5.12}$$

From there it is easy to calculate the hypothetical zebra-crossing's position on the ground.

The above can easily be used for validation purposes. If the height above ground h is assumed known we can check for the individual stripe's widths (these will all be identical due to the particular construction chosen to compute the backprojection) and see how good this conforms to a given width. If we assume h unknown we can calculate the h that results in a given width of the stripes (50 cm for a German zebra crossing) and check whether this height is within sensible bounds. The latter corresponds to a particular canonical frame (compare Section 4.4.2.2) which can be parameterised by the location of the vanishing line and the height.

If instead of a quasi-calibrated camera we use a fully calibrated camera (all internal parameters and possibly the height above ground) as would have been the case within a commercially available system (and within MOVIS we would also know the pitch-angle), this simply allows the use of tighter bounds and could ultimately lead to the application of additional (or a single, combined) χ^2-test on, e. g., the position of the horizon and reconstructed height of camera.

5.5 Results and Discussion

Figures 5.15 and 5.16 show several examples of hypotheses for zebra crossings that were generated using the model of a quasi-calibrated camera as described above. Although both the grouping as well as the verification are based on geometric constraints alone, the recognition has nonetheless proven extremely reliable.

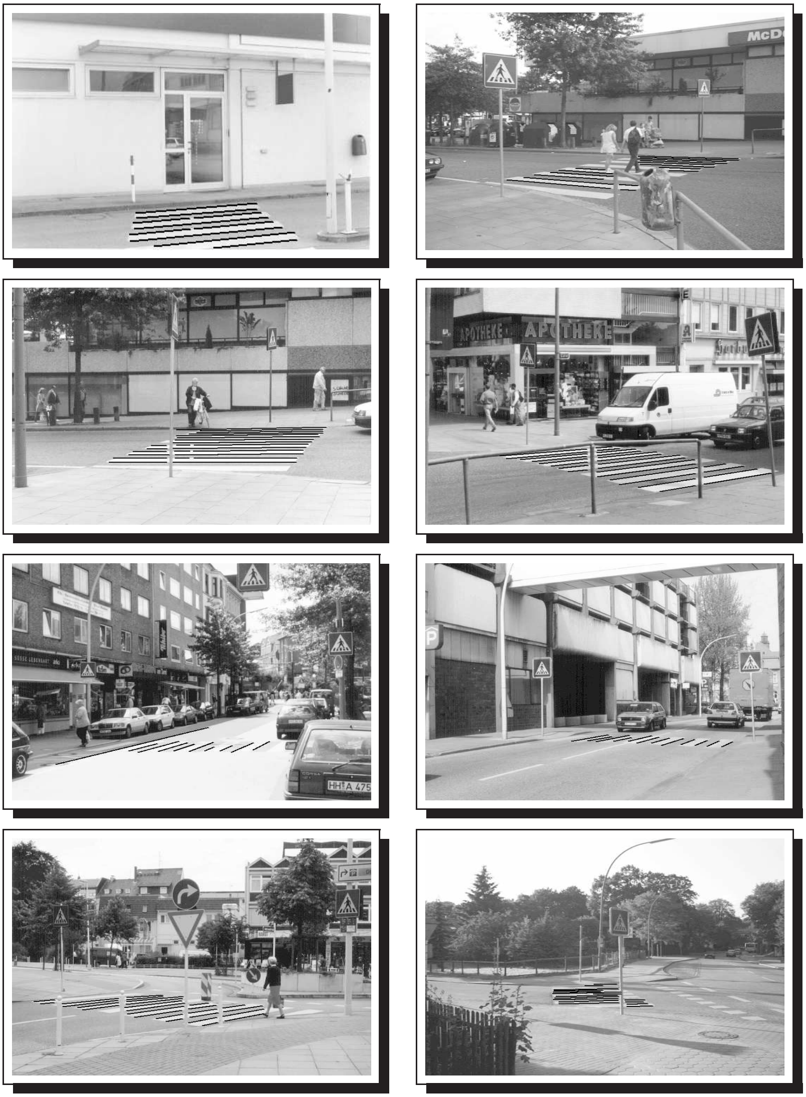


Figure 5.15: *Examples of recognised zebra crossings in outdoor-scenes.*

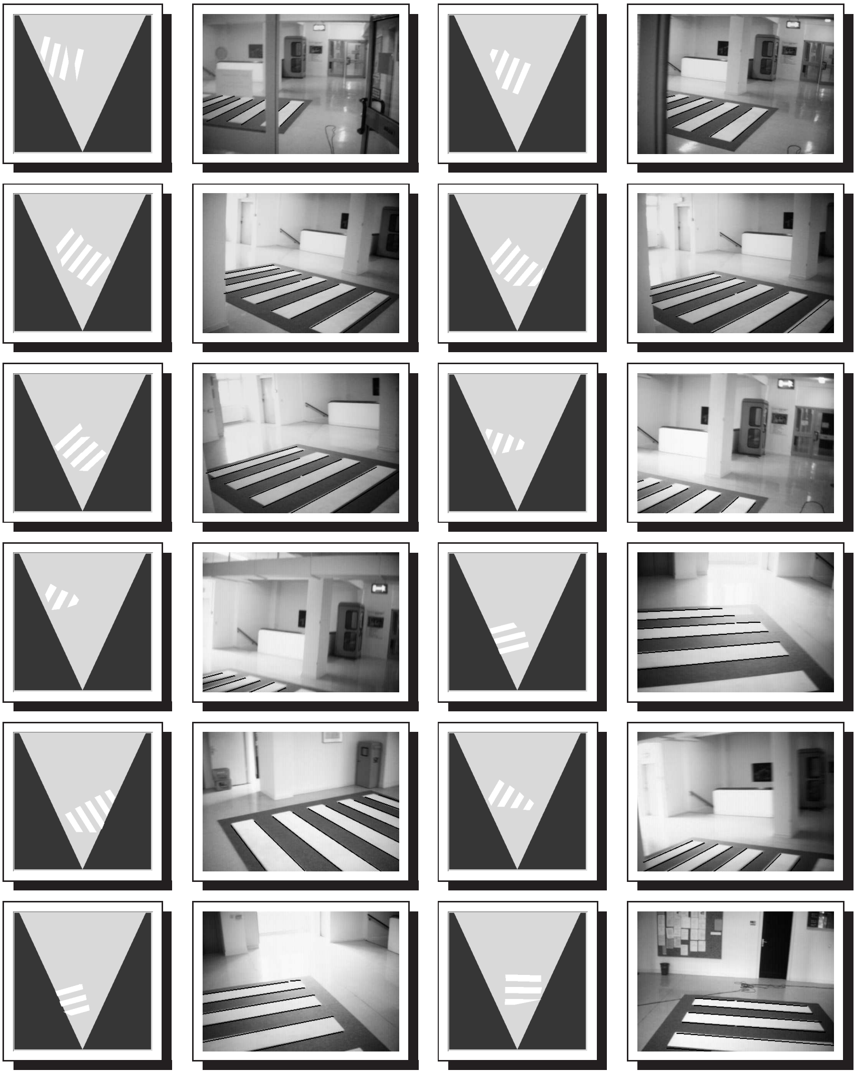
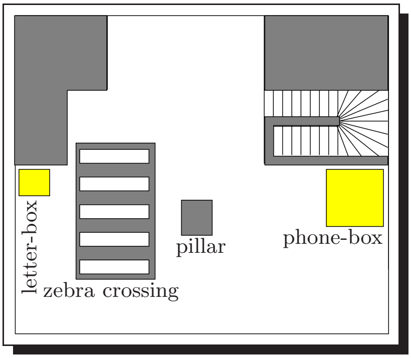


Figure 5.16: *Examples of recognised zebra crossings in indoor-scenes. To the left of each image you can see a simulated birds-eye view.*



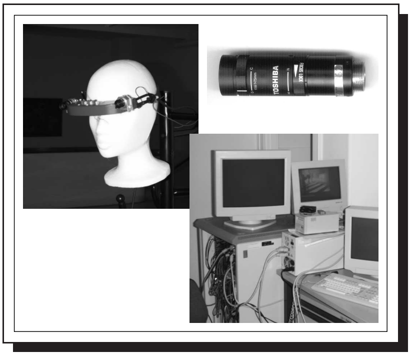


Figure 5.17: *The indoor-environment and hardware-prototype used for testing MOVIS.*

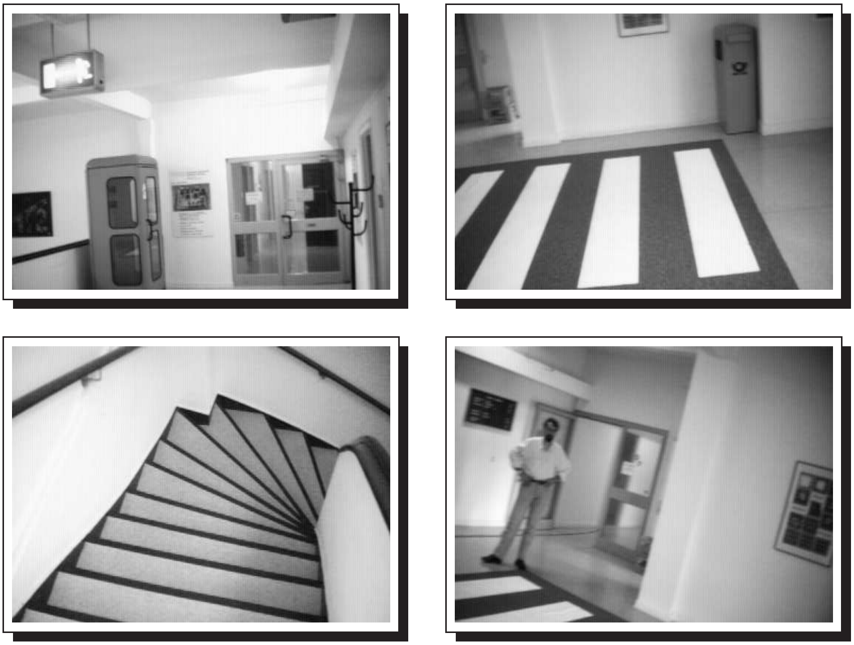


Figure 5.18: *Sample views taken using the original MOVIS hardware.*

Extensive tests of the algorithm were performed as part of MOVIS. These included 184 randomly taken images of street scenes, with image sizes varying from 439 pxl × 299 pxl up to 1024 pxl × 682 pxl and of varying quality. Figure 5.15 alone contains images taken by three different operators with four different cameras (three SLR-cameras, scanned in using two different scanners, and one digital camera) in three different resolutions (1024 pxl × 682 pxl, 800 pxl × 600 pxl, 439 pxl × 299 pxl), but all recognised using the same set of parameters[9]. In addition, the algorithm was tested within an indoor-environment using the actual MOVIS-hardware which consisted of a portable spectacle-like device containing two miniature colour cameras, connected to a stationary computer by a 30 m cable. This hardware was capable of producing an image size of 512 pxl × 286 pxl (using only half-frames). More than 300 of these images were tested off-line, and several thousand online, as part of demonstrations given to interested visitors. Figure 5.17 gives an idea of the indoor-environment and actual hardware used, Figure 5.18 shows a number of sample-views taken with the MOVIS-Equipment.

All these tests impressively demonstrated that even with a haphazardly chosen set of parameters constant over all images[10] more than 70 % of all zebra crossings with at least 3 visible stripes are correctly identified; and many of the approximately 30 % false negatives already failed due to problems during edge detection (usually insufficient contrast or extremely narrow stripes). The only other noteworthy source of false negatives was the perceptual-grouping approach introduced in Section 5.3.1.2 for reasons of efficiency. The grouping itself, once a suitable set of 4 lines had been found, performed extremely reliably.

What is more, so far not a single false positive has ever been observed, although it is of course clear from the algorithm described above that false positives can occur. It should, however, be noted that with the model of a quasi calibrated camera, false positives are limited to two cases. In the first one, a structure will result in a false positive only from a single position — slightly changing the position of the observer will eliminate the false positive. This is therefore not a problem for an application like MOVIS, where the observer is constantly moving. The other case is that of markings on the ground that do have the geometry of a zebra crossing. It is unclear how this could ever be distinguished from a real zebra crossing based on geometry alone, as its geometry is effectively that of a zebra crossing.

The high reliability of the algorithm would not have been possible without the

[9]It would in fact be advisable to use a different set of parameters for the digital camera for maximum performance, as it has a higher variance in the edgel positions due to the fact that it is a 1-chip colour camera.

[10]Due to the differences in image geometry and optical resolution two sets of parameters were used, one for the outdoor images, and one for the indoor ones.

combination of projective geometry with statistical methods as described in Chapter 4. A first implementation of the above algorithm, based only on the usual methods of projective geometry, never recognised more than at most two zebra crossings even with a finely tuned set of parameters. What is more, the current algorithm is extremely stable with regards to variations in the parameters, as all parameters basically only specify a probability, usually used in a χ^2 test. And it is this use of the χ^2 test as the main decision instrument (rather than finely tuned thresholds on direct measurements) which would allow us to easily incorporate additional information or additional constraints — at least as long as those data can be modelled by variance alone. The next chapter gives some more examples.

Chapter 6

Detecting Orthogonal Structures

[The universe] cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language, and the letters are triangles, circles and other geometrical figures, without which means it is humanly impossible to comprehend a single word.

Galileo Galilei, Opere Il Saggiatore, 1564–1642

6.1 Introduction

When moving within a man-made environment we are surrounded by orthogonal structures. This is particularly true for buildings, and a number of publications [36, 87, 97, 143] describe the reconstruction of such orthogonal structure from single images rather than — or at least in addition to — the now customary multi-view approaches. However, all these reconstruction methods need as input essentially manually grouped regions or features. This chapter outlines an approach for the detection and grouping of orthogonal structures in images which could eventually serve as input to these algorithms and thereby as a step towards a fully automated single-view system. This particular application was chosen as the diverse scales (vanishing points versus line-continuation) and accuracies (long line segments versus short line segments, but also different accuracies for the 3D-model) allow me to showcase a number of different ideas and approaches.

The appearance based grouping was inspired by work done by Brillault-O'Mahony in the late 80s and early 90s [20, 21], where she presented an approach for the unsupervised qualitative reconstruction of a scene from edges alone (to be matched against a CAD-model) based on the assumption of a Legoland world, and where she introduced the notion of subjective structure as well as some first attempts to take errors into consideration. The use of orthogonality and vanishing points also owes much to work done at the Departimento di Fisica dell'Università di Genova, e. g. by Coelho, Straforini, Campani, Parodi, Piccioli, and Torre [15, 30, 108, 109, 111, 148]. The main difference here is that their work was based on the complete interpretation of the graph of all edges, identifying realisable solutions using traditional tools of consistent labelling. This approach of course only works well if a complete (and consistent) graph is given; in contrast the algorithms outlined in this chapter expect wrong and missing information and their performance therefore degrades more gracefully. An additional difference is the assumption of an essentially Legoland world (exactly 3 orthogonal directions) by Coelho et al., while most of the algorithms presented here can not only deal with $2n + 1$ directions (in n orthogonal sets), but in fact benefit from the presence of more than 3 directions (this is in particular true for the calibration described in Section 6.3.2).

The remainder of this chapter is organised as follows: Section 6.2 describes the 3D and the camera models. Section 6.3 describes the different stages of grouping in order, starting with the grouping of line segments by vanishing points in Section 6.3.1. There I present a new algorithm for the iterative improvement of vanishing-point positions in Section 6.3.1.1 and one for the automatic grouping of vanishing points in Section 6.3.1.2. It is well known that a partial camera-calibration is possible based on vanishing points, and in Section 6.3.2 I present a new objective function which takes the different uncertainties in the positions of the vanishing points into account and naturally extends the usual Legoland assumption to more general setups. Section 6.3.3 discusses how best to merge collinear line segments, extending our work from [54] to make use of vanishing-point information, and presenting a new algorithm which in the general case lowers the complexity of merging line segments from $\mathcal{O}(\mathcal{N}^{\in})$ to $\mathcal{O}(\mathcal{N}\log(\mathcal{N}))$. Section 6.3.4 finally combines the previous information, sketching a possible approach for grouping, again extending our work from [54] with vanishing-point information. Section 6.4 then allows a closer look at the performance of some of the algorithms outlined before, and with a particular focus on the integration of error models for 2D and 3D: Section 6.4.1 compares the relative performance of several 2D-error models, both new ones first introduced in this thesis as well as established ones from the literature, for the identification of collinear line segments; we will see how many of the established error measures perform rather poorly, but also how a computationally very simple measure performs much better than could have been expected. Following this look at 2D-error models, Section 6.4.2 introduces a simple 3D-error model and its application to the grouping of line segments by vanishing points in Section 6.4.2.1 and the merging of collinear line segments in Section 6.4.2.2. Section 6.5 finally presents and discusses some results.

6.2 Model

The model is further subdivided into the underlying 3D-model (see Section 6.2.1) and the camera models in Sections 6.2.2ff — the same as used in Sections 5.2.2ff on the detection of repeated parallel structures with known cross-ratio.

6.2.1 3D Model

In order to model generic views of buildings and clusters of buildings, as well as similar box-like structures, we will make the following abstractions:

1. All objects consist of planar faces only, mainly the walls.
2. All walls are vertical.
3. All intersections between walls are right-angles.
4. All walls contain mostly vertical and horizontal texture (e. g. the lines delimiting windows or doors).

5. For each individual wall the vertical and horizontal line segments delimiting windows and doors are mostly aligned with each other.
6. All remaining objects are essentially untextured or randomly textured.

Note that this model does not require individual buildings to be aligned in any particular way, except for sharing a common vertical orientation. We will therefore as a rule get one vertical direction and $2n$ horizontal directions (grouped into n pairs of orthogonal directions, corresponding to one house-corner each), not all of which are necessarily visible in any one image. Very often we will indeed only have 3 dominant directions ($n = 1$), corresponding to three mutually orthogonal directions in reality.

As was the case with the model of a zebra crossing in Section 5.2.1, the above is only an approximation of the truth. Anybody who owns a house, and in particular an older one, knows that walls are rarely absolutely vertical, corners never completely orthogonal, window sills are never absolutely accurately aligned, and edges never completely parallel. And although these deviations are usually small when compared to resolution related artifacts, it is none the less necessary to account for them by an adequate error model, as we will see below. In keeping with the tenor of this thesis this error model will however only model slight (accidental) deviations from the above 3D-model, such as can reasonably be described by Gaussians.

It is easily possible to incorporate saddle roofs into this model as the intersection of two rectangular areas with corresponding angles with the ground-plane α and $\pi - \alpha$. It is, however, my experience that there is generally not enough evidence for roofs in any given image (except for aerial images) to afford the automatic segmentation of roofs from edges alone; this is only reasonably possible within a supervised system (and even then evidence if often too scarce).

6.2.2 Projective Camera Model

This is the model discussed in Section 2.3.4, which is the most general linear camera model available, and can be parameterised by the concatenation of a 3D–2D projection and a general 8 DOF projective transformation. This model is used here together with a Gaussian sphere parameterisation as described in Section 2.9, which projects straight lines into great circles on the sphere and points onto points.

Only very little can be said about the structure's appearance after projection in the case $n = 1$, i. e. a so-called Legoland world with only three mutually orthogonal directions, namely

- For each image, we will observe at most $2n$ horizontal and one vertical vanishing-point belonging to the observed structure, as well as an unknown number m of additional vanishing points not belonging to the observed structure; this follows from Items 1, 4, and 6.
- Line segments on parallel walls share the same two vanishing points; this follows from Items 1 and 4
- Line segments that were collinear in 3D are also collinear in the image, compare Item 5.

The main reason that so little can be said about the structure's appearance after projection is due to the fact that for $n = 1$ and a projective camera it is impossible to distinguish the vertical and horizontal directions (although it is possible to make an educated guess based on the structure of Y-junctions). This changes, however, as soon as $n > 1$ (or, more accurately, as soon as more than 2 horizontal directions can be observed within the projection). We then get:

- The vanishing points of all sets of horizontal line segments lie on a great circle on the Gaussian sphere corresponding to the horizon; this follows from 4.
- All vertical line segments on all walls intersect in one common vanishing point on the Gaussian sphere which is not located on the great circle of horizontal vanishing points; this follows from Items 2 and 4.

If saddle-roofs are taken into account it is also possible to state that all roofs with the same gradient will produce vanishing points on a circle (not great circle) located between the horizontal great circle and the vertical vanishing point. There will, however, be generally insufficient data to observe this circle in actual images.

It is clear from the above that the grouping and recognition of orthogonal structures is difficult from arbitrary projective transformations, in particular as no information about the possible viewpoint is given. This changes considerably once a constrained perspective camera model is used as in the next section.

6.2.3 Constrained Perspective Camera Model

This camera model constrains the transformation from the 3D-world into the 2D-image to be a perspective transformation as described in Section 2.3.3, and adds the knowledge about an approximate horizontal and vertical direction within the image, as well as the assumption that the underlying image was taken by a human or otherwise known operator, i. e. from approximately head-height. This additional knowledge allows us to differentiate between the two horizontal and the one vertical direction even for a Legoland world, giving us access to the full set of conditions for the projective case above. In addition, we can also state that:

- The horizon's position is approximately known.
- The horizon cuts across the individual walls, i. e. each wall will have corners above as well as below the horizon.

The above allows us to distinguish between up and down in addition to horizontal and vertical. This distinction can significantly aid the verification or reconstruction, as can a comparison of the horizon's calculated position with its assumed position. This is similar to the approach used in Section 5.4.1.

6.2.4 Quasi-Calibrated Camera Model

The quasi-calibrated camera model adds approximate knowledge about the camera's internal parameters — focal length f, aspect-ratio a, principal-point $(x_0, y_0)^\mathsf{T}$, and, for non-CCD cameras, skew s — as well as approximate knowledge about the height h from which the image was taken. This is for example the case when an image was taken with a known camera. This knowledge allows for a qualitative (and nearly quantitative) reconstruction. In particular, we get:

- vanishing points which are orthogonal in 3D will be nearly orthogonal on the Gaussian sphere.

This allows for the automatic selection of three mutually orthogonal directions (not possible under a less restrictive model), which can then be used for calibration of the internal camera-parameters [26, 28, 41, 155, 157] described in Section 6.3.2, which in turn allows for a possible reconstruction of the scene up to scale, which is determined by the only approximately known height h from which the image was taken.

6.3 Grouping

Based on the models described above it is possible to outline a scheme for the grouping and segmentation of orthogonal structures. In a first step a new algorithm for the iterative refinement and automatic grouping of vanishing points is used to identify the main directions within the image, this is described in Section 6.3.1. These vanishing points can then be used for a partial camera-calibration as described in Section 6.3.2, where I present a new objective function which takes the different uncertainties in the positions of the vanishing points into account and naturally extends the usual Legoland assumption to more general setups. Based on the identification of the individual directions collinear line segments can be merged (Section 6.3.3, which presents an extension from our work in [54] as well as a new algorithm which in the general case lowers the complexity of merging line segments from $\mathcal{O}(\mathcal{N}^{\in})$ to $\mathcal{O}(\mathcal{N}\log(\mathcal{N})))$; this information can then be used to identify areas corresponding to individual walls (Section 6.3.4, which again extends [54] to make use of vanishing-point information).

6.3.1 Vanishing Point Detection

Vanishing points or vanishing-directions[1] are easily the single most important feature used here for the grouping and segmentation of orthogonal (block-like) structure — Parodi and Torre [110] showed in 1993 that using vanishing-point information it is possible to reduce the algorithmic complexity of scene interpretation from an NP-problem to linear time in the number of line segments, see also [109].

Traditionally, two different approaches for vanishing-point extraction exist and have remained mainly unchanged ever since Barnard [13] and Magee and Aggarwal [94] published their algorithms in 1983 and 1984 respectively[2]. Both suggested the use of the Gaussian sphere as an accumulator array for a Hough-transform. Barnard suggested a Hough-transform on lines, while Magee and Aggarwal used a Hough-transform on line-intersections, which avoids many of the pitfalls of Barnard's approach but is essentially an $\mathcal{O}(N^2)$ procedure, as opposed to Barnard's $\mathcal{O}(N)$ approach, where N is the number of line segments.

A plethora of algorithms for the detection of vanishing points have since been suggested. Most of these are incremental improvements to Barnard's [93, 107, 125, 142] or Magee's and Aggarwal's [15, 26] algorithms, although some interesting new approaches have also been tried [20, 31, 99, 149, 157]. Some of these are limited to particular applications, usually assuming one or more vanishing points either at infinity or at known positions [51, 90, 98, 105, 147], or requiring a calibrated camera [98].

As with edge detection, for which a similar number of algorithms exist, it is not particularly important which algorithm is finally chosen, as long as it gets the job done. I will in the following describe how, based on an initial vanishing-point position, that point's position can be improved upon in a way which fits well into the

[1]Using the ray-space or Gaussian-sphere model there is indeed no difference between a direction and its vanishing point, compare Section 2.9.

