Camera self-calibration from corner correspondences

Shen, Fei

2005

Camera Self-Calibration From Corner Correspondences

Shen Fei

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirements for the degree of

Doctor of Philosophy

2005

Acknowledgement

I would like to express my sincere gratitude and appreciation to my supervisor, Associate Prof. Wang Han for his enthusiastic encouragement, excellent guidance and kind support throughout the duration of this research.

Special thanks are also dedicated to all the staff and colleagues in the Intelligent Machines Research Laboratory (IMRL) at the school of Electrical and Electronic Engineering (EEE) for their help and encouragement during the course of my study.

Finally, I would like to acknowledge the school of Electrical and Electronic Engineering (EEE) in Nanyang Technological University (NTU) and the University itself, for giving me an opportunity to pursue the Ph.D degree and to use their facilities.

Summary

This thesis studies the problem of camera self-calibration, which aims to obtain the camera calibration parameters using only the information of corner correspondences over an image sequence. The motivation for our work is to make it possible to obtain the metric reconstruction of a scene from a pre-recorded image sequence where neither camera calibration nor 3D scene information is available.

Our contribution is twofold. First, the corner detection problem is analyzed. As the corners are the main features used in our work, the self-calibration accuracy is directly affected by the performance of the corner detector. Two novel corner detectors are proposed. The first algorithm is an improved version of the well-known SUSAN corner detector. The SUSAN principle is extended to a general local region constraint and the local region is found by a new approach namely BDRAN instead of the traditional USAN approach. This results in a much more stable algorithm than SUSAN when applied on the real images. The second algorithm is based on the modified Hough transform. The local edges are detected dynamically and organized into simple lines by a modified Hough transform. A corner will be reported if no less than two simple lines exist. This algorithm can compensate some disadvantages of the first algorithm. Extensive experiments on both synthetic and real images are used to validate the proposed algorithms.

The other important contribution of our work is to develop a self-calibration algorithm by studying a new kind of restricted camera motion, i.e. small rotation plus

general translation. This is inspired by the successful self-calibration approaches for pure translation and pure rotation. Compared with the algorithms with general motion, the new algorithm is strictly linear and much more simple. Compared with the algorithms with pure translation and pure rotation, the required motion for the new algorithm is much less restrictive and can be easily satisfied at most situations. Experiments on both simulated data and real data validate the new algorithm.

KEY WORDS: Corner Detection, USAN, BDRAN, Hough Transform, Self-Calibration, General Motion, Restricted Motion, Small Rotation

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Reconstruction of three-dimensional (3D) scene from two-dimensional (2D) images is a traditionally important problem and has drawn a lot of attentions in the computer vision community. Although it is quite straightforward to project the 3D world onto the 2D image plane, the corresponding reverse projection is never a trivial problem. One key issue is the computation of camera calibration which is the precondition for an Euclidean reconstruction of the scene. Early works focused on calibrated views. In this case, camera calibration is obtained off-line before any vision task is performed. Although this approach achieves success in a certain degree, it fails in some scenarios. In the case of reconstruction from a pre-recorded image sequence, we always have no means to obtain the calibration information. Also, the camera might be zoomed out or zoomed in during operations. This will cause significant changes in the camera calibration and the off-line calibration will become invalid. Due to the above reasons, the attentions have been transferred to the uncalibrated views recently.

The creative work of Faugeras [22] and Hartley [33] has shown that it is possible to reconstruct a scene up to a projective transformation from two or more uncalibrated images. The feature correspondences are the only requirement for the projective reconstruction. It has been successfully applied in some typical computer vision applications, such as path planning for robot navigation [6], the grasping of objects by robotic arms [76], and the recognition of 3D objects [96]. However, with the rapid evolution of computer technology, today even personal computers can display complex 3D models. This results in a dramatically increasing demand for virtual reality systems which can virtualize existing objects or scenes. For such tasks, the virtual quality is the most important issue and projective structure is obviously not sufficient.

Self-calibration is a state-of-the-art technology which is most suitable to solve the Euclidean reconstruction problem from uncalibrated views. It provides a way to compute the camera calibration from images only, and then the projective reconstruction can be upgraded to the Euclidean one. The first approach was proposed by Maybank and Faugeras [46]. Since then, many different approaches have been proposed for self-calibration, and they will be reviewed in the main part of our thesis.

1.2 Objectives

Most existing self-calibration technologies are feature-based. Features are extracted from each view and they are matched across different views. Matched features correspond to the same 3D feature. It is clear that not all possible image features are suitable for this task. Often image points are preferred since they are most easily handled by other modules, but line segments [31] or other features (such as regions) can also be used. The selected feature points should contain as much

information as possible to allow matching. The **corner** is the ideal candidate for this task. In addition to the rich information they contain, corners are more likely to be independent of camera pose and illumination changes than other features. Many corner detectors exist and Schmid *et al.* [69] concluded that Plessey corner detector [28] gives the best detection result. But it has also been reported [17] that the localization performance of Plessey is not good, general corners are localized several pixels away from their true locations. This will seriously affect the self-calibration accuracy. Also the computational complexity of Plessey is high which makes it not suitable for real time applications. On the contrary, SUSAN corner detector [74] is well known for its computational efficiency and good localization performance. Unfortunately, it has been reported [79] that its detection performance is unstable in real applications. All the above shows that it is necessary to improve the current corner detectors to achieve a better tradeoff among detection, localization and speed. Thus corner detection analysis will be one focus of our thesis.

Another important issue is the self-calibration algorithm itself. Early approaches [54] [38] [80] [30] start from the projective calibration and then immediately try to solve for the intrinsic parameters. They all have the problem of coping with too many parameters at one time for nonlinear equations. A better way is the stratified approach. Before computing metric calibration, an affine calibration is computed first, or equivalently, the plane at infinity is located first in the projective space, which is observed to be the most difficult step. Till now no linear algorithms have been reported for general motions. Hartley [33] uses chirality inequalities to give a range of possible values for the location of plane at infinity. This is not an exact solution for affine calibration and suffers from the high risk that the accurate solution may not be found. Pollefeys *et al.* [59] uses the modulus constraint for affine calibration. In their algorithm, each pair of views with general motion in-between provides one polynomial equation of degree four in the coefficients of plane at infinity. In general, three equations should only leave no more than 64 solutions.

But it's not easy to solve the polynomial equations of high degree. Also the problem may arise that which solution should be chosen among those 64 potential solutions. However the problem can be simplified if a restricted motion is used. Very simple algorithms exist for the case of pure translation [51] [3] and pure rotation [35]. However, in practice, it is virtually impossible to ensure that the camera motion is a pure translation or a pure rotation, which is too restrictive to be satisfied in most situations. This limits the applications of these algorithms. Thus developing a simple algorithm for affine calibration under less restrictive motion is another goal of our thesis.

1.3 Main Contributions

This work mainly includes two aspects of contributions: one focuses on the corner detection part, which aims to increase the self-calibration accuracy, and the other focuses on the development of self-calibration algorithm, which tries to develop a simple and efficient algorithm for affine calibration. More specifically,

- We improve the SUSAN corner detector to give a much more stable corner detection algorithm. We analyze the stability performance of SUSAN corner detector and conclude that its instability is due to the violation of USAN constraint (SUSAN principle) by the edges of real images. To solve this problem, the USAN constraint is extended to the general local region constraint. By replacing USAN with a new local region definition namely BDRAN, the corresponding local region constraint holds very well with the real images and this results in a stable corner detection algorithm. Experiments on real images are presented to show the novelty of the newly developed approach.

- Although efficient enough in most situations, the above algorithm has some problems when dealing with some special corners, i.e. X-Junctions. To com-

pensate the potential problems, another new corner detection algorithm based on the modified Hough transform is developed. In this approach, the corner detection problem is simplified into organizing the local edges into a set of simple lines in a local coordinate system. A corner will be reported if no less than two simple lines exist. Local edges are detected based on both gradient magnitude threshold and gray level analysis. Extensive experiments are presented to compare it with other popular corner detectors.

- Inspired by the successful self-calibration approaches for pure translation and pure rotation, we propose a novel stratified self-calibration algorithm by studying a new kind of restricted camera motion, i.e. small rotation plus general translation, which is much less restrictive than two traditional restricted motions. Compared with the algorithms with general motion, the new algorithm is strictly linear and much simpler. Compared with the algorithms with pure translation and pure rotation, the required motion for the new algorithm is much less restrictive and can be easily satisfied at most situations. Experiments on both simulated data and real data will be used to validate the new self-calibration algorithm.

1.4 Thesis Outline

Chapter 2 introduces some important geometry concepts used in the rest of the text. First, projective geometry is introduced as the natural mathematical framework for the camera calibration problem. Then we introduce the single view geometry based on an ideal pinhole camera model. After that we introduce the two view geometry which relates the two images of a scene.

Chapter 3 studies the problem of relating two images of a same scene. The relationship between the two images can be fully described by the fundamental matrix.

First, we give a review to various corner detection algorithms. Then it will be shown how to match the detected corners automatically across two images. After that, a review will be given on the algorithms for the recovery of the fundamental matrix from the corner correspondences.

Chapter 4 proposes a novel corner detector which is modified from the well-known SUSAN corner detector. First, the stability performance of SUSAN is analyzed and it comes to the conclusion that the instability of SUSAN is due to the violation of USAN constraint, which is the principle of SUSAN, by the edges of real images. To solve this problem, the USAN constraint is extended to the general local region constraint. By replacing USAN with a new local region definition namely BDRAN, the corresponding local region constraint holds very well with the real images and this results in a stable corner detection algorithm. Experiments on real images will be presented to show the novelty of the newly developed approach.

Chapter 5 proposes another novel corner detection algorithm based on the modified Hough transform. The problem of detecting corners is simplified into detecting simple lines (straight lines that pass through the coordinate origin) in a local coordinate system. As simple lines can be expressed using only one parameter, Hough transform can detect them quickly. Other edge detectors have some problems when marking edge points around corners. Instead of using a general global edge detector, we detect edge points locally based on both gradient magnitude threshold and gray level analysis. The process is dynamic and can provide correct edge points for Hough transform. Gradient direction is used to reduce the effect of noise and speed up the algorithm. Experiments will be presented to compare the new algorithm with other popular corner detectors.

Chapter 6 gives an extensive review over existing self-calibration technologies assuming that the camera motion between sequential views is general. The self-calibration problem is stratified into three strata: projective calibration, affine calibration and

metric calibration. On each stage, representative approaches were briefed and analyzed.

Chapter 7 reviews the self-calibration algorithms with two traditional restricted motion, i.e. pure translation and pure rotation, and proposes a novel new algorithm by studying a new kind of restricted motion, i.e. small rotation plus general translation, which is much less restrictive compared with traditional restricted motions. The new algorithm is linear and efficient. Experiments on both simulated data and real data will be used to validate the new algorithm.

Chapter 8 concludes our work and identifies the directions for the future work.

Chapter 2

Camera Geometry

2.1 Introduction

This chapter reviews most of the geometry concepts used in the rest of the text and has three main parts. As projective geometry is the theoretical framework for the camera calibration work, we will first introduce some basic concepts of projective geometry. The $3D$ space is stratified into four levels from low to high: projective, affine, metric and Euclidean. The relations among different levels will be explored and some special geometry entities will be described. Then we will introduce the single view geometry based on an ideal pinhole camera model. Two important concepts are focused: the projection matrix which linearly maps a $3D$ world point to a $2D$ image point, and the homography which describes the transfer from a general world plane to the image plane. Lastly, we will introduce the two view geometry which relates the two images of a scene. Two important concepts are focused: the fundamental matrix which gives a necessary condition for two points of the two views to be corresponding points, and the image homography which describes the transfer from the first view to the second view for the points located on a world plane. The important relationships between the fundamental matrix and the image

homography will also be explored.

This chapter is organized as follows. Section 2.2 introduces the three-dimensional geometry. Section 2.3 introduces the single view geometry. Section 2.4 introduces the two view geometry. Section 2.5 concludes this chapter.

Notations:

P^n	the projective n-space
$\sim$	equality up to a nonzero scalar
$\times$	vector product
$[]_\times$	vector product in the matrix form
m	a general point $(x, y, w)^T$ in the $2D$ projective space
M	a general point $(X, Y, Z, W)^T$ in the $3D$ projective space
l	a general line
Π	a general plane
Ω	a general conic
Q	a general quadric
H	homography
H^T	the transpose of H
H^{-1}	the inverse of H (i.e. $HH^{-1} = I$)
T	a general projective transformation
T_A	a general affine transformation
T_M	a general metric transformation
T_E	a general Euclidean transformation
Π_∞	plane at infinity
Π_{REF}	reference plane
Ω_∞	absolute conic
P	camera projection matrix
K	camera calibration matrix
R	camera rotation matrix
t	camera translation vector
F	the fundamental matrix
$H_{\Pi I}$	the homography between a world plane Π and the image plane I
H_{12}^{Π}	the image homography between two views for the points of a world plane Π

2.2 Three-Dimensional Geometry

The world is usually represented in the Euclidean frame. But when we start from images, it is not possible for us to use the full Euclidean structure. It can be interesting to represent the world with certain simpler representations. These representations can be stratified into four levels from low to high: projective, affine, metric and Euclidean. Each level is closely related to a group of geometry transformations which give a structure ambiguity in the corresponding level. For example, the affine level is related with a group of affine transformations, and two sets of points related by an affine transformation are equivalent to each other. Also notice that the transformation groups are subgroups of each other, e.g. Euclidean group is a subgroup of metric group, and both are subgroups of affine group, and all are subgroups of projective group. Another important concept related to the representation levels are their invariants, which play a very important role to upgrade the current representation to a higher level of the stratification. All the above geometry concepts will be detailed in this part. Our knowledge of the $3D$ geometry comes from Pollefeys [56], Faugeras [19] [20], and Semple [71].

2.2.1 Three-Dimensional Projective Space

General Projective Space

Projective $n-$space, P^n, corresponds to the Euclidean $n-$space R^n. A point x of P^n is given by a $(n+1)-$vector of coordinates $(x_1 ... x_{n+1})^T$. At least one of these coordinates should differ from zero. These coordinates are called *homogeneous* coordinates. When $x_{n+1} \neq 0$, x relates to the point $(\frac{x_1}{x_{n+1}} ... \frac{x_n}{x_{n+1}})$ of R^n. Those points with coordinates $x_{n+1} = 0$ are said to be **at infinity**. x is only defined up to a non-zero scaling, such that for a non-zero λ, λx defines the same point as x. Conventionally, it is chosen that $x_{n+1} = 1$.

A **collineation** is a mapping between projective spaces, which preserves collinearity (i.e. collinear points are mapped to collinear points). A collineation from P^m to P^n is mathematically represented by a $(n+1) \times (m+1)$ matrix H. Points are transformed linearly: $x \mapsto x' \sim Hx$. Observe that matrices H and λH with λ a nonzero scalar represent the same collineation.

A **projective basis** is a set of $n+2$ points such that no $n+1$ of them are linearly dependent. The set $e_i = [0...1...0]^T$ where 1 is in the i-th position $(1 \leq i \leq n+1)$ and $e_{n+2} = [11...1]^T$ is the standard projective basis. A projective point of P^n can be described as a linear combination of any $n+1$ points of the standard basis. For example:

$$x = \sum_{i=1}^{n+1} x_i e_i \tag{2.1}$$

It can be shown that any projective basis can be transformed via a uniquely determined collineation into the standard projective basis. Similarly, if two sets of points $m_1, ...m_{n+2}$ and $m'_1, ..., m'_{n+2}$ both form a projective basis, then there exists a uniquely determined collineation T such that $m'_i \sim Tm_i$ for every i $(1 \leq i \leq n+2)$. This collineation T describes the change of projective basis. In particular, T is invertible.

Projective 2D Space

Projective 2D space P^2 corresponds to the Euclidean 2D plane R^2. A point of P^2 is represented by a 3-vector $m = [x, y, w]^T$ and is related with the point $(\frac{x}{w}, \frac{y}{w})$ of R^2 when $w \neq 0$. A line l is also represented by a 3-vector. A point m is located on a line l if and only if:

$$l^T m = 0 \tag{2.2}$$

This equation also means that the line l passes through the point m. This symmetry in the equation shows that there is no formal difference between points and lines in P^2. This is known as the principle of **duality**. A line passing through two points

m_1 and m_2 is given by their vector product $l \sim m_1 \times m_2$. By the dual formulation, the intersection of two lines l_1 and l_2 is given by $m \sim l_1 \times l_2$.

A **conic** in P^2 is the locus of all points m satisfying a homogeneous quadratic equation:

$$m^T C m = 0 \tag{2.3}$$

where C is a 3×3 symmetric matrix only defined up to scale. A conic has 5 degrees of freedom.

Projective 2D Transformation

Transformations in the projective $2D$ space are represented by **homographies** of $P^2 \mapsto P^2$. A homography is represented by a 3×3 matrix H. Again H and λH represent the same homography for all nonzero scalars λ. A point is transformed as follows:

$$m \mapsto m' \sim Hm \tag{2.4}$$

Consider a line l that passes through m. As $l^T m = 0$ and $H^{-1}H = I$, we can have the following (with $H^{-T} = (H^{-1})^T = (H^T)^{-1}$):

$$l^T m = l^T H^{-1} H m = l^T H^{-1} m' = (H^{-T} l)^T m' = 0 \tag{2.5}$$

As any line l' that passes through m' satisfies $l'^T m' = 0$, equation (2.5) gives the transformation for a line l as:

$$l \mapsto l' \sim H^{-T} l \tag{2.6}$$

The transformation for a conic under a homography H can be obtained in the similar way. Using equations (2.3) and (2.4), we can have the following:

$$m^T C m = m^T H^T H^{-T} C H^{-1} H m = m'^T (H^{-T} C H^{-1}) m' = 0 \tag{2.7}$$

As any conic that contains m' satisfies $m'^T C' m' = 0$, equation (2.7) gives the transformation for a conic C as:

$$C \mapsto C' \sim H^{-T} C H^{-1} \tag{2.8}$$

Projective 3D Space

Projective 3D space P^3 corresponds to the Euclidean 3D space R^3. A point of P^3 is represented by a 4-vector $M = [X, Y, Z, W]^T$ and is related with the point $(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W})$ of R^3 when $W \neq 0$. In P^3 the dual entity of a point is a plane, which is also represented by a 4-vector. A point M is located on a plane Π if and only if

$$\Pi^T M = 0 \tag{2.9}$$

A line can be given by the linear combination of two points $\lambda_1 M_1 + \lambda_2 M_2$ or by the intersection of two planes $\Pi_1 \cap \Pi_2$.

A **quadric** is the locus of all points M satisfying a homogeneous quadratic equation:

$$M^T Q M = 0 \tag{2.10}$$

where Q is a 4×4 symmetric matrix only defined up to scale. A quadric has 9 degrees of freedom.

Projective 3D Transformation

A transformation in the projective $3D$ space is represented by a 4×4 invertible

matrix T:

$$T \sim \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \tag{2.11}$$

T is only defined up to a nonzero scale factor and has therefore 15 degrees of freedom. Similar to the projective $2D$ transformations, the following equations can be obtained for transformations of points, planes and quadrics in $3D$ space:

$$M \mapsto M' \sim TM \tag{2.12}$$

$$\Pi \mapsto \Pi' \sim T^{-T}\Pi \tag{2.13}$$

$$Q \mapsto Q' \sim T^{-T}QT^{-1} \tag{2.14}$$

2.2.2 Affine Stratum

The affine stratum is a subset of the projective space whose associated transformations are affine ones:

$$T_A \sim \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A & a \\ 0 & 1 \end{bmatrix} \tag{2.15}$$

where A is a 3×3 matrix and a is a 3-vector. Obviously, T_A has 12 degrees of freedom. Any $3D$ point $M = [X, Y, Z, 1]^T$ will be transformed to $M' = [X', Y', Z', 1]^T$ by an

affine transformation as:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + a \tag{2.16}$$

2.2.3 Metric Stratum

The metric stratum is a subset of the affine space whose associated transformations are metric ones (i.e. rotation, translation and scaling):

$$T_M \sim \begin{bmatrix} \sigma r_{11} & \sigma r_{12} & \sigma r_{13} & t_X \\ \sigma r_{21} & \sigma r_{22} & \sigma r_{23} & t_Y \\ \sigma r_{31} & \sigma r_{32} & \sigma r_{33} & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \sigma \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.17}$$

where σ is the scaling, t is the translation vector and R is the rotation matrix, which satisfies $RR^T = I$ and $\det(R) = 1$. A rotation matrix only has 3 degrees of freedom. Thus a metric transformation therefore has 7 degrees of freedom, 3 for rotation, 3 for translation and 1 for scaling. Any $3D$ point $M = [X, Y, Z, 1]^T$ will be transformed to $M' = [X', Y', Z', 1]^T$ by a metric transformation as:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \sigma \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix} = \sigma R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{2.18}$$

2.2.4 Euclidean Stratum

The Euclidean stratum is a subset of the metric space whose associated transformations are Euclidean ones (i.e. rotation and translation):

$$T_E \sim \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.19}$$

The Euclidean stratum does not differ much from metric stratum. The difference is that the scale is fixed. Euclidean transformations have 6 degrees of freedom, 3 for orientation and 3 for translation. Any $3D$ point $M = [X, Y, Z, 1]^T$ will be transformed to $M' = [X', Y', Z', 1]^T$ by an Euclidean transformation as:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{2.20}$$

Although Euclidean structure is the way to represent the world, it is only available when the absolute yardstick is available and this condition cannot be satisfied when we start from the images. Thus the metric representation of the world is the highest level that we can recover from the images. For this reason, the Euclidean stratum will not be discussed further in our work.

2.2.5 Special Geometry Entities and Invariants

This part discusses some special geometry entities and their properties in each geometry stratum. They will play an important role in the camera calibration task.

Plane at Infinity

As stated, a point of P^3 with coordinates $(X, Y, Z, 0)^T$ is said to be at infinity and will be represented by M_∞. All the points at infinity form a special plane which shall be known as the **plane at infinity** Π_∞ whose canonic coordinates are given by $[0001]^T$.

An arbitrary projective transformation T_P changes Π_∞ from its canonic position to:

$$\Pi_\infty{}' \sim T_P^{-T}\Pi_\infty \sim [\pi_\infty^T, 1]^T \tag{2.21}$$

where π_∞ is a 3-vector determined by T_P. This shows that, in a specific projective representation, Π_∞ can be anywhere instead of its canonical position.

An arbitrary affine transformation T_A changes Π_∞ from its canonic position to:

$$\Pi_\infty{}' \sim T_A^{-T}\Pi_\infty \sim \begin{bmatrix} A^{-1} & 0 \\ -a^T A^{-T} & 1 \end{bmatrix} \begin{bmatrix} 0_3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0_3 \\ 1 \end{bmatrix} = \Pi_\infty \tag{2.22}$$

It shows that Π_∞ always has the coordinates $[0001]^T$ and is invariant under the affine transformation. Note that the position of a specific point at infinity might change, but it will still stay within the plane at infinity. As metric transformation is a special case of affine transformation, Π_∞ is also invariant under the metric transformation.

Reference Plane

In the following parts of our thesis, we will use the **reference plane** Π_{REF} to represent a special plane with coordinates $[0001]^T$. Due to its simple form, different views are more easily to be related by Π_{REF}. It is easy to check that the reference plane is exactly the plane at infinity in both affine and metric stages.

Absolute Conic

The **absolute conic** Ω_∞ is a special conic which is an imaginary circle located in

the plane at infinity Π_∞. It has the equation:

$$\Omega_\infty : X^2 + Y^2 + Z^2 = 0 \text{ and } W = 0 \tag{2.23}$$

Note that the absolute conic is a $3D$ entity. But when only the plane at infinity is under consideration, Ω_∞ is a $2D$ entity which can be represented by:

$$\Omega_\infty \sim \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \sim I \tag{2.24}$$

where I is the 3×3 identical matrix. In our work, without further explanation, Ω_∞ will be considered as a $2D$ conic by default.

Let $M_\infty \sim [X, Y, Z, 0]^T$ be a point at infinity, then the corresponding $2D$ point in the plane at infinity can be parameterized as $m_\infty \sim [X, Y, Z]^T$. Applying an affine transformation T_A to M_∞ results in $m_\infty \to m'_\infty \sim Am_\infty$. This shows that the affine transformation affects the points of Π_∞ as a $2D$ homography $H \sim A$ when only Π_∞ is under consideration. Similarly, applying a metric transformation to M_∞ results in $m_\infty \to m'_\infty \sim \sigma R m_\infty$. This shows that the metric transformation affects the points of Π_∞ as a $2D$ homography $H \sim \sigma R$ when only Π_∞ is under consideration.

The above analysis and the equation (2.8) show that an affine transformation T_A changes Ω_∞ from its canonic position to (note that I is the 3×3 identical matrix):

$$\Omega_\infty \to \Omega'_\infty \sim A^{-T}\Omega_\infty A^{-1} \sim A^{-T} I A^{-1} \sim A^{-T}A^{-1} \tag{2.25}$$

This shows that, in a specific affine representation, the absolute conic can be changed to any general conic instead of its canonical form.

Similarly, a metric transformation T_M changes Ω_∞ from its canonic position to (note

that σ is a scaling and R is a rotation matrix which satisfies $R^T R = I$):

$$\Omega_\infty \rightarrow \Omega'_\infty \sim \sigma R^{-T} \Omega_\infty \sigma R^{-1} \sim R^{-T} I R^{-1} \sim R^{-T} R^{-1} \sim I \sim \Omega_\infty \tag{2.26}$$

This shows that the absolute conic is invariant under the metric transformation.

Invariants

If we use the **invariant** to represent a property of a configuration of geometric entities that is not altered by any transformation belonging to a specific stratum, the plane at infinity is an invariant of the affine stratum, and it together with the absolute conic are invariants of the metric stratum. There are much more invariants associated with each stratum of the $3D$ geometry.

The cross-ratio is an invariant with the projective representation. Assume four points M_1, M_2, M_3 and M_4 are collinear. They can be expressed as $M_i = M + \lambda_i M'$ (i=1,2,3,4). The cross-ratio is defined as:

$$\{M_1; M_2; M_3; M_4\} = \frac{\lambda_1 - \lambda_3}{\lambda_1 - \lambda_4} : \frac{\lambda_2 - \lambda_3}{\lambda_2 - \lambda_4} \tag{2.27}$$

The cross-ratio is not dependent on the choice of M and M' and is invariant under any projective transformation of P^3. Other invariants of projective stratum include relation of incidence, collinearity and tangency.

All invariants of the projective stratum are *a fortiori* affine invariants. The plane at infinity is also an affine invariant and it results in more affine invariants. If lines or planes having their intersection in the plane at infinity are called parallel, parallelism is added as a new invariant property of the affine stratum. The ratio of lengths along a certain direction is also a new invariant and this is equivalent to a cross-ratio with one of the points at infinity.

All invariants of the affine stratum are a fortiori metric invariants. The absolute

conic is also a metric invariant and it results in more metric invariants such as relative distances and angles.

2.3 Single View Geometry

In this part, two important geometry concepts involved in the image formation process will be introduced in detail. The first one is the **projection matrix** of a camera which maps the $3D$ world points to the $2D$ image points. The second one is the **homography** which describes the transfer of the points from a world plane to the image plane. The ideal pinhole camera model is used in our work and we do not consider any nonlinear distortion introduced by the real camera. The knowledge of this section comes from the book of Zhang and Xu [95].

2.3.1 Projection Matrix

The ideal pinhole camera is a perspective projection from $3D$ projective world space to the $2D$ projective image plane I. Such a process can be illustrated as fig.2.1 and is completely determined by the optical center C and the corresponding image plane. Use $M = [X, Y, Z, 1]^T$ and $m = [u, v, 1]^T$ to represent the homogeneous coordinates of a $3D$ world point and its image respectively. Their relationship can be described by a 3×4 matrix P as following:

$$m \sim PM \tag{2.28}$$

where P shall be known as the projection matrix of the camera. It is defined up to a nonzero scalar and is of rank 3, therefore it has 11 degrees of freedom.

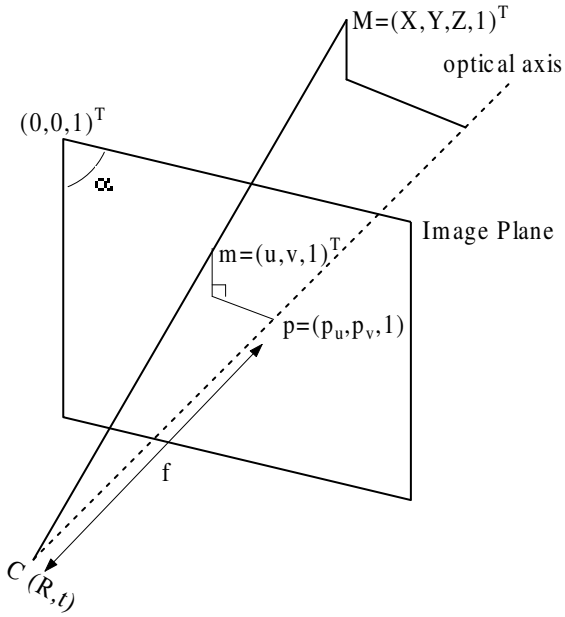


Figure 2.1: The pinhole camera model which maps a $3D$ world point M to a $2D$ image point m. C is the optical center whose position is described by a rotation R and a translation t. $p = [p_u, p_v, 1]^T$ is the principal point which is the intersection of the optical axis and the image plane. f is the focal length measured in mm. α is the angle between image axes.

The projection matrix P can be further decomposed as:

$$P = K[R|t] \tag{2.29}$$

where R and t are the rotation matrix and translation vector of the camera relative to the world coordinate system, encoding the extrinsic parameters of the camera; the 3×3 upper triangular matrix K is the camera calibration matrix, encoding five intrinsic parameters of the camera as:

$$K = \begin{pmatrix} f_u & s & p_u \\ 0 & f_v & p_v \\ 0 & 0 & 1 \end{pmatrix} \tag{2.30}$$

The physical meanings of five intrinsic parameters are (see fig.2.1):

f_u, f_v The focal length f measured in pixel units along u and v directions respectively. Use q_u and q_v to represent the width and height of a pixel measured in mm, we can have $f_u = \frac{f}{q_u}$ and $f_v = \frac{f}{q_v}$.

p_u, p_v The image coordinates of the principal point p, which is the intersection of the optical axis and the image plane.

s The skew parameter which accounts the effect of non-rectangular pixels.. Use α to represent the angle between u and v axes, we can have $s = f_u \cos\alpha$

Sometimes, instead of expressing the focal length with two different measurements f_u and f_v, we use an alternative expression as f_u and ζf_u. The aspect ratio $\zeta = \frac{q_u}{q_v}$ is the ratio of the pixel width q_u and the pixel height q_v.

2.3.2 Homography from World Plane to Image Plane

A homography H describes the collineation from $P^2 \rightarrow P^2$. Since the image is also a plane, there exists a homography $H_{\Pi I}$ which transforms the points of a general plane $\Pi \sim [\pi^T 1]^T$ to the points of the image plane I. If $M_\Pi \sim [m_\Pi^T 1]^T$ to represent a point of Π, it satisfies that $0 = \Pi^T M_\Pi = \pi^T m_\Pi + 1$. Hence:

$$M_\Pi \sim \begin{bmatrix} m_\Pi \\ 1 \end{bmatrix} = \begin{bmatrix} m_\Pi \\ -\pi^T m_\Pi \end{bmatrix} = \begin{bmatrix} I_{3\times 3} \\ -\pi^T \end{bmatrix} m_\Pi \tag{2.31}$$

If the camera projection matrix $P = [A|a]$, the projection $m_{\Pi I}$ of M_Π on the image can be given by:

$$m_{\Pi I} \sim P M_\Pi = [A|a] \begin{bmatrix} I_{3\times 3} \\ -\pi^T \end{bmatrix} m_\Pi = [A - a\pi^T] m_\Pi \tag{2.32}$$

This shows that the homography $H_{\Pi I}$ is related with the projection matrix by:

$$H_{\Pi I} \sim A - a\pi^T \tag{2.33}$$

2.4 Two View Geometry

In this part, we will explore some important relationships between two images of a same scene. The first one is the **fundamental matrix**. Given a point in the first image, the fundamental matrix constrains the position of the correspondence point in the other one. The second one is the **image homography**, which describe the transfer from one image to the other for the points located on a specific plane. The relationship between the fundamental matrix and the image homography will also be studied. The knowledge of this section comes from the book of Zhang and Xu [95].

2.4.1 Fundamental Matrix

Consider the case that a world point M is viewed by two cameras shown as fig.2.2. Let C_1 and C_2 be the optical centers of these two cameras respectively. m_1 is the projection of M on the first image plane I_1. Although the exact position of M cannot be known from m_1 alone, it is bounded to be on the line of sight C_1m_1. Notice that a unique plane L is defined by the points C_1, C_2 and m_1. Assume L intersects the second image plane I_2 at the line l_2, it is easy to check that all the points on the line C_1m_1 have their image points to lie on l_2 in I_2. This means that the correspondence of m_1 is bound to be on l_2. Similarly, assume L intersects I_1 at l_1, the correspondence of m_2 in the first image is bound to be on l_1. Such a relationship between two corresponding points in two views is known as the **epipolar constraint**: the correspondence of every point on l_1 is located on l_2 and vice verse.

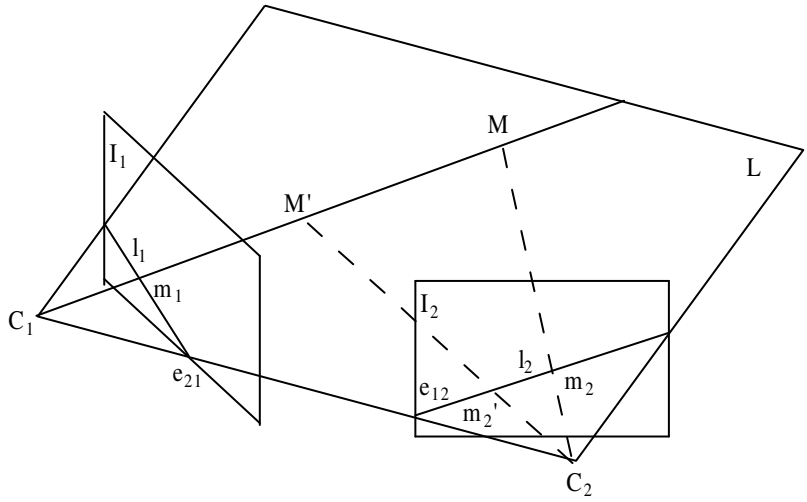


Figure 2.2: Epipolar constraint: the correspondence of every point on l_1 is located on l_2 and vice verse.

In the following parts, L will be called the **epipolar plane**, and l_1 and l_2 will be called the **epipolar lines** of m_2 and m_1 respectively. Different epipolar planes will correspond to different epipolar lines in the first image, but all of them will intersect at a same point e_1, which is the intersect of the line C_1C_2 and I_1. e_1 is called the **epipole** of the first view, which is actually the projection of C_2 on I_1. Similarly, the epipole e_2 also exists in the second view, which is the projection of C_1 on I_2.

Now we will show how to express the epipolar constraint mathematically. Use $P_1 = K_1[R_1|t_1]$ and $P_2 = K_2[R_2|t_2]$ to represent the projection matrices of these two cameras respectively. Without loss of generality, we can assume that the world coordinate system is aligned with the first camera coordinate system, which means $R_1 = I$ and $t_1 = 0$. We have the following equations for the projections of M on the two images:

$$\begin{aligned} m_1 &\sim K_1[I|0]M \\ m_2 &\sim K_2[R_2|t_2]M \end{aligned} \tag{2.34}$$

Eliminating M from the above two equations, we can obtain the following equation:

$$m_2^T K_2^{-T} [t_2]_\times R_2 K_1^{-1} m_1 = 0 \tag{2.35}$$

where $[]_\times$ represents the vector product in the matrix form and $x \times y = [x]_\times y$ holds for any vectors x and y. For $x = [x_1, x_2, x_3]^T$, $[x]_\times$ is an asymmetric matrix:

$$[x]_\times = \begin{bmatrix} 0 & x_3 & -x_2 \\ -x_3 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{bmatrix} \tag{2.36}$$

Let $F = K_2^{-T} [t_2]_\times R_2 K_1^{-1}$, equation (2.35) becomes:

$$m_2^T F m_1 = 0 \tag{2.37}$$

This equation is a necessary condition that m_1 and m_2 are corresponding points. F is known as the fundamental matrix. As F is defined up to a non-zero scalar and $det(F) = 0$ (this is due to $det([t_2]_X) = 0$), F has seven degrees of freedom. When the camera calibrations are unknown, F can be computed from point correspondences (at least 7 correspondences) of the two images based on equation (2.37).

