Line Primitives and Their Applications in Geometric Computer Vision

Lilian Zhang

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel
eingereicht im Jahr 2013

Kiel Computer Science Series (KCSS) 2013/1 v1.0 dated 2013-08-15

ISSN 2193-6781 (print version)
ISSN 2194-6639 (electronic version)

Electronic version, updates, errata available via https://www.informatik.uni-kiel.de/kcss

The author can be contacted via http://www.mip.informatik.uni-kiel.de

Published by the Department of Computer Science, Kiel University

Multimedia Information Processing Group

Please cite as:

▻ Lilian Zhang. *Line Primitives and Their Applications in Geometric Computer Vision.* Number 2013/1 in Kiel Computer Science Series. Department of Computer Science, 2013. Dissertation, Faculty of Engineering, Kiel University.

```
@book{lzPostgraduate2013,
  author    = {Lilian Zhang},
  title     = {Line Primitives and Their Applications in Geometric Computer Vision},
  publisher = {Department of Computer Science, CAU Kiel},
  year      = {2013},
  number    = {2013/1},
  series    = {Kiel Computer Science Series},
  note      = {Dissertation, Faculty of Engineering, Kiel University}
}
```

About this Series

The Kiel Computer Science Series (KCSS) covers dissertations, habilitation theses, lecture notes, textbooks, surveys, collections, handbooks, etc. written at the Department of Computer Science at Kiel University. It was initiated in 2011 to support authors in the dissemination of their work in electronic and printed form, without restricting their rights to their work. The series provides a unified appearance and aims at high-quality typography. The KCSS is an open access series; all series titles are electronically available free of charge at the department's website. In addition, authors are encouraged to make printed copies available at a reasonable price, typically with a print-on-demand service.

Please visit `http://www.informatik.uni-kiel.de/kcss` for more information, for instructions how to publish in the KCSS, and for access to all existing publications.

1. Gutachter:	Prof. Dr. Reinhard Koch Christian-Albrechts-Universität zu Kiel
2. Gutachter:	Prof. Dr. Bodo Rosenhahn Leibniz Universität Hannover
3. Gutachter:	Prof. Dr. Jianwei Zhang Universität Hamburg

Datum der mündlichen Prüfung: 8. August 2013

Zusammenfassung

Linien-Primitive, wie sie häufig in strukturierten Szenen anzufinden sind, liefern mehr strukturelle Informationen über die Szene als Punkt-Primitive. Außerdem sind Linien-Primitive eng mit euklidischen Transformationen verwandt, da die Repräsentation dieser als dualer Vektor (auch als Plücker-Koordinaten bekannt) das Gegenstück zum dualen Quaternion darstellt, welches einer euklidischen Transformation entspricht. Diese geometrischen Eigenschaften von Linien-Primitiven sind Motivation für diese Arbeit, in der folgende Beiträge präsentiert werden:

Zunächst wird ein Algorithmus zum Matching von Liniensegmenten vorgestellt, welcher eine Kombination lokaler Merkmale von Linien und geometrischer Eigenschaften von Linienpaaren in Bildern ausnutzt. Hierbei wird einerseits ein neuer Liniendeskriptor zur Darstellung lokaler Merkmale von Linien vorgestellt; weiterhin erzeugt der Algorithmus einen relationalen Graphen, der die paarweise Konsistenz von Linienkorrespondenzen erfasst. Experimente zeigen, dass der vorgestellte Algorithmus robust gegenüber verschiedener Bildtransformationen ist und effizienter arbeitet als herkömmliche graphenbasierte Linien-Matching-Algorithmen.

Außerdem wird in dieser Arbeit, unter Ausnutzung von Symmetrieeigenschaften von Linien, eine komplette Analyse der Lösungen des Perspective-3-Line-Problems (P3L) vorgestellt, welches eine Poseschätzung der Kamera basierend auf drei Referenzlinien und deren 2-dimensionaler Projektionen erlaubt. Im Allgemeinen kann damit für drei Linien ein P3L-Polynom abgeleitet werden, das zur Lösung des Perspective-n-Line-Problems (PnL) genutzt wird. Der hier vorgestellte robuste PnL-Algorithmus ist in der Lage die Kamerapose effizient und präzise zu berechnen, sowohl auf der Basis von wenigen als auch von vielen Linienkorrespondenzen. Weiterhin wird für drei Linien unter speziellen Beschränkungen (z. B. der Manhattan-World-Annahme, welche drei zueinander orthogonale dominante Richtungen vorraussetzt) die Lösung des P3L-Problems zur Schätzung von Fluchtpunkten und zur Klassifizierung von Linien

ausgenutzt. Der vorgestellte Algorithmus zur Schätzung von Fluchtpunkten erreicht eine hohe Genauigkeit und Effizienz unter Ausnutzung der Manhattan-World-Eigenschaften. Ein weiterer Vorteil des hier vorgestellten Systems ist, dass dieses leicht für die Berücksichtigung von Bildern katadioptrischer oder unkalibierter Kameras generalisiert werden kann.

Der dritte Beitrag dieser Arbeit befasst sich mit Structure-from-Motion auf der Basis von Linien-Primitiven. Um die Beschränkungen der Plücker-Repräsentation von Linien zu umgehen, wird hier die Cayley-Repräsentation von Linien vorgestellt, welche auf den geometrischen Eigenschaften von Linien in der Plücker-Repräsentation aufbaut. Zur Erstellung des Modells zur Linienausrichtung werden zwei Ableitungen von Funktionen zur Linienprojektion präsentiert: eine basiert auf der dualen Beziehung zwischen Punkten und Linien; die andere auf der Beziehung zwischen den Plücker-Koordinaten und der Plücker-Matrix. Hierbei werden Struktur- und Bewegungsparameter durch einen inkrementellen Ansatz initialisiert und durch eine modifizierte Bündelblockausgleichung optimiert. Quantitative Auswertungen bestätigen die Verbesserung dieses Ansatzes gegenüber konventionellen Algorithmen zur Linienrekonstruktion.

Abstract

Line primitives are widely found in structured scenes which provide a higher level of structure information about the scenes than point primitives. Furthermore, line primitives in space are closely related to Euclidean transformations, because the dual vector (also known as Plücker coordinates) representation of 3D lines is the counterpart of the dual quaternion which depicts an Euclidean transformation. These geometric properties of line primitives motivate the work in this thesis with the following contributions:

Firstly, by combining local appearances of lines and geometric constraints between line pairs in images, a line segment matching algorithm is developed which constructs a novel line band descriptor to depict the local appearance of a line and builds a relational graph to measure the pair-wise consistency between line correspondences. Experiments show that the matching algorithm is robust to various image transformations and more efficient than conventional graph based line matching algorithms.

Secondly, by investigating the symmetric property of line directions in space, this thesis presents a complete analysis about the solutions of the Perspective-3-Line (P3L) problem which estimates the camera pose from three reference lines in space and their 2D projections. For three spatial lines in general configurations, a P3L polynomial is derived which is employed to develop a solution of the Perspective-n-Line problem. The proposed robust PnL algorithm can efficiently and accurately estimate the camera pose for both small numbers and large numbers of line correspondences. For three spatial lines in special configurations (e. g., in a Manhattan world which consists of three mutually orthogonal dominant directions), the solution of the P3L problem is employed to solve the vanishing point estimation and line classification problem. The proposed vanishing point estimation algorithm achieves high accuracy and efficiency by thoroughly utilizing the Manhattan world characteristic. Another advantage of the proposed framework is that it can be easily generalized to

images taken by central catadioptric cameras or uncalibrated cameras.

The third major contribution of this thesis is about structure-from-motion using line primitives. To circumvent the Plücker constraints on the Plücker coordinates of lines, the Cayley representation of lines is developed which is inspired by the geometric property of the Plücker coordinates of lines. To build the line observation model, two derivations of line projection functions are presented: one is based on the dual relationship between points and lines; and the other is based on the relationship between Plücker coordinates and the Plücker matrix. Then the motion and structure parameters are initialized by an incremental approach and optimized by sparse bundle adjustment. Quantitative validations show the increase in performance when compared to conventional line reconstruction algorithms.

Acknowledgements

First of all, I am most grateful to my supervisor, Prof. Reinhard Koch, not only for his guidance, but especially for his outstanding support – both with difficult academic problems and in daily life. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, the completion of this thesis would not have been possible.

I am very grateful for the humane environment the Multimedia Image Processing group has provided during the years of my study in Kiel. I would like to express my heartfelt gratitude to all the MIPer: Anatol Frick, Andreas Jordt, Anne Jordt, Arne Petersen, Daniel Jung, Falko Kellner, Ingo Schiller, Johannes Brünger, Kristine Haase, Markus Franke, Oliver Fleischmann, Renate Staecker, Robert Wulff, Sandro Esquivel and Torge Storm for both great academic and social interaction.

I have benefited greatly from collaborations with researchers from outside the group. In particular I would like to thank Chi Xu and Huimin Lu for the inspiring discussion. Furthermore, I want to thank Björn Duderstadt for proofreading my thesis.

I also owe a special debt of gratitude to Prof. Xiaoping Hu, Prof. Meiping Wu, Prof. Wenqi Wu and Ass. Prof. Yuanxin Wu, who once offered me valuable courses and advice during my study. I would extend my sincere thanks to my former colleagues at the ING in Changsha and my Chinese friends in Kiel for their support and endless humour.

Lastly, I want to thank my beloved family, for their loving considerations and great confidence in me all through these years. Most particularly, I cannot thank my wife Chengfeng Wu and my son Zhengping Zhang enough for putting up with my absence during the last four years.

Contents

Glossary

BFGS	Broyden-Fletcher-Goldfarb-Shanno method
CM1	Consistency Measure method 1
CM2	Consistency Measure method 2
Iter	Iterative method
LBD	Line Band Descriptor
LM	Levenberg-Marquardt iteration
LP	Line matching leveraged by Point correspondences
LS	Line Signature
LSR	Line Support Region
MLE	Maximum Likelihood Estimation
MSLD	Mean-Standard deviation Line Descriptor
P3L	Perspective-3-Line
P3P	Perspective-3-Point
PnL	Perspective-n-Line
PnP	Perspective-n-Point
R3	3-line RANSAC
R4	4-line RANSAC
RANSAC	Random Sampling Consensus
RPnL	Robust Perspective-n-Line algorithm

RPnP	Robust Perspective-n-Point algorithm
SBA	Sparse Bundle Adjustment
SfM	Structure from Motion
SLAM	Simultaneous Localization And Mapping
SVD	Singular Value Decomposition

... a science must deal with a subject and its properties.

Aristotle

Chapter 1

Introduction

1.1 Motivation

Features, defined as the "interesting" parts of an image, are used as a starting point for many computer vision algorithms. The framework of the overall algorithm and its performance are largely influenced by the adopted feature primitives in the fundamental step. Two most popular forms of features are points (often extracted from corners or the center of regions) and lines (often extracted from edges). A considerable amount of work has been dedicated to point primitives and their applications in computer vision. On the contrary, much less work is directed at line primitives, especially, in a systematic study.

Line primitives and point primitives provide complementary information about the image. In many applications, line primitives are both desirable and indispensable, especially when point primitives are deficient in the scene. Fig.1.1 shows an example of a low-texture image pair in which the point primitives are problematically matched while the corresponding line primitives are well established.

Moreover, line primitives provide a higher level of structure information about the scene than point primitives. Fig.1.2 shows an example of a 3D point map and a 3D line map. It is easier to understand the geometry of the scene from a map of 3D lines than from a map of 3D points. Also, if we have some knowledge about the geometry of the scene, then it is more straightforward in many situations to apply these geometric constraints on the line primitives than on the point primitives. One typical example is the Manhattan world which consists of three mutually orthogonal dominant directions. For scenes under the Manhattan world assumption, it is easy to apply the geometric constraints on the line primitives, i. e., lines

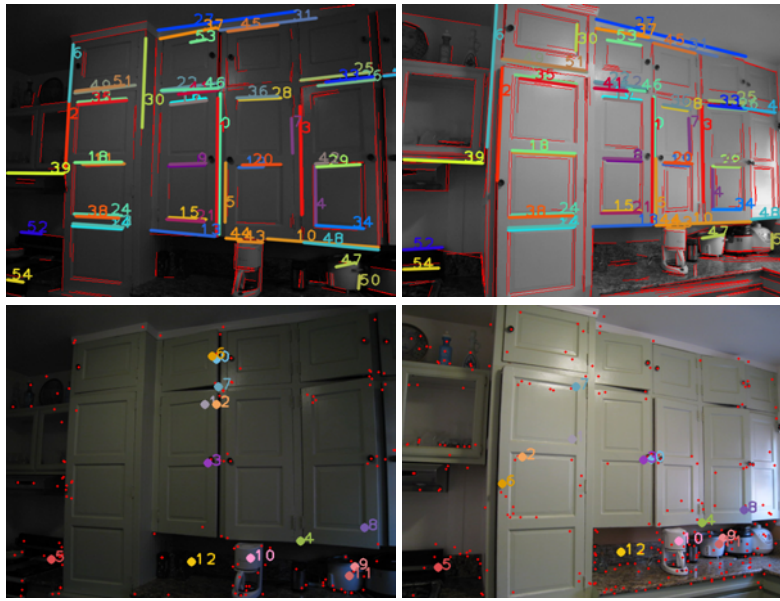


Figure 1.1. An example of a low-texture image pair in which point primitives are deficient. The image pair is from the Line Signature Dataset (`http://www-scf.usc.edu/~luwang/matching_results.html`, accessed on 5th, June, 2011). The point and line matches between two images are marked in the same number with the same color. The unmatched points and lines are shown in red.

should be either parallel or orthogonal to each other. There are no *"direct"* geometric constraints on the point primitives in the Manhattan world. The man-made structured environments (indoor or urban outdoor) are often subject to the Manhattan world assumption and line primitives are often abundant in these scenes. Fig.1.3 shows two example images of an indoor and outdoor Manhattan scene.

Because of the aforementioned reasons, line primitives deserve more attention in the computer vision community. However, to date, the achievements on line primitives are not in parallel to the achievements on point primitives. For example, there are plenty of point-based Structure from

Figure 1.2. An example of the point and line maps. The first picture is a sample image from the Wadham College Dataset (`http://www.robots.ox.ac.uk/~vgg/data/data-mview.html`, accessed on 12th, October, 2012). The second picture shows the reconstructed point cloud of the 3D scene and the third picture shows the reconstructed line structure of the 3D scene.

Motion (SfM) algorithms featuring remarkable real-time performance, while few line-based SfM algorithms can run in real-time. Even for some basic tasks, such as feature matching, point primitives are much better addressed than line primitives. The lower achievement on line primitives is partially due to the intrinsic difficulties when dealing with lines: (i), there is no epipolar constraint on 2D lines in two images; (ii), there is no global linear and minimal parametrization for 3D lines representing their four degrees of freedom by four parameters [HZ04].

This thesis is dedicated to a systematic study about line primitives and their applications in geometric computer vision. More specifically, the geometric properties of lines both in 2D and 3D will be investigated and their related tasks, such as line matching, camera pose estimation from 2D/3D line correspondences, structure from motion based on 2D/2D line correspondences, and vanishing point estimation from lines in a Manhattan world, will be addressed. The accuracy of the algorithms, as well as their time performance, are taken into consideration. The applicability of algorithms in real-time applications is of special interest, because it is the main shortage of existing state-of-the-art algorithms in the discussed field using line primitives.

The first specific problem addressed in this thesis is to match line primitives in the images. The existing geometry-based graph matching algorithms [WNY09] suffer from the high computational cost while the

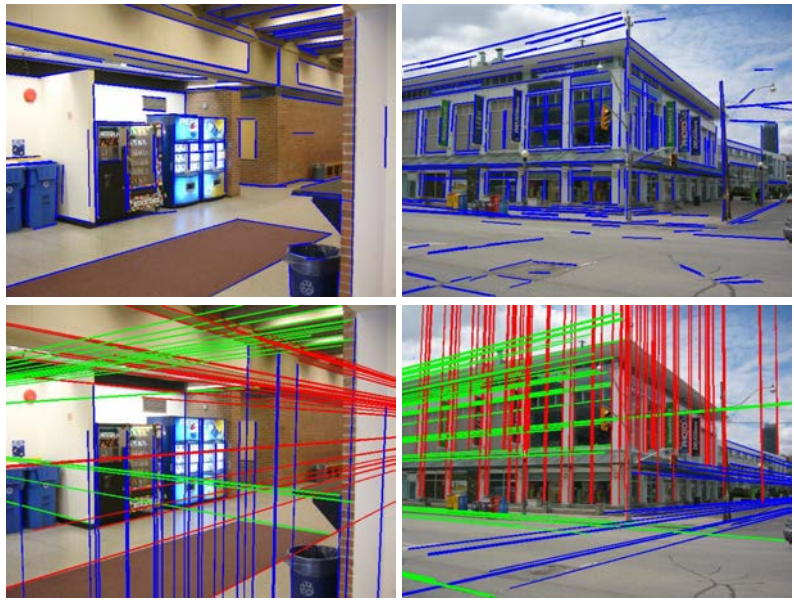


Figure 1.3. Two example images of an indoor and outdoor Manhattan scene. The images are from the York Urban Dataset (`http://www.elderlab.yorku.ca/YorkUrbanDB/`, accessed on 8th, May, 2012). The first row shows the images with extracted lines imposed. The second row shows three orthogonal dominant directions. Lines corresponding to the same Manhattan direction are shown in the same color.

appearance-based descriptor matching algorithms [WWH09] perform poorly for images having large variations. *Can we combine the geometric constraints between lines and their local appearances to develop a line matching algorithm which has better efficiency and is more robust to image variations?* This motivates the first part of our work as presented in Chapter 3.

The second problem addressed in this thesis is to determine the pose of a calibrated camera from n correspondences between 3D reference lines and their 2D projections in the image which is known as Perspective-n-Line (PnL) problem. When $n = 3$, the problem is called Perspective-3-Line (P3L) problem. Their duplicate problems for point primitives are known as

Perspective-n-Point (PnP) problem and Perspective-3-Point (P3P) problem, which are well addressed in the recent work [LMNF09; HR11; LXX12]. In [LX11], a P3P equation system is built with the depth of points in the camera frame as variables. It is reduced to a fourth order polynomial by applying the symmetric property of the P3P problem. Inspired by the success of P3P parametrization, an interesting question arises: *Is it possible to build a fourth order P3L polynomial by finding a symmetric structure in the P3L parameter space?* Our work presented in Chapter 4 is going to answer this question.

For the PnL problem, many issues in this field remain open: (1) the small line sets ($3 < n \leqslant 5$) are sensitive to noise [AD03]; (2) the computational complexity to discover the global optimum is expensive [AD03; MR11b] ; (3) it is hard to find a solution that is both accurate and efficient. In Chapter 4, inspired by the framework of [LXX12] which addresses the PnP problem, we develop a solution of the PnL problem.

Employing the work in Chapter 3 and Chapter 4, we address the problem of SfM based on line correspondences among images in Chapter 5. The approach ranges from the representation of 3D lines, their projections and the initialization procedure to the final Sparse Bundle Adjustment (SBA). In [SVCCM12], it is stated that if the 3D lines are parametrized by the Plücker coordinates, then the optimization of 3D line parameters will suffer from the tiresome Plücker constraints. It is natural to ask: *Can we get rid of the Plücker constraints during optimization?* This partially motivates the work in Chapter 5. Another motivation originates from the clarity of the point projection function and the dual relationship between points and lines [HZ04]. Chapter 5 seeks for a novel formulation of the line projection function which is close to the form of the point projection function. Besides, to initialize the bundle adjustment process, the existing algorithms either need user input [TK95] or are sensitive to image noise [BS05]. In Chapter 5, we develop an incremental initialization process to make up for the shortcomings of the existing work.

The SfM algorithms reconstruct the scene structure from images. On the contrary, *if we have some knowledge about the scene, how can we employ the information to guide image processing?* This motivates our work in Chapter 6 which addresses the problem of estimating vanishing points from lines in a Manhattan world. This problem has been addressed for

more than a decade [CY99]. Surprisingly, the existing work seldom well utilizes the special characteristic of the Manhattan world that lines should be orthogonal or parallel to each other. Most of the existing work first estimates vanishing points of lines in general configurations, then apply the Manhattan world assumption to choose the best solution of vanishing points [Tar09; MR11b]. We try to develop a framework to better take advantage of the Manhattan world characteristic.

1.2 Contributions of this thesis

The research described in this thesis contributes to the field of line primitives and their applications in geometric computer vision. The specific contributions made in each of the four technical chapters are the following:
Chapter 3:

- To characterize the local appearance of a line segment in the image, the Line Band Descriptor (LBD) is proposed, which is efficient to compute.
- The local appearances of line segments and the geometric constraints between line pairs are combined to build a relational graph which reduces the dimension of the graph matching problem.
- A multi-scale strategy is employed to improve the robustness of the line matching algorithm against image variations. Besides, a spectral method is employed to solve the graph matching problem, which improves the efficiency of the line matching algorithm.

Large parts of this work have been pre-published in the following papers:

1. L. Zhang, R. Koch, Line matching using appearance similarities and geometric constraints, in: DAGM-OAGM2012, LNCS 7476, pp. 236-245, 2012 [ZK12a].
2. L. Zhang, R. Koch, An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency, JVCI 24(2013) 794-805 [ZK13].

Chapter 4:

- A general eighth order P3L polynomial is derived. Then the solution of the P3L problem is analyzed by investigating the symmetric structure of the problem. The configurations of three lines in space are discussed. By parameterizing the line directions in the camera frame and employing the fact that the angles between lines are invariant under the Euclidean transformation, we show that for three lines in special configurations, i. e., their directions are linear dependent or orthogonal, the symmetric structure of line directions can be applied to reduce the order of the P3L polynomial to either fourth or second order (it depends on the specific configuration). A complete scenario of three lines in all configurations is presented which unifies the previous work [LHD88; Cag93; QZ08] in the same framework. Moreover, the analysis reveals that the P3P problem is equal to a special case of the P3L problem: three planar lines form a triangle in space.

- A robust and efficient solution of the PnL problem is proposed. Small line sets can be robustly handled to achieve highly accurate results, and large line sets can be efficiently handled because the complexity of the algorithm is $O(n)$.

This work is conducted by collaborating with Dr. Chi Xu. Parts of this work has been pre-published in the following paper:

3. L. Zhang, C. Xu, K. Lee, R. Koch, Robust and efficient pose estimation from line correspondences, in: ACCV2012, PartIII, LNCS 7726, pp. 217-230, 2013 [ZXLK12].

4. C. Xu, L. Zhang, L. Cheng, R. Koch, Pose estimation from line correspondences: a complete 3D line configuration analysis, Submitted to IJCV, 2013 [XZCK13].

Chapter 5:

- To circumvent the tiresome Plücker constraints on the Plücker coordinates, the Cayley representation of 3D lines is defined which enables the algorithm to update the spatial lines with an unconstrained optimization engine.

- A novel derivation of the line projection function based on the relationship between the Plücker coordinates and the Plücker matrix is presented, which is consistent with the derivation based on the dual relationship between points and lines.
- An incremental initialization approach is proposed to boot the nonlinear optimization procedure and the SBA algorithm is adjusted to line primitives.

Large parts of this work have been pre-published in the following papers:

5. L. Zhang, R. Koch, Hand-held monocular SLAM based on line segments, in: IMVIP, pp. 8-15, 2011 [ZK11].
6. L. Zhang, R. Koch, Structure and motion from line correspondences: representation, projection, initialization and sparse bundle adjustment, Submitted to JVCI, 2012 [ZK12b].

Chapter 6:

- Following the parametrization method of line directions proposed in Chapter 4, the vanishing point estimation problem in a Manhattan world is converted to a P3L problem. The method designed for the perspective camera in Chapter 4 is extended to the central catadioptric camera by employing an unifying camera model. Besides, the multiplicity of vanishing point solutions is analyzed which is consistent with the previous results [Che91; MR11b] while our analysis is more straightforward.
- When the intrinsic parameters of the perspective camera is unknown, the algorithm is adjusted to estimate the vanishing points together with the focal length of the camera by sampling four lines in the Manhattan world.
- To measure the consistency between lines and vanishing points, the method based on the orthogonal constraint and the method based on the collinear constraint are introduced and evaluated. Besides, to refine the results of the RANSAC process, a simple iterative approach (Iter) and the Maximum Likelihood Estimator (MLE) are compared as well.

The applications of the proposed algorithm are demonstrated in the end.

Large parts of this work have been pre-published in the following papers:

7. L. Zhang, R. Koch, Vanishing point estimation and line classification in a Manhattan world, in: ACCV2012, PartII, LNCS 7725, pp. 38-51, 2013 [ZK12c].

8. L. Zhang, H. Lu, R. Koch, Vanishing point estimation and line classification in a Manhattan world with a unifying camera model, Submitted to IJCV, 2013 [ZLK13].

The thesis is structured in the following way: In this first chapter the problems that are the subjects of this thesis have been introduced at a high level. In Chapter 2, basic concepts upon which the thesis is based are briefly presented. The following four technical chapters report the main work of this thesis as enumerated above. The specific related work is discussed in each of the technical chapters that addresses the corresponding problem. Finally, Chapter 7 concludes the thesis by summarizing the key points and suggesting directions for future research. Detailed derivations for certain topics can be found in the appendix.

To the man who only has a hammer, everything he encounters begins to look like a nail.

Abraham Maslow

Chapter 2

Basic concepts

Having introduced the problems, this chapter gives a summary of the notation and mathematical identities used and referred to in this thesis.

2.1 Notation

To improve readability and for clarity of the equations, this thesis follows the notation given in Tab.2.1. In addition, a vector $\boldsymbol{v}$ expressed in the camera frame c is represented as $\boldsymbol{v}^c$. The rotation matrix transforming a vector in the world frame w to the camera frame is represented as $\boldsymbol{R}_w^c$.

Table 2.1. Notation used throughout this thesis

Scalars are Latin or Greek lower-case characters	s, α
Vectors are bold faced lower-case characters	$\boldsymbol{v}$
Matrices are bold faced upper-case characters	$\boldsymbol{M}$
Entries of vectors and matrices	v_i, m_{ij}
The field of real numbers	$\Re$
Functions	$f()$
3D objects(points or lines), are italic upper-case characters	P, L
2D objects(points or lines), are bold faced lower-case characters	$\boldsymbol{p}, \boldsymbol{l}$

2.2 Matrix and decomposition

This section introduces matrices with particular forms that occur throughout the thesis, and also the useful matrix decomposition.

2.2.1 Symmetric and skew-symmetric matrices

In linear algebra, a symmetric matrix is a square matrix that is equal to its transpose. Let $\boldsymbol{M}$ be a symmetric matrix, then $\boldsymbol{M} = \boldsymbol{M}^T$. A skew-symmetric matrix is a square matrix $\boldsymbol{A}$ whose transpose is equal to its negative; that is, it satisfies the condition $\boldsymbol{A} = -\boldsymbol{A}^T$. Of particular interest are 3×3 skew-symmetric matrices. If $\boldsymbol{a} = (a_1, a_2, a_3)^T$ is a 3-vector, then a skew-symmetric matrix can be defined by the vector as follows:

$$[\boldsymbol{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{2.1}$$

Matrix $[\boldsymbol{a}]_\times$ is singular and $\boldsymbol{a}$ is its null-vector (right or left). The cross product of two 3-vectors $\boldsymbol{a} \times \boldsymbol{b}$ is related to skew-symmetric matrices according to

$$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b} = \left(\boldsymbol{a}^T [\boldsymbol{b}]_\times\right)^T. \tag{2.2}$$

2.2.2 Cofactor and adjugate matrices

In linear algebra, the cofactor describes a particular construction that is useful for calculating both the determinant and inverse of square matrices. If $\boldsymbol{M}$ is a square matrix, then the minor of its entry m_{ij}, denoted by $det(\bar{\boldsymbol{M}}_{ij})$, is the determinant of the sub-matrix $\bar{\boldsymbol{M}}_{ij}$ obtained by removing from $\boldsymbol{M}$ its i-th row and j-th column. Let c_{ij} be the ij-th entry of the cofactor matrix $cof(\boldsymbol{M})$, then

$$c_{ij} = (-1)^{i+j} det(\bar{\boldsymbol{M}}_{ij}). \tag{2.3}$$

If $\boldsymbol{M}$ is invertible, then $cof(\boldsymbol{M}) = det(\boldsymbol{M})\boldsymbol{M}^{-T}$ in which $\boldsymbol{M}^{-T}$ is the inverse transpose of $\boldsymbol{M}$. The cofactor matrix is related to the way matrices distribute with respect to the cross product as given in the following lemmas [HZ04].

Lemma 2.1. *For a matrix $\boldsymbol{M} \in \Re^{3\times 3}$, and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \Re^3$,*

$$(\boldsymbol{M}\boldsymbol{a}) \times (\boldsymbol{M}\boldsymbol{b}) = cof(\boldsymbol{M})\,(\boldsymbol{a} \times \boldsymbol{b})\,. \tag{2.4}$$

Lemma 2.2. *For a matrix* $\boldsymbol{M} \in \Re^{3\times 3}$*, and a vector* $\boldsymbol{a} \in \Re^3$*,*

$$\boldsymbol{M}[\boldsymbol{a}]_\times \boldsymbol{M}^T = [cof(\boldsymbol{M})\ \boldsymbol{a}]_\times. \tag{2.5}$$

Especially, if $\boldsymbol{M}$ *is a rotation matrix, then*

$$\boldsymbol{M}[\boldsymbol{a}]_\times \boldsymbol{M}^T = [\boldsymbol{M}\boldsymbol{a}]_\times. \tag{2.6}$$

The adjugate matrix is the transpose of the cofactor matrix: $adj(\boldsymbol{M}) = cof(\boldsymbol{M})^T = det(\boldsymbol{M})\boldsymbol{M}^{-1}$. When $\boldsymbol{M}$ is non-invertible, the relationship $adj(\boldsymbol{M})\boldsymbol{M} = \boldsymbol{M}adj(\boldsymbol{M}) = det(\boldsymbol{M})$ is still valid.

2.2.3 Rotation matrix and its representations

In linear algebra, a rotation matrix is a matrix that is used to perform a rotation in Euclidean space. In this thesis, of particular interest are rotation matrices in 3D. If $\boldsymbol{R}$ is a rotation matrix in $SO(3)$, then

$$\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}, \quad det(\boldsymbol{R}) = 1, \tag{2.7}$$

in which $\boldsymbol{I}$ is an identity matrix. The nine entries in $\boldsymbol{R}$ are subject to six constraints as embedded in Eq.(2.7). Therefore, a rotation matrix has only three degrees of freedom. There are various methods to parametrize a rotation matrix. The methods used in this thesis are introduced in the following. A detailed survey can be found in [Shu93].