[2]What is now known as the Barnard algorithm was already published in 1982 by Fischler, Barnard, Bolles, and Lowry[45].

framework of this thesis due to its use of error propagation and statistical properties, as the final calculation of the vanishing point's position from all corresponding line segments is based on one of the accurate approaches described in Section 4.4. For the examples in this thesis I simply calculated an initial position as the intersection of two handpicked line segments, however, any of the above-mentioned algorithms should be able to generate appropriate initialisations.

6.3.1.1 Implementation

The algorithm for the iterative improvement of vanishing points presented in this section has, to my knowledge, not been presented before. It is both efficient as well as highly accurate, taking into account the line-segments' individual distributions (which no other algorithm for the calculation of vanishing points does to this extent, as far as I know). Its iterative nature is comparable to [31, 136, 150].

Starting with a number $i = 1 \dots N$ of initial positions $\mathbf{p}_i$, possibly with covariance matrix $\boldsymbol{\Sigma}_{\mathbf{p}_i}$

1. For each $i = 1 \dots N$, repeat until convergence[3]:
   (a) Calculate an updated vanishing point $\mathbf{p}_i$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{p}_i}$ using all supporting line segments (i. e. that pass a χ^2-test for a given significance-level p_0).
   (b) Optionally: (temporarily) remove all supporting line segments, i. e. all line segments consistent with the assumption that they could pass through the given intersection $\mathbf{p}_i$ (i. e. that pass a χ^2-test for a given significance-level $p_1 \leq p_0$).

As for Step 1a it can not be stressed enough how important it is to use an adequate model capable of handling intersections at infinity. The Gaussian sphere (Section 2.9) is well suited for this purpose, as is the scaled version of a Gaussian sphere proposed by Kanatani, the N-vectors. Any Euclidean model is unsuitable, as intersections far away from the image centre will result in greatly overestimated and biased error-regions, as we have seen in Section 4.4. And as the potential vanishing point could be anywhere, from the image centre to infinity, and the line segments vary greatly in length and therefore accuraccy, it is also necessary to use an apropriate error model and χ^2 value instead of a fixed threshold. In the following I essentially use the same error measure which I also used when finding

[3]There is actually no guarantee that this iteration will converge at all, but experience shows that a couple of iterations is usually all it takes, so you can simply iterate 10 times or until the residuum (or the number of line segments) converges.

the intersection of n lines in Section 4.4 — I will consider a line segment $\boldsymbol{\ell}'$ as supporting a possible vanishing point $\mathbf{p}$ (both given as homogeneous N-vectors) if

$$\frac{\mathbf{p}^{\mathsf{T}}\boldsymbol{\ell}'\boldsymbol{\ell}'^{\mathsf{T}}\mathbf{p}}{\mathbf{p}^{\mathsf{T}}\boldsymbol{\Sigma}'_{\boldsymbol{\ell}}\mathbf{p}+\boldsymbol{\ell}'^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{p}}\boldsymbol{\ell}'} \leq \chi^2_{p,1} \tag{6.1}$$

where p is a suitably chosen required minimum probability (significance level). Usually, p can be chosen rather small (around $p = 5\,\%$) when looking for lines $\boldsymbol{\ell}'$ passing through the vanishing point $\mathbf{p}$, since most outliers are clearly recognisable as such; it might, however, be advisable to use a higher value (say $p = 50\,\%$) when selecting the line segments from which to calculate an updated position in Step 1a or when removing line segments from the set of available line segments in Step 1b, in order to avoid the use / removal of ambiguous line segments.

The covariance matrix $\boldsymbol{\Sigma}'_{\boldsymbol{\ell}}$ can be calculated from the true covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\ell}}$ of a line segment $\boldsymbol{\ell}$ in $(\alpha, x, y)^{\mathsf{T}}$ format as follows:

$$\boldsymbol{\Sigma}'_{\boldsymbol{\ell}} = \mathbf{J}_{\boldsymbol{\ell}'\boldsymbol{\ell}} \begin{pmatrix} \sigma_\alpha^2 + {\sigma'_\alpha}^2 & 0 & 0 \\ 0 & \sigma_x^2 + {\sigma'_x}^2 & \sigma_{xy} \\ 0 & \sigma_{xy} & \sigma_y^2 + {\sigma'_y}^2 \end{pmatrix} \mathbf{J}^{\mathsf{T}}_{\boldsymbol{\ell}'\boldsymbol{\ell}} \tag{6.2}$$

$$\text{with} \qquad \mathbf{J}_{\boldsymbol{\ell}'\boldsymbol{\ell}} = \begin{pmatrix} \cos(\alpha) & 0 & 0 \\ \sin(\alpha) & 0 & 0 \\ -x\cos(\alpha) - y\sin(\alpha) & -\sin(\alpha) & \cos(\alpha) \end{pmatrix}. \tag{6.3}$$

Note the additional terms ${\sigma'_\alpha}^2$, ${\sigma'_x}^2$ and ${\sigma'_y}^2$, which model the error in the 3D-model, where even for new buildings not all line segments might be exactly horizontal or vertical, and may not be exactly aligned either — and certainly they won't be for older buildings. Section 6.4.2.1 discusses the effect of these additional model-errors in more detail.

6.3.1.2 Identification of Vanishing Points

In order to allow automatic grouping it is important to be able to distinguish between vertical and horizontal vanishing points, and also to be able to group the corresponding horizontal vanishing points, i. e. such which in 3D are orthogonal. In the following I will describe possible approaches to solve both problems.

Differentiating between horizontal and vertical vanishing points is easy if we know that we are dealing with a constrained- or quasi-calibrated camera: the vertical vanishing point will be close to $(0, 1, 0)^{\mathsf{T}}$ with most of it's uncertainty along the y-position. All other vanishing points must by necessity be horizontal. For a projective camera model we will in general need 4 or more vanishing points to decide which one is the vertical (i. e. a model with $2n+1$, $n > 1$ vanishing-directions); we can then fit a great circle through *all* vanishing points and, removing them individually, find the vanishing point whose removal decreases the fitting error the most (alternatively we can start by fitting $2n+1$ great circles to $2n$ vanishing points each, and see which one is the best fit — the numerical expense is comparable for any reasonably small number n).

Grouping originally orthogonal pairs of vanishing points will be more complicated. In the case of a quasi-calibrated camera we can simply group vanishing points which are nearly orthogonal, but even for a constrained camera we will usually need to get a better idea of the true focal length first. Different methods for the calibration of a camera from vanishing points are described in the next Section, including an algorithm which can calculate the focal length of an unknown camera even if the correspondence between vanishing points is not yet known.

6.3.2 Focal Length Calculation

It is well known that vanishing points can be used to determine the internal camera-parameters principal point and focal length, as well as the external rotation (or rather a set of possible orientations, if the correlation between 2D vanishing points and 3D orientations isn't known) [26, 41, 164]. In the following I will concentrate on the calculation of the internal parameters focal length and principal point, as the skew and aspect-ratio are generally known for modern CCD-cameras. The underlying idea here is that for a calibrated camera any two directions which are orthogonal in 3D will also be orthogonal on the Gaussian sphere — as we have seen in Section 2.9.3. And we have seen that condition 3 essentially results in $2n+1$ vanishing directions in the image which form n sets of orthogonal directions (where each set shares the same vertical vanishing point). So all we need to do is to find the focal length f and principal point $(t_x, t_y)^\mathsf{T}$ for which the 3 angles in a set will be closest to a right-angle, i. e. we want to solve

$$\min_{f,t_x,t_y} \sum_{k=1}^{n} \sum_{i=1}^{2} \sum_{j=i+1}^{3} \left(\frac{\pi}{2} - \alpha\left(\mathbf{p}_{k,i}, \mathbf{p}_{k,j}, f, t_x, t_y\right) \right)^2 \tag{6.4}$$

where $\mathbf{p}_{k,i}$ is the i^{th} vanishing point in the k^{th} set; $\alpha\left(\mathbf{p}_{k,i}, \mathbf{p}_{k,j}, f, t_x, t_y\right)$ is the angle between two such vanishing points as a function of the original vanishing points $\mathbf{p}_{k,i}$ and $\mathbf{p}_{k,j}$ and the parameters f, t_x, t_y. Alternatively, if only the vertical vanishing-direction is known, but no correlation between horizontal vanishing points, we get the equation

$$\min_{f,t_x,t_y} \sum_{k=2}^{2n+1} \left(\frac{\pi}{2} - \alpha\left(\mathbf{p}_1, \mathbf{p}_k, f, t_x, t_y\right)\right)^2 \tag{6.5}$$

assuming that $\mathbf{p}_1$ is the vertical vanishing point. Note that the sum in (6.4) has $3n$ terms, while the sum in (6.5) has only $2n$ terms. This is of particular relevance in the common case $n = 1$; in that case there exists an exact solution for f, t_x, t_y using (6.4), while based on (6.5) we can only calculate an (exact) solution for f and either t_x or t_y. In both cases no error propagation will be needed (or, indeed, can be used), as the solution is unambiguous.

The interesting case for the purpose of this thesis is the case where either $n > 1$, or where we are calculating only f — e. g. in the lucky case where we are using a zoom-lens of unknown focal length, but have a principal point exactly in the middle of the image and therefore unaffected by zoom (for practical applications the reader would be well advised to consider [60], which generally discourages the calculation of the focal length without simultaneous calculation of the principal point). Then it becomes possible to take the different uncertainty-regions of the vanishing points into account, which as we have seen in Section 4.4.3 can vary considerably: for vanishing points near infinity (this is usually the case for the one vertical vanishing point) it will be a narrow but long region, while for vanishing points closer to the image we will get a small and nearly circular region.

So what does the hitherto unknown function $\alpha\left(\mathbf{p}_1, \mathbf{p}_k, f, t_x, t_y\right)$ look like? Assuming we already knew the focal length f and principal point (t_x, t_y) we can give the "corrected" position $\mathbf{p}'_i$ of the real vanishing point $\mathbf{p}_i$ on a Gaussian-sphere by

$$\mathbf{p}'_i = \frac{1}{\sqrt{\left(x_i - z_i t_x\right)^2 + \left(y_i - z_i t_y\right)^2 + z_i^2 f^2}} \begin{pmatrix} x_i - z_i t_x \\ y_i - z_i t_y \\ z_i f \end{pmatrix}^{\mathsf{T}} . \tag{6.6}$$

If the coordinates of the vanishing points are given by homogeneous coordinates we can model this effect by multiplication with a matrix

$$\mathsf{T} = \begin{pmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & f \end{pmatrix} \tag{6.7}$$

and subsequent normalisation; if the vanishing points were already given using Kanatani's N-vectors, that is in a format $(x_i, y_i, z_i \hat{f})$ with an approximate focal length $\hat{f}$ (which I would personally recommend), we can instead multiply with the matrix

$$\mathsf{T} = \begin{pmatrix} 1 & 0 & -t_x/\hat{f} \\ 0 & 1 & -t_y/\hat{f} \\ 0 & 0 & f/\hat{f} \end{pmatrix} \tag{6.8}$$

and, again, normalise. Note that we are of course not restricted to a 3 DOF matrix T — setting $t_x = 0, t_y = 0$ would give a 1 DOF matrix which solves only for f, and we could also solve for additional parameters *iff* we have enough constraints at our disposal.

The difference of angles in (6.4) and (6.5) is usually approximated by the cosine of the angle, i. e.

$$\frac{\pi}{2} - \alpha\left(\mathbf{p}_{k,i}, \mathbf{p}_{k,j}, f, t_x, t_y\right) \approx \cos\left(\alpha\left(\mathbf{p}_{k,i}, \mathbf{p}_{k,j}, f, t_x, t_y\right)\right) \ ; \tag{6.9}$$

this is done both to avoid the costly computation of the angle (which can be replaced by a scalar product), but also since the cosine is an excellent approximation of the angle near the correct solution $\alpha\left(\mathbf{p}_{k,i}, \mathbf{p}_{k,j}, f, t_x, t_y\right) = {}^{\pi}/_{2}$. We can now replace the error-term in Equations (6.4) and (6.5) by

$$d_{kij} = {\mathbf{p}'_{k,i}}^{\mathsf{T}} \mathbf{p}'_{k,j} = \cos\left(\alpha\left(\mathbf{p}_{k,i} \mathbf{p}_{k,j}, f, t_x, t_y\right)\right) \ . \tag{6.10}$$

Taking into account the vanishing-points' covariances we can therefore replace (6.4) and (6.5) with

$$\min_{f,t_x,t_y} \sum_{k=1}^{n} \sum_{i=1}^{2} \sum_{j=i+1}^{3} \frac{d_{kij}^2(\mathsf{T})}{\sigma_{d_{ij}}^2(\mathsf{T})} \tag{6.4a}$$

$$\min_{f,t_x,t_y} \sum_{k=2}^{2n+1} \frac{d_{1k}^2(\mathsf{T})}{\sigma_{d_{1k}}^2(\mathsf{T})} \tag{6.5a}$$

where the variance of the error measure (6.10) can be calculated using (3.53) as

$$
\begin{aligned}
d_{ij} &= \mathbf{p}_i'^{\mathsf{T}} \mathbf{p}_j' = \frac{\mathbf{p}_i^{\mathsf{T}} \mathbf{T}^{\mathsf{T}} \mathbf{T} \mathbf{p}_j}{\|\mathbf{T}\mathbf{p}_i\| \, \|\mathbf{T}\mathbf{p}_j\|} && (6.11) \\
\mathbf{J}_{d_{ij}\mathbf{p}_i} &= \frac{\mathbf{p}_i^{\mathsf{T}}}{\|\mathbf{T}\mathbf{p}_i\|^3 \|\mathbf{T}\mathbf{p}_j\|} \mathbf{T}^{\mathsf{T}}\mathbf{T} \left(\mathbf{p}_i \mathbf{p}_j^{\mathsf{T}} - \mathbf{p}_j \mathbf{p}_i^{\mathsf{T}}\right) \mathbf{T}^{\mathsf{T}}\mathbf{T} && (6.12) \\
\mathbf{J}_{d_{ij}\mathbf{p}_j} &= \frac{\mathbf{p}_j^{\mathsf{T}}}{\|\mathbf{T}\mathbf{p}_i\| \, \|\mathbf{T}\mathbf{p}_j\|^3} \mathbf{T}^{\mathsf{T}}\mathbf{T} \left(\mathbf{p}_j \mathbf{p}_i^{\mathsf{T}} - \mathbf{p}_i \mathbf{p}_j^{\mathsf{T}}\right) \mathbf{T}^{\mathsf{T}}\mathbf{T} && (6.13) \\
\sigma_{d_{ij}} &= \mathbf{J}_{d_{ij}\mathbf{p}_i} \boldsymbol{\Sigma}_{\mathbf{p}_i} \mathbf{J}_{d_{ij}\mathbf{p}_i}^{\mathsf{T}} + \mathbf{J}_{d_{ij}\mathbf{p}_j} \boldsymbol{\Sigma}_{\mathbf{p}_j} \mathbf{J}_{d_{ij}\mathbf{p}_j}^{\mathsf{T}} \\
&= \frac{\mathbf{p}_i^{\mathsf{T}} \mathbf{T}^{\mathsf{T}}\mathbf{T} \left(\mathbf{p}_i \mathbf{p}_j^{\mathsf{T}} - \mathbf{p}_j \mathbf{p}_i^{\mathsf{T}}\right) \mathbf{T}^{\mathsf{T}}\mathbf{T} \boldsymbol{\Sigma}_{\mathbf{p}_i} \mathbf{T}^{\mathsf{T}}\mathbf{T} \left(\mathbf{p}_i \mathbf{p}_j^{\mathsf{T}} - \mathbf{p}_j \mathbf{p}_i^{\mathsf{T}}\right) \mathbf{T}^{\mathsf{T}}\mathbf{T} \mathbf{p}_i}{\|\mathbf{T}\mathbf{p}_i\|^6 \|\mathbf{T}\mathbf{p}_j\|^2} && (6.14) \\
&+ \frac{\mathbf{p}_j^{\mathsf{T}} \mathbf{T}^{\mathsf{T}}\mathbf{T} \left(\mathbf{p}_j \mathbf{p}_i^{\mathsf{T}} - \mathbf{p}_i \mathbf{p}_j^{\mathsf{T}}\right) \mathbf{T}^{\mathsf{T}}\mathbf{T} \boldsymbol{\Sigma}_{\mathbf{p}_j} \mathbf{T}^{\mathsf{T}}\mathbf{T} \left(\mathbf{p}_j \mathbf{p}_i^{\mathsf{T}} - \mathbf{p}_i \mathbf{p}_j^{\mathsf{T}}\right) \mathbf{T}^{\mathsf{T}}\mathbf{T} \mathbf{p}_j}{\|\mathbf{T}\mathbf{p}_j\|^6 \|\mathbf{T}\mathbf{p}_i\|^2}
\end{aligned}
$$

Substituting (6.11) and (6.14) in (6.4a) we get a rather lengthy term, even though $\|\mathbf{T}\mathbf{p}_i\|^2 \|\mathbf{T}\mathbf{p}_j\|^2$ cancels out. However, this equation greatly simplifies if we assume that we are already near the minimum, i. e. $\mathbf{p}_i^{\mathsf{T}} \mathbf{T}^{\mathsf{T}} \mathbf{T} \mathbf{p}_j \approx 0$, Equation (6.4a) can then be written as

$$
\min_{\mathbf{T}} \sum_{k=1}^{n} \sum_{i=1}^{2} \sum_{j=i+1}^{3} \frac{\left(\mathbf{p}_{k,i}^{\mathsf{T}} \mathbf{T}^{\mathsf{T}} \mathbf{T} \mathbf{p}_{k,j}\right)^2}{\mathbf{p}_{k,j}^{\mathsf{T}} \mathbf{T}^{\mathsf{T}} \mathbf{T} \boldsymbol{\Sigma}_{\mathbf{p}_{k,i}} \mathbf{T}^{\mathsf{T}} \mathbf{T} \mathbf{p}_{k,j} + \mathbf{p}_{k,i}^{\mathsf{T}} \mathbf{T}^{\mathsf{T}} \mathbf{T} \boldsymbol{\Sigma}_{\mathbf{p}_{k,j}} \mathbf{T}^{\mathsf{T}} \mathbf{T} \mathbf{p}_{k,i}} \qquad (6.4b)
$$

6.3.3 Merging Line Segments

As stated in Constraint 5 on Page 150 I assumed that each face of a building is essentially structured by horizontal and vertical line segments — the line segments delimiting windows etc.— where several line segments align with each other. The face itself could therefore be described by the lines passing through these line segments or, put another way, we could merge the individual small segments into longer segments (and ultimately calculate the most likely line through these segments).

Joining collinear line segments into longer segments requires the comparison of each line segment with all other line segments, this has a complexity of $\mathcal{O}(N^2)$. If we sort the line segments by vanishing point first (as all collinear line segments must by necessity share the same vanishing point) we still end up with a complexity of $\mathcal{O}(\sum_{i=1}^{k} N_i^2) \approx \mathcal{O}(N^2)$, where N_i is the number of line segments belonging to the

i^{th} vanishing point and $N = N_0 + N_1 + \cdots + N_k$, with N_0 the number of outliers not belonging to any vanishing point. However, sorting the line segments by inclination allows us to compare each line segment with only a much smaller number of other line segments. If $\hat{\sigma}_\alpha = \max(\sigma_{\alpha_i})$ is the maximum standard-deviation observed for any line segment, then comparing only line segments with an angle α_i to our original line segment with angle α_j so that $\alpha_j - n\hat{\sigma}_\alpha < \alpha_i < \alpha_j + n\hat{\sigma}$ (using $<$ in a cyclic sense, obviously) ensures that we will only miss very few line segments — e. g. 0.3 % for $n = 3$ or $0.2 \cdot 10^{-6}$ % for $n = 6$, which should be sufficient for most applications. This reduces the average complexity to something nearer to $\mathcal{O}(\sum_{i=1}^{k} N_i \log(N_i)) \approx \mathcal{O}(N \log(N))$, the complexity of sorting the line segments by angle. The worst case complexity will of course remain unchanged, the worst case being represented by, e. g., N segments all belonging to only one line.

In addition to sorting the line segments by angle it can also be beneficial to limit the maximum distance between either the segments' centres or their endpoints. The reason for this is simple: The further away we move from each line-segment's centre, the more tolerant the segment will become with regard to what other line segments will be accepted as a match. Figure 6.1 shows such an erroneous match. The distance from which on such erroneous matches are possible is largely a function of σ_{α_i}, and although the exact limit is, of course, somewhat arbitrary, I found that a value of

$$d \leq \frac{7}{\tan\left(3\sigma_{\alpha_i}\right)} \tag{6.15}$$

(i. e. as long as the 3-σ region does not deviate from the line for more than approximately 7 pxl) works reasonably well. It is, in fact, a good idea not only to limit the maximum distance between segments, but to sort all segments by distance (from the vanishing point, or some given segment), matching closest segments first and only gradually expanding the distance over which we match. Such an approach greatly reduces the risk of false positives, and in practice allows an approach where two collinear line segments are immediately replaced by a single new line segment, further reducing the number of comparisons required. Figure 6.2 shows an example where segments belong to two close, parallel lines. If segments further away are compared first, then there is a chance that segments from two different lines are getting merged; all the segments in between would then not fit anymore. Fitting neighbouring segments first greatly reduces this risk.

There is an additional problem with line segments near the horizon (or, indeed, any line connecting two vanishing points), which can not usually be assigned to just one vanishing point, compare Figure 6.3 on this point. In cases like the one described in the next section, were it is assumed that the maximum extent of a merged line segment is also the maximum extent of the underlying face, it is

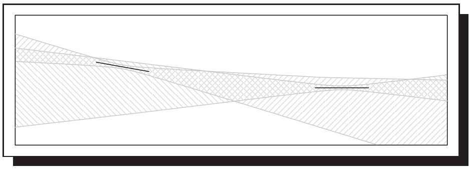


Figure 6.1: *Beyond a certain distance line segments will often match erroneously.*

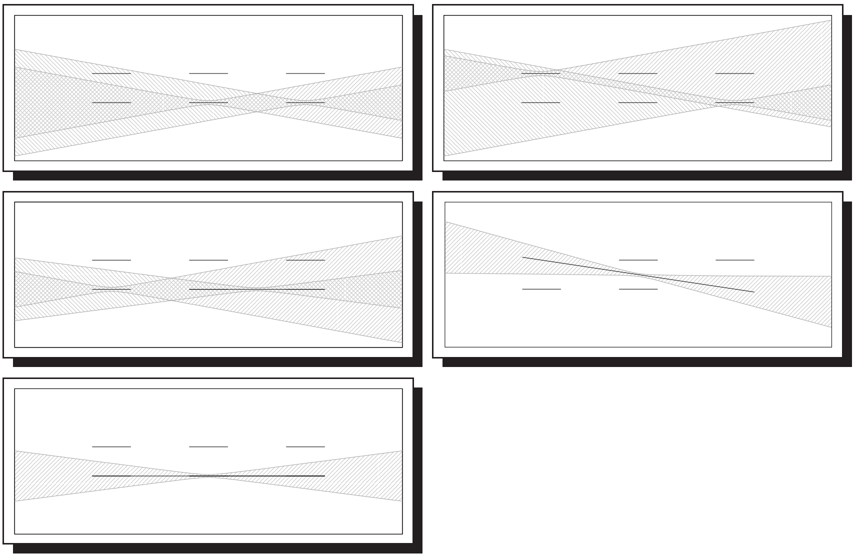


Figure 6.2: *Line segments should be ordered by distance (left column) prior to comparison.*

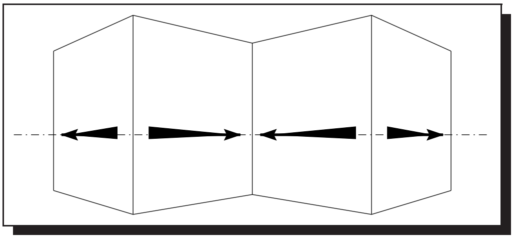


Figure 6.3: *Unclear assignment of a line segment to more than one vanishing point. Note how segments on the same line might belong to either the right or left vanishing point.*

important to avoid such wrong merges. The approach which I use is to only merge line segments which can not be assigned to any other vanishing point (with a certain probability, compare (6.1) — I would usually choose a significance level around or below $p = 1\,\%$). In addition to Section 6.3.4 below, which uses merged line segments to define faces of a building, I will revisit the merging-process in Section 6.4.2.2, where I will discuss the effects of different error models in more detail.