Geometrically, Fm_1 defines the epipolar line l_2 of point m_1 in the second image. The epipoles e_2 can be computed from F as follows. As $l_2 \sim Fm_1$ and it contains e_2, we can have $e_2^T F m_1 = 0$. This equation holds for all possible m_1, thus we can have:

$$e_2^T F = 0 \tag{2.38}$$

This equation shows that e_2 is the left null-space of F. Similarly, $F^T m_2$ defines the epipolar line l_1 of point m_2 in the first image and the epipole e_1 satisfies:

$$F e_1 = 0 \tag{2.39}$$

which means that e_1 is the right null-space of F.

2.4.2 Image Homography

If we use $H_{\Pi 1}$ ($H_{\Pi 2}$) to represent the homography between a general plane Π and the first (second) view, it is possible to define an image homography H_{12}^{Π} which describes the transfer from the first view to the second view for points located on Π. The image homography is given by the relation of $H_{12}^{\Pi} = H_{\Pi 2} H_{\Pi 1}^{-1}$. From equation (2.33), we can have:

$$H_{12}^{\Pi} = (A_2 - a_2 \pi^T)(A_1 - a_1 \pi^T)^{-1} \tag{2.40}$$

where A_i and a_i are the first 3×3 and the last 3×1 sub-matrices of P_i respectively $(i = 1, 2)$.

2.4.3 Relation between Fundamental Matrix and Image Homography

There exists an important relationship between the image homographie H_{12}^{Π} and the fundamental matrix F. For a point m_1 in the first image, $m_2 \sim H_{12}^{\Pi} m_1$ is the corresponding point in the second image for the plane Π. m_1 and m_2 should satisfy the epipolar constraint, which means:

$$(H_{12}^{\Pi} m_1)^T F m_1 = 0 \tag{2.41}$$

The above equation holds for every possible m_1. Also, the epipolar line l_2 is given by either Fm_1 or $[e_2]_X m_2$, we can have $Fm_1 \sim [e_2]_X m_2$, which makes the following satisfied: $m_2^T [e_2]_X H_{12}^{\Pi} m_1 = 0$. Comparing with $m_2^T F m_1 = 0$, we can have the following relationship:

$$F \sim [e_2]_X H_{12}^{\Pi} \tag{2.42}$$

Consider a special line e_2 in the second image. As $e_2^T e_2 \neq 0$, the line e_2 does not contain the epipole e_2 and thus does not coincide with any epipolar line of the second image. C_2 and e_2 determine a unique plane Π_e in the space. Assume M is a $3D$ point in this plane and m_i $(i = 1, 2)$ are the images in the two views respectively. We can have the equation $m_2 \sim [e_2]_X F m_1$ as m_2 is the intersection of the line e_2 and the epipolar line Fm_1 of m_1. Therefore the image homography for the plane Π_e is given by:

$$H_{12}^{\Pi_e} \sim [e_2]_X F \tag{2.43}$$

The above equation is important because the image homography for a specific plane is available from the fundamental matrix and the epipole information. This is always used to set up a projective frame for all the views in the projective calibration stage.

2.5 Conclusion

In this chapter some important geometry concepts were introduced. We first introduced some basic concepts of projective geometry. The $3D$ space is stratified into four levels from low to high: projective, affine, metric and Euclidean. The relations among different levels were explored and some special geometry entities were described. Then we introduced the single view geometry based on an ideal pinhole camera model. Two important concepts were focused: the projection matrix which linearly maps a $3D$ world point to a $2D$ image point, and the homography which describes the transfer from a general world plane to the image plane. Lastly, we introduced the two view geometry which relates the two images of a scene. Two important concepts were focused: the fundamental matrix which gives a necessary condition for two points of the two views to be corresponding points, and the image homography which describes the transfer from the first view to the second view for the points located on a world plane. The important relationships between the

fundamental matrix and the image homography were also explored.

Chapter 3

Relating Images

3.1 Introduction

Given a set of images of the same scene, the first important step is to recover the geometry relationships among these images. Consider the simplest case of a stereo rig, which consists of two images of the same scene. Their relationship is fully described by the fundamental matrix. When the camera parameters are unknown, the two images need to be matched first, and then the fundamental matrix can be computed from the feature correspondences.

Over the years, numerous algorithms for image matching have been proposed. Although some work makes use of the template matching [26] [24] [27] [13], most of them are based on the features, in which a number of features are selected first in both views, and followed by a matching process [5] [12] [39] [45] [61] [72] [88] [16]. It is clear that not all possible image features are suitable for matching. Often points are used since they are most easily handled by other modules, but other features, i.e. lines, contours or regions, can also be matched [15] [70]. In this work, corners are chosen as the features used for matching and further process. Compared

with other features, corners possess some attractive properties which make them especially suitable for the matching problem. First, corners contain rich information. Image corners always correspond to the object vertexes of the 3D world, thus they can keep the most structural information of the scene. Also a corner is always located far away (typically more than 10 pixels) from other corners. This is very helpful to solve the matching ambiguity problem. Furthermore, as corners are two-dimensional features, some corner properties, i.e. angles and orientations, can guide the matching process.

After the corner correspondences over two views are available, they can be used to compute the fundamental matrix. As the fundamental matrix has 7 degrees of freedom, in theory 7 pairs of corner correspondences are sufficient to give an exact solution [40] [78]. When more matches are given, the solution can be found by the methods of least square or the nonlinear approaches [91] [44] [43] [32] [9] [53]. In fact the corner matching is often tightly coupled with the recovery of fundamental matrix. Since the correspondence problem is an ill-posed problem, the set of matched corners can be contaminated with an important number of wrong matches or outliers. In this case, hypothetical matches are used to compute the epipolar geometry by robust approaches [94] [92] [77]. These techniques use robust techniques like RANSAC [23] or LMedS [68] and feedback the results to the matcher to obtain more accurate matches, which are then used to refine the epipolar geometry.

This chapter gives a review over the above mentioned technologies. This chapter is organized as follows. In section 3.2, the corner detection algorithms will be reviewed. In section 3.3, we briefly explain how the corners can be matched. In section 3.4, the algorithms for the recovery of fundamental matrix from corner correspondences are reviewed. In section 3.5, we will show how the corner correspondences and the fundamental matrix can be refined. Section 3.6 concludes this chapter.

Notations:

(u, v)	the image coordinates of a pixel
$I(u, v)$	the gray level intensity of a pixel
I_u	the image partial derivative along the u direction
I_v	the image partial derivative along the v direction
I_{uu}	the second-order image partial derivative along the u direction
I_{vv}	the second-order image partial derivative along the v direction
I_{uv}	the second-order image partial derivative along both directions
∇I	the gradient of the image
$\|\nabla I\|$	the gradient magnitude of the image
θ	the gradient direction of the image
$\triangle$	the cornerness measure of a corner detector
$G(\sigma)$	Gaussian smoothing function with standard deviation σ
F	the fundamental matrix

3.2 Corner Detection

It's hard to define the term **corner** as there are many kinds of corners in real applications. Ideally, we would like to treat all the pixels in the 2D image which human would associate with the word "corner" as corners. This definition is so fuzzy that it is suitable for all situations. A more practical definition would be the point of maximal planar curvature in the line of the steepest gray level slope [63]. This definition describes the differential geometry properties of the corners. Another definition is more intuitional: the junction point of two or more straight-line edges. Based on this definition, two characteristics of corners can be generalized: 1) a corner is also an edge; 2) at least two straight lines pass through the corner.

Given an image, the earliest approaches involve first extracting edges as a chain code, and then searching for points having maximal curvature [4] [17] [49] or performing a polygonal approximation on the chains and then searching for the line segment intersections [39]. These techniques rely heavily on the performance of the prior segmentation of the shape, and might be led astray by errors in the segmentation. More recently, people showed more interest on the approaches which can be applied directly to gray level images. Template-based corner detectors are the most intuitive approaches [73] [62]. Ideal corners are assumed to exist in the images. Developing a template model for the ideal corner and determining the similarity, or correlation, between the template and all sub-windows in a given image, we can mark the points with local maximal similarity as corners. The idea is very intuitive but not practical. First, even L-Junctions have various angles and orientations. It's impossible to develop templates to cover all the situations. What's more, corners that have more complicated structures, i.e. T-Junction or X-junction, are very common in real applications. It's also very hard to develop templates to cover these complex corner types. Second, even if we assume that twelve masks can cover any corner having an arbitrary angle and orientation, the high computational cost makes the algorithm

almost useless in real time applications. Third, the template-based methods are very sensitive to strong edges because of the digitization effect when generating an image. So the template-based corner detectors are too limited for real applications and have been abandoned by most of the researchers.

Nowadays, the mainly used corner detection algorithms can be roughly classified into three catalogers: geometry-based corner detector, auto-correlation corner detector and structure-based corner detector. Geometry-based approaches consider the gray level intensities in a given neighborhood to be discrete quantized noisy samples from an underlying continuous gray intensity surface. They aim to analyze the differential geometry properties of the corner model from that continuous surface. Many famous algorithms, such as Kitchen-Rosenfeld corner detector [41], Wang-Brady corner detector [86] [85] and Deriche-Giraudon corner detector [17], all belong to this category. Auto-correlation corner detectors measure the intensity correlation between the local window and the neighborhood window to estimate the intensity changes under small window shifts. The representative approach is the well-known Plessey corner detector [28]. Structure-based corner detectors analyze the structure of the local window. Points at flat area, edge points and corner points are shown to have different structures around them. This observation can be used to distinguish these three kinds of features effectively. The most famous approach in this category will be the SUSAN corner detector [74].

In the following parts, we will use $I(u, v)$ to denote the gray level intensity of a pixel with coordinates $(u, v)^T$. Also, I_u and I_v will be used to approximate the partial derivatives of the image, and I_{uu}, I_{vv} and I_{uv} will be used to approximate the second-order partial derivatives of the image. The gradient of the intensity is defined as $\nabla I = (I_u, I_v)^T$. The gradient direction is defined as $\tan\theta = \frac{I_v}{I_u}$, and the gradient magnitude is defined as $|\nabla I| = \sqrt{I_u^2 + I_v^2}$. Also, we will use cornerness measure $\triangle$ to represent the corner response of a certain pixel by a certain corner

detection algorithm.

3.2.1 Geometry-based Corner Detectors

This type of corner detectors considers the gray level intensities in a given neighborhood to be discrete quantized noisy samples from an underlying continuous gray intensity surface. A smooth function $I(x, y)$ is fitted to the image neighborhood, then the differential geometry properties of $I(x, y)$ can be analyzed. The knowledge of differential geometry can be obtained from [14] [18].

Beaudet Corner Detector

Perhaps the first corner detector based on differential geometry analysis is the Beaudet corner detector [8]. The cornerness measure used in their method is the determinant of the Hessian matrix:

$$\triangle_{Beaudet} = I_{uu}I_{vv} - I_{uv}^2 \tag{3.1}$$

This corner operator is also named as DET. The reason for choosing this operator is that DET is equivalent to the product of the Gaussian curvature $k_{max}k_{min}$ and a term that is an increasing function of the gradient magnitude:

$$\triangle_{Beaudet} = (k_{max}k_{min}) \cdot (1 + I_u^2 + I_v^2)^2 \tag{3.2}$$

Essentially, corners are detected at points where both the curvature and the gradient magnitude are high. Beaudet corner detector is a rotational invariant operator, so it is quite stable. Unfortunately, it cannot localize corner positions well.

Kitchen-Rosenfeld Corner Detector

Kitchen and Rosenfeld proposed a cornerness measure based on the change of gradi-

ent direction along an edge contour multiplied by the local gradient magnitude [41]. Consider the gradient direction θ. The partial derivatives of θ are given by:

$$\theta_u = \frac{I_{uv}I_u - I_{uu}I_v}{I_u^2 + I_v^2} \tag{3.3}$$

$$\theta_v = \frac{I_{vv}I_x - I_{uv}I_v}{I_u^2 + I_v^2} \tag{3.4}$$

Projecting the vector of the change of gradient direction $(\theta_u, \theta_v)^T$ along the edge direction $(-I_v, I_u)$, and multiplying the result by the local gradient magnitude $|\nabla I|$, the following can be obtained:

$$\triangle_{Kitchen-Rosenfeld} = \frac{I_{uu}I_v^2 - 2I_{uv}I_uI_v + I_{vv}I_u^2}{I_u^2 + I_v^2} \tag{3.5}$$

The disadvantages of Kitchen-Rosenfeld corner detector are also obvious. It depends on the second-order derivatives, so it is very sensitive to noise. Also it is a little sensitive to strong edge points as the cornerness value is proportional to the gradient magnitude.

Zuiniga-Haralick Corner Detector

Zuniga and Haralick proposed three different geometry-based corner detectors in [97]. They chose to represent the gray level image surface $I(u, v)$ by a cubic polynomial in the coordinates of the neighborhood array as follows:

$$I(u,v) = k_1 + k_2u + k_3v + k_4u^2 + k_5uv + k_6v^2 + k_7u^3 + k_8u^2v + k_9uv^2 + k_{10}v^3 \tag{3.6}$$

They associated corners with two things: the occurrence of an edge point and the significant changes in edge direction. The three approaches are summarized as following:

- Incremental change along tangent line

Consider two points P_1 and P_2, which are very close to the point (u, v) under consideration, along a direction perpendicular to the gradient, which is the tangent line to the edge boundary. The point is declared as a corner point if all three points are edge points and the change in gradient direction between P_1 and P_2 is greater than a threshold.

- Incremental change along contour line

  Consider two points P_1 and P_2, which are very close to the point (u, v) under consideration, lying on a contour line instead of a tangent line. The point is declared as a corner point if all three points are edge points and the change in gradient direction between P_1 and P_2 is greater than a threshold.

- Instantaneous rate of change

  The directional derivative of the gradient direction, along the edge direction is computed. If it is greater than some threshold, and if the point under consideration is an edge point, the point is declared to be a corner point. The cornerness measure can be expressed as:

$$\triangle_{Zuiniga-Haralick} = -2\frac{k_2^2 k_6 - k_2 k_3 k_5 + k_3^2 k_4}{(k_2^2 + k_3^2)^{3/2}} \tag{3.7}$$

Wang-Brady Corner Detector

Wang-Brady corner detector is based on the observation that the total curvature of the gray level image is proportional to the second order directional derivative in the direction tangential to edge normal, and inversely proportional to the edge strength (norm of the edge normal) [85, 86]. Let $n = \nabla I/|\nabla I|$ be the edge normal direction, t be the edge tangential direction (the unit tangent vector perpendicular to n) and k be the total curvature of the image surface. Differentiating $I(u, v)$ in the direction

of n and t, we have:

$$\frac{\partial^2 I}{\partial n^2} = \frac{I_u^2 I_{uu} + 2I_u I_v I_{uv} + I_v^2 I_{vv}}{|\nabla I|^2} \tag{3.8}$$

$$\frac{\partial^2 I}{\partial t^2} = \frac{I_v^2 I_{uu} - 2I_u I_v I_{uv} + I_u^2 I_{vv}}{|\nabla I|^2} \tag{3.9}$$

The toal curvature k can be computed as:

$$k = k_n + k_t = \frac{(1 + I_u^2) I_{vv} + (1 + I_v^2) I_{uu} - 2I_u I_v I_{uv}}{(1 + I_u^2 + I_v^2)^{3/2}} \tag{3.10}$$

From $I_{uu} + I_{vv} = I_{nn} + I_{tt}$ and $|\nabla I|^2 \gg 1$, the Wang-Brady cornerness measure can be approximated from the above equation as:

$$\triangle_{Wang-Brady} = k = \frac{\partial^2 I}{\partial n^2} / |\nabla I| \tag{3.11}$$

To suppress strong edge points and noise, they introduced a new technique known as false corner response suppression which modifies the cornerness measure as:

$$\triangle_{Wang-Brady} = (\frac{\partial^2 I}{\partial t^2})^2 - S_{Wang} |\nabla I|^2 \tag{3.12}$$

where S_{Wang} is a constant measure of image surface curvature varying with different differentiation masks. Wang-Brady corner detector does not need smoothing due to this technology. However, if the image is very noisy, some smoothing ($\sigma = 0.6$) will do help. Also, in their implementation, a linear interpolation scheme is incorporated to improve the performance of corner localization.

Compared to Kitchen-Rosenfeld approach, Wang-Brady corner detector works better in both detection and localization. What's more, Wang-Brady operator is of low algorithm complexity, meeting the requirement for real time applications.

Deriche-Giraudon Corner Detector

Deriche and Giraudon solved the localization problem of Beaudet corner detector [17]. Their approach is a multi-scale analysis of Gaussian space. Two important corner behaviors in the scale space are combined to develop a new corner detector: 1) the exact position of a corner can be detected as a stable zero-crossing in the scale space; 2) the local maximum in Beaudet cornerness measure moves in the scale space along the bisector line that passes through the exact position of the corner point. The implementation consists of the following four steps:

- Compute two images $DET1$ and $DET2$ corresponding to two different scale values $\sigma_1 < \sigma_2$, where $DET = I_{uu}I_{vv} - I_{uv}^2$ is Beaudet cornerness measure.
- Detect and threshold all the local maximum (only positive value) of $DET1$ and $DET2$.
- For each position (u_2, v_2) of a local maximum in $DET2$, search for a local maximum in $DET1$. The searching process is implemented in a local window centered on u_2, v_2. The first local maximum u_1, v_1 will be taken.
- Computer the line equation joining (u_1, v_1) and (u_2, v_2). In the corner case, this gives the estimated bisector line where the exact corner position must lie. Along this line, starting from (u_2, v_2) and moving in the direction toward (u_1, v_1). Report the first zero-crossing position (u, v) of the Laplacian as a corner point.

The most significant feature of Deriche-Giraudon corner detector is its localization performance. In order to improve the precision in localization further, they fit a quadratic surface around each local maximum (u_i, v_i) and refine the position of each local maximum at sub-pixel precision. They proved that even under very noisy situations, the corners can still be localized very well. However, this success is achieved at the cost of increasing computational complexity. This might be a problem for real time applications.

3.2.2 Auto-correlation Corner Detectors

This type of corner detectors measures the intensity correlation of the local window and the neighborhood window to estimate the intensity changes under small window shifts. If the shift under every direction produces large correlation, a corner point will be reported.

Moravec Corner Detector

Moravec corner detector functions by considering a local window in an image and determining the average changes of intensity which results from shifting the window by a small amount in four directions [52]. Three cases arise:

- If the windowed image patch is flat, all shifts will result in a negligible change.
- If the window straddles an edge, a shift along the edge will result in a negligible change, but a shift perpendicular to the edge will result in a large change.
- If the windowed patch is a corner, all shifts will result in a large change. Thus a corner can be declared when the minimum change produced by any of the shift is large.

Moravec cornerness measure is given by the correlation function below:

$$\Delta_{Moravec} = \sum_{x,y} [I(u+x, v+y) - I(u,v)]^2 \tag{3.13}$$

where the shift (x, y) comprises $\{(1,0), (0,1), (1,1), (1,-1)\}$. This approach is quite sensitive to noise and strong edges as only four directions are checked.

Plessey Corner Detector

The idea of Moravec corner detector was improved by Harris and Stephens [28]. They measured the auto-correlation using the first-order derivatives, which can expand

the Moravec cornerness measure as following:

$$\begin{aligned} \triangle_{Moravec} &= \textstyle\sum_{u,v} g(\sigma,u,v)[I(u+x,v+y)-I(u,v))]^2 \\ &= \textstyle\sum_{u,v} g(\sigma,u,v)[uI_u+vI_v+O(u^2,v^2)]^2 \\ &\approx (u,v)M(u,v)^T \end{aligned} \tag{3.14}$$

where

$$\begin{aligned} M &= \begin{pmatrix} \widehat{I_u^2} & \widehat{I_uI_v} \\ \widehat{I_uI_v} & \widehat{I_v^2} \end{pmatrix} \\ \widehat{I_u^2} &= G(\sigma)\bigotimes I_u^2 \\ \widehat{I_v^2} &= G(\sigma)\bigotimes I_v^2 \\ \widehat{I_uI_v} &= G(\sigma)\bigotimes(I_uI_v) \end{aligned} \tag{3.15}$$

Let a and b be the eigenvalues of M. Then a and b will be proportional to the principal curvatures, and form a rotationally invariant description of M. Similar to Moravec corner detector, three cases arise:

- If both curvatures are small, the local auto-correlation function is flat. Any small shifts of the patch cause little change in the cornerness measure, which indicates a flat region.
- If one curvature is high and the other is low, the local auto-correlation is ridge-shaped. Only shift along the ridge will cause little change in the cornerness measure, which indicates an edge.
- If both curvatures are high, the local auto-correlation function is sharply peaked. Shift in any direction will significantly change the cornerness measure, which indicates a corner.

After analyzing the relationship between the matrix and the eigenvalues, they in-

troduced Plessey cornerness measure as following:

$$\begin{aligned} \triangle_{Plessey} &= det(M) - K_{Plessey} Tr^2(M) \\ &= (\widehat{I}_u^2 \widehat{I}_v^2 - \widehat{I_u I_v}^2) - K_{Plessey}(\widehat{I}_u^2 + \widehat{I}_v^2)^2 \end{aligned} \tag{3.16}$$

where K is usually set to 0.0401. This algorithm is consistent, but computationally expensive, mainly due to the Gaussian filter that is applied three times.

Zheng-Wang Corner Detector

Zheng and Wang improved the Plessey corner detector by presenting a new approach named gradient-direction corner detector [90]. Instead of calculating the three Gaussian smoothed gradient-multiple images, only two second order gradient-multiple images are required. The cornerness measure of the new corner detector can be described as following:

$$\triangle_{Zheng-Wang} = I_u^2 I_{vv}^2 + I_v^2 I_{uu}^2 - K(I_u^2 + I_v^2)^2 \tag{3.17}$$

K is a parameter function $K(u, v)$ used for false corner response suppression. It is defined as the convolution of the Gaussian operator $G(\sigma, u, v)$ with the original cornerness measure. This approach is comparable with the Plessey corner detector with respect to the detection performance and has better localization performance, particularly in the case of large Gaussian convolution. Furthermore, it requires fewer Gaussian convolution operations than Plessey, hence the algorithm complexity can be reduced greatly.

Ando Corner Detector

Recently, Ando attempted to analyze 1D and 2D features, such as edges, ridges and corners in image using gradient covariance matrix [1]. They also considered the matrix (3.15) and defined the OMNIVAR detector Q_{EG} and the UNIVAR detector

P_{EG} as follows:

$$Q_{EG} = \frac{4(\widehat{I_u^2}\widehat{I_v^2} - \widehat{I_u I_v}^2)}{(\widehat{I_u^2} + \widehat{I_v^2})^2} \tag{3.18}$$

$$P_{EG} = \frac{(\widehat{I_u^2} - \widehat{I_v^2})^2 + 4\widehat{I_u I_v}^2}{(\widehat{I_u^2} + \widehat{I_v^2})^2} \tag{3.19}$$

Q_{EG} and P_{EG} have the following properties:

- P_{EG} reaches 1, where grayness varies one-dimensionally.
- Q_{EG} reaches 1, where the grayness variation is omnidirectional in such a sense that:

$$\widehat{I_u^2} = \widehat{I_v^2}, \widehat{I_u I_v} = 0 \tag{3.20}$$

Some observations can be further derived:

- P_{EG} reaches 1 in slants of grayness.
- P_{EG} reaches 1 near edges or ridges.
- Q_{EG} reaches 1 at a center of circular symmetry.
- Q_{EG} reaches 1 at a center of rotational periodicity with a period $\pi/2$.
- Q_{EG} reaches 1 at a center of rotational skew periodicity with a period $\pi/2$.

From the last two observations, they expected that Q_{EG} will respond at orthogonal intersection of lines or edges. This means that OMNIVAR frequently appears in unidirectionally varying regions in polar coordinates such as peaks, corners and vertices. They also showed the intrinsic relationship between their OMNIVAR detector and Plessey corner detector. Ando algorithm has good corner localization and noise robustness, but the false alarm rate is high. This is due to that, besides corners, Q_{EG} also responds to the centers of a dot or blob in an image.

3.2.3 Structure-based Corner Detector

This type of corner detectors analyzes the structure of the local window. In SUSAN corner detector, the structure means what the USAN area looks like. Flat points, edges and corners are shown to have different USAN structures around them. This observation can be used to distinguish these three kinds of points effectively. Structure-based corner detectors are much different from the previous two types. The most obvious difference is that no image derivatives are used, thus these approaches are less sensitive to the image noise.

SUSAN Corner Detector

The first structure-based corner detector was proposed by Smith and Brady, namely SUSAN corner detector [74]. Considering an arbitrary pixel in the image and a corresponding circular window around it (the central pixel shall be called the nucleus). Provided the image is not textured, there is a compact region within the window whose pixels have similar brightness to the nucleus and this area will be called USAN, standing for "Univalue Segment Assimilating Nucleus". They showed that the USAN area is at a maximum when the nucleus lies in a flat region of the image surface, it falls to half of this maximum very near a straight edge, and falls even further when inside a corner. This analysis leads directly to formulation of the SUSAN principle: the inverted USAN area has edges and corners strongly enhanced, with corners more strongly enhanced than edges. Based on the SUSAN principle, they gave SUSAN cornerness measure as following:

$$\triangle_{SUSAN} = \begin{cases} g - n(u,v) & \text{if} \quad n(u,v) < g \\ 0 & \text{if} \quad n(u,v) \geq g \end{cases} \tag{3.21}$$

where $n(u,v)$ denotes the size of USAN area and g is the geometric threshold which is always set to be half of the mask size. The corners are marked by finding the

local maximum of $\triangle_{SUSAN}$. SUSAN has very good speed performance, but it has been reported [79] that it has poor stability in real images and this problem will be analyzed in detail in next chapter.

Trajkovic-Hedley Corner Detector

Trajkovic and Hedley also considered the USAN model [79]. Consider an arbitrary line l that passes through the nucleus and intersects the boundary of the circular window at two opposite points P and P'. Their cornessness measure at any pixel is given by:

$$\triangle_{Trajkovic-Hedley} = \min_{P,P'}[(I_P - I_N)^2 + (I_{P'} - I_N)^2] \tag{3.22}$$

where N is the central point and I_P is the image intensity at the point P.

Three cases can occur:

- If the nucleus is a flat point, there is at least one line l, so that both P and P' belong to the USAN. The cornerness measure value will be low.
- If the nucleus is an edge, there is exactly one line tangential to the edge, so that both P and P' belong to the USAN and the cornerness measure value is low.
- If the nucleus is a corner, for every line l at least one of the points P and P' does not belong to the USAN, hence the cornerness measure will be high.

In their implementation, they used a linear or circular interpixel approximation to overcome the problem of lines at certain orientations being detected as corners.

COP Corner Detector

The same concept of USAN was also considered in [75] recently and a new corner detector, COP, was proposed to detect the corners more easily and effectively. The COP corner detector is composed of two oriented crosses, one of which responds

strongly to $\pm 45^o$ edges and the other one to the horizontal and vertical edges. According to the COP response, they pre-defined 15 raw directions, d_i (i=1,...,15). To provide smoothing effect in real scene process, they defined each probability p_i from the raw direction distribution as:

$$p_i = w_{ik} n_k \tag{3.23}$$

where w_{ik} is the weight factor and n_k the frequency, that is, the number of occurrences of d_k in the local mask window. Probabilistic dominant directions, pd_i, are then defined as:

$$pd_i = \max_{i=1,2,4,5} p_i \tag{3.24}$$

Since corner is a point where orientation changes abruptly, it needs at least two dominant orientations. From the distribution of pd_i in a given circular window, they defined T_{ij} and G_i as:

$$T_{ij} = \int \int_{\Omega_i^j} p_i d\Omega_i^j \quad (i = 1, 2, 4, 5, j = 1, 2) \tag{3.25}$$

where T_{ij} represents the probability of pd_i in a sector, Ω_i^j, its area size is about 6 pixels. The rate of change of T_{ij} in i-direction, G_i, is given by:

$$G_i = abs(T_{i1} - T_{i2}) \quad i = 1, 2, 4, 5 \tag{3.26}$$

from which the corner dominant direction is determined by:

$$dom_i = \max(G_k) \quad k = 1, 2, 4, 5 \tag{3.27}$$

The corner response function Δ_{COP} is defined as:

$$\Delta_{COP} = G_i \times G_j (i \neq j) \tag{3.28}$$

The non-maximum suppression is needed because more than one point will have high Δ_{COP} around a corner. COP can handle more types of two-dimensional features such as Y-Junctions, T-Junctions and X-Junctions because they also have at least two dominant directions.

3.2.4 Remarks

Among all the above algorithms, Schmid [69] concluded that the Plessey corner detector gives the best detection performance. But it has also been reported [17] that the localization performance of Plessey is not good, general corners are localized several pixels away from their true locations. This will seriously affect the camera calibration accuracy. Also the computational complexity of Plessey is high which makes it not suitable for real time applications. On the contrary, SUSAN corner detector is well known for its computational efficiency and good localization performance. Unfortunately, it has been reported [79] that its detection performance is unstable in real applications. All the above shows that it is necessary to improve the current corner detectors to achieve a better tradeoff among detection, localization and speed. Two novel corner detectors will be described in the next two chapters.

Most corner detectors report the corners by providing their image coordinates. However, corners are in nature combined with more attributes than just their positions. A richer corner description might provide additional constraints for computer vision systems to converge to a less ambiguous solution much faster. For example, corner matching algorithm can use the additional information to resolve ambiguous correspondences and to eliminate corners whose attributes do not match certain criteria. This problem is usually solved by two-stage approaches. A specific corner detector is used to extract the corner positions first. Then some independent modules are used to augment other corner attributes. Rohr [63] [64] introduced a general analytical parametric model for the corner structure and used an iteration computation

to compute the corner parameters until the the computed corner model fits the real corner structure. Their algorithm is computationally expensive, as the model fitting process requires iterative solutions to nonlinear systems. Rosin [65] calculated the corner attributes using two approaches: 1) intensity and gradient orientation histograms; 2) local threshold. The first approach uses a histogram of either the intensities or the gradient orientations of the pixels in the window centered on the corner. For L-Junctions, the orientations histogram should show two main peaks. To distinguish the two peaks of noise corrupted histogram, they smooth the histogram over a range of scales, and the scale maximizing a heuristic measure of "2-peakness" is chosen. The second approach locally thresholds the image in a circular window centered on the corner to distinguish the foreground object that the corner lies on from the background. They chose Tsai's moment preserving threshold method in their paper. There are also attempts which try to incorporate attributes estimation within the detection stage. Mehrotra *et al.* searched for corners based on half-edges detection from which the corner orientation and angles can be computed [48]. Chabat *et al.* also reported a corner detector which can report the corner orientation from histogram search [11].

3.3 Corner Matching

When the epipolar geometry between two views is unknown, the matching problem is very hard to solve because no constraints are available. But under some assumptions, it is possible to automatically match points between images. One of the most useful assumptions is that the images are not too different (i.e. same illumination, similar pose). In this case the coordinates of the corners and the intensity distributions around them are similar in both images. This allows to restrict the search range and to match corners through intensity cross-correlation.

Given a corner m_1 with coordinates $(u_1, v_1)^T$ in the first image, we consider a rectangular search area of size $(2d_u + 1) \times (2d_v + 1)$ around the same position in the second image, which is assumed to contain the potential match for m_1. This reflects some *a priori* knowledge about the disparity of the two images. For each corner m_2 with coordinates $(u_2, v_2)^T$ within the search area, a correlation operation will be performed between two correlation windows (the size of the correlation window is assumed to be $(2n + 1) \times (2m + 1)$) centered at m_2 and m_1 respectively. The correlation score of m_1 and m_2 is given by:

$$Score = \frac{\sum_i \sum_j [I_1(u_1 + i, v_1 + j) - \overline{I_1(u_1, v_1)}] \times [I_2(u_2 + i, v_2 + j) - \overline{I_2(u_2, v_2)}]}{(2n + 1)(2m + 1)\sqrt{\sigma^2(I_1) \times \sigma^2(I_2)}} \tag{3.29}$$

where $\overline{I_k(u, v)}$ and $\sigma(I_k)$ are the average intensity and the standard deviation at point (u, v) over the correlation window in the k-th image $(k = 1, 2)$. The score ranges from -1, for two totally different correlation windows, to 1, for two identical correlation windows. For two corners to be considered as a match, their correlation score must be higher than a given threshold. It is possible for a corner in the first image to be paired to several corners in the second image and vice verse. To resolve the match ambiguities, relaxation process can be applied, which reorganize the candidate matches by propagating some constraints, such as continuity and uniqueness. The detail can be found in [94] and it gives one-to-one corner match for the two images.

3.4 Recovery of Epipolar Geometry

After the matches between two images are available, they can be used to compute the fundamental matrix F. Any pair of image points $m_1 = [u_1, v_1, 1]^T$ and $m_2 =$

$[u_2, v_2, 1]^T$ will give a constraint on the unknowns of F as:

$$m_2^T F m_1 = 0 \Rightarrow a^T f = 0 \tag{3.30}$$

where

$$\begin{aligned} a &= [u_1u_2, v_1u_2, u_2, u_1v_2, v_1v_2, v_2, u_1, v_1, 1]^T \\ f &= [F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}]^T \end{aligned} \tag{3.31}$$

where F_{ij} is the element of F at the i-th row and j-th column. If n pairs of corner matches are available, we can have the following linear system to solve:

$$Af = 0 \tag{3.32}$$

where $A = [a_1, ..., a_n]^T$ is a $n \times 9$ matrix.

3.4.1 Seven-Point Solution

A fundamental matrix F has only 7 degrees of freedom. Thus 7 pairs of corner matches allow us to have a solution for the epipolar geometry [40] [78]. In this case, $n = 7$ and $rank(A) = 7$. From equation (3.32), F is given by the right null-space of A. Through the singular value decomposition, we can obtain vectors f_1 and f_2 which span the null space of A. The null space is a linear combination of f_1 and f_2, which correspond to matrices F_1 and F_2 respectively. Since the determinant of F must be zero, i.e.:

$$\det[\alpha F_1 + (1 - \alpha)F_2] = 0 \tag{3.33}$$

we can obtain a polynomial equation in α. The maximum number of real solutions is 3. For each solution α, the fundamental matrix is given by:

$$F = \alpha F_1 + (1 - \alpha)F_2 \tag{3.34}$$

3.4.2 Eight-Point Solution

When 8 pairs of corner matches are given, the matrix A in equation (3.32) is of rank 8. To avoid the trivial solution $f = 0$, constraints must be imposed on f. Normally, we will enforce $|f|^2 = 1$ as zero vector cannot have a unit norm. Thus equation (3.32) changes to a minimization problem:

$$\min |Af|^2 \quad \text{subject to} \quad |f|^2 = 1 \tag{3.35}$$

The solution of the above equation is right null-space of A and can be computed using singular value decomposition. It is trivial to reconstruct the fundamental matrix F from f. However, in the presence of noise, the computed F will not satisfy the rank-2 constraint. This means that there will not be real epipoles through which all epipolar lines pass. To solve this problem, the closest rank-2 approximation to F can be chosen as the true fundamental matrix. It has been pointed out [32] that in practice it is very important to normalize the coordinates of the image points to the range of $[-1, 1]$, so that all the elements of A are of the same order of magnitude.

3.4.3 The Solution of More Points

The eight-point solution can be easily extended to the situation when more corner matches are available. A problem with the equation (3.35) is that the quantity we are minimizing is not physically meaningful. It is always better to minimize a geometrically meaningful criterion. One such quantity is the distance from a point m_2 to its corresponding epipolar line $l_2 \sim Fm_1$, which will be represented by $d(m_2, Fm_1)$. The sum of such distances for all the corner matches should be minimized. As the two images should play a symmetric role, the distance from m_1 to the epipolar line $l_1 \sim F^T m_2$ should also be considered. Thus the following

nonlinear criterion should be satisfied:

$$\min \sum_i (d^2(m_{1i}, Fm_{2i}) + d^2(m_{2i}, F^T m_{1i})) \tag{3.36}$$

The above nonlinear minimization problem can be solved by Levenbergh-Marquardt minimization routine [60] with respect to the unknowns of F. The result obtained by linear algorithms can be used for initialization.