• **Angle-axis representation:** Assuming $\boldsymbol{R}$ is a time varying variable, according to the orthogonal constraint in Eq.(2.7), we get the derivate of $\boldsymbol{R}\boldsymbol{R}^T$ as:

$$\dot{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{R}\dot{\boldsymbol{R}}^T = 0 \Rightarrow \dot{\boldsymbol{R}}\boldsymbol{R}^T = -(\dot{\boldsymbol{R}}\boldsymbol{R}^T)^T, \tag{2.8}$$

which means $\dot{\boldsymbol{R}}\boldsymbol{R}^T$ is a skew-symmetric matrix. Let $\boldsymbol{t}$ be the 3-vector to form the skew-symmetric matrix, then we have

$$\dot{\boldsymbol{R}} = [\boldsymbol{t}]_\times \boldsymbol{R}, \quad R_0 = \boldsymbol{I}. \tag{2.9}$$

By solving this dynamic system, we get

$$\boldsymbol{R}(\boldsymbol{t}) = \exp^{[\boldsymbol{t}]_\times}. \tag{2.10}$$

Hence, for a given 3-vector $\boldsymbol{t}$, the matrix $\boldsymbol{R}(\boldsymbol{t}) = \exp^{[\boldsymbol{t}]_\times}$ is a rotation matrix representing a rotation through an angle $\|\boldsymbol{t}\|$ about the axis represented by the vector $\boldsymbol{t}$. This representation of a rotation is called the angle-axis representation [HZ04]. For the skew-symmetric matrix $[\boldsymbol{t}]_\times$, it follows that $[\boldsymbol{t}]_\times^3 = -\|\boldsymbol{t}\|^2[\boldsymbol{t}]_\times = -\|\boldsymbol{t}\|^3[\bar{\boldsymbol{t}}]_\times$, where $\bar{\boldsymbol{t}}$ represents a unit vector in the direction $\boldsymbol{t}$. Then, the rotation matrix $\boldsymbol{R}(\boldsymbol{t})$ can be expanded as:

$$\begin{aligned}\boldsymbol{R}(\boldsymbol{t}) &= \boldsymbol{I} + [\boldsymbol{t}]_\times + [\boldsymbol{t}]_\times^2/2! + [\boldsymbol{t}]_\times^3/3! + [\boldsymbol{t}]_\times^4/4! + \cdots \\ &= \boldsymbol{I} + \|\boldsymbol{t}\|[\bar{\boldsymbol{t}}]_\times + \|\boldsymbol{t}\|^2[\bar{\boldsymbol{t}}]_\times^2/2! - \|\boldsymbol{t}\|^3[\bar{\boldsymbol{t}}]_\times/3! - \|\boldsymbol{t}\|^4[\bar{\boldsymbol{t}}]_\times^2/4! + \cdots \\ &= \boldsymbol{I} + \sin\|\boldsymbol{t}\|\ [\bar{\boldsymbol{t}}]_\times + (1-\cos\|\boldsymbol{t}\|)\ [\bar{\boldsymbol{t}}]_\times^2.\end{aligned} \tag{2.11}$$

Let $\theta = \|\boldsymbol{t}\|$, then Eq.(2.11) is equivalent to the Rodrigues formula for a rotation matrix:

$$\boldsymbol{R}(\theta, \bar{\boldsymbol{t}}) = \boldsymbol{I} + \sin\theta\ [\bar{\boldsymbol{t}}]_\times + (1-\cos\theta)\ [\bar{\boldsymbol{t}}]_\times^2. \tag{2.12}$$

• **Quaternion representation:** Let $\boldsymbol{v} = \sin(\theta/2)\bar{\boldsymbol{t}}$ and $\alpha = \cos(\theta/2)$, then the rotation can also be represented by a unit 4-vector $\boldsymbol{q} = (\boldsymbol{v}, \alpha)^T$, called as unit quaternion. The Rodrigues formula, Eq.(2.12), can be rewritten as [Shu93]:

$$\boldsymbol{R}(\boldsymbol{q}) = (\alpha^2 - \|\boldsymbol{v}\|^2)\boldsymbol{I} + 2\boldsymbol{v}\boldsymbol{v}^T + 2\alpha[\boldsymbol{v}]_\times. \tag{2.13}$$

• **Cayley-Gibbs-Rodrigues representation:** Let $\boldsymbol{s} = \boldsymbol{v}/\alpha = \tan(\theta/2)\bar{\boldsymbol{t}}$. Since $\|\boldsymbol{v}\|^2 + \alpha^2 = 1$, we have $\alpha^2(\|\boldsymbol{s}\|^2 + 1) = 1$. From Eq.(2.13), the rotation matrix parametrized by $\boldsymbol{s}$ is

$$\begin{aligned}\boldsymbol{R}(\boldsymbol{s}) &= \frac{(1-\|\boldsymbol{s}\|^2)\boldsymbol{I} + 2\boldsymbol{s}\boldsymbol{s}^T + 2[\boldsymbol{s}]_\times}{1+\|\boldsymbol{s}\|^2} \\ &= (\boldsymbol{I} - [\boldsymbol{s}]_\times)^{-1}(\boldsymbol{I} + [\boldsymbol{s}]_\times).\end{aligned} \tag{2.14}$$

Eq.(2.14) is generally known as Cayley transform [Shu93; Kra99]. The vector $\boldsymbol{s}$ is called Cayley-Gibbs-Rodrigues vector. According to the definition of $\boldsymbol{s}$, it is obvious that the rotation axis is determined by the vector $\boldsymbol{s}$ and the rotation angle θ equals to $2\arctan\|\boldsymbol{s}\|$. The Cayley-Gibbs-

Rodrigues representation has minimum dimension but the disadvantage that $\|\boldsymbol{s}\| \to \infty$ as $\theta \to \pi$. Thus, rotation through π cannot be represented. Given the rotation matrix $\boldsymbol{R}$, the Cayley-Gibbs-Rodrigues vector $\boldsymbol{s}$ can be easily computed as:

$$[\boldsymbol{s}]_\times = (\boldsymbol{R} - \boldsymbol{I})(\boldsymbol{R} + \boldsymbol{I})^{-1}. \tag{2.15}$$

• **Euler-angle representation:** Another prevalent parametrization of a rotation matrix is the Euler angle representation introduced by Leonhard Euler. In $\Re^3$, the 3-vector $\boldsymbol{t}$ can be parametrized as:

$$\boldsymbol{t} = \bar{\varphi}\boldsymbol{e}_x + \bar{\psi}\boldsymbol{e}_y + \bar{\gamma}\boldsymbol{e}_z = \bar{\varphi}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \bar{\psi}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \bar{\gamma}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{2.16}$$

According to Eq.(2.10), we have

$$\begin{aligned} \boldsymbol{R} &= \exp^{[\bar{\varphi}\boldsymbol{e}_x + \bar{\psi}\boldsymbol{e}_y + \bar{\gamma}\boldsymbol{e}_z]_\times} = \exp^{[\varphi\boldsymbol{e}_x]_\times} \exp^{[\psi\boldsymbol{e}_y]_\times} \exp^{[\gamma\boldsymbol{e}_z]_\times} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{2.17}$$

Hence, any rotation matrix can be decomposed as a product of three elemental rotation matrices (Eq.2.17). Each elemental rotation matrix represents an elemental rotation around a single coordinate axis. Here, $[\bar{\varphi}, \bar{\psi}, \bar{\gamma}]$ are the Lie-Cartan coordinates of the first kind relative to the basis $[\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z]$ and $[\varphi, \psi, \gamma]$ are the Lie-Cartan coordinates of the second kind (i. e., the Euler angles) [FFS10]. The Euler angle representation is also a minimum dimension representation but it might encounter the Gimbal lock problem [TW04].

2.2.4 Singular value decomposition

The Singular Value Decomposition (SVD) is one of the most useful matrix decompositions. Suppose $\boldsymbol{M}$ is an $m \times n$ matrix whose entries come from

the field of real numbers (complex numbers are not of interest in this thesis), then the singular value decomposition of $\boldsymbol{M}$ is as follows:

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{2.18}$$

in which $\boldsymbol{U}$ is an $m \times m$ orthogonal matrix, the matrix $\boldsymbol{D}$ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal, and $\boldsymbol{V}$ is a $n \times n$ orthogonal matrix. It is conventional to write $\boldsymbol{V}^T$ instead of $\boldsymbol{V}$ in this decomposition. Another common convention is to list the singular values in descending order. In this case, the diagonal matrix $\boldsymbol{D}$ is uniquely determined by $\boldsymbol{M}$.

The most common application of SVD decomposition is in the solution of over-determined systems of linear equations. A set of homogeneous linear equations can be written as $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{0}$ in which $\boldsymbol{M}$ is known and a non-zero $\boldsymbol{x}$ is to be determined which satisfies the equation. The least-square solution of $\boldsymbol{x}$ is the column of $\boldsymbol{V}$ corresponding to the smallest singular value, i. e., the last column of $\boldsymbol{V}$.

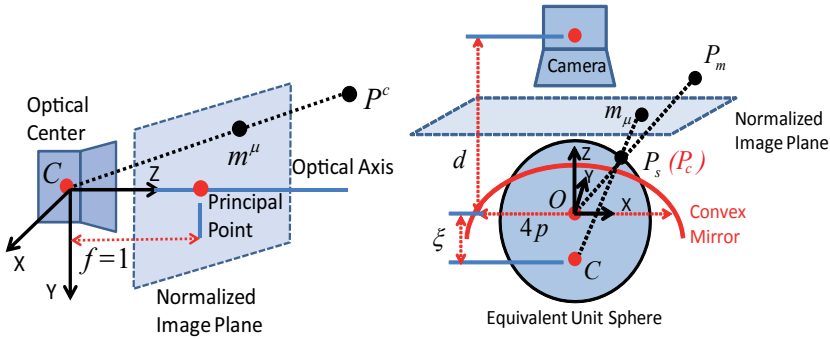
2.3 Camera model

In this thesis, most of the algorithms are designed for central perspective cameras, while some of them can be easily extended to central catadioptric cameras under the unifying camera model. Both camera models have a single effective viewpoint which is a desirable property as it enables the creation of perspective images without parallax. In the following, both camera models are briefly introduced.

2.3.1 Perspective camera model

This section introduces the central perspective projection of points in space onto the image plane. The complete chain of transformations that allows for mathematically modeling the image of a 3D point is presented, which follows the framework of [Ble09].

• **Normalized pinhole camera model:** As illustrated in Fig.2.1(a), under the pinhole model, the camera is represented by a 2D image plane and a 3D point called the optical center. The line from the optical center



(a) Normalized pinhole projection (b) Unifying point projection

Figure 2.1. Illustration of the projection model.

perpendicular to the image plane is called the optical axis of the camera and the point where the optical axis meets the image plane is called the principal point. A point in 3D space is projected onto the image plane by drawing a ray from the 3D point towards the optical center. Under the normalized pinhole model, with focal length $f = 1$, the image frame coincides with the normalized image coordinate system. By introducing homogeneous coordinate, the mapping of a 3D point $P^c = [x_c, y_c, z_c, 1]^T$, expressed in the camera frame, to a 2D point $\boldsymbol{m}^\mu = [x_\mu, y_\mu, 1]^T$, expressed in the normalized image frame, is:

$$\begin{bmatrix} x_\mu \\ y_\mu \\ 1 \end{bmatrix} \sim z_c \begin{bmatrix} x_\mu \\ y_\mu \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}, \tag{2.19}$$

where $\sim$ means equality up to scale.

• **Radial and tangential distortion:** The normalized pinhole camera model is an ideal projection. The real cameras introduce distortion. Here, the distortion model in [HS97] is introduced. It accounts for radial and tangential distortion. The distorted image coordinates $\boldsymbol{m}^d = [x_d, y_d]^T$ are

expressed as a function of the normalized image coordinates as:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \underbrace{(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \begin{bmatrix} x_\mu \\ y_\mu \end{bmatrix}}_{\textit{radial distortion}} + \underbrace{\begin{bmatrix} 2\kappa_4 x_\mu y_\mu + \kappa_5 (r^2 + 2x_\mu^2) \\ 2\kappa_5 x_\mu y_\mu + \kappa_4 (r^2 + 2y_\mu^2) \end{bmatrix}}_{\textit{tangential distortion}}, \tag{2.20}$$

in which $r = \sqrt{x_\mu^2 + y_\mu^2}$ is the distance to the principal point and κ_i, $i = 1, \ldots, 5$ are the parameters of the distortion model. Typical distortions are dominated by the coefficients κ_1 and κ_2. In many cases κ_3, κ_4 and κ_5 are assumed zero.

• **Intrinsic camera parameters:** In order to model the image of a 3D point in the pixel coordinate system, the intrinsic camera parameters have to be considered. First, the focal length of the camera f, given in pixels, represents the distance from the optical center to the image plane, i. e., the sensor. Second, the sensor may have non-square pixels. Hence, the aspect ratio α represents the ratio of scale factors between x and y directions. Third, the sensor pixels may even be non-rectangular. Hence, the skew parameter s describes the skew of the sensor axes. At last, the origin of the pixel coordinate system is at the top-left corner of the image plane instead of the principal point. The coordinates of the principal point in the image plane are (μ_0, ν_0), given in pixels. The intrinsic parameters are used to compose the affine transformation matrix $\boldsymbol{K}$, called calibration matrix, which describes the mapping from the distorted image coordinate system to the pixel coordinate system as:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f & s & \mu_0 \\ 0 & \alpha f & \nu_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}. \tag{2.21}$$

• **Extrinsic camera parameters:** In general, points in space are expressed in terms of a different Euclidean coordinate frame, e. g., the world coordinate frame w, instead of the camera frame c. The two coordinate frames are related via a rotation $\boldsymbol{R}_w^c$ and a translation $\boldsymbol{c}^w$, called the extrinsic camera parameters or the camera pose. By introducing homogeneous coordinates, the mapping of a point $P^w = [x_w, y_w, z_w, 1]^T$, expressed in the

world frame, to a point $P^c = [x_c, y_c, z_c, 1]^T$ expressed in the camera frame is:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_w^c & -\boldsymbol{R}_w^c \boldsymbol{c}^w \\ \boldsymbol{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}. \tag{2.22}$$

• **Central perspective camera model:** Combining the aforementioned transformations given in Eq.(2.19)-Eq.(2.22) and assuming the image distortion is corrected, the projection of a 3D point onto the image plane can be written as:

$$\begin{aligned} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} &\sim \begin{bmatrix} f & s & \mu_0 \\ 0 & \alpha f & \nu_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_w^c & -\boldsymbol{R}_w^c \boldsymbol{c}^w \\ \boldsymbol{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \\ &= \boldsymbol{K}\boldsymbol{R}_w^c \left[\boldsymbol{I}\right| - \boldsymbol{c}^w] \, P^w = \boldsymbol{P} \, P^w, \end{aligned} \tag{2.23}$$

in which the 3×4 matrix $\boldsymbol{P}$ is called the projection matrix. $\boldsymbol{P}$ has 11 degrees of freedom, defined up to an arbitrary scale.

2.3.2 The unifying camera model

The unifying projection model proposed by Mei [Mei07] for a camera with a single effective view point is introduced. Baker and Nayar [BN98] derived the class of central catadioptric cameras with this property under the assumption of the pinhole camera model. The four configurations that have this property are an orthographic camera associated to a parabolic mirror or a perspective camera associated to a hyperbolic, elliptical or planar mirror. A central catadioptric projection of a 3D point can be done in the following steps as shown in Fig.2.1(b).

(1). A 3D point P^m in the mirror frame is projected onto the unit sphere as $P^s = (x_s, y_s, z_s)^T$.

(2). The point on the sphere P^s is then shifted to the equivalent single viewpoint frame centered at $C(0, 0, -\xi)$ as $P^c = (x_s, y_s, z_s + \xi)^T$.

(3). The point P^c is then projected onto the normalized image plane from C as $\boldsymbol{m}^\mu = (\frac{x_s}{z_s+\xi}, \frac{y_s}{z_s+\xi}, 1)$ which is denoted as a function $\boldsymbol{m}^\mu = \boldsymbol{\hbar}(P^s)$.

(4). Then the radial and tangential distortions are added as $\boldsymbol{m}^d = \boldsymbol{\rho}(\boldsymbol{m}^\mu)$ in which $\boldsymbol{\rho}(\boldsymbol{m}^\mu)$ is the distortion function as given in Eq.(2.20).

(5). The final projection involves a generalized camera projection matrix K (with γ the generalized focal length, (μ_0, ν_0) the principal point, s the skew and α the aspect ratio) as:

$$\boldsymbol{m}^p = \boldsymbol{K}\boldsymbol{m}^d = \begin{bmatrix} \gamma & s & \mu_0 \\ 0 & \alpha\gamma & \nu_0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{m}^d. \tag{2.24}$$

Here, the parameters: ξ (the distance between the sphere center and the equivalent single viewpoint) and γ (the generalized focal length) are dependent on the mirror shape (i. e., its foci distance d and latus rectum $4p$) and the camera focal length f as given in Tab.2.2.

Table 2.2. The unifying model parameters [Mei07]

	Parabola	Hyperbola	Ellipse	Planar	Perspective
ξ	1	$\frac{df}{\sqrt{d^2+4p^2}}$	$\frac{df}{\sqrt{d^2+4p^2}}$	0	0
γ	$-2pf$	$\frac{-2pf}{\sqrt{d^2+4p^2}}$	$\frac{2pf}{\sqrt{d^2+4p^2}}$	$-f$	f

The backward projection from a point in the normalized image plane to the unit sphere is called lifting [Mei07] which can be calculated by the inverse function $\boldsymbol{\hbar}^{-1}$ as:

$$P^s = \boldsymbol{\hbar}^{-1}(\boldsymbol{m}^\mu) = \begin{bmatrix} \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} x \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} y \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} - \xi \end{bmatrix}. \tag{2.25}$$

Note that in the central perspective case, there is no mirror and only points with $z_c > 0$ are considered. Thus, it falls back to the standard projection model in Eq.(2.23) with an extra normalization to the sphere.

2.4 Spatial line representation

In this section, we give only a brief overview of the Plücker coordinates and the Plücker matrix representations which are used in this thesis. The summaries in [HZ04; BS05] give more detailed and complete introduction.

2.4.1 Plücker coordinates

The Plücker coordinates of a line L in space with direction $\boldsymbol{v}$ through a point P can be represented as [Wu+05]

$$L = (\boldsymbol{m},\ \boldsymbol{v}) \quad with \quad \boldsymbol{m} = P \times \boldsymbol{v}, \tag{2.26}$$

where $\boldsymbol{m}$ is called the line moment and is normal to the plane through the line and the coordinate origin, with the magnitude equal to the distance from the line to the origin. The Plücker constraints

$$\| \boldsymbol{v} \| = 1, \quad \boldsymbol{v}^T \boldsymbol{m} = 0, \tag{2.27}$$

guarantee that the degrees of freedom of an arbitrary line in space are four.

Bartoli and Sturm [BS05] introduced a slightly different definition of the Plücker coordinates. Given two homogeneous 3D points $A = (\boldsymbol{a}^T|a)^T$ and $B = (\boldsymbol{b}^T|b)^T$, one can represent the line joining them by a 6-vector

$$(\boldsymbol{m}', \boldsymbol{v}') : \begin{cases} \boldsymbol{m}' = \boldsymbol{a} \times \boldsymbol{b}, \\ \boldsymbol{v}' \ = a\boldsymbol{b} - b\boldsymbol{a}. \end{cases} \tag{2.28}$$

It is easy to verify that the definition of $\boldsymbol{v}'$ is exactly the direction of the line but unnormalized, and $\boldsymbol{m}'$ is equal to the line moment up to scale.

2.4.2 Plücker matrix

Hartley and Zisserman [HZ04] introduced two dual representations of a line which is defined by the conjunction of two points or the intersection of two planes and is represented by a 4×4 skew-symmetric homogeneous matrix. In particular, the line joining the two points A and B is represented by Plücker matrix $\boldsymbol{L}$ in vector notation as

$$\boldsymbol{L} = AB^T - BA^T. \tag{2.29}$$

A dual Plücker matrix $\boldsymbol{L}^*$ is obtained for a line formed by the intersection of two planes Π_1 and Π_2:

$$\boldsymbol{L}^* = \Pi_1\Pi_2^T - \Pi_2\Pi_1^T, \tag{2.30}$$

and has similar properties to $\boldsymbol{L}$. The matrix $\boldsymbol{L}^*$ can be obtained directly from $\boldsymbol{L}$ by a simple rewriting rule:

$$\frac{l_{34}^*}{l_{12}} = \frac{l_{42}^*}{l_{13}} = \frac{l_{23}^*}{l_{14}} = \frac{l_{14}^*}{l_{23}} = \frac{l_{13}^*}{l_{42}} = \frac{l_{12}^*}{l_{34}}, \tag{2.31}$$

where l_{ij} and l_{ij}^* are the i-th row and j-th column elements of matrices $\boldsymbol{L}$ and $\boldsymbol{L}^*$, respectively.

Supposing that the two 3D points A and B are represented by their homogeneous coordinates, then, according to the definition of the Plücker matrix (2.29), yield

$$\begin{aligned} \boldsymbol{L} &= \begin{bmatrix} \boldsymbol{a} \\ a \end{bmatrix} [\boldsymbol{b}^T \quad b] - \begin{bmatrix} \boldsymbol{b} \\ b \end{bmatrix} [\boldsymbol{a}^T \quad a] \\ &= \begin{bmatrix} -[(\boldsymbol{a} \times \boldsymbol{b})]_\times & b\boldsymbol{a} - a\boldsymbol{b} \\ (a\boldsymbol{b} - b\boldsymbol{a})^T & 0 \end{bmatrix}. \end{aligned} \tag{2.32}$$

Combine equations (2.26), (2.28) and (2.32) to obtain the relationship between the Plücker matrix and the Plücker coordinates as

$$\boldsymbol{L} = \begin{bmatrix} -[\boldsymbol{m}']_\times & -\boldsymbol{v}' \\ \boldsymbol{v}'^T & 0 \end{bmatrix} \sim \begin{bmatrix} [\boldsymbol{m}]_\times & \boldsymbol{v} \\ -\boldsymbol{v}^T & 0 \end{bmatrix}. \tag{2.33}$$

Eq.(2.33) will be used when deriving the line projection function.

2.5 Summary

This chapter gives a brief introduction about the notions and the basic mathematical identities upon which this thesis is based. A few particular forms of matrices are introduced and the camera models used in this thesis are described. The representations of spatial lines which will be employed in latter derivations are introduced as well.

Remember, a line cannot exist alone; it always brings a companion along. Do remember that one line does nothing; it is only in relation to another that it creates a volume.

Henri Matisse

Chapter 3

Line matching based on appearance similarity and geometric consistency

We begin by addressing the problem of line segment matching between image pairs. A line matching algorithm which utilizes both the local appearance of lines and their geometric attributes is presented. To overcome the problem of segment fragmentation and geometric variation, we extract lines in the scale space. To depict the local appearance of lines, we design a novel line descriptor called Line Band Descriptor (LBD). To evaluate the pairwise geometric consistency, we define the pairwise geometric attributes between line pairs. Then we build a relational graph for candidate line matches and employ a spectral technique to solve this matching problem efficiently.
Large parts of this work have been pre-published in [ZK12a; ZK13]

3.1 Introduction

One of the challenging areas in computer vision is feature matching, which is a basic tool for applications in scene reconstruction [TK95], pattern recognition and retrieval [DD05], stereo SLAM [CLK09] and so on. Most of the existing matching methods in the literature are based on local point features [MS05a] which are deficient for low-texture scenes [WNY09]. On the contrary, line features are often abundant in these situations. Moreover, line features and other local features provide complementary information about scenes. Therefore, line segment matching is both desirable and indispensable in many applications. Although some progress was

achieved recently for the line matching problem [WNY09; FWH10], these approaches are computationally quite expensive, prohibiting their usage in many applications.

Several reasons make line matching a difficult problem [FWH10; SZ97], including: inaccurate locations of line endpoints, fragmentation of lines, lack of strongly disambiguating geometric constraints (e.g. epipolar constraints), lack of distinctive appearance in low-texture scenes, instabilities for large image transformations. To deal with these challenges, the approach in this chapter is built on three strategies.

The first is to extract lines in the scale space making the matching algorithm robust to the scale changes. Though there is some work on detecting and tracking scale invariant lines [Chm05; ANL08], the proposed multi-scale line extraction approach simply applies the EDLine [AT11] detector to a scale-space pyramid consisting of a set of octave images, because it is more efficient to detect features in the scale space [Low04; BETG08; SLS11; ERB11] than to directly extract scale invariant regions [TVG04].

The second strategy is to characterize the local appearance of line segments by the Line Band Descriptor (LBD) which is more efficient to compute than MSLD [WWH09]. Different from the edge descriptors proposed in [MZS03; MS08], the proposed line descriptors are not designed to overcome the large scale changes because it is inefficient to adjust the scale of support region for each line segment. Instead, the multi-scale line extraction approach is adopted to solve this problem more efficiently.

The third strategy is to combine the local appearance of lines and the geometric constraints between line pairs to build a relational graph. The dimension of the graph matching problem is reduced by checking the appearance similarities and geometric consistencies. A spectral method [LH05] is employed to solve the matching problem which avoids the combinatorial explosion inherent to the graph matching problem. The geometric relationship of corresponding line pairs in two images may be not exactly affine invariant because they are often not coplanar. However, for images without strong view point changes, most of the correctly corresponding line pairs tend to establish strong agreement links among each other while the incorrect assignments have weak links in the graph and few of them have strong links by accident. This property makes the

spectral technique a promising strategy to efficiently solve the matching problem.

Compared to state-of-the-art methods, experiments validate that the proposed line matching approach is faster to generate the matching results. It is also robust against various image transformations including occlusion, rotation, blurring, illumination changes, scale changes, and moderate view point changes even for non-planar scenes or low-texture scenes.

The rest of this chapter is organized as follows. The related work about line matching is reviewed in Sec.3.2. Sec.3.3 presents the way to extract lines in the scale space and to construct the line descriptors. Sec.3.4 introduces the processes to generate candidate matching pairs, to build the relational graph and to solve the graph matching problem via spectral technique. The descriptor performance evaluation is presented in Sec.3.5 and the experimental matching results are reported in Sec.3.6. Finally, we draw the summary in the last section.

3.2 Related work

Existing approaches to match lines are of three types: those that match individual line segments, those that match groups of line segments and those that perform line matching by employing point correspondences.

For matching lines in image sequences or small baseline stereos where extracted corresponding segments are similar, approaches based on matching individual lines are suitable [DF90; NPVCL08; WPHB09] because of their better computational performance. Among the first group, Wang et al. [WWH09] proposed a descriptor named Mean-Standard deviation Line Descriptor (MSLD) for line matching based on the appearance of the pixel support region. This approach achieves good matching results for moderate image variations in textured scenes.

Generally, approaches which match groups of line segments have the advantage that more geometric information is available for disambiguation. A large number of methods have been developed around the idea of graph-matching [AF87; CKP95; HS89; WH97], however, most of them are only for small baseline stereo image pairs. Bay et al. [BFG05] presented a wide baseline stereo line matching method which compares the histograms

of neighboring color profiles and iteratively eliminates mismatches by a topological filter. The results shown in their work are for structured scenes with small number of lines, thus the performance on images featuring a larger range of conditions is not clear. Wang et al. [WNY09] used line signatures to match lines between wide baseline images. To overcome the unreliable line detection problem, a multi-scale line extraction strategy which extracts lines by verifying multiple merging thresholds from the edge image is employed. A line signature is constructed for each extracted line. This approach significantly improves the repeatability of line signatures and therefore has a good matching performance. However, this method is computationally quite expensive because of the huge number of line signatures.

Given a set of point correspondences, Schmid and Zisserman [SZ97] took the epipolar constraint of line endpoints for short baseline matching and presented a plane sweep algorithm for wide baseline matching. Lourakis et al. [LHO00] used two lines and two points to construct a projective invariant for matching planar surfaces. Kim and Lee [KL10] presented a line matching method by using coplanar Line Intersection Context Features (LICF). More recently, Fan et al. [FWH10] explored an affine invariant from two points and one line. They utilized this affine invariant to match lines with known point correspondences. The main drawback of these approaches is the requirement of known epipolar geometry or point correspondences. Besides, their performance in low texture scenes is limited because of the lack of good point correspondences.

3.3 Line detection and description

In this section, we first present the approach to detect lines in the scale space. Then the way to construct the line descriptor is introduced. The main reason for proposing this new line descriptor is to depict the local appearances of lines more efficiently than MSLD [WWH09] without losing the matching performance.

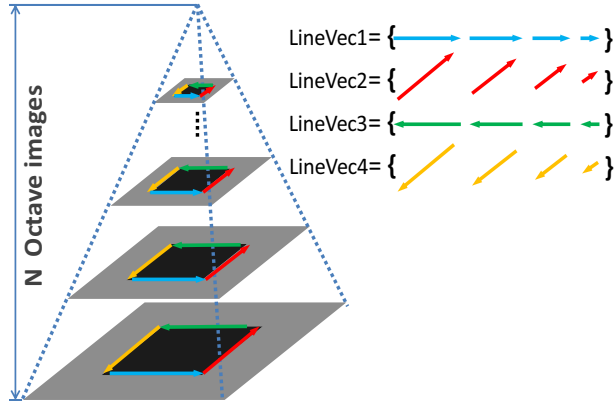


Figure 3.1. Illustration of the line detection in the scale space. The original image is down-sampled to generate a set of octave images. For each octave image, line segments are extracted by the EDLine [AT11] detector. For all the extracted lines, they are re-organized to form a set of LineVecs. Lines in the same LineVec have the same direction and are corresponding to the same region in the original image.

3.3.1 Detecting lines in the scale space

To overcome the fragmentation problem of line detection and to improve the performance for large scale changes, as illustrated in Fig.3.1, in our line detection framework we employ a scale-space pyramid consisting of n octave images which are generated by down-sampling the original image with a set of scale factors and Gaussian blurring. There is no intra-layer between two consecutive octaves. We first apply the EDLine [AT11] algorithm to each octave producing a set of lines in the scale space. Each line has a direction which is given by making the gradients of most edge pixels pointing from the left side of line to the right side of line. Then we re-organize them by finding corresponding lines in the scale space. For all lines extracted in the scale space, they are assigned a unique ID and stored into a vector called LineVec if they are related to the same event in the image (i. e., the same region of the image with the same direction). The final extracted results are a set of LineVecs. The line detection approach is different from Wang's [WNY09] by re-organizing all the line segments extracted in the scale space to form LineVecs, which reduces the dimension

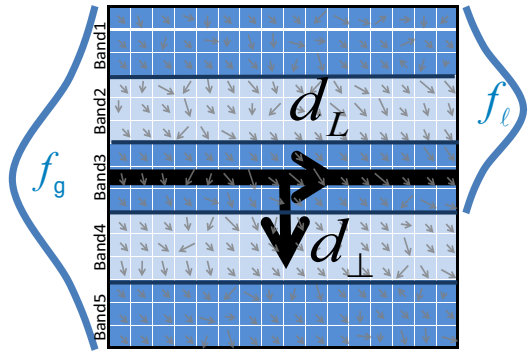


Figure 3.2. Illustration of the band representation. A local rectangular region around the line is chosen as the line support region (LSR). Two directions $\boldsymbol{d}_L$ and $\boldsymbol{d}_\perp$ are introduced. The LSR is divided into m bands with the width of w, (e.g. $m = 5, w = 3$). A global Gaussian function f_g is applied to all rows in the LSR. For each band (e. g., Band2), a local Gaussian function f_ℓ is applied to rows in the band and its nearest two neighbor bands. Small arrows represent the gradients of pixels in the LSR.

of the graph matching problem.

As shown in Fig.3.1, each LineVec may include more than one line in the scale space. To depict the local appearance of a LineVec, for each line in it, we will generate a line descriptor from the octave image where the line is extracted. The representation of the line support region and the construction of the line descriptor are introduced in the following.

3.3.2 The band representation of the line support region

Given a line segment in the octave image, the descriptor will be computed from the Line Support Region (LSR) which is a local rectangular region centered at the line as shown in Fig.3.2. This support region is divided into a set of bands $\{B_1, B_2, \ldots, B_m\}$ where each band is a sub-region of the LSR and parallel with the line. The numbers of bands m and the width of each band w will be discussed in Sec.3.5.1. Fig.3.2 illustrates an example of the LSR when $m = 5$ and $w = 3$. The length of the band naturally equals the length of the segment.

Similar to MSLD[WWH09], two directions which form a local 2D coordinate frame are introduced to distinguish parallel lines with opposite gradient directions and to make the descriptor rotation invariant. According to the line direction $\boldsymbol{d}_L$, the orthogonal direction $\boldsymbol{d}_\perp$ is defined as the clockwise orthogonal direction of $\boldsymbol{d}_L$. The middle point of the line is chosen as the origin of this local coordinate frame. The gradient of each pixel in the LSR is projected into this local frame $\boldsymbol{g}' = (\boldsymbol{g} \cdot \boldsymbol{d}_\perp, \boldsymbol{g} \cdot \boldsymbol{d}_L)^T \triangleq (g'_{d_\perp}, g'_{d_L})^T$ in which $\boldsymbol{g}$ and $\boldsymbol{g}'$ are the pixel gradients in the image frame and the local frame respectively.

Motivated by SIFT[Low04] and MSLD, two Gaussian functions are applied to each row of the LSR along $\boldsymbol{d}_\perp$. First, a global weighting coefficient $f_g(i) = (1/\sqrt{2\pi}\sigma_g)e^{-d_i^2/2\sigma_g^2}$ is assigned to the i-th row in the LSR, in which d_i is the distance of the i-th row to the center row of LSR and $\sigma_g = 0.5(m \cdot w - 1)$. Second, considering a band B_j, for rows in the band B_j and in its nearest neighbor bands B_{j-1}, B_{j+1}, a local weighting coefficient $f_\ell(k) = (1/\sqrt{2\pi}\sigma_\ell)e^{-d'^2_k/2\sigma_\ell^2}$ is assigned to the k-th row, in which d'_k is the distance of the k-th row to the center row of B_j and $\sigma_\ell = w$. The purpose of the global Gaussian window is to give less emphasis to gradients that are far from the line mitigating the sensitivity to small changes in the position of the LSR along the direction $\boldsymbol{d}_\perp$. The purpose of the local Gaussian window is to reduce boundary effects. It avoids that the descriptor changes abruptly as pixels move from one band to the next.

By this representation, we gain two advantages compared to the subregion representation introduced in [WWH09]: Firstly, it is more robust to small position changes in the direction $\boldsymbol{d}_L$ because in this case, most part of the image content in the band keeps unchanged with a little variation in the band boundary. Note that this feature is important since generally the position accuracy of a line is lower in the direction $\boldsymbol{d}_L$ than in the direction $\boldsymbol{d}_\perp$ due to the unstable line endpoints. Secondly, it is more computationally efficient because there is no overlap between bands in the direction $\boldsymbol{d}_L$ and the Gaussian weights are applied to each row rather than each pixel directly.

3.3.3 The construction of the line band descriptor

For a band B_j in the LSR, the band descriptor $\boldsymbol{bd}_j$ is computed from rows of B_j and its nearest two neighbor bands B_{j-1}, B_{j+1}. Specially, for the top and bottom bands B_1 and B_m, rows which are outside of the LSR will not be considered when computing the band descriptor of B_1 and B_m. After computing $\{\boldsymbol{bd}_j\}$, the line band descriptor LBD is simply generated by concatenating them:

$$LBD = (\boldsymbol{bd}_1^T, \boldsymbol{bd}_2^T, \ldots, \boldsymbol{bd}_m^T)^T. \tag{3.1}$$

Now, we construct the band descriptor $\boldsymbol{bd}_j$. For the k-th row in the band B_j or its neighbors, we accumulate the gradients of pixels within this row as:

$$\begin{aligned} v_{1k}^j &= \lambda \sum_{g'_{d_\perp}>0} g'_{d_\perp}, \quad v_{2k}^j = \lambda \sum_{g'_{d_\perp}<0} -g'_{d_\perp}, \\ v_{3k}^j &= \lambda \sum_{g'_{d_L}>0} g'_{d_L}, \quad v_{4k}^j = \lambda \sum_{g'_{d_L}<0} -g'_{d_L}, \end{aligned} \tag{3.2}$$

where the Gaussian coefficient $\lambda = f_g(k) f_\ell(k)$.

By stacking these four accumulated gradients of all rows associated with the band B_j, the band description matrix ($\boldsymbol{M}$) is constructed as:

$$\boldsymbol{M}_j = \begin{pmatrix} v_{11}^j & v_{12}^j & \cdots & v_{1n}^j \\ v_{21}^j & v_{22}^j & \cdots & v_{2n}^j \\ v_{31}^j & v_{32}^j & \cdots & v_{3n}^j \\ v_{41}^j & v_{42}^j & \cdots & v_{4n}^j \end{pmatrix} \in \Re^{4\times n}, \tag{3.3}$$

where n is the number of rows associated with B_j:

$$n = \begin{cases} 2w, & j = 1 \| m; \\ 3w, & else. \end{cases}$$

Now $\boldsymbol{bd}_j$ is simply constructed using the mean vector $\boldsymbol{\mu}_j$ and the standard deviation vector $\boldsymbol{\sigma}_j$ of the matrix $\boldsymbol{M}_j$: $\boldsymbol{bd}_j = (\boldsymbol{\mu}_j^T, \boldsymbol{\sigma}_j^T)^T \in \Re^8$.