6.3.4 Rectangular Areas

The information about the lines calculated in the previous section can be used to identify rectangular areas. The approach presented here uses the maximum extent of two crossing sets of collinear line segments belonging to different vanishing points and was, in a similar form, but without the additional information about the vanishing points, first presented by us in [54]. This approach relies heavily on Features 4 and 5, i. e. collinear vertical and horizontal structure within a face, usually from windows and doors. This structure will result in a high number of collinear short line segments, which can be considered to be part of a set of imaginary longer lines. Joining these segments, both in the vertical as well as horizontal direction, will lead to a mesh of crossing lines. A face can then be defined as the maximum extent of this mesh bounded by a quadrilateral corresponding to the two vanishing points (called here the smallest bounding rectangle).

The algorithm itself is simple and consists of three steps, namely the identification and assembly of collinear line segments into longer segments as described in Section 6.3.3, the identification of intersecting line segments and their bounding box,

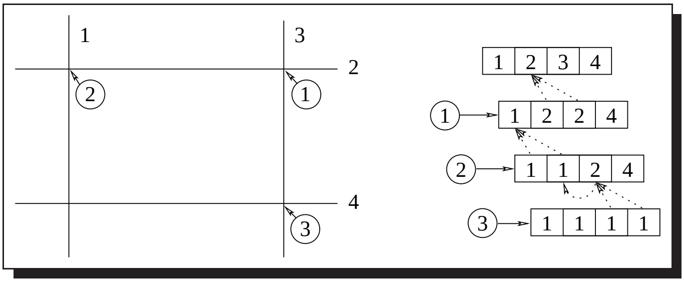


Figure 6.4: *Labelling line segments within a mesh can require recursive look-up of labels.*

and finally a process which will merge overlapping bounding boxes with the same set of vanishing points (if desired). Each step will be outlined below.

Once the individual line segments have been merged into longer segments, we can calculate the intersections between any two line segments from two different vanishing points in a straightforward manner, the algorithmic complexity is in the order of $\mathcal{O}(\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} N_i N_j) \approx \mathcal{O}(k^2 N^2)$, where k is the number of vanishing points, N_i the number of line segments belonging to the i^{th} vanishing point, and N the overall number of line segments. One then needs to classify the intersections into internal ones (i. e. the point of intersection is within both line segments) and external ones (the point of intersection is only within at most 1 segment). This is the actual approach used in the examples given in Section 6.5. It is however possible to reduce the algorithmic complexity to something like $\mathcal{O}(\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} N_i + N_j) \approx \mathcal{O}(k^2 N)$ if we instead plot the individual line segments into an image and only calculate intersections if the new segment is passing through a pixel which is already occupied by another segment. Some additional bookkeeping is needed to detect cases where more than 2 line segments pass through the same pixel, but this is easily incorporated.

Once all internal intersections have been found, we need to identify all line segments within a single mesh and the mesh's bounding box. This can either be done following the calculation of the intersections, or concurrently with it. In both cases the approach is as follows: Each line segment is assigned a unique number, and a table mapping from the segment's original to its current number is created, where at the beginning original and current number are identical. Every time two line segments, potentially belonging to two so far separate clusters, have been classified as intersecting each other, we assign both *clusters* the lower of the two numbers

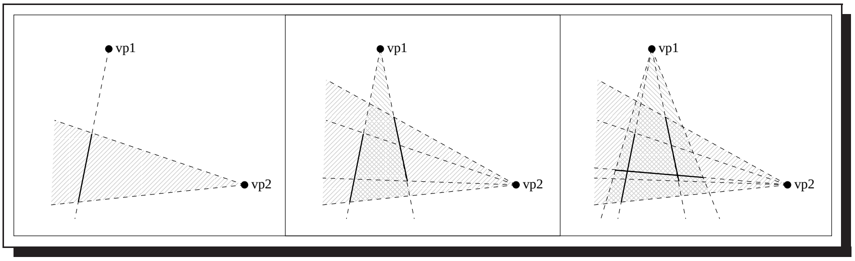


Figure 6.5: *Calculating the bounding box as the maximum extent of angles.*

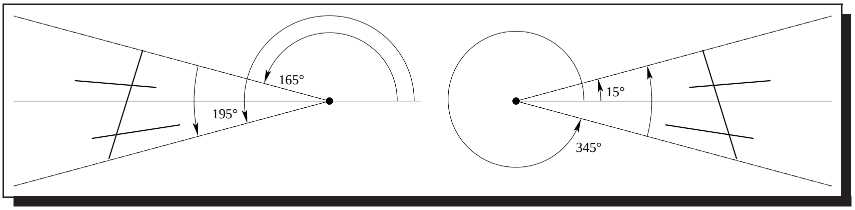


Figure 6.6: *Which of two angles is the minimum and which the maximum depends on the position of the vanishing point relative to the segments.*

— note that this will usually require a recursive identification of current numbers, as some previous intersection might have lowered the number of a cluster without, so far, affecting the numbers stored for all of it's segments. A very simple example is given in Figure 6.4: merging Segments 1 and 2 in Step 2 implicitly also changes the number of Segment 3, which was merged with Segment 2 in the previous step. However, only when Segments 3 and 4 are merged in Step 3 is this change noticed. In general it will be necessary to add a final resolution step once all intersections have been processed, in this step all current numbers are getting updated, starting from the lowest to the highest segment-number — this guarantees, together with the rule that a new cluster is always assigned the lower number of the two intersecting segments, that a unique and consistent result can be reached. The algorithmic complexity of the entire process is linear in the number of intersections and therefore usually quadratic in the number of segments.

The final resolution step mentioned above can also be combined with the calculation of the final bounding boxes around each cluster. For each new cluster found when consolidating the cluster-numbers we calculate the bounding box as the minimum and maximum angle with respect to the two vanishing points each. As each cluster originally consists of only one line segment this is always a region with variations in only one direction. As we identify additional segments belonging to the same cluster, we can simply calculate the new bounding box as the minimum and maximum of the delimiting angles of the two boxes being merged. This process is exemplarily illustrated in Figure 6.5. The minimum and maximum must, of course, be calculated in a fashion suitable for angles, the numerically bigger angle can in fact be the smaller one for our purpose — but need not be — depending on where the vanishing point is relative to the segments, compare Figure 6.6, where $\min_\circ(15°, 345°) = 345°$, but $\min_\circ(165°, 195°) = 165°$.

Each bounding box, once found as described above, is then assumed to be completely inscribed into one face of, e. g., a building. The approach described above has the advantage that it is relatively robust with respect to occlusions and missing lines, as long as the face in question is sufficiently highly structured (and the occlusion isn't). It is therefore more applicable to high-rises or apartment blocks than to suburban one-family houses. Additional disadvantages are the algorithm's inability to differentiate effectively between a long building and a row of identically structured and aligned single buildings (other than by an arbitrarily chosen parameter to describe what constitutes a gap), and that the algorithm is only applicable to rectangular (convex) areas. It never the less performs quite well even in its very limited form outlined above, and Section 6.5 shows some examples of detected buildings, highlighting both its strengths as well as its weaknesses.

6.4 Verification

This section takes a closer look at how well some of the theoretically derived approaches described previously work for actual data. I start off with Section 6.4.1, which takes a look at several different error models for the representation of line segments in 2D. There I use the example of merging line segments as discussed in Section 6.3.3, and we will see that the choice of error model critically influences the accuracy with which we can detect collinear line segments. We then have a look at two different variables which are meant as a (rough) model for errors in 3D in Section 6.4.2, namely an additional positional or angular error-term as discussed in Section 6.3.1.1. In Section 6.4.2.1 this is done for the task of assigning the individual line segments to vanishing points, and we will see that for this application an additional angular term is the most appropriate model, while for the detection of collinear line segments discussed in Section 6.4.2.2 the additional term for the positional error is the more influential one.

6.4.1 Error Models for 2D

In the following I will have a closer look at identifying collinear line segments. Since I stated in constraint 4 on page 149 that most structure in the (modelled) world is either horizontal or vertical I can try to detect horizontal or vertical structure in the image by first merging collinear line segments and later finding the area were line segments from two different directions (vanishing points) intersect — I described this in Section 6.3.4 and we used this in, e. g.,[54]. In order to use such an approach I need to be able to find as many good line-continuation as possible without finding too many bad ones. I therefore need an error measure with a particularly high discriminative power. In Section 6.4.1.1 I will introduce a number of different error measures known from the literature and compare their respective performance with each other and the $(\alpha, \bar{x}, \bar{y})^\mathsf{T}$ parameterisation which I introduced in Section 4.6.2. How this is done I will describe in Section 6.4.1.2, and I will discuss the outcome, which clearly shows the superiority of the $(\alpha, \bar{x}, \bar{y})^\mathsf{T}$ parameterisation over all other models, in Section 6.4.1.3.

6.4.1.1 Error Measures

Over the years a number of authors have discussed possible error measures for the identification of line-continuation. Often this was done in the context of (erroneous) edge-extraction, where gaps in the extracted contour — often due to junctions in the original image — need to be bridged.[4] The first such method which I will use here was described by Coelho et al. in 1990 [30]. There he simply calculated the maximum orthographic distance between each segment's endpoints and the line through the other segment, together with a limit on the maximum distance between the two closest endpoints of the two segments. The orientation (angle) of the lines is not considered at all. If by $(x_{i,j}, y_{i,j})^\mathsf{T}$ we mean the j^{th} endpoint of the i^{th} line, and by $(a_i, b_i, c_i)^\mathsf{T}$ the normal form of the i^{th} line, this can be written as in (6.16).

The next error measure described here was theoretically derived by Imiya in 1996, originally for lines in 3D [63]. For lines in 2D his measure simplifies to the angular distance between the two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, which can be calculated as $\alpha_{1,2} = \arccos\left(\frac{\mathbf{x}_1^\mathsf{T}\mathbf{x}_2}{\|\mathbf{x}_1\|\,\|\mathbf{x}_2\|}\right)$. As this is of course dependent on the particular parameterisation chosen I use for the comparison both the customary $(a, b, c)^\mathsf{T}$ parameterisation as well as Kanatani's N-vectors. As there is a one-to-one mapping

[4] The correct solution in such cases is of course to improve the underlying lower-level algorithms, as was done by us [19] as well as other authors [80] — but here we are discussing the case where the underlying structure might be purely logical and hence not visible within the image.

$$(\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2)^\mathsf{T} \mathbf{J}^\mathsf{T} \boldsymbol{\Sigma}_{\boldsymbol{\ell}_1}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{\ell}} \boldsymbol{\Sigma}_{\boldsymbol{\ell}_2}^{-1} \mathbf{J} (\boldsymbol{\ell}_1 - \boldsymbol{\ell}_2) \leq \chi^2_{p,2} \qquad \text{Section 4.6.2} \qquad (4.63)$$

$$\max_{i,j=1}^{2} \left(x_{(i+1) \bmod 2,j} a_i + y_{(i+1) \bmod 2,j} b_i + c_i\right) \leq T \qquad \text{Coelho et al. [30]} \qquad (6.16)$$

$$\frac{\mathbf{x}_1^\mathsf{T}\mathbf{x}_2\mathbf{x}_2^\mathsf{T}\mathbf{x}_1}{\mathbf{x}_1^\mathsf{T}\mathbf{x}_1\mathbf{x}_2^\mathsf{T}\mathbf{x}_2} \leq T \qquad \text{Imiya [63]} \qquad (6.17)$$

$$\mathbf{x}_2^\mathsf{T}\mathbf{x}_1 \left(\mathbf{x}_2^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{x}_1}\mathbf{x}_2 + \mathbf{x}_1^\mathsf{T}\boldsymbol{\Sigma}_{\mathbf{x}_2}\mathbf{x}_1\right)^- \mathbf{x}_1^\mathsf{T}\mathbf{x}_2 \leq \chi^2_{p,1} \qquad (6.18)$$

$$\frac{\mathbf{x}_2^\mathsf{T}\mathbf{S}(\mathbf{x}_1)^\mathsf{T}\mathbf{S}(\mathbf{x}_1)\mathbf{x}_2}{\mathbf{x}_1^\mathsf{T}\mathbf{x}_1\mathbf{x}_2^\mathsf{T}\mathbf{x}_2} \leq T \qquad (6.19)$$

$$\begin{aligned} \mathbf{x}_2^\mathsf{T}\mathbf{S}(\mathbf{x}_1)^\mathsf{T}\Big(& \mathbf{S}(\mathbf{x}_1)\boldsymbol{\Sigma}_{\mathbf{x}_2}\mathbf{S}(\mathbf{x}_1)^\mathsf{T} \\ & + \mathbf{S}(\mathbf{x}_2)\boldsymbol{\Sigma}_{\mathbf{x}_1}\mathbf{S}(\mathbf{x}_2)^\mathsf{T}\Big)^- \mathbf{S}(\mathbf{x}_1)\mathbf{x}_2 \leq \chi^2_{p,2} \qquad \text{Förstner [49]} \qquad (6.20) \end{aligned}$$

$$(\mathbf{x}_1 - \mathbf{x}_2)^\mathsf{T} (\mathbf{x}_1 - \mathbf{x}_2) \leq T \qquad (6.21)$$

$$(\mathbf{x}_1 - \mathbf{x}_2)^\mathsf{T} (\boldsymbol{\Sigma}_{\mathbf{x}_1} + \boldsymbol{\Sigma}_{\mathbf{x}_2})^- (\mathbf{x}_1 - \mathbf{x}_2) \leq \chi^2_{p,2} \qquad \text{Förstner [49]} \qquad (6.22)$$

Figure 6.7: *Error measures compared in this section.*

between the angle $\alpha_{1,2}$ and the cosine (at least for the range of angles in question, $0 \leq \alpha_{1,2} \leq \pi/2$) it is of course possible to use the cosine or even the squared value of the cosine instead of the angle with exactly the same results, and this is what I'll do in the following. I will calculate this value both with and without error propagation, resulting in 4 possible error measures based on the scalar-product of two vectors: I use directly the scalar-product $\cos\alpha$, based on either the $(a, b, c)^\mathsf{T}$ parameterisation or Kanatani's N-vectors, and do so with (6.18) or without (6.17) error propagation.

A seemingly similar measure could be based on the cross-product of the two (normalised) vectors, rather than on the scalar-product; the result would be a vector whose length is a function $\sin\alpha$ of the angular distance α between the two duals — ideally the resulting vector should be the $\mathbf{0}$-vector. This measure can again be calculated based on either the $(a, b, c)^\mathsf{T}$ parameterisation or Kanatani's N-vectors, and with (6.20) or without (6.19) error propagation, resulting in another 4 possible measures. The version using error propagation has recently been proposed by Förstner [49].

Finally, I can also directly compare the parameterisations of two lines based on the two line segments. In the following I'll do so for, again, the $(a, b, c)^\mathsf{T}$ parameterisation and Kanatani's N-vectors, and do so both with and without error propagation. Of course we need to normalise both parameterisations for a direct comparison, as both parameterisations use 3 parameters to describe a 2 degree of freedom object (a line) — compare Section 4.6.3. I use either $a^2+b^2 = 1$ or $\|\mathbf{x}\| = 1$ as constraints. All in all I started off comparing the 14[5] partly redundant error measures given in Figure 6.7, where $\boldsymbol{\ell}_i$ is the line in $(\alpha, \bar{x}, \bar{y})^\mathsf{T}$-parameterisation as described in Section 4.6.2, $(a_i, b_i, c_i)^\mathsf{T}$ is a line's normal parameterisation with $a_i^2 + b_i^2 = 1$, $(x_{i,j}, y_{i,j})$ is the j^{th} endpoint of the i^{th} line segment, $\mathbf{x}_i$ is a line in either $(a_i, b_i, c_i)^\mathsf{T}$ parameterisation with $a_i^2+b_i^2 = 1$ or one of Kanatani's N-vectors with $\|\mathbf{x}_i\| = 1$, $\boldsymbol{\Sigma}_{\mathbf{x}_i}$ its covariance matrix, and $\mathbf{S}(\mathbf{x}_i)$ is the vector's skew-symmetric matrix used to calculate the cross-product, it is:

$$\mathbf{S}(\mathbf{x}_i) = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix} . \tag{6.23}$$

The next section describes the experimental setup on which these 14 different error measures were tested.

6.4.1.2 Experimental Setup

The experimental setup consists of 4 lines split into 5 segments each, i. e. 20 segments altogether, as can be seen in Figure 6.8 (left). These line segments are projectively disturbed by 15 different projective transformations

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_x & p_y & 1 \end{pmatrix} \tag{6.24}$$

with $p_x \in \{0, 1/10000, 1/1000\}$ and $p_y \in \{0, 1/10000, 1/1000, 1/300, 1/100\}$ resulting in deformations up to the one in Figure 6.8 (right). To each segments orientation and position random Gaussian noise is then added according to the segment's length and Equations (4.21) and (4.23), and the distance between each segment and all other segments is then calculated using each of the 14 error measures described in Section 6.4.1.1 above. This was done 1000 times for each of the 15 projective deformations and each of the 40 known good and 150 known bad pairings, resulting in 39900000 different computed values. For each error measure I have then plotted the receiver-operator-characteristic curve (ROC), which is the percentage of true

[5]Equations (4.63) and (6.16) result in one error measure each, while the 6 Equations (6.17)–(6.22) each produce two measures.

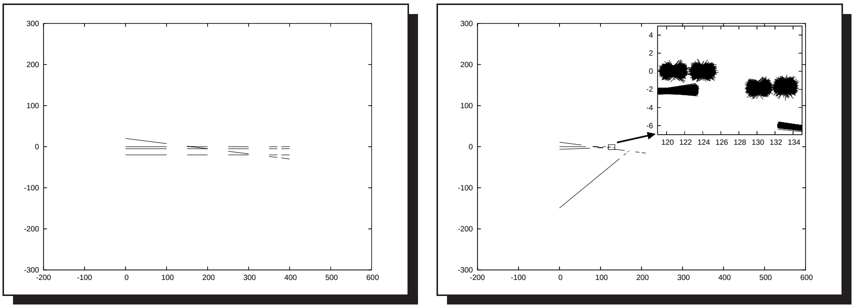


Figure 6.8: *Experimental setup containing collinear line segments (left) and maximally distorted set (right). The inset detail shows how widely the smallest line segments vary in both position and orientation; line segments that small would not be used in practical applications but rather help to analyse border-line conditions.*

positives over the percentage of false positives. These curves will be discussed in the next section.

6.4.1.3 Results

Figure 6.9 shows receiver-operator-characteristic curves (ROC-curves) for all 14 error measures when run on all 15 projectively distorted sets of line segments. Ideally these curves would coincide with the left and top side of the box (i. e. going from $(0, 0)$ straight to $(0, 100)$ and from there to $(100, 100)$). However, in almost all practical applications the ROC-curves will deviate from this ideal form to a bigger or lesser extent, and this deviation can be measured by the area above the curve (aac[6]) — an aac of 50 % (a diagonal curve) would be due to pure chance.

So what can be seen from Figure 6.9? We would expect the methods which use error propagation (the "alpha-xy" measure, and all "weighted" measures) to perform better than the ones without error propagation — if error propagation really were linear, we would in fact expect all methods which use error propagation to perform exactly identical. The superior performance of methods which use error propagation is in fact partly born out by Figure 6.9, with the exception of the two

[6]Often the inverse of the curve is plotted and the area under curve, auc, is used. However, I find the direction used here more natural.

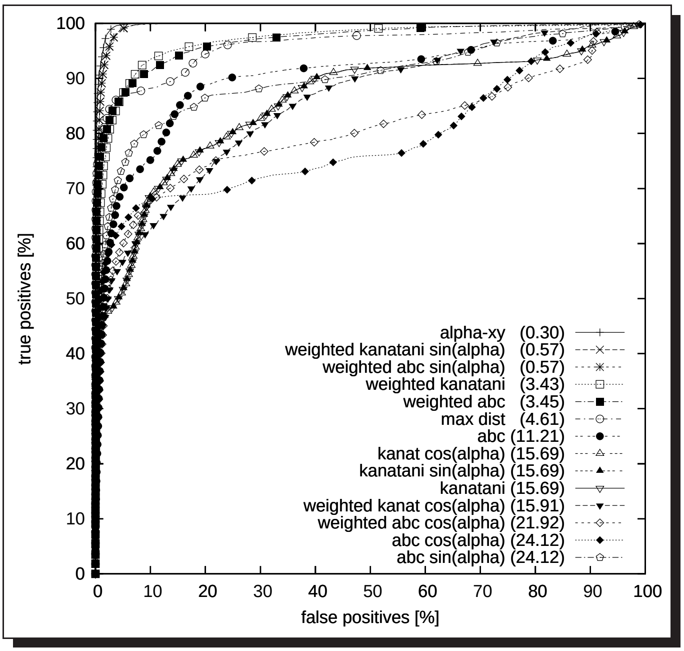


Figure 6.9: *ROC-curve for all 14 error measures and all 15 projective distortions. The area above the curve (aac) in percent is given in parentheses, the smaller the area, the better is the performance of the algorithm.*

methods based on the scalar-product (denoted "weighted cos(alpha)") — this will be explained below.

The first thing I would like to discuss is, however, the fact that the 14 different measures result in only 10 distinguishable curves in Figure 6.9. The first two curves which nearly coincide are the "weighted sin(alpha)" ones, compare Equation (6.20). Despite a considerable difference in the underlying parameterisation — $(a, b, c)^\mathsf{T}$ versus Kanatani's N-vectors — there is virtually no difference in the measures' performance, and this is clearly due to the proper use of error propagation, which almost completely egalised the difference in scaling of the 3rd component.

The next set of curves with nearly identical performance are the ones labelled "kanatani cos(alpha)" (6.17), "kanatani sin(alpha)" (6.19), and finally "kanatani" (6.21). This might seem surprising at first, but is easily explained: the two N-vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ are points on the surface of a unit sphere which are quite close together — the maximum distance between the two points for the undistorted set of line segments was $\|\mathbf{x}_1 - \mathbf{x}_2\| = 0.065$, and for differences that small it is, up to a very good approximation, $\|\mathbf{x}_1 - \mathbf{x}_2\| = \alpha$, which explains why all measures based on the angular difference of two N-vectors as well as their direct distance would perform equally well.

The last set of curves with nearly identical performance are the ones labelled "abc cos(alpha)" (6.17) and "abc sin(alpha)" (6.19); the reason is once more that both are functions of the same angular difference α. The interesting question here is why the curve labelled "abc" (6.21) behaves differently; this is mostly due to the fact that the calculation of the direct distance uses a different, non-spherical normalisation (namely $a^2 + b^2 = 1$), but also that even for the undistorted set any angles up to $\pi/2$ can be observed. The latter, together with the observation that the majority of line segments will be normalised to approximately $(0, 0, 1)^\mathsf{T}$, also explains those measures' particularly bad discriminating power, which makes them perform poorest out of this particular set.

In the following I will summarise the three groups as "weighted sin(alpha)", "abc alpha", and "kanatani alpha" and talk of only 10 different groups.

So if error propagation is such a good thing — and it certainly is for this reason that the first 4 out of 10 (5 out of 14) best-performing measures all use error propagation — then how come that the "weighted kanat cos(alpha)" and "weighted abc cos(alpha)" measures perform so poorly? The "kanatani cos(alpha)" version in fact performs better than the version using error propagation! The reason is that for these measures a scalar representation of what is essentially a 2 DOF distance is calculated first, and only then is error propagation applied. However, one of the reasons error propagation is needed here is that the directional information (coded

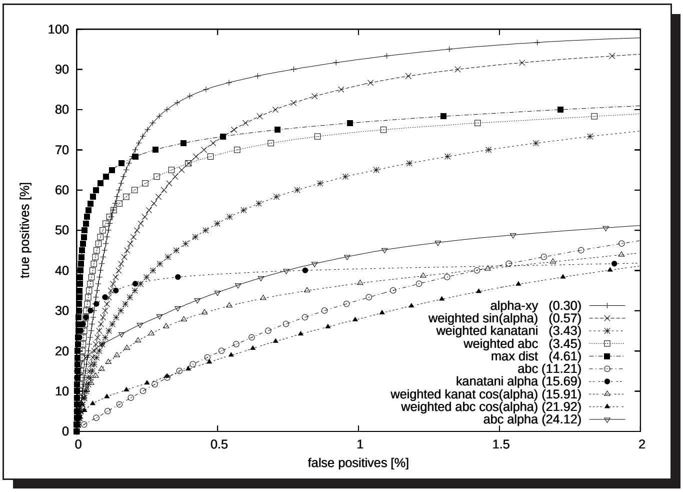


Figure 6.10: *Enlarged detail of Figure 6.9*

in the first two elements of the vector $\mathbf{x}$) has a variance completely different from the positional information (coded in the third element). The scaling underlying Kanatani's N-vectors accounts for some of this difference, but since the scaling needed is ultimately a function of the line-segment's length there is no one single scaling which could completely balance out the different variances for all segments; any algorithm which calculates an error measure first and a scaling later is therefore doomed to failure.