3.4.4 Robust Methods

The most important disadvantage of the previous approaches is that they can not cope with outliers. If the set of matches is contaminated with even a small set of outliers, the result will probably be unusable. This is typical for all types of least-squares approaches (even nonlinear ones). This problem can be solved by some robust methods, such as RANSAC (RANdom SAmpling Consensus) [23], M-Estimators [55] and LMedS (Least Median of Squares) [92]. The method of LMedS will be briefly described here. Given n sets of corner correspondences m_{1i} and m_{2i} $(i = 1, ..., n)$. A Monte Carlo type technique is used to draw m random samples of 8 different point correspondences. For each sample, indexed by J, a fundamental matrix F_J can be computed. For each F_J, the median of the squared residuals, denoted by M_J, with respect to the whole set of corner correspondences can be computed as:

$$M_J = \text{med}[d^2(m_{2i}, F_J m_{1i}) + d^2(m_{1i}, F_J^T m_{2i})] \tag{3.37}$$

The F_J whose corresponding M_J is smallest will be chosen as the correct solution.

3.5 Corner Matching Guided by Epipolar Geometry

Once the fundamental matrix has been determined robustly, it can be used to establish a new set of correspondences using the approach described in section 3.3. The situation is a little different here, one more constraint, i.e. the epipolar constraint, must be satisfied for a pair of corner correspondences. For a corner in the first image, the corresponding epipolar line in the second image can be computed and only the corners within a narrow band (1 or 2 pixels width) centered at the epipolar line are considered for matching. As a result, the search range will be highly reduced. The same constraints as in section 3.3 will be applied to select the most consistent matches, except that the constraint on the disparity is replaced by the epipolar constraint. If needed, the fundamental matrix can also be refined using all the new corner correspondences.

3.6 Conclusion

This chapter discussed the problem of relating two images of the same scene. First, various corner detection algorithms were reviewed and it came to the conclusion that it is necessary to improve the current corner detectors to achieve a better tradeoff among detection, localization and speed. Then it was shown how to match the detected corners across two images automatically. It was achieved by assuming that these two images are not too different. Lastly, the algorithms used to compute the fundamental matrix from the corner matches were reviewed.

Chapter 4

Improve The Stability of SUSAN Corner Detector

4.1 Introduction

Among all the proposed corner detection algorithms, SUSAN corner detector [74] is well-known for its computational efficiency, noisy tolerance and reasonable accuracy. But it has been reported that SUSAN has poor stability on the real images [79]. This chapter analyzes the stability performance of SUSAN and concludes that its instability is due to the violation of USAN constraint, which is the principle of SUSAN, by the edges of real images. To solve this problem, the USAN constraint is extended to the general local region constraint. By replacing USAN with a new local region definition namely BDRAN, the corresponding local region constraint holds very well with the real images and this results in a stable corner detection algorithm. Experiments on real images will be presented to show the novelty of the newly developed approach.

This chapter is organized as follows. The SUSAN principle is summarized in section

4.2. The extension of USAN constraint to the general local region constraint is given in section 4.3. Section 4.4 gives a computational analysis to the stability performance of SUSAN and shows the main reason that makes SUSAN unstable. Section 4.5 compares two approaches for mask segmentation and concludes that the criterion of thresholding can give more consistent results than the criterion of brightness similarity. Section 4.6 introduces a new local region definition namely BDRAN, based on which a stable corner detection algorithm is proposed. Experiments are given in section 4.7 to compare the performance of the new algorithm and SUSAN. Section 4.8 concludes this chapter.

Notations:

x_0	the nucleus (central pixel) of a mask
x	an arbitrary pixel within the mask
$I(x)$	the gray level intensity of a pixel
n_{MAX}	the mask area
$USAN$	Univalue Segment Assimulating Nucleus
n_{USAN}	the USAN area
c_{USAN}	the USAN response
R_B	the bright region of the mask
R_D	the dark region of the mask
x_B	an arbitrary pixel of the bright region
x_D	an arbitrary pixel of the dark region
$I_{MIN}(R_B)$	the minimal brightness of the pixels of the bright region
$I_{MAX}(R_D)$	the maximal brightness of the pixels of the dark region
$BRAN$	Bright Region Assimulating Nucleus
$DRAN$	Dark Region Assimulating Nucleus
$BDRAN$	Bright or Dark Region Assimulating Nucleus
n_{BRAN}	the BRAN area
c_{BRAN}	the BRAN response
n_{DRAN}	the DRAN area
c_{DRAN}	the DRAN response
n_{BDRAN}	the BDRAN area
c_{BDRAN}	the BDRAN response

4.2 Principle of SUSAN Corner Detector

The SUSAN principle will be summarized first [74]. Consider a circular mask centered at an arbitrary image pixel, which shall be known as the *nucleus*. If the brightness of each pixel within the mask is compared with the brightness of the nucleus, then an area of the mask can be defined which has the same (or similar) brightness as the nucleus. This area of the mask shall be known as the *USAN* of the nucleus, an acronym standing for Univalue Segment Assimilating Nucleus. As shown in fig.4.1, the USAN area is at a maximum of the mask area when the nucleus lies on a flat region of the image surface, it falls to half of this maximum when very near a straight edge and falls even further when inside a corner. Thus the inverted USAN area will have edges strongly enhanced and have corners more strongly enhanced.

More specifically, a simple equation can be used to determine the brightness similarity of two pixels:

$$s(x, x_0) = \begin{cases} 1 & \text{if} \quad |I(x) - I(x_0)| \leq t \\ 0 & \text{if} \quad |I(x) - I(x_0)| > t \end{cases} \tag{4.1}$$

where x_0 is the nucleus and x is any pixel within the mask, $I(\cdot)$ is the brightness of any pixel, t is the brightness threshold whose typical values are within the range from 10 to 30, and $s(x, x_0)$ is the degree of brightness similarity of x and x_0. The comparison is done for each pixel within the mask, and a running total n_{USAN} of the outputs s is made:

$$n_{USAN}(x_0) = \sum_x s(x, x_0) \tag{4.2}$$

This total n_{USAN} is just the number of pixels in the USAN, i.e. it gives the USAN area. The *USAN response* c_{USAN} of the nucleus is defined as the inverted USAN

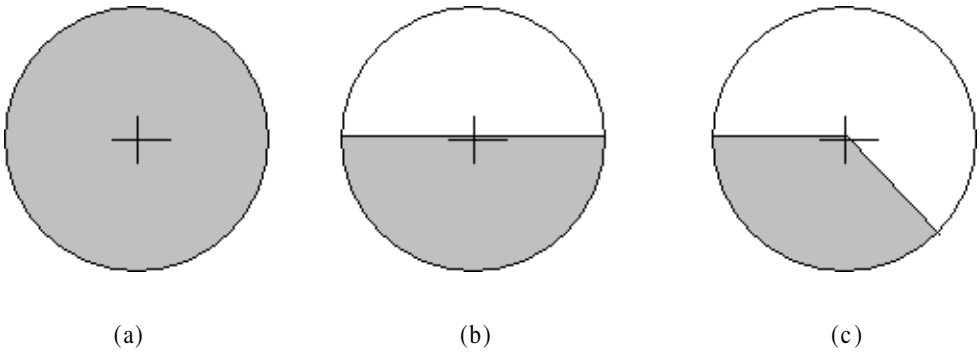


Figure 4.1: Three representative shapes of USAN (the position of '+' is the nucleus and the gray area is the respective USAN): (a)the nucleus lies on a flat area; (b)the nucleus is an edge; (c)the nucleus is a corner.

area:

$$c_{USAN}(x_0) = n_{MAX} - n_{USAN}(x_0) \tag{4.3}$$

where n_{MAX} is the maximal value that n can take and equals to the mask area. As c_{USAN} strongly enhances the corner features, a geometry threshold g_c can be set and a corner will be reported if its USAN response is bigger than g_c. As the USAN always includes the nucleus, the USAN area of a digital edge is always bigger than half of n_{MAX} and its USAN response is always smaller than half of n_{MAX}. Thus it is safe to set g_c to be exactly $\frac{n_{MAX}}{2}$.

The principle of SUSAN corner detector can be summarized into the *USAN constraint* as follows:

USAN Constraint:

As shown in fig.4.2, *the USAN response (inverted USAN area) of a flat point or an edge is always smaller than* $g_c = \frac{n_{MAX}}{2}$, *while the USAN response of a corner is always bigger than* $g_c = \frac{n_{MAX}}{2}$.

Note that the above constraint is for digital images. For the continuous signals, the corresponding constraint is almost the same except that the USAN responses of continuous edges might equal to g_c.

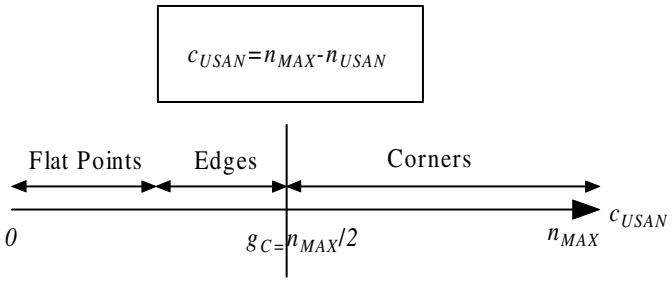


Figure 4.2: The ideal USAN response distribution of feature points. Note that $g_c = \frac{n_{MAX}}{2}$ can ideally separate the corners from other feature points.

4.3 Extension of USAN Constraint to General Local Region Constraint

If we treat the mask centered at a nucleus as a small image, it can be segmented into several regions based on the criterion of brightness similarity. Suppose R_N is the one that contains the nucleus. All the pixels of R_N will have similar brightness with the nucleus, thus R_N is exactly the USAN of the nucleus. This shows that USAN is in fact determined by the mask segmentation (region segmentation for the mask) result based on brightness similarity. As many approaches exist for mask segmentation [10], we can extend USAN to a general local region concept as follows:

Local Region Definition:

Consider a circular mask centered at a pixel and suppose it can be segmented into a set of regions based on a certain criterion, the local region of the central pixel is defined as the set of pixels within the mask that belong to the same region with the nucleus.

It is easy to check that USAN is a special case of the local region whose corresponding mask segmentation criterion is the brightness similarity. Similarly, if we use *local region area* to represent the number of pixels in it and *local region response* to represent the inverted local region area, the USAN constraint can be extended to a

general local region constraint:

Local Region Constraint:

The local region response of a flat point or an edge is always smaller than $g_c = \frac{n_{MAX}}{2}$, *while the local region response of a corner is always bigger than* $g_c = \frac{n_{MAX}}{2}$.

In the above constraint, n_{MAX} remains the same meaning with the USAN case, which equals to the mask area. Local region constraint will always hold as long as the local region of each pixel can be correctly found. The USAN constraint is a spacial case of the local region constraint, whose corresponding local region definition is given by USAN.

4.4 Stability Analysis of SUSAN Corner Detector

SUSAN corner detector performs very well on ideal images, whose edges and corners are associated with sharp discontinuities. But because of low-frequency components or the smoothing introduced by most sensing devices, sharp discontinuities rarely exist in real images. In the experiments given by [79], we found that SUSAN cannot work well on such images. This part will focus on the stability analysis of SUSAN. The real signal is modelled by convolving the ideal signal with the Gaussian smoothing function.

4.4.1 Analysis of USAN Constraint

A computational analysis with 1D edge signals will be given in this part. We will try to show how the real edge signals violate the USAN constraint and how this leads to the instability of SUSAN corner detector. So the analysis will focus on edge signals (1D signals) instead of corners (2D signals) although the corner detection is

essentially a 2D problem. For each 1D signal, we assume the exact edge is located at $x_E = 0$. The ideal step edge signal can be modelled in 1D as:

$$I(x) = \begin{cases} I_1 & \text{if} \quad x < 0 \\ I_2 & \text{if} \quad x \geq 0 \end{cases} \tag{4.4}$$

Without loss of generalization, we can assume that $I_1 < I_2$. However step edges are rare in real images. We model the 1D real edge signal by convolving the ideal step edge signal with the Gaussian smoothing function $G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$. σ is the standard deviation, which controls the smoothness degree. Thus the real edge signal can be expressed as:

$$\hat{I}(x) = I_1 + \nabla I * \int_{-\infty}^{x} G(s)ds \tag{4.5}$$

in which $\nabla I = I_2 - I_1$ is the brightness contrast of $I(x)$. For the USAN constraint to hold for both signals, the USAN responses at the exact edge position $x_E = 0$ should always be no bigger than g_c.

For 1D signals, the circular mask centered at point x_E is the line segment $[-r, r]$ where r is the mask radius. Obviously, the mask area $n_{MAX} = 2r$ and the geometry threshold $g_c = r$. Assume the brightness threshold equals to t. The USAN responses of x_E for the ideal and real edge signals are respectively given by (refer to appendix 1 of this chapter):

$$\begin{aligned} c_{USAN}(x_E) &= r \\ c_{USAN}(\widehat{x}_E) &= 2r - 2x_R \end{aligned} \tag{4.6}$$

where x_R satisfies:

$$\int_0^{x_R} G(s)ds = \frac{t}{\nabla I} \tag{4.7}$$

Obviously, USAN constraint holds very well for the ideal step edge as $c_{USAN}(x_E) = r = g_c$. For USAN constraint to hold also for the real edge, the following needs to

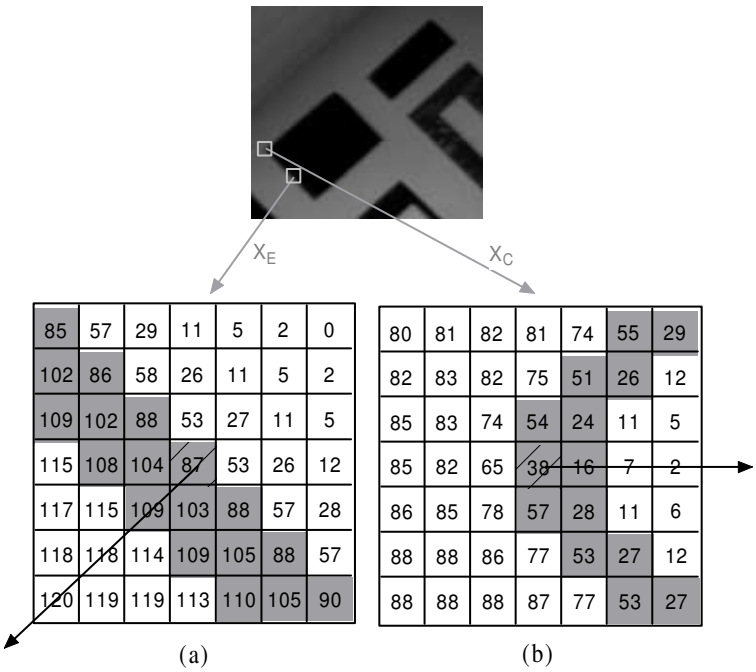


Figure 4.3: A visual example to explain the violation of USAN constraint. X_E and X_C are an edge pixel and a corner pixel of a small lab image respectively. The USAN computation is based on $t = 25$. (a) computed USAN area at X_E (gray pixels); (b) computed USAN area at X_C (gray pixels).

be satisfied:

$$c_{USAN}(\widehat{x}_E) \leq r \Longrightarrow x_R \geq \frac{r}{2} \tag{4.8}$$

or equivalently:

$$\frac{t}{\nabla I} \geq \int_0^{\frac{r}{2}} G(s)ds \tag{4.9}$$

Consider a typical situation where $r = 3$ (this equals to the digital 7×7 mask) and $\sigma = 1.0$. For a real edge to satisfy the USAN constraint, its brightness contrast ∇I should satisfy (note that when $\sigma = 1.0$, $\int_0^{1.5} G(s)ds = 0.433$):

$$\nabla I \leq \frac{t}{\int_0^{\frac{r}{2}} G(s)ds} = \frac{t}{0.433} \tag{4.10}$$

The above equation shows that for a t, all the edges whose brightness contrasts are bigger than $\frac{t}{0.4332}$ will violate the USAN constraint. As t is the threshold to

determine the brightness similarity of two pixels, obviously t cannot be set too big in most situations, whose typical values are within the range from 10 to 30. This means that, when applied to the real images, the USAN constraint will be violated by many edges if a reasonable brightness threshold is used.

To explain the above analysis, we show a visual example here. Fig.4.3 shows one part of a real lab image. We choose an edge pixel X_E and a corner pixel X_C from the image, and try to analyze their USAN responses. As perfect circular mask does not exist for digital images, we use a 7×7 mask as an approximation (please refer to Appendix 3 for knowledge of digital circular masks). The mask corresponds to the radius of 3 pixels and it contains 49 pixels, which means $n_{MAX} = 49$ and $g_c = 24.5$. The masks centered at X_E and X_C are shown as fig.4.3.a and fig.4.3.b respectively. Using $t = 25$, we can compute their respective USAN areas and draw them as gray in fig.4.3. Their USAN responses can be computed as $c_{USAN}(X_E) = 49 - 18 = 31$ and $c_{USAN}(X_C) = 49 - 14 = 35$. It is clear that both $c_{USAN}(X_E)$ and $c_{USAN}(X_C)$ are bigger than g_c. In this case, X_E and X_C cannot be separated by using g_c alone. This verifies our above analysis.

In a summary, the USAN responses of some real edges are also bigger than $g_c = \frac{n_{MAX}}{2}$, which violates the USAN constraint (fig.4.4). The consequence of such a violation is that **the corners cannot be distinguished from the edges based on their USAN responses**.

4.4.2 Violation of USAN Constraint Makes SUSAN Unstable

The violation of USAN constraint will make SUSAN corner detector wrongly mark a lot of strong edges as corners. The authors of SUSAN have proposed two extra steps to reject these strong edges. The first addition is to find the center of gravity (CoG)

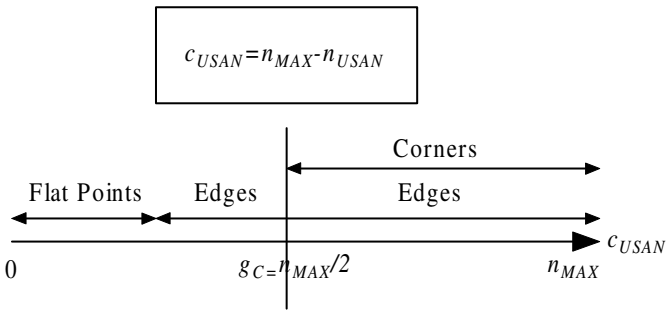


Figure 4.4: The actual USAN response distribution of feature points with consideration of smoothness and blur. Note that $g_c = \frac{n_{MAX}}{2}$ cannot separate the corners from other feature points as the USAN responses of real edges are also bigger than g_c.

of the USAN area. Then the distance from the nucleus to the CoG is computed. They argue that an USAN corresponding to a proper corner will have a CoG that is not near to the nucleus. If the distance is short, the corresponding corner candidate will be rejected. The second addition is to enforce contiguity in the USAN. All the pixels within a mask, lying in a straight line pointing outwards from the nucleus in the direction of the USAN CoG must be part of USAN, for a corner to be detected. This is in fact to force the USAN to have a degree of uniformity.

But we argue that the above two additions can solve the problem of SUSAN partly but far from completely. The main problem is that each addition has a potential to reject the true corners. For the first addition, as the USAN area of a true edge or a real corner is very small, its CoG will be very sensitive to noise. If noisy points make the USAN CoG of a real corner closer to the nucleus, the corner will be falsely rejected; if noisy points make the USAN CoG of a real edge further from the nucleus, the edge will be falsely accepted. The second addition always cannot hold at true corners. Take fig.4.3 as an example. The arrow lines draw the direction from the nucleus to the USAN CoG. Clearly, the second addition does not hold at both fig.4.3.a and fig.4.3.b. Although it can reject the strong edge X_E correctly, it also falsely rejects the true corner X_C.

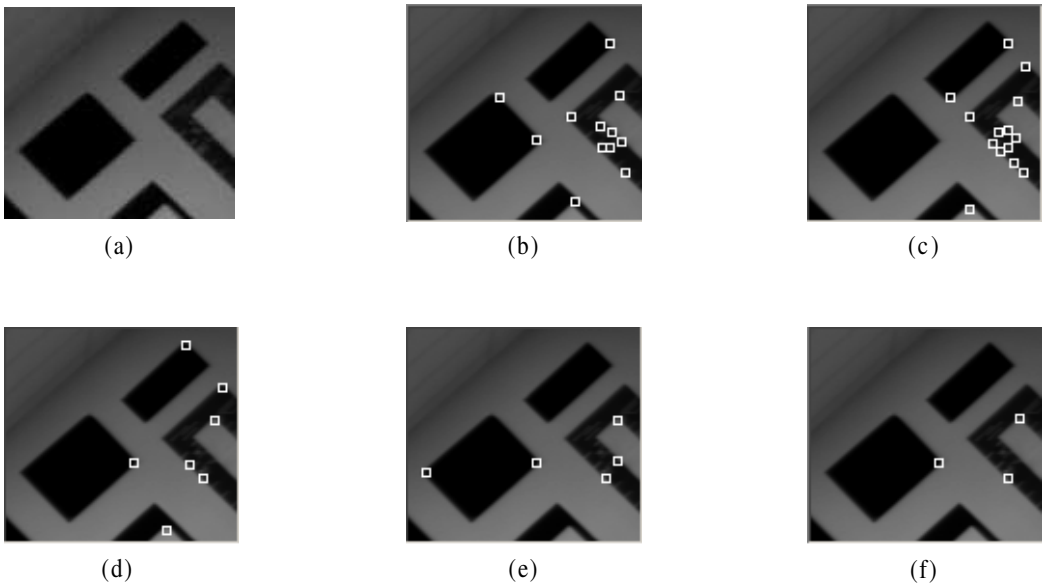


Figure 4.5: A real image example to show the instability of SUSAN. (a) one part of the lab image; (b) corner map of SUSAN with $t = 10$; (c) corner map of SUSAN with $t = 15$; (d) corner map of SUSAN with $t = 20$; (e) corner map of SUSAN with $t = 25$; (f) corner map of SUSAN with $t = 30$.

A complete corner detection result by SUSAN for the previous testing image is shown in fig.4.5. Five different thresholds ($t = 10, 15, 20, 25, 30$) are tried and the corner maps are shown from fig.4.5 (b) to (f). We can see from the figure that, no matter which threshold is chosen, either the number of missed corners or the number of falsely accepted corners is quite high. In fact, the results are quite unpredictable. When t goes from 10 to 30, some corners (i.e. the right corner of the left rectangle) are reported at first, then are missed, then are reported again. This example shows that SUSAN corner detector is unstable when applied to real images and verifies our above analysis.

4.5 Mask Segmentation

As stated, the USAN is determined by the mask segmentation result based on the criterion of brightness similarity. In this part, we will show that such a segmentation

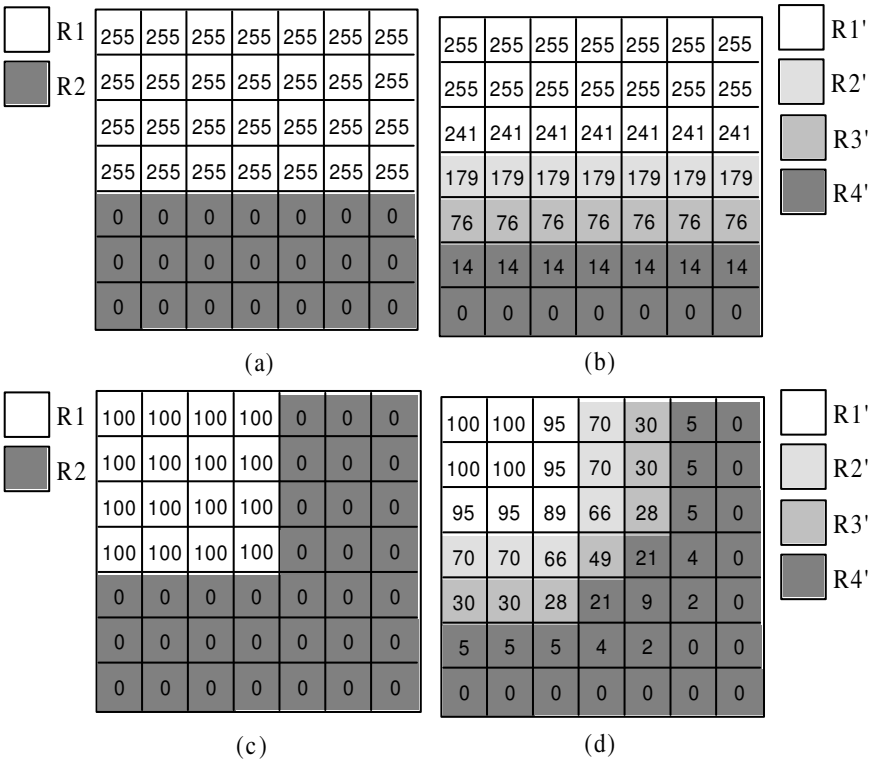


Figure 4.6: Mask segmentation results for four masks based on brightness similarity with $t = 25$. (a): the mask centered at an ideal edge; (b): the smoothed version of (a); (c): the mask centered at an ideal corner; (d): the smoothed version of (c).

criterion cannot produce consistent region results for ideal and real edge/corner masks, and this leads to the violation of USAN constraint. Also, we will show that by thresholding, the masks can be segmented consistently regardless of the smoothness or blur.

4.5.1 Mask Segmentation Using Brightness Similarity

Fig.4.6.a shows the segmentation result for an ideal edge mask with $t = 25$. Two regions are available here: $R1 = \{row1, row2, row3, row4\}$ and $R2 = \{row5, row6, row7\}$. But when we segment its smoothed version with the same parameter $t = 25$, four regions are available instead of original two regions (fig.4.6.b): $R1' = \{row1, row2, row3\}$, $R2' = \{row4\}$, $R3' = \{row5\}$ and $R4' = \{row6, row7\}$. The edges of original bright

region form a new region $R2'$ instead of belonging to $R1'$ as expected and the edges of original dark region form a new region $R3'$ instead of belonging to $R4'$ as expected. Such a segmentation is not consistent with the ground truth region information of the mask and this makes the USAN of the edge changes from $R1$ at the ideal case to $R2'$ at the smoothed case. As $R2'$ has a much smaller area than $R1$, it explains why the USAN response of the real edge is significantly enlarged. Similarly, fig.4.6 (c) and (d) show the segmentation results for an ideal corner mask and its smoothed version respectively with $t = 25$. Again, the segmentation results are not consistent: two regions are available for the ideal case and four regions are available for the smoothed case. This leads to the significant enlargement of the USAN response of the real corner.

4.5.2 Mask Segmentation Using Thresholding

As the local region is fully determined by the mask segmentation criterion, it is necessary to find a criterion to give consistent mask segmentation results regardless of the smoothness or blur.

Notice that for edge masks and most corner masks, their ground true region information contains two regions: the bright region and the dark region. We will use R_B to represent the bright region and x_B to represent any pixel of R_B. Similarly, R_D and x_D will be used to represent the dark region and any of its pixels. Also, we will use $I_{MIN}(R_B)$ to represent the smallest brightness of the pixels of R_B, that is:

$$I_{MIN}(R_B) = \min(I(x_B)) \quad \forall x_B \in R_B \tag{4.11}$$

We will use $I_{MAX}(R_D)$ to represent the biggest brightness of the pixels of R_D, that is:

$$I_{MAX}(R_D) = \max(I(x_D)) \quad \forall x_D \in R_D \tag{4.12}$$

As $I(x_B) > I(x_D)$ for any x_B and x_D, we can always have $I_{MIN}(R_B) > I_{MAX}(R_D)$. For any flat mask, although only one region exists, we can still assume two regions: the only region can be treated as R_B, and R_D can be assumed to be empty. In this case, $I_{MAX}(R_D) = -\infty$.

Obviously, if $I_{MIN}(R_B)$ is known, we can give an accurate segmentation to the mask by thresholding: for each pixel x within the mask,

$$x \in \begin{cases} R_B & \text{if} \quad I(x) \geq I_{MIN}(R_B) \\ R_D & \text{if} \quad I(x) < I_{MIN}(R_B) \end{cases} \tag{4.13}$$

Similarly, if $I_{MAX}(R_D)$ is known, we also can segment the mask successfully as follows: for each pixel x within the mask,

$$x \in \begin{cases} R_B & \text{if} \quad I(x) > I_{MAX}(R_D) \\ R_D & \text{if} \quad I(x) \leq I_{MAX}(R_D) \end{cases} \tag{4.14}$$

Both equations (4.13) and (4.14) give the perfect segmentation results.

Notice that for any mask containing edges or corners, $I_{MIN}(R_B)$ and $I_{MAX}(R_D)$ will always occur with these edges or corners. Thus $I_{MIN}(R_B)$ can be approximated as the brightness of the edges or corners of R_B, and $I_{MAX}(R_D)$ can be approximated as the brightness of the edges or corners of R_D. Thus for the masks centered at edges or corners, the brightness of the nucleus $I(x_0)$ either approximates $I_{MIN}(R_B)$ or $I_{MAX}(R_D)$ depending on whither $x_0 \in R_B$ or $x_0 \in R_D$.

Fig.4.7 shows the segmentation results for the previous mask examples by the approach of thresholding. For the ideal edge mask fig.4.7.a, as $x_0 \in R_B$ and $I(x_0) = 255$, we can have $I_{MIN}(R_B) = 255$. Thus the mask can be segmented by equation (4.13) and gives the following result: $R_1 = \{row1, row2, row3, row4\}$ and $R_2 = \{row5, row6, row7\}$. For its smoothed version (fig.4.7.b), again, as $x_0 \in R_B$

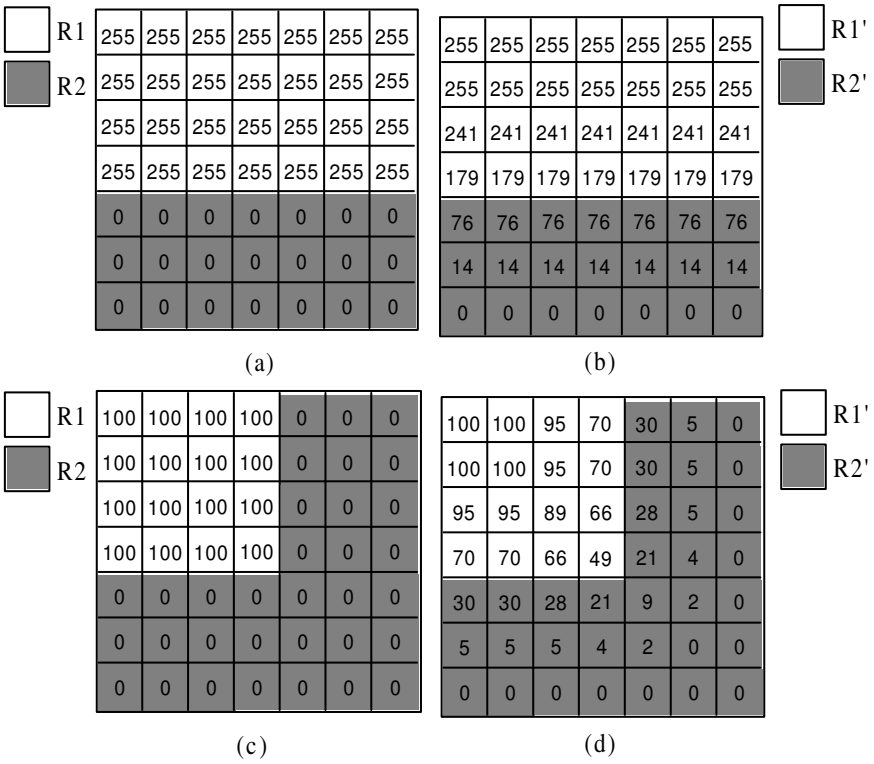


Figure 4.7: Mask segmentation results based on thresholding. (a): the mask centered at an ideal edge; (b): the smoothed version of (a); (c): the mask centered at an ideal corner; (d): the smoothed version of (c).

and $I(x_0) = 179$, we can have $I_{MIN}(R_B) = 179$. Equation (4.13) gives the segmentation result as: $R'_1 = \{row1, row2, row3, row4\}$ and $R'_2 = \{row5, row6, row7\}$. Obviously, the segmentation result is consistent with fig.4.7.a. For the ideal corner mask and its smoothed version, the segmentation results are shown in fig.4.7 (c) and (d) and it is easy to see that they are consistent also.

Note that we have assumed that we know whether $x_0 \in R_B$ or $x_0 \in R_D$. But actually this information is unknown to us and this problem will be solved in the next part.

4.6 BDRAN: A New Local Region Definition

As the criterion of thresholding can give consistent mask segmentation results, it corresponds to a local region definition which is more suitable for the corner detection task than USAN.

Two situations need to be considered. First, assume $x_0 \in R_B$. If the image is clean, all the pixels of the same region have the same brightness value, we can have $I_{MIN}(R_B) \approx I(x_0)$. If the image is smoothed, the brightness values of the boundary pixels of the brighter region will be averaged down by those pixels of the darker region. So the edges and corners of the brighter region will have smaller brightness values than those flat pixels, which means $I_{MIN}(R_B) \approx I(x_0)$ again. So the local region, which equals R_B, consists of the set of pixels within the mask which satisfy $I(x) \geq I(x_0)$. The local region given by the assumption of $x_0 \in R_B$ is named as BRAN, an acronym standing for Bright Region Assimulating Nucleus. Second, assume $x_0 \in R_D$. Similarly we can have $I_{MAX}(R_D) \approx I(x_0)$ and the local region, which equals R_D, consists of the set of pixels within the mask satisfy $I(x) \leq I(x_0)$. The local region given by the assumption of $x_0 \in R_D$ is named as DRAN, an acronym standing for Dark Region Assimulating Nucleus. As any pixel belongs to either R_B or R_D, its true local region is given by either BRAN or DRAN. To give a certain tolerance to the image noises, the above definitions for BRAN and DRAN need to be modified a little as follows: BRAN consists of the set of pixels which satisfy $I(x) \geq I(x_0) - t$ and DRAN consists of the set of pixels which satisfy $I(x) \leq I(x_0) + t$. t is our brightness threshold which has two functions: 1) it can clear the noise effects, 2) it can compensate the difference between $I(x_0)$ and $I_{MIN}(R_B)$ for BRAN or the difference between $I(x_0)$ and $I_{MAX}(R_D)$ for DRAN.

Use n_{BRAN} and n_{DRAN} to represent the BRAN area and DRAN area respectively. As for any pixel x within the mask, either $I(x) \geq I(x_0)$ or $I(x) \leq I(x_0)$ (x_0 satisfies

both equations), the union of BRAN and DRAN will contain all the pixels of the mask. Also notice that both BRAN and DRAN contain the nucleus, and we can have:

$$n_{BRAN} + n_{DRAN} > n_{MAX} \tag{4.15}$$

As we don't know whether $x_0 \in R_B$ or $x_0 \in R_D$, we need to choose one among BRAN and DRAN as the true local region. Three typical situations are analyzed as follows.

Consider the flat masks first. As all the pixels within a flat mask satisfy $I(x) = I(x_0)$, they all satisfy $I(x) \geq I(x_0)$ and $I(x) \leq I(x_0)$. This means both BRAN and DRAN equal the mask. In this case, either BRAN and DRAN can be chosen as the true local region.

For the edges masks, as the area of the true local region approximately equals $\frac{n_{MAX}}{2}$, the true local region is the one among BRAN and DRAN whose area is close to $\frac{n_{MAX}}{2}$. If $n_{BRAN} \approx \frac{n_{MAX}}{2}$, from equation (4.15), we can have $n_{DRAN} > n_{MAX} - n_{BRAN} > \frac{n_{MAX}}{2}$, thus $n_{DRAN} > n_{BRAN}$. Similarly if $n_{DRAN} \approx \frac{n_{MAX}}{2}$, we can have $n_{BRAN} > n_{DRAN}$. The above analysis shows that, for edges masks, the one among BRAN and DRAN whose area is smaller is the true local region.

For the corner masks, as the area of the true local region is less than $\frac{n_{MAX}}{2}$, the true local region is the one among BRAN and DRAN whose area is smaller than $\frac{n_{MAX}}{2}$. If $n_{BRAN} < \frac{n_{MAX}}{2}$, from equation (4.15), we can have $n_{DRAN} > n_{MAX} - n_{BRAN} > \frac{n_{MAX}}{2}$, thus $n_{DRAN} > n_{BRAN}$. Similarly if $n_{DRAN} < \frac{n_{MAX}}{2}$, we can have $n_{BRAN} > n_{DRAN}$. The above analysis shows that, for corner masks, the one among BRAN and DRAN whose area is smaller is the true local region.