Substituting in Eq.3.1, yields:

$$LBD = (\boldsymbol{\mu}_1^T, \boldsymbol{\sigma}_1^T, \boldsymbol{\mu}_2^T, \boldsymbol{\sigma}_2^T, \cdots, \boldsymbol{\mu}_m^T, \boldsymbol{\sigma}_m^T)^T \in \Re^{8m}. \tag{3.4}$$

Similar to [WWH09], the mean part $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_m\}$ and the standard deviation part $\{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_m\}$ of LBD are normalized separately because of their different magnitudes. Furthermore, to reduce the influence of non-linear illumination changes, the value of each dimension of LBD is restrained such that it is no larger than a threshold (0.4 is empirically found to be a good value). Finally, we re-normalize the restrained vector to get a unit LBD.

3.4 Graph matching using spectral technique

After introducing the line detection and description, in this section we present the method to construct the relational graph between two groups of LineVecs and to establish the matching results from this graph. Before that, some pre-processes are introduced first to reduce the dimension of the graph matching problem by excluding the clear non-matches.

3.4.1 Generating the candidate matching pairs

LineVecs detected in the reference and query images are deemed to be non-matches if they fail to pass the following tests according to their unary geometric attributes and their local appearance similarities.

• **Unary geometric attribute:** The unary geometric attribute considered in our work is the direction of LineVecs. Note that lines in the same LineVec have the same direction, so each LineVec has a unique direction. At first glance, the directions of corresponding LineVecs in the image pair are ambiguous and unreliable as image pairs can have arbitrary rotation changes. Though this is exactly true, there is often an approximate global rotation angle between image pairs. We can employ this attribute whenever it is available to reduce the number of candidate matches.

In [FWH10], the approximate rotation relationship between the reference and query images are calculated from the point feature correspondences. Inspired by this, although we do not have such point cor-

Figure 3.3. Illustration of line direction histograms. The first two images show the reference and query images with detected lines and the plot shows their direction histograms. The resolution of each bin is 20 degrees, so there are 18 bins for each histogram.

respondence information, we can directly compute the LineVec direction histograms of the reference and query images. We first calculate the two direction histograms from two images, then get the normalized histograms $(\boldsymbol{h}_r, \boldsymbol{h}_q)$ in which the subscript r denotes the reference image and q denotes the query image. Then we shift $\boldsymbol{h}_q$ by an angle θ varying from 0 to 2π and search for the approximate global rotation angle θ_g. By taking the angle as index in the histogram for simplicity, θ_g is estimated as:

$$\theta_g = \underset{0 \leqslant \theta \leqslant 2\pi}{\operatorname{argmin}} \|\boldsymbol{h}_r(x) - \boldsymbol{h}_q(x-\theta)\|. \tag{3.5}$$

Since it is not always suitable to approximate the perspective transformation of images by a global rotation change, we have to check whether the estimated rotation angle is genuine. In practice, if the perspective transformation can be approximated by a rotation, then the shifted histogram distance $\|\boldsymbol{h}_r(x) - \boldsymbol{h}_q(x-\theta_g)\|$ is small. Fig.3.3 gives an example of line direction histograms of an image pair. The estimated θ_g is 0.349 rad and the shifted histogram distance is 0.243. Besides, if the repeatability of the extracted lines in the images is low, then the histogram based method may fail, i. e., a wrong rotation angle may be accepted by the algorithm. To improve the robustness of this method, for lines falling in the same bin of the direction histogram, their length are accumulated as well. So, corresponding to a direction histogram, there is a length vector whose

i-th element is the accumulated length of all lines falling in the i-th bin of the direction histogram. In our implementation, we accept the estimated global rotation angle when the minimal shifted histogram distance is smaller than a threshold t_h and the minimal shifted length vector distance is smaller than a threshold t_l. In Sec.3.6.1, we will experimentally discuss these two thresholds. Once θ_g is accepted, for a pair of LineVecs to be matched, if $|\alpha - \theta_g| > t_\theta$ with α being the angle between their directions, then the line pair is considered to be a non-match without further checking their appearance similarities. If there is no accepted rotation angle between two images, then only the appearance similarities will be tested.

• **Local appearance similarity:** The local appearance similarity is measured by the distance of line descriptors. For each line in the LineVec, we generate a LBD descriptor vector ν from the octave image where the line is extracted. When matching two sets of LineVecs extracted from an image pair, the distances between all descriptors of a reference LineVec and a test LineVec are evaluated, and the minimal descriptor distance is used to measure the LineVec appearance similarity s. If $s > t_s$ in which t_s is the local appearance dissimilarity tolerance, then the corresponding two LineVecs will not be considered further.

After checking the unary geometric attributes of LineVecs and their local appearance similarities, the pairs passing these tests are taken as candidate matches. A set of loose thresholds should be chosen, otherwise there will be a larger chance of missing correct matches. In our implementation, the thresholds are empirically set as $t_\theta = \pi/4$, and $t_s = 0.35$. The number of candidate matches is larger than the number of real matches because one cannot only rely on the aforementioned verifications to decide the final matching results. However, the checking still significantly reduces the dimension of the following graph matching problem compared with direct combinations.

3.4.2 Building the relational graph

For a set of candidate matches, we build a relational graph whose nodes represent the potential correspondences and the weights on the links represent pairwise consistencies between them.

Given a set of κ candidate matches, the relational graph is represented

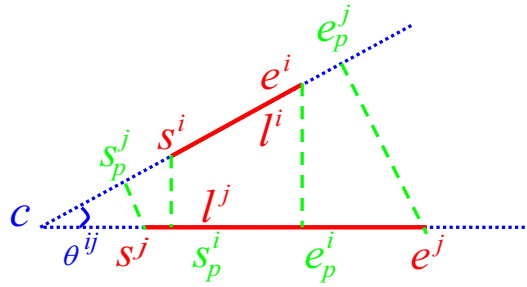


Figure 3.4. Illustration of the pairwise geometric attributes. $\boldsymbol{c}$ is the intersection of two lines. $(\boldsymbol{s}^i, \boldsymbol{e}^i)$ are the endpoints of line $\boldsymbol{l}^i$ and $(\boldsymbol{s}_p^i, \boldsymbol{e}_p^i)$ are their projections onto the line $\boldsymbol{l}^j$. Similarly, $(\boldsymbol{s}^j, \boldsymbol{e}^j)$ are the endpoints of line $\boldsymbol{l}^j$ and $(\boldsymbol{s}_p^j, \boldsymbol{e}_p^j)$ are their projections onto the line $\boldsymbol{l}^i$.

by an adjacency matrix $\boldsymbol{A}$ with a size of $\kappa \times \kappa$ following the terminology in [LH05]. The value of the element in row i and column j of $\boldsymbol{A}$ is the consistent score of candidate LineVec matches (L_r^i, L_q^i) and (L_r^j, L_q^j) where L_r^i, L_r^j are LineVecs in the reference image and L_q^i, L_q^j are LineVecs in the query image. The consistent score is computed from the pairwise geometric attributes and appearance similarities of the candidate matches.

For describing the pairwise geometric attributes of two LineVecs (L^i, L^j), as shown in Fig.3.4, we choose two lines $(\boldsymbol{l}^i, \boldsymbol{l}^j)$ which lead to the minimal descriptor distance between these two LineVecs and locate their endpoint positions in the original images. Then we describe the geometric attributes of $(\boldsymbol{l}^i, \boldsymbol{l}^j)$ by their intersection ratios (ρ^i, ρ^j), projection ratios (ϱ^i, ϱ^j) and relative angle θ^{ij}. ρ^i and ϱ^i are computed as:

$$\rho^i = \frac{\overrightarrow{\boldsymbol{s}^i\boldsymbol{c}} \cdot \overrightarrow{\boldsymbol{s}^i\boldsymbol{e}^i}}{|\overrightarrow{\boldsymbol{s}^i\boldsymbol{e}^i}|^2}, \quad \varrho^i = \frac{|\overrightarrow{\boldsymbol{s}^i\boldsymbol{s}_p^i}| + |\overrightarrow{\boldsymbol{e}^i\boldsymbol{e}_p^i}|}{|\overrightarrow{\boldsymbol{s}^i\boldsymbol{e}^i}|}. \tag{3.6}$$

ρ^j and ϱ^j can be calculated in the same way. The relative angle θ^{ij} is easily calculated from the line directions. These three attributes are invariant to changes of translation, rotation, and scale.

As introduced in Sec.3.4.1, we use the LBD descriptor vector $\boldsymbol{v}$ to

represent the local appearance of a line. Supposing the descriptors with minimal distances for LineVecs (L_r^i, L_q^i) in the reference and query images are $(\boldsymbol{\nu}_r^i, \boldsymbol{\nu}_q^i)$ and for LineVecs (L_r^j, L_q^j) are $(\boldsymbol{\nu}_r^j, \boldsymbol{\nu}_q^j)$ respectively, we get two sets of pairwise geometric attributes and local appearances for two candidate matches (L_r^i, L_q^i) and (L_r^j, L_q^j) as: $\{\rho_r^i, \rho_r^j, \varrho_r^i, \varrho_r^j, \theta_r^{ij}, \boldsymbol{\nu}_r^i, \boldsymbol{\nu}_r^j\}$ and $\{\rho_q^i, \rho_q^j, \varrho_q^i, \varrho_q^j, \theta_q^{ij}, \boldsymbol{\nu}_q^i, \boldsymbol{\nu}_q^j\}$. Then the consistent score a_{ij} is computed as:

$$a_{ij} = \begin{cases} 5 - d_\rho - d_\varrho - d_\theta - s_{\boldsymbol{\nu}}^i - s_{\boldsymbol{\nu}}^j, & if\ \Gamma\ is\ true; \\ 0, & else, \end{cases} \tag{3.7}$$

where d_ρ, d_ϱ and d_θ are the geometric similarities; $s_{\boldsymbol{\nu}}^i$, $s_{\boldsymbol{\nu}}^j$ are the local appearance similarities; and Γ is the condition. They are defined as:

$$\begin{cases} d_\rho = min(\frac{|\rho_r^i - \rho_q^i|}{t_\rho}, \frac{|\rho_r^j - \rho_q^j|}{t_\rho}), \\ d_\varrho = min(\frac{|\varrho_r^i - \varrho_q^i|}{t_\varrho}, \frac{|\varrho_r^j - \varrho_q^j|}{t_\varrho}), \\ d_\theta = \frac{|\theta_r^{ij} - \theta_q^{ij}|}{t_\theta}, \\ s_{\boldsymbol{\nu}}^i = \frac{\|\boldsymbol{\nu}_r^i - \boldsymbol{\nu}_q^i\|}{t_s}, \\ s_{\boldsymbol{\nu}}^j = \frac{\|\boldsymbol{\nu}_r^j - \boldsymbol{\nu}_q^j\|}{t_s}, \\ \Gamma \equiv \{d_\rho, d_\varrho, d_\theta, s_{\boldsymbol{\nu}}^i, s_{\boldsymbol{\nu}}^j\} \leqslant 1, \end{cases} \tag{3.8}$$

where $\Gamma \leqslant 1$ means that each element in Γ is not larger than 1. Compared with [WNY09], the definition of d_ρ in our work is more robust against the fragmentation problem of line detection because only if one pair of matched lines in the reference and query images is well extracted, then d_ρ can be very small no matter how badly the other pair is extracted. The definition of d_ϱ shares the same advantage. t_ρ, t_ϱ, t_θ and t_s are thresholds. In our implementation, they are set as $t_\rho = 1$, $t_\varrho = 1$, $t_\theta = \pi/4$ and $t_s = 0.35$. For all the candidate matches, we compute the consistent score among them and obtain the adjacency matrix $\boldsymbol{A}$. The diagonal entries of $\boldsymbol{A}$ equal zero as suggested by [LSH11] for better results. We also let $a_{ji} = a_{ij}$ to keep the symmetry.

3.4.3 Generating the final matching results

The matching problem is now reduced to finding the cluster of matches $\mathcal{LM}$ that maximizes the total consistent scores $\sum_{(L_r^i,L_q^i),(L_r^j,L_q^j)\in\mathcal{LM}} a_{ij}$ such that the mapping constraints are met. We use an indicator vector $\boldsymbol{x}$ to represent the cluster such that $\boldsymbol{x}(i) = 1$ if $(L_r^i, L_q^i) \in \mathcal{LM}$ and zero otherwise. Thus, the problem is formulated as:

$$\boldsymbol{x}^* = argmax(\boldsymbol{x}^T \boldsymbol{A}\ \boldsymbol{x}), \tag{3.9}$$

where $\boldsymbol{x}$ is subject to the mapping constraints. The general quadratic programming techniques are too computationally expensive to solve this problem. We employ the spectral technique which relaxes both the mapping constraints and the integral constraints on $\boldsymbol{x}$ such that its elements can take real values in $[0, 1]$.

By the Raleigh's ratio theorem [LH05], the $\boldsymbol{x}^*$ that will maximize $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ is the principal eigenvector of $\boldsymbol{A}$. What still remains is to binarize the eigenvector using mapping constraints and obtain a robust approximation of the optimal solution. The mapping constraints applied here are the sidedness constraint [BFG05; HS89] and the one-to-one constraint. Alg.1 summarizes the proposed line matching algorithm.

The final line matches can be directly retrieved from the matching results of LineVecs $\mathcal{LM}$. Note that, lines in the LineVec are located in the same region of image with the same direction, hence, for each pair of LineVec matches, it is enough to retrieve only one pair of line matches.

3.5 The descriptor performance evaluation

Before testing the proposed graph matching algorithm, we first analyze the influence of the LSR parameters, i. e., the number of bands m and the width of each band w, then evaluate the performance of LBD by comparing it with the well-known MSLD [WWH09] descriptor.

Mikolajczyk and Schmid [MS05b] established a benchmark to evaluate the performance of the local descriptors. We employ this framework to compare the performance of the line descriptors. The dataset in this experiment includes eight groups of images with following transformations:

Algorithm 1: Line matching based on appearance similarity and geometric consistency

Require: A pair of reference and query images

Extract LineVecs from the reference and query images by EDLine [AT11] in scale space to obtain two sets of LineVecs $\mathcal{L}_r$ and $\mathcal{L}_q$;
Estimate the global rotation angle θ_g of the image pair from the direction histograms of $\mathcal{L}_r$ and $\mathcal{L}_q$;
Compute the LBD descriptors of LineVecs in $\mathcal{L}_r$ and $\mathcal{L}_q$;
Generate a set of candidate matches $\mathcal{CM} = \{(L_r^1, L_q^1), (L_r^2, L_q^2), \ldots, (L_r^\kappa, L_q^\kappa)\}$ by checking the unary geometric attributes and local appearance similarities of LineVecs in $\mathcal{L}_r$ and $\mathcal{L}_q$;
Build the adjacency matrix $\boldsymbol{A}$ with a size of $\kappa \times \kappa$ according to the consistence scores of pairs in $\mathcal{CM}$;
Get principal eigenvector $\boldsymbol{x}^*$ of $\boldsymbol{A}$ by using ARPACK[LMSY11];
Initialize the matching result: $\mathcal{LM} = \oslash$ and flag: $stop = false$;
while $!stop$ **do**
  Find $p = \underset{1 \leqslant i \leqslant \kappa}{\operatorname{argmax}}(\boldsymbol{x}^*(i))$;
  if $\boldsymbol{x}^*(p) = 0$ **then**
    $stop = true$;
  else
    $\mathcal{LM} = \mathcal{LM} \cup \{(L_r^p, L_q^p)\}$, $\mathcal{CM} = \mathcal{CM} - \{(L_r^p, L_q^p)\}$ and $\boldsymbol{x}^*(p) = 0$;
    for each pair $(L_r^j, L_q^j) \in \mathcal{CM}$ **do**
      if (L_r^j, L_q^j) conflicts with (L_r^p, L_q^p) **then**
        $\mathcal{CM} = \mathcal{CM} - \{(L_r^j, L_q^j)\}$ and $\boldsymbol{x}^*(j) = 0$;
      end if
    end for
  end if
end while
return $\mathcal{LM}$

illumination changes, in-plane rotation, JPEG compression, image blurring, image occlusion, view point changes in the low-texture scene and

Figure 3.5. Examples in the image dataset including eight groups of image transformations. For each group, there are six images in the dataset raising from small to large transformations (Images in (f) are generated by increasing baseline between views in the low texture environment). The first and the last image of each group are shown here. The left image in each pair is chosen as the reference image.

the texture scene, and scale variations. There are six images in each group raising from small to large image transformations. Fig.3.5 shows example images in the dataset, image sets of (a), (c) and (d) are from [MS05b], and the rest are captured by ourselves to make sure images contain some line

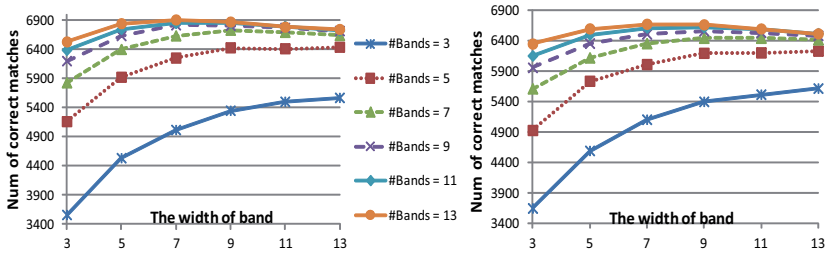
features. The images are either of planar scenes or acquired with fixed camera position. Therefore, they are always related by a homography (plane projective transformation). The ground truth of homographies is known. In order to better evaluate the descriptor performance for different image transformations, in this section we only consider lines extracted in the original image rather than in the octave images.

Since the ground truth of the image homographies is available, we first transfer the extracted lines in the query image into the reference image, then establish the ground truth of line correspondences by searching the parallel and close reference lines of the transferred lines. For the matching performance of descriptors reported in this section, we choose the nearest neighbor matching criterion to match lines according to their descriptor distance avoiding the prejudice of a distance threshold because different types of descriptors prefer different thresholds. Another advantage of this matching criterion is that the recall ratio (the number of correct matches divided by the number of ground correspondences) and the matching precision (the number of correct matches divided by the number of total matches) are only decided by the number of correct matches because the denominators for different descriptors are equal.

3.5.1 The descriptor dimension

The influence of the LSR parameters are analyzed experimentally. We vary m and w from 3 to 13, respectively. Fig.3.6 shows how the number of correct matches of all images is influenced by these two parameters. It is clear that LBD and MSLD share similar rules: the performance increases fast at the beginning with the increment of m or w, then reaches the best performance when $m = 9$ and w is about 7 or 9, after that there is a steady performance decrease. The results are well explained by the fact that larger values of m and w (i. e., larger LSR) make the descriptor more distinctive while they also reduce the repeatability of the LSR.

We also evaluate the time performance of these two descriptors which is reported in Tab.3.1. Although the time performance may change from image to image, the relative relationship will keep the same. We only show the results which are generated from an example image with the size of 900×600 and 573 extracted lines. Basically, the larger m and w are, the



(a) LBD (b) MSLD

Figure 3.6. The analysis of the descriptor dimension. The number of bands and the width of each band both vary from 3 to 13.

Table 3.1. The time performance of descriptors (**LBD**, MSLD). The results are given in **ms** by varying the number of bands m and the width of each band w. The time is measured on a 3.4GHz Intel(R) Core4 processor with 8GB of RAM.

m \ w	3	5	7	9	11	13
3	**4**, 13	**7**, 27	**9**, 41	**12**, 62	**14**, 85	**17**, 115
5	**6**, 23	**12**, 44	**15**, 72	**20**, 107	**24**, 151	**28**, 208
7	**9**, 33	**15**, 63	**21**, 103	**28**, 154	**34**, 217	**42**, 303
9	**12**, 42	**20**, 82	**28**, 137	**37**, 200	**45**, 281	**52**, 383
11	**15**, 52	**24**, 100	**34**, 165	**44**, 245	**53**, 346	**63**, 470
13	**17**, 61	**29**, 119	**40**, 195	**51**, 296	**65**, 418	**74**, 566

more computing time is consumed. LBD is less sensitive to the increase of m and w than that of MSLD, especially for the increase of w.

Based on the aforementioned evaluation, through the rest of the chapter, the descriptor will be computed from a LSR with $m = 9$ and $w = 7$, resulting in a 72-dimensional descriptor. Then the computing times of LBD and MSLD for the example image are 28ms, and 137ms, respectively.

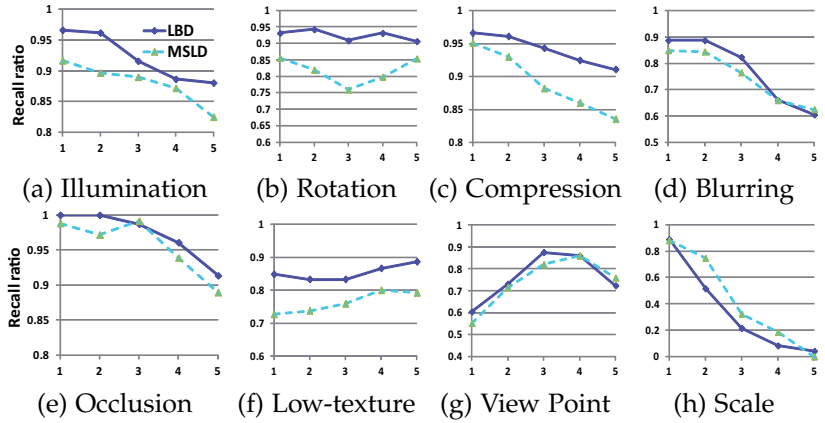


Figure 3.7. The comparison of the descriptor performance in terms of the recall ratio over five test images for each image transformation in the dataset.

3.5.2 Further comparison of MSLD and LBD

In this section, we report the comparison details of the descriptor performance for each group of images in our dataset (Fig.3.5). For each group of images, the recall ratios of MSLD and LBD are given in Fig.3.7.

Fig.3.7.(a) shows the performances of MSLD and LBD for the image illumination changes. From image 1 to image 5, the lighting condition gets worse. The recall ratios decrease with the increment of the lighting distortion. Fig.3.7.(b) shows the results for images which are generated by a set of in-plane rotation varying from 15° to 75°. It is interesting that when the rotation angle is 45° (between image 3 and the reference image), LBD and MSLD perform worst because of the aliasing of discrete lines. Fig.3.7.(c) and (d) show the descriptor performance against the image compression and the image blurring, respectively. Not surprisingly, the performance decreases with the increment of the image compression ratio or the image blurring. Fig.3.7.(e) shows the descriptor performance against image occlusion. To evaluate the occlusion effect, we first artificially add some vertical line features in a background image, then shift the region

of interest along the vertical direction of the artificial image to generate a set of smaller images as shown in Fig.3.5.(e). This process makes sure that for most of the lines, their LSR in the image sequence will change gradually (some part of the LSR moves out or in). The results show that the descriptor performance decreases with the increment of the image occlusion. Fig.3.7.(f) shows the descriptor performance in the low-texture scene. Images in this sequence are captured in front of the window with small view point changes. The results do not show drastic change in performance because of the small baseline between images. Fig.3.7.(g) shows the descriptor performance against large view point change. The view angles between the query images and the reference image range approximately from -70° to 60°. No doubt, the descriptors perform better when the absolute value of the view angle is smaller (image 3 and image 4). Fig.3.7.(h) shows the most challenging case for the descriptors, i. e., the large scale change. The scale ratios between the query images and the reference image range from 0.9 to 0.3. The performance decreases fast with the scale change.

Conclusively, for the most kinds of image transformations (Fig.3.7.(a-f)), it is clear that LBD performs better than MSLD. For the large scale change (Fig.3.7.(h)), as explained in Sec.3.1, all these two descriptors perform badly though MSLD is slightly better, because in this experiment, lines are only extracted in the original image. However, this can be made up by extracting lines in the scale space as addressed in Sec.3.3.1 and will be illustrated in the following experiments.

3.6 Line matching experiments

In this section, we first experimentally analyze the direction histogram based rotation estimation method proposed in Sec.3.4.1. Then we illustrate the performance improved by the multi-scale line extraction strategy and the geometric consistency verification. At last, we compare the proposed line matching algorithm with the state-of-the-art methods. The following experiments are performed on a 3.4GHz Intel(R) Core4 processor with 8GB of RAM.

Table 3.2. The experimental analysis of the direction histogram based rotation estimation method. For each image pair, the following results are reported: the ground truth and the estimated rotation angle in degrees, the minimal shifted histogram distance and the length vector distance.

	ground truth and estimated angle					histogram distance and length vector distance				
Img	1	2	3	4	5	1	2	3	4	5
a	0, 0	0, 0	0, 0	0, 0	0, 0	0.06, 0.06	0.07, 0.05	0.05, 0.05	0.08, 0.13	0.11, 0.10
b	15, 10	30, 30	45, 40	60, 60	**75, 70**	0.49, 0.48	0.42, 0.47	0.38, 0.42	0.15, 0.13	**0.53, 0.56**
c	0, 0	0, 0	0, 0	0, 0	0, 0	0.09, 0.09	0.05, 0.41	0.08, 0.05	0.14, 0.08	0.22, 0.16
d	0, 0	0, 0	0, 0	0, 0	0, 0	0.08, 0.07	0.19, 0.13	0.34, 0.25	0.37, 0.26	0.38, 0.30
e	0, 0	0, 0	0, 0	0, 0	0, 0	0.04, 0.04	0.04, 0.06	0.08, 0.15	0.09, 0.15	0.07, 0.12
f	0, 0	0, 0	0, 0	0, 0	0, 0	0.17, 0.10	0.24, 0.29	0.18, 0.20	0.17, 0.21	0.37, 0.30
g	**0, 80**	0, 0	0, 0	0, 0	0, 0	**0.58, 0.63**	0.30, 0.37	0.09, 0.10	0.21, 0.44	0.22, 0.45
h	0, 0	0, 0	0, 0	0, 0	0, 0	0.13, 0.11	0.16, 0.24	0.29, 0.30	0.36, 0.33	0.30, 0.34

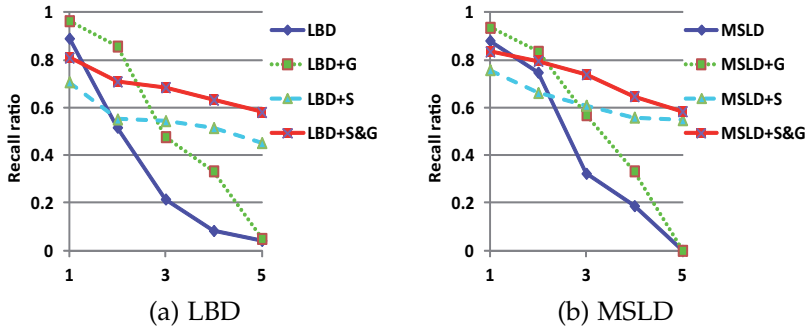
3.6.1 The analysis of the rotation estimation method

In Sec.3.4.1, we propose the direction histogram based rotation estimation method which is employed to reduce the number of candidate matches when the estimated rotation angle is accepted. The robustness of this method is very important for the matching algorithm. Generally, there are two kinds of errors: false negative (i. e., a correct rotation angle is rejected) and false positive (i. e., a wrong rotation angle is accepted). For false negative errors, the matching algorithm will be less efficient because more candidate line matches will be generated without checking their directions, but the matching algorithm can still match lines accurately. For false positive errors, the matching algorithm may fail because the wrong rotation angle is employed to generate wrong candidate line matches. Therefore, our priority goal is to control the false positive error as low as possible while keeping a small false negative error. In Sec.3.4.1, there are two thresholds: the minimal shifted histogram distance threshold t_h and the minimal shifted length vector distance threshold t_l. It is hard to give a theoretic analysis about the optimal setting of these thresholds because they depend on the scene environment, the image transformation and the line detection method. Here, we use the image set in Fig.3.5 to experimentally choose the proper values of these thresholds. For each image transformation, five images are compared to the reference image.

Tab.3.2 reports the ground truth and the estimated rotation angle, the minimal shifted histogram distance and the minimal shifted length vector distance between two images. Notice that the precision of the estimated rotation angle equals to the resolution of the direction histogram. In our implementation, it is 20 degrees. In this experiment, for the image pair g.1, the estimated rotation angle is obviously wrong. Hence, it should be rejected. In order to control the false positive error, we choose a pair of conservative thresholds ($t_h = t_l = 0.5$) although there is a false negative error (the correct estimated rotation angle is rejected for image pair b.5). This threshold setting generally works well even for a challenging image set as shown in the following experiment. If the efficiency of the matching algorithm is not important while the matching failure is definitely not acceptable, then one can set t_h and t_l to zero, i. e., the estimated rotation angle will always be rejected and the unary geometric attribute of lines will never be employed to generate the candidate matches.

3.6.2 The improvement of the matching performance

As introduced in Sec.3.1, in order to mitigate the problem of segment fragmentation and image variation, we propose to extract lines in the scale space (marked as +S). Besides, the geometric constraints are applied to improve the matching robustness (marked as +G). The sign +S&G denotes that both strategies are applied. We now show the influence of these two strategies on the matching performance. Here, we only illustrate the performance improvement for large scale changes, although the approaches are generally effective for other image transformations. Fig.3.8 shows the improved matching performance of the local appearance based algorithms (LBD and MSLD). Taken Fig.3.8.(a) for example, when the scale variation is large, the performance gain of LBD+S&G is obvious. Although when the scale variation is small, the recall ratio of LBD+S&G is slightly smaller than that of LBD or LBD+G because more lines are extracted in the scale space as compared to in the original image. The results also show that LBD+G is better than LBD, but it is still not robust to scale changes because we employ the geometric consistency verification as post process. When both strategies are applied, the matching performance of LBD+S&G is less sensitive to the scale changes and always better than



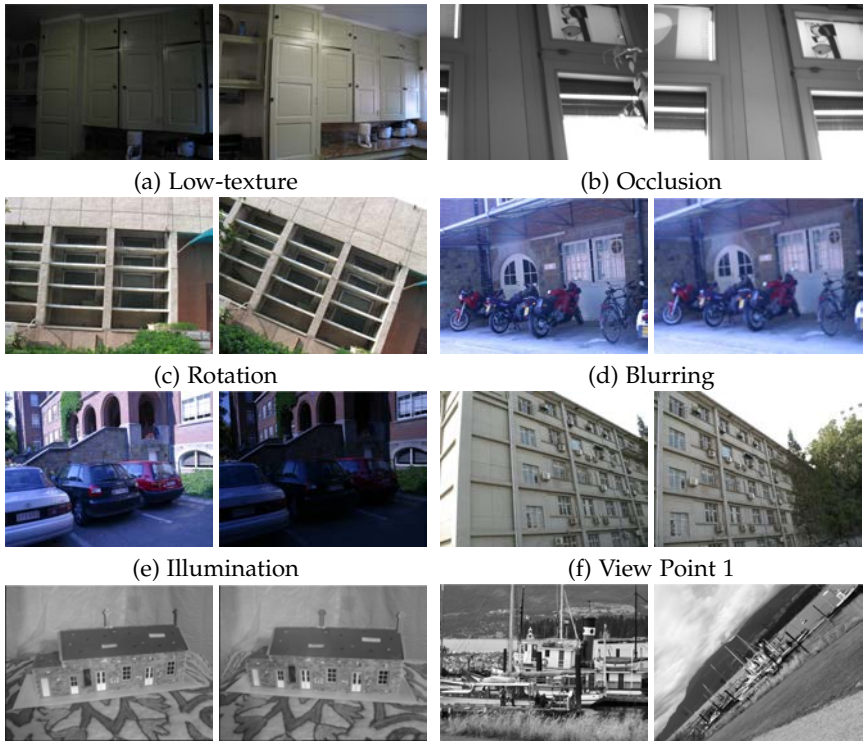
(a) LBD (b) MSLD

Figure 3.8. Illustration of the performance improvement. In (a), LBD is the original descriptor performance, LBD+G is generated by applying the geometric constraints, LBD+S is generated by detecting lines in the scale space and LBD+S&G is generated by employing both strategies. (b) is related to the MSLD descriptor.

LBD+S. The proposed two strategies are also effective for the MSLD based matching algorithms as shown in Fig.3.8.(b). The performance comparison between LBD+S&G and MSLD+S&G together with other state-of-the-art methods are presented in the next section.

3.6.3 Comparison with state-of-the-art methods

For a fair comparison with previous work, in this section we conduct the comparison experiment on the images demonstrated in the literature [WNY09; FWH10] (except the occlusion image pair which is captured in our office). The differences between the image set in Fig.3.9 and the previous image set (Fig.3.5) are that: (1), this image set only includes an image pair rather than an image sequence for each group of transformation; (2), most of the image pairs in this image set are more challenging. As introduced in Sec.3.1, the existing approaches to match lines are mainly of three types. Hence, the proposed algorithm LBD+S&G, is compared with three representatives from three groups which are recently reported to feature remarkable performance: the Line Signature (LS) [WNY09], the Line matching leveraged by Point correspondences (LP) [FWH10] and



(a) Low-texture (b) Occlusion

(c) Rotation (d) Blurring

(e) Illumination (f) View Point 1

(g) View Point 2 (h) Scale

Figure 3.9. Image dataset for comparison experiment. In each group, there are two images with large transformations.

the Mean-Standard deviation Line Descriptor (MSLD) [WWH09] (here, we use MSLD+S&G instead because it performs better than MSLD as illustrated in Fig.3.8(b)). The implementations of LS and LP are by the courtesy of their authors while MSLD+S&G is implemented by ourselves with parameter settings as recommended by its authors.

In this experiment, line detectors used by LBD+S&G and MSLD+S&G are the same as described in Sec.3.3.1. The extracted LineVecs are also used as input to LP. For LS, it uses its own multi-scale line detector

Table 3.3. The summary of parameters of the LBD+S&G algorithm and their settings in the experiments

Descriptor	m	number of bands	9
	w	width of band	7
Histogram	t_h	threshold of histogram distance	0.5
	t_l	threshold of length vector distance	0.5
Consistency	t_ρ	threshold of intersection ratio difference	1
	t_ϱ	threshold of projection ratio difference	1
	t_θ	threshold of relative angle difference	$\pi/4$
	t_s	threshold of appearance dissimilarity	0.35

Table 3.4. Comparison of our approach (LBD+S&G) with three line matching algorithms (MSLD[WWH09]+S&G, LP [FWH10], LS [WNY09]). For each image pair, the following results are reported: the number of total matches, the matching precision and the computing time.