One other thing that strikes the eye is that all measures which use proper error propagation exhibit an essentially concave curvature, while for all other error-measures the ROC-curves show two or more inflection points. The former suggests that all error-conditions can indeed be caught equally well with just one threshold, while the latter is du to an overlap of several curves with different thresholds for different error-conditions.

The last point to note is the performance of the "max dist" (6.16) error measure, the very simple one proposed by Coelho et al. [30] in 1990, which simply uses the maximum orthographic distance between either line-segment's endpoints and the other line. At least for our test-set this measure performs nearly as well as the direct distance using error propagation, and up to a true-positive rate of about 70 % (and a false negative rate of only 0.15 %) actually performs better than any

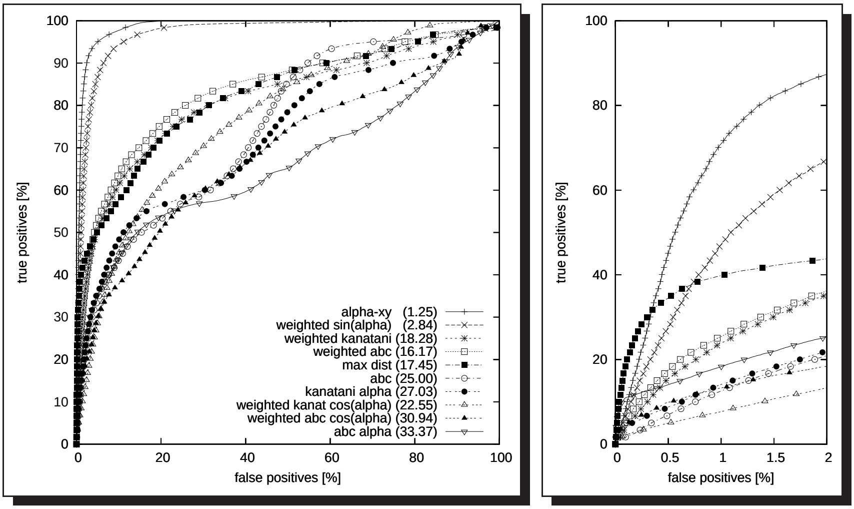


Figure 6.11: *ROC-curves and enlarged detail for the test-set with the strongest projective distortion.*

other measure, see Figure 6.10. It should, however, be noted that this is to some extent at least a feature of our test-set rather than the algorithm. But even for the set with the strongest projective distortion the “max dist” algorithm still performs best up to a true-positive rate of about 30 % (false negative rate 0.33 %), see Figure 6.11.

So which algorithm should one use? For most applications the answer really is quite simple: only the $(\alpha, \bar{x}, \bar{y})^{\mathsf{T}}$-measure proposed in Section 4.6.2 and the cross-ratio based measure proposed by Förstner [49] perform consistently well over a wide range of errors. In my tests the $(\alpha, \bar{x}, \bar{y})^{\mathsf{T}}$-measure actually performs slightly better than the cross-ratio based measures, but it should be noted that my test-setup actually favours the $(\alpha, \bar{x}, \bar{y})^{\mathsf{T}}$-measure, since in my opinion it captures the actual errors most faithfully — virtually all line-fitting algorithms calculate a segment’s centre point and angle, and this is where the errors are. The simple distance measure proposed by Coelho et al. [30] can be an alternative only if speed is of the essence and it is sufficient to find only a fraction of the true positives.

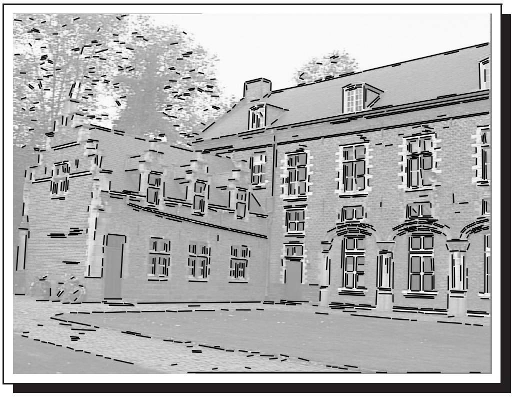


Figure 6.12: *Image of Leuven-Kasteel and straight line segments used in this section, used with permission of Katholieke Universiteit Leuven, Robot Vision Group (ESAT-PSI).*

6.4.2 Error Models for 3D

In the previous section we have seen a comparison of different error models that all describe errors in the image measurements themselves. However, when dealing with real-life buildings objects there are also, as a rule, errors in 3D which we need to deal with. This section outlines two possible ways to capture those errors and, for a given image, demonstrates under which condition which model is most adequate. In Section 6.4.2.1 I use the example of assigning line segments to vanishing points, and we will see that errors in 3D are best modelled by an additional directional variance, while Section 6.4.2.2 deals with line-continuation, where the influence of errors in 3D is dominated by an additional positional uncertainty.

6.4.2.1 Assigning Line Segments to Vanishing Points

Assigning individual line segments to vanishing points allows me to highlight the effect of different approaches to model the uncertainty in the underlying 3D-model, as mentioned in Sections 4 and 6.3.1. There exist essentially 3 different methods to introduce the model's uncertainty into the relevant equation, (6.1): we can add a third term σ_{model} to the denominator, we can add additional terms to the individual elements of the vanishing point's covariance matrix $\mathbf{\Sigma_p}$, or to the line's covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\ell}}$, as was suggested in Section 6.3.1.2. Here, I will only have

${\sigma'_\alpha}^2$	N_{vp1}	N_{vp2}	N_{vp3}	$\sum N_{\mathrm{vp}}$
$1 \cdot 10^{-8} \equiv 0.006°$	200	77	224	501
$1 \cdot 10^{-7} \equiv 0.018°$	200	77	224	501
$1 \cdot 10^{-6} \equiv 0.057°$	203	77	225	505
$1 \cdot 10^{-5} \equiv 0.18°$	214	79	233	526
$1 \cdot 10^{-4} \equiv 0.57°$	246	91	261	598
$1 \cdot 10^{-3} \equiv 1.81°$	308	123	299	730
$1 \cdot 10^{-2} \equiv 5.73°$	395	184	318	897
$1 \cdot 10^{-1} \equiv 18.1°$	454	213	334	1001
$1 \cdot 10^{-0} \equiv 57.3°$	459	219	341	1019

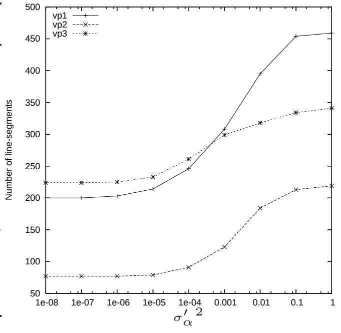


Figure 6.13: *Number of line segments classified as belonging to each of the three vanishing points for different values of* ${\sigma'_\alpha}^2$.

a look at the latter method, since in Section 6.4.2.2 I will then study the effect of the same parameters on finding and merging collinear line segments. I will also simplify the 3D error-model somewhat and use ${\sigma'_y}^2 = {\sigma'_x}^2$. This models a circular uncertainty in the position of the line-segment's centre point, while what would really be needed is an uncertainty only perpendicular to the line segment. However, since any small uncertainty along the direction of the line is actually of little or no consequence this will not change the outcome dramatically.

In order to see the effect of these parameters, I have taken an example image (Figure 6.12) and, by hand, identified the 3 vanishing points. This was done by selecting a number of line segments for each direction and calculating the most likely intersection as described in Sections 4.4.1 and 6.3.1. I then used Equation (6.1) to find all line segments belonging to any of these three vanishing points for different values of ${\sigma'_\alpha}^2$ and ${\sigma'_x}^2$, where "belonging" was rather generously defined as an error below $\chi^2_{5\,\%,1}$. Figure 6.13 gives the number of line segments classified to belong to each of the 3 vanishing points for different values of ${\sigma'_\alpha}^2$. We see that for values below approximately ${\sigma'_\alpha}^2 = 10^{-6}$, corresponding to an angle of about $\alpha = 0.06°$, the number of line segments stays essentially constant. This is due to the fact that in those cases ${\sigma'_\alpha}^2 \ll \sigma^2_\alpha$ and therefore has little influence — σ^2_α is usually in the region of $10^{-6} \leq \sigma^2_\alpha \leq 10^{-5}$ and σ^2_x in the region $\sigma^2_x \approx 10^{-3}$. We also see that for values above approximately ${\sigma'_\alpha}^2 = 10^{-1}$, corresponding to an angle of about $\alpha = 18°$, the number of line segments per vanishing point becomes stationary once again, this is simply due to the fact that for bigger values of ${\sigma'_\alpha}^2$ any line segment, be it as random as it may, can be assigned to at least one vanishing point — and indeed, for ${\sigma'_\alpha}^2 = 1$, corresponding to an angle of about $\alpha = 57°$, all 1019 line

${\sigma'_x}^2$	N_{vp1}	N_{vp2}	N_{vp3}	$\sum N_{\text{vp}}$
$1 \cdot 10^{-2}$	200	77	224	501
$1 \cdot 10^{-1}$	200	77	224	501
1	201	77	224	502
$1 \cdot 10^{1}$	203	77	224	504
$1 \cdot 10^{2}$	220	83	224	527
$1 \cdot 10^{3}$	254	107	227	588
$1 \cdot 10^{4}$	321	156	240	717
$1 \cdot 10^{5}$	397	234	271	902
$1 \cdot 10^{6}$	444	279	296	1019

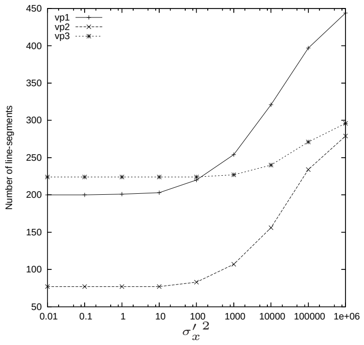
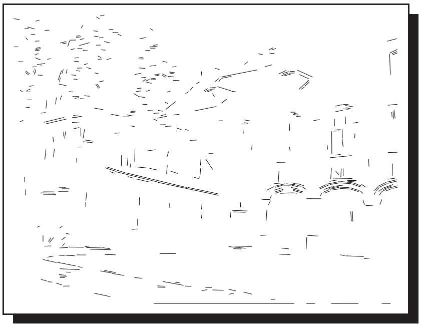


Figure 6.14: *Number of line segments classified as belonging to each of the three vanishing points for different values of ${\sigma'_x}^2$.*

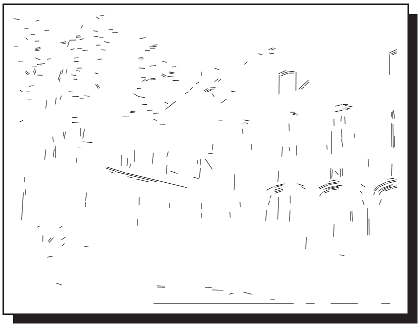
segments have been assigned to one of the three vanishing points.

The influence of ${\sigma'_x}^2$ is, in contrast, negligible. Figure 6.14 shows the analogous plot for different values of ${\sigma'_x}^2$. And although the tendency is of course the same — no influence of ${\sigma'_x}^2$ below a certain value, and from there an increase until all line segments are assigned to one vanishing point — is the interpretation completely different: while for ${\sigma'_\alpha}^2$ an influence was observable as soon as ${\sigma'_\alpha}^2 \approx \sigma_\alpha^2$, we now only see an increase once ${\sigma'_x}^2 > 10^4\sigma_x^2$ — using the same range of *relative* values as before we would hardly observe any increase at all.

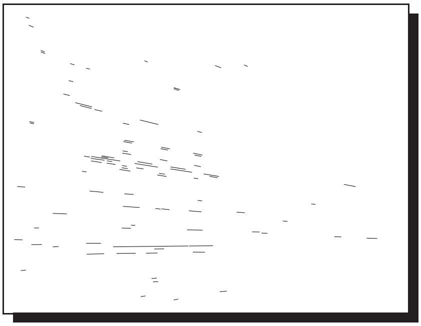
But not only does the effect start later, the result is also less useful. A reasonable value for ${\sigma'_\alpha}^2$ and this particular image, which shows a rather old and not very well aligned building, is ${\sigma'_\alpha}^2 = 10^{-3}$, corresponding to a variance of 1.8° and resulting in 730 assigned line segments — approximately the same number of line segments also get assigned to a vanishing point if we choose ${\sigma'_x}^2 = 10^4$, corresponding to a variance of 100 pxl. Figure 6.15 shows, in its top-row, the unassigned line segments. We notice from Figure 6.15(b) that using ${\sigma'_x}^2 = 10^4$ missed more of the vertical line segments than using ${\sigma'_\alpha}^2 = 10^{-3}$ (see Figure 6.15(a)), while at the same time more of the segments belonging to the arcs above the doors and windows, which come with random orientations, were assigned to one of the vanishing points. The latter is corroborated by Figures 6.15(c) and 6.15(d) which show the line segments belonging to the second vanishing point, which in the case ${\sigma'_x}^2 = 10^4$ clearly contains a number of false classifications.



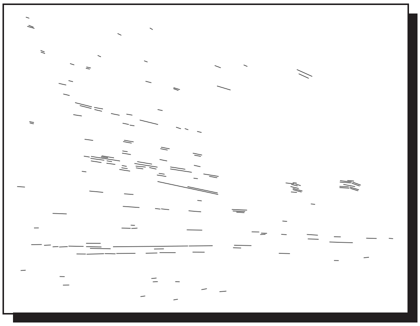
(a) Unassigned segments, ${\sigma'_\alpha}^2 = 10^{-3}$.



(b) Unassigned segments, ${\sigma'_x}^2 = 10^4$



(c) Second vanishing point, ${\sigma'_\alpha}^2 = 10^{-3}$.



(d) Second vanishing point, ${\sigma'_x}^2 = 10^4$

Figure 6.15: *Assigning line segments to vanishing points.*

${\sigma'_\alpha}^2$	$N_{\text{orig.}}$	N_{merged}
$1 \cdot 10^{-9} \equiv 0.002°$	160	71
$1 \cdot 10^{-8} \equiv 0.006°$	160	71
$1 \cdot 10^{-7} \equiv 0.018°$	160	71
$1 \cdot 10^{-6} \equiv 0.057°$	161	72
$1 \cdot 10^{-5} \equiv 0.18°$	161	72
$1 \cdot 10^{-4} \equiv 0.57°$	157	72
$1 \cdot 10^{-3} \equiv 1.81°$	110	51
$1 \cdot 10^{-2} \equiv 5.73°$	0	0
$1 \cdot 10^{-1} \equiv 18.1°$	0	0

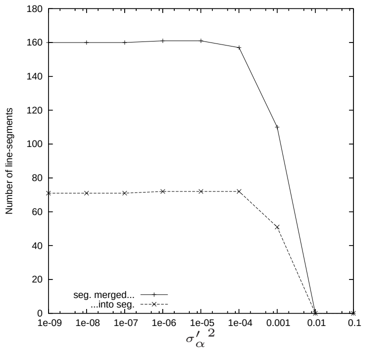


Figure 6.16: *Number of line segments that got merged, and number of line segments they got merged into, for different values of* ${\sigma'_\alpha}^2$.

${\sigma'_x}^2$	$N_{\text{orig.}}$	N_{merged}
$1 \cdot 10^{-5}$	160	71
$1 \cdot 10^{-4}$	160	71
$1 \cdot 10^{-3}$	162	72
$1 \cdot 10^{-2}$	195	88
$1 \cdot 10^{-1}$	294	122
1	411	155
$1 \cdot 10^{1}$	545	180
$1 \cdot 10^{2}$	707	175
$1 \cdot 10^{3}$	844	155
$1 \cdot 10^{4}$	903	113
$1 \cdot 10^{5}$	934	80
$1 \cdot 10^{6}$	938	76

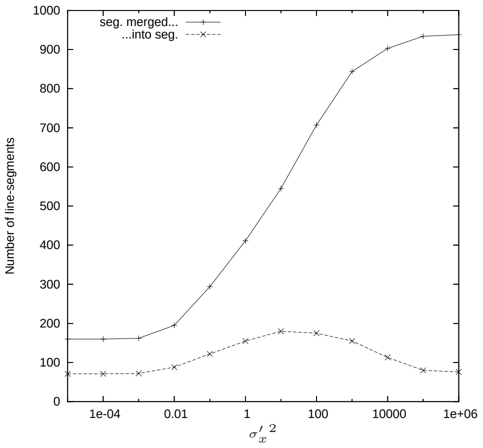


Figure 6.17: *Number of line segments that got merged, and number of line segments they got merged into, for different values of* ${\sigma'_x}^2$.

6.4.2.2 Merging Line Segments

The same tests just run for vanishing points can of course also be run when it comes to merging line segments. Here we would expect ${\sigma'_x}^2$ to be the important contributing factor, and ${\sigma'_\alpha}^2$ to be mostly irrelevant. A casual glance at Fig-

${\sigma'_\alpha}^2$		$N_{\text{orig.}}$	N_{merged}
$1 \cdot 10^{-7} \equiv$	$0.018°$	178	79
$1 \cdot 10^{-6} \equiv$	$0.057°$	179	80
$1 \cdot 10^{-5} \equiv$	$0.18°$	182	81
$1 \cdot 10^{-4} \equiv$	$0.57°$	183	82
$1 \cdot 10^{-3} \equiv$	$1.81°$	225	97
$1 \cdot 10^{-2} \equiv$	$5.73°$	263	116
$1 \cdot 10^{-1} \equiv$	$18.1°$	294	131
$1 \cdot 10^{0} \equiv$	$57.3°$	461	190
$1 \cdot 10^{1} \equiv$	$181.°$	517	207

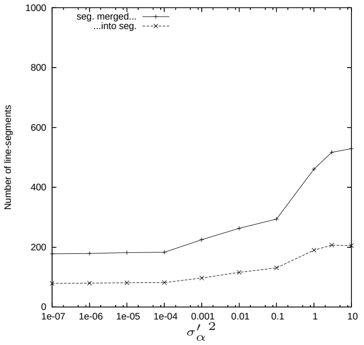
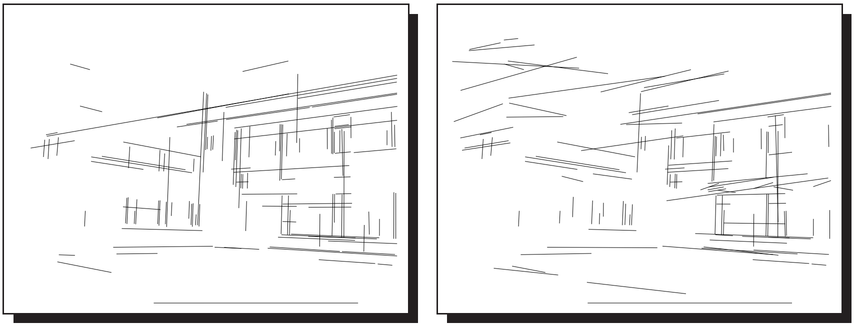


Figure 6.18: *Number of line segments that got merged, and number of line segments they got merged into, for different values of* ${\sigma'_\alpha}^2$ *and a fixed maximum distance between segments of* 500 pxl.

ures 6.16 and 6.17 seems to corroborate this expectation: From Figure 6.17 we see that the influence of ${\sigma'_x}^2$, which only started at around ${\sigma'_x}^2 \approx 10^4\sigma_x^2 \approx 10$ in the last section, now starts at ${\sigma'_x}^2 \approx \sigma_x^2 \approx 10^{-3}$, and it already levels of at around ${\sigma'_x}^2 \approx 10^2$ for the number of final line segments, and ${\sigma'_x}^2 \approx 10^5$ for the number of segments getting merged. The influence of ${\sigma'_\alpha}^2$, by comparison, seems negligible — the number of segments merged is essentially constant for values around ${\sigma'_\alpha}^2 \approx \sigma_\alpha^2$, and then actually levels off to 0.

This latter observation requires some explanation — it is, of course, unreasonable to expect that with increasing ${\sigma'_\alpha}^2$ fewer segments should get matched. The reason for this behaviour is the limitation placed on the maximum distance discussed in Section 6.3.3 and embodied in Equation (6.15), which is a function of the lines effective σ_α^2, which is in turn the sum of the segments true directional variance and the added error-term, ${\sigma'_\alpha}^2 + \sigma_\alpha^2$. For values of ${\sigma'_\alpha}^2 > 10^{-4}$ this term is clearly getting dominated by ${\sigma'_\alpha}^2$; for ${\sigma'_\alpha}^2 = 10^{-4}$ the maximum distance is still 233 pxl, but for ${\sigma'_\alpha}^2 = 10^{-3}$ this already gets reduced to 74 pxl and then to 23 pxl for ${\sigma'_\alpha}^2 = 10^{-2}$ — it is clear, that this constraint considerably limits the potential number of segments to merge.

I have therefore run the same tests with a fixed maximum distance of 500 pxl. For variations of ${\sigma'_x}^2$ the results are virtually the same (slightly more segments get merged, most of them erroneously, due to the longer reach in particular when dealing with smaller, more uncertain, segments), but for ${\sigma'_\alpha}^2$ the results are quite



(a) ${\sigma'_x}^2 = 10^{-1}$, 294 segments merged into 122 **(b)** ${\sigma'_\alpha}^2 = 10^{-2}$, 263 segments merged into 116

Figure 6.19: *Merged line segments generated for* ${\sigma'_x}^2 = 10^{-1}$ *and* ${\sigma'_\alpha}^2 = 10^{-2}$ *respectively. The result for* ${\sigma'_x}^2 = 10^{-1}$ *is clearly much better, although approximately the same number of segments are being generated in both cases.*

different. Looking at Figure 6.18, we see that again the number of line segments merged stays essentially constant for values of ${\sigma'_\alpha}^2 \leq \sigma_\alpha^2$; for bigger values of ${\sigma'_\alpha}^2$ the number of line segments merged increases, but far below the increase which we noted for variations in ${\sigma'_x}^2$, and levels off at much fewer line segments being merged into more new line segments than happens for ${\sigma'_x}^2$.

It is also interesting to look at the results of both algorithms respectively. At ${\sigma'_x}^2 = 10^{-1} \approx 100\sigma_x^2$ and ${\sigma'_\alpha}^2 = 10^{-2} \approx 10^4\sigma_\alpha^2$ we get approximately the same number of merged segments (122 versus 116 new segments, generated from 294 versus 263 original segments). Since ${\sigma'_\alpha}^2 = 10^{-2}$ actually merges fewer segments we would also expect it to make fewer errors, but looking at Figure 6.19 we see that although some erroneous segments were created in both cases (the values for ${\sigma'_x}^2$ and ${\sigma'_\alpha}^2$ were on purpose chosen somewhat too big), the errors are much more pronounced (and much more clearly wrong) for ${\sigma'_\alpha}^2 = 10^{-2}$ in Figure 6.19(b).

The next section shows some results of the combined algorithm.

6.5 Results and Discussion

In this section I show a number of results when running the algorithm described above on a number of different datasets. None of the results are perfect, but many are quite usable despite the algorithm's extremely simple structure; it could be (and, in fact, has been) easily improved by some simple modifications outside the scope of this thesis. Using subjective structures as described by Brillault-O'Mahony in [20, 21] or corner-information could be one such possible extension; in [54] we successfully used the colour in a neighbourhood of the line segment; and one could also combine this approach with colour or texture based region-merging — all this will also be discussed in the outlook in Section 8.2.

Figure 6.20 shows a number of images with mostly correctly identified regions, i. e. the regions found correspond to meaningful structures in the image, although faces with little texture might not be represented correctly and most regions are generally somewhat smaller than the actual face they are meant to represent, as they are generated as the smallest rectangle containing all features. Additional reasoning (or heuristics) would be needed to detect the entire face, but I will discuss in Section 8.2 why I think that this job would be better left to colour- or texture-based algorithms.