For all the above situations, the true local region can be chosen as the one among BRAN and DRAN whose area is smaller and such a local region is named as BDRAN, an acronym standing for Brighter or Dark Region Assimulating Nucleus.

BDRAN can be expressed as follows:

$$BDRAN = \begin{cases} BRAN & \text{if} \quad n_{BRAN} \leq n_{DRAN} \\ DRAN & \text{if} \quad n_{BRAN} > n_{DRAN} \end{cases} \tag{4.16}$$

Obviously, the BDRAN area n_{BDRAN} is given by:

$$n_{BDRAN} = \min(n_{BRAN}, n_{DRAN}) \tag{4.17}$$

If we define the BRAN response as $c_{BRAN} = n - n_{BRAN}$, the DRAN response as $c_{DRAN} = n - n_{DRAN}$ and the BDRAN response $c_{BDRAN} = n - n_{BDRAN}$, we can compute c_{BDRAN} as the follows:

$$c_{BDRAN} = \max(c_{BRAN}, c_{DRAN}) \tag{4.18}$$

The local region constraint whose local region definition is given by BDRAN can be described as follows:

BDRAN Constraint:

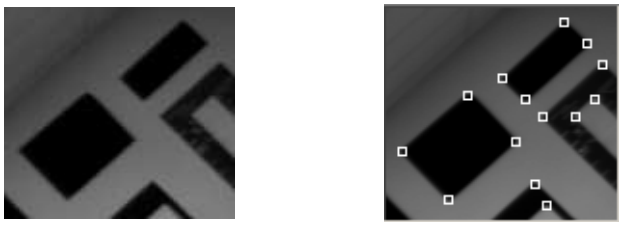
The BDRAN response of a flat point or an edge is always smaller than $g_c = \frac{n_{MAX}}{2}$, *while the BDRAN response of a corner is always bigger than* $g_c = \frac{n_{MAX}}{2}$.

4.6.1 Corner Detection Based on BDRAN constraint

A corner detection algorithm based on the BDRAN constraint can be described as follows:

Algorithm: Corner Detection Based on BDRAN Constraint

For each pixel, consider a circular mask centered at it and perform the following steps:



(a) (b)

Figure 4.8: A real image example to show the stability of BDRAN constraint. (a) one part of the lab image; (b) corner map of BDRAN constraint with $t = 25$.

1. *Find its BRAN and compute the BRAN response* c_{BRAN}.

2. *Find its DRAN and compute the DRAN response* c_{DRAN}.

3. *Compute its BDRAN response as* $c_{BDRAN} = \max(c_{BRAN}, c_{DRAN})$.

4. *If* $c_{BDRAN} > g_c$, *the current pixel will be marked as the corner candidate.*

5. *Use non-maximum suppression to ensure each corner will only be marked once.*

Fig.4.8.b shows the corner map of the above algorithm with $t = 25$ for the previous small lab image. All the true corners are correctly reported and no false alarms are given.

4.6.2 Analysis of BDRAN Constraint

Similar to the computation analysis of the USAN constraint, we can compute the BDRAN responses of the 1D ideal step edge and real edge as follows (refer to appendix 2 of this chapter):

$$\begin{aligned} c_{BDRAN}(x_E) &= r \\ c_{BDRAN}(\widehat{x}_E) &= r - x_R \end{aligned} \tag{4.19}$$

where x_R satisfies:

$$\int_0^{x_R} G(s)ds = \frac{t}{\nabla I} \tag{4.20}$$

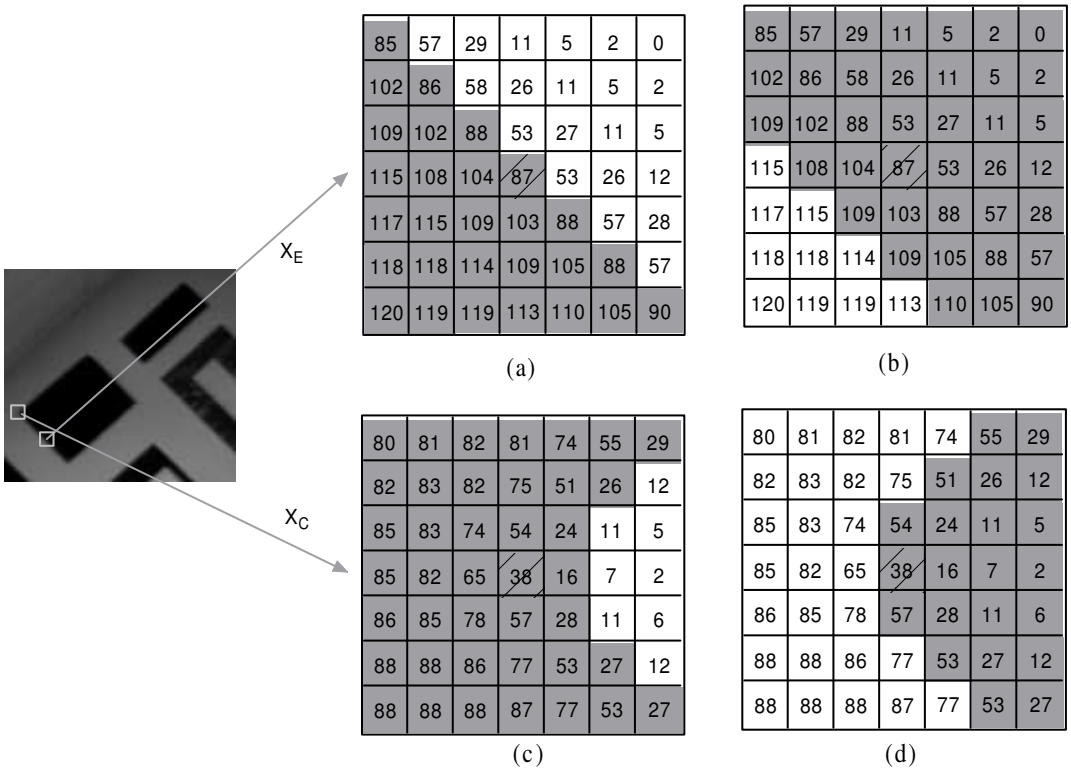


Figure 4.9: A visual example to explain the validity of BDRAN constraint. X_E and X_C are an edge pixel and a corner pixel of a small lab image respectively. The BRAN/DRAN computations are based on $t = 25$. (a) computed BRAN area at X_E (gray pixels); (b) computed DRAN area at X_E (gray pixels); (c) computed BRAN area at X_C (gray pixels); (d) computed DRAN area at X_C (gray pixels).

Obviously, BDRAN constraint holds very well for both ideal and real edges as their BDRAN responses are always no bigger than $g_c = r$ no matter what t is chosen. For a fixed t, the bigger ∇I corresponds to the smaller x_R, and $c_{BDRAN}(\widehat{x}_E)$ approaches g_c more. This is desirable because a stronger edge should have a bigger BDRAN response. But no matter how strong is the edge, its BDRAN response is always no bigger than g_c. Thus g_c can safely separate the corners from the edges.

We also give a visual explain to the above analysis as shown in fig.4.9. To compare with the USAN constraint, the same testing image is used here. The gray areas in fig.4.9.a and fig.4.9.b show the BRAN area and DRAN area of X_E respectively with $t = 25$. We can compute $c_{BRAN}(X_E) = 49-28 = 21$ and $c_{DRAN}(X_E) = 49-39 = 10$.

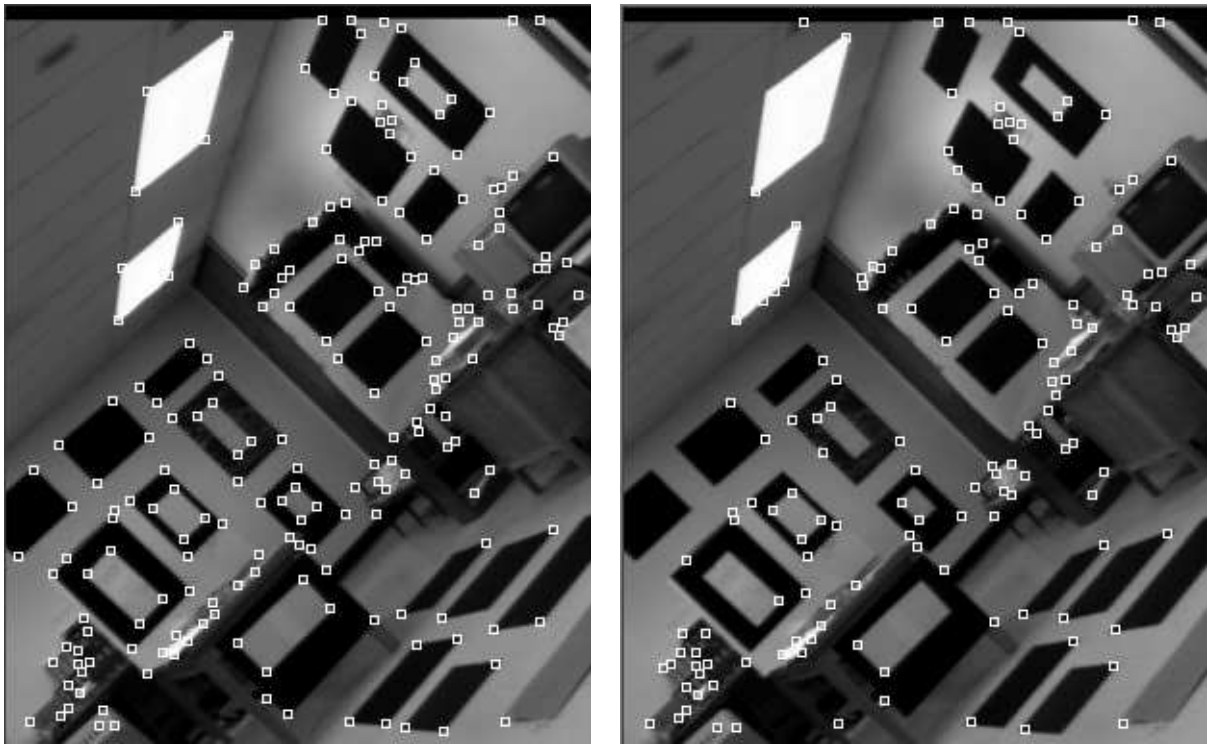
As $c_{BRAN}(x_E) > c_{DRAN}(x_E)$, BDRAN will choose BRAN as the true local region and $c_{BDRAN}(x_E) = 21$. The gray areas in fig.4.9.c and fig.4.9.d show the BRAN area and DRAN area of X_C respectively with $t = 25$. We can compute $c_{BRAN}(X_C) = 49-41 = 8$ and $c_{DRAN}(X_C) = 49-22 = 27$. As $c_{DRAN}(X_C) > c_{BRAN}(X_C)$, BDRAN will automatically choose DRAN as the true local region and $c_{BDRAN}(X_C) = 27$. Note that $g_c = 24.5$. Obviously the BDRAN constraint holds very well here as $c_{BDRAN}(X_E) < g_c$ and $c_{BDRAN}(X_C) > g_c$. Using g_c alone can easily separate the corners from the edges.

The above analysis has shown that BDRAN constraint holds very well for both ideal and real images. This is the reason why BDRAN constraint outperforms USAN constraint.

4.7 Experiments

We compare the performance of the BDRAN corner detector with the SUSAN corner detector in this part. Experiments are performed on both indoor and outdoor images. The experimental results are compared both visually and quantitatively.

The first experimental image is a lab scene with many rectangles hanging on the wall and the ground. The vertices of these rectangles are well-defined L-Junctions. We set the parameters of both BDRAN and SUSAN to 25 and the obtained corner maps are shown as fig.4.10 (a) and (b) respectively. BDRAN performs very well on this test image. Most of the desired corners are correctly reported. Only several corners of the vertices of the rectangles on the ground are missed. This is because their corner angles are too close to 180^o such that they appear more like edges than corners. On the contrary, SUSAN cannot work well on this image. Many obvious vertices of the rectangles are missed. Such a result is due to the false rejection of true corners by the two additions of USAN constraint. This verifies our previous

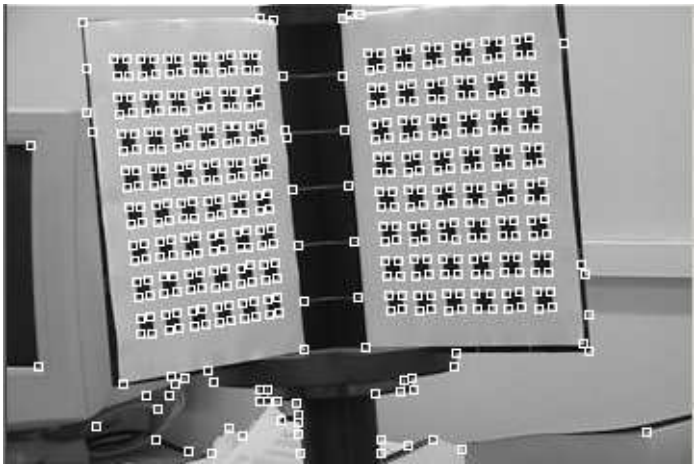


(a) (b)

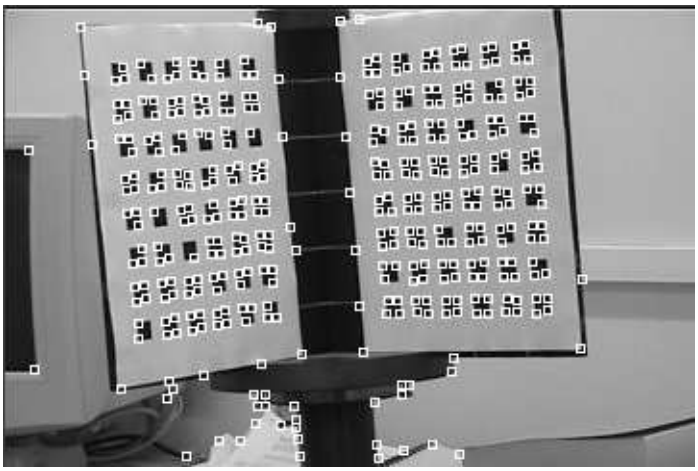
Figure 4.10: Corner maps of the lab image obtained by: (a) BDRAN with $t = 25$; (b) SUSAN with $t = 25$.

analysis.

The second experimental image contains a calibration grid. Such an image is widely used for camera calibration because the corners of the grid squares are organized regularly and can be easily matched with other images. To perform the calibration task, all the corners of the grid should correctly be found and well localized. The parameters of BDRAN and SUSAN are set to be 25 and 35 for this image, and their corner maps are shown in fig.4.11 (a) and (b) respectively. BDRAN outperforms SUSAN again for this image. For BDRAN, all the corners of the grid squares are found correctly and well localized, no false alarms are reported with the pixels on the grid. SUSAN misses quite some corners of the grid squares. What's more, some reported corners are not localized well. The collinearity of the corners on a line is not preserved and they are not separated equally also. Obviously, such a result



(a)



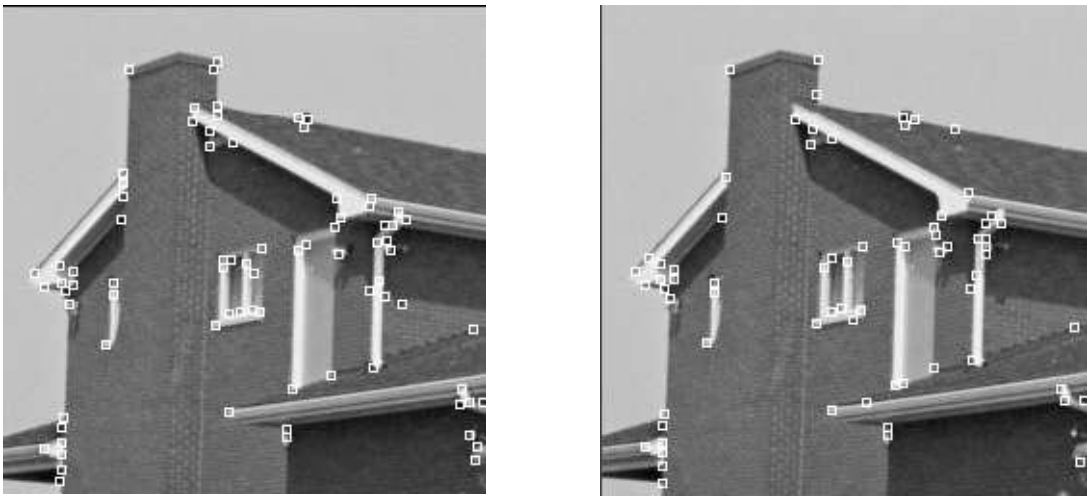
(b)

Figure 4.11: Corner maps of the calibration grid image obtained by: (a) BDRAN with $t = 25$; (b) SUSAN with $t = 35$.

cannot satisfy the requirement of camera calibration.

The next experimental image is an outdoor house image. Although the image contains a lot of noise and textures, its intensity changes over edges and corners are quite sharp. The corner detection results of BDRAN and SUSAN, with $t = 25$ for each method, are shown as fig.4.12.a and fig.4.12.b respectively. Both BDRAN and SUSAN perform quite well on this image and their detection results are quite similar. This shows that both methods are not sensitive to image noise and textures.

The last experimental image is an outdoor museum image, which is full of ill-defined



(a) (b)

Figure 4.12: Corner maps of the house image obtained by: (a) BDRAN with $t = 25$; (b) SUSAN with $t = 25$.

corners. The results are shown in fig.4.13 (a) and (b). Both of the thresholds are set to be 25. Both BDRAN and SUSAN show great resistances to image noise. Besides, BDRAN shows strong resistance to the image smoothness also. With most true corner being detected, it detects very few strong edges in this outdoor image. Compared with BDRAN, SUSAN suffers much from the image smoothness. Many strong edges are wrongly marked as corners (please pay attention to the roof edges).

We also compare BDRAN and SUSAN quantitatively on the first three test images. To do this, we mannully select the ground truth corners from the images. Then we compute the number of ground truth corners N_{GT}, the number of detected true corners N_T, the number of falsely detected corners N_F and the detection rate R_D as the overall performance indicator for each algorithm. R_D is computed using the following equation:

$$R_D = \frac{N_T}{N_F + N_{GT}} \tag{4.21}$$

To have a high R_D, a corner detector should be able to detect true corners as many as possible and report false corners as few as possible. The maximal value for R_D equals 1 where all the true corners are detected ($N_T = N_{GT}$) and no false corners are

(a) (b)

Figure 4.13: Corner maps of the museum image obtained by: (a) BDRAN with $t = 25$; (b) SUSAN with $t = 25$.

reported ($N_F = 0$). The comparison results are shown in fig.4.14. For the calibration image (fig.4.14.a), BDRAN detects much more true corners than SUSAN, while it reports only a little more false corners than SUSAN. So the overall performance of BDRAN is much better than SUSAN. For the lab image (fig.4.14.b), the situation is almost the same and the performance increase is obvious. For the house image (fig.4.14.c), both detectors give almost the same number of true corners, but SUSAN reports a little more false corners. The overall performance of BDRAN is slightly better than SUSAN.

Is our previous analysis caused by improper selection of threshold for SUSAN? We run both BDRAN and SUSAN on the first three test images over a wide range of thresholds, saying from 5 to 100, and plot their detection rates against the thresholds in fig.4.15. The solid and dotted lines show the detection rates of BDRAN and SUSAN respectively. Two important conclusions can be observed from these figures:

	BDRAN	SUSAN
N_{GT}	433	433
N_T	430	341
N_F	23	19
R_D	94%	75%

(a)

	BDRAN	SUSAN
N_{GT}	162	162
N_T	146	95
N_F	49	60
R_D	69%	43%

(b)

	BDRAN	SUSAN
N_{GT}	69	69
N_T	63	60
N_F	16	26
R_D	74%	63%

(c)

Figure 4.14: Quantitative comparisons for BDRAN and SUSAN. (a) the calibration image; (b) the lab image; (c) the house image.

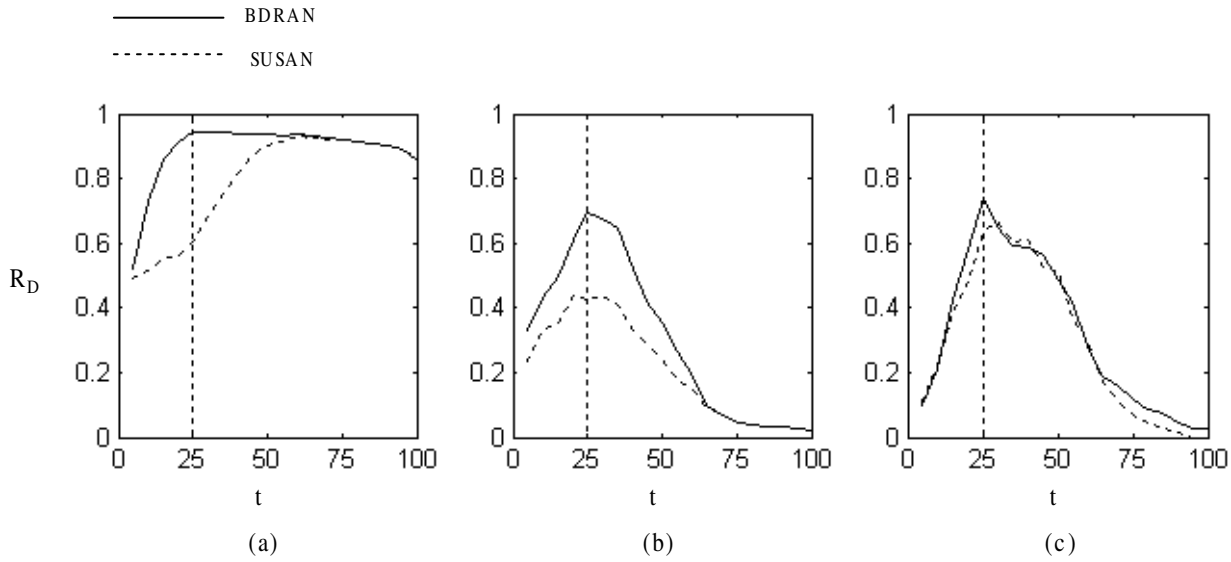


Figure 4.15: Corner detection performance with respect to the threshold for BDRAN and SUSAN. (a) the calibration image; (b) the lab image; (c) the house image.

- BDRAN has overall better detection performance than SUSAN. When the same threshold is used, the detection rate of BDRAN is almost always bigger than that of SUSAN. Also the peak detection rates of BDRAN are bigger than those of SUSAN for all the images. The improvement is most prominent for the lab image, where $\max(R_D) = 0.69$ for BDRAN and $\max(R_D) = 0.43$ for SUSAN. BDRAN has achieved a 60% increase in the maximal detection rate.

- The detection rate of BDRAN is much less sensitive to the threshold selection than SUSAN. For BDRAN, the peaks of R_D all correspond to $t = 25$ for all the images. Also notice that BDRAN gives quite stable R_D with thresholds

around 25. This means that its performance does not critically depend on the threshold selection. $t = 25$ is suitable for most cases. But the situation is different for SUSAN. For the calibration image, a high threshold (bigger than 45) gives more reasonable detection performance. But for the lab and house images, the detection rates are high for the thresholds within the range of 20 to 40. This shows that, to give a good detection result for SUSAN, we need to tune its threshold carefully. This is a very undesired property when it is applied to high level applications.

4.8 Conclusion

This part analyzed the stability performance of the well-known SUSAN corner detector and concludes that it cannot work stably on real images due to the violation of USAN constraint by the edges of real images. To solve this problem, the USAN constraint is extended to the general local region constraint. By replacing USAN with a new local region definition namely BDRAN, the corresponding local region constraint holds very well with the real images and this results in a stable corner detection algorithm. Experiments on real images were presented to show the novelty of the newly developed approach.

Appendix1: USAN Response Computation of Edges

This part explains the USAN response computation for the 1D ideal step edge and 1D real edge. For each edge signal, we assume the exact edge is located at $x_E = 0$. The mask $M(x_E)$ centered at x_E is the line segment $[-r, r]$, where r is the mask radius. $M(x_E)$ can be divided into the left part $M_1(x_E) = [-r, 0]$ and the right part $M_2(x_E) = [0, r]$. Obviously, the mask area $n_{MAX} = 2r$. Using $n_{USAN1}(x_E)$ and

$n_{USAN2}(x_E)$ to represent the USAN areas of $M_1(x_E)$ and $M_2(x_E)$ respectively, the USAN response of the edge is given by:

$$c_{USAN}(x_E) = n_{MAX} - n_{USAN}(x_E) = 2r - n_{USAN1}(x_E) - n_{USAN2}(x_E) \tag{4.22}$$

USAN Response of Step Edge

An ideal 1D step edge signal can be modelled as:

$$I(x) = \begin{cases} I_1 & \text{if} \quad x < 0 \\ I_2 & \text{if} \quad x \geq 0 \end{cases} \tag{4.23}$$

Without loss of generalization, we can assume that $I_1 < I_2$. The whole signal can be considered as two regions: the black one $x < 0$ and the white one $x \geq 0$. They are separated by the exact edge position $x_E = 0$. Use t to represent the brightness threshold for USAN. A typical t should satisfy $t < I_2 - I_1$ for us to distinguish the dark region and the bright region.

For any point x to have the similar brightness as x_E, $I(x)$ should satisfy:

$$I(x_E) - t \leq I(x) \leq I(x_E) + t \Longrightarrow I_2 - t \leq I(x) \leq I_2 + t \tag{4.24}$$

For any point $x_1 \in M_1(x_E)$, as $I(x_1) = I_1$ always holds, $I(x_1)$ never satisfies the above equation which means x_1 never belongs to the USAN of x_E, thus $n_{USAN1}(x_E) = 0$. For any point $x_2 \in M_2(x_E)$, as $I(x_2) = I_2$ always holds, $I(x_2)$ always satisfies the above equation which means x_2 always belongs to the USAN of x_E, thus $n_{USAN2}(x_E) = r$. So the USAN response of x_E is given by:

$$c_{USAN}(x_E) = r \tag{4.25}$$

USAN Response of Real Edge

The real edge signal can be modelled by convolving the ideal step edge signal with the Gaussian smoothing function $G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$. σ is the standard deviation, which controls the smoothness degree. A typical real edge signal can be expressed as:

$$\hat{I}(x) = I_1 + \nabla I * \int_{-\infty}^{x} G(s)ds \tag{4.26}$$

in which $\nabla I = I_2 - I_1$ is the brightness contrast of $I(x)$. Obviously, the exact edge $x_E = 0$ has the brightness:

$$\hat{I}(x_E) = I_1 + \nabla I * \int_{-\infty}^{0} G(s)ds = \frac{I_1 + I_2}{2} \tag{4.27}$$

For any point x to have the similar brightness as x_E, $I(x)$ should satisfy:

$$I(x_E) - t \leq I(x) \leq I(x_E) + t \Longrightarrow \frac{I_1 + I_2}{2} - t \leq I(x) \leq \frac{I_1 + I_2}{2} + t \tag{4.28}$$

As equation (4.26) is monotonically increasing, if x_L and x_R are the smallest and the biggest numbers within $M(x_E)$ that satisfy the above equation, all the points within the range of $[x_L, x_R]$ will belong to the USAN of x_E, which means $c_{USAN} = x_R - x_L$. Obviously, x_L and x_R can be computed by:

$$\begin{array}{rcl} I_1 + \nabla I * \int_{-\infty}^{x_L} G(s)ds & = & \frac{I_1+I_2}{2} - t \\ I_1 + \nabla I * \int_{-\infty}^{x_R} G(s)ds & = & \frac{I_1+I_2}{2} + t \end{array} \tag{4.29}$$

or equivalently:

$$\begin{array}{l} \int_{x_L}^{0} G(t)dt = \frac{t}{\nabla I} \\ \int_{0}^{x_R} G(t)dt = \frac{t}{\nabla I} \end{array} \tag{4.30}$$

This shows that $x_L < 0$, $x_R > 0$ and $x_L = -x_R$. So the USAN response of x_E is

given by:

$$c_{USAN}(\widehat{x_E}) = 2r - 2x_R \tag{4.31}$$

where x_R can be computed from equation (4.30).

Appendix2: BDRAN Response Computation of Edges

This part explains the BDRAN response computation for the 1D ideal step edge and 1D real edge. As $c_{BDRAN}(x_E) = \max(c_{BRAN}(x_E), c_{DRAN}(x_E))$, to compute the BDRAN response, we need to compute the BRAN response and the DRAN response first.

Similar to the USAN computation, the mask $M(x_E)$ is divided into the left part $M_1(x_E) = [-r, 0]$ and the right part $M_2(x_E) = [0, r]$ again. Use n_{BRAN1} and n_{BRAN2} to represent the BRAN areas of these two parts respectively. If any $x_1 \in M_1(x_E)$ belongs to BRAN, its brightness $I(x_1)$ satisfies $I(x_1) \geq I(x) - t$. Also, as $I(x_1) \leq I(x) < I(x) + t$ always holds, any x_1 that belongs to BRAN also belongs to USAN, which means $n_{BRAN1}(x_E) = n_{USAN1}(x_E)$. For any $x_2 \in M_2(x)$, as $I(x_2) \geq I(x) > I(x) - t$ always holds, we can have $n_2^{BRAN} = r$. Thus the BRAN response is given by:

$$c_{BRAN}(x_E) = r - n_{USAN1}(x_E) \tag{4.32}$$

Similarly, if we use $n_{DRAN1}(x_E)$ and $n_{DRAN2}(x_E)$ to represent the DRAN areas of $M_1(x_E)$ and $M_2(x_E)$ respectively. We can have $n_{DRAN1}(x_E) = r$ and $n_{DRAN2}(x_E) = n_{USAN2}(x_E)$. Thus the DRAN response is given by:

$$c_{DRAN}(x_E) = r - n_{USAN2}(x_E) \tag{4.33}$$

The above two equations hold for both ideal and real edges.

For the ideal edge, as $n_{USAN1}(x_E) = 0$ and $n_{USAN1}(x_E) = r$, the BRAN and DRAN responses are given by:

$$\begin{aligned} c_{BRAN}(x_E) &= r \\ c_{DRAN}(x_E) &= 0 \end{aligned} \tag{4.34}$$

this gives the BDRAN response as:

$$c_{BDRAN}(x_E) = \max(c_{BRAN}(x_E), c_{DRAN}(x_E)) = r \tag{4.35}$$

For the smoothed edge, as $n_{USAN1}(\widehat{x}_E) = -x_L = x_R$ and $n_{USAN2}(\widehat{x}_E) = x_R$, the BRAN and DRAN responses are given by (equations (4.32) and (4.33)):

$$c_{BRAN}(\widehat{x}_E) = c_{DRAN}(\widehat{x}_E) = r - x_R \tag{4.36}$$

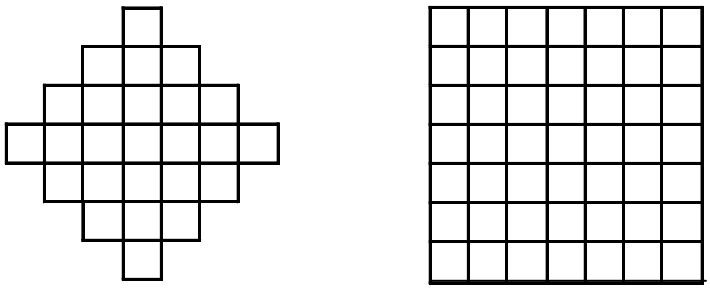
this gives the BDRAN response as:

$$c_{BDRAN}(x_E) = \max(c_{BRAN}(x_E), c_{DRAN}(x_E)) = r - x_R \tag{4.37}$$

where x_R can be computed from equation (4.30).

Appendix3: Digital Circular Mask

Due to the symmetric reason, the circular mask is preferred for the corner detection task. But the ideal circular mask does not exist for digital images, so certain approximations will be used. A circular mask should have the same distance from its central pixel to its boundary pixels. So the shape of the digital circular mask will depend on how the distance is defined. There are two popular definitions. Consider two pixels p and q with coordinates (x, y) and (s, t), D_4 distance is defined as



(a) (b)

Figure 4.16: Two commonly used digital circular masks with radius 3 pixels. (a) the mask corresponding to D_4 distance; (b) the mask corresponding to D_8 distance.

$D_4(p,q) = |x-s|+|y-t|$ and D_8 distance is defined as $D_8(p,q) = max(|x-s|, |y-t|)$. These two distance definitions correspond to two commonly used digital circular masks, as shown in fig.4.16 (with radius 3 pixels). Notice that the mask shape generally does not affect the algorithm performance. For convenience, we choose the second one for analysis in our thesis.

Chapter 5

Corner Detection Based on Modified Hough Transform

5.1 Introduction

Although the corner detection algorithm proposed in the last chapter works well for most situations, it has several intrinsic shortcomings due to the used edge and corner models. First, as it assumes step edge models by default, it has high possibility to mark some roof edges as corners by mistake. Also, as it assumes L-Junction corner models by default, it has some difficulty to correctly mask some X-Junction corners. To solve the above problems, a corner detector based on a more general model for edges and corners is highly desired.

In this chapter, the problem of detecting corners is simplified into detecting simple lines (straight lines that pass through the coordinate origin) in a local coordinate system. As simple lines can be expressed using only one parameter, Hough transform can detect them quickly. Other edge detectors have some problems when marking edge points around corners. Instead of using a general global edge detector, we

detect edge points locally based on both gradient magnitude threshold and gray level analysis. The process is dynamic and can provide correct edge points for Hough transform. Gradient direction is used to reduce the effect of noise and speed up the algorithm. Experiments will be presented to show that the new algorithm works well over a wide kind of images, and is fast enough for real time applications.

This chapter is organized as follows. A more general edge and corner model is introduced in section 5.2. A local edge detection approach based on both gradient magnitude threshold and gray level analysis is detailed in section 5.3. In section 5.4, Hough transform is modified to organize the local edges into simple lines. Section 5.5 gives a brief analysis on that how the new algorithm can deal with X-Junction corners and roof edges correctly. Section 5.6 gives some experiments to compare the new algorithm with other popular corner detectors. Section 5.7 concludes this chapter.

Notations:

x_0	the nucleus (central pixel) of a mask
x	an arbitrary pixel within the mask
$I(x)$	the gray level intensity of a pixel
I_u	the image partial derivative along the u direction
I_v	the image partial derivative along the v direction
∇I	the gradient of the image
$\|\nabla I\|$	the gradient magnitude of the image
θ	the gradient direction of the image
Edge Point	a pixel whose edge response is large enough under certain edge measurement
Straight Line	a straight line in an image formed by some edge points
Simple Line	a straight line that passes through the coordinate origin

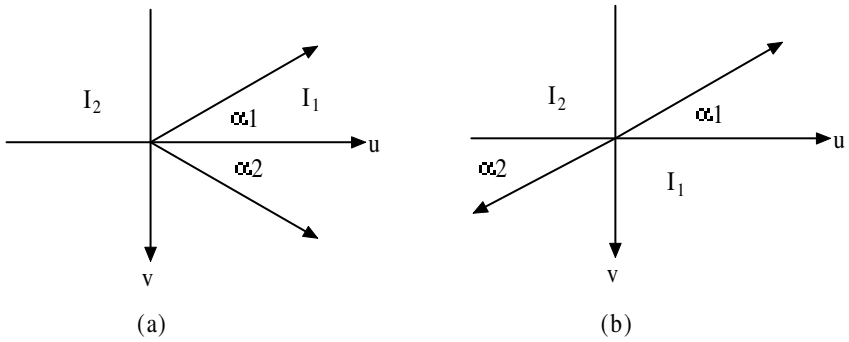


Figure 5.1: (a) An ideal L-Junction model, (b) an ideal edge model

5.2 Corner Model Description

In the following parts, using "edge point", we mean such a pixel, under certain edge measurement, whose edge response is large enough. Using "straight line", we mean a straight line in an image formed by some edge points. Using "simple line", we mean a straight line that passes through the coordinate origin.

A corner is defined as the intersection of two or more straight lines. Based on this definition, two characteristics of corners can be generalized. Firstly, a corner point is also an edge point. Secondly, at least two lines pass through the corner point. The purpose of our corner detection approach is to find points that are consistent with the corner definition and possess these two characteristics.