	Img	LBD +S&G	MSLD +S&G	LP	LS		Img	LBD +S&G	MSLD +S&G	LP	LS		Img	LBD +S&G	MSLD +S&G	LP	LS
Total Matches	a	54	44	12	54	Match Precision (%)	a	94	92	67	96	Time(s)	a	0.11	0.35	5.5	8
	b	54	57	50	76		b	100	100	94	100		b	0.04	0.07	4.5	1
	c	263	240	253	188		c	100	100	100	100		c	0.38	0.48	13	26
	d	106	121	101	43		d	100	98	100	100		d	0.55	0.59	38	5
	e	245	223	262	241		e	100	100	100	100		e	0.59	0.65	28	8
	f	446	445	422	281		f	100	100	100	100		f	1.75	2.49	31	10
	g	87	78	117	151		g	100	100	91	98		g	0.20	0.42	22	8
	h	44	33	54	14		h	95	88	76	29		h	0.51	0.54	54	8

because the structure of line signature in [WNY09] is quite different from the structure of LineVec. The parameter settings of the proposed algorithm are summarized in Tab.3.3. The comparison results are reported in Tab.3.4. All the matched lines are checked one by one manually to test whether a matched line pair is correct or not. It is clear that LP performs worst for the low-texture scene, because the local appearances of lines are indistinguishable and the images lack stable point correspondences. The results also show that LBD+S&G performs slightly better than MSLD+S&G even for images with large scale variations because the drawback of LBD is made up by the multi-scale line extraction strategy. Surprisingly, LS has a bad matching result for image pair (h) which is inconsistent with the result

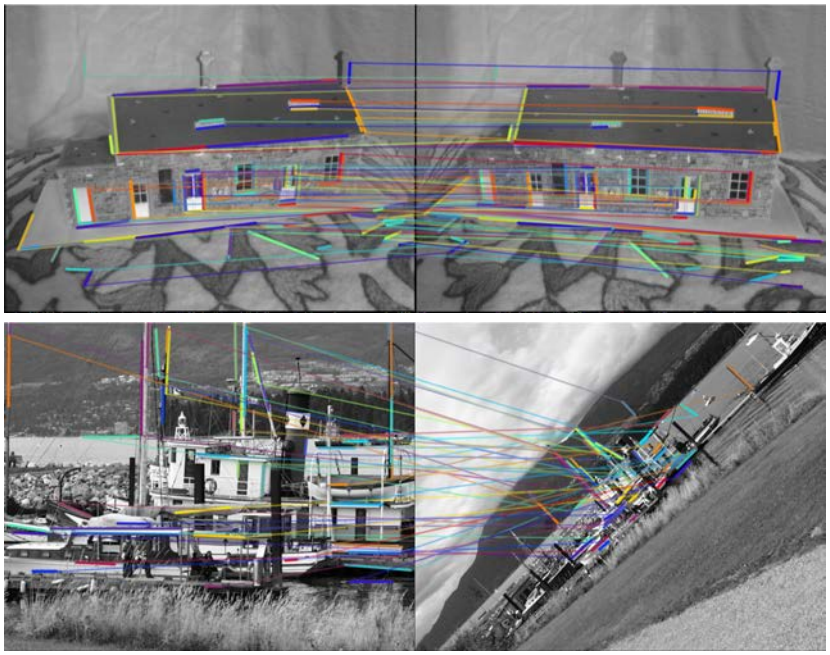


Figure 3.10. Illustration of the LBD+S&G matching results.

presented in [WNY09]. If we change the role of reference and query images, then we can get the same good result. This illustrates that the matching results of LS are dependent on the order of images in a pair. Compared to MSLD+S&G, LP and LS, the most superior feature of LBD+S&G is its time performance. Here, the time of LP given in Tab.3.4 is its complete processing time which includes generating point correspondences and matching lines.

Fig.1.1 and Fig.3.10 illustrate the matching results of LBD+S&G in three challenging scenes. The matched lines in each pair are assigned the same color and one of their endpoints is connected to illustrate their correspondences. These figures are better viewed in color. The image pair in Fig.1.1 is a low-texture planar scene with illumination and view

point changes. The first image pair in Fig.3.10 is a non-planar scene with moderate view point changes. The second image pair in Fig.3.10 is a textured scene with strong scale and rotation variations. The matching algorithm performs worse for these three image pairs than for the rest of the image pairs. Nevertheless, the results shown in the two figures are still quite acceptable and establish many line correspondences with few mismatches.

It is worth to note that similar to the parameter detection methods adopted in [WNY09] and [FWH10], it is empirical to find good parameter settings. However, these parameter settings are fixed as presented in Tab.3.3 for all the experiments. The results show that the algorithm works well for a large range of image variations.

3.7 Summary and discussion

In this chapter, we address the problem of line matching for image pairs under various situations: low-texture scenes, partial occlusion, rotation changes, blurred images, illumination changes, moderate viewpoint changes, and scale changes. We show the robustness and the efficiency of our graph matching process. The good performance achieved by the proposed algorithm is mainly because we detect lines in the scale space and combine the local appearance and geometric constraints together, which eliminates lots of mismatches. The source code of the proposed algorithm, the image dataset and the matching results are available on our website[1].

The dimension of the LBD descriptor to characterize the local appearance of line segment is a 72-vector with float elements. For retrieving line features from a large data base, it may be not efficient enough. The idea of Brief [CLSF10] or its enhancements Brisk [SLS11] and ORB [ERB11] can be employed to improve the matching process. In order to conduct a set of meaningful binary tests, the LSR should be normalized to a regular region with a fixed size.

Besides, the geometric constraints are enforced globally in this algorithm by using the spectral technique. For images undergoing a moderate

[1] http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Lilian+Zhang.

view point transformation, the global geometric constraints are maintained well. For strong wide baseline images of the non-planar scenes, the global constraints may be violated; then it is better to enforce the local geometric constraints like the approach in [WNY09], although it is more time consuming.

Straight lines go too quickly to appreciate the pleasures of the journey. They rush straight to their target and then die in the very moment of their triumph without having thought, loved, suffered or enjoyed themselves.

René Crevel

Chapter 4

Camera pose estimation from 2D/3D line correspondences

In this chapter we address the problem of camera pose estimation given a set of 2D/3D line correspondences. We first derive a general eighth order Perspective-3-Line (P3L) polynomial and analysis the solution of the P3L problem by investigating the symmetric property of line directions. Then we propose a non-iterative solution for the Perspective-n-Line (PnL) problem, which can efficiently and accurately estimate the camera pose for both a small number and a large number of line correspondences.
Large parts of this chapter have been pre-published in [ZXLK12; XZCK13]

4.1 Introduction

Determining the pose of a calibrated camera from n correspondences between 3D reference features and their 2D projections has numerous applications in robotics [Che91], computer vision [HZ04] and augmented reality [ZDB08]. For point features, the Perspective-n-Point (PnP) problem has been well studied with some recent remarkable progress [LMNF09; HR11; LXX12]. For line features, the Perspective-n-Line (PnL) problem remains a challenging topic. When the number of line correspondences is three, the problem is known as Perspective-3-Line (P3L) problem, which is of special interest, because the solution of the P3L problem plays a fundamental role when addressing the PnL problem.

Given a triplet of 2D/3D line correspondences, we first derive a general eighth order P3L polynomial by parametrizing the camera orientation as a rotation axis and an angle around this axis. In order to reduce the order

of the P3L polynomial, we investigate the symmetric structure of the line directions motivated by the parametrization of the P3P polynomial. The analysis shows that when the directions of lines are linearly dependent or orthogonal to each other, then the symmetric structure can be applied to generate a lower order P3L polynomial. Unfortunately, when the directions of three lines are linearly independent and non-orthogonal, the symmetric structure cannot be applied to reduce the complexity of the P3L problem. In this analysis, a complete scenario of three lines in all configurations is presented, which unifies the previous work [LHD88; Cag93; QZ08] in the same framework. Moreover, the analysis reveals that the P3P problem equals to a special case of the P3L problem: three lines form a triangle in space.

Based on the general eighth order P3L polynomial, we propose a Robust Perspective-n-Line (RPnL) solution which is the counterpart of the Robust Perspective-n-Point (RPnP) algorithm [LXX12]. In the framework of RPnL, we first divide the line correspondences into a set of line triplets by selecting a rotation axis. For each line triplet, we build an eighth order polynomial. Then the rotation axis in the camera frame is estimated by picking the global minimum of a cost function in a least square sense. After the estimation of the rotation axis, the rotation angle and translation vector are solved by SVD efficiently. A 3D alignment scheme [Ume91] is employed to normalize the estimated camera pose.

As demonstrated in the experimental results, the advantages of RPnL compared to the existing solutions of the PnL problem are as follows: (i) It is one of the non-iterative algorithms, hence no initialization is required; Furthermore, the computational complexity of RPnL is linear in the number of correspondences; (ii) When only a few line correspondences are available ($3 < n \leqslant 5$), RPnL drastically improves the accuracy and robustness compared to existing methods. (iii) When there are many line correspondences, RPnL achieves the same accuracy as the state-of-the-art methods in a fraction of their computing time.

The remainder of this chapter is organized as follows. After a brief review of the related work in Sec.4.2, we derive the P3L polynomial for three spatial lines in general configurations, then present a complete analysis about the solution of the P3L problem in Sec.4.3. Then the framework of RPnL algorithm is introduced in Sec.4.4. We evaluate the performance

of RPnL and the state-of-the-art methods with extensive simulations and validate the proposed algorithm on the real image sequences as well in Sec.4.5. Finally, we summarize this chapter in Sec.4.6.

4.2 Related work

The problem of estimating the camera pose from 2D/3D line correspondences has been addressed for more than two decades. In one of the earliest works, Dhome et al. [DRLR89] proposed a closed-form solution for the P3L problem by a polynomial approach. To derive the P3L polynomial, 3D lines are transformed into a model coordinate system and 2D lines are transformed into a virtual image plane in the viewer coordinate system. Later, Chen [Che91] proposed another approach to derive the P3L polynomial by introducing a canonical configuration. Unfortunately, the description of the overall rotation was unclear because three rotation axes in Eq.(4) of [Che91] were neither aligned with the camera coordinate frame nor aligned with the world coordinate frame. Some extra rotation are required to align the coordinate systems. The eighth order P3L polynomials derived in [DRLR89; Che91] are for three spatial lines in general configurations. For three lines in some special configurations, they may have some special geometric properties which can be applied to reduce to complexity of the P3L problem. Linnainmaa et al. [LHD88] proposed a solution of the P3L problem for three lines lying in a common plane and forming a triangle. A quartic equation is built by applying the constraint that the length of the triangle side is invariant under Euclidean transformation. Caglioti [Cag93] addressed the P3L problem for three lines lying in a common plane and intersecting at a common point, which is called the planar 3-line junction perspective problem. Recently, Qin and Zhu [QZ08] proposed a solution of the P3L problem for three lines forming a Z-shape in space, i. e., two lines are parallel and the third line intersects with both of them. The approaches in [LHD88; Cag93; QZ08] have great variation among each other. In this work, we will show their unification together with other special line configurations under the same framework.

The solutions of the P3L problem are not uniquely determined [Che91].

To find a unique pose solution, at least four line correspondences should be established. Liu et al. [LHF90] proposed an iterative method to first estimate the camera orientation and then the translation. Kumar and Hanson [KH94] improved the iterative algorithm which estimates the camera orientation and translation simultaneously (named as R_and_T). Christy and Horaud [CH99] proposed an iterative algorithm to estimate the camera pose with either a weak perspective or a para-perspective camera model. It is well known that these iterative algorithms require an initialization and may converge to a local minimum. Besides, in the absence of a good initialization, the computational cost of the iterative algorithms is generally high.

In the work of Liu et al. [LHF90], a non-iterative algorithm was also proposed when there are more than eight line correspondences. Ansar and Daniilidis et al. [AD03] improved this algorithm to make it work for four or more line correspondences. The algorithm employes the lifting approach to convert the polynomial system to a linear system about the components of the rotation matrix. This algorithm has $O(n^2)$ computational complexity and its accuracy is severely affected by the image noise. Recently, Mirzaei and Roumeliotis [MR11a] presented an algebraic approach to estimate the global optimum of the camera pose. The algorithm is non-iterative and has $O(n)$ computational complexity. The camera orientation is represented by the Cayley-Gibbs-Rodriguez (CGR) parametrization in their work. After relaxing the constraints on the rotation matrix, three cubic equations with three unknowns are generated to form a polynomial system with 27 candidate solutions. In their latter work [MR11b], they built a more precise polynomial system consisting of three fifth order equations and one cubic equation with four unknowns, which yields 40 candidate solutions. In their work, the polynomial system is solved by using algebraic geometry and the optimal solution is picked from those candidates in a least-square sense. The polynomial system is still computationally expensive because of the construction of the 120×120 Macaulay matrix. Besides, it returns too many candidate solutions.

Actually, the algorithm proposed in [MR11a] for the PnL problem is the counterpart of the approach presented in [HR11] for the PnP problem because both of them employ algebraic geometry to solve a polynomial system. Inspired by the success of the RPnP approach [LXX12] for the

PnP problem, in this work we seek to establish its counterpart for the PnL problem although that is more challenging. Considering their sub-problems, a fourth order polynomial can be generated [LX11; KSS11] for the Perspective-3-Point problem while a higher order polynomial must be involved for the P3L problem when 3D lines are in general configurations. These higher order polynomials increase the number of local minima of the cost function for the PnL problem. Furthermore, the mathematical expression of the perspective projection of lines is much different from that of points. These challenges are addressed in the following, and the RPnL algorithm is proposed and validated.

4.3 The Perspective-3-Line problem

In this section, we first derive a general P3L polynomial for three lines in general configurations. Then we discuss the solutions of the P3L problem by classifying three lines into three configurations.

4.3.1 The general P3L polynomial

Given a calibrated camera and three reference lines $L_i (i = 0, 1, 2)$ with their corresponding 2D projections on the image plane as $\boldsymbol{l}_i$, an eighth order polynomial can be achieved by using the 2D/3D correspondences. The derivation of the P3L polynomial is as follows.

Let $L_i = (\boldsymbol{v}_i, P_i)$ be a 3D line, in which $\boldsymbol{v}_i$ is the normalized vector giving the direction of the line and P_i is a point on the line. Let $\boldsymbol{l}_i = (\boldsymbol{p}_{is}, \boldsymbol{p}_{ie})$ be the projection of L_i on the image plane, in which $\boldsymbol{p}_{is}$ and $\boldsymbol{p}_{ie}$ are the endpoints of $\boldsymbol{l}_i$. For a given $\boldsymbol{l}_i$, a projection plane Π_i can be determined which passes through the projection center O_c, $\boldsymbol{l}_i$ and L_i. The normal of Π_i is denoted as $\boldsymbol{n}_i$ (see Fig.4.1).

By selecting a line L_0, we can form a model coordinate framework $O_m X_m Y_m Z_m$ whose Z_m-axis aligns with $\boldsymbol{v}_0$ and whose origin is located at the origin of the world frame. This can be achieved by choosing a rotation matrix $\boldsymbol{R}_w^m$ which rotates the direction vector in the world frame $\boldsymbol{v}_0^w$ to the model frame $\boldsymbol{v}_0^m$ and setting it as Z_m-axis, i. e., $\boldsymbol{v}_0^m = \boldsymbol{R}_w^m \boldsymbol{v}_0^w = [0, 0, 1]^T$. One possible choice of $\boldsymbol{R}_w^m$ is $[\boldsymbol{e}_1^T; \boldsymbol{e}_2^T; \boldsymbol{v}_0^{wT}]$ where $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are one pair

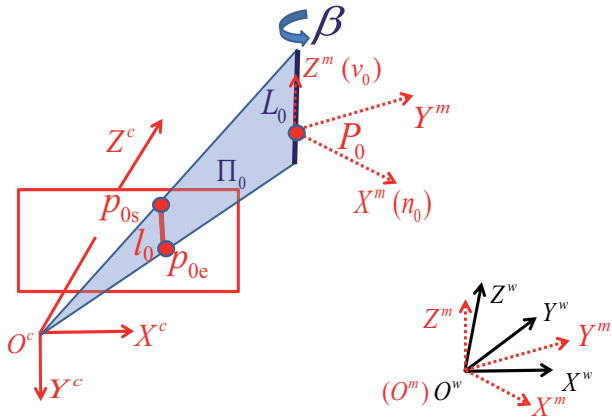


Figure 4.1. The geometry of P3L: the projection of a 3D line on the image plane and the coordinate frameworks.

of the orthogonal base of the null space formed by the linear system ${\boldsymbol{v}_0^w}^T \boldsymbol{x} = 0$. Then, apply the rotation matrix to the rest of the lines so as to transform $\boldsymbol{v}_i^w$ in the world frame to the model coordinate frame as $\boldsymbol{v}_i^m = \boldsymbol{R}_w^m \boldsymbol{v}_i^w$.

As L_0 lies on the plane Π_0, the line direction vector $\boldsymbol{v}_0$ is perpendicular to the plane normal $\boldsymbol{n}_0$. Hence, the rotation matrix from the camera to the model coordinate frame $\boldsymbol{R}_m^c$ must satisfy the constraint that $\boldsymbol{n}_0^T \boldsymbol{R}_m^c \boldsymbol{v}_0^m = 0$. In order to enforce this constraint, $\boldsymbol{R}_m^c$ can be parameterized as

$$\begin{aligned} \boldsymbol{R}_m^c &= \boldsymbol{R'}\, Rot(X,\alpha) Rot(Z,\beta) \\ &= \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} \\ r'_{21} & r'_{22} & r'_{23} \\ r'_{31} & r'_{32} & r'_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{4.1}$$

in which $\boldsymbol{R'}$ is an arbitrary orthogonal rotation matrix whose first column $[r'_{11}\ r'_{21}\ r'_{31}]^T$ equals to $\boldsymbol{n}_0$, $Rot(X,\alpha)$ denotes a rotation around the X-axis, and $Rot(Z,\beta)$ denotes a rotation around the Z-axis. In this way, $\boldsymbol{R}_m^c$ can be determined by two unknown variables α and β. It is easy to verify that

the constraint is fulfilled, i. e.,

$$\boldsymbol{n}_0^T\ \boldsymbol{R'}\ Rot(X,\alpha)Rot(Z,\beta)\ \boldsymbol{v}_0^m = [1,0,0]\ Rot(X,\alpha)Rot(Z,\beta)\ [0,0,1]^T \equiv 0. \tag{4.2}$$

By using the geometric constraint that $\boldsymbol{v}_i (i = 1, 2)$ should be perpendicular to the normal $\boldsymbol{n}_i$ of the plane Π_i, in the camera frame, we have the following two constraints

$$\begin{cases} \boldsymbol{n}_1 \cdot \boldsymbol{v}_1^c = \boldsymbol{n}_1^T \boldsymbol{R}_m^c \boldsymbol{v}_1^m = 0 \\ \boldsymbol{n}_2 \cdot \boldsymbol{v}_2^c = \boldsymbol{n}_2^T \boldsymbol{R}_m^c \boldsymbol{v}_2^m = 0 \end{cases}. \tag{4.3}$$

Let $\boldsymbol{n}_1^T \boldsymbol{R'} = [nx_1', ny_1', nz_1']$, $\boldsymbol{n}_2^T \boldsymbol{R'} = [nx_2', ny_2', nz_2']$, $\boldsymbol{v}_1^m = [vx_1, vy_1, vz_1]^T$ and $\boldsymbol{v}_2^m = [vx_2, vy_2, vz_2]^T$. By substituting Eq.(4.1) into Eq.(4.3), we have

$$\begin{cases} \sigma_1 \cos\beta + \sigma_2 \sin\beta + \sigma_3 = 0 \\ \sigma_4 \cos\beta + \sigma_5 \sin\beta + \sigma_6 = 0 \end{cases}, \tag{4.4}$$

in which

$$\begin{cases} \sigma_1 = vy_1\ ny_1' \cos\alpha + vy_1\ nz_1' \sin\alpha + vx_1\ nx_1' \\ \sigma_2 = vx_1\ ny_1' \cos\alpha + vx_1\ nz_1' \sin\alpha - vy_1\ nx_1' \\ \sigma_3 = vz_1\ nz_1' \cos\alpha - \ vz_1\ ny_1' \sin\alpha \\ \sigma_4 = vy_2\ ny_2' \cos\alpha + vy_2\ nz_2' \sin\alpha + vx_2\ nx_2' \\ \sigma_5 = vx_2\ ny_2' \cos\alpha + vx_2\ nz_2' \sin\alpha - vy_2\ nx_2' \\ \sigma_6 = vz_2\ nz_2' \cos\alpha - \ vz_2\ ny_2' \sin\alpha \end{cases}.$$

By solving Eq.(4.4), we have

$$\cos\beta = \frac{\sigma_2\sigma_6 - \sigma_3\sigma_5}{\sigma_1\sigma_5 - \sigma_2\sigma_4}, \quad \sin\beta = \frac{\sigma_3\sigma_4 - \sigma_1\sigma_6}{\sigma_1\sigma_5 - \sigma_2\sigma_4}. \tag{4.5}$$

Substituting Eq.(4.5) into $\cos^2\beta + \sin^2\beta = 1$, we have

$$(\sigma_2\sigma_6 - \sigma_3\sigma_5)^2 + (\sigma_3\sigma_4 - \sigma_1\sigma_6)^2 = (\sigma_1\sigma_5 - \sigma_2\sigma_4)^2. \tag{4.6}$$

Substituting $\sin^2\alpha = 1 - \cos^2\alpha$ into Eq.(4.6) and rearranging the terms,

we have

$$\sum_{k=0}^{4} u_k \cos^k \alpha = \sin \alpha \sum_{k=0}^{3} v_k \cos^k \alpha, \tag{4.7}$$

where u_k and v_k are coefficients which can be computed from nx'_i, ny'_i, nz'_i, vx_i, vy_i and vz_i for $i = 1, 2$. Taking the squares of both sides of Eq.(4.7) and letting $x = \cos \alpha$, an eighth order polynomial can be constructed as:

$$f(x) = \sum_{k=0}^{8} \delta_k x^k = 0, \tag{4.8}$$

where δ_k are computed from u_k and v_k as:

$$\begin{cases} \delta_0 = u_0^2 - v_0^2 \\ \delta_1 = 2(u_0 u_1 - v_0 v_1) \\ \delta_2 = u_1^2 + 2u_0 u_2 + v_0^2 - v_1^2 - 2v_0 v_2 \\ \delta_3 = 2(u_0 u_3 + u_1 u_2 + v_0 v_1 - v_1 v_2 - v_0 v_3) \\ \delta_4 = u_2^2 + 2u_0 u_4 + 2u_1 u_3 + v_1^2 + 2v_0 v_2 - v_2^2 - 2v_1 v_3 \\ \delta_5 = 2(u_1 u_4 + u_2 u_3 + v_1 v_2 + v_0 v_3 - v_2 v_3) \\ \delta_6 = u_3^2 + 2u_2 u_4 + v_2^2 - v_3^2 + 2v_1 v_3 \\ \delta_7 = 2(u_3 u_4 + v_2 v_3) \\ \delta_8 = u_4^2 + v_3^2 \end{cases}, \tag{4.9}$$

Eq.(4.8) is called the general P3L polynomial. Although in case of lines in some special configurations (such as orthogonal, parallel or intersection), a lower order of polynomial may be derived [QZ08; LZOY10] as discussed in Sec.4.3.2, the P3L polynomial will not be lower than eighth order for three spatial lines in general configurations. It is worth to point out that we decompose the over-all rotation from the world frame to the camera frame into a sequence of simple rotations by a different approach from [Che91; DRLR89]. Our decomposition is simpler than theirs despite the fact that the same constraints are enforced: the orthogonal constraint that the plane normal is orthogonal to the line direction, and the triangular constraint that the square sum of sine and cosine equals 1.

4.3.2 Discussion of the P3L solutions

In [GHTC03], a complete solution classification for the P3P problem is discussed which employs the Wu-Ritt's zero decomposition algorithm [WT86] to give a complete triangular decomposition for the P3P equation system. It is important to reveal the relationship of P3L solutions as well. The order of the P3P polynomial is 4, while that of the P3L polynomial reaches 8 as shown in Eq.(4.8). Similar algebraic analysis as in [GHTC03] will be intractable to address the complicated P3L problem. In this section, we discuss the solutions by investigating the symmetric structure of the problem. Let us first review the P3P equation system [LXX12]:

$$\begin{cases} t_0^2 + t_1^2 - t_0 t_1 \cos\alpha_{01} = d_{01}^2 \\ t_1^2 + t_2^2 - t_1 t_2 \cos\alpha_{12} = d_{12}^2 \\ t_0^2 + t_2^2 - t_0 t_2 \cos\alpha_{02} = d_{02}^2 \end{cases}, \tag{4.10}$$

in which t_i denotes the depth of point P_i ($i = 0, 1, 2$), d_{ij} denotes the distance between two points, and α_{ij} denotes the view angle of $\angle P_i O P_j$. According to Bezout's Theorem [CLO05], the P3P equation system has at most eight solutions. By utilizing the symmetric structure of the P3P problem (for any $\boldsymbol{t} = [t_0, t_1, t_2]^T$ being the solution of Eq.(4.10), $\boldsymbol{t} = -\boldsymbol{t}$ is also a solution), the parameter space of P3P can be easily divided into two parts: $t_0 > 0$ and $t_0 < 0$, and the P3P equation system can be easily reduced to form a fourth order polynomial [LX11].

Inspired by the success of the P3P parametrization, let us consider an interesting question: *Is it possible to reduce the order of the P3L polynomial from eight to any lower order by finding a symmetric structure in the P3L parameter space?* The answer is conditional. The symmetric property of P3L is closely related to the parameterization of the solution and the 3D configuration of lines, which is discussed in the following.

Symmetric structure of the P3L problem

Since the translation between the camera frame and the world frame can be linearly estimated given the rotation matrix, we focus on the solution of the rotation matrix in the following discussion. Two parametrization

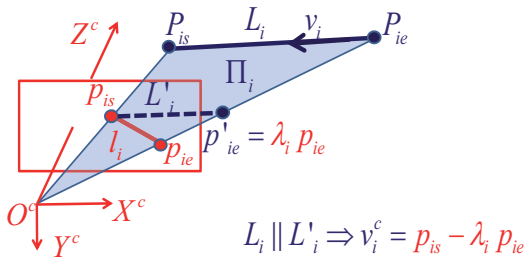


Figure 4.2. The parametrization of the line direction in the camera frame

methods can be used to solve the P3L problem:

(1) Rotation matrix $\boldsymbol{R}_w^c$: Directly parametrize the rotation matrix to several unknown variables, e. g., Euler angles.

(2) The directions of lines $\boldsymbol{V}^c$: Parametrize the directions of lines in the camera framework as $\boldsymbol{V}^c = [\boldsymbol{v}_0^c, \boldsymbol{v}_1^c, \boldsymbol{v}_2^c]$.

In the derivation of the general P3L polynomial, the rotation angles α and β are used to parametrize $\boldsymbol{R}_w^c$. To explore the symmetric structure, we primarily use $\boldsymbol{V}^c$ for parametrization in the discussion.

Without loss of generality, for a spatial line $L_i (i = 0, 1, 2)$ with direction $\boldsymbol{v}_i$, its projection in the image plane is $\boldsymbol{l}_i$. For a calibrated camera, we normalize the image plane to unit focal length and get the endpoints of the image line $\boldsymbol{p}_{is}(x_{is}, y_{is}, 1)$, $\boldsymbol{p}_{ie}(x_{ie}, y_{ie}, 1)$ as shown in Fig.4.2. For each line in space, it should lie on its projection plane Π_i as defined in Sec.4.3.1. In this plane, we choose a line L_i' passing through the point $\boldsymbol{p}_{is}$ that is parallel with L_i. The line L_i' will intersect the ray $O_c\boldsymbol{p}_{ie}$ at the point $\boldsymbol{p}'_{ie}(\lambda_i x_{ie}, \lambda_i y_{ie}, \lambda_i)$. Since $L_i \parallel L_i'$, the direction of the spatial line in the camera frame $\boldsymbol{v}_i^c$ can be parametrized as:

$$\boldsymbol{v}_i^c = \boldsymbol{p}_{is} - \boldsymbol{p}'_{ie} = \begin{bmatrix} x_{is} - \lambda_i x_{ie} \\ y_{is} - \lambda_i y_{ie} \\ 1 - \lambda_i \end{bmatrix}, \quad i = 0, 1, 2, \tag{4.11}$$

where λ_i is the parameter whose geometrical meaning is the depth ratio of the spatial endpoints because $d(P_{ie})/d(P_{is}) = d(\boldsymbol{p}'_{ie})/d(\boldsymbol{p}_{is}) = \lambda_i/1$.

Here, the depth $d()$ of a point is its projection on the optical axis, i. e., its Z-coordinate.

For the P3L problem, the direction of a line has the **symmetric structure** that $(\boldsymbol{v}_i, P_i)$ and $(-\boldsymbol{v}_i, P_i)$ represent the same line in space. Besides, the angles between lines in space are invariant under the Euclidean transformation. Formally, the relationship between the rotation matrix and line directions can be expressed as:

$$\boldsymbol{V}^c \ \boldsymbol{M} = \boldsymbol{R}_w^c \ \boldsymbol{V}^w, \tag{4.12}$$

in which the symmetric kernel $\mathbf{M}$ has eight combinations:

$$\boldsymbol{M} \in \begin{Bmatrix} diag[1,1,1], & diag[1,1,-1], & diag[1,-1,1], & diag[-1,1,1], \\ diag[1,-1,-1], & diag[-1,1,-1], & diag[-1,-1,1], & diag[-1,-1,-1] \end{Bmatrix}. \tag{4.13}$$

Here, $diag[m_1, m_2, m_3]$ means a diagonal matrix with diagonal entries m_1, m_2 and m_3. Given a solution of $\boldsymbol{V}^c$, only a subset of the symmetric kernels leads to a meaningful rotation matrix $\mathbf{R}_w^c$ in different configurations. For three lines in the world frame, their direction vectors $\boldsymbol{V}^w = [\boldsymbol{v}_0^w, \boldsymbol{v}_1^w, \boldsymbol{v}_2^w]$ have three possible configurations: (i) $Rank(\boldsymbol{V}^w) = 3$; (ii) $Rank(\boldsymbol{V}^w) = 2$ and; (iii) $Rank(\boldsymbol{V}^w) = 1$. Notice that, for the third configuration, since all three lines are parallel in space and only one direction can be determined, the rotation $\boldsymbol{R}_w^c$ from the world frame to the camera frame has infinite solutions. This line configuration is degenerated. Now we consider the first two in the following. For each configuration, we will discuss the solutions of the line directions in the camera frame $\boldsymbol{V}^c$ and the rotation matrix $\boldsymbol{R}_w^c$.

Line configuration 1: directions are linearly independent

Now we consider the first 3-line configuration: $Rank(\boldsymbol{V}^w) = 3$, i. e., line directions are linearly independent. There are four cases in this configuration: (1.a) $\boldsymbol{v}_i^w \perp \boldsymbol{v}_j^w \perp \boldsymbol{v}_k^w$ $(i, j, k \in \{0, 1, 2\})$; (1.b) $\boldsymbol{v}_i^w \perp \boldsymbol{v}_j^w$, $\boldsymbol{v}_i^w \perp \boldsymbol{v}_k^w$ and $\boldsymbol{v}_j^w \not\perp \boldsymbol{v}_k^w$; (1.c) $\boldsymbol{v}_i^w \perp \boldsymbol{v}_j^w$, $\boldsymbol{v}_i^w \not\perp \boldsymbol{v}_k^w$ and $\boldsymbol{v}_j^w \not\perp \boldsymbol{v}_k^w$; (1.d) $\boldsymbol{v}_i^w \not\perp \boldsymbol{v}_j^w \not\perp \boldsymbol{v}_k^w$.

Case 1.a: $\boldsymbol{v}_i^w \perp \boldsymbol{v}_j^w \perp \boldsymbol{v}_k^w$.

(i) Solution of $\boldsymbol{V}^c$: Since the relationships between lines are invariant

under the Euclidean transformation, in the camera frame we have

$$\boldsymbol{v}_0^c \cdot \boldsymbol{v}_1^c = 0,\ \boldsymbol{v}_0^c \cdot \boldsymbol{v}_2^c = 0,\ \boldsymbol{v}_1^c \cdot \boldsymbol{v}_2^c = 0. \tag{4.14}$$

By substituting Eq.(4.11) into Eq.(4.14), we get three second order polynomials:

$$\begin{cases} a_0 + \lambda_0 a_1 + \lambda_1 a_2 + \lambda_0\lambda_1 a_3 = 0 \\ b_0 + \lambda_0 b_1 + \lambda_2 b_2 + \lambda_0\lambda_2 b_3 = 0 \\ c_0 + \lambda_1 c_1 + \lambda_2 c_2 + \lambda_1\lambda_2 c_3 = 0 \end{cases}, \tag{4.15}$$

where a_j, b_j and c_j, $j = 0, \ldots, 3$ are directly computed from the endpoints in the image. After the variable resultant and substitution of λ_1, λ_2, we get a quadratic equation about λ_0 as:

$$g(\lambda_0) = w_2\lambda_0^2 + w_1\lambda_0 + w_0 = 0, \tag{4.16}$$

where the coefficients w_0, w_1, w_2 are easily derived from Eq.(4.15) as:

$$\begin{aligned} w_0 &= a_0b_0c_3 - a_2b_0c_2 - a_0b_2c_1 + a_2b_2c_0 \\ w_1 &= a_1b_0c_3 - a_3b_0c_2 + a_0b_1c_3 - a_2b_1c_2 - a_0b_3c_1 + a_2b_3c_0 - a_1b_2c_1 + a_3b_2c_0 \\ w_2 &= a_1b_1c_3 - a_3b_1c_2 - a_1b_3c_1 + a_3b_3c_0. \end{aligned}$$

After solving λ_0, the remaining two parameters λ_1 and λ_2 are linearly computed from Eq.(4.15). Then line directions in the camera frame are computed as:

$$\boldsymbol{V}^c = [\bar{\boldsymbol{v}}_0^c, \bar{\boldsymbol{v}}_1^c, \bar{\boldsymbol{v}}_2^c], \tag{4.17}$$

where $\bar{\boldsymbol{v}}_i^c$ is the normalized vector of $\boldsymbol{v}_i^c$. From Eq.(4.16), it is easy to infer that there are at most two solutions of $\boldsymbol{V}^c$ for case 1.a.