Figure 6.21 shows some typical errors. Accidental alignment of line segments as seen in Figure 6.21(a) is maybe the most frequent error; in a town-setting, where usually all lines along the side of the road share a common vanishing point, it is quite difficult to prevent such unwanted accidental alignments of line segments based on geometry alone. In [54] we used colour-information as an additional feature, which worked quite well; other approaches could be based on precomputed region-information, based on colour or texture. Figure 6.21(b) shows another, not quite so common source for errors — here the accidental intersection of segments creates spurious regions which do not correspond to actual faces on the house. In Figure 6.21(c) we can observe the effect of occlusion, where a street-lamp in front of the building is considered as part of the face, resulting in regions that extend far beyond the actual border of the face. We also see that overlapping regions with two different orientations are being found, this is partly correct (the top and bottom portion of the building do have different, non-perpendicular orientations), but due to the occlusion and another accidental alignment between a window-frame and the "A" in the shop-sign both regions overlap. Such overlap can also be observed in Figure 6.21(d), where here it is simply due to the fact that at different heights the faces have a different extent. None of these problems can be solved by geometric constraints alone, as is the topic of this thesis, but an additional region based approach and possibly region- or corner based reasoning could well improve

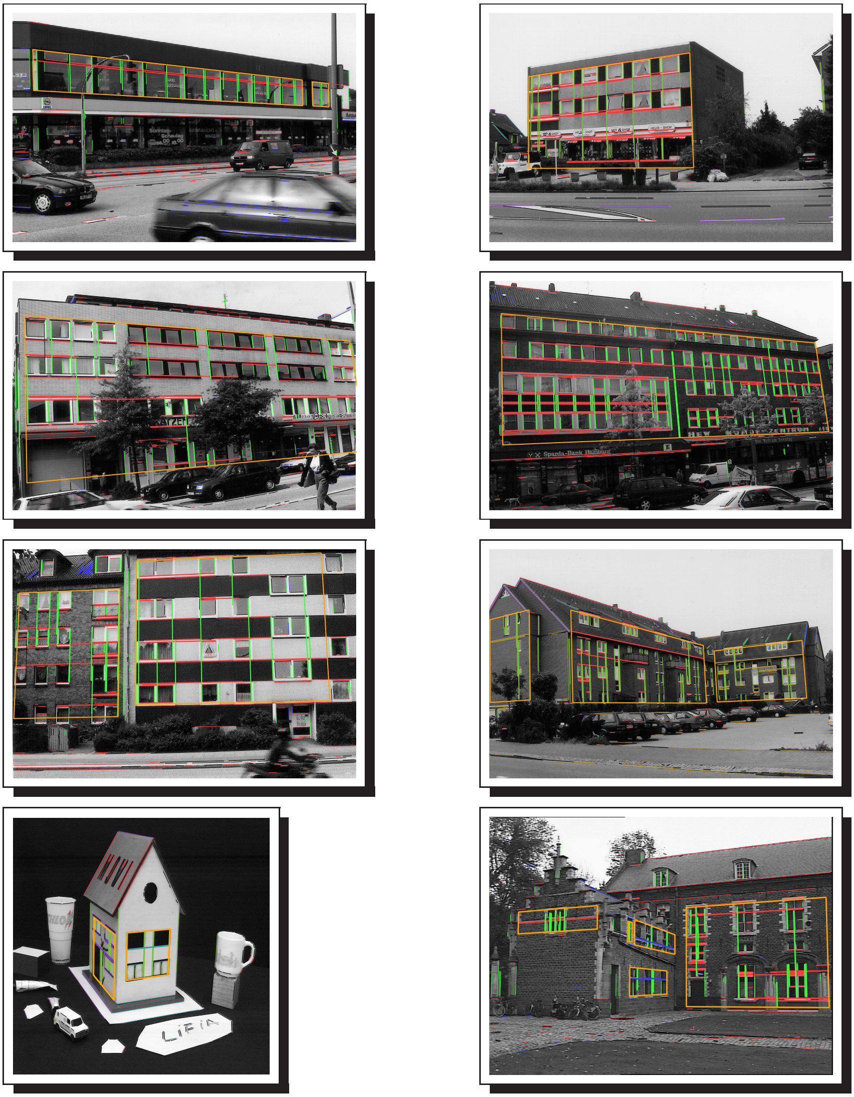
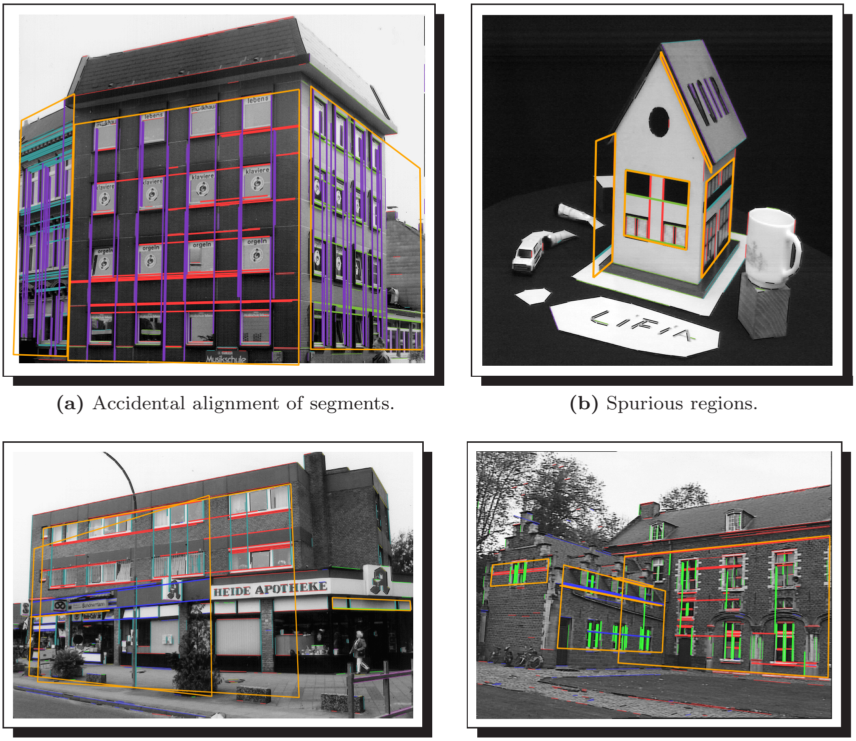


Figure 6.20: *Regions correctly identified in images from different datasets.*



(a) Accidental alignment of segments.

(b) Spurious regions.

(c) Overlapping regions.

(d) Overlapping regions.

Figure 6.21: *Incorrectly identified regions in images from different datasets.*

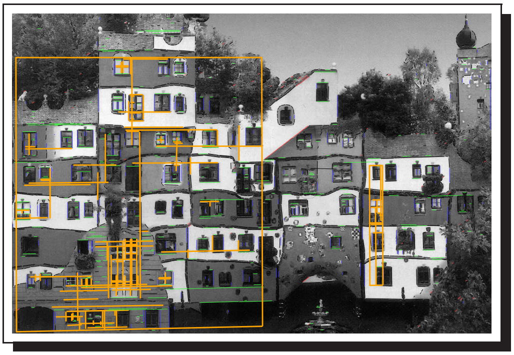


Figure 6.22: *Grouping applied to a house by Hundertwasser. The algorithm reveals that despite its unconventional exterior much of the original structure of the house was retained.*

the situation, see Section 8.2.

Figure 6.22 shows the application of the same algorithm to an image of a house by Hundertwasser. These houses are well known for their lack of structure; however they are essentially remodelled "normal" houses from around 1900 and can not totally deny their heritage — we see that in particular in the vertical alignment of windows enough information is retained to group a considerable portion of the facade — although the poor quality of the image (a scanned in postcard) and the non-orthodoxy of the building made it necessary to hand-tune some parameters (mostly concerned with line-extraction though).

Figure 6.23 finally shows what a purely region-based reconstruction could look like for properly identified regions — the regions for Figure 6.23 were in fact identified by hand (but based on the mechanisms described above, only their extent was slightly altered, most notably to provide a correct ground-line). The reconstruction looks quite reasonable even though no additional calibration of the camera was performed. Note that this is only shown here as an example for what is theoretically possible based on the individual building-blocks provided above; an actual implementation would need additional modules based not just on geometry to be successful.

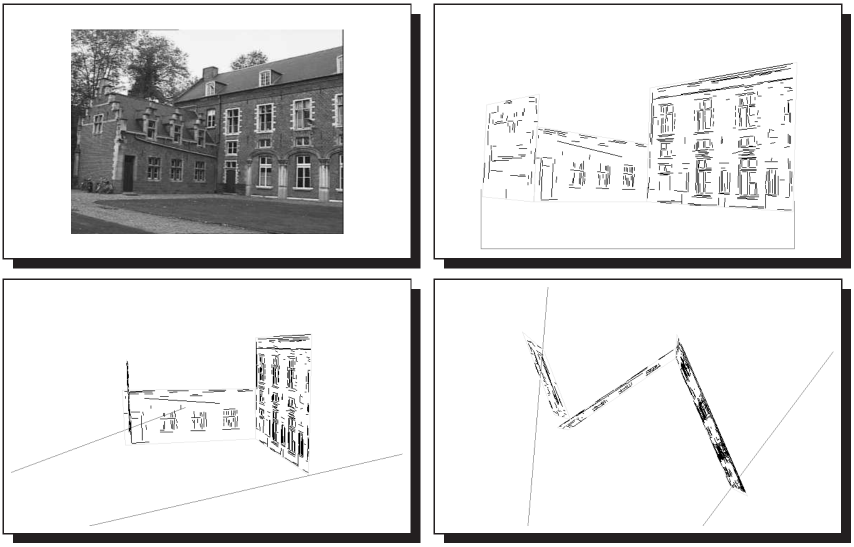


Figure 6.23: *Monocular reconstruction with unknown camera.*

Chapter 7

Detecting Surfaces of Revolution

Revolutions never occur in mathematics.

Michael Crowe, Historia Mathematica

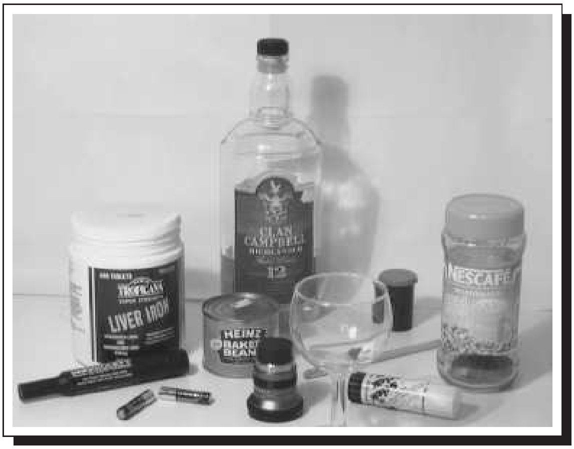


Figure 7.1: *Some of the objects of revolution we encounter each day (as found in the office at 46 Banburry Road, Oxford).*

7.1 Introduction

This section mainly deals with the detection and grouping of surfaces of revolution or SORs, and in particular with the calculation of the SOR's axis. Objects of revolution similar to the ones found in every household are shown in Figure 7.1. Additionally, the algorithms for the calculation of the axis presented here are also suitable for the detection and grouping of planar symmetric objects and even applicable to arbitrary straight homogeneous generalised cylinders (SHGC). This will be pointed out were applicable.

Planar symmetric objects, SORs and SHGCs in general, make up much of the man-made environment which surrounds us, quite possibly surpassed only by orthogonal structures as described in Section 6. It is therefore not surprising that the computer vision community has, over the years, devoted some work towards the recognition of such objects. Consequently, the work described here is based on earlier work not only on the recognition of SORs, but also SHGCs, symmetry-detection and, to a lesser extent, the detection of conics in images. All of these are discussed in more detail below.

The expressiveness of the generalised cylinder representation, where a variable planar cross section is swept along a space curve, the cylinder's axis or spine, and is deformed according to a sweeping rule, has always been of interest to vision researchers. I will however start this overview with the 1989 paper by Ponce et. al., which contains many references to previous work [121]. Prior to his publication, researchers often assumed that the apparent contours on each side of an (SH)GC's axis were related by some sort of qualitative symmetry. Ponce et. al. were the first to show that for a SHGC (and therefore also all more general GCs) no such relation between the two contours exists. However, the same proof also shows that for the more special case of an SOR the two contours are *always* symmetric with respect to the image of the cylinder's axis, or can be projectively transformed into a frame where this is the case. His work was built on by Sato [131–133] and others [58, 153, 163], however, they all assumed parallel projection; Abdallah [10] was the first to present an extension to arbitrary projective views. His work in turn was based on previous work on SORs [165][3–5, 9]. These too, either in their own right [38, 50, 82, 91, 124, 160, 161][8], or as a subclass of SHGCs [57, 58, 104, 118, 121, 131–133, 153, 163], have seen considerable interest. Also of marginal relevance in the context of this chapter is some of the work on symmetry under projection with regard to polyhedral or planar objects [35, 53, 55, 56, 83, 92, 102, 128, 152, 162], which is needed as a prerequisite to the axis calculation, and work on the detection of conics in images [17, 42, 48, 76, 78, 122, 126], on which the reconstruction of SORs (not described here, but e. g. in [8]) builds.

This chapter concentrates on the calculation of an SOR's (projected) axis; knowledge of the axis is central to grouping [4, 9] as well as recognition [50, 91][5] and reconstruction [32, 161][8]. The need to perform many such calculations in grouping makes it advisable to preface the snake-like algorithm which I gave in [9] and which gives excellent results with a faster algorithm which can be used to weed out many unwanted contour-pairings and provide the initialisation for the snake-like algorithm. I therefore compare the performance of a number of established algorithms on a number of different features and demonstrate that the most popular algorithm, total least squares of Euclidean distances, is also the most error-prone and essentially unusable for this application. These comparisons are done on real contour-data derived from real images which previously appeared in publications about the grouping and recognition of SORs.

The remainder of this chapter is structured as follows: in Section 7.2 I describe both the object and camera model used. In Section 7.3 I present my algorithm for the grouping and recognition of SORs with the detection of the SOR's axis of symmetry as its main component. This is described in more detail in Section 7.4, where I describe the different algorithms and feature-sets which can be used for the calculation of the axis. There a comparison of the algorithms' absolute (Section 7.4.4.1) and relative (Section 7.4.4.2) performance is given, and I will demonstrate that

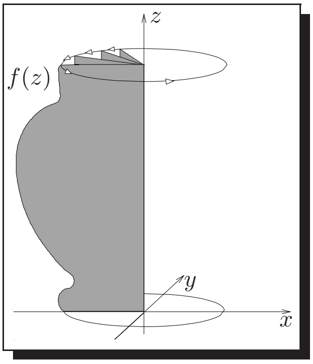


__Figure 7.2:__ The sweeping-rule or generating curve $f(z)$.

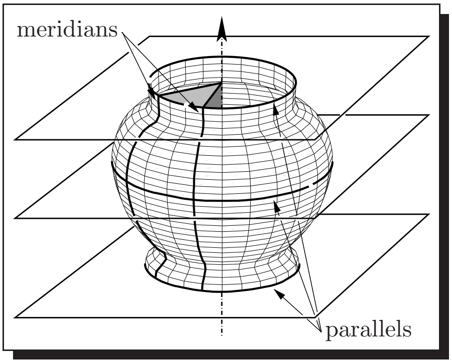


__Figure 7.3:__ Parallels and meridians.

the most widely used algorithm is at the same time the least reliable, and recommend much better alternatives instead. Section 7.5 discusses some of the more noteworthy observations made in the previous section, and Section 7.6 summarises the main observations from the previous sections. In Section 7.7 finally I give a short description on how the results discussed so far are applicable to objects with a planar symmetric contour generator and SHGCs.

7.2 Model

In Section 7.2.1 I will discuss the underlying 3D-model of an SOR, followed by a discussion of the different camera models in Sections 7.2.2 ff. In how far the comparisons given here are also applicable to SHGCs and planar symmetric objects is discussed in Section 7.7.

7.2.1 3D Model

There are two traditional models for the construction of Surfaces of Revolution. Most commonly used is that of a generating function $f(z)$ being rotated around the axis of revolution, resulting in a surface

$$\vec{S} = (f(z)\cos(\varphi), f(z)\sin(\varphi), z)^{\mathsf{T}} , \tag{7.1}$$

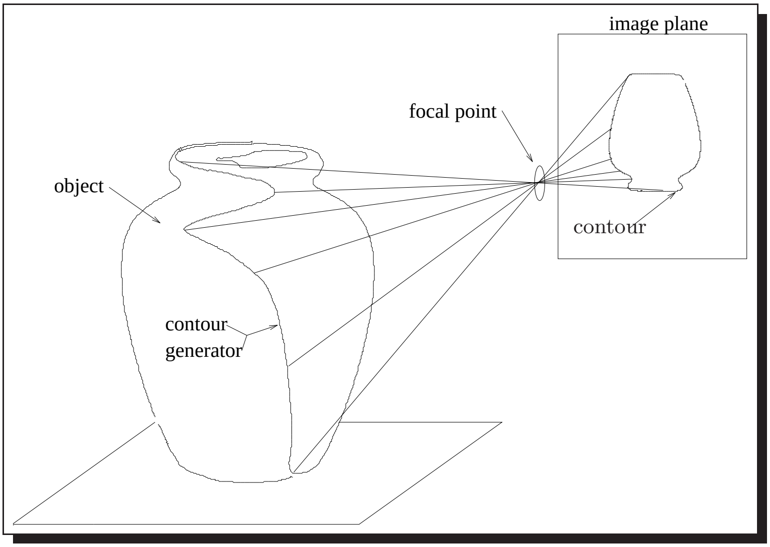


Figure 7.4: *Contour and contour-generator of an SOR. Note that the contour generator is a space curve on the surface of the SOR.*

compare Fig. 7.2. My intentions, however, are better served if we understand an SOR as a special case of a Straight Homogeneous Generalised Cylinder. A SHGC can be constructed by sweeping a cross section of arbitrary (planar) shape along a straight axis and scaling it according to to a sweeping rule or scaling function $f(z)$; an SOR is therefore a SHGC with a circular cross section, where the axis goes through the centre of the cross section, and the cross section is orthogonal to the axis; the sweeping rule $f(z)$ is nothing but the generating function described above. Figure 7.3 illustrates this model, where each parallel corresponds to a scaled and translated version of the reference cross section; curves which are in a plane with the axis are called meridians and are projections of the scaling-function. It is customary to assume that the sweeping rule is a proper function of the position along the axis z, but this is not required for grouping. Due to self-occlusion it is, however, impossible to *reconstruct* any part of the sweeping rule which is not a function of the axis position z from image contours alone.

7.2.1.1 The Contour Generator

In contrast to objects with a planar outline there is, unfortunately, no straightforward invariant relation between an SOR's contour in an image, when viewed from an arbitrary direction, and the planar generating function. What is more, even under weak perspective or orthographic projection there is no straightforward relation between views of the same object taken under different angles, and 2D invariants such as Arbter's affine-invariant Fourier descriptors [12] can not be used. The concept of the contour-generator is the main difference between SORs and the cases discussed in Sections 5 and 6, where a contour in the image generally corresponded to a surface discontinuity in 3D, a so called edge or crease. For SHGCs, however, the contour in the image is most often formed by the intersections between the image plane and rays through the image centre that are *tangent* to the object; the curve on the object-surface where these rays touch is called the contour-generator, and is due to so called limbs, rims, or occluding contours, whose 3D position on the object is a function of the viewing position rather than the object itself (see also Figure 7.4). It can be shown that the contour generator of an SOR, is in general a space curve even under parallel projection and is in general different for each individual viewing position.

7.2.2 Projective Camera Model

It is apparent from the above that very little of general validity can be said about the appearance in an image of a group of objects as varied as SORs (or, in the more general case, SHGCs):

1. For any two contour points on the same parallel (cross section), the tangents to the contours (and, of course, also the cross section) at these points intersect on the projection of the axis [121]. The location of the corresponding point on the 3D-axis is viewpoint independent[50].
2. The curvature of a SHGC's contour at a point is zero iff either the curvature of the sweeping rule or that of the cross section is zero at this point [121]. This means in particular that, ignoring self-occlusion, any inflection of the sweeping-rule results in an inflection of the contour (but not vice-versa).
3. The contours of an SOR to the left and right of the projection of the axis are related by a plane harmonic homology [5]:

$$\mathsf{H} = \mathsf{I}_3 - 2\,\frac{\mathbf{v}\boldsymbol{\ell}^{\mathsf{T}}}{\mathbf{v}^{\mathsf{T}}\boldsymbol{\ell}}. \tag{2.59}$$

This is equivalent to projective symmetry as described in Section 2.8.

Additionally we can make the following statement about two cross sections of an SOR

4. Two cross sections of a SHGC are related by a plane homology:

$$\mathsf{H} = \mathsf{I}_3 + \frac{1 - \mathrm{cr}}{\mathrm{cr}} \cdot \frac{\mathbf{v}\boldsymbol{\ell}^\mathsf{T}}{\mathbf{v}^\mathsf{T}\boldsymbol{\ell}} \tag{2.58}$$

   with the vertex $\mathbf{v}$ on the axis of the SHGC, $(a, b, c)^\mathsf{T}\mathbf{v} = 0$. For any two points on the same meridian, the tangents to the cross sections at these points intersect on the vanishing line of a plane parallel to the cross sections [132].

So what additional information could we get from the use of more restrictive camera models?

7.2.3 Quasi-Calibrated Camera Model

In the case of a quasi calibrated camera (here: an aspect-ratio of 1) we can replace property 3 with a quasi-invariant approximation:

3b. The contours of an SOR to the left and right of the projection of the axis are to a very good approximation related by a plane harmonic homology with a fixed point at infinity, i. e. $\mathbf{v} = (\ell_1, \ell_2, 0)^\mathsf{T}$. This is equivalent to an affine (skewed) symmetry between the two sides. This is only a quasi-invariant, but usable for all but the most extreme of wide-angle lenses [5].

7.2.4 Weak Perspective Camera Model

The weak perspective camera model, as described in Section 2.3.1, assumes that all rays are parallel to each other and orthogonal to the image plane. For normal cameras, this is an accurate model only when viewing a planar object parallel to the image plane; otherwise it is a valid simplification only if the extent of the object's depth is much smaller than its distance from the camera, requiring a telephoto-lens. Although the weak perspective camera model is usually of only limited practical value, it has none the less been added here as as it is the model of choice for much of the literature on the subject [121, 131, 132, 153, 163].

The main reason for the prevalence of this model in much of the literature is that the constraints 3 and 4 simplify considerably to:

3c. The two contours of an SOR to the left and right of the projection of the axis are related by a (Euclidean) symmetry-transformation, i. e. a particular plane harmonic homology with $\mathbf{v} = (\ell_2, -\ell_1, 0)^\mathsf{T}$ in Equation (2.59).

4c. Two cross sections of the same SHGC are related by a similarity transformation, i. e. a plane homology according to Equation (2.58) with $\boldsymbol{\ell} = (0, 0, 1)^\mathsf{T}$ and a direct relation between the cross-ratio and the sweeping-rule, $cr = f(z_1)/f(z_2)$, for cross sections at z_1 and z_2.

The latter constraint, which is not used in this thesis, makes reconstruction of the objects considerably simpler.

7.3 Grouping

Grouping the outline of an SOR is essentially a three-step process. In a first step we need to identify corresponding curve-fragments on both sides of the axis based on local properties of the contour; how this is done will be described in Section 7.3.1. Next we can use these corresponding curve-fragments to calculate the axis $\boldsymbol{\ell}$ and vertex $\mathbf{v}$ in constraint 3 above based on a number of distinguished points from both sides of the outline and verify that the two sides are related by a plane (harmonic) homology. This will be described in more detail in Section 7.3.2. In Section 7.3.3 finally we group several outline-pairs which all share the same plane harmonic homology into one object — this approach of course ignores the possibility that two axes might have been aligned by accident, or might appear aligned in one particular view only (the vertices are the same for all objects where the axes were parallel in 3D, and therefore have little discriminating power).

The assumption of a common plane harmonic homology makes a reliable calculation of the axis and vertex parameters necessary, while the need to calculate many such homologies makes a fast algorithm mandatory. It is for this reason that particular attention has been payed to the calculation of the homology (and, in particular, the axis). Section 7.4 compares several commonly used methods and feature-sets.

7.3.1 Matching Curves

Corresponding curve fragments on both side's of an SOR's contour are projectively related[1], as stated in Restriction 3 or 3b. They consequently have the same projective invariants. Associating corresponding curve fragments is then a matter of matching curve segments with the same projective invariant. Using an index into a hash-table, similar to [4], this is essentially an $\mathcal{O}(n)$ process in the number of curve fragments n, although it can become $\mathcal{O}(n^2)$ for pathologic cases. Invariants for this task abound, possible candidates can be taken from [25, 35, 127] — using the invariants from [127], which are based on bitangents, has the additional advantage that the number of features n is small, typically in the order of $n = 25$ bitangents [4]; it has the disadvantage that it constrains the set of recognisable objects to have at least one pair of concavities.

Once pairs of matching contour fragments have been found, it is then possible to find the transformation between each pair's two segments and test whether the transformation is a plane harmonic homology, as required by Restriction 3, see the next section. This is necessary as many other contour-fragments will also be related by a projective transformation, e. g. instances of repeated structure as described in Section 5.

7.3.2 The Transformation

Once corresponding contour-segments have been found, we can then progress towards the calculation of the plane harmonic homology according to Restriction 3 above. This is done using a two-step approach. First, an approximate solution is calculated based on a small number of distinguished points (compare Section 2.7.3). This is then followed by a slower but more accurate calculation which directly uses the contour-information to fit a type of projective snake. As the convergence of this latter algorithm depends on the quality of the former, we should already calculate a good as possible approximation in that first step — this can then also be used to weed out many obviously wrong hypotheses.