A local coordinate system is used in our method. Assume that there is only one corner point and this point is set to be the origin in such a system. By this definition, less parameters are needed to describe a corner model. Fig.5.1.a shows an ideal L-Junction example. Two lines intersect at the origin, dividing the whole region into two sub regions. Using (α_1, α_2) to represent the orientations of the two corner boundary lines and (I_1, I_2) to represent the gray level values of two sub regions, this corner model can be described as $E(\alpha_1, \alpha_2, I_1, I_2)$. Similarly, other complex corner types, such as T-Junction, Y-Junction, X-Junction and so on, can also be

described in this way. Let n represents the number of sub-regions generated by corner boundaries, $E(\alpha_1, ..., \alpha_n, I_1, ..., I_n)$ is a suitable description for all kinds of corners. If we decompose a straight line into two half lines, the edge model in the local coordinate system (fig.5.1.b) can also be described as $E(\alpha_1, \alpha_2, I_1, I_2)$, which is similar with the description of L-Junction. The difference is the relationship between α_1 and α_2. If $\alpha_1 \approx \alpha_2$, the origin is an edge point. Otherwise it is a corner point.

We move a window with a suitable size pixel by pixel to detect corners. Each window is treated as a local coordinate system, with the central pixel being the origin. All the pixels inside the window constitute a region of interest, whose information will be used to determine if the central pixel is a corner point. If the central pixel is a corner point, the following two conditions should be satisfied.

- Under certain edge measurement, its edge response should be large enough, as corner point should also be an edge point.
- We should be able to detect at least two straight lines passing through it, that is at least two simple lines should exist. Considering the advantages of Hough transform in detecting straight lines, we modify it to check the possible simple lines in the local coordinate system.

5.3 Local Edge Detection

Suppose we are working on a window whose central pixel is an edge point. To detect simple lines, edge points that have a high likelihood of being on simple lines should been selected firstly. We argue that the edge points generated by common edge detectors are not suitable for the process. The reasons can be generalized as below. Firstly, for edge detectors that depend on Gaussian smoothing, there is obviously rounding effect at corner neighborhood, which leads to the ill localization of corner

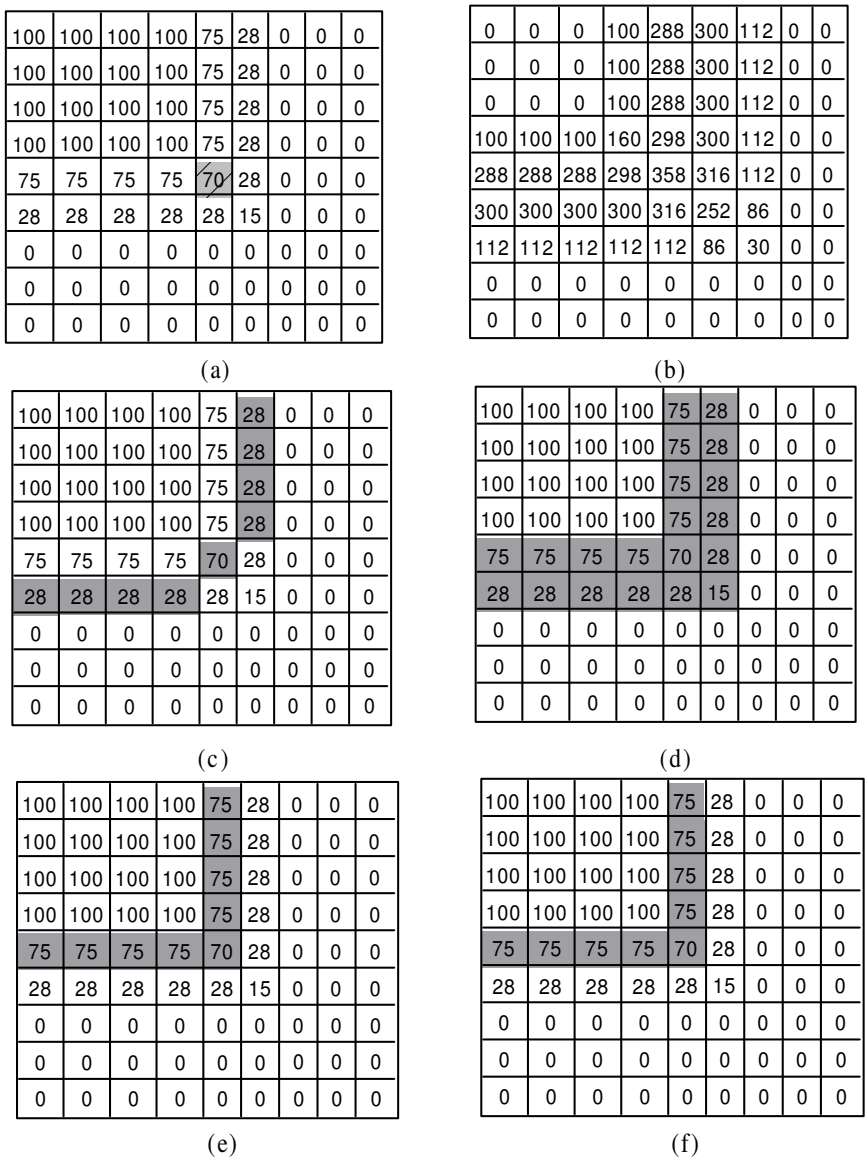


Figure 5.2: (a) A L-Junction model smoothed by Gaussian, (b) the gradient magnitude map, (c) edges produced by Sobel edge detector, (d) pixels that satisfy equation (5.1) with $t_1 = 200$, (e) pixels that satisfy equation (5.2) with $t_2 = 20$, (f) edges produced by our local edge detection method.

position. Secondly, the non-maximum suppression used in common edge detectors will make the straight lines curved. This effect can be seen from fig.5.2. Fig.5.2.a is an ideal L-Junction example smoothed by Gaussian function. The central pixel, marked as gray, is the exact corner position. The gradient magnitude of each pixel is given in fig.5.2.b. Fig.5.2.c shows the edge detection result of Sobel for the previous figure, where non-maximum suppression of gradient magnitude is used. Edge points are highlighted as gray pixels. These edge points are more likely to form one curved line than two straight lines, which is not desirable.

To overcome the problems mentioned above, we detect the edge points window by window locally instead of finding them globally. The edge response of each pixel in the window is given by its gradient magnitude: $|\nabla I|$. To make sure that the lines are only one pixel wide, we need a thinning algorithm, but non-maximum suppression is not qualified based on above analysis. Remember that the corner point is the intersection of two or more straight lines. So it's natural for us to require that the edge points that form the simple lines and the corner point belong to same sub region, which means their gray level values are similar. This constraint not only thins the edge points effectively, but also reduces the noise effect strongly, taking the place of Gaussian smoothing.

For any pixel x in the region of interest, if the following two equations are satisfied, it has a high likelihood of being on simple lines.

$$|\nabla I(x)| > t_1 \tag{5.1}$$

$$|I(x) - I(x_0)| < t_2 \tag{5.2}$$

where x_0 represents the central pixel and t_1, t_2 are the thresholds to be determined. Obviously, the detected edge points are subjective to the window on which we are currently working. So our edge detection scheme is a dynamic process. Applying the

above edge detection method to fig.5.2.a with thresholds $t_1 = 200$ and $t_2 = 20$, the gray pixels in fig.5.2.d and fig.5.2.e show those pixels that satisfy the equations (5.1) and (5.2) respectively. The final edge pixels, i.e. those that satisfy both equations, are drawn as gray in fig.5.2.f. The detected edges fit our requirements very well, as two simple lines are formed nicely.

To speed the algorithm and reduce the effect of noisy points, we further discard some edge points making use of the gradient direction information. Considering an edge point $x = (u, v)$, there is only one simple line that passes through it and this line can be described as:

$$ucos\theta + vsin\theta = 0 \tag{5.3}$$

θ is the gradient direction of the simple line. So the gradient direction can be computed using:

$$\theta = arctg(-\frac{v}{u}) \tag{5.4}$$

Also an edge point's gradient direction is determined by its partial derivatives:

$$\theta' = arctg(\frac{I_v}{I_u}) \tag{5.5}$$

θ is the ideal value of gradient direction and θ' is the real version. They should equal to each other if this pixel is really on simple lines. But θ' cannot be accurately estimated because of the discrete property of digital image and the noise affect. So we need to set a threshold to determine the similarity between θ and θ'. The edge points whose θ and θ' don't satisfy the following equation will be discarded.

$$|\theta - \theta'| < t_3 \tag{5.6}$$

where t_3 is the angle threshold.

5.4 Simple Line Detection

Assume we have a set of edge points that have a high likelihood of being on simple lines. Hough transform is modified to organize these points into simple lines, whose parameter equation is:

$$usin\theta + vcos\theta = 0 \tag{5.7}$$

There is only one parameter θ in this equation. Regarding (u, v) as fixed, the equation corresponds to one value $\theta = arctg(-\frac{v}{u})$ in the θ space, or parameter space. All edge points on the same simple line will correspond to the same θ in parameter space. As points that even lie on a non-simple line contribute to different θ values, the modified Hough transform reports simple lines steadily. Assume the sliding window size is n*n. The most outer circle of the window consists $(n-1)*4$ pixels. Thus we can get an angle resolution of $\frac{360}{(n-1)*4}$ degrees. This means that two θs differing more than $\frac{360}{(n-1)*4}$ degrees represent two different simple lines.

Line extraction with ordinary Hough transform has a computational complexity $O(n*m)$, where n is the number of edge points and m is the size of a discretization of the parameter space. Reducing the size of parameter dimension can speed Hough transform significantly. Our implementation of Hough transform is only one dimensional, which makes our corner detector fast enough for real time applications.

5.5 Analysis

This part analyzes the insufficiency of the corner detector algorithm based on the local region constraint with an ideal X-Junction model and an ideal roof edge model, and shows how the new algorithm can compensate the potential shortcomings.

Fig.5.3.a shows an ideal image which consists four squares. The gray level value of

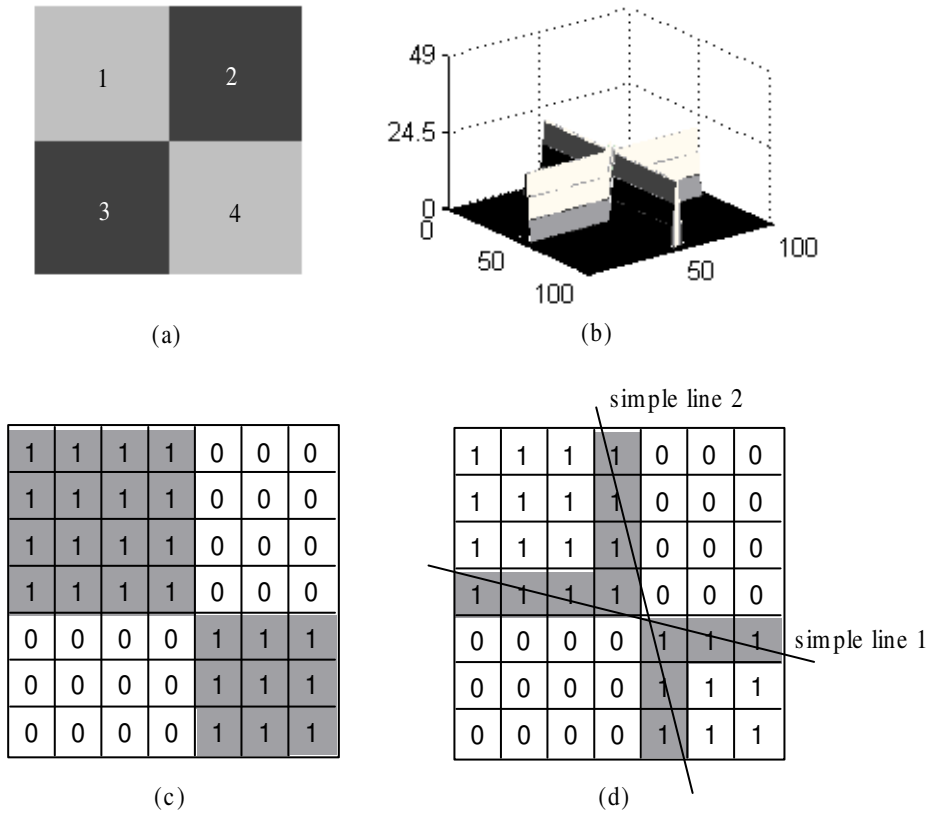
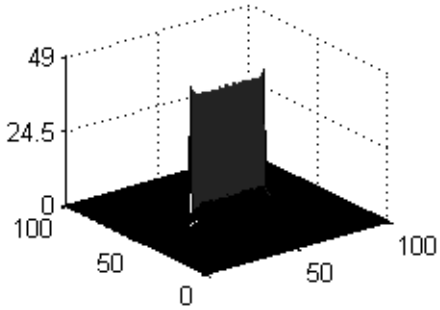


Figure 5.3: (a) An ideal X-Junction model, (b) its local region area response, (c) the local region of the vertex of square 1 (marked as gray), (d) the local edges of the vertex of square 1 (marked as gray).

square 1 equals to square 3, and the gray level of square 2 equals to square 4. All the four vertexes of these four squares are well-defined X-Junctions. Fig.5.3.b shows the 3D plot of the local region response distribution with a 7×7 mask (both USAN and BDRAN give the same result). It is clear that the local region responses of all the corners are as low as the edges and are smaller than $g_c = 24.5$ (g_c is the corner threshold which equals to the half of the mask size). The reason is obvious if we take a close look at one mask example. Fig.5.3.c shows the mask centered at the corner of the square 1 and the gray area is the corresponding local region. Obviously, the local region is contributed by two parts: 16 pixels of square 1 and 9 pixels of square 2. The local region response therefore equals to $49 - 16 - 9 = 24 < g_c$. Thus it will

(a)



(b)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1	1	1	1	1	1	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(c)

1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	1	1	1
0	0	0	0	1	1	1
0	0	0	0	1	1	1
0	0	0	0	1	1	1

simple line 1

(d)

Figure 5.4: (a) A straight line of one pixel width, (b) its local region area response, (c) the local region of one edge (marked as gray), (d) the local edges of one edge (marked as gray).

not be marked as a corner as expected. The main reason for this is that the local region constraint is developed based on L-Junction models and it does not take into account of the connectivity property, which is always required for a global region. Fig.5.3.d shows the edges (marked as gray) given by our local edge detection scheme with the same mask. They form up two simple lines, thus the central pixel will be marked as a corner by the new algorithm.

Fig.5.4.a shows a straight line of one pixel wide. All the edges of the line are ideal roof edges. Fig.5.4.b shows the 3D plot of the local region response distribution with a 7×7 mask (both USAN and BDRAN give the same result). It is clear that the local region responses of all the edges have very big local region responses

and they have high possibilities to be marked as corners. The output of corners will be randomly selected by non-maximum suppression, which makes both their detection and localization not stable. Fig.5.4.c shows the mask centered at one edge and the gray area is the corresponding local region. Obviously, the local region response equals to $49 - 7 = 42 > g_c$. The main reason for this is that the local region constraint is developed based on the step edge models and does not take into account of the roof edge models. Fig.5.4.d shows the edges (marked as gray) given by our local edge detection scheme with the same mask. They form up one simple line, thus the central pixel will be discarded by the new algorithm.

The algorithm proposed here can work very well with those corners intersected by straight lines. But due to the basic assumption about the corners, i.e. a corner is formed by the intersection of two or more straight lines, this approach generally cannot deal with corner points on curves. This is one limitation of the algorithm. Fortunately the BDRAN corner detector proposed in the previous chapter can work very well with curve corners.

5.6 Experiments

In this part, we compare the performance of our new corner detector with other three widely-used corner detectors, which are Wang and Brady corner detector (WANG-BRADY), Plessey corner detector (PLESSEY) and SUSAN corner detector (SUSAN). Experiments are performed on synthetic image and real images.

5.6.1 Parameters selection

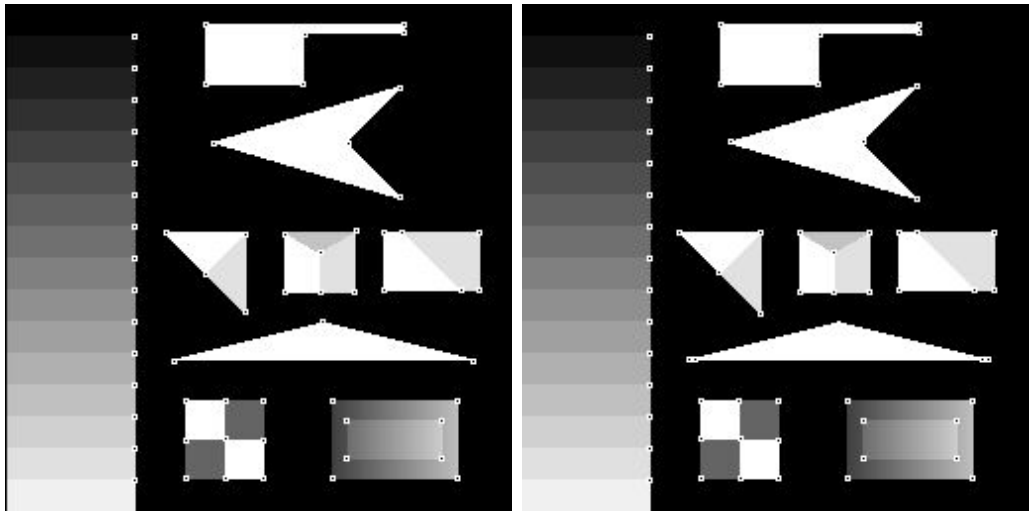
Three parameters, which are set manually, must be provided to the new corner detector. Parameter t_1 is the threshold of edge strength. The value of t_1 is not

critical and values from 20 to 200 (default 100) were found to be suitable. Parameter t_2 is the threshold for brightness comparison. This parameter is of the same nature as the brightness threshold for SUSAN corner detector [74]. For low contrast image, lower t_2 is wanted and vice-versa. The suitable values for t_2 are in the range from 5 to 35 (default 25). Parameter t_3 is the threshold for angle comparison. As this value is reduced, some corners whose angles approach 0^o or 180^o will be missed, but the algorithm can be more resistant to noise. We found that values ranging from 10 to 25 (default 18) work well. As our algorithm works window by window, the sliding window size also affects the algorithm result. Too small a window size will make our algorithm sensitive to noise and image digitization, while too large a window will slow the algorithm speed. From experiments, we found that 7*7 window is the best choice. It is noted that the choice of parameters is not critical for our corner detector. The ranges of threshold values given above will provide reasonable response for a wide range of situations.

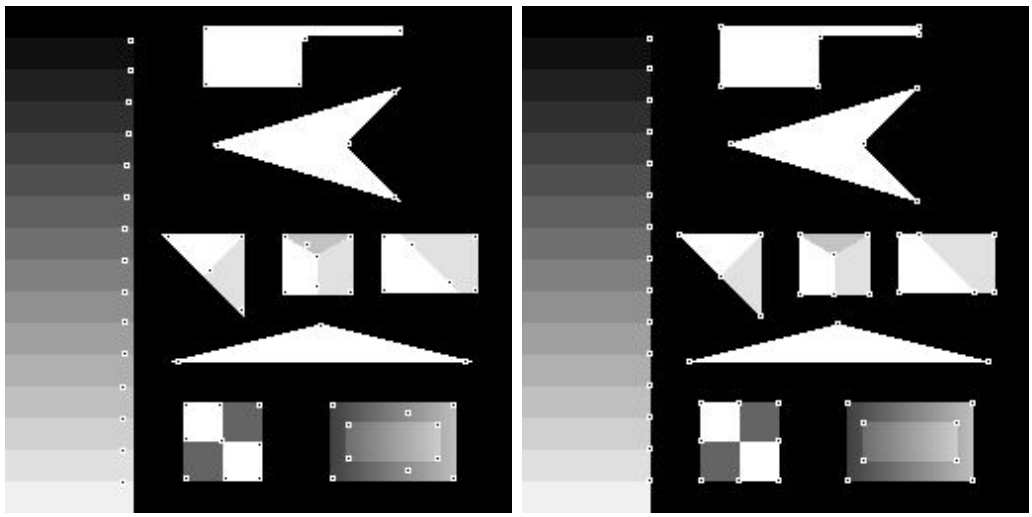
For other three corner detectors, a detail description of setting the parameters can be found in [79]. We mention it roughly here. Wang-Brady requires three parameters: t_1, t_2 and S. t_1 defines the edge gradient that each edge point should have. The suitable value ranges from 20 to 200. t_2 is the corner response threshold and the suitable value ranges from 500 to 20000. S is a measure of image curvature used for the suppression of the false corners and the recommended value is in the range from 0.0 to 0.5. Plessey needs only one threshold. Its suitable range is from 50 to 500. SUSAN also needs to tune one parameter, brightness threshold, depending on the contrast of the image. The suitable value is from 5 to 30.

5.6.2 Algorithm Performance

In all the experiments, the control parameters of each method have been selected in order to obtain the best possible results.



(a) (b)



(c) (d)

Figure 5.5: Corner maps of the synthetic image obtained by: (a) New Algorithm; (b) Wang and Brady Corner Detector; (c) Plessey Corner Detector; (d) SUSAN Corner Detector

First, we compare the performance of these four corner detectors on a synthetic image. This synthetic image consists of many corner types, including L-Junction, Y-Junction, T-Junction, Arrow-Junction and X-Junction. It has been widely used to evaluate how accurately an corner detector responds to different corner types. Fig.5.5.a was its corner map obtained by our detector with parameters set to 50/10/19 ($t_1/t_2/t_3$). Fig.5.5 (b) to (d) showed the corner images produced by WANG-BRADY, PLESSEY and SUSAN individually with parameters to be 50/1000/0.0 ($t_1/t_2/S$, WANG-BRADY), 50 (PLESSEY) and 10 (SUSAN). From the result, we can see that all the corners in this image are correctly reported by our corner detector. No wrong corners were reported. Wang-Brady corner detector missed one corner, at the obtuse angle of the triangle, and this is because the angle is very close to 180^o, so it appears as an edge point, not a corner. Also, Wang-Brady corner detector spotted spurious corners when the angles are very sharp. This is because the first order derivatives operators perform badly on this synthetic image. This shows the disability of WANG-BRADY of detecting corners whose angle is very large or very small. Although Plessey corner detector reported all the corners, it marked three corners wrongly. What's more, Plessey corner detector suffers great deviation in corner localization. In the T-Junctions at the low part of the ladder, the localization deviation is even as large as 4 pixels. This is caused by smoothing first order derivatives of the image. SUSAN performs quite well too. But it missed the X-Junction in the middle of the square at the bottom of the synthetic image and the reason for this has been analyzed previously.

We also tested the corner detectors on a house image. This image is corrupted by a lot of noise and textures. The result of our new corner detector is shown in fig.5.6.a. We can see that the new corner detector can filter the noise and textures effectively. Keeping the main corners detected, our algorithm reported few noisy corners. The results of other three corner detectors are shown from fig.5.6 (b) to (d). Wang and Brady corner detector produces wrong responses at some noisy points. Although

(a) (b)

(c) (d)

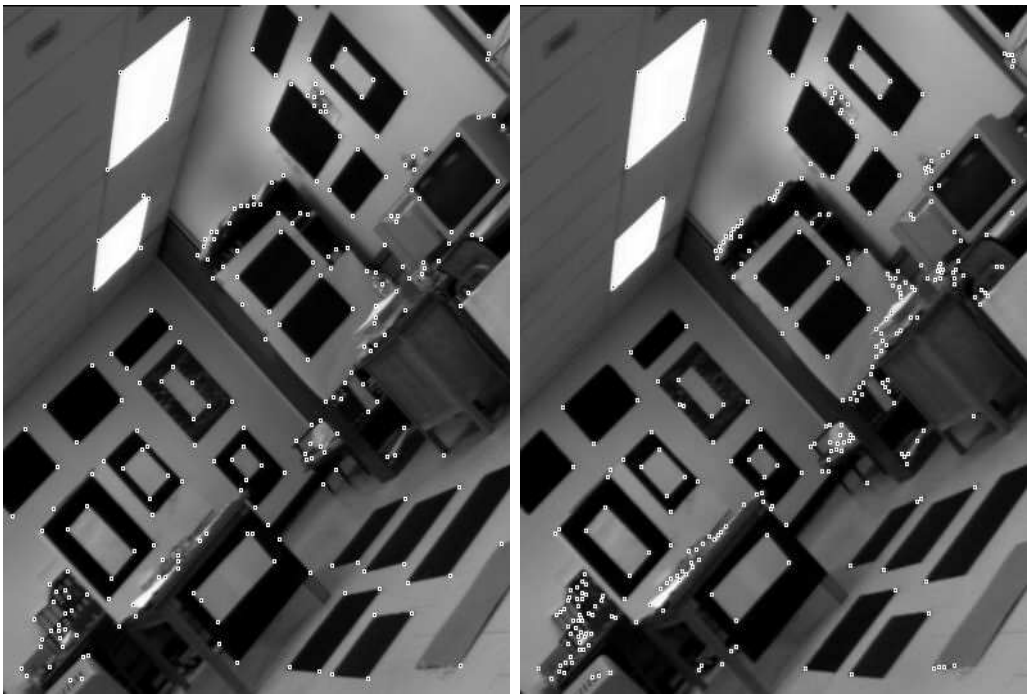
Figure 5.6: Corner maps of the house image obtained by: (a) New Algorithm; (b) Wang and Brady Corner Detector; (c) Plessey Corner Detector; (d) SUSAN Corner Detector

Plessey detected almost all the corners in the image, the result shows its sensitivity to noise. SUSAN is less sensitive to noise. The result is some better than Wang-Brady and Plessey. In this experiment, the parameters were set to be 130/30/18 ($t_1/t_2/t_3$, New Detector), 100/10000/0.2 ($t_1/t_2/S$, WANG-BRADY), 120 (PLESSEY) and 25 (SUSAN) individually.

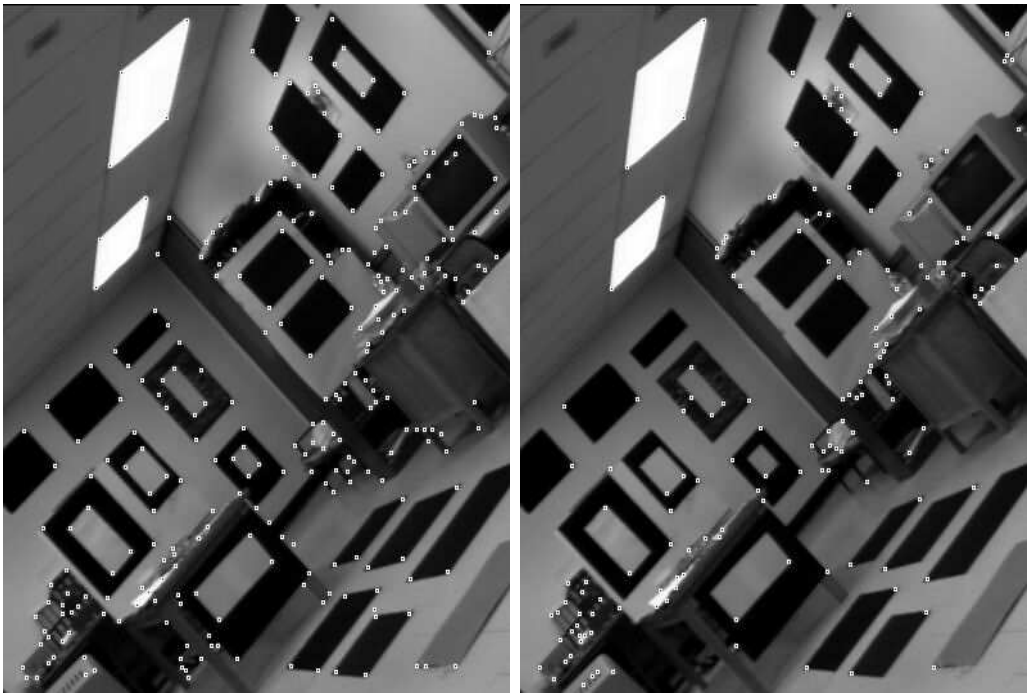
The next experimental image is an indoor scene. There are many rectangles with well-defined corners in the image. The boundaries of these rectangles are smoothed in a certain degree. We can test the ability of these detectors to detect smooth corners based on this image. We set the parameters of these four corner detectors to be 130/30/18 ($t_1/t_2/t_3$, New Detector), 100/8000/0.2 ($t_1/t_2/S$, WANG-BRADY), 100 (PLESSEY) and 25 (SUSAN) and obtained the corner maps shown as fig.5.7 (a) to (d). Our algorithm and Plessey corner detector performed best in this test image. Almost all the corners were detected successfully. WANG-BRADY missed some corners that are corrupted by noise and reported some noisy points. Although SUSAN is less sensitive to noise, it missed many smoothed corners, which shows it can not work well on highly smoothed images.

The last experiment is one natural scene full of ill-defined corners. The results are shown in fig.5.8 (a) to (d). The parameters were set to be 130/25/18 ($t_1/t_2/t_3$, New Detector), 80/8000/0.2 ($t_1/t_2/S$, WANG-BRADY), 110 (PLESSEY) and 20 (SUSAN). WANG-BRADY missed some corners whose angles approach 180^o and responded to many noisy points. SUSAN suffered from the smoothness of the image much, but it had good resistance to noisy points. Our corner detector and PLESSEY had better performance than the others in this image. PLESSEY detected more subtle corner points and it worked quite well for L-Junctions. Our corner detector can filter noisy points efficiently and resisted the smoothness of the image. Also our corner detector got more accurate corner locations than PLESSEY.

We also compare these corner detection algorithms quantitatively on the first three



(a) (b)



(c) (d)

Figure 5.7: Corner maps of the lab indoor image obtained by: (a) New Algorithm; (b) Wang and Brady Corner Detector; (c) Plessey Corner Detector; (d) SUSAN Corner Detector

(a) (b)

(c) (d)

Figure 5.8: Corner maps of the nature scene obtained by: (a) New Algorithm; (b) Wang and Brady Corner Detector; (c) Plessey Corner Detector; (d) SUSAN Corner Detector

test images. To do this, we mannully select the ground truth corners from the images. Then we compute the number of ground truth corners N_{GT}, the number of detected true corners N_T, the number of falsely detected corners N_F and the detection rate $R_D = \frac{N_T}{N_F+N_{GT}}$ as the overall performance indicator for each algorithm. The comparison results are shown in fig.5.9. It shows that the new detector has overall better corner detection performance than traditional approaches. For the synthetic image, only the new detector gives $R_D = 100\%$. Traditional approaches all miss one corner. But no false corners are reported by SUSAN, two false corners are reported by WANG-BDRAN and three false corners are reported by PLESSEY. For the house image, the number of detected true corners is almost the same for all the corner detectors. The main difference lies on the number of detected false corners. As the new detector gives the smallest N_F among all the algorithms, its overall performance is best. SUSAN also reports quite few false corners, with its overall performance better than WANG-BRADY and PLESSEY. For the lab image, the new detector and PLESSEY have much better overall performances than WANG-BRADY and SUSAN. WANG-BRADY reports too many false corners, while SUSAN misses too many true corners.

Also, speed is an important requirement for real-time jobs such as robot navigation. The new algorithm is based on 1D Hough transform, which determines the low algorithm complexity. To give an explicit explanation, the algorithms were run on several $256 * 256$ size images for hundreds of times on a Pentium III 500 PC and the average running time was shown below. Our algorithm delivers a speed performance of $180ms$ per frame. WANG-BRADY takes about $110ms$. SUSAN has the best speed performance, using about $45ms$. PLESSEY is the slowest one among these algorithms and takes about $538ms$ per frame. Although our algorithm is a little slower than WANG-BRADY and SUSAN, it is 3 times faster than PLESSEY. This experiment shows the ability of implementing our algorithm in real-time applications.

	New Detector	WANG-BRADY	PLESSEY	SUSAN
N_{GT}	61	61	61	61
N_T	61	60	60	60
N_F	0	2	3	0
R_D	100%	95%	94%	98%

(a)

	New Detector	WANG-BRADY	PLESSEY	SUSAN
N_{GT}	69	69	69	69
N_T	64	64	62	60
N_F	4	79	68	26
R_D	87%	43%	45%	63%

(b)

	New Detector	WANG-BRADY	PLESSEY	SUSAN
N_{GT}	172	172	172	172
N_T	157	152	160	100
N_F	111	173	133	66
R_D	55%	44%	52%	42%

(c)

Figure 5.9: Quantitative comparisons of the corner detection algorithms. (a) the synthetic image; (b) the house image, (c) the lab image.

5.7 Conclusion

In this chapter, we proposed a new corner detection algorithm which can compensate some shortcomings of the approach based on local region constraint. A general corner model which can handle all kinds of corners are used and the new algorithm is strictly consistent with this model. The problem of detecting corners is simplified into detecting simple lines in a local coordinate system. The local edges that form the simple lines are selected based on both gradient magnitude information and the gray level analysis. Gradient direction information is used to reduce the effect of noisy points and speed up the algorithm. Hough transform is modified to detect the simple lines, whose parameter space is only one dimension and the algorithm complexity is low enough for real time applications. Both synthesis and real images

are used to evaluate the new corner detector and the results have proven that the performance is quite good.

Chapter 6

Camera Self-Calibration with General Motion

6.1 Introduction

Camera calibration is the process of estimating the projection matrix of a camera. Traditional approaches calibrate a camera based on the scene knowledge, i.e. the relative distances or angles in the scene. It has been shown that the camera can be calibrated if the relative positions of more than five points in general positions are known. A calibration object is always used for this purpose, which is designed so that the $3D$ coordinates of some object points are known *a priori* and the image points corresponding to these object points can be accurately found. Although traditional approaches achieve success in a certain degree, they fail in some scenarios. In the case of working with a pre-recorded image sequence, we always have no means to obtain the calibration information by traditional approaches. Also, the camera might be zoomed out or zoomed in during operations. This will cause significant changes in the camera calibration and the off-line calibration will become invalid. Due to the above reasons, the attentions have been transferred to calibrate a camera without

the scene knowledge, which is known as the camera **self-calibration**. In this case, an unknown camera is always moved around freely and the different views of the same scene are taken. The relationships among these views provide constraints on the camera projection matrices.

This chapter gives a literature review over existing self-calibration technologies assuming that the camera motion between sequential views is general. Section 6.2 introduces some basic computational routines involved in most camera calibration algorithms. Section 6.3 gives a general off-line camera calibration algorithm for comparison. Section 6.4 reviews the projective calibration approaches. Section 6.5 reviews the affine calibration approaches. Section 6.6 reviews the metric calibration approaches. Section 6.7 reviews the Zhang algorithm of calibration which lies between the traditional calibration and self-calibration. Section 6.8 concludes this chapter.

Notations:

$\sim$	equality up to a nonzero scalar
$[]_\times$	vector product in the matrix form
M	a $3D$ scene point with coordinates $(X, Y, Z, W)^T$
m	an image point with coordinates $(u, v, 1)^T$
I	the total number of views
J	the total number of scene points
P	camera projection matrix
K	camera calibration matrix
R	camera rotation matrix
t	camera translation vector
Π_∞	plane at infinity
π_∞	the location of Π_∞ at a projective frame
Π_{REF}	reference plane
Ω_∞	absolute conic
w_∞	the location of Ω_∞ at an affine frame
F	the fundamental matrix
e	the epipole
$H_{\Pi I}$	the homography between a world plane Π and the image plane I
$H_{\infty I}$	the infinity homography
H_{12}^{Π}	the image homography between two views for a world plane Π
H_{12}^{∞}	the infinity homography between two views
T_{PA}	the projective transformation that brings projective calibration to affine
T_{AM}	the affine transformation that brings affine calibration to metric
T_{PM}	the projective transformation that brings projective calibration to metric

6.2 Camera Self-Calibration Analysis

Self-calibration is to recover the camera projection matrix without the scene knowledge. To express the problem mathematically, we can consider a situation that a scene is viewed by I cameras with projection matrices:

$$P_i = K_i[R_i|t_i] \ \text{(i=1,...,I)} \tag{6.1}$$

where K_i is the camera calibration matrix of the i-th camera, R_i and t_i are the rotation matrix and the translation vector of the i-th camera towards the world coordinate system. Any scene point M will be projected to an image point m_i in the i-th view, thus $\{m_1, ..., m_I\}$ are a set of corresponding image points over these I views. The self-calibration problem is to recover P_i from J sets of corresponding image points m_{ij} $(j = 1, ..., J)$.