(ii) Solution of $\boldsymbol{R}_w^c$: Since $Rank(\boldsymbol{V}^w) = 3$, based on the symmetric structure of lines (Eq.4.12), we have

$$\boldsymbol{R}_w^c = \boldsymbol{V}^c\ \boldsymbol{M}\ (\boldsymbol{V}^w)^{-1}, \tag{4.18}$$

in which $\boldsymbol{M}$ is one of the symmetric kernels in Eq.(4.13). Only a part of kernels can generate valid rotation solutions which satisfy the constraints: $det(\boldsymbol{R}_w^c) > 0$ and $(\boldsymbol{R}_w^c)^T\boldsymbol{R}_w^c = \boldsymbol{I}$. Since lines are orthogonal to each other

in case 1.a, $\boldsymbol{V}^w$ and $\boldsymbol{V}^c$ are orthogonal matrices. By enforcing the first constraint, we have $det(\boldsymbol{R}_w^c) = det(\boldsymbol{V}^c)det(\boldsymbol{M})det((\boldsymbol{V}^w)^{-1}) > 0$, which requires $det(\boldsymbol{M}) > 0$. Therefore, the valid kernels are:

$$\boldsymbol{M} \in \{diag[1,1,1],\ diag[1,-1,-1],\ diag[-1,-1,1],\ diag[-1,1,-1]\}. \tag{4.19}$$

By enforcing the second constraint, we have

$$\begin{aligned}(\boldsymbol{R}_w^c)^T\boldsymbol{R}_w^c &= \left(\boldsymbol{V}^c\boldsymbol{M}(\boldsymbol{V}^w)^{-1}\right)^T\left(\boldsymbol{V}^c\boldsymbol{M}(\boldsymbol{V}^w)^{-1}\right)\\ &= (\boldsymbol{V}^w)^{-T}\boldsymbol{M}(\boldsymbol{V}^c)^T\boldsymbol{V}^c\boldsymbol{M}(\boldsymbol{V}^w)^{-1}\\ &= I.\end{aligned} \tag{4.20}$$

For all the symmetric kernels in Eq.(4.19), the second constraint is satisfied in Eq.(4.20). Hence, for line configurations in case 1.a, there are at most two solutions of $\boldsymbol{V}^c$, each of which can generate four solutions of $\boldsymbol{R}_w^c$, i. e., $\boldsymbol{R}_w^c = \boldsymbol{V}^c\ \boldsymbol{M}\ (\boldsymbol{V}^w)^{-1}$ in which $\boldsymbol{M}$ is one of the symmetric kernels in Eq.(4.19).

Case 1.b: $\boldsymbol{v}_i^w \perp \boldsymbol{v}_j^w$, $\boldsymbol{v}_i^w \perp \boldsymbol{v}_k^w$ and $\boldsymbol{v}_j^w \not\perp \boldsymbol{v}_k^w$.

(i) Solution of $\boldsymbol{V}^c$: Without loss of generality, by assuming $\boldsymbol{v}_0^w \perp \boldsymbol{v}_1^w$, $\boldsymbol{v}_0^w \perp \boldsymbol{v}_2^w$ and $\boldsymbol{v}_1^w \not\perp \boldsymbol{v}_2^w$, in the camera frame we have

$$\boldsymbol{v}_0^c \cdot \boldsymbol{v}_1^c = 0,\ \boldsymbol{v}_0^c \cdot \boldsymbol{v}_2^c = 0,\ \boldsymbol{v}_1^c \cdot \boldsymbol{v}_2^c = \|\boldsymbol{v}_1^c\|\ \|\boldsymbol{v}_2^c\| \cos\beta_{12}, \tag{4.21}$$

in which β_{12} is the angle between lines L_1 and L_2 in space. By substituting Eq.(4.11) into Eq.(4.21) and taking square of the last equation, we get

$$\begin{aligned}&a_0 + \lambda_0 a_1 + \lambda_1 a_2 + \lambda_0\lambda_1 a_3 = 0\\ &b_0 + \lambda_0 b_1 + \lambda_2 b_2 + \lambda_0\lambda_2 b_3 = 0\\ &c_0 + \lambda_1 c_1 + \lambda_2 c_2 + \lambda_1^2 c_3 + \lambda_2^2 c_4 + \lambda_1\lambda_2 c_5 + \lambda_1^2\lambda_2 c_6 + \lambda_1\lambda_2^2 c_7 + \lambda_1^2\lambda_2^2 c_8 = 0\end{aligned} \tag{4.22}$$

After the variable resultant and substitution of λ_1, λ_2, we get a quartic equation about λ_0 as:

$$g(\lambda_0) = w_4\lambda_0^4 + w_3\lambda_0^3 + w_2\lambda_0^2 + w_1\lambda_0 + w_0 = 0. \tag{4.23}$$

After solving λ_0, the remaining two parameters λ_1 and λ_2 are linearly computed from Eq.(4.22). From Eq.(4.23), it is obvious that there are at

most four solutions of $\boldsymbol{V}^c$.

(ii) Solution of $\boldsymbol{R}_w^c$: Similar to case 1.a, since $Rank(\boldsymbol{V}^w) = 3$, the rotation matrix can be computed from Eq.(4.18). By enforcing the constraint $\det(\boldsymbol{R}_w^c) > 0$, the possible symmetric kernels are the same as Eq.(4.19). Now considering the constraint $(\boldsymbol{R}_w^c)^T \boldsymbol{R}_w^c = \boldsymbol{I}$, since $\boldsymbol{v}_0 \perp \boldsymbol{v}_1$, $\boldsymbol{v}_0 \perp \boldsymbol{v}_2$ and $\boldsymbol{v}_1 \not\perp \boldsymbol{v}_2$, we have

$$\begin{aligned}(\boldsymbol{R}_w^c)^T \boldsymbol{R}_w^c &= \left(\boldsymbol{V}^c \boldsymbol{M} (\boldsymbol{V}^w)^{-1}\right)^T \left(\boldsymbol{V}^c \boldsymbol{M} (\boldsymbol{V}^w)^{-1}\right) \\ &= (\boldsymbol{V}^w)^{-T} \boldsymbol{M} \boldsymbol{A} \boldsymbol{M} (\boldsymbol{V}^w)^{-1},\end{aligned} \tag{4.24}$$

in which

$$\boldsymbol{A} = (\boldsymbol{V}^c)^T \boldsymbol{V}^c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & a \\ 0 & a & 1 \end{bmatrix}, \tag{4.25}$$

and $a = \cos\beta_{12} \neq 0$. Since $\boldsymbol{M} = diag[m1, m2, m3]$, in order to meet the constraint $(\boldsymbol{R}_w^c)^T \boldsymbol{R}_w^c = \boldsymbol{I}$, we have

$$\begin{gathered}\boldsymbol{M}\boldsymbol{A}\boldsymbol{M} = \boldsymbol{V}^{wT}\boldsymbol{V}^w \\ \Rightarrow \begin{bmatrix} m_1 m_1 & 0 & 0 \\ 0 & m_2 m_2 & m_2 m_3 a \\ 0 & m_3 m_2 a & m_3 m_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & a \\ 0 & a & 1 \end{bmatrix}.\end{gathered} \tag{4.26}$$

From the possible kernels in Eq.(4.19), we have $\boldsymbol{M} = diag[1, 1, 1]$ or $diag[1, -1, -1]$. Therefore, for line configurations in case 1.b, there are at most four solutions of $\boldsymbol{V}^c$, each of which can generate two solutions of $\boldsymbol{R}_w^c$, i. e., $\boldsymbol{R}_w^c = \boldsymbol{V}^c\ \boldsymbol{M}\ (\boldsymbol{V}^w)^{-1}$ in which $\boldsymbol{M} = diag[1, 1, 1]$ or $diag[1, -1, -1]$.

Case 1.c: $\boldsymbol{v}_i^w \perp \boldsymbol{v}_j^w$, $\boldsymbol{v}_i^w \not\perp \boldsymbol{v}_k^w$ and $\boldsymbol{v}_j^w \not\perp \boldsymbol{v}_k^w$.

Assuming $\boldsymbol{V}^c$ be a solution of line directions in the camera frame, since $Rank(\boldsymbol{V}^w) = 3$, the rotation matrix can be computed from Eq.(4.18). By enforcing the constraint $\det(\boldsymbol{R}_w^c) > 0$, the possible symmetric kernels are the same as Eq.(4.19). Now consider the constraint $(\boldsymbol{R}_w^c)^T \boldsymbol{R}_w^c = \boldsymbol{I}$. Without loss of generality, supposing that $\boldsymbol{v}_0 \perp \boldsymbol{v}_1$, $\boldsymbol{v}_0 \not\perp \boldsymbol{v}_2$ and $\boldsymbol{v}_1 \not\perp \boldsymbol{v}_2$, we have

$$\boldsymbol{A} = (\boldsymbol{V}^c)^T \boldsymbol{V}^c = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ a & b & 1 \end{bmatrix}, \tag{4.27}$$

in which $a = \cos\beta_{02} \neq 0$ and $b = \cos\beta_{12} \neq 0$ (β_{ij} is the angle between lines L_i and L_j in space). In order to meet the constraint $(\boldsymbol{R}_w^c)^T\boldsymbol{R}_w^c = \boldsymbol{I}$, we have

$$\begin{aligned} &\boldsymbol{MAM} = \boldsymbol{V}^{wT}\boldsymbol{V}^w \\ \Rightarrow &\begin{bmatrix} m_1m_1 & 0 & m_1m_3a \\ 0 & m_2m_2 & m_2m_3b \\ m_3m_1a & m_3m_2b & m_3m_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ a & b & 1 \end{bmatrix}. \end{aligned} \tag{4.28}$$

From the possible kernels in Eq.(4.19), we have $\boldsymbol{M} = diag[1,1,1]$. In this case, the symmetric property of the line direction cannot be applied to reduce the complexity of the P3L problem because each solution of $\boldsymbol{V}^c$ can only generate one solution of $\boldsymbol{R}_w^c$, i. e., $\boldsymbol{R}_w^c = \boldsymbol{V}^c(\boldsymbol{V}^w)^{-1}$. According to the P3L polynomial (Eq.4.8), there are at most eight solutions of $\boldsymbol{R}_w^c$ together with eight solutions of $\boldsymbol{V}^c$.

Case 1.d: $\boldsymbol{v}_i^w \measuredangle \boldsymbol{v}_j^w \measuredangle \boldsymbol{v}_k^w$.

Similar to lines in case 1.c, assume that $\boldsymbol{V}^c$ is a solution of line directions in the camera frame. As $\boldsymbol{v}_0^w \measuredangle \boldsymbol{v}_1^w \measuredangle \boldsymbol{v}_2^w$, to meet the constraint $(\boldsymbol{R}_w^c)^T\boldsymbol{R}_w^c = \boldsymbol{I}$, we have

$$\begin{aligned} &\boldsymbol{MAM} = \boldsymbol{V}^{wT}\boldsymbol{V}^w \\ \Rightarrow &\begin{bmatrix} m_1m_1 & m_1m_2a & m_1m_3b \\ m_2m_1a & m_2m_2 & m_2m_3c \\ m_3m_1b & m_3m_2c & m_3m_3 \end{bmatrix} = \begin{bmatrix} 1 & a & b \\ a & 1 & c \\ b & c & 1 \end{bmatrix}, \end{aligned} \tag{4.29}$$

in which $a = \cos\beta_{01} \neq 0$, $b = \cos\beta_{02} \neq 0$ and $c = \cos\beta_{12} \neq 0$ (β_{ij} is the angle between lines L_i and L_j in space). From the possible kernels in Eq.(4.19), we have $\boldsymbol{M} = diag[1,1,1]$. The symmetric property of the line direction cannot be applied to reduce the complexity of the P3L problem in this case. According to the P3L polynomial (Eq.4.8), there are at most eight solutions of $\boldsymbol{R}_w^c$ together with eight solutions of $\boldsymbol{V}^c$.

Line configuration 2: directions are linearly dependent

We now consider the second 3-line configuration: $Rank(\boldsymbol{V}^w) = 2$, i. e., line directions are linearly dependent. We separate this configuration into four cases: (2.a), $\boldsymbol{v}_i \parallel \boldsymbol{v}_j \perp \boldsymbol{v}_k$ ($i,j,k \in \{0,1,2\}$); (2.b), $\boldsymbol{v}_i \parallel \boldsymbol{v}_j \measuredangle \boldsymbol{v}_k$; (2.c), no lines

are parallel in space while their directions are linearly dependent and their projections form a triangle; (2.d), no lines are parallel in space while their directions are linearly dependent and their projections form a junction (i. e., intersect at a single point).

Case 2.a: $\boldsymbol{v}_i \parallel \boldsymbol{v}_j \perp \boldsymbol{v}_k$.

(i) Solution of $\boldsymbol{V}^c$: In this case, there are three possible situations: $\boldsymbol{v}_0 \parallel \boldsymbol{v}_1$, $\boldsymbol{v}_0 \parallel \boldsymbol{v}_2$ or $\boldsymbol{v}_1 \parallel \boldsymbol{v}_2$. Without loss of generality, supposing that $\boldsymbol{v}_0 \parallel \boldsymbol{v}_1$, then in the camera frame $\boldsymbol{v}_0^c$ and $\boldsymbol{v}_1^c$ should be equal up to a scale which yields:

$$\begin{cases} (x_{0s} - \lambda_0 x_{0e})(1 - \lambda_1) = (x_{1s} - \lambda_1 x_{1e})(1 - \lambda_0) \\ (y_{0s} - \lambda_0 y_{0e})(1 - \lambda_1) = (y_{1s} - \lambda_1 y_{1e})(1 - \lambda_0) \\ (x_{0s} - \lambda_0 x_{0e})(y_{1s} - \lambda_1 y_{1e}) = (x_{1s} - \lambda_1 x_{1e})(y_{0s} - \lambda_0 y_{0e}) \end{cases}. \tag{4.30}$$

By using the variable resultant, a quadratic about λ_0 can be generated from the first two rows of Eq.(4.30):

$$g'(\lambda_0) = w_2' \lambda_0^2 + w_1' \lambda_0 + w_0' = 0. \tag{4.31}$$

Note that $\lambda_0 = 1$ must be one root of Eq.(4.31) because $\lambda_0 = \lambda_1 = 1$ is a solution of the first two rows of Eq.(4.30). However, in general $\lambda_0 = \lambda_1 = 1$ does not satisfy the third row of Eq.(4.30). So $\lambda_0 = 1$ is a trivial root of Eq.(4.31) and is discarded. If it does satisfy the third row of Eq.(4.30), then it is kept. Actually, in this case, we have $w_1^2 - 4w_2 w_0 = 0$, i. e., $\lambda_1 = 1$ is the double root of Eq.(4.31). This happens when the projections of the parallel lines remain parallel in the image plane.

After solving λ_0, λ_1 can be computed from Eq.(4.30) and λ_2 can be computed by the constraint $\boldsymbol{v}_0^c \cdot \boldsymbol{v}_2^c = 0$. Since $\boldsymbol{v}_0$ is parallel to $\boldsymbol{v}_1$, they only determine one direction in the camera frame. The second direction is determined by $\boldsymbol{v}_2^c$. The third direction is determined by the cross product of $\boldsymbol{v}_0^c$ and $\boldsymbol{v}_2^c$ under the orthogonality constraint, i. e.,

$$\boldsymbol{V}^c = [\bar{\boldsymbol{v}}_0^c,\ \bar{\boldsymbol{v}}_2^c,\ \overline{\boldsymbol{v}_0^c \times \boldsymbol{v}_2^c}]. \tag{4.32}$$

If the projections of two parallel lines are not parallel in the image plane, then there is one solution of $\boldsymbol{V}^c$ corresponding to the non-trivial

root of Eq.(4.31). If the projections of two parallel lines are parallel in the image plane as well, then there is one solution of $\boldsymbol{V}^c$ corresponding to the double root of Eq.(4.31).

(ii) Solution of $\boldsymbol{R}_w^c$: By letting $\boldsymbol{V'}^w = [\boldsymbol{v}_0^w,\ \boldsymbol{v}_2^w,\ \overline{\boldsymbol{v}_0^w \times \boldsymbol{v}_2^w}]$, the solution of $\boldsymbol{V}^c$ will generate four solutions of the rotation matrix as $\boldsymbol{R}_w^c = \boldsymbol{V}^c\ \boldsymbol{M}\ (\boldsymbol{V'}^w)^{-1}$ in which $\boldsymbol{M}$ is one of the symmetric kernels in Eq.(4.19) because $\boldsymbol{V}^c$ defined in Eq.(4.32) and $\boldsymbol{V'}^w$ are orthogonal matrices.

Case 2.b: $\boldsymbol{v}_i \parallel \boldsymbol{v}_j \measuredangle \boldsymbol{v}_k$.

(i) Solution of $\boldsymbol{V}^c$: As in case 2.a, there are three possible situations as well: $\boldsymbol{v}_0 \parallel \boldsymbol{v}_1$, $\boldsymbol{v}_0 \parallel \boldsymbol{v}_2$ or $\boldsymbol{v}_1 \parallel \boldsymbol{v}_2$. Supposing $\boldsymbol{v}_0 \parallel \boldsymbol{v}_1$, then Eq.(4.31) can be built. After solving λ_0, λ_1 can be computed from Eq.(4.30) and λ_2 can be computed by the constraint

$$\boldsymbol{v}_0^c \cdot \boldsymbol{v}_2^c = \|\boldsymbol{v}_0^c\|\ \|\boldsymbol{v}_2^c\| \cos\beta_{02}, \tag{4.33}$$

in which β_{02} is the angle between two lines L_0 and L_2 in space, which is known in the world frame.

Again, there is only one solution of λ_0 from Eq.(4.31), either the non-trivial root or the double root. By substituting λ_0 into Eq.(4.33), at most two real solutions of λ_2 can be computed. Hence, there are at most two solutions of $\boldsymbol{V}^c$.

(ii) Solution of $\boldsymbol{R}_w^c$: In this case, each solution of $\boldsymbol{V}^c$ can generate 2 solutions of the rotation matrix as given in the following. The derivation of the rotation solution in this section is different from the derivation in Sec.4.3.2 because $\boldsymbol{V}^w$ is singular, i. e., Eq.(4.18) is not hold. Since $\boldsymbol{V}^c = \boldsymbol{R}_w^c\ \boldsymbol{V}^w$ and $Rank(\boldsymbol{V}^c) = Rank(\boldsymbol{V}^w) = 2$, from the Singular Value Decomposition of $\boldsymbol{V}^c$ and $\boldsymbol{V}^w$, we have

$$\boldsymbol{V}^c = \boldsymbol{U}^c\ \boldsymbol{S}\ \boldsymbol{D} = \boldsymbol{R}_w^c\ \boldsymbol{U}^w\ \boldsymbol{S}\ \boldsymbol{D}, \tag{4.34}$$

in which $\boldsymbol{S} = diag[s_0, s_1, 0]$. One solution of the rotation matrix can be generated as $\boldsymbol{R}_w^c = \boldsymbol{U}^c\ (\boldsymbol{U}^w)^{-1}$. By inverting three line directions in the camera frame simultaneously, i. e., right multiplying the symmetric kernel

matrix $diag[-1,-1,-1]$, we have

$$\begin{aligned} \boldsymbol{V}^c\ diag[-1,-1,-1] &= \boldsymbol{U}^c\ \boldsymbol{S}\ \boldsymbol{D}\ diag[-1,-1,-1] \\ &= \boldsymbol{U}^c\ diag[-1,-1,-1]\ \boldsymbol{S}\ \boldsymbol{D} \\ &= \boldsymbol{U}^c\ diag[-1,-1,\ \ \mathbf{1}]\ \boldsymbol{S}\ \boldsymbol{D} \\ &= \boldsymbol{R}_w^{'c}\ \boldsymbol{V}^w \\ &= \boldsymbol{R}_w^{'c}\ \boldsymbol{U}^w\ \boldsymbol{S}\ \boldsymbol{D}. \end{aligned} \tag{4.35}$$

Here, the third row holds because the third singular value in $\boldsymbol{S}$ is zero. From Eq.(4.35), the second solution of the rotation matrix generated by $\boldsymbol{V}^c$ is

$$\boldsymbol{R}_w^{'c} = \boldsymbol{U}^c\ diag[-1,-1,1]\ (\boldsymbol{U}^w)^{-1}. \tag{4.36}$$

Except $diag[1,1,1]$ and $diag[-1,-1,-1]$, for $\boldsymbol{M}$ being one of the symmetric kernels in Eq.(4.13), $\boldsymbol{V}^c\boldsymbol{M} = \boldsymbol{U}^c\boldsymbol{S}\boldsymbol{D}\boldsymbol{M} \neq \boldsymbol{U}^c\boldsymbol{M}\boldsymbol{S}\boldsymbol{D}$ for lines in case 2.b. It cannot generate a solution of the rotation matrix. Therefore, one solution of $\boldsymbol{V}^c$ can only generate two solutions of the rotation matrix in case 2.b.

Note that, if three lines are in case 2.a, i. e., two lines are parallel and the third line is orthogonal to them, then the SVD decomposition of $\boldsymbol{V}^c$ is

$$\boldsymbol{V}^c = \boldsymbol{U}^c\boldsymbol{S}\boldsymbol{D} = \boldsymbol{U}^c \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -\sqrt{2}/2 & 0 & -\sqrt{2}/2 \\ -\sqrt{2}/2 & 0 & \sqrt{2}/2 \end{bmatrix}. \tag{4.37}$$

It is easy to verify that

$$\boldsymbol{V}^c diag[-1,1,-1] = \boldsymbol{U}^c\boldsymbol{S}\boldsymbol{D}\ diag[-1,1,-1] = \boldsymbol{U}^c\ diag[1,-1,-1]\boldsymbol{S}\boldsymbol{D}, \tag{4.38}$$

and

$$\boldsymbol{V}^c diag[1,-1,1] = \boldsymbol{U}^c\boldsymbol{S}\boldsymbol{D}\ diag[1,-1,1] = \boldsymbol{U}^c\ diag[-1,1,-1]\boldsymbol{S}\boldsymbol{D}. \tag{4.39}$$

Hence, for three lines in case 2.a, the solution of $\boldsymbol{V}^c$ can generate two more solutions of the rotation matrix as $\boldsymbol{R}_w^{'c} = \boldsymbol{U}^c\ \boldsymbol{M}\ (\boldsymbol{U}^w)^{-1}$ in which $\boldsymbol{M} = diag[1,-1,-1]$ or $diag[-1,1,-1]$. This is consistent with the previous result as given in case 2.a.

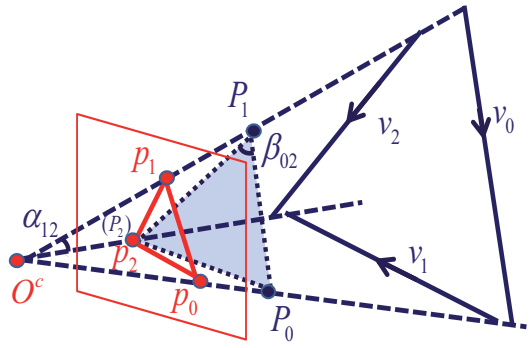


Figure 4.3. The parametrization of the line direction in the camera frame when their directions form a triangle.

Case 2.c: No lines are parallel in space while their directions are linearly dependent and their projections form a triangle.

(i) Solution of V^c: In this case, we can shift the three lines into a common plane to form a triangle $\triangle P_0P_1P_2$ as shown in Fig.4.3. Three image lines intersect at three points in the image plane as $\boldsymbol{p}_i(x_i, y_i, 1)$, $i = 0, 1, 2$. We choose the 3D point P_2 coincident with the image point $\boldsymbol{p}_2$. $P_0(\lambda_0 x_0, \lambda_0 y_0, \lambda_0)$ and $P_1(\lambda_1 x_1, \lambda_1 y_1, \lambda_1)$ are on the rays of $O^c\boldsymbol{p}_0$ and $O^c\boldsymbol{p}_1$ which make $P_0P_1 \parallel \boldsymbol{v}_0^c$, $P_0P_2 \parallel \boldsymbol{v}_1^c$, and $P_1P_2 \parallel \boldsymbol{v}_2^c$. Then the line directions in the camera frame are parametrized as:

$$[\boldsymbol{v}_0^c, \boldsymbol{v}_1^c, \boldsymbol{v}_2^c] = \begin{bmatrix} \lambda_0 x_0 - \lambda_1 x_1 & x_2 - \lambda_0 x_0 & x_2 - \lambda_1 x_1 \\ \lambda_0 y_0 - \lambda_1 y_1 & y_2 - \lambda_0 y_0 & y_2 - \lambda_1 y_1 \\ \lambda_0 - \lambda_1 & 1 - \lambda_0 & 1 - \lambda_1 \end{bmatrix}. \tag{4.40}$$

According to the law of sine, we have

$$\frac{|P_0P_1|}{\sin\beta_{12}} = \frac{|P_0P_2|}{\sin\beta_{02}} = \frac{|P_1P_2|}{\sin\beta_{01}}, \tag{4.41}$$

where β_{ij} is the angle between line directions $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, which is known in

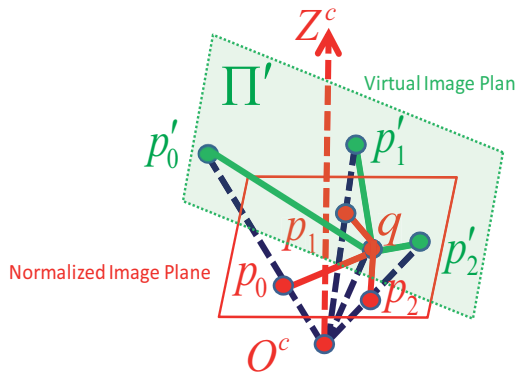


Figure 4.4. Virtual image plane

the world frame. According to the law of cosine, we have

$$\begin{aligned} |P_0P_1|^2 &= |O^cP_0|^2 + |O^cP_1|^2 - 2|O^cP_0| \cdot |O^cP_1| \cos\alpha_{01}, \\ |P_0P_2|^2 &= |O^cP_0|^2 + |O^cP_2|^2 - 2|O^cP_0| \cdot |O^cP_2| \cos\alpha_{02}, \\ |P_1P_2|^2 &= |O^cP_1|^2 + |O^cP_2|^2 - 2|O^cP_1| \cdot |O^cP_2| \cos\alpha_{12}, \end{aligned} \tag{4.42}$$

where α_{ij} is the angle of $\angle \boldsymbol{p}_i O^c \boldsymbol{p}_j$ which can be computed from the line projections. By taking the square of Eq.(4.41), and substituting Eq.(4.42) into it, after some straightforward manipulations, we obtain a quartic equation about λ_0 as:

$$g'(\lambda_0) = w'_4\lambda_0^4 + w'_3\lambda_0^3 + w'_2\lambda_0^2 + w'_1\lambda_0 + w'_0 = 0. \tag{4.43}$$

This quartic polynomial can have at most four real roots. Therefore, there are at most four solutions of $\boldsymbol{V}^c$.

(ii) Solution of $\boldsymbol{R}_w^c$: As discussed in case 2.b, when $Rank(\boldsymbol{V}^w) = 2$, each solution of $\boldsymbol{V}^c$ will generate two solutions of $\boldsymbol{R}_w^c$, as $\boldsymbol{R}_w^c = \boldsymbol{U}^c\ \boldsymbol{M}\ (\boldsymbol{U}^w)^{-1}$ in which $\boldsymbol{M} = diag[1,1,1]$ and $diag[-1,-1,1]$.

Case 2.d: No lines are parallel in space while their directions are linearly dependent and their projections form a junction.

(i) Solution of $\boldsymbol{V}^c$: As shown in Fig.4.4, the projections of three lines

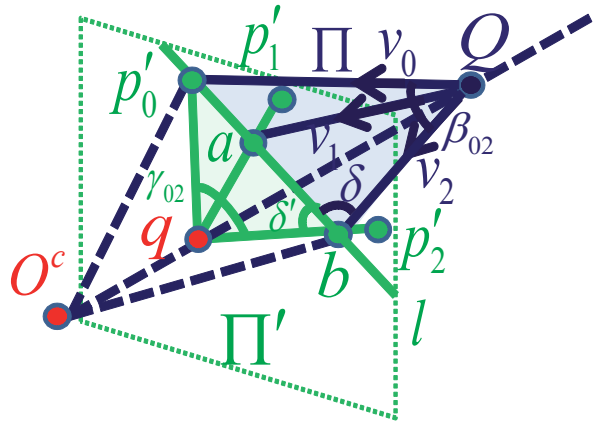


Figure 4.5. The parametrization of the line direction in the camera frame when their directions form a junction.

in the normalized image plane are $\boldsymbol{qp}_0$, $\boldsymbol{qp}_1$ and $\boldsymbol{qp}_2$, which intersect at the point $\boldsymbol{q}(x_q, y_q, 1)$. Generally, the junction point $\boldsymbol{q}$ is not the principal point of the image, i. e., the ray $O^c\boldsymbol{q}$ is not the normal of the image plane. In this case, we consider a virtual image plane Π' passing through the point $\boldsymbol{q}$ with the normal $O^c\boldsymbol{q}$ as shown in Fig.4.4. The projections of lines in the normalized image plane are projected onto this virtual image plane as $\boldsymbol{qp}'_0$, $\boldsymbol{qp}'_1$ and $\boldsymbol{qp}'_2$. The coordinates of point $\boldsymbol{p}'_i(x'_i, y'_i, z'_i)$ can be easily calculated by intersecting the ray $O^c\boldsymbol{p}_i$ with the plane Π'.

We now choose a point $Q(\lambda_q x_q, \lambda_q y_q, \lambda_q)$ on the ray $O^c\boldsymbol{q}$ which makes the line $Q\boldsymbol{p}'_0$ parallel to the line direction $\boldsymbol{v}_0$ as shown in Fig.4.5. Since the directions of three lines in space are linearly dependent, they are parallel to a common plane Π. We shift this common plane to pass through the line $Q\boldsymbol{p}'_0$. Supposing $\boldsymbol{l}$ be the intersection line between the virtual image plane Π' and the plane Π, the point $\boldsymbol{p}'_0$ must lie on the line $\boldsymbol{l}$. In the virtual image plane Π', $\boldsymbol{l}$ intersects $\boldsymbol{qp}'_1$ and $\boldsymbol{qp}'_2$ at point $\boldsymbol{a}$ and point $\boldsymbol{b}$, respectively. Then, $Q\boldsymbol{a}$ and $Q\boldsymbol{b}$ must be parallel with the line direction $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in space. The coordinates of $\boldsymbol{a}$ and $\boldsymbol{b}$ are

parameterized as $[x_q + \lambda_a(x'_1 - x_q), y_q + \lambda_a(y'_1 - y_q), 1 + \lambda_a(z'_1 - 1)]^T$ and $[x_q + \lambda_b(x'_2 - x_q), y_q + \lambda_b(y'_2 - y_q), 1 + \lambda_b(z'_2 - 1)]^T$, respectively.

γ_{ij} is the angle between lines $\boldsymbol{q}\boldsymbol{p}'_i$ and $\boldsymbol{q}\boldsymbol{p}'_j$ which can be calculated in the virtual image plane Π'. β_{ij} is the angle between line directions $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ which is known in the world frame. Let δ' be the angle $\angle \boldsymbol{p}'_0\boldsymbol{b}\boldsymbol{q}$ and δ be the angle $\angle \boldsymbol{p}'_0\boldsymbol{b}Q$. Considering the triangle $\triangle \boldsymbol{p}'_0\boldsymbol{b}\boldsymbol{q}$ and $\triangle \boldsymbol{p}'_0\boldsymbol{b}\boldsymbol{Q}$, according to the law of sine, we have

$$\frac{|\boldsymbol{p}'_0\boldsymbol{b}|}{\sin\gamma_{02}} = \frac{|\boldsymbol{p}'_0\boldsymbol{q}|}{\sin\delta'}, \quad \frac{|\boldsymbol{p}'_0\boldsymbol{b}|}{\sin\beta_{02}} = \frac{|\boldsymbol{p}'_0 Q|}{\sin\delta}. \tag{4.44}$$

Substituting the coordinates of $\boldsymbol{p}'_0$, $\boldsymbol{b}$, $\boldsymbol{q}$ and Q into Eq.(4.44) and taking the square of it, we have

$$\frac{(x_q + \lambda_b(x'_2 - x_q) - x'_0)^2 + (y_q + \lambda_b(y'_2 - y_q) - y'_0)^2 + (1 + \lambda_b(z'_2 - 1) - z'_0)^2}{\sin^2\gamma_{02}}$$
$$= \frac{(x_q - x'_0)^2 + (y_q - y'_0)^2 + (1 - z'_0)^2}{\sin^2\delta'}, \tag{4.45}$$

and

$$\frac{(x_q + \lambda_b(x'_2 - x_q) - x'_0)^2 + (y_q + \lambda_b(y'_2 - y_q) - y'_0)^2 + (1 + \lambda_b(z'_2 - 1) - z'_0)^2}{\sin^2\beta_{02}}$$
$$= \frac{(\lambda_q x_q - x'_0)^2 + (\lambda_q y_q - y'_0)^2 + (\lambda_q - z'_0)^2}{\sin^2\delta}. \tag{4.46}$$

δ and δ' can be computed by the linear relationship and the homogenous relationship between their cotangent values. Details of the derivation are given in Appendix A.1. After solving δ and δ', at most two real solutions of λ_b can be computed by solving the quadratic equation given in Eq.(4.45). By substituting the real solutions of λ_b into Eq.(4.46), λ_q is computed by solving this quadratic equation. Each real solution of λ_b can generate at most two real solutions of λ_q. On the other hand, for each solution of λ_b, one solution of λ_a can be computed by the constraint that $\boldsymbol{p}'_0$, $\boldsymbol{a}$ and $\boldsymbol{b}$ lie on the intersection line $\boldsymbol{l}$, i. e., $(\boldsymbol{p}'_0 \times \boldsymbol{b})^T\boldsymbol{a} = 0$. So there are at most four real solutions of $(\lambda_a, \lambda_b, \lambda_q)$ in total. The line directions in the

Table 4.1. The summary of the maximum number of solutions of $\boldsymbol{V}^c$ and $\boldsymbol{R}_w^c$. Here, i, j and k are non-repeating indices varying from 0 to 2. $\parallel$, $\perp$ and $\not\perp$ mean parallel, orthogonal and non-orthogonal. ∞ means infinity.