It is generally not trivial to decide which points from each side of the contour correspond to each other. Only for a small number of *distinguished points* is this correspondence easily found. Particularly useful in this respect are points of bitangency as described in Section 2.7.3; corresponding bitangent-lines will intersect each other on the (projection of the) axis of symmetry [50] — I will call this point

[1]In the case of an SOR and a quasi-calibrated camera or better this is to a very good approximation an affine relation, see [5]

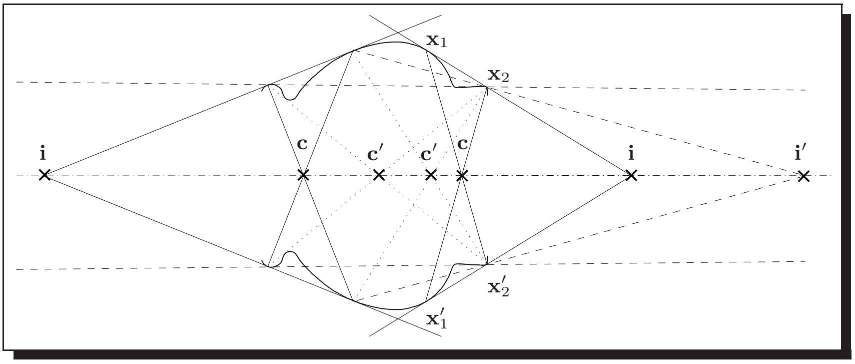


Figure 7.5: *Features used: bitangent intersections are marked* $\mathbf{i}$, *crosspoints are marked* $\mathbf{c}$. *The points marked* $\mathbf{i}'$ *and* $\mathbf{c}'$ *are interpair features.*

the (bitangent) *intersection.* It is marked $\mathbf{i}$ in Fig. 7.5. Additionally, for any two pairs of distinguished points $\{\mathbf{x}_1, \mathbf{x}'_1\}, \{\mathbf{x}_2, \mathbf{x}'_2\}$ the lines through the point-pairs $\{\mathbf{x}_1, \mathbf{x}'_2\}$ and $\{\mathbf{x}_2, \mathbf{x}'_1\}$ will intersect on the axis too, I will call this point a *cross-point.* It is marked $\mathbf{c}$ in Fig. 7.5. Other possible features with essentially the same properties are inflections; however, these aren't used here.

So far we considered each bitangent-pair separately, calculating only *intra-pair* features. However, if more than two distinguished points on each side of the contour are known, each pairing of two points and their corresponding points on the other side of the contour can be used to calculate additional intersections and crosspoints. Figure 7.5 shows a selection of such *interpair* features marked $\mathbf{i}'$ and $\mathbf{c}'$.

The entire approach therefore goes as follows. Given profile fragment pairings (Section 7.3.1):

1. Find an approximate transformation between the two curve fragments by matching a number of distinguished points.
2. Test whether the transformation could be a plane harmonic homology. This could e. g. be done by calculating a number of cross-ratios and compare them to the expected cross-ratio (which should be -1) using the approach from Section 4.6.4; or by projecting points from one side of the contour onto the other side using the equation in property 3 and calculate the distance between the transformed point and its opposite number using the approach

from Section 4.6.3.

3. If the transformation could be a plane harmonic homology, calculate a more accurate transformation using a type of projective snake as described in [9].

7.3.3 Grouping Transformations

It then remains to group separate curve fragment (pairs) which may have arisen from the same profile curve. Grouping is based on the similarity of the parameters of the transformation (i. e. the symmetry axis and corresponding direction) and corresponding pairs of matched concavities are aligned along their common central axis. The associated outline curve fragments can be joined using existing local edgel chain topology and smooth curve continuation, but this is outside the scope of this section.

The comparison itself simply needs to compare the two axes, using the approach described in Section 4.6.2, and the two vertices, using the approach described in Section 4.6.3. As both approaches calculate a χ^2 error measure we can then simply add the two error measures in order to compute an error measure for the entire plane harmonic homology (assuming that the cross-ratio has been fixed at -1). However, the vertex-position comes with little discriminating power (all objects with parallel axes in 3D will have the same vertex), and it might for many applications be sufficient to only compare the axis, or compare the axis first and use the combined error measure only if that initial test was passed.

7.4 The Calculation of the Homology

In the following I will mainly make use of constraint 3, which allows the computation of the plane harmonic homology relating the two sides of a projected SOR. This, in turn, can be used for further grouping, recognition and reconstruction, as we have seen in the previous section.

Rather than trying to solve for the plane harmonic homology all at once (for which usually no closed form solution exists), it is far easier to compute separate results for the axis and vertex. Doing so basically means to compute a best-fit line through a number of points (the axis), and the most likely intersection of a number of lines (the vertex). This is a standard problem in computer vision (and consequently should have a standard solution), but nonetheless many different algorithms for the solution of this problem are in widespread use, some of which were presented in Section 4.3 and 4.4. I will show that the four most commonly used candidates can all be reduced to essentially the same equation, which together with 4 different feature-sets allow us a systematic comparison of 16 different variants. These are described in Sec. 7.4.1. We will see in Sec. 7.4.4 that the most commonly used algorithm, total least squares on the Euclidean plane, which did so well for the calculation of lines through edgels in Section 4.3, tends to be the least reliable for this application. This is another nice example that no silver bullet exists in projective geometry and that it always pays off to incorporate an analysis of the error-behaviour of features. We also need a vertex **v** for the calculation of the plane harmonic homology, and 3 different variants for the calculation of the vertex will be discussed in Sec. 7.4.2, although I will show in Sec. 7.4.4 that the choice of the vertex is of only secondary importance. Section 7.4.3 finally describes the error measure which I will use to assess the goodness of the calculated transformation.

7.4.1 Axis Calculation

We have seen in Section 7.3.2 that the axis of an SOR can be found as a line through a number of feature-points such as bitangent-intersections and cross-points. The most common approach for the calculation of a line through points minimises the orthogonal Euclidean distance between the points and the line:

$$\min_{a,b,c} \frac{1}{N} \sum_{i=1}^{N} (ax_i + by_i + c)^2 + \lambda(a^2 + b^2 - 1)\,. \tag{7.2}$$

This is essentially (4.17); the functional implicitly assumes that the error in the feature-points is Gaussian and independently, identically, and isotropically distributed (iiid). In the context of computer vision often a slightly different formulation is chosen, based on homogeneous coordinates

$$\min_{a,b,c} \frac{1}{N} \sum_{i=1}^{N} (ax_i + by_i + cz_i)^2 + \lambda(a^2 + b^2 + c^2 - 1) \tag{7.3}$$

with $x_i^2 + y_i^2 + z_i^2 = 1$. This functional minimises (locally) orthogonal distances between points on a unit sphere and a great circle representing the line. It is again implicitly based on the assumption of iiid Gaussian noise, but this time on the unit sphere — projecting this back into the image plane we will observe that points further away from the image centre have a much larger standard deviation, and that the error-distribution is a skewed Gaussian. This can be helpful to mirror the fact that features further away from the image plane are indeed usually less accurate than the ones closer to the image plane (as we observed in Section 4.4.3, compare Figure 4.10 on Page 101) — but of course this need not be the case.

Alternatively, both functionals can explicitly consider error-distribution of the feature points which more closely resemble their true distributions. In the following I assume iiid Gaussian noise in the bitangent points and use standard linear error propagation to propagate these error to the feature-points, multiplying the bitangent-points' covariance matrix on both sides with the Jacobian of the feature points. We then again get Gaussian noise, but now with a separate distribution $\boldsymbol{\Sigma}_{p_i}$ for each point $\mathbf{p}_i = (x_i.y_i, z_i)^\mathsf{T}$, and therefore

$$\min_{a,b,c} \frac{1}{N} \sum_{i=1}^{N} \frac{(ax_i + by_i + c)^2}{\sigma_{d_i}^2} + \lambda(a^2 + b^2 - 1) \tag{7.4}$$

$$\min_{a,b,c} \frac{1}{N} \sum_{i=1}^{N} \frac{(ax_i + by_i + cz_i)^2}{\sigma_{d_i}^2} + \lambda(a^2 + b^2 + c^2 - 1) \tag{7.5}$$

where we can calculate the variance in the direction of the line as $\sigma_{d_i}^2 = \boldsymbol{\ell}^\mathsf{T} \boldsymbol{\Sigma}_{p_i} \boldsymbol{\ell}$ — the Jacobian of the distance turns out to be the line itself — this is essentially what we started from when fitting a line to edgels in Section 4.3.

Closer inspection shows that the four equations (7.2)–(7.5) can be subsumed by the more general expression

$$\min_{\boldsymbol{\ell}} \frac{1}{N} \sum_{i=1}^{N} \frac{\boldsymbol{\ell}^\mathsf{T} \mathbf{p}_i \mathbf{p}_i^\mathsf{T} \boldsymbol{\ell}}{\boldsymbol{\ell}^\mathsf{T} \mathbf{W}^\mathsf{T} \boldsymbol{\Sigma}_{\mathbf{p}_i} \mathbf{W} \boldsymbol{\ell}} + \lambda(\boldsymbol{\ell}^\mathsf{T} \mathbf{W} \boldsymbol{\ell} - 1) \tag{7.6}$$

with $\boldsymbol{\ell} = (a, b, c)^\mathsf{T}$, $\mathbf{p}_i = (x_i, y_i, z_i)^\mathsf{T}$ with either $z_i = 1$ (Euclidean coordinates) or $x_i^2 + y_i^2 + z_i^2 = 1$ (homogeneous coordinates), and different values for $\mathbf{W}$ and $\boldsymbol{\Sigma}_{\mathbf{p}_i}$. The $\mathbf{W}$ is a diagonal matrix with either $\{1, 1, 0\}$ (for Euclidean coordinates) or $\{1, 1, 1\}$ (for homogeneous coordinates) as its diagonal elements. The $\boldsymbol{\Sigma}_{\mathbf{p}_i}$ is either the identity-matrix (implicit error model) or a full covariance matrix (explicit model). The minimum can be calculated explicitly if an implicit error-model is used, or else using Kanatani's unbiased estimator [77] as described in Section 4.3.2.1.

In addition to differences in the geometric- and error model I can subdivide algorithms by features used, compare Figure 7.5. These are in our case intersections only versus intersections and crosspoints, intra-pair features only versus intra- and interpair features.

I am coding the different combinations as follows:

8	4	2	1
Error Model	Geom. Model	Features	Combinations
expl. / impl.	xyz / $xy1$	$\mathbf{i}$ & $\mathbf{c}$ / $\mathbf{i}$	inter / intra
1/0	1/0	1/0	1/0

This results in 16 different methods for the calculation of the axis, numbered 0–15. Alg. 3 is the one most commonly used, while Alg. 10 was, e. g., used in [3–5].

7.4.2 Vertex Calculation

I have only implemented three different algorithms. We will, however, see in Section 7.4.4 that for SORs the actual vertex model chosen makes little difference. The three models are labelled 0 (an affine model without explicit error-model, implemented as the average angle towards the vertex at infinity), 2 (a projective model using Euclidean coordinates and no explicit error-model), and 14 (a projective model using homogeneous coordinates and an explicit error-model) — the latter two are calculated by substituting $\boldsymbol{\ell}$ with $\mathbf{v}$ in (7.6) and $\mathbf{p}_i$ with the line through two corresponding distinguished points. As for the number of features used, each bitangent-pair creates exactly 2 lines through the vertex; pairing non-corresponding distinguished points is not possible.

7.4.3 Error Measure

For contours related by a planar harmonic homology, it is directly possible to quantify the quality of the *calculated* planar harmonic homology even without ground truth — we can simply use H to map one side of the contour onto the other side, and use some error measure between the two curves to assess the goodness of fit. This is often done using the Hausdorff distance of the two contours, basically the maximum distance between the two sets. This is, however, not a very intuitive or descriptive measure, and I use instead the average difference between edgels on each side of the contour,

$$\varepsilon = \frac{1}{y_2 - y_1} \int_{y_1}^{y_2} \|\mathbf{p}_{\text{right}}(y) - \mathsf{H}\mathbf{p}_{\text{left}}(y)\| \, \mathrm{d}\, y \,, \tag{7.7}$$

where (7.7) assumes that the object was rotated into an upright position. Note that even for perfect symmetry this error measure or residual will not be zero, as the position of the edgels along the contour will be noisy. We can therefore only expect a value in the same order as the standard deviation of our edge detection algorithm (or, more accurately, $\sqrt{2}$ times the standard deviation, as both sides will be subject to measurement errors). In this thesis I used a very simple implementation of the Canny edge finder [24] to extract the edges. Its standard deviation on grey-level images is in the order of $0.1\,\text{pxl} \leq \sigma \leq 0.3\,\text{pxl}$.

7.4.4 Results

Each of the 16 algorithms for the calculation of the axis and 3 algorithms for the calculation of the vertex were run on a total of 49 images[2] of 6 SORs (see Fig. 7.6) which have previously appeared in publications about the recognition of SORs [3–5]. This resulted in 48 different values for the harmonic homology in each image and $48 \times 49 = 2352$ different harmonic homologies overall. For each homology I also calculated the residual as described in Sec. 7.4.3 and used this to determine the relative goodness of fit for each approach.

When comparing these different algorithms it is important to remember that any algorithm could perform best for one particular set of features due to statistical fluctuations (and we will see in Sec. 8.1 that the particular shapes of some of the objects do indeed skew the outcome). To alleviate these effects of random fluctuations, I use different measures of fitness to assess the quality of the algorithms. These measures are either based on the actual residual calculated (in Sec. 7.4.4.1), or on an algorithm's relative performance compared to all other algorithms, its ranking (in Sec. 7.4.4.2).

7.4.4.1 Absolute Performance

Figure 7.7 shows the range (minimum, median, and maximum) of residuals encountered for each of the 48 combinations of axis- and vertex models, ordered by axis model firstly and features used secondly. In the following I will mostly be interested in the maximum residual calculated, as some algorithms could clearly result in unacceptably wrong results, which of course need to be avoided if the number of false negatives is to be kept small; only then will I consider the median residual, which gives information about the algorithms' average performance. The minimum residual is of little interest to us as, given enough trials, it will always be in the order of $\varepsilon \approx \sqrt{2}\sigma_{\mathbf{p}_i}$.

I already mentioned before that the actual algorithm used for the calculation of the vertex is of little importance for our comparison (assuming SORs), and this is born out by Fig. 7.7, where results look similar for all three vertex models. I will therefore discuss algorithms by axis model in the following.

Maybe the most interesting result, when studying Fig. 7.7, and at first glance contrary to this thesis' line of argument, is that a more complicated error-model will not necessarily improve results; using more features, on the other hand, can in

[2]The relevant contours and bitangents were selected by hand, so as not to confound the comparison with additional issues.

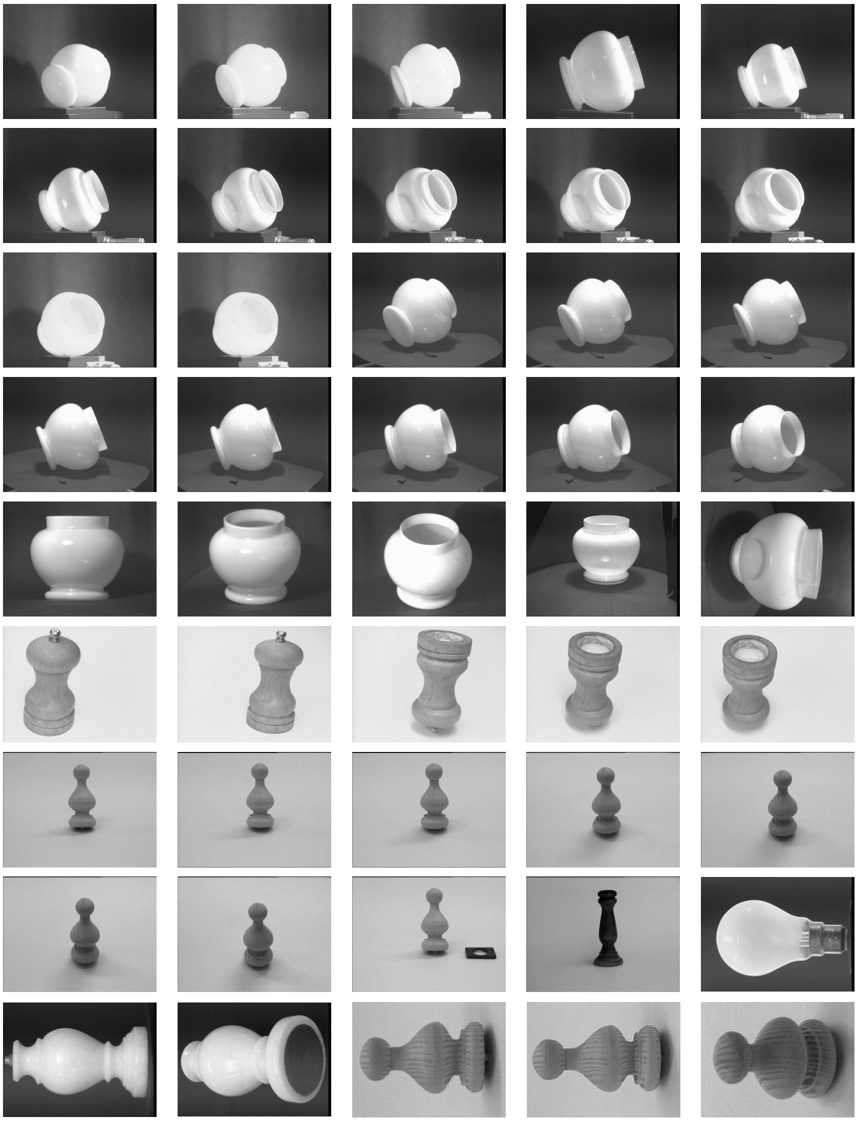
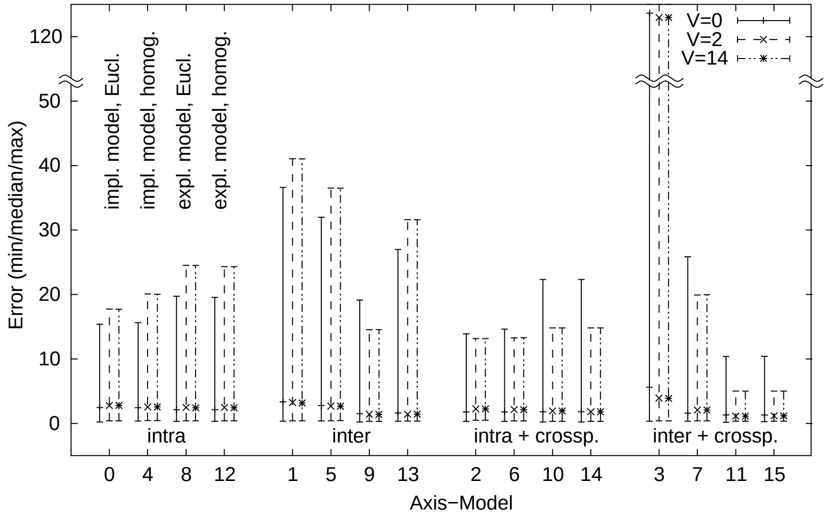


Figure 7.6: *45 of the 49 images of SORs used.*



	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
max	18.	20.	25.	24.	41.	37.	15.	32.	13.	13.	15.	15.	123	20.	**5.0**	**5.0**
med	2.8	2.6	2.4	2.4	3.2	2.7	**1.4**	**1.4**	2.2	2.2	2.0	**1.8**	3.9	2.1	**1.1**	**1.2**
min	.40	.46	.41	.41	.40	.44	**.32**	**.36**	.49	**.39**	**.32**	**.32**	.41	.41	**.34**	**.35**

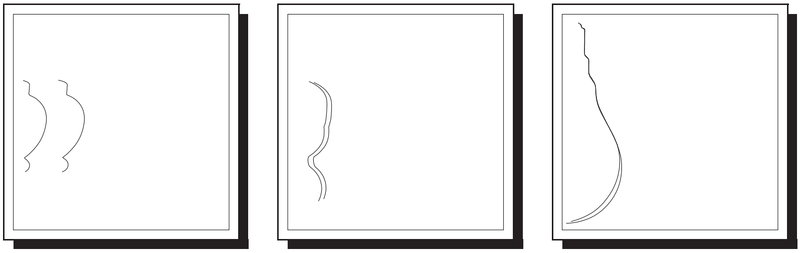
Figure 7.7: *Range of residuals. For each algorithm the minimum, median and maximum residual are plotted — note the discontinuity along the ordinate. The table shows the numerical values for the second best vertex model.*

fact considerably decrease performance. The former can be easily seen in the case where only intra-pair features are used (first and third block in Figure 7.7). These features are reasonably reliable, so that good results can be obtained even without an explicit error-model, while the error model used obviously isn't completely accurate at least for some cases — we see from the table to Fig. 7.7 that an explicit error-model did in all cases reduce the median (and in most cases also the minimum) error, just not the maximum error. However, even the maximum error decreases once we are also using the less reliable interpair features together with an explicit error-model (ignoring models 9 and 13 which, according to theory, should have performed similarly, but surprisingly didn't).

The latter, that more features can give worse results, can be seen when we compare the algorithms using intra-pair bitangent-intersections, 0, 4, 8, and 12 (the first block), with the ones using interpair intersections, 1, 5, 9, and 13 (the second block) — the maximum and median error actually increase for the algorithms which do not use an explicit error-model, although many more features are used ($2N(N-1)$ versus N features). The most striking example is provided by Algorithm 3, which uses all available feature points and simple orthogonal regression in the image plane. This is the most commonly used model for the calculation of a line through several points, and the one which performed so well when fitting a line to edgels in Section 4.3 — but clearly the solutions found by this approach cannot be relied on at all for applications where the underlying assumption of iiid measurements is not valid. Even the median residual for this method is higher than that from any other model, and the maximum error can be absolutely intolerable. For the algorithms with an explicit error model, on the other hand, the median error decreases drastically by about 70 %. Figure 7.8 illustrates how the maximum errors from Algorithms 3, 9 and 15 will affect the result, to give an idea how good or bad the relative errors are.

The third thing to be learned from Fig. 7.7 is that the additional use of cross-points will, as a rule, improve the results calculated. This is particularly true for algorithms which use an explicit error-model.

To sum up: as expected a high number of features is indeed preferable, but only if used together with an explicit error-model; without such a model the emphasis should be put on accurate rather than numerous features — this is in direct conflict with the assumption underlying many algorithms that more features are always better. And although even the algorithm with the lowest maximum residual (axis model 15, vertex-model 14 — the most refined model using the most features) will produce noticeable errors for some input-constellations, we can see from Fig. 7.8, right, that the results are even in the worst case much more usable than for some of the other algorithms, Fig. 7.8 left and middle. It should also be noted that this



Axis model 3
Vertex model 2
$r = 122.763$

Axis model 9
Vertex model 2
$r = 14.5296$

Axis model 15
Vertex model 14
$r = 5.03427$

Figure 7.8: *Cases of maximum residual for three axis models (using the best vertex-model). The graphs show the left contour mapped onto the right one*

particular object is actually not quite symmetric, although in this case 8 out of the 48 algorithms tested performed better. Ranking the relative performance of all algorithms is indeed another possibility to determine fitness, and will be done in the next section.

7.4.4.2 Relative Performance

Although any algorithm might return the smallest residual for one particular outline, we would nonetheless expect that the better an algorithm is suited for the task, the more often should it show up among the best N algorithms; conversely the more often it is placed among the worst N algorithms, the more unsuitable would we deem this algorithm. Table 7.1 lists, for each algorithm, how often it was observed among the best 3 algorithms. From this table it seems as if performance is mostly a matter of features used — all algorithms from the first block (intra-pair intersections only) perform considerably worse than any of the algorithms from the last block, using the maximum number of features, and algorithms using an intermediate number of features perform somewhere in between. The image becomes somewhat clearer if we also consider the 3 worst algorithms, shown in Table 7.2. The Algorithms 0–7, which use no explicit error model, account for 84 % of the worst 3 algorithms.