If the image correspondences are the only available knowledge, the camera parameters provide no constraints on P_i. Thus P_i will be only constrained by the following equation:

$$P_i M_j \sim m_{ij} \quad \forall i, j \tag{6.2}$$

where both P_i and M_j are unknowns. Obviously, if P_i can be computed from the above equation, M_j can be computed also, which means that the self-calibration problem is equivalent to the $3D$ reconstruction problem. Thus we often represent a solution as $\{P_i, M_j\}$.

For any solution $\{P_i, M_j\}$, the following equation will hold for any 4×4 invertible matrix T:

$$m_{ij} \sim P_i M_j \sim P_i T^{-1} T M_j \tag{6.3}$$

Thus $\{P_i T^{-1}, T M_j\}$ is also a solution for equation (6.2). This means that, without additional constraints, the camera projection matrices and the scene structure can

only be determined up to an arbitrary projective transformation from the ground truth calibration. This is called the **projective calibration** of an image sequence and will be represented by $\{P_i^P, M_j^P\}$.

Projective calibration is not sufficient in many situations. For visualization, for example, at least a metric representation of the scene structure is needed. Some additional constraints allow us to upgrade the projective calibration to the **metric calibration** $\{P_i^M, M_i^M\}$, which differs from the ground truth calibration by a metric transformation. This is the highest-level calibration that we can obtain from images where the absolute yard is not available. Without loss of generality, we can assume that the world coordinate system is aligned with the first camera coordinate system, which means $R_1 = I$ and $t_1 = 0$. Thus the expected metric camera projection matrices take the following form:

$$\begin{aligned} P_1^M &\sim K_1[I|0] \\ P_i^M &\sim K_i[R_i|t_i] \ \ (i > 1) \end{aligned} \tag{6.4}$$

where R_i and t_i are the rotation matrix and translation vector of the i-th view with respect with the first camera. For metric calibration, the most traditional assumption is that a same camera takes all the views, without changing focus or zooming. This is equivalent to say that all the cameras have the same calibration matrix $K_i = K$. Thus P_i^M are more constrained in addition to the equation (6.2).

From projective calibration, early approaches immediately try to compute the metric calibration. They all have the problem of coping with too many parameters at one time for nonlinear equations. A better way is the stratified approach. Before computing metric calibration, an affine calibration $\{P_i^A, M_j^A\}$ is computed first, which differs from the ground truth calibration by an affine transformation.

6.3 Basic Routines for Camera Calibration

Before discussing camera self-calibration algorithms in detail, we first introduce some basic computational routines involved in most camera calibration approaches.

6.3.1 Routine 1: Computation of Projection Matrix

The first routine is to compute the projection matrix P from a set of point correspondences M_k (scene points) and m_k (image points) [81] [82] [83] .

As any scene point $M = [X, Y, Z, W]^T$ is related with its image $m = [u, v, 1]^T$ by the equation $m \sim PM$, two linear equations on the unknown elements of P will be available by eliminating the unknown scale factor:

$$\begin{aligned} P_{11}X + P_{12}Y + P_{13}Z + P_{14}W - uP_{31}X - uP_{32}Y - uP_{33}Z - uP_{34}W = 0 \\ P_{21}X + P_{22}Y + P_{23}Z + P_{24}W - vP_{31}X - vP_{32}Y - vP_{33}Z - vP_{34}W = 0 \end{aligned} \tag{6.5}$$

where P_{ij} is the element of P which is located in i-th row and j-th column. For K pairs of points, we can have $2K$ linear equations:

$$Ap = 0 \tag{6.6}$$

where A and p are given by:

$$A = \begin{bmatrix} M_1^T & 0_4^T & -u_1 M_1^T \\ 0_4^T & M_1^T & -v_1 M_1^T \\ ... & ... & ... \\ M_K^T & 0_4^T & -u_k M_K^T \\ 0_4^T & M_K^T & -v_2 M_K^T \end{bmatrix} \tag{6.7}$$

$$p = [P_{11}, P_{12}, P_{13}, P_{14}, P_{21}, P_{22}, P_{23}, P_{24}, P_{31}, P_{32}, P_{33}, P_{34}]^T \tag{6.8}$$

Obviously, K should be no less than 6 to provide a unique solution for p. To avoid the trivial solution $p = 0$, constraints must be imposed on p. Normally, we will enforce $|p|^2 = 1$ as zero vector cannot have a unit norm. Thus equation (6.6) changes to a minimization problem:

$$\min |Ap|^2 \quad \text{subject to} \quad |p|^2 = 1 \tag{6.9}$$

The solution of the above equation is right null-space of A and can be computed using singular value decomposition.

The above approach can give an initial estimation to P and the result can be input into a nonlinear optimization routine to find a more accurate solution. The computed P can re-project any M into an image point $m' = [u', v', 1]^T$. The re-projection error is defined as the distance between m and m'. The sum of re-projection errors over all the points should be minimized and this can be expressed mathematically as:

$$\min(\sum_{k=1}^{K}(u'_k - u_k)^2 + (v'_k - v_k)^2) \tag{6.10}$$

The above nonlinear minimization problem can be solved by Levenbergh-Marquardt minimization routine with respect to the unknowns of P.

6.3.2 Routine 2: Decomposition of Projection Matrix

The second routine is to estimate the camera calibration matrix K and the camera rotation R from the camera projection matrix P [30]. As P, R and t are related by $P = K[R|t]$, we can have the following equation if we just consider the first 3×3 submatrix of P:

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} = KR \tag{6.11}$$

As K is a upper triangular matrix and R is an orthogonal matrix, we can compute K and R from P by QR-decomposition.

6.3.3 Routine 3: Computation of Scene Structure

The last routine is to compute the $3D$ scene points M given two projection matrices P_1 and P_2, and a pair of point correspondence m_1 and m_2 [30].

For the first view, $M = [X, Y, Z, W]^T$ is related with $m_1 = [u_1, v_1, 1]^T$ by the equation $m_1 \sim P_1 M$, two linear equations on the unknowns of M will be available by eliminating the unknown scale factor:

$$\begin{bmatrix} P_{11}^1 - P_{31}^1 u_1 & P_{12}^1 - P_{32}^1 u_1 & P_{13}^1 - P_{33}^1 u_1 & P_{14}^1 - P_{34}^1 u_1 \\ P_{21}^1 - P_{31}^1 v_1 & P_{22}^1 - P_{32}^1 v_1 & P_{23}^1 - P_{33}^1 v_1 & P_{24}^1 - P_{34}^1 v_1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = 0 \qquad (6.12)$$

where P_{ij}^1 is the element of P_1 which is located in i-th row and j-th column. The situation is similar for the second view and we can have four linear equations on the unknowns of M:

$$AM = 0 \qquad (6.13)$$

where A is given by:

$$A = \begin{bmatrix} P_{11}^1 - P_{31}^1 u_1 & P_{12}^1 - P_{32}^1 u_1 & P_{13}^1 - P_{33}^1 u_1 & P_{14}^1 - P_{34}^1 u_1 \\ P_{21}^1 - P_{31}^1 v_1 & P_{22}^1 - P_{32}^1 v_1 & P_{23}^1 - P_{33}^1 v_1 & P_{24}^1 - P_{34}^1 v_1 \\ P_{11}^2 - P_{31}^2 u_2 & P_{12}^2 - P_{32}^2 u_2 & P_{13}^2 - P_{33}^2 u_2 & P_{14}^2 - P_{34}^2 u_2 \\ P_{21}^2 - P_{31}^2 v_2 & P_{22}^2 - P_{32}^2 v_2 & P_{23}^2 - P_{33}^2 v_2 & P_{24}^2 - P_{34}^2 v_2 \end{bmatrix} \qquad (6.14)$$

where P_{ij}^2 is the element of P_2 which is located in i-th row and j-th column. Normally, the constraint $|M|^2 = 1$ is enforced to avoid a trivial zero solution. Thus equation

(6.13) changes to a minimization problem:

$$\min |AM|^2 \quad \text{subject to} \quad |M|^2 = 1 \tag{6.15}$$

The solution of the above equation is right null-space of A and can be computed using singular value decomposition.

The above approach can give an initial estimation to M and the result can be input into a nonlinear optimization routine to find a more accurate solution. The computed M can be re-projected into the first image as $m_1' = [u_1', v_1', 1]^T$ by P_1 and re-projected into the second image as $m_2' = [u_2', v_2', 1]^T$ by P_2. Also the sum of re-projection errors for both two images should be minimized and this can be expressed mathematically as:

$$\min((u_1' - u_1)^2 + (v_1' - v_1)^2 + (u_2' - u_2)^2 + (v_2' - v_2)^2) \tag{6.16}$$

Similarly, this minimization problem can be solved by Levenbergh-Marquardt minimization routine with respect to the unknowns of M.

6.4 Traditional Camera Calibration

Traditional camera calibration corresponds to the situation that the $3D$ coordinates of some scene points and the $2D$ coordinates of the corresponding image points are known. Many approaches exist for the traditional calibration and some well-known approaches can be found in [21] [88] [42] [81] [82] [83] [89]. The difference between the methods mainly lies in the type of used calibration object, i.e. planar or not, or the complexity of the used camera model. For the ideal pinhole camera model, a general algorithm exists and can be fully described by the basic routines:

Algorithm: Traditional Camera Calibration

1. Accurately find the 3D coordinates of the reference points M_k of the calibration object and the 2D coordinates of the corresponding image points m_k.

2. Estimate the projection matrix P using basic routine 1.

3. Decompose the projection matrix into the calibration matrix K and the rotation matrix R using basic routine 2.

Note that if using a single image for calibration, the calibration object is required to be 3D rather than planar. For a planar calibration object, more images are needed to accomplish the calibration task. The calibration accuracy highly depends on the possible errors of the point position measurement in both 3D space and image space.

6.5 Projective Calibration

Projective calibration aims to find a solution $\{P_i^P, M_i^P\}$ which differs from the ground truth calibration by a projective transformation. The only requirement for projective calibration is the point correspondences over the image sequence. Many approaches for projective calibration have been developed in the literature and an extensive review will be given in this part. A more detailed review can be found in [66] [67].

6.5.1 Recovering Projection Matrices for a Stereo Rig

As a unique projective frame can be determined by two views, we will first review several techniques to recover projective projection matrices P_1 and P_2 for a stereo rig.

The Standard Basis

Faugeras [22] sets up a $3D$ projective frame by fixing five non-coplanar scene points as the standard $3D$ projective basis and assuming the first four points project to the standard image basis. For the first camera, we can have:

$$\lambda_i e_i = P_1 E_i \ \text{(i=1,2,3,4)} \tag{6.17}$$

where e_i is a 4-vector which takes the form $e_i = [0...1...0]^T$ (1 is in the i-th position), E_i is a 5-vector which takes the form $E_i = [0...1...0]^T$ (1 is in the i-th position), λ_i are non-zero scalars that account for the use of homogeneous coordinates. This places the following constraints on P_1:

$$P_1 = \begin{bmatrix} \lambda_1 & 0 & 0 & \lambda_4 \\ 0 & \lambda_2 & 0 & \lambda_4 \\ 0 & 0 & \lambda_3 & \lambda_4 \end{bmatrix} \tag{6.18}$$

Similar result can be obtained for the second camera. We then constrain the projections of the fifth $3D$ projective basis $E_5 = [11111]^T$ in each image and this, together with the epipolar geometry between two views, can be used to determine P_1 and P_2 uniquely. A drawback of this approach is that it places entire trust on the correct measurement of the five chosen points used to form the projective basis. Any errors from the localization of these points will lead to significant instability in the estimated projection matrices.

SVD Solution

Hartley *et al.* [33] determines the projective frame by specifying P_1 and P_2 for the two views. The fundamental matrix F of two views can be decomposed into UDV^T by singular value decomposition, where D is the diagonal matrix $D = diag(r, s, 0)$.

Then the two projection matrices can be chosen as following:

$$P_1 = [I_{3\times 3}|0_3] \tag{6.19}$$

$$P_2 = [Udiag(r, s, \gamma)EV^T|U(0, 0, \gamma)^T] \tag{6.20}$$

where γ is any nonzero number that lies between r and s and

$$E = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.21}$$

The detailed proof can be found in [33]. This approach utilizes all the corner correspondences to determine the projective frame, thus the effect of inaccuracies in the localization of the image points can be minimized.

Quasi-Euclidean Calibration

Beardsley *et al.* [7] [6] improves the above approach in two ways. First, he gives a more compact form for P_1 and P_2. Also, he introduces a concept so-called **quasi-Euclidean** calibration which is strictly projective but is close to being Euclidean in the sense that there is only a small distortion from the ground truth Euclidean calibration.

Without loss of generality, the first projection matrix can be set to be $P_1 = [I|0]$ and the second projection matrix can be assumed to be $P_2 = [A|a]$ under a certain projective frame. As the epipole e_2 is the projection of the first optical center $C_1 = [0001]^T$ by P_2, we can have $P_2[0001]^T \sim e_2$ which constrains a as $a \sim e_2$. Also from equation (2.40), the image homography is given by $H_{12}^{\Pi} \sim A - a\pi^T$ for a general plane $\Pi = [\pi^T, 1]^T$ and $H_{12}^{REF} \sim A$ for the reference plane $\Pi_{REF} = [0001]^T$. Thus we can factorize A as $A \sim H_{12}^{REF} \sim H_{12}^{\Pi} + a\pi^T$. Based on the relationship between the fundamental matrix and the image homography, we can have $H_{12}^{\Pi_e} \sim [e_2]_\times F$

(equation (2.43)), where Π_e is the plane determined by C_2 and e_2. Thus we can have $A \sim [e_2]_\times F + a\pi_e^T$. All the above analysis suggests a choice for P_2:

$$P_2 = [H_{12}^{REF}|ce_2] = [[e_2]_\times F + e_2 b^T|ce_2] \tag{6.22}$$

where b is an arbitrary 3-vector and c is a nonzero scalar. P_2 has 4 degrees of freedom, 3 for b and 1 for c. Different choices of b correspond to different choices of projective frame, giving different amounts of projective distortion away from the metric calibration. To minimize the projective distortion, b can be chosen such that $[e_2]_\times F + e_2 b^T$ is as close to an estimated R_2 as possible and this will give a quasi-Euclidean calibration. The scalar c determines the overall scale of the recovered structure and can be chosen arbitrarily.

6.5.2 Recovering Scene Structure for a Stereo Rig

Once the projective frame for the two views is determined, the recovery of projective structure is straightforward in theory. Consider a pair of point correspondence m_1 and m_2, their corresponding $3D$ point M is given by the intersection of two back projected rays: the first ray is given by the optical center C_1 of the first camera and m_1, and the second ray is given by the optical center C_2 of the second camera and m_2. Unfortunately, noises in the measured corner positions perturb these two rays, and this makes them almost certainly not intersect at a point. The computation of M to allow for noise is not a trivial task in the projective frame.

Solution by Intersecting Camera Rays

Although the two corresponding camera rays do not intersect at exact one point, the most straightforward approach is to find the point that simultaneously lies closest to these two rays and such a point is the midpoint of the perpendicular between two rays [30]. But as the structure we are trying to recover is an arbitrary projective

frame, where the distance and perpendicularity are not measurable, we cannot apply this approach directly. However, in [7], Beardsley *et al.* make use of this approach as they argue that their quasi-Euclidean projective frame is very close to the true Euclidean frame.

Suppose $P_i = [A_i|a_i]$ ($i = 1, 2$, A_i and a_i are known at this stage), the projections of M to m_1 and m_2 give the following constraint:

$$m_i \sim P_i M \sim [A_i|a_i]M \tag{6.23}$$

Each back projected ray is defined by two $3D$ points: the optical center C_i and the intersection M_i^∞ of the ray with the plane at infinity Π_∞. The optical center is given by $C_i \sim [a_i^T, 1]^T$ and M_i^∞ is given by $M_i^\infty \sim [A_i^{-1}m_i, 0]^T$. Thus the two rays are given by:

$$M = \begin{bmatrix} a_i \\ 1 \end{bmatrix} + \lambda_i \begin{bmatrix} M_i^\infty \\ 0 \end{bmatrix} \tag{6.24}$$

The midpoint of the perpendicular between these two rays is given by solving the following equation:

$$\sum_i [I - M_i^\infty M_i^{\infty T}]M = \sum_i [a_i] - \sum_i ((a_i^T M_i^\infty)M_i^\infty) \tag{6.25}$$

where the homogeneous vector M_i^∞ is normalized to unit magnitude.

An alternative approach [66] [67] can be forcing the two rays to be coincident at a point in the $3D$ space. Remember that the corresponding epipolar lines of two views are on the same epipolar plane. Given the image point m_1 in the first image, we can compute the epipolar line $l_2 \sim Fm_1$ in the second image and compute m_{2p}, which is the orthogonal projection m_2 onto l_2. Use m_{2p} to obtain the second back projected ray instead of m_2, and it is guaranteed to intersect with the first back projected ray in the projective $3D$ space. The disadvantage of this method is that

all the errors are assumed to be in the second image.

Least-Square Solution through Pseudo-Inverse

This approach parameterizes M as $(X, Y, Z, 1)^T$ [66]. The projection of M to m_1 gives the constraint that $P_1 M = \lambda_1 m_1$. Eliminating λ_1, we can get a pair of equations:

$$\begin{bmatrix} p^1_{11} - p^1_{31}u_1 & p^1_{12} - p^1_{32}u_1 & p^1_{13} - p^1_{33}u_1 \\ p^1_{21} - p^1_{31}v_1 & p^1_{22} - p^1_{32}v_1 & p^1_{23} - p^1_{33}v_1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p^1_{14} - p^1_{34}u_1 \\ p^1_{24} - p^1_{34}v_1 \end{bmatrix} \tag{6.26}$$

where p^1_{ij} is the element of P_1 whose position is i-th row and j-th column. Similarly, the projection of M of m_2 gives the constraint of $P_2 M = \lambda_2 m_2$ and this results in also a pair of equations:

$$\begin{bmatrix} p^2_{11} - p^2_{31}u_2 & p^2_{12} - p^2_{32}u_2 & p^2_{13} - p^2_{33}u_2 \\ p^2_{21} - p^2_{31}v_2 & p^2_{22} - p^2_{32}v_2 & p^2_{23} - p^2_{33}v_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p^2_{14} - p^2_{34}u_2 \\ p^2_{24} - p^2_{34}v_2 \end{bmatrix} \tag{6.27}$$

where p^2_{ij} is the element of P_2 whose position is i-th row and j-th column. The above two pairs of equations form a constraint system:

$$AX_3 = b \tag{6.28}$$

where A is a 4×3 matrix, $X_3 = [X, Y, Z]^T$ and b is a 3-vector. The system is over constrained because there are only three unknowns and can be solved by minimizing $(b - AX_3)^2$, which results in the use of left pseudo-inverse:

$$X_3 = A^+ b \tag{6.29}$$

where $A^+ = (A^T A)^{-1} A^T b$.

This approach parameterizes all the $3D$ points as $(X, Y, Z, 1)^T$, thus it assumes that none the $3D$ points lie on the reference plane and this can be troublesome if the reference plane passes through the data set.

SVD Solution

The projective structure recovery problem can also be solved by the basic computational routine 3 (SVD) [30] and this approach always gives better performance than the pseudo-inverse method due to the parametrization of $3D$ points as $(X, Y, Z, W)^T$ which is more general than the parametrization $(X, Y, Z, 1)^T$ used in the pseudo-inverse approach. Also it is interesting to note that this approach is just a disguised form of the ray intersection method, although the measure being minimized is not Euclidean distance, but something more abstract.

Nonlinear Solution by Minimizing Re-Projection Errors

All the above approaches can give an initial estimation to the projective $3D$ structure and the result can be input into the nonlinear minimization equation described in the basic routine 3 to minimize the re-projection errors and to find a more accurate solution. This kind of minimization is intuitively correct because the image is the only place where we can recover the projective $3D$ structure when no further information is available [7].

6.5.3 Projective Calibration for a Long Image Sequence

If we work with a long image sequence which consists of I $(I > 2)$ views, the first two views can be used to get a unique projective frame $\{P_i, M_j\}$ $(i = 1, 2;\ j = 1, ..., J)$ based on the above approaches. This projective frame is valid for all the views. For the i-th view $(i > 2)$, the point correspondences of m_{ij} and M_j can be used to obtain the projective projection matrix P_i based on the basic routine 1. Then the scene structure M_j can be optionally updated either by Iterated Extended Kalman

Filter (IEKF) [6] or by minimizing the re-projection errors over all the views [30].

6.5.4 Remarks

For an image sequence, the projective calibration gives the projective projection matrices for the cameras as:

$$\begin{aligned} P_1^P &= [I|0] \\ P_i^P &= [H_{1i}^{REF}|e_i] \ \ (i > 1) \end{aligned} \tag{6.30}$$

where H_{1i}^{REF} is given by equation (6.22). The scene structure is recovered up to a projective ambiguity $M_j^P \sim T M_j^M$. Within this specific projective frame, the image homography between the first view and the i-th view for any plane Π is given by:

$$H_{1i}^{\Pi} = H_{1i}^{REF} - e_i \pi^T \tag{6.31}$$

H_{1i}^{Π} can be computed as long as π is available.

6.6 Affine Calibration

Affine calibration aims to find a solution $\{P_i^A, M_i^A\}$ which differs from the ground truth calibration by an affine transformation. In addition to the point correspondences over the image sequence, we assume that the camera calibration matrix is constant, which means $K_i = K$. This part reviews several promising approaches for affine calibration.

6.6.1 From Projective Calibration to Affine

In the projective frame $\{P_i^P, M_j^P\}$, the plane at infinity Π_∞ no longer stays in its canonic position $[0001]^T$ and changes to a general position $[\pi_\infty, 1]^T$. As Π_∞ is an invariant of the affine frame, this suggests an approach to upgrade the projective calibration to affine: find $\Pi_\infty = [\pi_\infty^T, 1]^T$ in the projective frame under consideration, and apply a projective transformation T_{PA} to bring Π_∞ to its canonical position $[0001]^T$:

$$\begin{bmatrix} 0_3 \\ 1 \end{bmatrix} \sim T_{PA}^{-T} \begin{bmatrix} \pi_\infty \\ 1 \end{bmatrix} \quad or \quad T_{PA} \begin{bmatrix} 0_3 \\ 1 \end{bmatrix} \sim \begin{bmatrix} \pi_\infty \\ 1 \end{bmatrix} \tag{6.32}$$

Any transformation taking the following form will satisfy equation (6.32):

$$T_{PA} \sim \begin{bmatrix} A & 0_3 \\ \pi_\infty^T & 1 \end{bmatrix} \quad (\det A \neq 0) \tag{6.33}$$

We will choose the simplest one:

$$T_{PA} = \begin{bmatrix} I_{3\times 3} & 0_3 \\ \pi_\infty^T & 1 \end{bmatrix} \tag{6.34}$$

Then affine calibration can be upgraded from the projective calibration by T_{PA}:

$$\begin{aligned} P_i^A &\sim P_i^P T_{PA}^{-1} \\ M_j^A &\sim T_{PA} M_j^P \end{aligned} \tag{6.35}$$

Vanishing Points

Two parallel lines in the $3D$ Euclidean space intersect at a point at infinity. But in the image plane, they might be no longer parallel and intersect at a finite point, which shall be known as the **vanishing point**. As the vanishing point is the image of a certain point at infinity, the corresponding vanishing points over views provide

constraints on the unknowns of π_∞. Techniques to automatically detect vanishing points in an image can be found in [84] [47].

Consider a stereo rig and assume v_1 and v_2 are a pair of corresponding vanishing points. From equation (6.31), the image homography for Π_∞, which shall be known as **infinity homography** later, is given by $H_{12}^{\infty} \sim H_{12}^{REF} - e_2\pi_\infty^T$, which means:

$$v_2 \sim [H_{12}^{REF} - e_2\pi_\infty^T]v_1 \tag{6.36}$$

This results in one linear equation for the coefficients of π_∞ (from the three equations only two are independent due to the epipolar correspondence of v_1 and v_2 and one is needed to eliminate the unknown scale factor). Thus three pairs of corresponding vanishing points are enough to give a unique solution for π_∞.

Cheirality Invariants

In [29] [34] [35], Hartley uses cheirality to set a bound for all possible values of π_∞. The essence of cheirality is to use information about which points are visible in an image, and hence in front of the camera. This constraint will result in a set of inequalities constraining the coefficients of π_∞. Using the different views available and linear programming with the goal of maximizing the margin by which the inequalities are satisfied, a convex hull of possible solutions in the 3-parameter space can be obtained. The computation of π_∞ can be solved by either a random search [29] [34] or by a dense search [35] over this parameter space. The random search cannot provide an exact solution for π_∞, but points inside the convex hull are approximate solutions for π_∞, which give a quasi-affine calibration. This suffers from a high risk that the accurate solution may not be found. The dense search tries all possible trials in the convex hull and chooses the one which gives the best solution in the latter metric calibration stage as the accurate solution. The cost for this is the increasing computational complexity.

Modulus Constraint

Pollefeys *et al.* makes use of the modulus constraint to solve Π_∞ based on the property of infinity homography [59]. Consider a stereo rig. From the viewpoint of the metric stratum, as $P_1^M \sim K[I|0]$ and $P_2^M \sim K[R_2|t_2]$, the infinity homography is given by $H_{12}^\infty \sim KR_2K^{-1}$. From the viewpoint of the projective stratum, as $P_1^P \sim [I|0]$ and $P_2^P \sim [H_{12}^{REF}|e_2]$, the infinity homography is given by $H_{12}^\infty \sim H_{12}^{REF} - e_2\pi_\infty^T$. As H_{12}^∞ should be consistent at both projective stratum and metric stratum, we can have:

$$H_{12}^\infty \sim H_{12}^{REF} - e_2\pi_\infty^T \sim KR_2K^{-1} \tag{6.37}$$

This equation shows that H_{12}^∞ is conjugated with a rotation matrix which implies that the 3 eigenvalues of H_{12}^∞ must have the same moduli. As the eigenvalues λ of H_{12}^∞ can be solved by the following equation:

$$\det(H_{12}^\infty - \lambda I) = 0 \Rightarrow l_3\lambda^3 + l_2\lambda^2 + l_1\lambda + l_0 = 0 \tag{6.38}$$

where l_i are the coefficients composed of the elements of H_{12}^∞, the following condition is a necessary condition for the roots of equation (6.38) to have equal moduli:

$$l_3 l_1^3 = l_2^3 l_0 \tag{6.39}$$

Then if we express H_{12}^∞ using $H_{12}^{REF} - e_2\pi_\infty^T$, equation (6.39) becomes to a constraint on π_∞, which shall be known as the **modulus constraint**. Thus each additional view besides the first one provides a polynomial equation of degree four in the three unknowns of π_∞, at least four views are required to give a solution. In [56] [58], the modulus constraint was further improved and an affine calibration can be obtained from three views. This is achieved by considering the infinity homography between any two views H_{ij}^∞ instead of H_{1i}^∞ only.

6.6.2 Remarks

For an image sequence, the affine calibration results in a unique affine frame $\{P_i^A, M_j^A\}$, both of which differ from their ground truth values by an affine transformation. The location of the plane at infinity Π_∞ will be determined at this stage, thus the infinity homography H_{1i}^∞ can be computed. The affine projection matrices take the form:

$$\begin{aligned} P_1^A &= [I|0] \\ P_i^A &= [H_{1i}^\infty|e_i] \ \ (i>1) \end{aligned} \tag{6.40}$$

Within this specific affine frame, the image homography between the first view and the i-th view for any plane Π is given by:

$$H_{1i}^\Pi = H_{1i}^\infty - e_i\pi^T \tag{6.41}$$

H_{1i}^Π can be computed as long as π is available.

Affine calibration has been reported to be the most difficult step. Most existing algorithms either depend on some scene knowledge (vanishing points) or are highly nonlinear (modulus constraint). In some special cases, the motion of the camera is restricted and this knowledge can often be exploited to design simpler algorithms for affine calibration. This will be discussed in the next chapter.

6.7 Metric Calibration

Metric calibration aims to find a solution $\{P_i^M, M_i^M\}$ with P_i^M satisfy:

$$\begin{aligned} P_1^M &= K[I|0] \\ P_i^M &= K[R_i|t_i] \ \ (i>1) \end{aligned} \tag{6.42}$$

and M_i^M differ from the ground truth scene structure by a metric transformation. This is the highest level of calibration that we can obtain from the image sequence as the absolute yardstick is not available. Some existing approaches for metric calibration will be reviewed in this part.

6.7.1 From Affine Calibration to Metric

In the affine stage, the plane at infinity Π_∞ is located at its canonic position. But the absolute conic Ω_∞ no longer stays in its canonic position $I_{3\times3}$ and changes to a general position w_∞ (Ω_∞ is a $2D$ conic when only Π_∞ is under consideration). As Ω_∞ is an invariant of the metric stratum, this suggests an approach to upgrade the affine calibration to metric: find $\Omega_\infty = w_\infty$ in the affine frame under consideration, and apply an affine transformation T_{AM} to bring Ω_∞ to its canonical position $I_{3\times3}$. As any affine transformation T_{AM} transfers Ω_∞ as $A^{-T}\Omega_\infty A^{-1}$ (refer to chapter 2), where A is the 3×3 sub-matrix of T_{AM}, we can have:

$$A^{-T} w_\infty A^{-1} \sim I \Rightarrow w_\infty \sim A^T A \tag{6.43}$$

This shows that any transformation T_{AM} taking the following form is suitable for metric calibration:

$$T_{AM} \sim \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} \tag{6.44}$$

where A satisfies equation (6.43) and can be computed from w_∞ by Cholesky factorization. The simplest choice for T_{AM} can be:

$$T_{AM} = \begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix} \tag{6.45}$$

Then metric calibration can be upgraded from the affine calibration by T_{AM}:

$$\begin{aligned} P_i^M &\sim P_i^A T_{AM}^{-1} \\ M_j^M &\sim T_{AM} M_j^A \end{aligned} \tag{6.46}$$

Hartley's method

Hartley proposes a metric calibration algorithm in [34]. Although he does not give a geometric interpretation to the used constraint, it is equivalent to the approach based on the absolute conic, which will be shown later.

Consider a stereo rig. Assume the affine calibration has been obtained as $P_1^A \sim [I|0]$ and $P_2^A \sim [H_{12}^\infty|e_2]$, and assume that the desired metric calibration takes the form $P_1^M \sim K[I|0]$ and $P_2^M \sim K[R_2|t_2]$. As T_{AM} upgrades the affine calibration to metric, for the first view, the following is satisfied:

$$[I|0]T_{AM}^{-1} \sim [K|0] \Rightarrow [I|0] \begin{bmatrix} A^{-1} & 0 \\ 0^T & 1 \end{bmatrix} \sim [K|0] \tag{6.47}$$

This gives the constraint that $A \sim K^{-1}$. For the second view, as $H_{12}^\infty \sim K R_2 K^{-1}$, the following can be obtained based on the property of rotational matrix:

$$K^{-1}H_{12}^\infty K \sim R_2 \Rightarrow K^{-1}H_{12}^\infty K(K^{-1}H_{12}^\infty K)^T \sim I \Rightarrow H_{12}^\infty K K^T H_{12}^{\infty T} \sim K K^T \tag{6.48}$$

Set $C \sim KK^T$, C is a symmetric matrix and has 6 degrees of freedom. We can have:

$$H_{12}^\infty C H_{12}^{\infty T} \sim C \tag{6.49}$$

By enforcing $|H_{12}^\infty| = 1$, the above equation results in a set of linear equations in the six independent coefficients of C. When more images are available, we can solve for C up to a constant factor. The camera calibration K can be computed from C

by Cholesky factorization. As $w_\infty \sim A^T A$, $A \sim K^{-1}$ and $C \sim KK^T$, we can have $w_\infty \sim C^{-1}$, which means that the constraint used by Hartley is actually based on the absolute conic.

Pollefeys' Method

Pollefey *et al.* explicitly uses the absolute conic to form the constraint for the metric calibration [57].

Consider the image of the absolute conic in the i-th view. The projection of Ω_∞ can be computed by the homography of its supporting plane Π_∞. At the metric stage, as $\Omega_\infty = I$ and $P_i^M \sim K[R_i|t_i]$ which means $H_{\infty i} = KR_i$, the image of Ω_∞ is given by ${H_{\infty i}}^{-T} I {H_{\infty i}}^{-1} = K^{-T} R_i^{-T} R_i^{-1} K^{-1} = K^{-T} K^{-1}$. This shows that the image of the absolute conic is invariant to the camera pose, and only depends on the camera intrinsic parameters. Furthermore, as we have assumed the constant camera calibration matrix, the image of the absolute conic should be the same on all the views.

Now consider the image of the absolute conic in the first view at the affine stage. As $\Omega_\infty \sim w_\infty$ and $P_1^A = [I|0]$ which means $H_{\infty 1} = I$, the image of Ω_∞ is given by ${H_{\infty 1}}^{-T} w_\infty {H_{\infty 1}}^{-1} = w_\infty$, which gives the following equation:

$$w_\infty \sim K^{-T} K^{-1} \tag{6.50}$$

This equation is important because it immediately relates the camera intrinsic parameters to the position of absolute conic in the affine stage. Also, as the image of the absolute conic is the same for all the views, the image of the absolute conic in the i-th view will also be w_∞. Using the infinity homography H_{1i}^∞ between the first view and the i-th view, we can have $w_\infty \sim {H_{1i}^\infty}^{-T} w_\infty {H_{1i}^\infty}^{-1}$ or

$$w_\infty \sim {H_{1i}^\infty}^T w_\infty H_{1i}^\infty \tag{6.51}$$

This equation gives constraint on the unknowns of w_∞ and is the same with the constraint given by Hartley.

6.7.2 Remarks

From projective calibration, it is also possible to upgrade it to the metric calibration directly. This can be described by a transformation T_{PM} which combines equations (6.34) and (6.45):

$$T_{PM} = T_{AM}T_{PA} = \begin{bmatrix} A & 0 \\ \pi_\infty^T & 1 \end{bmatrix} \tag{6.52}$$

Then metric calibration can be upgraded from the projective calibration by T_{PM}:

$$\begin{aligned} P_i^M &\sim P_i^P T_{PM}^{-1} \\ M_j^M &\sim T_{PM} M_j^P \end{aligned} \tag{6.53}$$

The geometric interpretation for this approach is that T_{PM} brings the absolute conic Ω_∞ in the projective frame to its canonic position at once. Notice that Ω_∞ is a $3D$ entity because Π_∞ is also unknown in the projective frame. Some algorithms falling in this catalogue can be found in [54] [46] [38] [80]. The problem of these approaches is that they cope with too many parameters at one time.

6.8 Flexible Camera Calibration

Recently, Zhang has introduced a flexible new technique for camera calibration which lies between the traditional calibration and self-calibration [93]. The proposed technique requires the camera to observe a planar pattern shown at more than two different orientations. In this case, although the scene information is also used, only 2D metric information of the planar pattern is required.

Given an image of the model plane, an homography H can be estimated. Denote H by $[h1, h2, h3]$. They derive two equations:

$$\begin{aligned} h_1^T K^{-T} K^{-1} h_2 &= 0 \\ h_1^T K^{-T} K^{-1} h_1 &= h_2^T K^{-T} K^{-1} h_2 \end{aligned} \tag{6.54}$$

So given one homography, there are two constraints on the camera intrinsic parameters. To effectively solve the equations, they give an analytical solution. Let:

$$B = K^{-T} K^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \tag{6.55}$$

Note B is symmetric, defined a $6D$ vector:

$$b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T \tag{6.56}$$

Let the i-th column vector of H be $h_i = [h_{i1}, h_{i2}, h_{i3}]^T$, we can have the following:

$$h_i^T B h_j = v_{ij}^T b \tag{6.57}$$

with:

$$v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T \tag{6.58}$$

Therefore, the two fundamental constraints (6.54), from a given homography, can be rewritten as 2 homogeneous equations in b:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \tag{6.59}$$

If n images of the model plane are observed, we have:

$$Vb = 0 \tag{6.60}$$

where V is a $2n \times 6$ matrix. If $n \geq 3$, a unique solution b defined up to a scale factor, as well as B, can be solved. Then the camera calibration matrix K can be solved from B by Cholesky factorization.