Configurations	Cases	$\#\boldsymbol{V}^c$	$\#\boldsymbol{R}_w^c$
$Rank(\boldsymbol{V}^w) = 3$	$\boldsymbol{v}_i \perp \boldsymbol{v}_j \perp \boldsymbol{v}_k$	2	2×4
	$\boldsymbol{v}_i \perp \boldsymbol{v}_j$, $\boldsymbol{v}_i \perp \boldsymbol{v}_k$ and $\boldsymbol{v}_j \not\perp \boldsymbol{v}_k$	4	4×2
	$\boldsymbol{v}_i \perp \boldsymbol{v}_j$, $\boldsymbol{v}_i \not\perp \boldsymbol{v}_k$ and $\boldsymbol{v}_j \not\perp \boldsymbol{v}_k$	8	8×1
	$\boldsymbol{v}_i \not\perp \boldsymbol{v}_j \not\perp \boldsymbol{v}_k$	8	8×1
$Rank(\boldsymbol{V}^w) = 2$	$\boldsymbol{v}_i \parallel \boldsymbol{v}_j \perp \boldsymbol{v}_k$	1	1×4
	$\boldsymbol{v}_i \parallel \boldsymbol{v}_j \not\perp \boldsymbol{v}_k$	2	2×2
	$\boldsymbol{v}_i$, $\boldsymbol{v}_j$, and $\boldsymbol{v}_k$ form a triangle	4	4×2
	$\boldsymbol{v}_i$, $\boldsymbol{v}_j$, and $\boldsymbol{v}_k$ form a junction	4	4×2
$Rank(\boldsymbol{V}^w) = 1$	$\boldsymbol{v}_i \parallel \boldsymbol{v}_j \parallel \boldsymbol{v}_k$	∞	∞

camera frame can thus be computed as:

$$\boldsymbol{v}_0^c = \frac{Q\boldsymbol{p}_0'}{|Q\boldsymbol{p}_0'|}, \quad \boldsymbol{v}_1^c = \frac{Q\boldsymbol{a}}{|Q\boldsymbol{a}|}, \quad \boldsymbol{v}_2^c = \frac{Q\boldsymbol{b}}{|Q\boldsymbol{b}|}. \tag{4.47}$$

There are at most four solutions of $\boldsymbol{V}^c = [\boldsymbol{v}_0^c, \boldsymbol{v}_1^c, \boldsymbol{v}_2^c]$.

(ii) Solution of $\boldsymbol{R}_w^c$: Similar to case 2.b and case 2.c, each solution of $\boldsymbol{V}^c$ can generate two solutions of $\boldsymbol{R}_w^c$ as $\boldsymbol{R}_w^c = \boldsymbol{U}^c\ \boldsymbol{M}\ (\boldsymbol{U}^w)^{-1}$ in which $\boldsymbol{M} = diag[1, 1, 1]$ and $diag[-1, -1, 1]$.

Summary of the P3L solutions

Considering three lines in space, there are three configurations: $Rank(\boldsymbol{V}^w) = 3$, $Rank(\boldsymbol{V}^w) = 2$ and $Rank(\boldsymbol{V}^w) = 1$. Based on the above analysis, in Tab.4.1, we summarize the maximum number of solutions of the line directions in the camera frame $\boldsymbol{V}^c$ and the rotation matrix $\boldsymbol{R}_w^c$.

In this section we give a complete solution analysis for three spatial lines in all possible configurations. By parameterizing the line direction in the camera frame instead of the rotation matrix directly, we employ the symmetric structure of the line direction to reduce the complexity of the P3L problem whenever it helps. The results are more straightforward

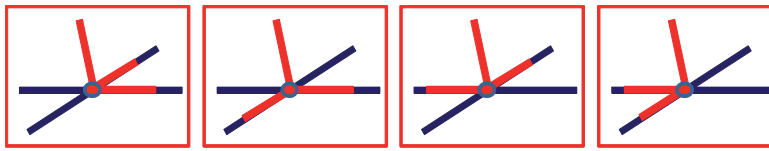


Figure 4.6. Four distinguishable junctions for the projections of three lines.

than others in the literature. Moreover, our approach unifies the previous approaches under the same framework. In [QZ08], the situation of three lines in Z-shape configuration is addressed which is a special situation in case 2.b. Our derivation is much simpler than the method in [QZ08]. In [LHD88], the situation of three lines forming a triangle in space is considered which is a special situation in case 2.c. In our case 2.c, three lines are parallel to a common plane, but they do not necessarily lie on the same plane. In contrast to ours, in [LHD88], three planar lines form a triangle in space, which actually equals to a P3P problem. The method in case 2.c solves a more general problem than P3P with the same complexity (a quartic polynomial). It offers an alternative approach for the P3P problem. In [Cag93], the situation of three lines lying on the same plane and intersecting at a point to form a junction is addressed which is a special situation in case 2.d. In our case 2.d, we find the maximum number of rotation solutions is 4×2, while in [Cag93], only two solutions of the rotation matrix are obtained, because the author ignores the symmetric property of the line direction. Actually, there are four distinguishable junctions for the projections of three lines as shown in Fig.4.6. Following the approach in [Cag93], each junction will generate two rotation solutions. Combining all the solutions, the maximum number of rotation solutions will be 4×2.

It must be pointed out that in our analysis, we only consider the line configurations in space, their projections in the image are not completely addressed because elaborating all the situations would be cumbersome. In [GHTC03], it is shown how complex it is to enumerate all the configurations of three points and their projections. The authors even gave up to present all their results of the two step geometric approach (cf. Sec.(5) of

[GHTC03]). Since the P3P problem is only a very special case of the P3L problem, one can imagine how tedious it would be to enumerate all the configurations of lines both in 2D and 3D.

Besides, after estimating the rotation matrix $\boldsymbol{R}_w^c$, the translation vector $\boldsymbol{c}$ between the camera frame and the world frame can be linearly estimated. The Euclidean transformation defined by $(\boldsymbol{R}_w^c, \boldsymbol{c})$ may transform the scene in front of the camera or behind it. Only the solutions that transform the scene in front of the camera are of interest.

In the next section, the P3L polynomial given in Eq.(4.8) will be used to solve the general PnL problem. We do not restrict lines in any special configuration in this chapter. In contrast, in Chapter 6, we will address the vanishing point estimation problem for lines in a Manhattan world, where spatial lines should be either parallel or orthogonal to each other. In that case, the results presented in this section can be directly employed to solve the vanishing point estimation problem.

4.4 The robust Perspective-n-Line algorithm

In this section, the robust Perspective-n-Line algorithm is presented, which estimates the camera pose from n pairs of 2D/3D line correspondences in general configurations.

4.4.1 Selecting a rotation axis to form the model frame

Given n reference lines $L_i (i = 1, \ldots, n)$ which are projected onto the normalized image plane as $\boldsymbol{l}_i$, we firstly select a line L_{i0} from $\{L_i\}$ as a rotation axis, based on which the model coordinate framework $O_m X_m Y_m Z_m$ is created (see Fig.4.1). The line with the longest projection length $|\boldsymbol{p}_{is}\boldsymbol{p}_{ie}|$ is selected as the rotation axis because longer edges are less affected by noise on their endpoints. Furthermore, the line with the second longest projection length is selected as an auxiliary line L_{i1}, so as to construct a polynomial equation system together with the rest of lines. The auxiliary line is introduced to keep the complexity of the proposed RPnL algorithm linear in the number of line correspondences.

4.4.2 Determinating the rotation axis

The line set $\{L_i\}$ can be divided into $n-2$ triplets $\{L_{i0}L_{i1}L_j|j=1,\ldots,n-2\}$. According to Eq.(4.8), each triplet yields an eighth order polynomial:

$$\begin{cases} f_1(x) = \sum_{k=0}^{8} \delta_{1k} x^k = 0 \\ f_2(x) = \sum_{k=0}^{8} \delta_{2k} x^k = 0 \\ \cdots \\ f_{n-2}(x) = \sum_{k=0}^{8} \delta_{(n-2)k} x^k = 0 \end{cases} . \tag{4.48}$$

Instead of directly solving the nonlinear equation system (4.48) by the linearization technique which would lead to an inconsistent result from redundant equations, we explore the local minima of the system in terms of the least square residual. We first define a cost function F as the square sum of the polynomials in Eq.(4.48), i. e., $F = \frac{1}{2}\sum_{i=1}^{n-2} f_i^2(x)$. The minima of F can be determined by finding the roots of its derivative $F' = \sum_{i=1}^{n-2} f_i(x) f_i'(x) = 0$. F' is a fifteenth order polynomial which can be easily solved by the eigenvalue method [PTVF07].

Remark 4.1. The 16-th order polynomial F has at most eight minima.

Proof. Assuming F has m stationary points, in which there are m_1 minima and m_2 maxima, $m_1 + m_2 \leqslant m$. As there exists at least one maximum between two minima, we have $m_1 - 1 \leqslant m_2$. As the stationary points of F are the real roots of F', we have $m \leqslant 15$. Therefore, $2m_1 - 1 \leqslant m_1 + m_2 \leqslant 15$, and we have $m_1 \leqslant 8$. □

In general, there are only a few real roots among the minima. These real roots are picked as candidate solutions. As soon as x is solved, the rotation angle α around the X-axis in Eq.(4.1) can be calculated and the rotation axis $\boldsymbol{v}_0$ in the camera frame can be determined, i. e., $\boldsymbol{v}_0^c = \boldsymbol{R}'\, Rot(X,\alpha)\, [0,0,1]^T$. The unknown variables remained in the camera pose are the rotation angle β around the axis $\boldsymbol{v}_0$ and the translation vector $\boldsymbol{c}$.

4.4.3 Solving the rotation angle and the translation vector

When the rotation axis $\boldsymbol{v}_0$ (i. e., the Z_m axis of the model coordinate frame) is determined, from Eq.(4.1), the rotation matrix from the camera to the

model coordinate framework $\boldsymbol{R}_m^c$ can be expressed as:

$$\boldsymbol{R}_m^c = \bar{\boldsymbol{R}}\ Rot(Z,\beta) = \begin{bmatrix} \bar{r}_{11} & \bar{r}_{12} & \bar{r}_{13} \\ \bar{r}_{21} & \bar{r}_{22} & \bar{r}_{23} \\ \bar{r}_{31} & \bar{r}_{32} & \bar{r}_{33} \end{bmatrix} \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.49}$$

in which $\bar{\boldsymbol{R}} = \boldsymbol{R}'\ Rot(X,\alpha)$, $c = \cos\beta$ and $s = \sin\beta$. According to the projection from 3D lines to 2D lines in the normalized image plane derived in Sec.5.4, we have:

$$\boldsymbol{n}_i \sim \boldsymbol{R}_m^c\left((P_i^m - \boldsymbol{c}) \times \boldsymbol{v}_i^m\right) = (\boldsymbol{R}_m^c P_i^m - \bar{\boldsymbol{c}}) \times (\boldsymbol{R}_m^c \boldsymbol{v}_i^m), \tag{4.50}$$

where P_i^m is a point on the line L_i^m with direction $\boldsymbol{v}_i^m$ in the model coordinate frame. The rotated translation vector $\bar{\boldsymbol{c}} = \boldsymbol{R}_m^c \boldsymbol{c} = [\bar{cx}, \bar{cy}, \bar{cz}]^T$. So we have

$$\boldsymbol{n}_i^T \boldsymbol{R}_m^c \boldsymbol{v}_i^m = 0, \quad \boldsymbol{n}_i^T (\boldsymbol{R}_m^c P_i^m - \bar{\boldsymbol{c}}) = 0; \quad i = 1, 2, \ldots, n. \tag{4.51}$$

For n lines, by substituting Eq.(4.49) into Eq.(4.51) and stacking these constraints, we get $2n$ homogenous linear equations with parameter vector $[c, s, \bar{cx}, \bar{cy}, \bar{cz}, 1]^T$. Letting $\boldsymbol{n}_i = [nx_i, ny_i, nz_i]^T$, $(\boldsymbol{n}_i^T \bar{\boldsymbol{R}}) = [\bar{nx}_i, \bar{ny}_i, \bar{nz}_i]$, $P_i^m = [px_i, py_i, pz_i]^T$ and $\boldsymbol{v}_i^m = [vx_i, vy_i, vz_i]^T$, we have:

$$\begin{bmatrix} & & & \cdots & & & \\ \bar{nx}_i vx_i + \bar{ny}_i vy_i & \bar{ny}_i vx_i - \bar{nx}_i vy_i & 0 & 0 & 0 & \bar{nz}_i vz_i \\ \bar{nx}_i px_i + \bar{ny}_i py_i & \bar{ny}_i px_i - \bar{nx}_i py_i & -nx_i & -ny_i & -nz_i & \bar{nz}_i pz_i \\ & & & \cdots & & & \end{bmatrix} \begin{bmatrix} c \\ s \\ \bar{cx} \\ \bar{cy} \\ \bar{cz} \\ 1 \end{bmatrix} = 0. \tag{4.52}$$

The rotation angle β and the rotated translation vector $\bar{\boldsymbol{c}}$ can be estimated by solving this linear system in Eq.(4.52) with SVD. Then, the rotation matrix from the camera frame to the world frame is computed as $\boldsymbol{R}_w^c = \boldsymbol{R}_m^c \boldsymbol{R}_w^m$ and the translation vector is computed as $\boldsymbol{c} = (\boldsymbol{R}_m^c)^T\ \bar{\boldsymbol{c}}$.

4.4.4 Determining the camera pose

Since the solution of $[c, s]$ is estimated from the linear system, it might not satisfy the triangular constraint $c^2 + s^2 = 1$ due to noise in the data. $\boldsymbol{R}_m^c$ computed from Eq.(4.49) should be normalized to a rotation matrix, so does the estimation of $\boldsymbol{R}_w^c$. This is achieved by a standard 3D alignment scheme [Ume91]. First, we translate the points on lines from the world frame $\{P_i^w\}$ to the camera frame $\{P_i^c\}$ by using the un-normalized estimation of $\boldsymbol{R}_w^c$ and $\boldsymbol{c}$, then project these points onto the projection plane of lines $\{\bar{P}_i^c\}$ as follows:

$$P_i^c = \boldsymbol{R}_w^c(P_i^w - \boldsymbol{c}), \quad \bar{P}_i^c = P_i^c - (P_i^c \cdot \boldsymbol{n}_i)\, \boldsymbol{n}_i; \quad i = 1, \ldots, n. \tag{4.53}$$

Finally, we retrieve the normalized camera pose by aligning two point sets $\{P_i^w\}$ and $\{\bar{P}_i^c\}$. In order to improve the alignment accuracy, the information of the line direction can also be employed. This is achieved by increasing the number of points in the two sets. To this end, for each 3D line, we add another point $\hat{P}_i^w$ into the world point set. In our implementation, $\hat{P}_i^w$ is the closest point on the line L_i to the origin of the world coordinate which can be computed from $(P_i^w, \boldsymbol{v}_i^w)$ as: $\hat{P}_i^w = P_i^w - (P_i^w \cdot \boldsymbol{v}_i^w)\, \boldsymbol{v}_i^w$. If P_i^w and $\hat{P}_i^w$ are coincident, then we shift $\hat{P}_i^w$ along the line direction. After that, we translate these additional world points into the camera frame and project them onto the projection plane of lines by Eq.(4.53) to get $\{\hat{\bar{P}}_i^c\}$. The normalized camera pose is estimated from the alignment of the enlarged two point sets $\{P_i^w, \hat{P}_i^w\}$ and $\{\bar{P}_i^c, \hat{\bar{P}}_i^c\}$.

After the pose normalization, we get a few candidate pose solutions which correspond to the minima of the polynomial system in Sec.4.4.2. For each candidate solution, we first evaluate it by the orthogonal error E_{er}, which is defined as

$$E_{er} = \sum_{i=1}^{n} \left(\boldsymbol{n}_i^T\, \boldsymbol{R}_w^c\, \boldsymbol{v}_i^w \right)^2. \tag{4.54}$$

We exclude those solutions with large orthogonal errors. From the remaining candidates, we select the one resulting in the smallest re-projection residual as the optimum. The re-projection residual E_{re} is defined as the difference between the observed image line and the re-projected image

line as follows [TK95]:

$$E_{re} = \sum_{i=1}^{n} \int_0^{\ell_i} h_i^2(s) ds = \sum_{i=1}^{n} \frac{\ell_i}{3} (h_{is}^2 + h_{is} h_{ie} + h_{ie}^2), \tag{4.55}$$

where ℓ_i is the length of image line $\boldsymbol{l}_i$, h_{is} and h_{ie} are the distances of observed line endpoints to the re-projected line. Sometimes, the re-projection residuals of two pose solutions E_{re}^1 and E_{re}^2 are very close. This happens when one of the solutions transforms the world scene in front of the camera while the other solution transforms the world scene behind the camera. In this case, we choose the former one as the final solution.

4.4.5 Discussion of the RPnL solution

The philosophy of the proposed RPnL solution is closely related to the previous works for Perspective-n-Point problem [FB81; LHD88; QL99], in which n points are divided into a serials of triplets, then unreasonable solutions are eliminated by finding a consistency among the solutions of the triplets. This idea can be regarded as a "bottom up" approach. In [FB81; LHD88], all the 3-point subsets are solved individually, then a random sampling scheme or the Hough transform method are used to find the most consistent solution. But Fischler and Bolles [FB81] and Linnainmaa et al. [LHD88] did not explore how to merge the information of all the n points in an equation system to achieve more accurate results. In [QL99], the P3L polynomials of the subsets are stacked together to form a system of nonlinear equations, and a linearization technology is used to solve the equations, but the local minimum problem of the nonlinear equations is not considered.

The key parameter directly related to the multiple solutions of P3L and the local minima of PnL is the rotation angle α. To illustrate the relationship between the solutions of the P3L polynomial f_i in Eq.(4.48) and the local minima of the cost function $F = \Sigma f_i^2$, let's consider a simplified case which contains only two 3-line subsets with corresponding P3L polynomials f_1 and f_2. Assuming that each f_i has only two local solutions: the solutions of f_1 are (α_1^a, α_1^b), and the solutions of f_2 are (α_2^a, α_2^b). The curves of f_1^2, f_2^2 and $F = \Sigma f_i^2$ are plotted in Fig.4.7. Clearly the local minima of F depends

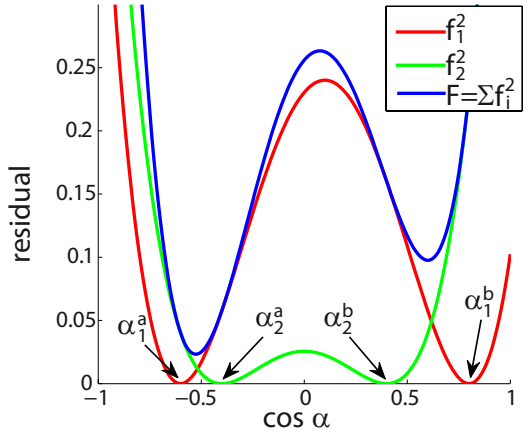


Figure 4.7. A simplified illustration of the local minima of the PnL problem with regard to α. In which f_i denotes the P3L polynomial in Eq.(4.48). The curves of f_i^2 and the cost function $F = \Sigma f_i^2$ are plotted. The local minima of F are highly related to the consistency between the solutions of f_i.

on the solutions of f_i. Moreover, the local minimum of F locating between α_1^a and α_2^a is lower than the other local minimum locating between α_1^b and α_2^b, despite the fact that the individual values of $f_1(\alpha_1^a)$, $f_1(\alpha_1^b)$,$f_2(\alpha_2^a)$, and $f_2(\alpha_2^b)$ are all the same. This is because the distance between α_1^a and α_2^a is smaller, which indicates these two triplets are more agreeable to each other. In practice, we know f_i has at most eight solutions, and $F = \Sigma f_i^2$ has at most eight local minima accordingly (see Remark.4.1). These local minima of F are examined one by one, so that an optimal solution can be ensured as the output of our RPnL method.

4.5 Experiments

In this section, we first compare the proposed algorithm with the state-of-the-art PnL algorithms in terms of the efficiency, accuracy and robustness. Then we test our algorithm on the real image sequences.

4.5.1 Experiments with synthetic data

Given a virtual perspective camera with image size of 640×480 pixels and focal length of 800 pixels, the 3D reference lines are randomly generated in the camera coordinate frame. Different levels of Gaussian noise are added to the projected image lines, and for each noise level 2000 test datasets are generated.

The following methods are compared:

(1) RPnL, the proposed algorithm. (2) RPnL++, the proposed algorithm plus a R_and_T optimization scheme [KH94]. (3) Mirzaei, a non-iterative solution by Mirzaei and Roumeliotis [MR11b]. It is one of the most accurate solutions for PnL so far. The local minima are retrieved by solving a 27×27 multiplication matrix and the global optimum is picked by evaluating the orthogonal errors defined as in Eq.(4.54). The approach is not very efficient because of the high dimensional Macaulay Matrix. (4) Mirzaei++, Mirzaei plus a R_and_T optimization scheme. (5) Ansar, a non-iterative solution by Ansar and Daniilidis [AD03] which converts the polynomial system to a set of linear systems by re-parameterizing the nonlinear terms with new variables. (6) Ansar++, Ansar plus a R_and_T optimization scheme. In our implementation, RPnL++, Mirzaei++ and Ansar++ take the pose solutions which are estimated by RPnL, Mirzaei and Ansar as the initial value respectively, then use the same R_and_T algorithm to iteratively search the optimum.

The following evaluations are conducted:

A. The accuracy: As can be seen in Fig.4.8, RPnL is as accurate as the best state-of-the-art solution. When $n \leqslant 5$, the mean rotation error and mean translation error of RPnL are significantly better than that of existing non-iterative solutions. When $n > 5$, the accuracy of RPnL is similar to that of Mirzaei. The pose estimation error of these non-iterative solutions can be further reduced by employing an iterative optimization scheme such as R_and_T [KH94]. After applying the iterative optimization, Fig.4.8 illustrates that the accuracy of RPnL++ is better than other methods for both $n \leqslant 5$ and $n > 5$.

B. The robustness: Robustness is one of the most significant advantages of RPnL comparing to other methods, which can be seen in Fig.4.9.(a). The correct rates of the compared methods are plotted, which are the ratio

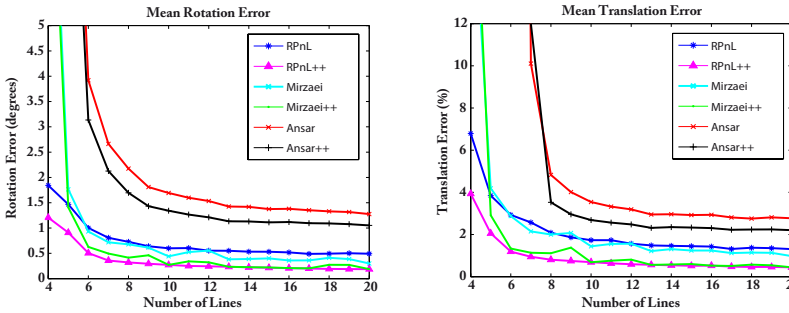
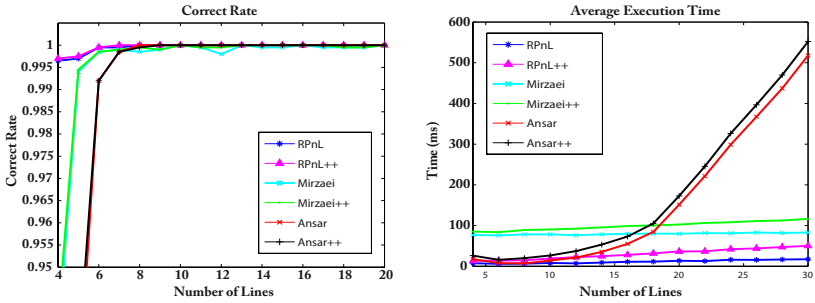


Figure 4.8. The mean rotation and translation errors of the compared methods are plotted as a function of the number of lines. The noise level $\sigma = 5$ pixels.

of the number of correct solutions over 2000 tests (a solution is deemed as a correct result when the deviation of the estimated rotation from the ground truth is less than 30 degrees). When $n \leqslant 5$, the correct rate of our solution is noticeably better than the other methods. When $n > 5$, RPnL is still slightly better. The reason may lie in two facts: RPnL chooses a line with the longest projection in the image plane as rotation axis which is less sensitive to the image noise, and RPnL has lower complexity than Mirzaei which makes RPnL more numerically stable. For the robustness of Ansar, Fig.4.9.(a) shows that when few reference lines are available, the algorithm is very sensitive to the image noise. When the number of reference lines large, the solution of Ansar turns to be stable because the number of linear equations is much bigger than the number of lifting variables.

C. The efficiency: RPnL is highly efficient and its computing time grows linearly with n. As can be seen in Fig.4.9.(b), the average execution times are plotted as a function of the number of lines from 4 to 30. The method was implemented in MATLAB and 2000 runs are performed. The efficiency of RPnL is significantly better than the other methods, and its computing time takes only a fraction of that of Mirzaei which is also linear in the number of correspondences. Fig.4.9.(b) clearly shows that the computational complexity of Ansar is $O(n^2)$.

D. The small line set: RPnL can accurately deal with the small line set.



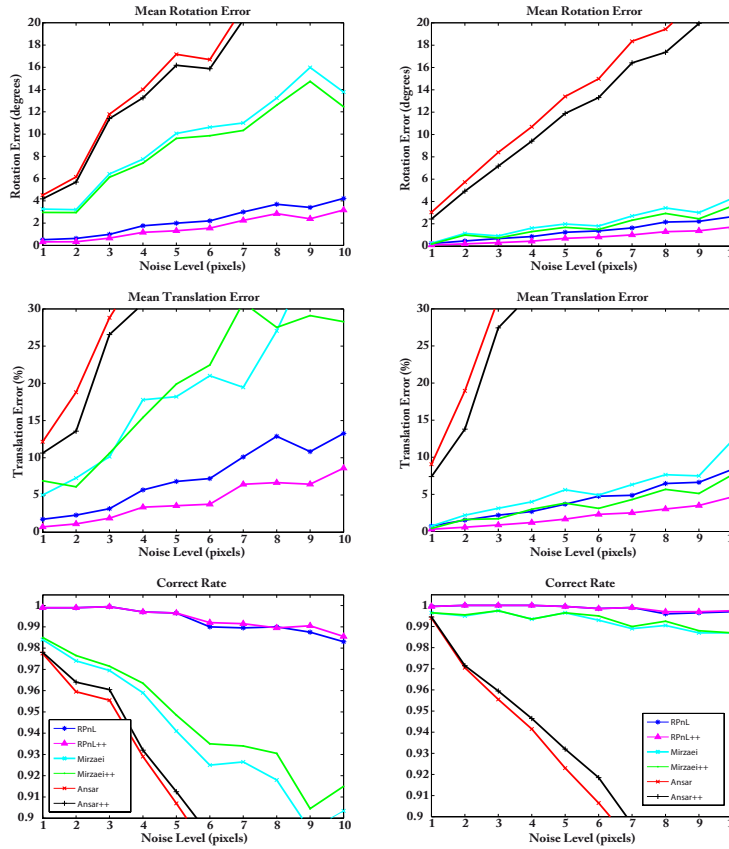
(a) Correct Rate (b) Computing Time

Figure 4.9. The correct rate (a) and computing time (b) of the compared methods are plotted as a function of the number of lines. The noise level $\sigma = 5$ pixels.

As can be seen in Fig.4.10, RPnL can achieve higher accuracy than existing methods for small line sets in the presence of image noise. The correct rates shown in the Fig.4.10 also demonstrate that the existing methods are very sensitive to noise when $n = 4$ or $n = 5$ while RPnL suffers less performance decrease from the increase of the noise level. Here the noise level is measured from the noise added to the endpoints of lines in pixels. The correct rates of RPnL++, Mirzaei++ and Ansar++ reported in Fig.4.10 show that in the absence of a good initialization, the iterative algorithms may converge to a local minimum. The performance of the iterative algorithms mainly depends on the performance of the non-iterative algorithms which are employed to initialize them. However, they can always slightly improve the accuracy and robustness of their initialization approaches in the cost of losing some computational efficiency.

4.5.2 Experiments with real images

In order to compare the PnL solutions in real situations, we apply the algorithms on a set of images with known 3D line model. The 3D model (as shown in the first row of Fig.4.11) is a laboratory reconstructed from images of our office by the structure-from-motion algorithm presented in the next chapter 5. The 3D position of line segments (especially their



(a) $n = 4$ (b) $n = 5$

Figure 4.10. The mean rotation errors, the mean translation errors and the correct rate of the compared methods for $n = 4$ (first column) and $n = 5$ (second column) as a function of the image noise level which varies from 1 to 10 pixels.

endpoints) are not perfectly reconstructed. In this case, the noise exists both in the 3D model and in the image measurement.

For each image, we extract lines using the LSD line detector [GJMR10],

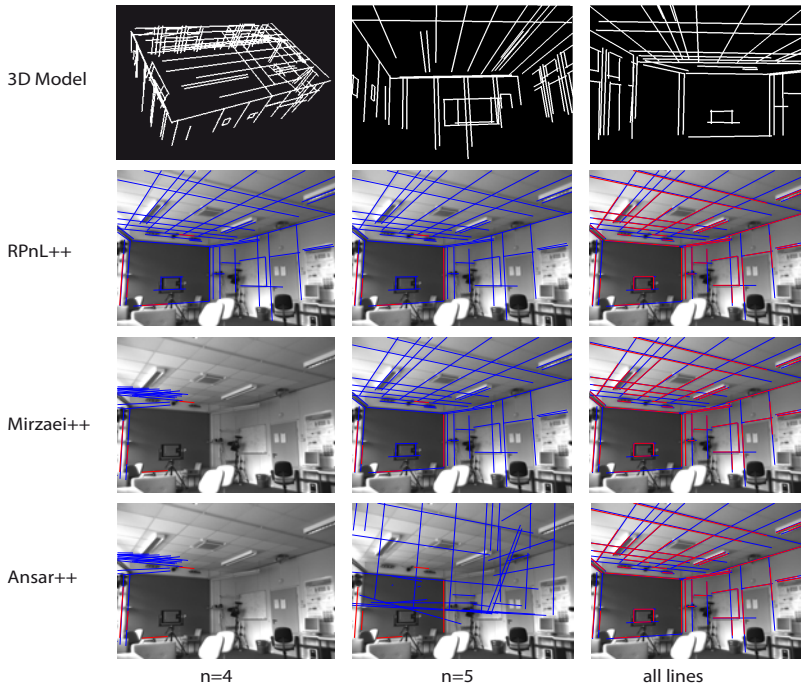


Figure 4.11. The first row shows the 3D line model of the laboratory viewed from different view points. The rest rows show the pose estimation results of RPnL++, Mirzaei++ and Ansar++, respectively. The experimental results illustrate the compared PnL solutions for real images when using 4, 5 or all the available line correspondences. The red lines are the used 2D line segments. The blue lines are the projection of the 3D line model using the estimated camera pose.

then establish the correspondences between the image lines and the 3D line model. We test the compared algorithms by using 4, 5 and all the available line correspondences. In order to demonstrate the accuracy of the results, we reproject the 3D line model into the image by using the estimated camera pose. Fig.4.11 shows the reprojection results on a sampled image from the image set. As shown in Fig.4.11, when using only four pairs of 2D/3D line correspondences, only RPnL++ robustly estimates the camera

pose, i. e., the 3D line model is correctly projected into the image. When using five pairs of correspondences, both RPnL++ and Mirzaei++ estimate the camera pose accurately. If all the available line correspondences are used, then all the compared methods can accurately estimate the camera pose for this sample image. This experiment clearly demonstrates the accuracy and robustness of RPnL++ in real situations.

4.6 Summary and discussion

In this chapter, we first derive an eighth order P3L polynomial for a triplet of lines in general configurations. The solutions of the P3L problem is discussed by investigating the symmetric structure of the line direction. A complete scenario for three spatial lines in all possible configurations is presented which unifies the previous work on the P3L problem for lines in special configurations.

By employing the general eighth order P3L polynomial, the PnL problem is addressed in this chapter. There are many open issues in this field: (1) The small line sets ($3 < n \leqslant 5$) are sensitive to the noise; (2) The computational complexity to discover the global optimum is expensive; (3) It is hard to find a solution that is both accurate and efficient. To solve these issues, a counterpart of RPnP is proposed which is named as RPnL. In the framework of RPnL, lines are separated into triplets by selecting a rotation axis, then a sixteenth order cost function is built from a set of P3L polynomials. The optimum solution of the pose is retrieved from the local minima. Experiment results show that RPnL, although of much lower computational complexity, is as accurate as the state-of-the-art algorithms. For a small number of line correspondences, RPnL achieves higher accuracy than existing non-iterative methods. The implementations of RPnL and RPnL++ are available on our website[1].

RPnL is suitable for applications that need to handle both small and large numbers of line correspondences, such as the line feature based object tracking or camera localization in augmented reality. An example of RPnL application in augmented reality will be presented after introducing the 3D reconstruction algorithm in the next chapter.

[1] http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Lilian+Zhang.

Line is a rich metaphor for the artist. It denotes not only boundary, edge or contour, but is an agent for location, energy, and growth. It is literally movement and change - life itself.

Lance Esplund

Chapter 5

Structure from motion based on 2D/2D line correspondences

After estimating the camera pose from 2D/3D line correspondences, in this chapter, we address the problem of scene reconstruction and camera pose estimation from line correspondences across multiple images, which ranges from the representation of lines, their projections and the initialization procedure to the final adjustment. The Cayley representation of spatial lines is developed, which is a nonlinear minimal parametrization circumventing the tiresome Plücker constraint. Then a novel line projection function is derived which is consistent with previous results. After building the line observation model, we employ a closed-form solution for the first image triplet, then develop an incremental initialization approach to initialize the motion and structure parameters. Finally, the Sparse Bundle Adjustment (SBA) is applied to refine the parameters, which updates the spatial lines by using the Cayley representation with an unconstrained optimization engine.
Large parts of this work have been pre-published in [ZK11; ZK12b]

5.1 Introduction

Structure and motion from multiple images is a classical and recurrent topic in computer vision. It has received a lot of attention and a number of algorithms have emerged in application areas such as scene modeling, augmented reality and robot navigation. This chapter addresses the problem of estimating camera motion and recovering the three-dimensional structure of a scene composed of straight line segments from a set of line correspondences across multiple views. This is useful, as line features are prominent in most man-made environments, and a map of line seg-

ments gives a higher level of relevant information on the structure of the environment than point features.