The usefulness of an explicit error-model becomes even more apparent if we look at a histogram of the ranks achieved with the Algorithms 6, 7, 11, and 15 (which, according to Tables 7.1 and 7.2, all performed similarly, while in theory 11 and 15

Table 7.1: *How often out of 49 runs each algorithm was among the best 3*

Vertex Model	Axis Model																Sum
	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15	
0	3	2	4	3	2	4	9	7	7	8	5	5	2	14	8	8	91
2					2	1	1	1		3	2	1	2	2	1	6	22
14			1	1	3		2	2		2	2	2	3	2	5	9	34
Sum	3	2	5	4	7	5	*12*	10	7	*13*	9	8	7	**18**	*14*	**23**	147

Table 7.2: *How often out of 49 runs each algorithm was among the worst 3*

Vertex Model	Axis Model																Sum
	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15	
0	8	5	1		11	2	1	2	1	1	2	2	19	2	1	1	59
2	5		1	1	11	1		1			1	4	19	1			45
14	6			1	11	1		1			2	1	18	2			43
Sum	*19*	5	2	2	**33**	4	1	4	1	1	5	7	**56**	5	1	1	147

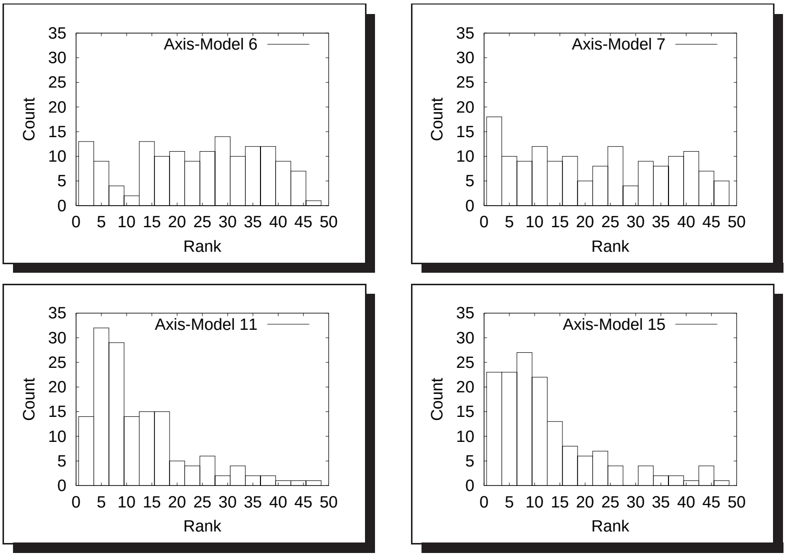


Figure 7.9: *Histograms of rank for axis models 6, 7, 11, and 1*

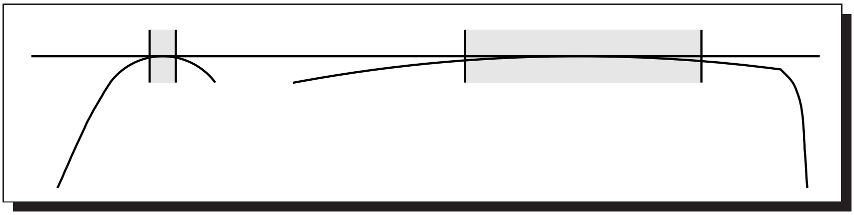


Figure 7.10: *The accuracy with which bitangent points can be located depends on the contour's curvature in that region. This has little influence on bitangent-intersections, but can considerably influence the position of cross-points and interpair intersections*

should exhibit superior performance). Figure 7.9 shows a clear difference between the algorithms which do not use an explicit error model (6 and 7, top row) on the one hand and the ones which do (11 and 15, bottom row) on the other. The former (as do most other models) show a nearly uniform distribution, which means that they are similarly likely to be among the N best as well as the N worst algorithms, while the latter's distribution looks somewhat like a Poisson distribution, with good ranks much more likely than bad ones. This shows that the overall likelihood of an acceptable result is much higher for axis models which use as many features as possible *together* with an explicit error-model.

I believe this to be strong evidence that the use of an explicit error-model, at least when used together with many features of varying quality, can considerably improve an algorithm's performance.

7.5 Discussion

We have seen in Sec. 7.4.4.1 and 7.4.4.2 that using more features and an explicit error-model will indeed overall improve the performance of an algorithm (as we expected). However, we also noticed some inconsistencies, and these will be explained in the following.

Considering only the maximum residuals in Fig. 7.7 we noticed that algorithms using more features (second and forth block in Fig. 7.7) can perform noticeably worse than the corresponding algorithms which use fever features, in particular if no explicit error-model is used. The reason for this is the particular shape of most of our test-objects. Consulting Fig. 7.6 we notice that most objects contain sec-

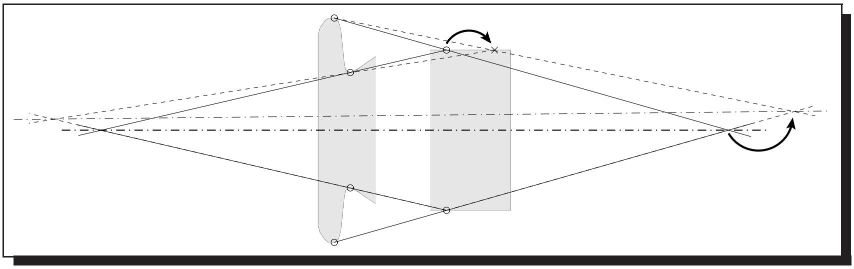


Figure 7.11: interpair *intersections can be quite inaccurate for many common objects, yet at the same time far enough away from the object to have considerable influence on the location of the axis*

tions of extremely low curvature (nearly straight in fact), and that in most cases a bitangent will touch the object in that area. This is true for the neck of the first object and the foot of the second and sixths, which together contribute about 66 % of all contours. The position of a bitangent point along such a low-curvature segment can only be calculated quite inaccurately, see Fig. 7.10, independent of the algorithm used (I used a Houghbased algorithm with iterative refinement, [127]). This has very little influence on the orientation of that particular bitangent (whose accuracy basically depends on the distance between the two points of tangency), and consequently little influence on the position of the bitangent-intersection (and, in consequence, little influence on the axis models 0, 4, 8, and 12); it can, however, greatly influence the position of interpair intersections (1, 5, 9, and 13) — Figure 7.11 gives an example. Given N bitangent-pairs, only 1 intersection (containing the erroneous bitangent point) will be calculated correctly, but $2(N-1)$ intersections will be incorrect (compare Fig. 7.5). Additionally, many of those intersections will be far away from the object, and will consequently have high influence on the final result (in particular if Euclidean coordinates were used). It is therefore not surprising that results can become nearly arbitrarily wrong. Using crosspoints (models 3, 7, 11, 15) can mediate this effect; while their position will be wrong too, they will actually be on the other side of the axis and therefore offset some of the effect. The correct solution of course would be an error model which computes a point's accuracy along the bitangent based on the contour's curvature around the point. However, as curvature is impossible to compute accurately for low-curvature contours (as I demonstrated in [7]), such a model will be difficult to implement (see Sec. 7.6).

7.6 Conclusions

A computer-vision system which aims to locate, group, identify and possibly reconstruct SORs needs to calculate the harmonic homology relating the two sides of the contour. A very accurate algorithm for the calculation of the harmonic homology was given by me about 10 years ago [9], however, this algorithm is based on numerical minimisation and, depending on the initial estimate of its parameters, might require many iterations in order to converge. While not a problem for a single outline (each iteration is quite fast), this can nonetheless severely limit its usefulness in the case of cluttered images containing many possibly symmetric objects, or in the case where huge numbers of images need to be analysed, as for image-database applications or within webcrawlers. Having a faster algorithm which serves both to weed out many wrong matches, as well as providing the following stages with accurate initial values, can provide a considerable speedup. Having an algorithm available which is based solely on distinguished points is also of particular importance for SHGCs, where no better algorithm is known — see also the discussion in Section 7.7.

In this section 48 different algorithms for this intermediate step have been compared and it has been demonstrated that using an explicit error model can considerably improve the results, in particular where many features of varying accuracy are used. This can mean the difference between completely useless results in the naive — but widely used — case on the one side and highly reliable results on the other.

There is, however, still room for improvements. We have seen in Sec. 7.5 that the proposed method, while already of very high accuracy even in the worst case, could most likely be further improved by the use of a curvature-based error model. Once such an error model were in place it would also enable us to use an additional kind of distinguished point, namely inflections, isolated points of zero curvature. All the properties of bitangent points given above also hold for inflections; it is however difficult to accurately compute both their orientation (needed for the intersection) and position (needed for crosspoints, interpair intersections and the calculation of the vertex) simultaneously. A curvature-based error-model would allow us to quantify this uncertainty and take it into account. While easily enough done in theory it unfortunately suffers from the fact that curvature for low-curvature regions cannot be calculated accurately in practice, as I demonstrated in [7].

7.7 SHGCs and Symmetric Contours

Much of what has been said about SORs is also applicable to objects with a planar symmetric contour generator and to SHGCs. The former is immediately obvious from the fact that I only used property 3 explicitly, relating to the projective symmetry between the two sides of the contour of an SOR. This symmetry-relationship of course also applies to all planar symmetric objects, as well as to all three-dimensional objects which only have a planar outline — examples include flying airplanes as viewed from the ground [12], tools like a pair of pliers or scissors [27, 83, 92], objects like spoons or ashtrays [101], or individual faces of complex objects [55, 128]. Properties 1 and 2 were only used implicitly, in the selection of the distinguished points, but since order of contact is a projective invariant those do of course hold here too.

Things are more difficult for SHGCs. Here only Properties 1 and 2 are valid; we can use those to locate distinguished points and to compute the axis as a line through the intersections of bitangents or inflections. However, neither cross sections nor interpair features are meaningful for SHGCs, and pairing corresponding contour-segments will usually be difficult. One possible approach is to use the Hough-like algorithm proposed in [121] to identify candidate pairs and from there proceed using bitangent points and intersections; otherwise we would have to investigate all $\mathcal{O}(n^2)$ possible combinations.

Chapter 8

Conclusion

“Tut, tut, child,” said the Duchess. “Everything’s got a moral if only you can find it.”

Lewis Carrol, Alice’s Adventures in Wonderland, 1832–1898

8.1 Discussion

In this thesis I have presented an approach for the combination of projective geometry with error propagation. I also presented three different application domains were the combination of projective geometry and error propagation was used successfully to group diverse features of images of two- and three-dimensional structures. I believe that these examples favourably showcase the advantage of error-propagation augmented projective geometry over standard projective geometry. In the following I will recapitulate the individual chapters and their main observations in more detail.

The thesis started with introductions into projective geometry in Chapter 2 and error propagation in Chapter 3. These only serve as a summary and reminder and do not contain any original material. The thesis proper started in Chapter 4, which describes the combination of error-propagation principles with projective geometry. Here I discussed a number of additional parameterisations, in addition to standard homogeneous coordinates, exemplarily concentrating on the $(\alpha, x, y)^\mathsf{T}$ parameterisation for lines in many examples. As a first example I revisited the line-fitting problem in Section 4.3. Starting from first principles I demonstrated that the usually employed algorithm, orthogonal regression or total least squares, is only applicable for independently, identically, and isotropically distributed (iiid) features; I then demonstrated that this is approximately the case for edgels — and, later on in Chapter 7, demonstrated that a violation of this restriction can result in completely unusable results. In addition to the exact solution for a line's covariance matrix I next gave an excellent and previously unpublished approximation to this matrix, which allows us to reap all the advantages of error propagation at virtually no extra costs — at least if fitting a line to iiid edgels (see Section 4.3.2.2). I next considered the problem of incrementally fitting a line to edgels along an edgel-chain. Here the main problems are to decide when to start fitting, and when to stop, and in Section 4.3.3 I gave a χ^2-based error measure which allows us to fit lines to considerably fewer edgels than was previously possible (using a fixed error measure).

I next had a look at the problem of finding a point as the intersection of lines (Section 4.4). In projective geometry this is usually considered the dual problem of the line-fit problem described above — however, I believe that I was able to satisfactorily show that for practical problems in computer vision this is usually *not* the case. I gave a number of different approaches which all manage to find the most likely intersection of a number of lines with varying distributions, but tackle the problem from different angles, thereby highlighting different aspects of the underlying geometric problem. The results of a Monte-Carlo simulation for different line-constellations were then used to give an intuitive explanation why the spherical normalisation used by many authors is indeed superior to a Euclidean normalisation.

The first thing which springs to the mind of any person working in computer vision when coincident lines or collinear points are mentioned is the cross-ratio of four such points or lines. The accurate calculation of such a cross-ratio traditionally requires the accurate calculation of the common line / point prior to the calculation of the cross-ratio. However, if done correctly this can become a costly operation, since we have seen in Sections 4.3 and 4.4 that in the general case (non-iiid features) only iterative solutions exist. In Section 4.5 I therefore presented a new approach which uses error-propagation principles to approximately minimise the error in the cross-ratio *without* prior calculation of a common line / point. The results from this approach can be virtually as good as the results of the best available algorithms, but at a fraction of the cost. I believe that this example clearly demonstrates that not only needs the use of error propagation not be costly, but it can in fact help to both increase the accuracy and to speed up algorithms at the same time.

In Section 4.6 I finally demonstrated how to compare geometric features considering their relative uncertainty. This offers the unwary a few pitfalls not present in standard stochastic: redundant parameterisations, which are very common in computer vision and projective geometry, need to be normalised, and the usually singular covariance matrices warrant special treatment. These are, however, easy to deal with in practice, and the use of a statistical test (I recommend the χ^2-test) allows us to replace the fixed thresholds all too common in computer vision ("*Is the difference at most 10 pixel?*") with a much more meaningful measure based on confidence in the result ("*Is there at most a 5 % chance that this observation does contradict my assumption?*"). The next three chapters then showed applications of these principles.

In Chapter 5 I presented an application where originally parallel lines of a given cross-ratio are grouped. The size of the features can vary considerably (by about a factor of 10) from image to image, but also within a single image, where both the width of a stripe (due to perspective foreshortening) as well as the length (due to occlusions) can again easily vary by a factor of 10. In addition there are only very lax bounds on the position of the line segments; the vanishing point can be located in the image, at infinity, or anywhere in between. All this makes fixed thresholds absolutely useless (a 10 pixel distance between lines and vanishing point is huge if the point is visible in the image, but nothing if the vanishing point is close to infinity), while the χ^2 measures remains the same for all possible vanishing-point locations. I do indeed believe that this problem can not be tackled without the use of error propagation, and the fact that all papers based on my work [6][134, 135, 137] either used at least some form of rudimentary error propagation or (usually) only showed examples with much more homogeneous features (line segments of similar length and distance, vanishing point and vanishing line of the structure far outside the image) underscores this belief (a proof is, of course, impossible).

In Chapter 6 I presented an application where the same features, short line segments, are used both to calculate points far away as well as close to the image (vanishing points), but also to group several short segments into one longer segment. Both operations are based on the same basic information derived from a simple line fit and simultaneous calculation of the line-segments' covariance matrices as described in Section 4.3; however, here I also demonstrated how additional knowledge about the uncertainty in the 3D-model can be incorporated. The vanishing points, assumed orthogonal in reality, are then used to calculate an approximation of the focal length in a statistically sound way, which takes into account the considerably different covariances of vanishing points at infinity compared to such close to the image, compare Section 4.4. But here I also addressed a possible representation of uncertainties both in 2D and 3D; in Section 6.4.1 I compared the performance of several error models from the literature on the identification of line-continuations, and in Section 6.4.2 I had a closer look at the effect that positional and orientational errors in 3D can have on the calculation of vanishing points (see Section 6.4.2.1) and line-continuations (see Section 6.4.2.2). The different pieces were finally used to outline how an algorithm for the grouping and segmentation of the individual faces of houses (or similar structures) could look like, and Section 6.5 presents a number of encouraging results even for so simple an algorithm.

In Chapter 7 finally I revisited the problem of fitting a line to points, but instead of iiid edgels, as was assumed in Section 4.3, I now used bitangent-intersections, features which are very much neither identically nor isotropically distributed. I compared several algorithms for the calculation of a line through these features and demonstrated that the most commonly used algorithm, total least squares, can fail completely under such a scenario (which violates the implicit assumption of iiid data), while algorithms which do use error propagation and therefore take into account the individual points' covariances perform consistently well. All in all I believe that Chapters 4–7 convincingly demonstrated both that error propagation need not be painful or time-consuming, but also that error propagation can greatly simplify or in fact afford computations which would otherwise have been difficult or impossible.

8.2 Research Directions

A number of incremental improvements suggest themselves: the algorithm presented in Section 6 could use corner-information and the subjective structures introduced by Brillault-O'Mahony in [21]; the positional error in the bitangent points in Chapter 7 should really be based on the curvature at that point — however, since I proved in [7] that for all interesting cases curvature can not be calculated from edgels alone this would either mean to find a new method for curvature calculation (possibly based on the entire gradient-field as opposed to just the gradient's maximum) or an heuristic approximation to curvature. All this is thematically well outside the scope of this thesis, but might make interesting projects for a student-thesis. Incremental improvements are also possible in the automatic determination of an edgel's covariance depending on imaging device and edge-finder used, or in the consideration of orientation dependent covariances; these would again make for nice student-theses.

A much more serious problem which this thesis completely ignored is the problem of bias due to projective foreshortening. The originally Gaussian (or assumed Gaussian) noise in the image can become strongly non-Gaussian when projected back onto the object; in particular will the distribution's mean in the image not normally be mapped onto the transformed distribution's mean in the object frame, but will have a systematic offset, and this offset can result in a bias of derived features. It is possible to account for and correct this bias, but only if the transformation between object and image is known up to an Euclidean transformation, which is not normally the case; it might be interesting to see what corrections are possible if only structural information is available as was the case for the examples in Chapters 5–7.

8.2.1 Towards Multi-modal Representations

However, after several years of work in projective geometry, all of it contour based, I have come to the conclusion that the confinement to edgels alone is simply too limiting to allow for anything more than incremental improvements. Contour based computer vision, which looked so promising 30 years ago, is really rather like sitting in Plato's cave, trying to guess what the world outside might look like from shadows alone[1]. Getting rid of texture and shading, which looked like a boon in the days when memory was counted in kilobytes and computing speed in

[1]We are, of course, in a much more fortunate position than the people in Plato's cave, since we do have a host of a-priori knowledge about the real world at our disposal

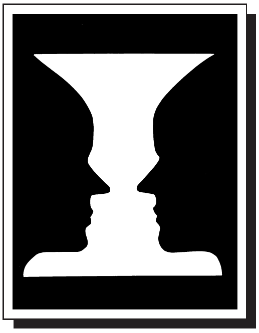


Figure 8.1: *Ambiguous image of either a vase or two faces. The ambiguity cannot be resolved without additional information such as shading.*

kilohertz, has now come back to haunt us. True, we can deal with images like the one in Figure 8.1 — if we know whether we are dealing with either SORs or human faces — but the loss of information if only contours are considered is hard to make up for. I still believe that the application described in Chapter 5 — the detection of pedestrian crossings — is best done using a line-based algorithm; but grouping the individual faces of houses in Chapter 6 is at least difficult without the use of colour or texture, it becomes essentially unsolvable if we are dealing with things like row- or terrace-houses, where individual houses differ by colour and texture alone, but otherwise have exactly the same geometry.

And of course here too we are dealing with measurements — but how do we model the error in, e. g., a colour? A hue based representation suggests itself, but what about the brightness? Even on a planar surface this will rarely be uniform, and this certainly isn't the case for any non-planar surface. Should brightness be modelled using a predictive filter? Some sort of Markov process? Maybe it shouldn't be modelled at all? Currently a host of different representations for colour coexist, and this is the easy case — modelling the error in texture representations might prove the real challenge. It is a wide field out there, and anybody not believing that a mixture of Gaussians is the answer to everything has his work cut out.

Own Publications

[1] A. Luo, W. Tao, S. Utcke, and H. Burkhardt. MOVIS: Über die Entwicklung eines ersten Prototypen einer Blindenbrille. Interner Bericht 3/98, Albert-Ludwigs-Universität, Freiburg, Institut für Informatik, 1998.

[2] V. Müller and S. Utcke. Advanced quality inspection through physics-based vision. In *Proc. of the the International Symposium Machine Vision in the Industrial Practice*, Steyr, Österreich, 1995.

[3] J. Mundy, A. Liu, N. Pillow, A. Zisserman, S. Abdallah, S. Utcke, S. Nayar, and C. Rothwell. An experimental comparison of appearance and geometric model based recognition. In *Proc. Object Representation in Computer Vision II*, LNCS 1144, pages 247–269. Springer-Verlag, 1996.

[4] J. L. Mundy, C. Huang, J. Liu, W. Hoffman, D. A. Forsyth, C. A. Rothwell, A. Zisserman, S. Utcke, and O. Bournez. MORSE: A 3D object recognition system based on geometric invariants. In M. Kaufmann, editor, *Image Understanding Workshop*, pages II:1393–1402, Monterey, CA, November 13–16 1994. ARPA.

[5] N. Pillow, S. Utcke, and A. Zisserman. Viewpoint-invariant representation of generalized cylinders using the symmetry set. *Image and Vision Computing*, 13(5):355–365, June 1995.

[6] S. Utcke. Grouping based on projective geometry constraints and uncertainty. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 739–746, Bombay, Jan. 1998. IEEE Computer Society, Narosa Publishing House, New Delhi.

[7] S. Utcke. Error-bounds on curvature estimation. In *Scale Space*, pages 657–666, Isle of Skye, Scotland, UK, June 2003. British Machine Vision Association, Springer-Verlag, Berlin.

[8] S. Utcke and A. Zisserman. Projective reconstruction of surfaces of revolution. In B. Michaelis and G. Krell, editors, *25. DAGM-Symposium Mustererkennung*, volume 2781 of *Lecture Notes in Computer Science*, pages 265–272, Magdeburg, Germany, Sept. 2003. DAGM, Springer-Verlag, Berlin.

[9] A. Zisserman, J. Mundy, D. Forsyth, J. Liu, N. Pillow, C. Rothwell, and S. Utcke. Class-based grouping in perspective images. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 183–188, Cambridge, MA, USA, June 1995. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, California.

Bibliography

[10] S. M. Abdallah and A. Zisserman. Grouping and recognition of straight homogeneous generalized cylinders. In *Asian Conf Comput Vision*, pages 850–857, Taipei, Jan. 2000.

[11] T. D. Alter and D. W. Jacobs. Uncertainty propagation in model-based recognition. *International Journal of Computer Vision*, 27(2):127–159, 1998.

[12] K. Arbter, W. E. Snyder, H. Burkhardt, and G. Hirzinger. Application of affine-invariant fourier descriptors to recognition of 3-D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):640–647, July 1990.

[13] S. T. Barnard. Interpreting perspective images. *Artificial Intelligence*, 21(4): 435–462, Nov. 1983.

[14] S. Becker and V. M. Bove, Jr. Semiautomatic 3-D model extraction from uncalibrated 2-d camera views. In *SPIE Visual Data Exploration and Analysis II*, pages 447–461, San Jose, California, Feb. 1995.

[15] P. Bellutta, G. Collini, A. Verri, and V. Torre. 3D visual information from vanishing points. In *Workshop on Interpretation of 3D Scenes*, pages 41–49, Austin, Texas, Nov. 1989. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, California.

[16] L. M. Biebermann. *Perception of Displayed Information.* Plenum Press, NY/London, 1973.

[17] F. L. Bookstein. Fitting conic sections to scattered data. *Computer Graphics and Image Processing*, 9:56–71, 1979.

[18] M. Born and E. Wolf. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light.* Cambridge University Press, Cambridge, UK, 7. edition, 1999.

[19] M. Breyer. Detektion von Polygonen geringer Größe und Komplexität in natürlichen Szenen basierend auf Kanteninformation. Studienarbeit, Arbeitsbereich Technische Informatik I, TU Hamburg Harburg, Apr. 1997.

[20] B. Brillault-O'Mahony. New method for vanishing point detection. *Computer Vision, Graphics and Image Processing: Image Understanding*, 54(2):289–300, Sept. 1991.

[21] B. Brillault-O'Mahony. High level 3D structures from a single view. *Image and Vision Computing*, 10(7):508–520, Sept. 1992.

[22] I. N. Bronstein and K. A. Semendjajew. *Taschenbuch der Mathematik*. G. Grosche and V. Ziegler and D. Ziegler, Harri Deutsch, Thun und Frankfurt (Main), 23. edition, 1987.

[23] Bundesanstalt für Straßenwesen (BASt). Zusätzliche technische Vorschriften und Richtlinien für Markierungen auf Straßen ZTV-M 84. FGSV, Sept. 1984.

[24] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, Nov. 1986.

[25] N. Canterakis. Complete projective semi-differential invariants. In *Proceedings of the International Workshop on Computer Vision and Applied Geometry*, Nordfjordeid, Norway, Aug. 1995.

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

[27] T.-J. Cham and R. Cipolla. A local approach to recovering global skewed symmetry. In *Proceedings of the 12th International Conference on Pattern Recognition*, volume I, pages 222–226, Jerusalem, Israel, Oct. 1994. International Association for Pattern Recognition, IEEE Computer Society Press, Los Alamitos, California.

[28] W. Chen and B. C. Jiang. 3D camera calibration using vanishing point concept. *Pattern Recognition*, 24(1):57–67, 1991.