Compared with classical techniques, this technique is considerably more flexible. Compared with self-calibration, it gains considerable degree of robustness. They have given very good experiment results in their paper and this approach has become very popular now. In the next chapter, we will use this approach to assess our new self-calibration algorithm.

6.9 Conclusion

This chapter gave an extensive review over existing self-calibration technologies. The self-calibration problem is stratified into three strata: projective calibration, affine calibration and metric calibration. On each stage, representative approaches were introduced and analyzed.

Chapter 7

Camera Self-Calibration with Restricted Motion

7.1 Introduction

As analyzed in the last chapter, affine calibration is the most difficult step in the self-calibration system. When the camera undergoes the general motion, existing algorithms either depend on some scene knowledge (vanishing points) or are highly nonlinear (modulus constraint). However the problem can be simplified if a restricted motion is considered. Very simple algorithms exist for the case of pure translation [51] [3] and pure rotation [35]. However, in practice, it is virtually impossible to ensure that the camera motion is a pure translation or a pure rotation, which is too restrictive to be satisfied in most situations. This limits the applications of these algorithms. In this chapter, we consider a new type of restricted camera motion, i.e. small rotation plus general translation, which is much less restrictive than two traditional restricted motions. It will be shown that a linear and efficient algorithm exists for affine calibration by assuming the camera motion between sequential views is small. Extensive simulations and experiments will be given to show

the novelty of the proposed algorithm.

This chapter is organized as follows. In section 7.2, the self-calibration algorithm with pure translation will be reviewed. In section 7.3, the self-calibration algorithm with pure rotation will be reviewed. In section 7.4, the new algorithm with small rotation plus general translation will be proposed. Both simulations and experiments will be given in section 7.5. Section 7.6 concludes this chapter.

Notations:

$\sim$	equality up to a nonzero scalar
M	a $3D$ scene point
m	an image point with coordinates $(u, v, 1)^T$
P	camera projection matrix
K	camera calibration matrix
f_u, f_v	the focal length measured in pixel units along u and v axes respectively
s	the camera skew
p_u, p_v	the coordinates of principal point
ζ	aspect ratio
q_u, q_v	the width and height of a pixel measured in mm
R	camera rotation matrix
t	camera translation vector
Π_{REF}	reference plane
Π_∞	plane at infinity
π_∞	the location of Π_∞ at a projective frame
H_{12}^{REF}	the image homography between two views for Π_{REF}
H_{12}^{∞}	the infinity homography between two views
θ_{HFOV}	the horizontal field of view
θ_{VFOV}	the vertical field of view
θ_{HVD}	the horizontal view displacement
θ_{VVD}	the vertical view displacement

7.2 Pure Translation

For a camera mounted on a robot arm or AGV, it is usually possible to force the camera to undergo the pure translational motion. In this case, a very simple linear algorithm exists for the affine calibration [25] [50] [51]. Consider a stereo rig. Pure translation implies that the rotation matrix of the second view $R_2 = I$. Thus the metric projection matrices of the cameras can be chosen as $P_1^M = K[I|0]$ and $P_2^M = K[I|t_2]$. The infinity homography can be uniquely determined as $H_{12}^{\infty} \sim KK^{-1} = I$ from equation (6.37). Also from equation (6.40), we can directly obtain the affine calibration by specifying P_1^A and P_2^A as:

$$\begin{aligned} P_1^A &\sim [I|0] \\ P_2^A &\sim [I|e_2] \end{aligned} \tag{7.1}$$

As $H_{12}^{\infty} \sim I$, the metric calibration equation (6.49) is therefore trivially satisfied. Consequently, no constraints can be obtained for a metric calibration if only pure translation is available.

But when more general motions are available besides the pure translation, a metric calibration can easily be obtained. In [2] [3], Armstrong proposed a stratified approach for metric calibration with a long image sequence. The first two views are forced to undergo a pure translation, while the other views undergo general motions. The first two views are used to obtain a unique affine frame. Then the affine projection matrices for the other views can be computed towards this frame and they can be used to upgrade the affine calibration to the metric one by equation (6.49).

7.3 Pure Rotation

Another useful restricted motion is pure rotation. Many real-world imaging situations do not permit translations - consider, for instance, cameras mounted on tripods and desk or wall-mounted active heads. In this case the fundamental matrix and the epipole can not be determined since no translation is involved. But a simple algorithm exists which directly goes for the metric calibration [35] [36]. Consider a stereo rig. Pure rotation implies that the translation vector of the second view $t_2 = 0$. Thus the metric projection matrices of the cameras can be chosen as $P_1^M = K[I|0]$ and $P_2^M = K[R_2|0]$. The image homography for any scene plane Π is given by $H_{12} \sim KR_2K^{-1}$ (equation (2.40)), which is independent of Π. Therefore H_{12} describes the transfer for all the points from the first view to the second. Of course it is also valid for the points at infinity and thus $H_{12}^{\infty} = H_{12}$. For each pair of point correspondences m_1 and m_2, H_{12} is constrained by:

$$m_2 \sim H_{12}m_1 \tag{7.2}$$

This gives two linear equations on the unknown entries of H_{12}. When more than four pairs of correspondences are available, H_{12} can be determined uniquely and thus H_{12}^{∞} can be obtained. When more than two views are available, equation (6.49) can be used for a metric calibration.

However, in many systems, the assumption that the motion is a pure rotation about the camera's optic center is only an approximation. For instance, with tripod-mounted cameras, the rotation center is typically a good few centimeters below the optic center of the camera and the pan and tilt axes might not intersect. The matter is further complicated with zoom lenes since the optic center may move by as much as a few centimeters along the optic axis as the lens is zoomed. How such translational misalignments affect the camera calibration result has been studied by

Hayman and Murray [37]. They showed that the approximation of pure rotation is indeed sufficient in such cases. Another study of translational misalignment can be found in [87].

7.4 Small Camera Rotation plus General Translation

The self-calibration algorithms for pure translation and pure rotation are both sound in theory and efficient. But they have limited applications because they can only be used in highly controlled environments as described above. In this section, we consider a new type of restricted camera motion, i.e. small rotation plus general translation. It will be shown that a linear and efficient algorithm exists for affine calibration by assuming that the camera motion between sequential views is small. The assumed restricted motion is much less restrictive than two traditional ones and can be valid in more real applications. Consider a person captures an image sequence of a scene using a hand-held video camera. It is almost impossible for him to force the camera motion to undergo such restricted motion as pure translation or pure rotation. But as the frame rate of normal video camera is above 10 frames per second, the camera rotation between sequential images is very small. The assumption of small camera rotation plus general translation is valid here. We can apply the following algorithm and benefit from its simplicity and efficiency.

7.4.1 Normalized Image Coordinates

Following the recommendation of Hartley [32], we work with the normalized image coordinates. The original image coordinates are translated so that the image origin changes to the image center, and then scaled so that the image coordinates are within

the range of $[-1, 1]$. The process can be described by a $2D$ affine transformation T:

$$m \rightarrow m' \sim Tm \tag{7.3}$$

where T is given by:

$$T \sim \begin{bmatrix} \frac{1}{2c_u} & 0 & -\frac{1}{2} \\ 0 & \frac{1}{2c_v} & -\frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

where c_u and c_v are the coordinates of the image center c. As $m \sim PM = K[R|t]M$, we can have $Tm \sim TK[R|t]M$, which shows that the image normalization process is equivalent to transform the original camera calibration matrix K into a new one $K' = TK$ which is given by:

$$K' = \begin{pmatrix} \frac{f_u}{2c_u} & \frac{s}{2c_u} & \frac{p_u - c_u}{2c_u} \\ 0 & \frac{f_v}{2c_v} & \frac{p_v - c_v}{2c_v} \\ 0 & 0 & 1 \end{pmatrix} \tag{7.5}$$

where f_u, f_v, s, p_u and p_v are the five intrinsic parameters of the camera (refer to chapter 2). We normalize the image coordinates of all the views using a same T, thus the effective camera calibration matrices are still constant.

7.4.2 Affine Constraint

Consider a stereo rig and assume a projective calibration has already been obtained as $P_1^P \sim [I|0]$ and $P_2^P \sim [H_{12}^{REF}|e_2]$. The affine calibration can be started from equation (6.37), which will be rewritten here for its importance:

$$H_{12}^{\infty} \sim H_{12}^{REF} - e_2 \pi_{\infty}^T \sim K R_2 K^{-1} \tag{7.6}$$

where K is the effective camera calibration matrix given by equation (7.5). If we use $diag(\cdot)$ to represent the vector which consists of the elements of the main diagonal of a matrix, applying $diag(\cdot)$ to both sides of the above equation will result in:

$$diag(H_{12}^{REF} - e_2\pi_\infty^T) \sim diag(KR_2K^{-1}) \tag{7.7}$$

or equivalently:

$$\begin{bmatrix} H_{11} - h_1\pi_1 \\ H_{22} - h_2\pi_2 \\ H_{33} - h_3\pi_3 \end{bmatrix} \sim \begin{bmatrix} r_{11} + \frac{s}{f_u}r_{21} + \frac{p_u - c_u}{f_u}r_{31} \\ r_{22} + \frac{p_v - c_v}{f_v}r_{32} - \frac{s}{f_u}r_{21} - \frac{s}{f_u}\frac{p_v - c_v}{f_v}r_{32} \\ r_{33} + \frac{s}{f_u}\frac{p_v - c_v}{f_v}r_{32} - \frac{p_u - c_u}{f_u}r_{31} - \frac{p_v - c_v}{f_v}r_{32} \end{bmatrix} \tag{7.8}$$

where $e_2 = [h_1, h_2, h_3]^T$, $\pi_\infty = [\pi_1, \pi_2, \pi_3]^T$, H_{ij} and r_{ij} are the elements of H_{12}^{REF} and R_2 at the i-th row and the j-th column ($i = 1, 2, 3; j = 1, 2, 3$). For simplicity, we set $k_1 = \frac{s}{f_u}$, $k_2 = \frac{p_u - c_u}{f_u}$ and $k_3 = \frac{p_v - c_v}{f_v}$, equation (7.8) can be rewritten as:

$$\begin{bmatrix} H_{11} - h_1\pi_1 \\ H_{22} - h_2\pi_2 \\ H_{33} - h_3\pi_3 \end{bmatrix} \sim \begin{bmatrix} r_{11} + k_1r_{21} + k_2r_{31} \\ r_{22} + k_3r_{32} - k_1r_{21} - k_1k_3r_{32} \\ r_{33} + k_1k_3r_{32} - k_2r_{31} - k_3r_{32} \end{bmatrix} \tag{7.9}$$

Equation (7.9) shall be known as the **affine constraint**.

7.4.3 Linear Algorithm for Affine Calibration

Order of magnitude reasoning will be used in this part. By **order of magnitude** or **order**, we mean a number's nearest power of ten. The order will be rounded from $\sqrt{10}$ and it will be denoted by $o(\cdot)$. As any number $x \neq 0$ can be written as $a \times 10^n$ ($1 \leq |a| < 10$) by scientific notation, if $|a| \leq \sqrt{10}$, $o(x) = 10^n$ and if $|a| > \sqrt{10}$, $o(x) = 10^{n+1}$. For two numbers x and y, assume $o(x) = 10^n$ and $o(y) = 10^m$, in

general the following equation can be used for order analysis:

$$o(xy) = 10^{n+m} \tag{7.10}$$

Furthermore, if $n > m + 1$, the following will give neglecting errors:

$$o(x + y) = 10^{n} \tag{7.11}$$

$$o(x - y) = 10^{n} \tag{7.12}$$

Decompose the rotation matrix R_2 of the second view as the multiplication of three rotation matrices around X, Y and Z axis respectively as following (pitch-roll-yaw convention):

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha) \tag{7.13}$$

where α, β and γ are pitch, roll and yaw measured in radians respectively. If the camera rotation is small enough, we can have $o(\theta) = 10^{-1}$, $\sin\theta \approx \theta$ and $\cos\theta \approx 1$ $(\theta = \alpha, \beta, \gamma)$. Multiplying out the right side of equation (7.13) and neglecting the second and higher-order terms, R_2 can be found as:

$$R_2 = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \tag{7.14}$$

The terms of k_1, k_2 and k_3 in the affine constraint equation (7.9) are totally dependent on the intrinsic parameters of a camera, whose orders follow the below rule for a commonly used camera (the order analysis of k_1 can be found in the appendix 1 and the order analysis of k_2 and k_3 can be found in the appendix 2):

$$o(k_1) = o(k_2) = o(k_3) = 10^{-1} \tag{7.15}$$

By neglecting the second and higher-order terms of the right side of the affine constraint, we can have the following by assuming α, β and γ are small angles:

$$\begin{aligned}
\begin{bmatrix} H_{11} - h_1\pi_1 \\ H_{22} - h_2\pi_2 \\ H_{33} - h_3\pi_3 \end{bmatrix} &\sim \begin{bmatrix} r_{11} + k_1 r_{21} + k_2 r_{31} \\ r_{22} + k_3 r_{32} - k_1 r_{21} - k_1 k_3 r_{32} \\ r_{33} + k_1 k_3 r_{32} - k_2 r_{31} - k_3 r_{32} \end{bmatrix} \\
&\approx \begin{bmatrix} 1 + k_1\gamma - k_2\beta \\ 1 + k_3\alpha - k_1\gamma - k_1 k_3\alpha \\ 1 + k_1 k_3\alpha + k_2\beta - k_3\alpha \end{bmatrix} \\
&\approx \begin{bmatrix} 1 + 10^{-2} - 10^{-2} \\ 1 + 10^{-2} - 10^{-2} - 10^{-3} \\ 1 + 10^{-3} + 10^{-2} - 10^{-2} \end{bmatrix} \\
&\approx \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}
\end{aligned} \tag{7.16}$$

This results in three linear equations:

$$H_{11} - h_1\pi_1 = H_{22} - h_2\pi_2 \tag{7.17}$$

$$H_{11} - h_1\pi_1 = H_{33} - h_3\pi_3 \tag{7.18}$$

$$H_{22} - h_2\pi_2 = H_{33} - h_3\pi_3 \tag{7.19}$$

two of which are independent. Thus each pair of views whose relative motion undergoes the small rotation gives two equations in the three unknowns of π_∞, three views (including the first view) can give a solution to π_∞, or equivalently the affine calibration, which can further be upgraded to the metric one by equation (6.49).

7.5 Experiments

In this section, we will give some experiments on both simulated data and real data to validate the self-calibration algorithm with the motion of small rotation plus general translation. The projective calibration follows the approach of Beardsley [7] (see section 6.5). In the affine stage, we implement the algorithm proposed in the last section, and the solution is used as the start point of the modulus constraint to get a more precise result. Then equation (6.49) is used to obtain the metric calibration and the camera calibration matrix is computed by the projection matrix decomposition routine introduced in the previous chapter. For sake of comparison, we also include the popular flexible camera calibration algorithm of Zhang [93] in our experiments.

7.5.1 Simulated Data

We simulate a situation of moving one camera in a static environment. Provided with the ground truth information, we can evaluate the quality of the calibration results quantitatively. The simulated scene consists of 200 points which are randomly generated in the common field of view of all the cameras. Image sequences ranging from 3 views to 21 views are generated by projecting the scene points onto 500×500 images. All the cameras have the same calibration matrix $K = I_{3\times 3}$. The world coordinate system is aligned with the first camera coordinate system. The positions and orientations of other cameras are all measured relative to the first camera. The accuracy of the calibration is assessed by comparing the computed camera intrinsic parameters with their ground truth values. The relative errors will be given.

Performance w.r.t. the noise level. In the first experiment, we generate a set of image sequences to test the performance of the proposed algorithm against the random noises. To do this, different levels of Gaussian noises (0 mean and σ standard

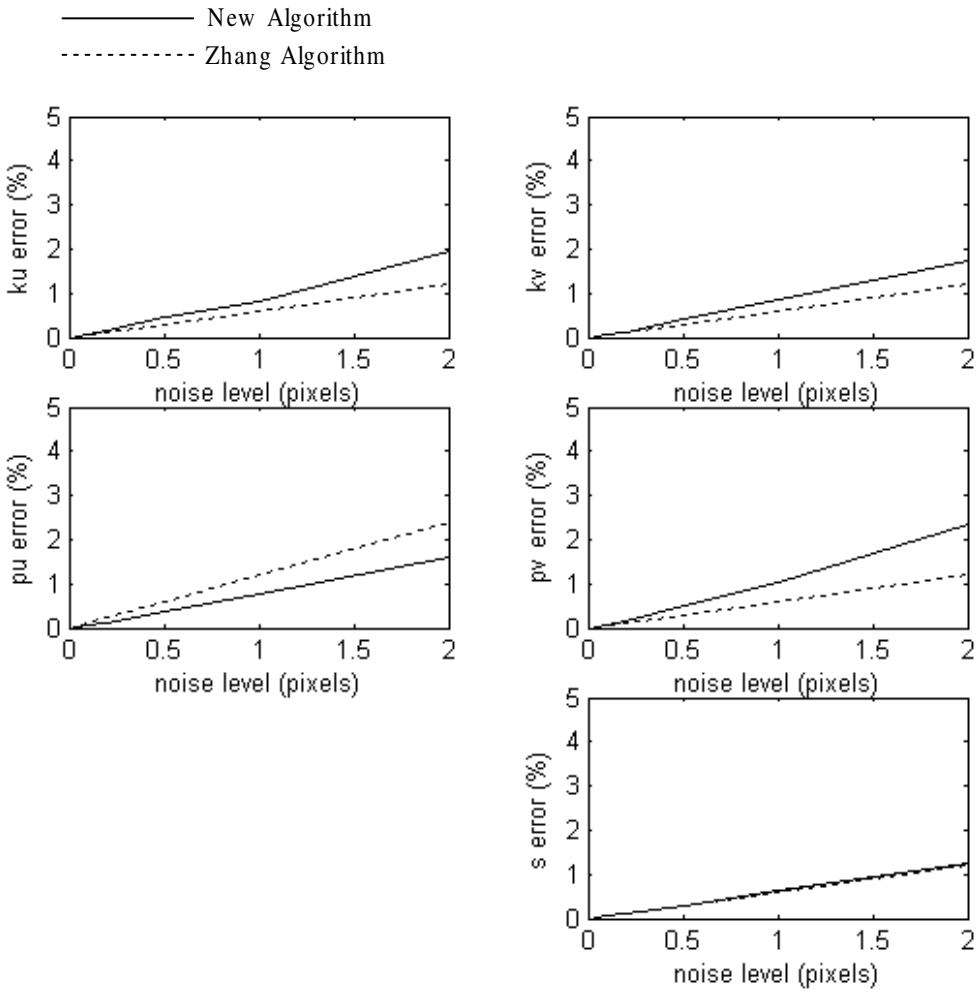


Figure 7.1: The median errors of computed camera parameters relative to the amount of noise added (10 views; camera rotations within 10 degrees).

deviation) are used to disturb the localizations of the image points. For each noise level, we perform 1000 independent trials and the median relative errors of these 1000 calibration results are computed. For this experiment, the image sequence has a fixed length of 10 views. For all the cameras except the first one, their translations are generated randomly within 10 units and their rotations are generated randomly within 10 degrees. The experiment results are shown as fig.7.1 where the noise levels (σ) range from 0 pixel to 2 pixels. The solid lines represent the new algorithm and the dotted lines represent the algorithm of Zhang. The performance of the new

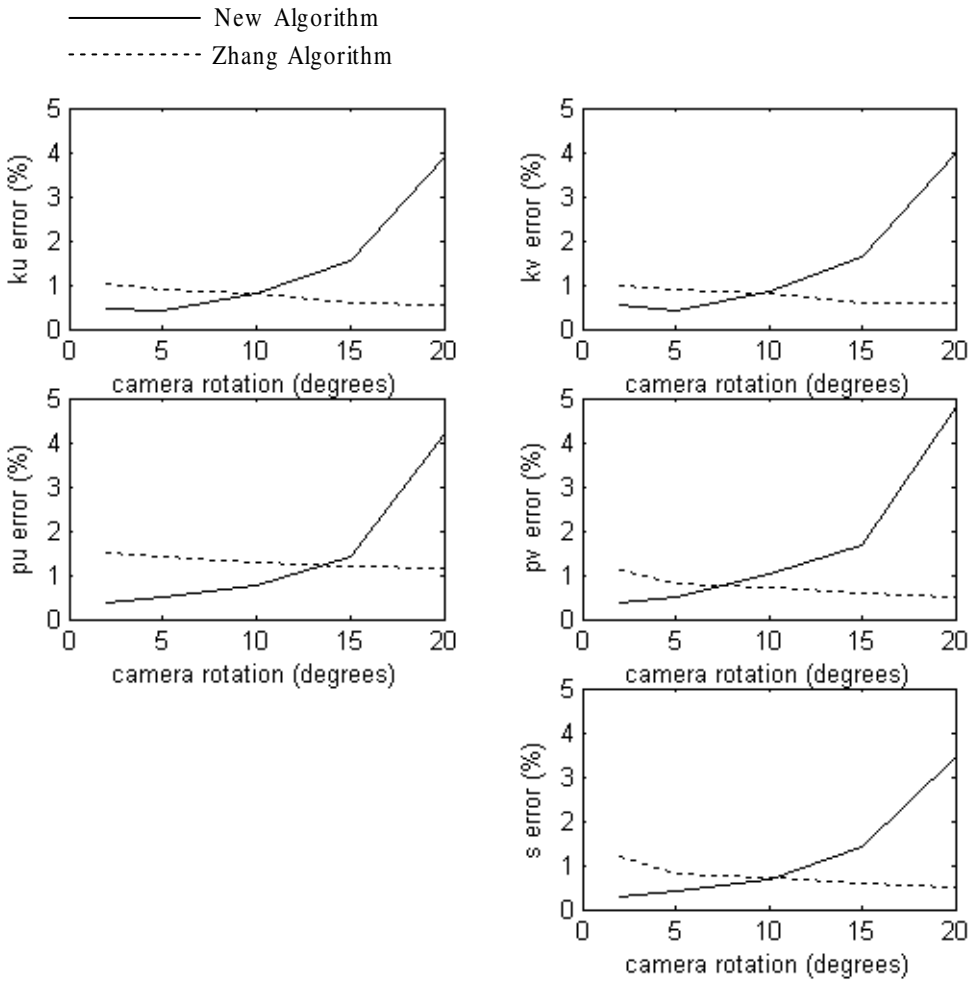


Figure 7.2: The median errors of computed camera parameters relative to the camera rotations (10 views; $\sigma = 1$).

algorithm is pretty good. The relative errors are linear with respect to the noise level. For $\sigma = 0.5$ (which is larger than the normal noise in practical calibration), the errors are less than 0.5%. Although the overall performance of Zhang algorithm is slightly better, it requires the scene information. What's more, the new algorithm gives better estimations on p_u.

Performance w.r.t. the camera rotations. As the basic assumption of our algorithm is that the cameras undergo small rotations, it will be interesting to see how the performance changes when this small rotation increases. To do so, we set a

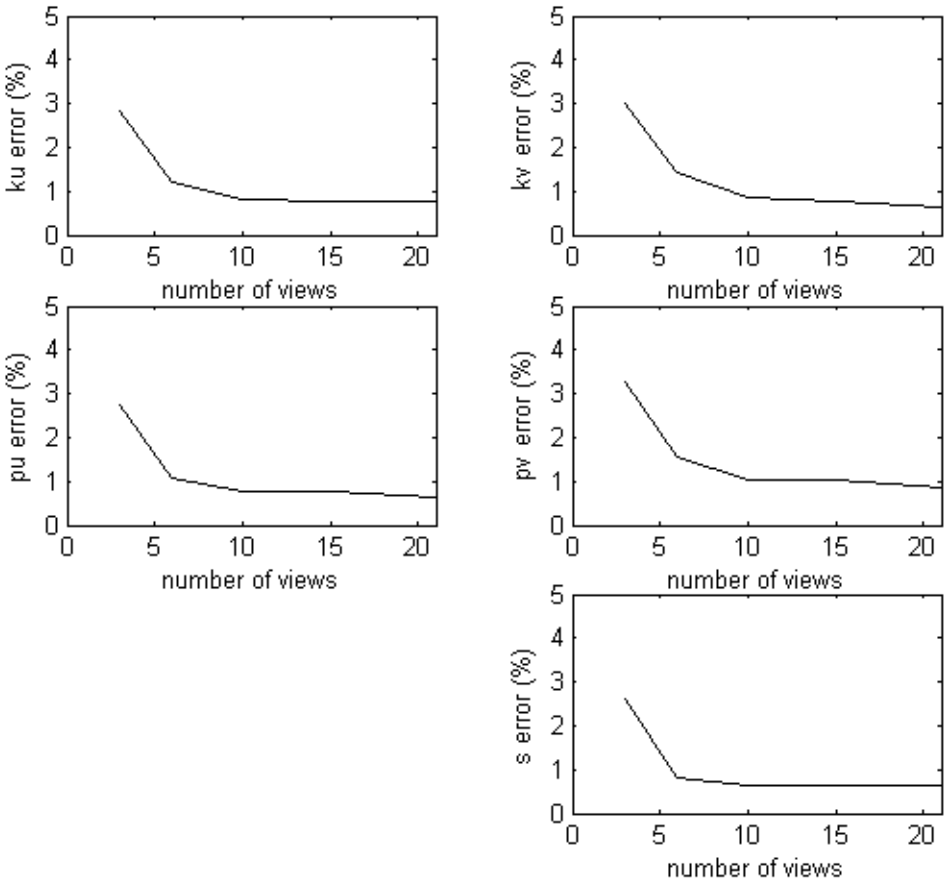


Figure 7.3: The median errors of computed camera parameters relative to the number of views used (Camera rotations within 10 degrees; $\sigma = 1$)

rotation limit θ_{limit} first, then we randomly generate the camera rotations within the range of $[0, \theta_{limit}]$. Different rotation limits are considered, ranging from 3 degrees to 20 degrees. For each θ_{limit}, we perform 1000 independent trials and the median relative errors of these 1000 calibration results are computed. In this experiment, the image sequence has a fixed length of 10 views and the noise level is fixed to be $\sigma = 1$. The experiments results are shown as fig.7.2. The solid lines represent the new algorithm and the dotted lines represent the algorithm of Zhang. We can see that the new algorithm works very well when the camera rotations are less than 5 degrees, with performance better than Zhang algorithm. When the camera

rotations increase to 15 degrees, the errors of the new algorithm increase almost linearly and the performance is comparable with Zhang algorithm. But when the camera rotations increase further to 20 degrees, the errors increase dramatically. This has shown that, to give a good calibration result, the camera rotations should be controlled to be less than 15 degrees for our algorithm.

Performance w.r.t. the number of views. In the last experiment, we generate a set of image sequences to test the performance of the proposed algorithm with respect to the number of views used for calibration. In this case, the noise level is fixed to be $\sigma = 1$. The camera rotations are generated randomly within 10 degrees. Fig.7.3 shows the relative calibration errors with the number of views varying from 3 to 21. The result shows that the more number of views used, the better calibration result we can get. The improvement is significant from 3 views to 5 views.

7.5.2 Real Data 1

The new algorithm is also verified on real data. The environment is to let a camera view a reference object which consists of two perpendicular Tsai calibration grids. A sequence of seven $720 * 480$ images are used in this experiment, which are shown in fig.7.4. The typical motion between sequential views is about $10cm$ for translation, and about 5^o for rotation. Each calibration grid contains 6×8 squares. So there are 384 corners on the reference object.

First, we detect the corners as the intersections of straight lines fitted to each square. These corners are matched mannully. We apply both the Zhang algorithm and the new algorithm to the matched corners. The calibration results are shown in fig.7.5. Instead of showing parameter s, we convert it to the parameter α, which is the angle formed by the image axes. The conversion is done by the equation $\alpha = arccos\frac{s}{f_u}$. For each parameter, two columns are given. The first column gives the absolute value and the second column gives the relative error with respect to the one given by Zhang algorithm. The first two rows show the calibration results of Zhang algorithm and new algorithm respectively using 7 images. These two results are very consistent to each other. The relative errors of the new algorithm are quite small. For f_u, f_v, α and p_u, their relative errors are all below 3%. Although the error of p_v is a little big, the result is still quite satisfying. The third row shows the result of the new algorithm using the first three views. The estimations for f_u, f_v, α and p_u are quite consistent for 3 and 7 images. But the estimation of p_v appears to be quite unstable. This is due to two reasons. Consider a horizontal line and a vertical line of the sample image. There are more corners in the horizontal line (24 corners) than in the vertical line (16 corners). Less corners will give less robust result. Also, the corners of the horizontal line are located in two planes, while the corners of the vertical line are located in a single plane. All these make the estimation of p_u more robust than the estimation of p_v. But as we can see, this instability of p_v estimation

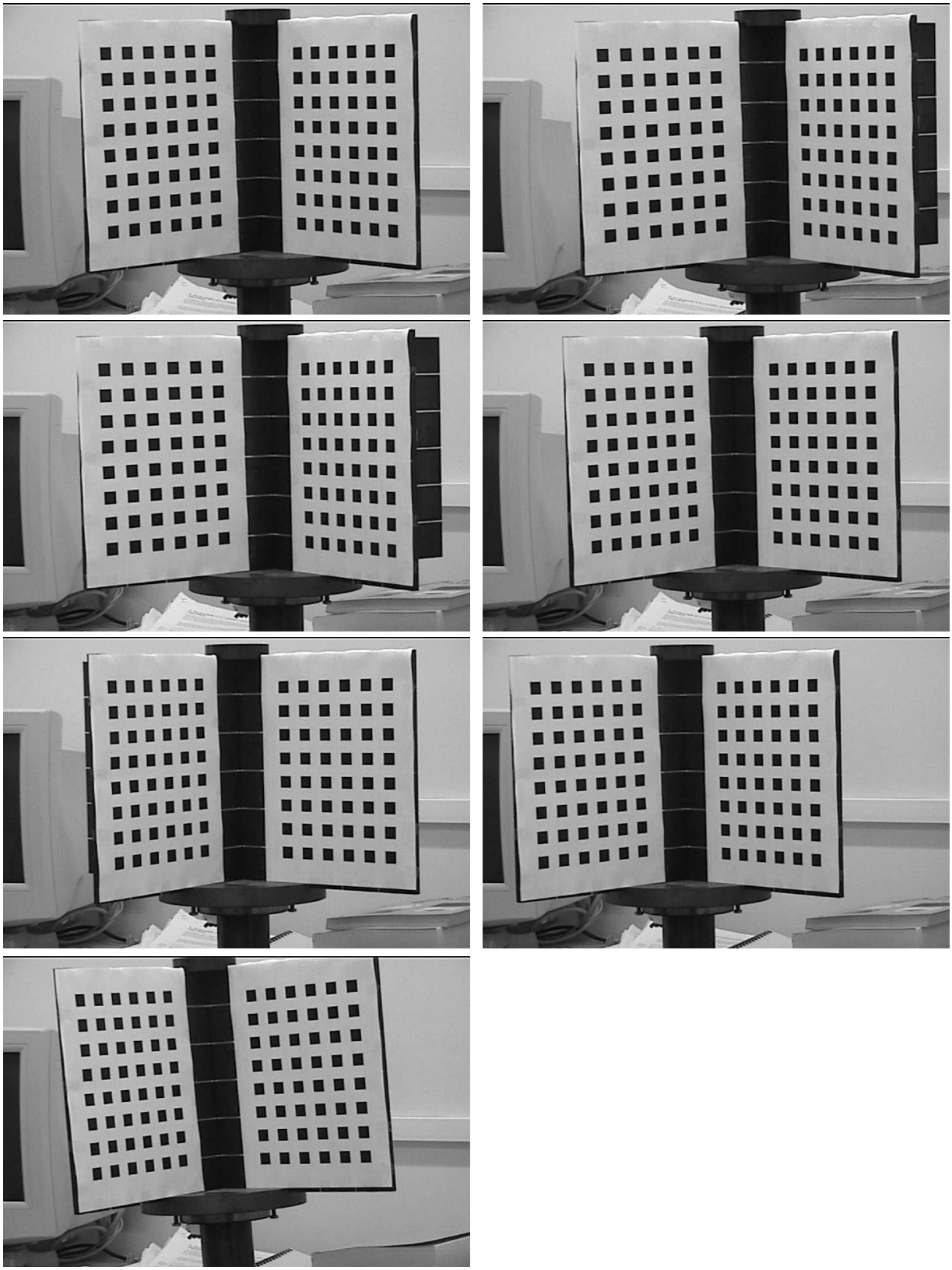


Figure 7.4: An image sequence of calibration grid

	f_u		f_v		angle		p_u		p_v	
	Abosulte value (pixels)	Relative Error (%)	Absolute Value (pixels)	Relative Error (%)	Absolute Value (degrees)	Relative Error (%)	Absolute Value (pixels)	Relative Error (%)	Absolute Value (pixels)	Relative Error (%)
Zhang Algorithm 7 Views	2643.5	----	2375.7	----	90.00	----	435.2	----	248.3	----
New Algorithm 7 Views	2618.5	1.1%	2349.0	1.3%	89.86	0.16%	447.0	2.7%	267.2	7.6%
New Algorithm 3 Views	2539.3	4.1%	2570.9	8.0%	89.81	0.20%	438.9	0.9%	344.2	38.6%

Figure 7.5: Calibration results of Zhang algorithm and the new algorithm for the calibration grid sequence.

can be reduced by using more views.

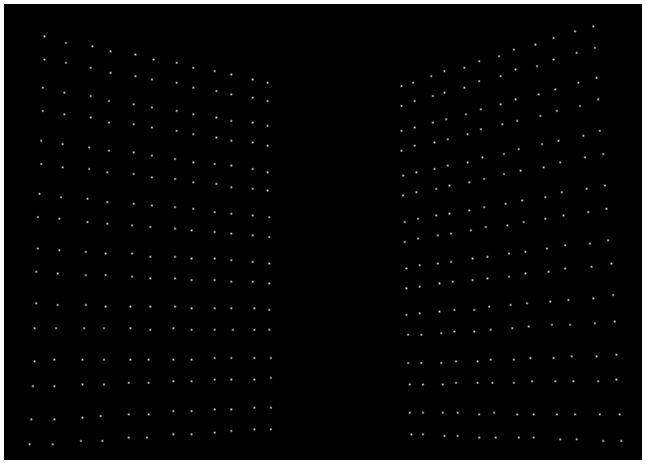
The corner detection method used above can give very good detection and localization performance, but it can only work for structured images such as the calibration grid. For normal images, traditional corner detectors must be used. Different corner detectors will give different sets of matched corners, and thus will give different calibration results. To see this, we also detect the corners using BDRAN and SUSAN respectively. For each approach, we mannully match the corners and use the new algorithm to compute the camera parameters. The results are shown in fig.7.6. The relative errors are measured in the same way as above. The first and second rows correspond to BDRAN and SUSAN respectively. The result based on BDRAN is much better than the one based on SUSAN as the estimation error is much smaller for almost every parameter. This can be understood from two aspects. First, when BDRAN is used, all the 384 corners can be found and matched correctly. But when SUSAN is used, we can only find 342 matched corners because some true corners are missed in certain views. More points will give better calibration result. Second, the calibration result also depends on the corner localization. BDRAN localizes

	f_u		f_v		angle		p_u		p_v	
	Abosulte value (pixels)	Relative Error (%)	Absolute Value (pixels)	Relative Error (%)	Absolute Value (degrees)	Relative Error (%)	Absolute Value (pixels)	Relative Error (%)	Absolute Value (pixels)	Relative Error (%)
New Algorithm BDRAN 7 Views	2587.9	2.1%	2420.8.6	1.9%	89.35	0.72%	450.2	3.4%	226.7	8.7%
New Algorithm SUSAN 7 Views	2784.3	5.3%	2483.4	4.5%	88.79	1.34%	463.1	6.4%	224.7	9.5%

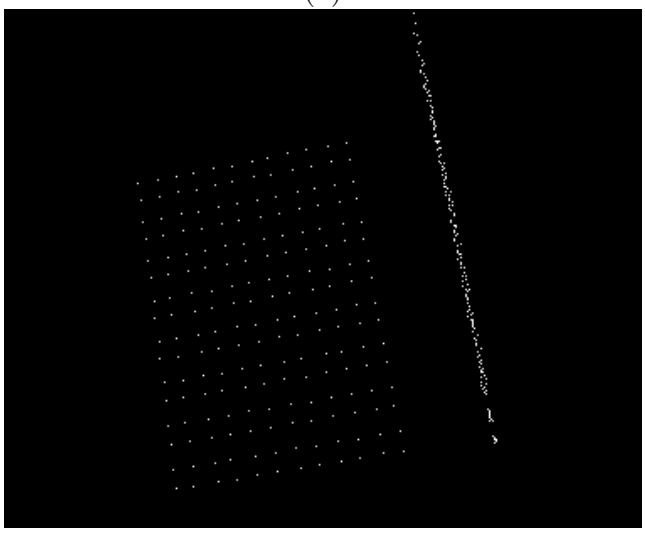
Figure 7.6: Calibration results of the new algorithm for the calibration grid sequence based on different corner detectors.

the corners more accurately than SUSAN. This makes the result based on BDRAN better.