Unlike point primitives with a simple representation and projection rule, there are two intrinsic difficulties when dealing with lines. Firstly, there is no global linear and minimal parametrization for 3D lines representing their four degrees of freedom by four parameters [HZ04]. Secondly, depending on the representation, it may be non-trivial to project a 3D line into the image plane [BS04].

To circumvent the tiresome Plücker constraint as discussed in Sec.8.2 of [SVCCM12], the Cayley representation of lines is defined in this chapter. The transformation between the Cayley representation and the Plücker coordinates representation is trivial. Besides, we give a novel derivation of the line projection function using the relationship between Plücker coordinates and the Plücker matrix [ZK11], which is consistent with the function derived from the dual relationship between points and lines. The resulting projection function is linear in terms of the Plücker coordinates. However, it is not well-adapted for nonlinear optimization because the Plücker coordinates have two degrees of internal gauge freedoms and are subject to the Plücker constraints. Instead, we adopt the Cayley representation to update line parameters during the optimization procedure to achieve an efficient nonlinear optimization approach with an unconstrained optimization engine.

For the scene reconstruction and camera motion estimation, we propose an initialization approach to boot the nonlinear optimization procedure by starting from the image triplet reconstruction [WHA92] and incrementally adding new views and new lines. Then we employ the Sparse Bundle Adjustment (SBA) for nonlinear estimation. The SBA algorithm implementation [LA09] is originally designed for point feature reconstruction. We redesign the SBA library to make it appropriate for line parameter adjustment.

To summarize, this chapter addresses the problem of structure and motion from line correspondences. More formally, we consider a situation in which a set of 3D line features $\{L_j\}$ is viewed by a set of cameras $\{\mathrm{Y}^i\}$. Denote by $\boldsymbol{\ell}_j^i$ the observation of the j-th 3D line as seen by the i-th camera. Lines may be viewed only in a subset of cameras. We intend to address the following reconstruction problem: given the set of image observations

$\{\boldsymbol{\ell}_j^i\}$, find the metric reconstruction of the 3D lines $\{L_j\}$ and the pose of cameras $\{\mathrm{Y}^i\}$ which best suit the line observation model. We focus on the metric reconstruction with known intrinsic camera parameters. Like the previous work [TK95; BS05; MTM00], the correspondences of line segments across multiple views are given in advance to allow a fair comparison of the efficiency and accuracy of the optimization approaches. For matching the line segments, there is a great amount of work on this topic [WNY09; WWH09; FWH10; ZK12a]. One can employ any of them to generate the correspondences.

Compared to the traditional multi-view algorithm (such as [TK95; BS05]), our contributions are as follows: (i) The Cayley representation of 3D lines is introduced in Sec.5.3 which can be updated without Plücker constraints. (ii) A novel line projection function is derived in Sec.5.4 for general camera models. (iii) A robust initialization approach is proposed in Sec.5.5 to boot the nonlinear optimization procedure. (iv) The sparse bundle adjustment algorithm is employed in Sec.5.6 which improves the speed significantly. We validate our algorithm and compare it to the existing work to demonstrate its efficiency and accuracy in Sec.5.7. We also build a real-time augmented reality system by combining algorithms proposed in the previous chapters and in this chapter. Finally, the conclusion is drawn in Sec.5.8.

5.2 Related work

This section briefly reviews the related work on the line representation, the line projection function and the reconstruction procedure.

5.2.1 The 3D line representation

The previous work is divided into two groups: the nonlinear minimal 4-parameter representation, and the linear over-parametrization representation. For the first group, Roberts [Rob88] used two direction cosines together with the 2D coordinates to represent the direction and position of a line. Ohwovoriole [Ohw80] represented a line as the intersection of two planes which are parallel to the X-axis and the Y-axis respectively. For

the linear over-parametrization group, Smith et al. [SRD06], and Andrew and Walterio [AW06] represented a line by its two endpoints. Hartley and Zisserman [HZ04] introduced two dual representations: a pair of points and a pair of planes. Montiel et al. [MTM00], and Eade and Drummond [ED06] used the midpoint and the direction of the segment to represent a 3D line. Weng et al. [WHA92] represented a 3D line by its closest point to the origin and its direction. Pottmann et al. [PHOW04], and Seo and Hong [SH96] used the Plücker coordinates representation, which includes the moments of lines to the origin and the directions of lines in space.

Hartley and Zisserman [HZ04] and Bartoli and Sturm [BS05] gave a summary of the 3D line representation. The nonlinear minimal representations use only four parameters which are equal to the degrees of freedom of a 3D line, hence there is no internal gauge freedom nor consistency constraint, which makes them well-adapted to the nonlinear optimization. However, the non-linearity makes it difficult to explicitly express the line projection function. Contrarily, those linear over-parametrization representations can be easily expressed in the projection function, but with internal gauge freedom which may induce numerical instabilities. Taylor and Kriegman [TK95] used the closest point and the direction (in total six parameters) to build the projection function, while using two direction cosines and 2D coordinates (in total four parameters) to update during the nonlinear optimization. Bartoli and Sturm [BS05] used the Plücker coordinates to build the projection function and exploited the orthonormal representation, which includes a 2×2 and a 3×3 orthogonal matrix to update during bundle adjustment.

5.2.2 The line projection function

The projection function of lines is dependent on the representation of lines. The simplest one is the same as the point projection function for those representing lines by their endpoints [MTM00; ED06; HML02]. Suffering from the problem of unstable detection of endpoints, the triangulation of two endpoint pairs may not correspond to the correct spatial line. For calibrated cameras, Weng et al. [WHA92], Taylor and Kriegmann [TK95], Goddard [God97], and Schindler et al. [SKD06] derived a line projection function from the rigid transformation in metric space. Faugeras and

Mourrain [FM95], Martinec and Pajdla [MP03], and Bartoli and Sturm [BS04] derived a 3×6 projection matrix for the Plücker coordinates under the perspective camera model. This linear projection function is similar to the point projection function but with higher dimension, which means it is computationally more costly. Hartley and Zisserman [HZ04] introduced a line projection function for the Plücker matrix representation, which is in quadratic form with respect to the 3×4 camera projective matrix.

In this chapter, we present a novel derivation of the line projection function based on the relationship between the Plücker coordinates and the Plücker matrix. We will show that it is consistent with another derivation, which is based on the dual relationship between points and lines.

5.2.3 The reconstruction procedure

For the problem of camera motion estimation and scene structure recovering from line correspondences, the existing approaches are subject to three categories: linear solutions, the Extended Kalman Filter (EKF) based methods, and nonlinear optimization. Given a set of at least thirteen line segments viewed in three frames, it is possible to determine the motion and structure. Liu and Huang [LH88a] and Spetsakis and Aloimonos [SA90] developed a linear algorithm for this problem based on these observations. Weng et al. [WHA92] presented a closed-form solution and established the uniqueness of the solution. The above approaches only work on three views. By using matrix factorization, Triggs [Tri96], Morris and Kanade [MK98], and Martinec and Pajdla [MP03] presented an improved linear algorithm without the limitation of only three views but requiring all lines to be visible in all views. Nevertheless, they can give a good initial guess for those iterative methods. Several approaches based on the EKF were proposed in the literature [SH96; VF90; CSSP92]. These EKF based algorithms suffer from the drawback of the EKF itself, which limits the number of update parameters. Nonlinear algorithms are almost always the last step to yield reliable results, especially in the presence of noise. Yen and Huang [YH83], and Liu and Huang [LH88b] iteratively solved a set of nonlinear equations for the motion parameters for three views. Montiel et al. [MTM00] described a nonlinear approach which considers any number of images greater than or equal to two.

The two most closely related papers using bundle adjustment are [TK95] and [BS05]. Taylor and Kriegman [TK95] initialized the algorithm from the partially known camera rotation parameters, otherwise, they booted their nonlinear algorithm by sampling the subset of the parameter space, which is computationally expensive and does not guarantee the correct convergence of their algorithm. Bartoli and Sturm [BS05] first used a linear algorithm to reconstruct lines associated with each triplet of consecutive images and then registered these triplets using the method proposed in [BS04]. The alignment of two groups of 3D lines is not robust when the extracted lines are contaminated by image noise in the two triplets. In contrast, we develop an incremental initialization procedure to boot the SBA algorithm, which is also suitable for online reconstruction.

There is another group of work in the literature, which reconstruct the 3D lines from the line drawings [LCT11; XLT12]. Generally, the 3D reconstruction from line drawings uses clean user input of a single view. It assumes that all edges of the planar faced object are visible and edges are connected at end points to form clear wire frame of the object. In contrast to that, the line feature based reconstruction assumes a noisy line detection result from multiple images, and lines may be visible only in parts of images. In [LCT11], heuristic rules summarized from the human visual perception were used to construct an objective function, the 3D object was reconstructed by minimizing this function. This work was extended by Xue et al. [XLT12], who first used the decomposition method proposed in [LCT11] to separate a complex line drawing into multiple simple ones, then generated a set of candidate 3D models from examples of some basic 3D models. At last, the candidate models were evaluated and combined together. This algorithm will be compared to ours in the experiment section.

5.3 The Cayley representation of 3D lines

In this section, we define the Cayley representation of spatial lines. A brief introduction of the related representations (the Plücker Coordinates and the Plücker Matrix) can be found in Sec.2.4.

There are two constraints (Eq.2.27) of the Plücker coordinates repre-

sentation of lines. It is not convenient to enforce these constraints in the optimization process. In [SVCCM12], the Plücker constraints are guaranteed at initialization time but they are not enforced further during the landmark updates. In order to automatically guarantee that the parameters of lines satisfy the Plücker constraints, the Cayley representation of spatial lines is developed based on the observation, that an orthogonal matrix can be easily constructed from the Plücker coordinates $(\boldsymbol{m}, \boldsymbol{v})$ of a line under the Plücker constraints as:

$$\boldsymbol{Q} = \left[\boldsymbol{v}, \ \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}, \ \frac{\boldsymbol{v} \times \boldsymbol{m}}{\|\boldsymbol{v} \times \boldsymbol{m}\|} \right]. \tag{5.1}$$

In Sec.2.2.3, the Cayley-Gibbs-Rodrigues representation of rotation matrices is introduced. Similar to Eq.(2.15), a skew symmetric matrix formed by the vector $\boldsymbol{s} = [s_x, s_y, s_z]^T$ can be computed from the orthogonal matrix $\boldsymbol{Q}$ and the identity matrix $\boldsymbol{I}$ as:

$$[\boldsymbol{s}]_\times = (\boldsymbol{Q} - \boldsymbol{I})(\boldsymbol{Q} + \boldsymbol{I})^{-1}. \tag{5.2}$$

Letting $\omega = \|\boldsymbol{m}\|$, the Cayley representation of a spatial line is defined as a 4-dimensional vector $\boldsymbol{\zeta} = (\omega, \boldsymbol{s})$. The relationship between the Cayley representation and the Plücker coordinates is simple and clear. Eq.(5.1) and Eq.(5.2) give the method to compute the Cayley representation of a line from the Plücker coordinates. As given in Eq.(2.14), transferring the Cayley representation to the Plücker coordinates is trivial:

$$\begin{aligned} \boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3] &= (\boldsymbol{I} - [\boldsymbol{s}]_\times)^{-1}(\boldsymbol{I} + [\boldsymbol{s}]_\times) \\ &= \frac{(1 - \|\boldsymbol{s}\|^2)\boldsymbol{I} + 2[\boldsymbol{s}]_\times + 2\boldsymbol{s}\boldsymbol{s}^T}{1 + \|\boldsymbol{s}\|^2}. \end{aligned} \tag{5.3}$$

According to the definition of $\boldsymbol{Q}$ in Eq.(5.1), we have

$$\boldsymbol{v} = \boldsymbol{q}_1, \quad \boldsymbol{m} = \omega \boldsymbol{q}_2. \tag{5.4}$$

Fig.5.1 gives the geometric interpretation of parameters in the Cayley representation. For the spatial line L, its direction vector $\boldsymbol{v}$, its moment vector $\boldsymbol{m}$ and their cross product $\boldsymbol{v} \times \boldsymbol{m}$ define the object coordinate frame

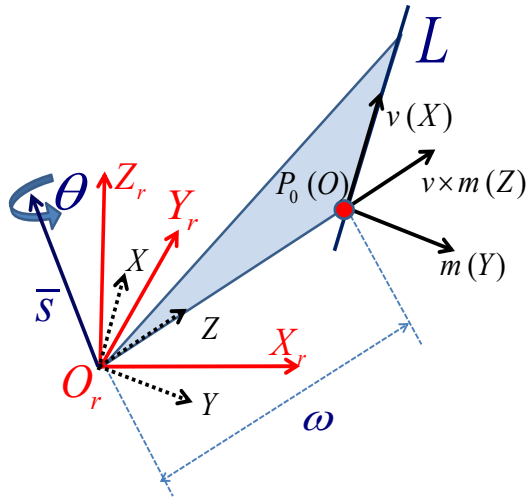


Figure 5.1. Geometric interpretation of the Cayley representation of a line. ω is the distance from the origin of the reference frame $O_rX_rY_rZ_r$ to the spatial line L. The rotation between $O_rX_rY_rZ_r$ and the object frame $OXYZ$ is determined by a rotation angle $\theta = 2\arctan\|\boldsymbol{s}\|$ around the rotation axis $\bar{\boldsymbol{s}} = \boldsymbol{s}/\|\boldsymbol{s}\|$.

$OXYZ$ at the point P_0, which is the closest point on the line L to the center of the reference frame $O_rX_rY_rZ_r$ (e. g., the world frame or the camera frame). Then ω in the Cayley representation is the distance between O_r and P_0, and the vector $\boldsymbol{s}$ in the Cayley representation encodes the rotation information as illustrated in Fig.5.1.

As pointed out by Roberts [Rob88], most of the 4-parameter nonlinear representations of lines have some singularities. For the Cayley representation, it is problematic when $\|\boldsymbol{m}\| = 0$ because $\boldsymbol{Q}$ defined in Eq.(5.1) degenerates to $[\boldsymbol{v}, \boldsymbol{0}_3, \boldsymbol{0}_3]$, which is not an orthogonal matrix anymore. Fortunately, in this case, only the direction vector $\boldsymbol{v}$ plays the role to define the spatial line. It is easy to construct an orthogonal matrix from $\boldsymbol{v}$, for example, $\boldsymbol{Q} = [\boldsymbol{v}, \boldsymbol{e}_1, \boldsymbol{e}_2]$ where $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are one pair of the orthogonal base of the null space formed by the linear system $\boldsymbol{v}^T\boldsymbol{x} = 0$. Hence, the singularity is solved easily by modifying the definition of the Cayley representation

$\boldsymbol{\zeta} = (\omega, \boldsymbol{s})$ as:

$$\boldsymbol{Q} = \begin{cases} [\boldsymbol{v},\ \boldsymbol{e}_1,\ \boldsymbol{e}_2] & if\ \ \|\boldsymbol{m}\| \leqslant \tau; \\ \left[\boldsymbol{v},\ \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|},\ \frac{\boldsymbol{v}\times\boldsymbol{m}}{\|\boldsymbol{v}\times\boldsymbol{m}\|}\right] & else. \end{cases} \tag{5.5}$$

$$\omega = \|\boldsymbol{m}\|, \quad [\boldsymbol{s}]_\times = (\boldsymbol{Q} - \boldsymbol{I})(\boldsymbol{Q} + \boldsymbol{I})^{-1},$$

where τ is a small number close to zero (in our implementation, $\tau = 10^{-7}$). Transferring back from the modified Cayley representation to the Plücker coordinates is the same as Eq.(5.4) because when $\omega = \|\boldsymbol{m}\| \leqslant \tau$, we have $\boldsymbol{m} = \omega \boldsymbol{q}_2 \simeq \boldsymbol{0}_3$.

It is worth to emphasize the following three remarks:

• If $\|\boldsymbol{m}\| \leqslant \tau$, then the spatial line passes through a point which is very close to the origin of the reference frame. Specially, in the camera frame, if a spatial line passes through the camera center, then its projection in the image plane will degenerate to a 2D point.

• Singularity is widely existing in the nonlinear parametrization of spatial lines. In our work, we solve this problem gently by keeping the consistency of the Cayley representation for both situations: $\|\boldsymbol{m}\| \leqslant \tau$ and $\|\boldsymbol{m}\| > \tau$.

• The dimension of the Cayley representation is four which equals to the degrees of freedom of a 3D line. Hence there is no internal gauge freedom. It is suitable for updating spatial lines in the SBA process. By contrast, the dimension of the Plücker coordinates is six, so there are two internal gauge freedoms. However, the Plücker coordinates representation is a linear parametrization of a 3D line, which is suitable for deriving the linear line projection function and for initialization as presented in the sequel.

5.4 Line projection function

This section presents two derivations of the line projection function: one is based on the dual relationship between points and lines, while the other is based on the relationship between the Plücker coordinates and the Plücker matrix. Then the line observation model is built, which is used to compute

the error function in the initialization process and the SBA process.

5.4.1 First derivation of the line projection function

Supposing points and lines in the image plane are represented by homogeneous coordinates [HZ04], then the dual relationship between points and lines can be described as: (1) the line through two image points $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{b}}$ is $\boldsymbol{\ell} = \bar{\boldsymbol{a}} \times \bar{\boldsymbol{b}}$; (2) the intersection of two image lines $\boldsymbol{\ell}$ and $\boldsymbol{\ell'}$ is the point $\bar{\boldsymbol{a}} = \boldsymbol{\ell} \times \boldsymbol{\ell'}$. We can employ this dual relationship to derive the line projection function from the point projection function.

As introduced in Sec.2.3.1, the camera matrix is $\boldsymbol{P} = \boldsymbol{K}\boldsymbol{R}[\boldsymbol{I} \mid -\boldsymbol{c}]$, where $\boldsymbol{K}$ is the camera calibration matrix, $\boldsymbol{R}$ and $\boldsymbol{c}$ represent the camera pose. Two homogeneous 3D points on the spatial line are $A^T = (\boldsymbol{a}^T|a)$ and $B^T = (\boldsymbol{b}^T|b)$. Their projections on the image plane are two homogeneous 2D points $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{b}}$. Then the projection of the spatial line on the image plane should pass through these two points. Hence, we have

$$
\begin{aligned}
\boldsymbol{\ell} &\sim \bar{\boldsymbol{a}} \times \bar{\boldsymbol{b}} \sim (\boldsymbol{P}A) \times (\boldsymbol{P}B) \\
&= (\boldsymbol{K}\boldsymbol{R}\boldsymbol{a} - a\boldsymbol{K}\boldsymbol{R}\boldsymbol{c}) \times (\boldsymbol{K}\boldsymbol{R}\boldsymbol{b} - b\boldsymbol{K}\boldsymbol{R}\boldsymbol{c}) \\
&= (\boldsymbol{K}\boldsymbol{R}\boldsymbol{a}) \times (\boldsymbol{K}\boldsymbol{R}\boldsymbol{b}) - (\boldsymbol{K}\boldsymbol{R}\boldsymbol{c}) \times (a\boldsymbol{K}\boldsymbol{R}\boldsymbol{b} - b\boldsymbol{K}\boldsymbol{R}\boldsymbol{a}) \\
&= cof(\boldsymbol{K}\boldsymbol{R}) \left[(\boldsymbol{a} \times \boldsymbol{b}) - \boldsymbol{c} \times (a\boldsymbol{b} - b\boldsymbol{a}) \right] \\
&= cof(\boldsymbol{K}\boldsymbol{R}) \left[\boldsymbol{m'} - \boldsymbol{c} \times \boldsymbol{v'} \right] \\
&\sim cof(\boldsymbol{K}\boldsymbol{R}) \left[\boldsymbol{m} - \boldsymbol{c} \times \boldsymbol{v} \right] \\
&= cof(\boldsymbol{K})\ \boldsymbol{R} \left[\boldsymbol{m} - \boldsymbol{c} \times \boldsymbol{v} \right], \qquad (5.6)
\end{aligned}
$$

where $cof()$ is the cofactor matrix and $\sim$ means two sides are equal up to scale. The fourth row is based on Lemma 2.1 presented in Sec.2.2.2. The fifth row is based on the definition of $\boldsymbol{m'}$ and $\boldsymbol{l'}$ in Eq.(2.28).

5.4.2 Second derivation of the line projection function

In this section, we give the second method to derive the line projection function based on the relationship between the Plücker coordinates and the Plücker matrix (cf. Sec.2.4). In [HZ04], a 3D line represented by a

Plücker matrix $\boldsymbol{L}$ is imaged as the line $\boldsymbol{\ell}$:

$$[\boldsymbol{\ell}]_\times \sim \boldsymbol{PLP}^T. \tag{5.7}$$

By substituting the relationship between the Plücker coordinates and the Plücker matrix in Eq.(2.33) into the Plücker matrix mapping equation (5.7), we have

$$\begin{aligned}
[\boldsymbol{\ell}]_\times &\sim \boldsymbol{KR}[\boldsymbol{I} \mid -\boldsymbol{c}] \begin{bmatrix} [\boldsymbol{m}]_\times & \boldsymbol{v} \\ -\boldsymbol{v}^T & 0 \end{bmatrix} (\boldsymbol{KR}[\boldsymbol{I} \mid -\boldsymbol{c}])^T \\
&= \boldsymbol{KR}\left[\boldsymbol{m} - \boldsymbol{c} \times \boldsymbol{v}\right]_\times \boldsymbol{R}^T \boldsymbol{K}^T \\
&= \boldsymbol{K}\left[\boldsymbol{R}\left(\boldsymbol{m} - \boldsymbol{c} \times \boldsymbol{v}\right)\right]_\times \boldsymbol{K}^T \\
&= \left[cof(\boldsymbol{K})\ \boldsymbol{R}\ \left(\boldsymbol{m} - \boldsymbol{c} \times \boldsymbol{v}\right)\right]_\times .
\end{aligned} \tag{5.8}$$

The third row is based on Eq.(2.6) and the last row is based on Eq.(2.5) in Lemma 2.2. Finally, we get the Plücker coordinates' mapping equation, which projects a 3D line into the image plane as:

$$\boldsymbol{\ell} \sim cof(\boldsymbol{K})\ \boldsymbol{R}\ (\boldsymbol{m} - \boldsymbol{c} \times \boldsymbol{v}). \tag{5.9}$$

If the camera calibration matrix is in the following form:

$$\boldsymbol{K} = \begin{bmatrix} f & 0 & \mu_0 \\ 0 & \alpha f & \nu_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.10}$$

where f is the focal length of the camera, α is the aspect ratio and (μ_0, ν_0) are the coordinates of the principal point in the image plane, then

$$cof(\boldsymbol{K}) = \begin{bmatrix} \alpha f & 0 & 0 \\ 0 & f & 0 \\ -\alpha f \mu_0 & -f\nu_0 & \alpha f^2 \end{bmatrix}. \tag{5.11}$$

Following the definition of the line moment $\boldsymbol{m} = P \times \boldsymbol{v}$, the line projection function can be further simplified as:

$$\boldsymbol{\ell} \sim cof(\boldsymbol{K})\ \boldsymbol{R}\ \left((P - \boldsymbol{c}) \times \boldsymbol{v}\right), \tag{5.12}$$

which is similar to the point projection function $\boldsymbol{x} = \boldsymbol{KR}[I \mid -c]X$.

Note that Eq.(5.6) and Eq.(5.8) have exactly the same results. These two derivations give more insight into the line projection function and its relationship with the point projection function. Compared to other line projection functions, the Plücker coordinates projection function (Eq.5.9) is the most compact representation to project a spatial line into the image plane under the perspective camera model. Goddard [God97], and Taylor and Kriegman [TK95] gave a similar line projection function derived from the rigid transformation, however, their camera model only takes the focal length as its internal parameter and our model generalizes theirs. Compared to the Plücker matrix mapping (Eq.5.7) or the line projection function introduced by Bartoli and Sturm [BS05], although they are mathematically equivalent with ours, they are obviously computationally more expensive because of their higher degree and dimension of the camera matrix $\boldsymbol{P}$.

5.4.3 Line observation model

Unlike point features whose error function simply is the distance between the observed location and the projected location in the image plane, only the measurement components that are orthogonal to the expected line projection can be used for correction because of the aperture problem [SVCCM12].

Since an image line is restricted to the image plane, only two degrees of freedom are present for each observed line segment. In [ZK11; God97], the closest point on the line segment to the image origin is chosen as observation, which is named line-point as shown in Fig.5.2(a). The error function is defined as the distance between the observed and the predicted line-points ($|\boldsymbol{p}_c\boldsymbol{p}'_c|$). This form is similar to the error function of point features. However, there is a singularity near the origin of the image plane because many distinct lines can go through the origin and this model considers them all identical. This is a potential source of error. In this work, we therefore choose the distances between the endpoints of the observed segment to the projected line as observation. We first use the line projection function (5.9) to predict the image line $\boldsymbol{\ell} = (l_x, l_y, l_z)^T$, then

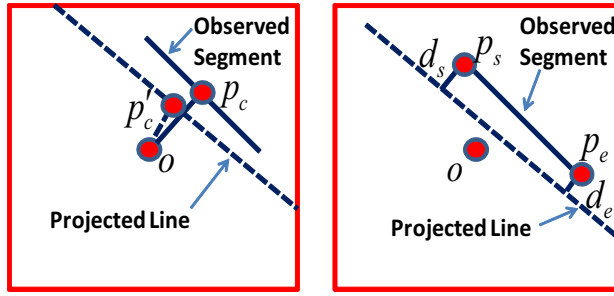


Figure 5.2. The illustration of line observation models: (a) the line-point model; (b) the endpoint distance model.

compute the distances (d_s, d_e) as:

$$d_s = \frac{\boldsymbol{\ell}^T \cdot \boldsymbol{p_s}}{\sqrt{l_x^2 + l_y^2}}, \quad d_e = \frac{\boldsymbol{\ell}^T \cdot \boldsymbol{p_e}}{\sqrt{l_x^2 + l_y^2}}, \tag{5.13}$$

where $\boldsymbol{p_s}$ and $\boldsymbol{p_e}$ are the endpoints of the image line segment in homogeneous coordinates.

Combine equations (5.9) and (5.13) to obtain our line observation model which is used in the initialization approach and the SBA process. The estimation error ε of the reconstruction result can be computed as:

$$\varepsilon^2 = d_s^2 + d_e^2. \tag{5.14}$$

This error function is close to the definition in Eq.(4.55) without explicitly weighting the contributions of segments by their lengths. We choose this formulation because it is easier to implement in the framework of the SBA library. The weights of segments can be embodied in the covariance of the measurement. If the length of a line segment in the image is shorter than a threshold t_l (in our implementation, $t_l = 15$ pixels), then the observation of the line in this image will be discarded, i. e., its weight equals zero.

5.5 Initialization

After introducing the line observation model, in this section we present our approach to obtain the initial values of the motion and structure parameters. We first employ the closed-form algorithm to solve the reconstruction problem from an image triplet, then add new views and new line features frame by frame.

5.5.1 Closed-form solution for an image triplet

Weng et al. [WHA92] presented a closed-form solution for the problem of three-view reconstruction based on line correspondences. They first estimated a set of intermediate parameters and then recovered the motion parameters from them. After estimating the camera motions, they reconstructed the structure of the scene in metric space.

Supposing the coordinate system being fixed on the first camera, and the poses of the second and the third cameras being $[\boldsymbol{R}|\ \boldsymbol{c}]$ and $[\boldsymbol{R'}|\ \boldsymbol{c'}]$ respectively, three intermediate matrices are defined as

$$(\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{G}) : \begin{cases} \boldsymbol{E} = \boldsymbol{r}_1 * \mathbf{c'}^T - \boldsymbol{c} * \mathbf{r'}_1^T \\ \boldsymbol{F} = \boldsymbol{r}_2 * \mathbf{c'}^T - \boldsymbol{c} * \mathbf{r'}_2^T \\ \boldsymbol{G} = \boldsymbol{r}_3 * \mathbf{c'}^T - \boldsymbol{c} * \mathbf{r'}_3^T \end{cases}, \tag{5.15}$$

where $\boldsymbol{r}_i$ and $\boldsymbol{r'}_i$ are the i-th columns of rotation matrices $\boldsymbol{R}$ and $\boldsymbol{R'}$ respectively, $\boldsymbol{c}$ and $\boldsymbol{c'}$ are the translations of the second and the third cameras with respect to the first camera.

The three intermediate matrices are exactly the trifocal tensor when the camera calibration matrix is identity [HZ04]. Hence it is possible to use the trifocal tensor computation algorithms to estimate them. Specially we can carry out the Hartley normalization and denormalization approach before and after solving the linear system, as highly recommended by Hartley and Zisserman [HZ04]. After estimating the intermediate parameters, we first recover the translation vectors, which is similar to retrieve the epipoles from the trifocal tensor [HZ04]. Then we determine the camera rotations and reconstruct the scene structure as in [WHA92]. The uniqueness of the solution is decided by assuming that most of the closest points on lines to

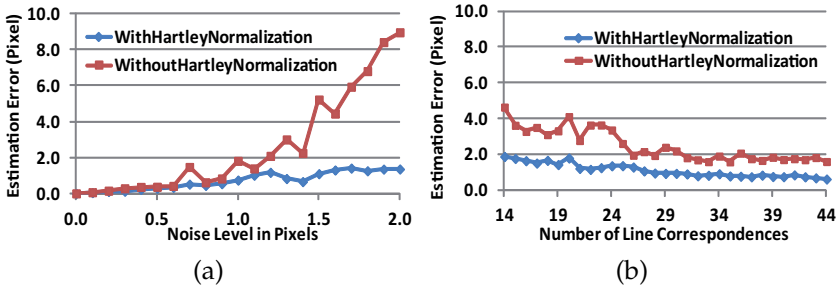


Figure 5.3. The estimation error of the closed-form solution of Weng et al. [WHA92]'s method with respect to the image noise level (a) and the number of line correspondences (b). The red square line shows the result without the Hartley normalization to the input data. The blue diamond line shows the result with Hartley normalization.

the origin are in front of the cameras. Cf. [WHA92] for the details of this algorithm.

We compare the closed-form solution with Hartley normalization and without Hartley normalization. In the first test, we fix the number of line correspondences to 44 triplets and vary the noise level of image measurements from 0 to 2 pixels, then we get the reconstruction result. At last, we evaluate the estimation error ε of the reconstruction result by Eq.(5.14). In the second test we fix the noise level to one pixel and vary the number of line correspondences from 14 to 44 triplets, then compute the estimation error. We run each test 2000 times to cover its statistical nature. Fig.5.3(a) shows that when the noise level is quite small, both methods achieve similar reconstruction accuracy, but if the noise level is larger than one pixel, then the method with Hartley normalization outperforms the previous one. Fig.5.3(b) further verifies that the one with Hartley normalization achieves a lower estimation error.

Once the closed-form solution of an image triplet is obtained, then the metric coordinate system is established and the rest of the views can be incrementally added. For each new view, we first estimate its pose through a set of 2D/3D line correspondences, and then find new line

features which can be reconstructed by triangulation. The next subsection presents the detail of these two steps.

5.5.2 Initialization of new frames and new lines

From the 3D reconstruction of the first three images, a set of 2D/3D line correspondences for images in the triplet is established. Now, for a new frame, we can generate the 2D/3D line correspondences by transferring the correspondences in the previous frame to the current frame according to the line segment matching results between two frames. After establishing the 2D/3D line correspondences in the current frame, the RPnL algorithm proposed in Chapter 4 is employed to estimate the camera pose of the current view.

Now we discuss the estimation of 3D lines given a set of views with known camera poses. We linearly estimate the spatial line by triangulating two planes: for a pair of corresponding image lines in two views, we first back-project lines to get two projection planes Π_1 and Π_2; second, according to the definition of the dual Plücker matrix in Eq.(2.30) we get $\boldsymbol{L}^*$; then according to the relationship between the Plücker matrix and the dual Plücker matrix (Eq.(2.31)) we get $\boldsymbol{L}$; at last, according to the relationship between the Plücker matrix and Plücker coordinates (Eq.(2.33)), we get the initial estimation of the corresponding spatial line $L = (\boldsymbol{m}, \boldsymbol{v})$. If a line is observed in more than two views, then the quasi-linear triangulation algorithm proposed in [BS05] can be employed.

5.6 Sparse bundle adjustment

Following the initial estimation of camera motion and scene structures, bundle adjustment is almost invariably used as the last step of the feature-based 3D reconstruction algorithm. The conventional bundle adjustment algorithms based on Levenberg-Marquardt (LM) or some other nonlinear least-square algorithms solve the normal equations repeatedly with complexity $O(n^3)$ in the number of unknown parameters for each iteration. However, in solving the feature-based 3D reconstruction problems, the normal equation matrix has a certain sparse block structure that one may

take advantage of to achieve very great time savings [HZ04]. Along the line of the presentation regarding sparse bundle adjustment, Lourakis and Argyros [LA09] implemented a SBA algorithm for the point feature reconstruction problem.

In this chapter, we employ their implementation with a few adjustments for our line feature-based reconstruction problem. The camera motions are parameterized by their rotation matrices and translation vectors with respect to the first view as $\{\mathrm{Y}^i = (\boldsymbol{R}^i, \boldsymbol{c}^i)\}$. The 3D lines are represented by their Plücker coordinates $\{L_j = (\boldsymbol{m}_j, \boldsymbol{v}_j)\}$ and updated by the 4-vector Cayley representation $\boldsymbol{\zeta} = (\omega, \boldsymbol{s})$ as defined in Sec.5.3. The image measurement of a line is represented by two endpoints of the segment as $(\boldsymbol{p}_{s_j}^i, \boldsymbol{p}_{e_j}^i)$. The SBA algorithm minimizes the reprojection error of all the 3D line and camera parameters, specifically

$$\min_{\mathrm{Y}^i, L_j} \sum_i \sum_j d_s^2(o(\mathrm{Y}^i, L_j),\ \boldsymbol{p}_{s_j}^i) + d_e^2(o(\mathrm{Y}^i, L_j),\ \boldsymbol{p}_{e_j}^i), \tag{5.16}$$

where $o(\mathrm{Y}^i, L_j)$ is the line projection model (Eq.(5.9)), which predicts the projection of the j-th line in the i-th image, and $d_s()$ and $d_e()$ are the distances of the measured endpoints to the predicted line as defined in Eq.(5.13).