[29] J. C. Clarke. Modelling uncertainty: A primer. Department of Engineering Science, Oxford University, Parks Rd., Oxford, OX1 3PJ, UK, 1998.

[30] C. Coelho, M. Straforini, and M. Campani. Using geometrical rules and a priori knowledge for the understanding of indoor scenes. In R. M. Haralick and W. Förstner, editors, *International Conference on Robust Computer Vision*, pages 229–234, Seattle, Oct. 1990.

[31] R. T. Collins and R. S. Weiss. An efficient and accurate method for computing vanishing points. Image Understanding and Machine Vision, Technical Digest Series, Optical Society of America, June 1989.

[32] C. Colombo, A. D. Bimbo, and F. Pernici. Metric 3D reconstruction and texture acquisition of surfaces of revolution from a single uncalibrated view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, submitted 2003. accepted 2004.

[33] A. Criminisi, I. Reid, and A. Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.

[34] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 434–442, Sept. 1999.

[35] R. W. Curwen, J. L. Mundy, and C. V. Stewart. Recognition of plane projective symmetry. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 1115–1122, Bombay, Jan. 1998. IEEE Computer Society, Narosa Publishing House, New Delhi.

[36] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In H. Rushmeier, editor, *SIGGRAPH*, volume 30 of *Computer Graphics Annual Conference Series*, pages 11–20, New Orleans, Louisiana, USA, Aug. 1996. ACM SIGGRAPH, Addison Wesley.

[37] Defence Advanced Research Projects Agency. *Image Understanding Workshop*, San Diego, CA, Jan. 1992. Defence Advanced Research Projects Agency, Morgan Kaufmann Publishers, San Mateo, CA.

[38] M. Dhome, J. T. Lapreste, G. Rives, and M. Richetin. Spatial localization of modelled objects of revolution in monocular perspective vision. In *Proc. 1st Int. Conf. on Computer Vision*, pages 475–485. 1990.

[39] L. E. Dickson. *Algebraic Invariants.* Number 14 in Mathematical Monographs. John Wiley & Sons, Inc., New York; London: Chapman & Hall, Limited, 1. edition, 1914.

[40] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[41] T. Echigo. A camera calibration technique using three sets of parallel lines. *Machine Vision and Applications*, 3:159–167, 1990.

[42] T. Ellis, A. Abbood, and B. Brillault. Ellipse detection and matching with uncertainty. *Image and Vision Computing*, 10(5):271–276, June 1992.

[43] O. Faugeras. *Three-Dimensional Computer Vision.* The MIT Press, Cambridge, Massachusetts, 1. edition, 1993.

[44] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Proceedings of the Second European Conference on Computer Vision*, volume 588 of *Lecture Notes in Computer Science*, pages 563–578, Berlin, Heidelberg, May 1992. Springer-Verlag.

[45] M. A. Fischler, S. T. Barnard, R. C. Bolles, and M. Lowry. Modelling and using physical constraints in scene analysis. In Kaufmann, editor, *Proceedings of the National Conference on Artificial Intelligence*, pages 30–35, Los Altos, Cal., 1982.

[46] 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(6):381–395, June 1981.

[47] R. A. Fisher. Dispersion on a sphere. In *Proc. Roy. Soc. Lond.*, volume A217, pages 295–305, 1953.

[48] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, May 1999.

[49] W. Förstner. Uncertainty and projective geometry. In E. Bayro-Corrochano, editor, *Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neurocomputing and Robotics*. Springer, 2004. to appear.

[50] D. A. Forsyth, J. L. Mundy, A. Zisserman, and C. A. Rothwell. Recognising rotationally symmetric surfaces from their outlines. In G. Sandini, editor, *Proceedings of the Second European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 639–647. Springer Verlag, 1992.

[51] P. Gamba, A. Mecocci, and U. Salvatore. Vanishing point detection by a voting scheme. In P. Delogne, editor, *Third International Conference on Image Processing*, volume 2, pages 301–304, Lausanne, Switzerland, Sept. 1996. IEEE Signal Processing Society, Ceuterick, Leuven.

[52] N. Georgis, M. Petrou, and J. Kittler. Error guided design of a 3D vision system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 366–379, 1998.

[53] R. Glachet, J. T. Lapreste, and M. Dhome. Locating and modelling a flat symmetric object from a single perspective image. *Computer Vision, Graphics and Image Processing: Image Understanding*, 57(2):219–226, Mar. 1993.

[54] M. Greiffenhagen. Segmentierung von Häuserfronten basierend auf kollinearen Strukturen. Studienarbeit, Arbeitsbereich Technische Informatik I, TU Hamburg Harburg, Sept. 1996.

[55] A. D. Gross. Toward object-based heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):794–802, Aug. 1994.

[56] A. D. Gross and T. E. Boult. Analyzing skewed symmetries. *International Journal of Computer Vision*, 13(1):91–111, 1994.

[57] A. D. Gross and T. E. Boult. Correction to “recovery of SHGCs from a single intensity view”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):471–479, Apr. 1996.

[58] A. D. Gross and T. E. Boult. Recovery of SHGCs from a single intensity view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18 (2):161–180, Feb. 1996. errata in [57].

[59] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, Cambridge, UK, second edition, 2003.

[60] R. I. Hartley and R. Kaucic. Sensitivity of calibration to principal point position. In *Proceedings of the Seventh European Conference on Computer Vision*, page 433 ff., Apr. 2002.

[61] P. V. C. Hough. Method and means for recognizing complex patterns. U. S. Patent No. 3069654, 1962.

[62] IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, New York City, New York, USA, June 1993. IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence, IEEE Computer Society Press, Los Alamitos, California.

[63] A. Imiya. A metric for spatial lines. *Pattern Recognition Letters*, 17:1265–1269, 1996.

[64] M. Irani and P. Anandan. Factorization with uncertainty. In D. Vernon, editor, *Proceedings of the sixth European Conference on Computer Vision*, volume 1842 of *Lecture Notes in Computer Science*, pages 539–553, Dublin, Ireland, June 2000. Springer, Berlin.

[65] G. James and R. C. James. *Mathematics Dictionary*. Chapman & Hall, New York, 5. edition, 1992.

[66] F. Jurie. Hypothesis verification in model-based object recognition with a gaussian error model. In H. Burkhardt and B. Neumann, editors, *Proceedings of the Fifth European Conference on Computer Vision*, volume 2 of *Lecture Notes in Computer Science*, pages 643–656, Freiburg, Germany, June 1998. Université Blaise-Pascal, Springer.

[67] B. Kamgar-Parsi, B. Kamgar-Parsi, and N. S. Netanyahu. A nonparametric method for fitting a straight line to a noisy image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):998–1000, Sept. 1989.

[68] K. Kanatani. *Group-Theoretical Methods in Image Understanding*, volume 20 of *Springer series in information sciences*. Springer-Verlag, Berlin, 1. edition, 1990.

[69] K. Kanatani. Computational projective geometry. *Computer Vision, Graphics and Image Processing: Image Understanding*, 54(3):333–348, Nov. 1991.

[70] K. Kanatani. Hypothesizing and testing geometric properties of image data. *Computer Vision, Graphics and Image Processing: Image Understanding*, 54 (3):349–357, Nov. 1991.

[71] K. Kanatani. *Geometric Computation for Machine Vision*. Number 37 in The Oxford Engineering Science Series. Clarendon Press, Oxford, 1993.

[72] K. Kanatani. Computational cross ratio for computer vision. *Computer Vision, Graphics and Image Processing: Image Understanding*, 60(3):371–381, Nov. 1994.

[73] K. Kanatani. Statistical analysis of geometric computation. *Computer Vision, Graphics and Image Processing: Image Understanding*, 59(3):286–306, May 1994.

[74] K. Kanatani. Statistical foundation for hypothesis testing of image data. *Computer Vision, Graphics and Image Processing: Image Understanding*, 60(3):382–391, Nov. 1994.

[75] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science, Amsterdam, The Netherlands, Apr. 1996.

[76] K. Kanatani. Geometric information criterion for model selection. *International Journal of Computer Vision*, 26(3):171–189, 1998.

[77] Y. Kanazawa and K. Kanatani. Optimal line fitting and reliability evaluation. *IEICE Transactions on Information & Systems*, E79–D(9):1317–1322, Sept. 1996.

[78] Q. Ke, T. Jiang, and S. De Ma. A tabu search method for geometric primitive extraction. *Pattern Recognition Letters*, 18:1443–1451, 1997.

[79] F. Klein. *Vergleichende Betrachtungen über neuere geometrische Forschungen*. Druck und Verlag von B. G. Teubner,Leipzig, Erlangen, 1872. Reprinted in Mathematische Annalen, 43 (1893).

[80] U. Köthe, P. Stelldinger, and H. Meine. Provably correct edgel linking and subpixel boundary reconstruction. In K. Franke, K.-R. Müller, B. Nickolay, and R. Schäfer, editors, *28. DAGM-Symposium Mustererkennung*, volume 4174 of *Lecture Notes in Computer Science*, pages 81–90, Berlin, Germany, Sept. 2006. DAGM, Springer-Verlag, Berlin, Heidelberg.

[81] E. Kreyszig. *Statistische Methoden und ihre Anwendungen*. Vandenhoeck & Ruprecht, Göttingen, 3. edition, 1968.

[82] A. Laurentini. Computing the visual hull of solids of revolution. *Pattern Recognition*, 32(3):377–388, Mar. 1999.

[83] Y. Lei and K. C. Wong. Detection and localisation of reflectional and rotational symmetry under weak perspective projection. *Pattern Recognition*, 32(2):167–180, Feb. 1999.

[84] T. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from an image. In B. Buxton and R. Cipolla, editors, *Proceedings of the Fourth European Conference on Computer Vision*, volume 1 of *Lecture Notes in Computer Science*, pages 546–555, Berlin, Heidelberg, New York, Apr. 1996. Springer.

[85] W. Li, G. Lu, and Y. Wang. Recognizing white line markings for vision-guided vehicle navigation by fuzzy reasoning. *Pattern Recognition Letters*, 18:771–780, 1997.

[86] B. Liao. *Ein Beitrag zur Kamerafokussierung bei verschiedenen Anwendungen der Bildverarbeitung*. Dissertation, Universität der Bundeswehr Hamburg, July 1993.

[87] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. EuroGraphics*, volume 18, pages 39–50, Sept. 1999.

[88] C. Lin, A. Huertas, and R. Nevatia. Detection of buildings from monocular images. In A. Grün, O. Kübler, and P. Agonnis, editors, *Ascona Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images*. Birkhäuser, 1995.

[89] C. Lin and R. Nevatia. Building detection and description from monocular aerial images. In *Image Understanding Workshop*. ARPA, 1996.

[90] S.-P. Liou and R. C. Jain. Road following using vanishing points. *Computer Vision, Graphics and Image Processing*, 39:116–130, 1987.

[91] J. Liu, J. Mundy, D. Forsyth, A. Zisserman, and C. Rothwell. Efficient recognition of rotationally symmetric surfaces and straight homogeneous generalized cylinders. In Proceedings of the Conference on Computer Vision and Pattern Recognition Proceedings of the Conference on Computer Vision and Pattern Recognition [62], pages 123–128.

[92] J. Lladós, H. Bunke, and E. Martí. Finding rotational symmetries by cyclic string matching. *Pattern Recognition Letters*, 18:1435–1442, 1997.

[93] E. Lutton, H. Maître, and J. Lopez-Krahe. Contribution to the determination of vanishing points using hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):430–438, Apr. 1994.

[94] M. J. Magee and J. K. Aggarwal. Determining vanishing points from perspective images. *Computer Vision, Graphics and Image Processing*, 26:256–267, 1984.

[95] K. V. Mardia. *Statistics of Directional Data*. Academic Press, London, 1972.

[96] J. Matas and J. Kittler. Junction detection using probabilistic relaxation. *Image and Vision Computing*, 11(4):197–202, May 1993.

[97] G. Mazzola, D. Krömker, and G. R. Hofmann. *Rasterbild — Bildraster. Anwendungen der Graphischen Datenverarbeitung zur geometrischen Analyse eines Meisterwerkes der Renaissance: Raffaels "Schule von Athen"*. Beiträge zur Graphischen Datenverarbeitung. Springer-Verlag, Berlin, 1987.

[98] J. C. McGlone and J. A. Shufelt. Projective and object space geometry for monocular building extraction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 54–61, Seattle, Washington, June 1994. IEEE Computer Society, IEEE Computer Society Press.

[99] G. F. McLean and D. Kotturi. Vanishing point detection by line clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11): 1090–1095, Nov. 1995.

[100] E. M. Mikhail. *Observations and Least Squares*. University Press of America, New York, 1976. With Contributions by F. Ackerman.

[101] D. P. Mukherjee, A. Zisserman, and J. M. Brady. Shape from symmetry—detecting and exploiting symmetry in affine images. In *Philosophical Transactions of the Royal Society of London, Series A*, volume 351, pages 77–106, 1995.

[102] D. P. Mukherjee, A. Zisserman, and M. Brady. Shape from symmetry – detecting and exploiting symmetry in affine images. Technical report, University of Oxford, Department of Engineering Science, 19 Parks Rd., Oxford OX1 3PJ, U.K., June 1993.

[103] J. L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. The MIT Press, Cambridge, Massachusetts, 1992.

[104] T. Nakamura, M. Asada, and Y. Shirai. A qualitative approach to quantitative recovery of SHGC's shape and pose from shading and contour. In Proceedings of the Conference on Computer Vision and Pattern Recognition Proceedings of the Conference on Computer Vision and Pattern Recognition [62], pages 116–122.

[105] H. Nakatani, S. Kimura, O. Saito, and T. Kitahashi. Extraction of vanishing point and its application to scene analysis based on image sequence. In *Proceedings of the Fifth International Conference on Pattern Recognition*, pages 370–372, Miami Beach, Florida, Dec. 1980. International Association for Pattern Recognition and theIEEE Computer Society.

[106] I. Overington. *Computer Vision: A unified, biologically-inspired approach*. Elsevier, Amsterdam, 1992.

[107] P. L. Palmer and A. T. Tai. An optimised vanishing point detector. In J. Illingworth, editor, *British Machine Vision Conference*, pages 529–538, Sheffield, UK, 1993. British Machine Vision Association Press.

[108] P. Parodi and G. Piccioli. A feature based recognition scheme for traffic scenes. In *Proceedings of Intelligent Vehicles Symposium 95*, pages 229–234, Detroit, 1995.

[109] P. Parodi and G. Piccioli. 3D shape reconstruction by using vanishing points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):211–217, Feb. 1996.

[110] P. Parodi and V. Torre. A linear complexity procedure for labelling line drawings of polyhedral scenes using vanishing points. In *Proceedings of the Fourth International Conference on Computer Vision*, 1993.

[111] P. Parodi and V. Torre. On the complexity of labelling perspective projections of polyhedral scenes, July 1993.

[112] B. Pasternak. Processing imprecise and structural distorted line drawings by an adaptable drawing interpretation kernel. In *Proc. of DAS-94: International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 349–365, Kaiserslautern, 1994.

[113] B. Pasternak, G. Gabrielides, and R. Sprengel. Wiz — a prototype for knowledge-based drawing interpretation. In F. Belli and F. J. Radermacher, editors, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 604 of *Lecture Notes in Computer Science*, pages 164–173, Paderborn, Germany, June 1992. Springer.

[114] B. Pasternak and B. Neumann. Adaptable drawing interpretation using object-oriented and constraint based graphic specification. In *Proceedings of the 2nd Int. Conference on Document Analysis and Recognition (ICDAR)*, pages 359–364, Tsukuba Science City, Japan, 1993.

[115] F. Pedersini, A. Sarti, and S. Tubaro. Estimation and compensation of subpixel edge localization error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1278–1284, 1997.

[116] X. Pennec. Toward a generic framework for recognition based on uncertain geometric features. *Videre: Online Journal of Computer Vision Research*, 1 (2):58–87, 1998.

[117] X. Pennec and J.-P. Thirion. A framework for uncertainty and validation of 3-D registration methods based on points and frames. *International Journal of Computer Vision*, 25(3):203–229, 1997.

[118] N. H. E. Pillow. *Recognition of Generalized Cylinders Using Geometric Invariance.* PhD thesis, Oriel College, Oxford, 1996.

[119] M. Pilu. Extraction of illusory linear clues in perspectively skewed documents. In *Proceedings of the Conference on Computer Vision and Pattern*

Recognition, volume I, pages 363–368, Kauai Marriott, Hawaii, Dec. 9–14 2001. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, California.

[120] J. Ponce. On characterizing ribbons and finding skewed symmetries. In *1989 IEEE International Conference on Robotics and Automation*, volume 1, pages 49–54, Washington, 1989. The Institute of Electrical and Electronics Engineers, Inc., IEEE Computer Society Press.

[121] J. Ponce, D. Chelberg, and W. B. Mann. Invariant properties of straight homogeneous generalized cylinders and their contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):951–966, Sept. 1989.

[122] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21(4):145–152, July 1987.

[123] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, UK, 3. edition, 1988.

[124] W. Puech, J.-M. Chassery, and I. Pitas. Cylindrical surface localization in monocular vision. *Pattern Recognition Letters*, 18:711–722, 1997.

[125] L. Quan and R. Mohr. Determining perspective structures using hierarchical hough transform. *Pattern Recognition Letters*, 9:279–286, May 1989.

[126] G. Roth and M. D. Levine. Geometric primitive extraction using a genetic algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):901–905, Sept. 1994.

[127] C. A. Rothwell. *Recognition Using Projective Invariance.* PhD thesis, University of Oxford, Oxford, U.K., July 1993.

[128] C. A. Rothwell, D. A. Forsyth, A. Zisserman, and J. Mundy. Extracting projective structure from single perspective views of 3D point sets. In *Proceedings of the Fourth International Conference on Computer Vision*, Berlin, Germany, May 1993. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, California.

[129] C. A. Rothwell, A. Zisserman, D. A. Forsyth, and J. L. Mundy. Canonical frames for planar object recognition. In G. Sandini, editor, *Proc. 2nd Europ. Conf. on Computer Vision*, volume 588 of *Lecture Notes in Computer Science*, pages 757–772, Berlin Heidelberg, 1992. Springer-Verlag. Also in [103, p. 228–251].

[130] K. B. Sarachik. The effect of gaussian error in object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 289–301, 1997.

[131] H. Sato and T. O. Binford. Builder-i: A system for the extraction of shgc objects in an edge image. In Image Understanding Workshop Image Understanding Workshop [37], pages 779–791.

[132] H. Sato and T. O. Binford. On finding the ends of SHGCs in an edge image. In Image Understanding Workshop Image Understanding Workshop [37], pages 379–388.

[133] H. Sato and T. O. Binford. Finding and recovering SHGC objects in an edge image. *Computer Vision, Graphics and Image Processing: Image Understanding*, 57(3):346–358, May 1993.

[134] F. Schaffalitzky and A. Zisserman. Planar grouping for automatic detection of vanishing lines and points. *Image and Vision Computing*, 18(9):647–658, June 2000.

[135] S. Se. Zebra-crossing detection for the partially sighted. In *Computer Vision and Pattern Recognition*, volume 2, page 2211, jun 2000.

[136] S. Se and M. Brady. Vision-based detection of kerbs and steps. In *British Machine Vision Conference*, pages 410–419, Essex, Sept. 1997.

[137] S. Se and M. Brady. Vision-based detection of staircases. In *Proc. Fourth Asian Conference on Computer Vision ACCV 2000*, volume 1, pages 535–540, Jan. 2000.

[138] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry.* Clarendon Press, Oxford, 1952.

[139] S. Shah and J. K. Aggarwal. Intrinsic parameter calibration procedure for a (high distortion) fish-eye lens camera with distortion model and accuracy estimation. *Pattern Recognition*, 29(11):1775–1788, 1996.

[140] L. S. Shapiro and M. Brady. Rejecting outliers and estimating errors in an orthogonal regression framework. Technical Report OUEL 1974/93, University of Oxford, Department of Engineering Science, 19 Parks Road, Oxford OX1 3PJ, Feb. 1993.

[141] X. Shen and P. Palmer. Uncertainty propagation and the matching of junctions as feature groupings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1381–1395, Dec. 2000.

[142] J. A. Shufelt. Performance evaluation and analysis of vanishing point detection techniques. In *Image Understanding Workshop*, Palm Springs, CA, Feb. 1996. ARPA, Morgan Kaufmann Publishers.

[143] H.-Y. Shum, R. Szeliski, S. Baker, M. Han, and P. Anandan. Interactive 3D modeling from multiple images using scene regularities. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments*, volume 1506 of *Lecture Notes in Computer Science*, pages 236–252, Freiburg, Germany, June 1998. Springer, Berlin. European Workshop, SMILE'98.

[144] C. C. Slama, C. Theurer, and S. W. Henriksen, editors. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, Va., 4. edition, 1980.

[145] N. W. Smirnow and I. W. Dunin-Barkowski. *Mathematische Statistik in der Technik*. VEB Deutscher Verlag der Wissenschaften, Berlin, 3. edition, 1973.

[146] C. E. Springer. *Geometry and Analysis of Projective Spaces*. W. H. Freeman and Company, San Francisco and London, 1964.

[147] M. Straforini, C. Coelho, and M. Campani. Extraction of vanishing points from images of indoor and outdoor scenes. *Image and Vision Computing*, 11 (2):91–99, Mar. 1993.

[148] M. Straforini, C. Coelho, M. Campani, and V. Torre. The recovery and understanding of a line drawing from indoor scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):298–303, 1992.

[149] A. Tai, J. Kittler, M. Petrou, and T. Windeatt. Vanishing point detection. In D. Hogg and R. Boyle, editors, *British Machine Vision Conference*, pages 109–118, School of Computer Studies, University of Leeds, Sept. 1992. British Machine Vision Association, Springer-Verlag, London.

[150] B. Tordoff and D. W. Murray. Guided sampling and consensus for motion estimation. In M. N. A. Heyden, G. Sparr and P. Johansen, editors, *Proceedings of the Seventh European Conference on Computer Vision*, volume 1 of *LNCS*, page 82 ff., Berlin, may 2002. Springer-Verlag, Berlin.

[151] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24(3):271–300, 1997.

[152] F. Ulupinar and R. Nevatia. Recovery of 3-D objects with multiple curved surfaces from 2-d contours. *Artificial Intelligence*, 67(1):1–28, May 1994.

[153] F. Ulupinar and R. Nevatia. Shape from contour: Straight homogeneous generalized cylinders and constant cross section generalized cylinders. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):120–135, Feb. 1995.

[154] L. Van Gool, M. Proesmans, and A. Zisserman. Grouping and invariants using planar homologies. In *Workshop on Geometrical Modeling and Invariants for Computer Vision*, Xi'an, China, Apr. 1995. Xidian University Press.

[155] L.-L. Wang and W.-H. Tsai. Camera calibration by vanishing lines for 3-D computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):370–376, Apr. 1991.

[156] G. S. Watson. *Statistics on Spheres.* Wiley-Interscience, New York, 1983.

[157] G.-Q. Wei and Z.-y. He. Determining vanishing point and camera parameter: New approaches. In *Proceedings of the Nineth International Conference on Pattern Recognition*, volume 1, pages 450–452, Rome, Italy, Nov. 1988. International Association for Pattern Recognition, IEEE Computer Society Press, Washington, D.C.

[158] I. Weiss. High-order differentiation filters that work. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):734–739, July 1994.

[159] R. S. Weiss, H. Nakatani, and E. M. Riseman. An error analysis for surface orientation from vanishing points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1179–1185, Dec. 1990.

[160] K.-Y. Wong, P. R. S. Mendonça, and R. Cipolla. Camera calibration from surfaces of revolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):147–161, Feb. 2003.

[161] K.-Y. K. Wong, P. R. S. Mendonça, and R. Cipolla. Reconstruction of surfaces of revolution from single uncalibrated views. In *British Machine Vision Conference*, pages 93–101, 2002.

[162] H. Zabrodsky, S. Peleg, and D. Avnir. Completion of occluded shapes using symmetry. In Proceedings of the Conference on Computer Vision and Pattern Recognition Proceedings of the Conference on Computer Vision and Pattern Recognition [62], pages 678–679.

[163] M. Zerroug and R. Nevatia. Volumetric descriptions from a single intensity image. *International Journal of Computer Vision*, 20(1–2):11–42, Oct. 1996.

[164] G. Zhou, E. Uzi, W. Feng, and B. Yuan. CCD camera calibration based on natural landmarks. *Pattern Recognition*, 31(11):1715–1724, 1998.

[165] A. Zisserman, D. Forsyth, J. Mundy, C. Rothwell, and J. Liu. 3D object recognition using invariance. Technical Report OUEL 2027/94, Robotics Research Group, University of Oxford, Parks Road, Oxford, June 1993.

Index