At last, we use the computed calibration by the new algorithm based on BDRAN corner detector to recover the $3D$ metric structure of the calibration object. Fig.7.7.a shows the structure from a general viewpoint. It can be shown that the affine constraint, i.e. parallelism, has been well recovered. Fig.7.7.b shows the structure such that one of its plane is on one edge. It can be shown that the metric constraint, i.e. coplanarity and perpendicularity, have been well recovered. The reconstructed structure has proven the accuracy of the new self-calibration algorithm.



(a)



(b)

Figure 7.7: Recovered metric structure of the calibration object (a) Perspective view from a general position (b) Perspective view with one plane on an edge

7.5.3 Real Data 2

In another experiment, we use a hand-held video camera to capture 10 views of a box scene. The image size is 512×384. The typical motion between sequential views is about $10cm$ for translation, and about 5^o for rotation. The BDRAN corner detector is used to detect the corners. For all the images, the brightness threshold of BDRAN is set to be 25. For each image, we mannully select the ground truth corners and count the number of ground truth corners N_{GT}, the number of detected true

	View 1	View 2	View 3	View 4	View 5	View 6	View 7	View 8	View9	View 10
N_{GT}	82	97	86	104	102	90	78	93	99	82
N_T	67	81	62	86	80	72	60	76	84	68
N_F	16	24	18	32	25	15	18	15	19	12
R_D	68.3%	66.9%	69.0%	59.6%	63.0%	70.6%	68.6%	70.3%	71.2%	72.3%

Figure 7.8: Quantitative analysis of corner detection results on the box sequence

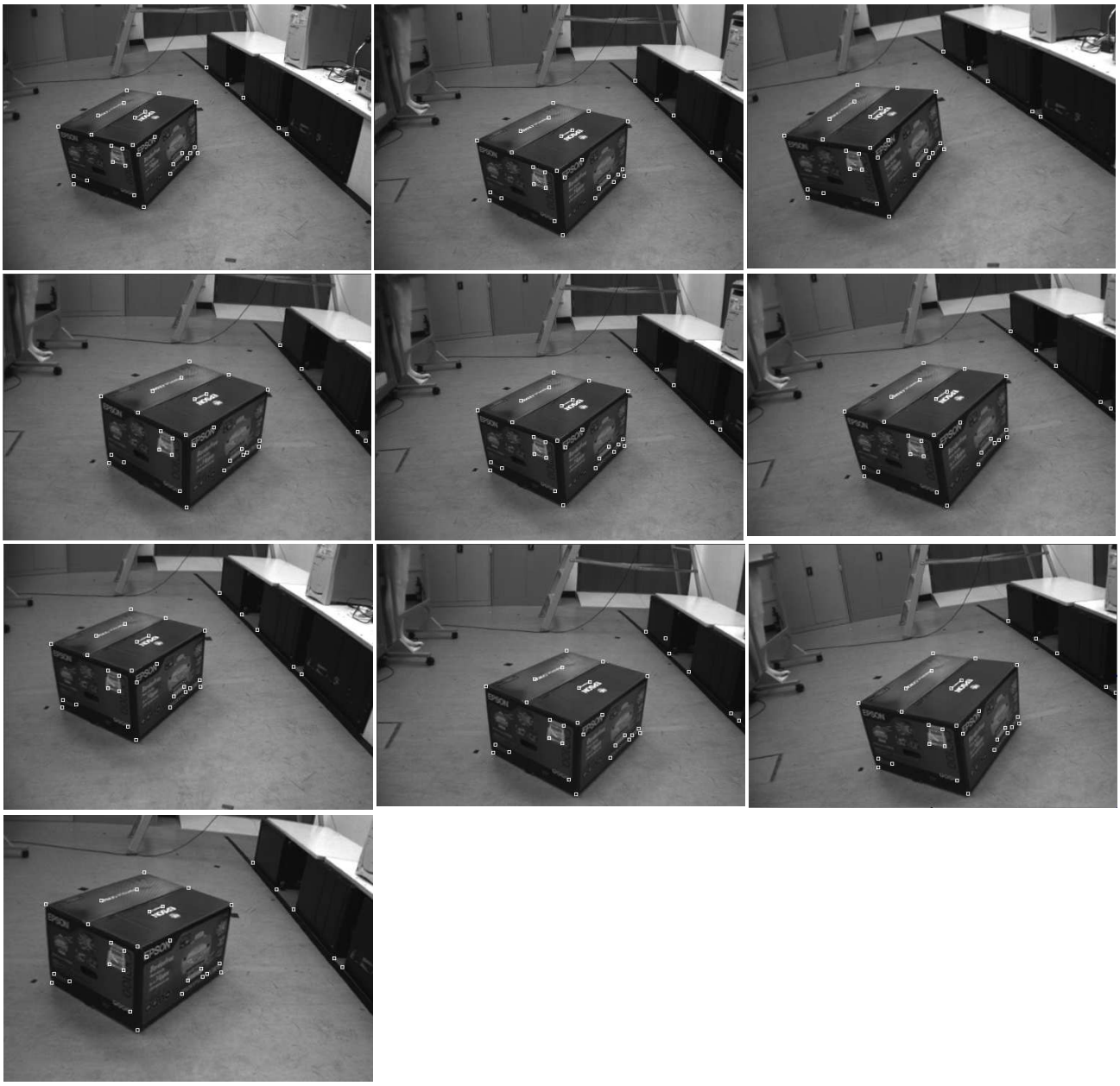
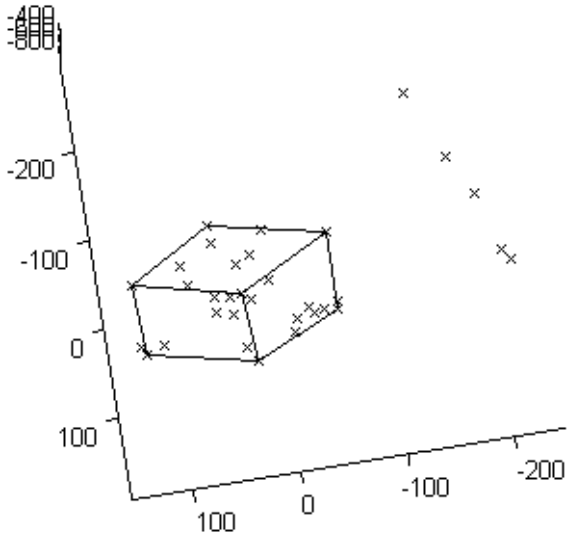


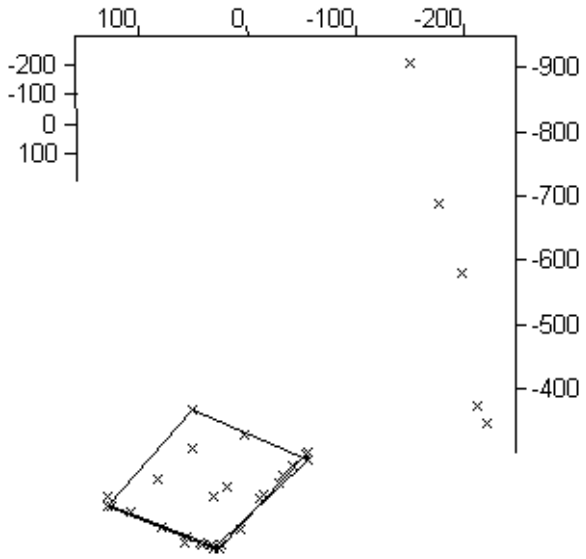
Figure 7.9: The box sequence with matched corners on

f_u	424.5
f_v	402.7
s	6.3
p_u	287.6
p_v	190.5

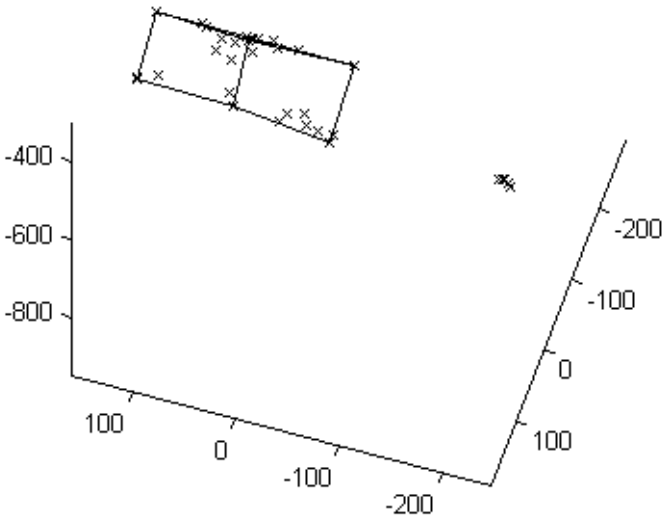
(a)



(b)

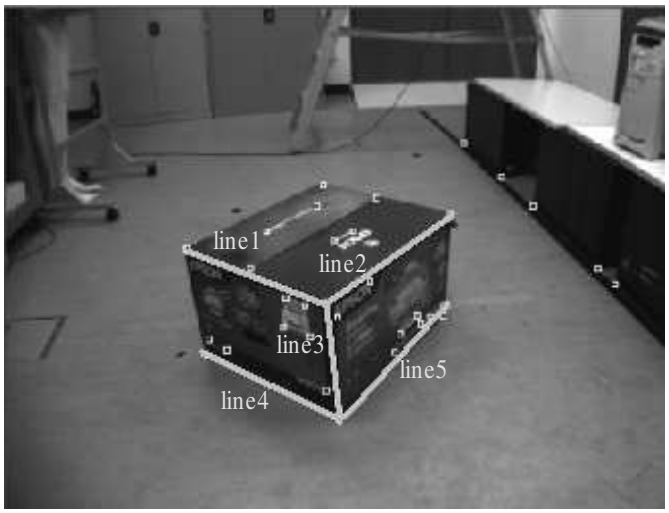


(c)



(d)

Figure 7.10: Calibration result of the new self-calibration algorithm. (a) the computed camera intrinsic parameters; (b) a general view of the recovered scene structure; (c) a top view of the recovered scene structure; (d) a side view of the recovered scene structure.



(a)

	Angle (degrees)	
	Computed	Ground Truth
line1 line2	86.85	90
line1 line3	83.13	90
line2 line3	91.72	90
line1 line4	1.28	0
line2 line5	6.07	0

(b)

Figure 7.11: Quantitative assessment of the recovered scene structure. (a) five lines which are aligned with prominent surface features of the box; (b) the computed angles and their ground truth values.

corners N_T, the number of falsely detected corners N_F and the detection rate R_D. The results are shown in fig.7.8 and it is shown that the performance of BDRAN corner detector is quite stable.

Then we mannully select the matched corners over the whole image sequence and find 33 matched corners in total. Fig.7.9 shows the images with the matched corners on. The new self-calibration algorithm is applied to the matched corners. Fig.7.10.a shows the computed camera intrinsic parameters using our new self-calibration algorithm: $f_u = 424.5, f_v = 402.7, s = 6.3, p_u = 287.6$ and $p_v = 190.5$. Fig.7.10 (b) to (d) show the recovered 3D structure of these corner points. The main corners of the box are connected for clarity. Fig.7.10.b gives a general view, fig.7.10.c gives a top view and fig.7.10.d gives a side view. Parallelism and orthogonality relations clearly have been retrieved. A quantitative assessment of these properties can be made by explicitly measuring angles between selected lines. In fig.7.11.a, we manually select five lines which are aligned with prominent surface features. Lines 1, 2 and 3 are identified for three orthogonal directions. Each line should be perpendicular to each

other and the angle between them should be 90 degrees. Lines 4 and 5 are of the same direction with lines 1 and 2 respectively. The angle between them should be 0 degree. The computed angle and the ground truth value are given as fig.7.11.b. It shows that this is indeed close to the expected values.

7.6 Conclusion

This chapter studied the self-calibration problem with restricted camera motions. First, two traditional algorithms with pure translation and pure rotation were reviewed. Although these algorithms are both sound in theory and efficient, however they can only be used in highly controlled environments. This limits the applications of these algorithms. To solve this problem, we proposed a simple and efficient self-calibration algorithm by considering a new type of restricted camera motion, i.e. small rotation plus general translation, which is much less restrictive than two traditional restricted motions. Experiments on both simulated data and real data were given to validate the new algorithm.

Appendix1: Order Analysis of k_1

From the knowledge of chapter 2 about the camera intrinsic parameters, we know that the image skew is given by $s = f_u \cos\alpha$, where α is the angle formed by u and v axes. Thus k_1 is given by:

$$k_1 = \frac{s}{f_u} = \cos\alpha \tag{7.20}$$

which means k_1 measures the perpendicular level of the image axes. For a common used camera, the image axes are almost perpendicular with each other, with a typical α no more than 5^o. Thus the approximation of the order of k_1 as $o(k_1) = 10^{-1}$ is suitable for most cameras.

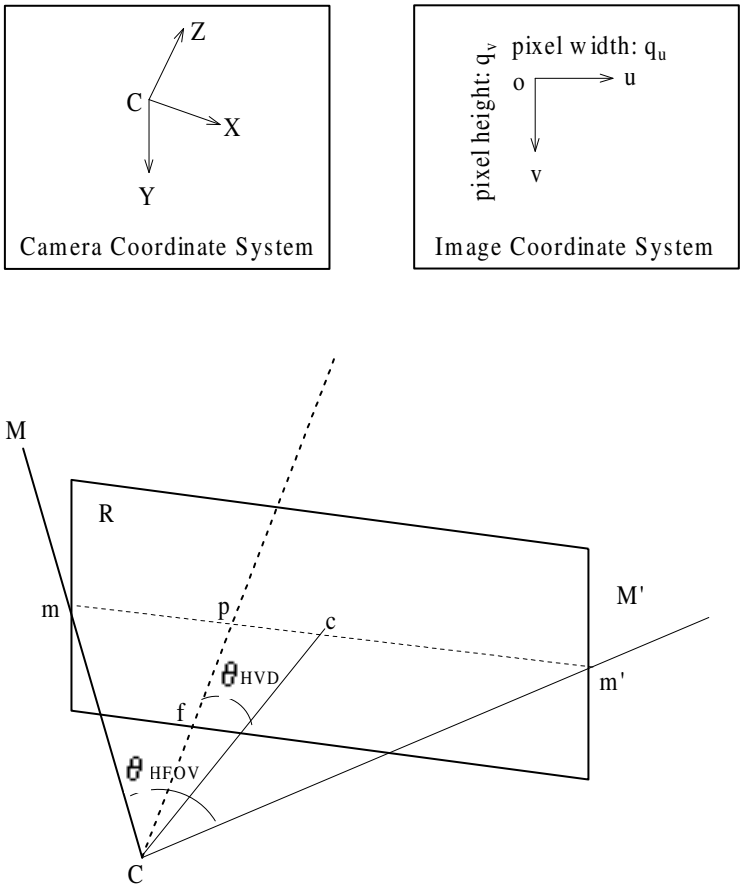


Figure 7.12: Physical explanation of horizontal field of view θ_{HFOV} and horizontal view displacement θ_{HVD}

Appendix2: Order Analysis of k_2 and k_3

k_2 and k_3 are highly related with the concept of the field of view (FOV) of the camera. FOV gives the range of a scene that will be captured by a specific camera and is usually measured in degrees. The FOV in the horizontal (HFOV) and vertical (VFOV) directions will be represented by θ_{HFOV} and θ_{VFOV} respectively. Fig.7.12 explains the concept of θ_{HFOV} with the ideal pinhole camera model. In the figure, C is the optical center and R is the image plane. The XZ plane of the camera coordinate system intersects the boundary of R at m and m'. The HFOV is then defined as:

$$\theta_{HFOV} = \angle mCm' \tag{7.21}$$

The same definition can be given for θ_{VFOV}. For commonly used normal and wide-angle cameras, their FOVs are within the range of 5^o to 50^o.

The principal point $p = [p_u, p_v]^T$ is given by the orthogonal projection from C to the image plane. Cp is the optical axis and its length $|Cp|$ is the focal length f. Use $c = [c_u, c_v]^T$ to represent the midpoint of mm'. The horizontal distance from c to p is given by $p_u - c_u$ measured in pixels, or equivalently $q_u(p_u - c_u)$ measured in millimeters, where q_u is the pixel width measured in millimeters. Thus k_2 is given by $k_2 = \frac{p_u - c_u}{f_u} = \frac{q_u(p_u - c_u)}{f} = \frac{|pc|}{|pC|} = \tan(\angle pCc)$. The angle of $\angle pCc$ will be known as the Horizontal View Displacement (HVD) and represented by θ_{HVD}, thus we can have:

$$k_2 = \tan\theta_{HVD} \tag{7.22}$$

θ_{HVD} of an ideal camera should equal to zero. But for the real cameras, it will not exactly equal to zero, and k_2 actually measures how θ_{HVD} approaches zero. Normally, θ_{HVD} is proportional to θ_{HFOV}. Define κ_H as:

$$\kappa_H = \frac{\theta_{HVD}}{\theta_{HFOV}} \tag{7.23}$$

A typical κ_H is within the range of 0 to 0.1. As $k_2 = \tan(\kappa_H \theta_{HFOV})$, thus for a commonly used camera ($\kappa_H < 0.1$ and $\theta_{HFOV} < 50^o$), we can have the order of k_2 as:

$$o(k_2) = 10^{-1} \tag{7.24}$$

Similarly, a Vertical View Displacement (VVD) θ_{VVD} can be defined and set $\kappa_V = \frac{\theta_{VVD}}{\theta_{VFOV}}$, thus we can have $k_3 = \tan(\kappa_V \theta_{VFOV})$. For a commonly used camera ($\kappa_V < 0.1$ and $\theta_{VFOV} < 50^o$), we can have the order of k_3 as:

$$o(k_3) = 10^{-1} \tag{7.25}$$

Chapter 8

Conclusion and Future Work

8.1 Summary

The work presented in this thesis deals with the camera calibration problem from corner correspondences over an image sequence. First, novel corner detection algorithms were proposed with the aim to improve the accuracy of the overall system. Also, the effort was given to develop a new self-calibration algorithm which achieves better tradeoff between the algorithm complexity and the feasible environment.

The first corner detection algorithm was described in chapter 4. We analyzed the stability performance of SUSAN corner detector and concluded that the instability of SUSAN is due to the violation of USAN constraint by the edges of real images. To solve this problem, the USAN constraint was extended to the general local region constraint. By replacing USAN with a new local region definition namely BDRAN, the corresponding local region constraint holds very well with the real images and this results in a stable corner detection algorithm. The analysis was supported by the experiments on the real images.

The second corner detection algorithm was described in chapter 5 to compensate

some problems of the previous approach when working on X-Junctions. The problem of corner detection was simplified into detecting simple lines in a local coordinate system. As simple lines can be expressed using only one parameter, Hough transform was modified to detect them quickly. Instead of using a general global edge detector, the edges were found locally based on both gradient magnitude threshold and gray level analysis. Gradient direction was used to reduce the effect of noise and speed up the algorithm. Extensive experiments on both synthetic and real images were used to compare this algorithm with some other popular corner detectors.

The novel camera self-calibration algorithm was described in chapter 7. This approach was inspired by the successful self-calibration approaches for traditional restricted motions, i.e. pure translation or pure rotation. We studied a new kind of restricted camera motion, i.e. small rotation plus general translation, which is much less restrictive than two traditional restricted motions, and proposed a simple algorithm. Compared with the algorithms with general motion, the new algorithm is strictly linear and much simpler. Compared with the algorithms with pure translation and pure rotation, the required motion for the new algorithm is much less restrictive and can be easily satisfied at most situations. Experiments on both simulated data and real data were shown to validate the new self-calibration algorithm.

8.2 Future Work

Although the presented work has been shown to perform successfully, several possibilities exist to further enhance the work. Some important directions of further research can be identified as below.

- It will be interesting to give more analysis to the corner detector proposed in the chapter 4. The BDRAN principle consists of two parts: the BRAN principle and the DRAN principle, both can be used independently as a cor-

ner detection algorithm. BRAN reports the bright/dark corners and ignore the dark/bright ones. DRAN reports the dark/bright corners and ignore the bright/dark ones. This means that the corner polarity can be preserved and such property might be helpful in some tracking algorithms.

- The proposed two corner detectors are actually unified in theory. Both of them depend on the local region analysis. The first one depends on the local region area analysis, and the second one depends on the local region edge analysis. Each of them presents their own advantages and disadvantages. The performance might be improved further if these two methods can be combined together.

- For the self-calibration algorithm, we have assumed fixed camera intrinsic parameters. The work can also be extended to allow certain changes in the intrinsic parameters. Certain additional constraints are required. There is a basic equation to describe the relationship of the number of known intrinsic parameters $\#known$ and the number of fixed intrinsic parameters $\#fixed$ [56]:

$$n \times (\#known) + (n-1) \times (\#fixed) > 7 \tag{8.1}$$

where n is the number of images required. So when the camera intrinsic parameters changes, which means $\#fixed = 0$, $\#known$ must not be 0. The most popular assumption is no skew ($s = 0$) and $f_v/f_u = 1$. In this case, $\#known = 2$ and the minimal number of images is 4. Consider two images with focal length f_1 and f_2 respectively (suppose $f_u = f_v$ for each image). Equation (7.16) will change to the following when the camera rotation is small:

$$\begin{bmatrix} H_{11} - h_1\pi_1 \\ H_{22} - h_2\pi_2 \\ H_{33} - h_3\pi_3 \end{bmatrix} \approx \begin{bmatrix} k \\ k \\ 1 \end{bmatrix} \tag{8.2}$$

where $k = f_1/f_2$. The three linear equations (7.17)-(7.19) will become the following two equations:

$$H_{11} - h_1\pi_1 = H_{22} - h_2\pi_2 \tag{8.3}$$

$$H_{11} - h_1\pi_1 = k(H_{33} - h_3\pi_3) \tag{8.4}$$

Thus for every image except the first one, it will introduce one unknown k and provide two linear equations. To have a unique solution, the number of images n (including the first one) needs to satisfy:

$$2(n-1) >= n - 1 + 3 = n + 2 \Rightarrow n >= 4 \tag{8.5}$$

Thus when at least 4 images are available, the affine calibration can be obtained.

- One assumption of our work is that the camera motion between sequential views is small rotation. But during real applications, this condition might be violated. It is interesting and useful to develop an approach to choose the most suitable algorithm based on the current motion that the camera undergoes.

- Another assumption of our work is that the scene is static. It will be interesting to extend the work to the situation where the scene contains moving objects. This might need to combine the work presented here and some tracking algorithms. If more than one moving object exist in the scene, each object might provide independent constraints on the calibration.

Author's Publications

1. F. Shen and H. Wang. Camera self-calibration with restricted motion. *Submitted*, 2004.

2. F. Shen and H. Wang. Corner detection based on local region constraint. *Submitted*, 2004.

3. F. Shen and H. Wang. Corner detection based on modified Hough transform. *Pattern Recognition Letters*, 23(8):1039-1049, 2002

4. F. Shen and H. Wang. A new stratified self-calibration algorithm under small camera rotations. In *Proc. Image and Vision Computing New Zealand*, 2002.

5. F. Shen and H. Wang. A linear algorithm to estimate the plane at infinity. In *Proc. International Conference on Control, Automation, Robotics and Vision*, 2002.

6. F. Shen and H. Wang. Corner detection based on finding edge points locally. In *Proc. International Conference on Computer Vision, Pattern Recognition and Image Processing*, 2002.

7. F. Shen and H. Wang. A local edge detector used for finding corners. In *Proc. International Conference on Information, Communications and Signal Processing*, 2001.

8. F. Shen, and H. Wang. Real time grey level corner detection. In *Proc. International Conference on Control, Automation, Robotics and Vision*, 2000.

Bibliography

[1] Shigeru Ando. Image frild categorization and edge/corner detection from gradient covariance. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 22(2):179–190, 2000.

[2] M. Armstrong. *Self-Calibration from Image Sequences*. PhD thesis, Department of Engineering Science, University of Oxford, 1996.

[3] M. Armstrong, A. Zisserman, and P. Beardsley. Euclidean structure from uncalibrated images. In *Proc. British Machine Vision Conference*, pages 297–316, 1994.

[4] H. Asada and M. Brady. The curvature primal sketch. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8:2–14, 1986.

[5] S.T. Barnard and W.B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, 1980.

[6] P.A Beardsley, A. Zisserman, and D.W. Murray. Navigation using affine structure and motion. In *Proc. European Conference on Computer Vision*, pages 85–96, 1994.

[7] P.A. Beardsley, A. Zisserman, and D.W. Murray. Sequential updating of projective and affine structure from motion. *Internation Journal of Computer Vision*, 23(1):235–259, 1997.

[8] P.R. Beaudet. Rotaional invariant image operators. In *Proc. International Conference on Pattern Recognition*, pages 579–583, 1978.

[9] B. Boufama and R. Mohr. Epipole and fundamental matrix estimation using virtual parallax. In *Proc. International Conference on Computer Vision*, pages 1030–1036, 1995.

[10] Rafael C.Gonzalez and Richard E. Woods. *Digital Image Processing, Second Edition*, chapter 10. Prentice-Hall, 2002.

[11] F. Chabat, G.Z. Yang, and D.M.Hansell. A corner orientation detector. *Image and Vision Computing*, 17:761–769, 1999.

[12] J. Cheng and T. Huang. Image registration by matching relational structures. *Pattern Recognition*, 17(1):149–159, 1984.

[13] C.H. Chou and Y.C. Chen. Moment preserving pattern matching. *Pattern Recognition*, 23(5):461–474, 1990.

[14] Nora Ni Chuiv. *Differential Geometry for Surveying Engineers*. University of New Brunswick, 1985.

[15] R. Deriche and O. Faugeras. Tracking line segments. In *Proc. European Conference on Computer Vision*, pages 259–268, 1990.

[16] R. Deriche and O.D. Faugeras. 2d curve matching using high curvature points: Application to stereo vision. In *Proc. International Conference on Pattern Recognition*, pages 240–240, 1990.

[17] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *International Journal of Computer Vision*, 10(2):101–124, 1993.

[18] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1990.

[19] O. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT press, 1993.

[20] O. Faugeras. Stratification of 3-d vision: projective, affine, and metric representations. *Journal of the Optical Society of America A*, 12(3):465–484, 1995.

[21] O. Faugeras and G. Toscani. Camera calibration for 3d computer vision. In *International Workshop on Machine Vision and Machine Intelligence*, pages 240–247, 1987.

[22] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proc. European Conference on Computer Vision*, pages 563–578, 1992.

[23] M. Fischler and R. Bolles. Random sampling consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[24] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6:35–49, 1993.

[25] L. Van Gool, T. Moons, M. Proesmans, and M. Van Diest. Affine reconstruction from perspective image pairs obtained by a translating camera. In *Proc. International Conference on Pattern Recognition*, pages 290–294, 1994.

[26] A. Goshtasby, S. H. Gage, and J. F. Bartholic. A two-stage cross-correlation approach to template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):374–378, 1984.

[27] M.J. Hannah. A system for digital stereo image matching. *Photogrammetric Engineering and Remote Sensing*, 55:1765–1770, 1989.

[28] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Vision Conference*, pages 189–192, 1988.

[29] R. Hartley. Cheirality invariants. In *Proc. D.A.R.P.A. Image Understanding-Workshop*, pages 743–753, 1993.

[30] R. Hartley. Euclidean reconstruction from uncalibrated views. In *Applications of Invariance in Computer Vision*, pages 237–256, 1994.

[31] R. Hartley. Projective reconstruction from line correspondence. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 903–907, 1994.

[32] R. Hartley. In defence of the 8-point algorithm. In *Proc. International Conference on Computer Vision*, pages 1064–1070, 1995.

[33] R. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–764, 1992.

[34] R. Hartley, E. Hayman, L. de Agapito, and I. Reid. Camera calibration and the search for infinity. In *Proc. International Conference on Computer Vision*, pages 510–517, 1999.

[35] R.I. Hartley. Self-calibration from multiple views with a rotating camera. In *Proc. European Conference on Computer Vision*, pages 471–478, 1994.

[36] R.I. Hartley. Self-calibration of stationary cameras. *International Journal of Computer Vision*, 22(1):5–23, 1997.

[37] E. Hayman and D.W. Murray. The effects of translational misalignment when self-calibration rotating and zooming camears. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 25:1015–1020, 2003.

[38] A. Heyden and K. Astrom. Euclidean reconstruction from constant instrinsic parameters. In *Proc. International Conference on Pattern Recognition*, pages 339–343, 1996.

[39] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1168–1180, 1989.

[40] T.S. Huang and A.N. Netravali. Motion and structure from feature correspondences: A review. In *Proceedings of the IEEE*, volume 252-268, pages 240–240, 1994.

[41] L. Kitchen and A. Rosenfeld. Gray level corner detector. *Pattern Recognition Letters*, 1:95–102, 1982.

[42] R. Lenz and R. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 10:713–720, 1988.

[43] Q.-T. Luong and O. Faugeras. The fundamental matrix: theory, algorithms, and stability analysis. *Internation Journal of Computer Vision*, 17(1):43–76, 1996.

[44] Q.T. Luong, R. Deriche, O.D. Faugeras, and T. Papadopoulo. On determining the fundamental matrix: Analysis of different methods and experimental results. In *Technical Report RR-1894, INRIA*, 1993.

[45] H. Maitre and Y. Wu. Improving dynamic programming to solve image registration. *Pattern Recognition*, 20(4):443–462, 1987.

[46] S. Maybank and O. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8:123–151, 1992.

[47] G. McLean and D. Kotturi. Vanishing point detection by line clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1090–1095, 1995.

[48] R. Mehrotra, S. Nichani, and N. Ranganathan. Corner detection. *Pattern Recognition*, 23:1223–1233, 1990.

[49] F. Mokhtarian and A. Mackworth. Scale-based description and recognition of planar curves and 2d shapes. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(1):34–43, 1986.

[50] T. Moons, L. Van Gool, M. Van Diest, and E. Pauwels. Affine reconstruction from perspective image pairs. In *Applications of Invariance in Computer Vision, Lecture Notes in Computer Science*, pages 297–316, 1994.

[51] T. Moons, L.V. Gool, M. Proesmans, and E. Pauwels. Affine reconstruction from perspective image pairs with a relative object-camera translation in between. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 18(1):77–83, 1996.

[52] H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. International Joint Conference on Artificial Intelligent*, page 584, 1977.

[53] M. Muhlich and R. Mester. The role of total least squares in motion analysis. In *Proc. European Conference on Computer Vision*, pages 305–321, 1998.

[54] Q.-T. Luong O. Faugeras and S. Maybank. Camera self-calibration:theory and experiments. In *Proc. European Conference on Computer Vision*, pages 321–334, 1992.

[55] S.I. Olsen. Epipolar line estimation. In *Proc. European Conference on Computer Vision*, pages 307–311, 1992.

[56] M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, Katholieke Universiteit Leuven, 1999.

[57] M. Pollefeys and L. Van Gool. Self-calibration from the absolute conic on the plane at infinity. In *Proc. Computer Analysis of Images and Patterns*, pages 175–182, 1997.

[58] M. Pollefeys and L. Van Gool. Stratified self-calibration with the modulus constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):707–724, 1999.

[59] M. Pollefeys, L. Van Gool, and A. Oosterlinck. The modulus constraint: A new constraint for self-calibration. In *Proc. International Conference on Pattern Recognition*, pages 349–353, 1996.

[60] W. Press, S. Teukolsky, and W. Vetterling. *Numerical recipes in C: the art of scientific computing*. Cambridge university press, 1992.

[61] Bernd Radig. Image sequence analysis using relational structures. *Pattern Recognition*, 17(17):161–167, 1984.

[62] K. Rangarajan, M. Shah, and D.V. Brackle. Optimal corner detector. *Comouting Vision, Graphics, and Image Processing*, 48:230–245, 1989.

[63] K. Rohr. Modeling and identification of characteristic intensity variations. *Image and Vision Computing*, 10:66–76, 1992.

[64] K. Rohr. Recognizing corners by fitting parametric model. *International Journal of Computer Vision*, 9(3):213–230, 1992.

[65] P.L. Rosin. Augmenting corner descriptors. *Graphical Models and Image Processing*, 58(3):286–294, 1996.

[66] C. Rothwell, G. Csurka, and O. Faugeras. A comparison of projective reconstruction methods for pairs of views. In *Proc. International Conference on Computer Vision*, pages 932–937, 1995.

[67] C. Rothwell, O. Faugeras, and G. Csurka. A comparison of projective reconstruction methods for pairs of views. *Computer Vision and Image Understanding*, 68(1):37–58, 1997.

[68] P. Rousseeuw. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.

[69] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. In *Proc. International Conference on Computer Vision*, pages 230–235, 1998.

[70] C. Schmid and A. Zisserman. Automatic line matching across views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 666–671, 1997.

[71] J.G. Semple and G.T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, 1952.

[72] L.G. Shapiro and R.M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:514–519, 1981.

[73] A. Singh and M. Shneier. Gray level corner detection: a generalization and a robust real time implementation. *Computer Vision, Graph and Image Process*, 51:54–59, 1990.

[74] S.M. Smith and M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[75] C.B. Sun, S.K.In, and D.Y.Choong. Cop: a new corner detector. *Pattern Recognition Letters*, 23:1349–1360, 2002.

[76] M. Taylor. *Visually Guided Grasping*. PhD thesis, University of Oxford, 1995.

[77] P. Torr. *Motion Segmentation and Outlier Detection*. PhD thesis, University of Oxford, 1995.

[78] P. Torr, P. Beardsley, and D. Murray. Robust vision. In *Proc. British Machine Vision Conference*, 1994.

[79] M. Trajkovic and M. Hedley. Fast corner detector. *Image and Vision Computing*, 16:75–87, 1998.

[80] B. Triggs. The absolute quadric. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–614, 1997.

[81] R. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[82] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–331, 1987.

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

[84] T. Tuytelaars, M. Proesmans, and L. Van Gool. The cascaded hough transform as support for grouping and finding vanishing points and lines. In *Proc. International Workshop on Algebraic Frames for Perception-Action Cycle*, pages 278–289, 1997.

[85] H. Wang and M. Brady. A practical solution to corner detection. In *Proc. International Conference on Image Processing*, volume 1, pages 919–923, 1994.

[86] H. Wang and M. Brady. Real-time corner detection algorithm for motion estimation. *Image and Vision Computing*, 13(9):695–703, 1995.

[87] Lei Wang, Sing Bing Kang, Heung-Yeung Shum, and Guangyou Xu. Error analysis of pure rotation-based self-calibration. In *Proc. International Conference on Computer Vision*, pages 464–471, 2001.

[88] J. Weng, N. Ahuja, and T. Huang. Matching two perspective views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):806–825, 1992.

[89] R. Willson. *Modeling and Calibration of Automated Zoom Lenses*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1994.

[90] H. Wang Z. Zheng and E.K. Teoh. Analysis of gray level corner detection. *Pattern Recognition Letters*, 20:149–162, 1999.

[91] Z. Zhang. Motion and structure from two perspective views: From essential parameters to euclidean motion via fundamental matrix. *Journal of the Optical Society of America A*, 14(11):2938C2950, 1997.

[92] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27:161–195, 1998.

[93] Z. Zhang. A flexible new technique for camera calibration. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 22(2):1330–1334, 2000.

[94] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78:87–119, 1995.

[95] Z. Zhang and G. Xu. *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach*. Kluwer Academic Publishers, 1996.

[96] A. Zisserman, D. Forsyth, J. Mundy, C. Rothwell, J. Liu, and N. Pillow. 3d object recognition using invariance. *Artificial Intelligence*, 78:239–287, 1995.

[97] O.A. Zuniga and R. Haralick. Corner detection using the facet model. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 30–37, 1983.