As mentioned in the introduction, the update of line parameters is more complicated than the update of point parameters, so we change the interface of the SBA library and add a pointer to the update function, accordingly adjusting some part of the library when necessary. The projection function and the Jacobian function are also implemented as required. In Appendix A.2, we present the detail of the computation of Jacobian matrices and the update rule. To make it clear, we summarize the whole procedure of our reconstruction approach in Alg.(2).

Note that the twelfth step is adopted to avoid the incremental initialization process getting worse. If the view angle between the i-th view and the k-th view is larger than some threshold, then we apply SBA to refine those parameters which have already been initialized. This approach generally prevents the initialization process from getting divergent. The threshold here is used to adjust the frequency of SBA running. If it is set to zero then SBA will be applied after each frame, while if it is set to a value larger than

Algorithm 2: Structure from motion based on 2D/2D line correspondences

Require: A set of line observations $\{\boldsymbol{\ell}_j^i\}$ with known correspondences and camera intrinsic parameters.

For the first three views, use the closed-form solution algorithm (Sec.5.5.1) to get the initial estimate of the camera motion and 3D lines which are viewed in all three views;
Refine the closed-form solution by SBA;
Set the view index $k = 3$ to record the last time to call SBA
for each new view Y^i **do**
  Find lines which have already been reconstructed and observed in the current view, then call the RPnL (Sec.4.4) to estimate the i-th view pose $\mathrm{Y}^i = (\boldsymbol{R}^i, \boldsymbol{c}^i)$;
  for each new line L_j in the current view **do**
    if L_j is observed in two views with known pose **then**
      Reconstruct L_j using the two-view triangulation approach (Sec.5.5.2) to get the initial estimate $L_j = (\boldsymbol{m}_j, \boldsymbol{v}_j)$
    end if
  end for
  if view angle between the i-th and k-th views $>$ the view angle threshold **then**
    Call SBA to refine the initialized parameters;
    Set $k = i$;
  end if
end for
Apply SBA to all the parameters and get the final estimate of the camera motion $\{\mathrm{Y}^i\}$ and the scene structure $\{L_j\}$;
return $\{\mathrm{Y}^i\}$, $\{L_j\}$.

2π then SBA will never be applied during the incremental initialization process until the final step. In the following experiments, we simply set the threshold to zero, which means the following reported computing time is the upper bound for the given set of line correspondences. Here, we emphasize the three strategies employed in Alg.(2): (i) the combination of the Cayley representation and the Plücker representation, (ii) the fast

incremental initialization process, (iii) the usage of SBA. The performance gain will be demonstrated in the following experiments.

5.7 Experiments and applications

In this section we first report the experiment results for synthetic and real data to verify Alg.2 proposed in this chapter and make comparisons with the most cited work [WHA92; TK95; XLT12]. Then we demonstrate an example application of our algorithm in augmented reality. The following experiments are performed on a desktop computer with 3.4GHz Intel(R) Core4 processor and 8GB of RAM.

5.7.1 Synthetic experiments

The simulated experimental setup consists of a set of views looking inwards at 3D lines placed randomly in a cube with edge length 100mm. To make sure any group of the synthetic data is non-degenerated, the wire frame of the cube, i. e., twelve edges, are always included in the synthetic scene. The focal length of the synthetic camera is 754.5 pixels. The end points of all lines are projected in all views, where their positions are corrupted by an additive Gaussian noise. For a large part of the synthetic data, Taylor's algorithm [TK95] cannot achieve a convergent result without given initial camera rotations. So in the synthetic experiments, we only compare our sparse bundle adjustment approach with the closed-form solution [WHA92]. In all of the simulation experiments we measure the quality of reconstruction results by computing the estimation error from Eq.(5.14).

We first test how the image noise level affects the reconstruction algorithms. In this simulation, we generate 44 line correspondences across three views, and vary the noise level from 0 to 2 pixels in steps of 0.2. Fig.5.4 (a) shows the estimation errors of both algorithms with respect to the change of the image noise level. The second simulation experiment is carried out to verify the effect of the number of line correspondences. We fix the image noise level to one pixel and vary the number of line correspondences from 14 to 44 across three views. The estimation errors

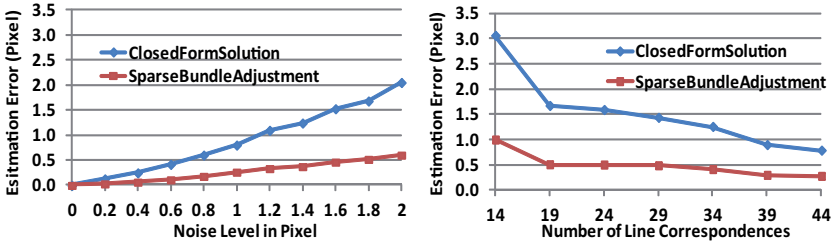
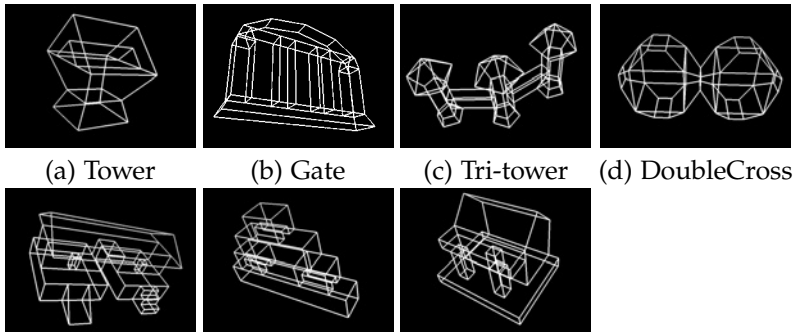


Figure 5.4. The estimation error of the closed-form solution and the sparse bundle adjustment results when varying the noise level of line endpoints (a) and the number of line correspondences (b). The red square line shows the results of sparse bundle adjustment and the blue diamond line shows the results of the closed-form solution. It is obvious that the sparse bundle adjustment achieves higher reconstruction accuracy.

are shown in Fig.5.4 (b).

In the last part of the synthetic experiments, we make a comparison between the line drawing based 3D reconstruction and the line feature based 3D reconstruction. In this experiment, we use the dataset from Xue et al. [XLT12] (Seven pieces of 3D models are available for us). For a fair comparison, the occlusion effect is not considered in the simulation, i. e., the internal line structure is also visible. For each model, we first generate three synthetic views, then reconstruct it using the proposed algorithm. The variance of the Gaussian noise added to the vertices is 0.05, i. e., supposing the coordinate of a vertex in 3D is $\boldsymbol{t} = [x, y, z]$, then the disturbed coordinate is $\boldsymbol{t}' = (1 + n(0, 0.05))\boldsymbol{t}$ where $n(\cdot, \cdot)$ is a Gaussian noise generator. Our reconstruction results are then compared with the results of the state-of-the-art line drawing based algorithm [XLT12][1] as reported in Tab.5.1. The reconstruction error is defined as the root mean square of Euclidean distances (RMSE) of corresponding vertices in ground truth and the reconstructed model. Note that the reconstructed model is aligned to the ground truth model before computing RMSE. Obviously,

[1]The reconstruction results of [XLT12] are by the courtesy of the author.



(a) Tower (b) Gate (c) Tri-tower (d) DoubleCross

(e) BigHouse (f) GateHole (g) House

Figure 5.5. Reconstruction of line drawing models in the dataset of [XLT12].

Table 5.1. Comparison of our approach (A) and Xue et al. [XLT12]'s (B). The value of RMSE is scaled with respect to the unit length of the 3D model.

	a Tower	b Gate	c Tri-tower	d DoubleCross	e BigHouse	f GateHole	g House
RMSE A	0.237	0.205	0.368	0.003	0.016	0.010	0.009
RMSE B	1.024	0.676	0.805	0.012	3.401	2.228	3.144

multiple images include more information about the scene than a single line drawing image, therefore multi-view line feature based 3D reconstruction has higher accuracy. Our reconstructed 3D models are illustrated in Fig.5.5. In terms of the computing time, it takes tens of milliseconds to reconstruct the 3D model by our algorithm while the line drawing algorithm spends a few minutes.

5.7.2 Experiments on real image sequences

Image dataset from Taylor

For real image sequences, we first carry out our experiments on the dataset from Taylor and Kriegman [TK95]'s work which includes the building

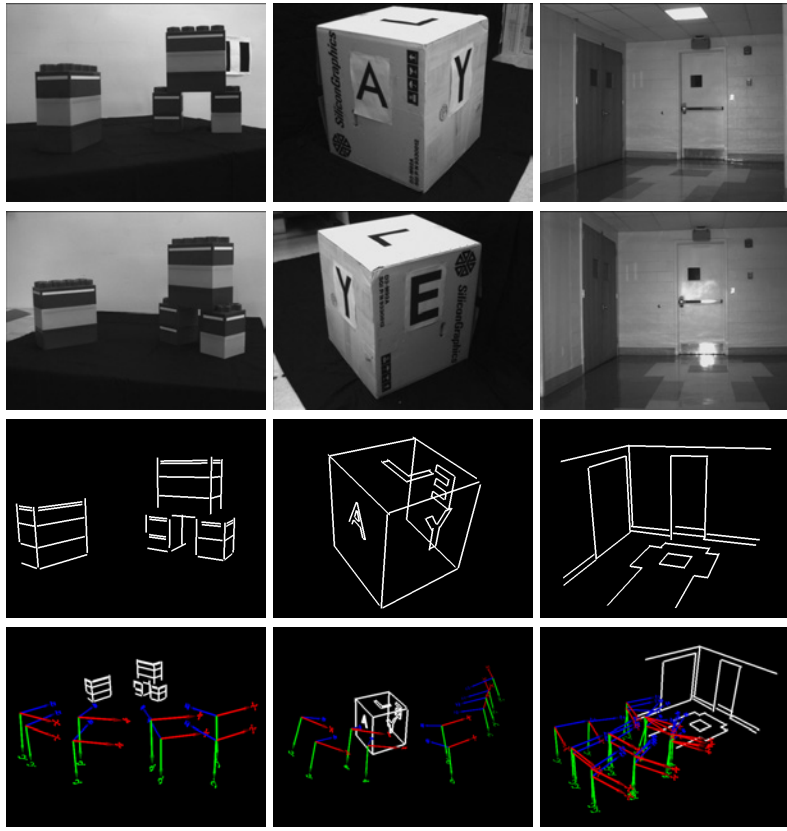


Figure 5.6. Reconstruction of three scenes including the building blocks, the Eli cube and the hallway. The first two rows are example images from sequences and the third row shows the snapshots of reconstructed scenes. In the last row, the reconstructed camera poses are shown by a set of small coordinate axes.

blocks, the Eli cube and the hallway sequences. The building blocks sequence includes eight views of three blocks and the Eli cube sequence includes ten views of a cube which has four printed letters on its four faces. The objects being viewed in the two examples are relatively small,

Table 5.2. Comparison of our approach (A) and Taylor and Kriegman [TK95]'s (B).

Scene	Num of Lines	Num of Views	Error A (Pixel)	Error B (Pixel)	Time A (s)	Time B (s)
Blocks	45	8	0.053	0.707	0.164	2.95
Eli cube	48	10	0.103	1.273	0.374	3.91
Hallway	33	24	0.133	1.625	0.873	7.17

so the third sequence is taken from a larger area inside their office and includes 24 views of the hallway scene.

Fig.5.6 shows the example images and the snapshot of the reconstructed scenes. The scene structures seem to be well recovered. The relationships between the lines in the reconstruction reflect the corresponding relationships of lines in the actual objects; parallel, perpendicular or coplanar lines in real appear in the same configuration in the reconstruction. As in [TK95], there is no claim about the locations of the endpoints of lines since they are simply taken to be the extrema of the backward projections of the measured image endpoints onto the reconstructed lines.

It is noteworthy that in our experiments, we do not employ the information of the initial camera rotation angles offered in the dataset which is necessary for Taylor and Kriegman [TK95]'s method. If we test their algorithm without the initial guess, then the reconstruction of the Eli cube and the hallway sequences will fail and the nonlinear iteration will get divergent. However, ours can successfully reconstruct these scenes without the guess.

There is no obviously visible difference between our reconstruction results and Taylor's. For comparison, we compute the reconstruction errors which are calculated from the projected line and the measured line endpoints by Eq.(5.14). Tab.5.2 gives the statistical results of both methods in terms of the reconstruction error and the computing time. It can be seen that our method is about ten times faster than Taylor's method and reduces the reconstruction error by one order. The reasons for the speedup are mainly because of the advantage of the fast initialization process and the sparse bundle adjustment. The higher reconstruction accuracy may benefit from the unconstrained Cayley parametrization of lines and the

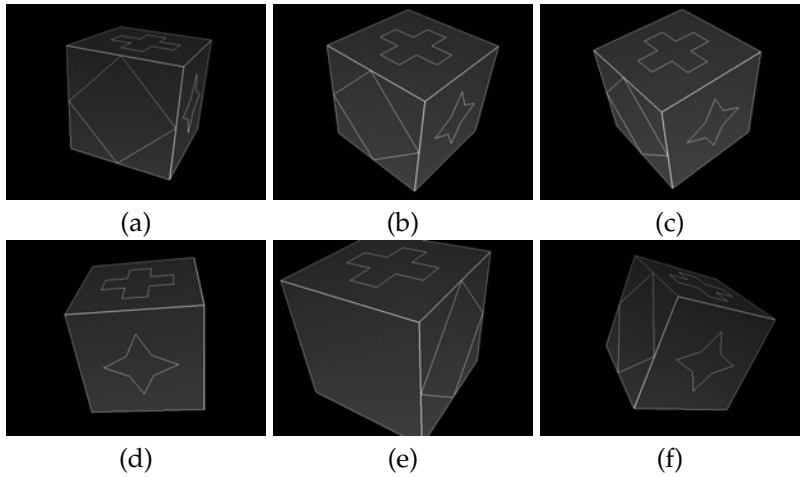


Figure 5.7. Six images in the sequence of the cube which includes 20 views and 44 lines. (a), (b) and (c) are the first three views in this sequence. (d) is the fifth view in the sequence which significantly improves the reconstruction result from this view. (e) is the twelfth view in the sequence which looks at the cube from the backside. (f) is the last view which almost returns to the start view point.

appropriate setting in the SBA algorithm (e. g., SBA allows more iterations while being still much faster than the conventional adjustment).

Self-gathered image dataset

The three sequences from Taylor's dataset only have small camera orientation variations (less than $\pi/2$). The reason may be that their algorithm needs a large number of global iterations when the variation of camera orientation is larger than $\pi/2$ as shown in the fourth simulation experiment of [TK95]. However, our approach theoretically does not suffer from this problem. To show that our method can reconstruct the scene and estimate the camera pose even when the camera turns a round in sequences, we carry out the following experiments on another cube and our laboratory sequences respectively.

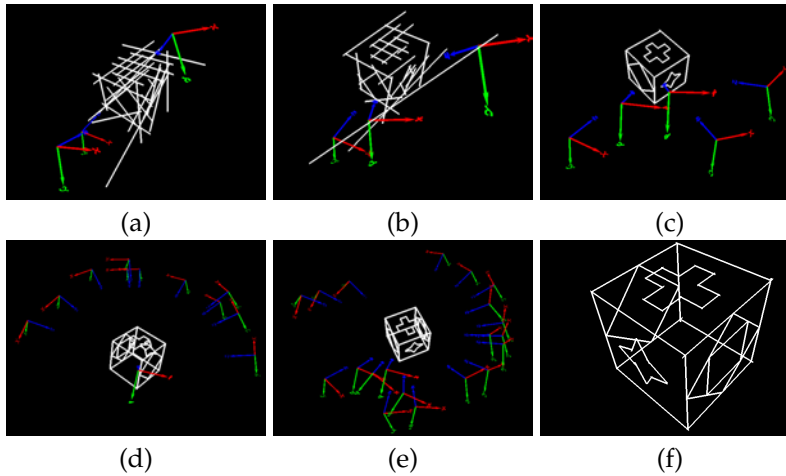


Figure 5.8. Snapshots of the incremental reconstruction procedure for the cube scene. (a) is the result of the closed-form solution from the first three views with the reprojection error as large as 4.805 pixels and (b) is the sparse bundle adjustment result of the first three views which reduces the reprojection error to 1.320 pixels. (c) is the result after reconstructing five views which significantly improves the reconstruction result with error 0.04 pixel. (d) is the result after twelve views which has large variations of camera orientations and the reprojection error keeps the same order as 0.045 pixel. (e) is the reconstruction result of the total twenty views with the final reprojection error 0.055 pixel and (f) shows the reconstructed scene structure without camera poses. The reconstruction of the whole sequence takes about 1.5 seconds.

The first sequence includes twenty views of a cube with some geometric graphs on its faces as shown in Fig.5.7. The relationships of line segments in this cube scene are not only parallel or orthogonal. In Fig.5.8, we show the reconstructed scene structure and the estimated camera pose during the incremental procedure. Because of the small camera baselines and view angles for the first three views, the closed-form solution gives a very rough result, and the SBA improves the result a little bit, but still has a large reprojection error. After five views, the camera baselines and view angles are getting large enough, so the reconstructed results turn to be

Figure 5.9. Six images taken from the sequence of the laboratory which includes 70 views and 165 lines. Those lines are shown in the images whenever they are detected.

stable. It is clearly shown in Fig.5.8(e) that the total twenty views are surrounding the cube.

In this cube sequence, just like all the previous sequences, some part of the scene (here, the top face of the cube) is kept in views during the whole sequence, which is greatly beneficial to our incremental procedure. To show that our approach can still work even if none of scene parts is kept in view all the time, we captured a second sequence in our laboratory, as shown in Fig.5.9. Note that in this sequence, the camera moves inside the scene and no part of the scene is viewed all the time. In Fig.5.10, we show the reconstructed scene snapshots taken from similar views as in Fig.5.9. The overview of the global scene is shown in Fig.5.11. Although the endpoints are not perfectly estimated, the basic geometric structure of the laboratory is clearly reconstructed, as parallel, perpendicular or coplanar lines are correctly reflected even under the challenge of imprecise line correspondences caused by the varying illumination and wide baseline views.

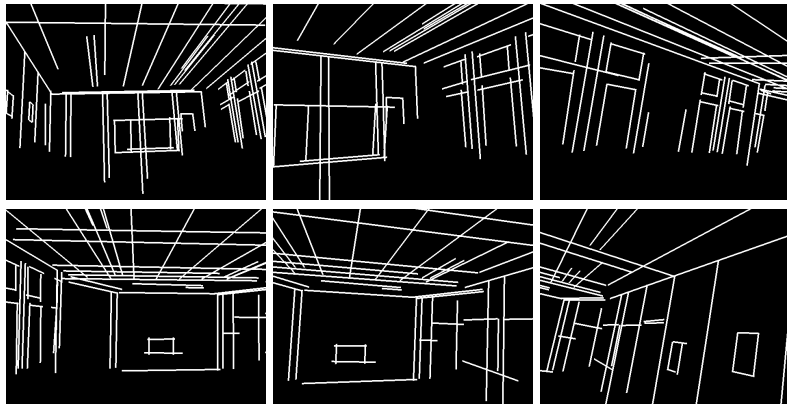
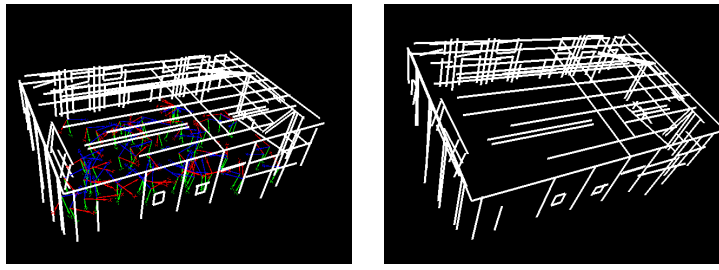


Figure 5.10. Snapshots of the reconstructed laboratory scene. The snapshots are taken from the similar views as their counterparts in Fig.5.9.



(a) (b)

Figure 5.11. The overview of the reconstructed laboratory scene with cameras (a) and without cameras (b). Note that all the camera poses are kept inside the room just as it should be and the calibration board is separated from the wall plane as shown in (b). The reconstruction process takes about 8 seconds with a final reprojection error of 0.195 pixel.

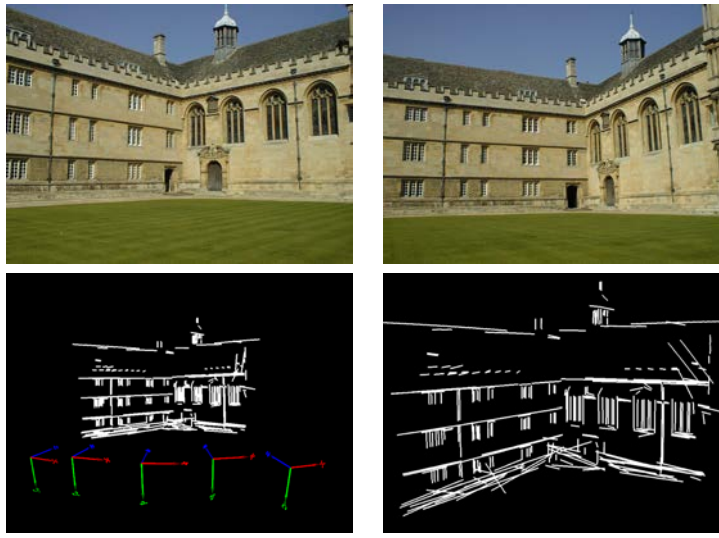


Figure 5.12. Sample images of the Wadham College and the reconstructed scene.

5.7.3 Experiment on dataset with mismatches

In the previous experiments, we assume having straight line correspondences between multi-view images with few mismatches. However, in the practical applications, it is inevitable that the correspondences established by feature matching algorithms are not perfect. To examine the performance of the proposed algorithm in practical situations, we test our algorithm on the Wadham College dataset from the Oxford Visual Geometry Group[2]. As described in the dataset, the line correspondences are automatically established across multi-view images with a lot of mismatches (about 15%). Due to the non-perfect line correspondences, some of the 3D lines are reconstructed badly. However, the camera poses and lines with correct matches are well reconstructed as shown in Fig.5.12. The reconstruction time for five images with more than 300 lines is about 0.8s. Although the mismatches of lines can be detected by evaluating their

[2] `http://www.robots.ox.ac.uk/~vgg/data/data-mview.html`, accessed on 10th, October, 2012

reprojection errors in the image or the geometric constraints on lines in three views, these approaches are not employed to remove the mismatches in the dataset for better illustrating the robustness of our method.

5.7.4 Application in augmented reality

In this section, we demonstrate the application of the reconstruction algorithm (SBA) proposed in this chapter, the camera pose estimation algorithm (RPnL) proposed in Chapter 4 and the line matching algorithm (LBD+S&G) proposed in Chapter 3. The algorithms are applied to the field of augmented reality which inserts virtual objects into a scene by superimposing them on an image captured by the camera.

The framework of the augmented reality system (AR system) is illustrated in Fig.5.13. A set of key frames of the scene is grabbed from various view points in advance. The 2D/2D line correspondences among key frames are established. Before running the AR system, we apply the SBA algorithm to reconstruct the scene. The 2D/3D line correspondences between key frames and the reconstructed scene are generated during the SBA process.

Now we start the real-time AR system in the scene with a hand-held camera. In the first step, for the initial frame in the sequence, we employ the DOT algorithm [Hin+10] to retrieve its closest image in the key frames. Then we employ the LBD+S&G algorithm to match lines extracted in the first image and its closest key frame. In the second step, according to the matching results, we transfer the 2D/3D line correspondences in the key frame to the first frame. In the third step, we employ the RPnL algorithm to estimate the camera pose of the first frame based on the 2D/3D correspondences. Then the virtual object, e. g., a car model, is superimposed into the first image using the estimated camera pose.

During the system running, for the $(i+1)$-th frame in the sequence, if the tracking process is failed, then the AR system is reinitialized by the DOT algorithm and the LBD+S&G algorithm as for the first image. If the tracking process runs successfully, then the LBD+S&G algorithm is employed to match lines extracted in the $(i+1)$-th and i-th frame. The 2D/3D line correspondences is transferred to the $(i+1)$-th frame based the matching results. Then the camera pose of $(i+1)$-th frame is estimated

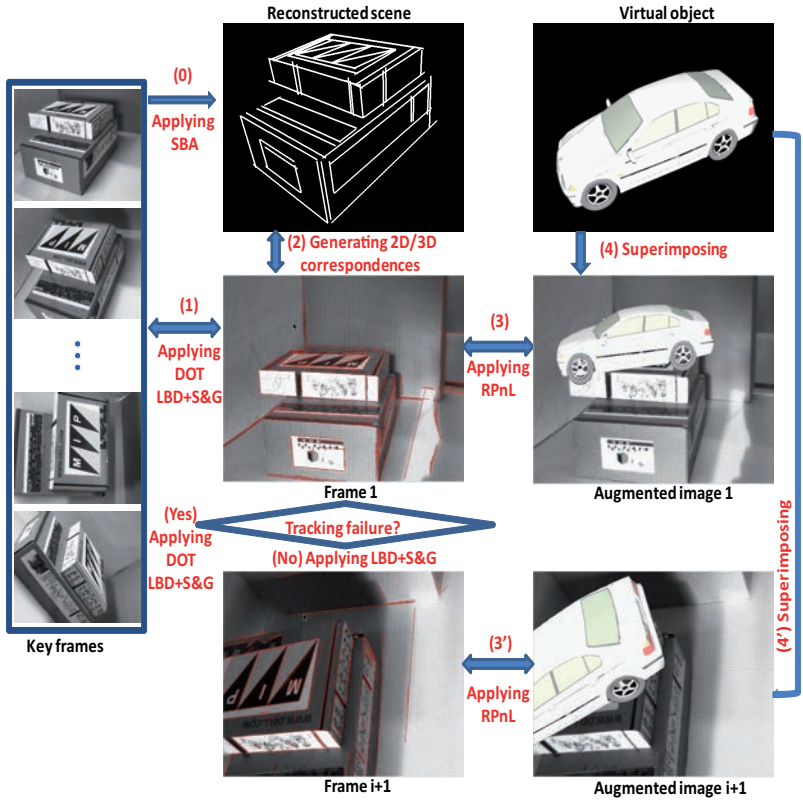
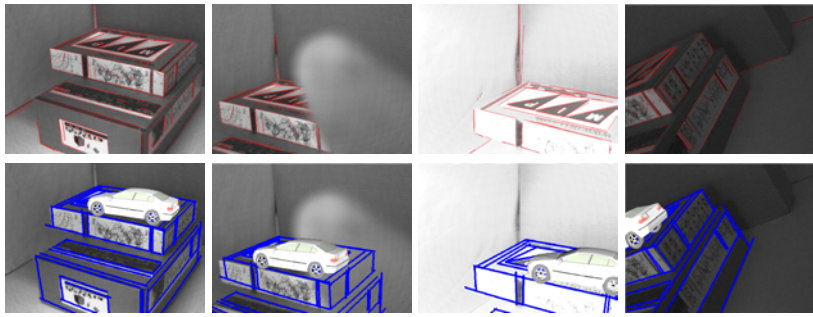


Figure 5.13. Framework of the augmented reality system.

by the RPnL algorithm and the virtual object is superimposed into the image. Tracking failure is detected when the reprojection error of the camera pose estimated by the RPnL algorithm is too large or the number of 2D/3D line correspondences is too small.

In Fig.5.14, we demonstrate the AR system running in real-time. The counterparts video is available on the web page[3]. The video is produced from the captured images and the augmented images during the AR

[3] http://sdrv.ms/12aw85X.



(a) Normal Condition (b) Image Occlusion (c) Strong Illumination (d) Weak Illumination

Figure 5.14. Demonstration of the AR system running in real-time under the challenge of image occlusions and illumination variations. The first row shows the input images with extracted lines and the second row shows the augmented images with the imposed box and car model.

system running phase. From the video, it can be seen that the system can run smoothly with a hand-held camera. The tracking process may fail due to the large occlusion, bad illumination condition or viewing out of the scene. However, the system can recover immediately when the camera returns to the normal condition.

5.8 Summary and discussion

This chapter presents a novel approach to reconstruct the scene structure and estimate the camera motion from a set of 2D/2D line correspondences across multiple views. Two difficulties when dealing with line primitives are addressed: the representation of spatial lines and their projections. Then an incremental initialization method starting from an image triplet is proposed. Finally, the SBA algorithm is adopted to solve the line reconstruction problem, which greatly speeds up the optimization procedure.

Simulations are carried out to investigate how the accuracy of the algorithms proposed in this chapter are affected by the variations of

different parameters, e. g., the noise level of image measurements and the number of line correspondences. The proposed multi-view reconstruction algorithm is compared to the line drawing based reconstruction algorithm on the line drawing dataset [XLT12]. We also test our algorithm on Taylor's dataset [TK95] and two other sequences. The experiments on these datasets prove that this new approach achieves higher accuracy of reconstruction results and more efficiency (reduce the computing time by one order generally). The experiments also show that the proposed algorithm can work in more complicated situations, e. g., no part of the scene is kept in view all the time or the camera makes a full 360-degree sweep.

To verify the performance of the algorithm on the dataset with mismatches, we also test our algorithm on the Wadham College dataset with about 15% of mismatches in the given line correspondences. The results show that the algorithm can still reconstruct the scene properly.

In the last experiment, combining with the work in the previous chapters, we build a AR system which runs in real-time with a hand-held camera. The experiment indicates that our algorithms can be improved to solve the Simultaneous Localization And Mapping (SLAM) problem as the work in [KM07], which will be addressed in our future research.

It must be pointed out that the proposed algorithm is dedicated to man-made linear structures which are abundant in architectural outdoor and indoor scenes. For more general scene representations, like curved lines, points, or textures, other approaches should be used or integrated into the line structure approach. For curved object contours, for example, the Micro Phase Shifting (MPS) [GN12] is a suitable technique.

Besides, the initial three views keep an important role for the successful reconstruction of the whole image sequence. For the bundle adjustment of a video sequence, some hierarchical method can be employed as proposed in [FFG09]. The way to identify and cluster images that potentially share a good number of line features should be investigated.

When the scene is under the Manhattan world assumption, i. e., lines are parallel or orthogonal to each other in space, then the scene can be reconstructed from a single view. The single view reconstruction algorithm can be a good alternative approach to solve the initialization problem instead of the closed-form solution of the image triplet. In the next chapter, we will investigate lines in the Manhattan world.

I heard that parallel lines actually do meet, but they are very discrete.

Anonymous

Chapter 6

Line primitives in a Manhattan world: vanishing point estimation and line classification

The problem of estimating vanishing points for visual scenes under the Manhattan world assumption has been addressed for more than a decade. Surprisingly, the special characteristic of the Manhattan world that lines should be orthogonal or parallel to each other is seldom well utilized.
In Chapter 4, we presented a complete scenario for three spatial lines in all possible configurations. Based on the results in Sec.4.3.2, in this chapter, we present an algorithm that accurately and efficiently estimates vanishing points and classifies lines by thoroughly taking advantage of the Manhattan world characteristic for images grabbed by a camera with a single effective viewpoint (e. g., perspective camera or central catadioptric camera). The algorithm is also extended to estimate the focal length of the camera when it is uncalibrated. The key novelty is to estimate three orthogonal line directions in the camera frame simultaneously instead of estimating vanishing points in the image plane directly. The performance of the proposed algorithm is demonstrated on four publicly available databases as well as on two self-gathered image sequences.
Large parts of this work have been pre-published in [ZK12c; ZLK13]

6.1 Introduction

Estimating vanishing points in an image has many applications ranging from single view reconstruction to autonomous navigation. For line primitives in a Manhattan world, their vanishing points are of special

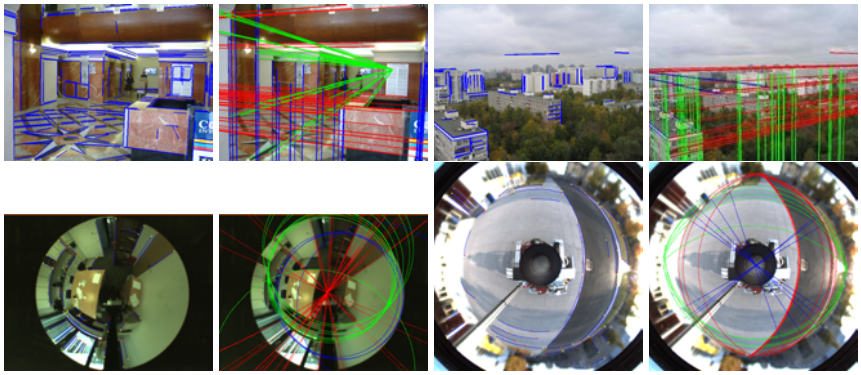


Figure 6.1. Illustration of the Manhattan world. Images in the first and third column show the original image with extracted lines imposed. Images in the second and fourth column show the classified lines corresponding to the vanishing points of three Manhattan directions (Lines with the same color have the same vanishing point).

interest because they directly give the camera orientation relative to the canonical 3D Cartesian frame defined by the Manhattan world [CY99; CY03; DEE08; MR11b; BSP12; WH12]. On the other hand, the projection of lines in the image of a Manhattan world can be classified by the estimated vanishing points accordingly as shown in Fig.6.1 which includes two indoor and outdoor urban scenes grabbed by perspective cameras and central catadioptric cameras, respectively.

In Chapter 4, the solution of the P3L problem is discussed by investigating the symmetric property of the problem. For an intrinsically calibrated perspective camera, the 3D line direction vector is parametrized by one-unknown-parameter which is the depth ratio of two spatial line endpoints. This representation automatically guarantees the constraint that the direction vector in the camera frame should be orthogonal to the normal of the plane defined by the image of the line and the camera center. Following the framework in Sec.4.3.2, in this chapter we address the problem of vanishing point estimation and line classification in a Manhattan world with the following contributions:

- In [GD01], Geyer and Daniilidis proposed a unifying model for both the central catadioptric camera and the perspective camera. By employing this model, the algorithm designed for the conventional perspective camera in Sec.4.3.2 is extended to the central catadioptric camera benefiting from the novel parametrization of the line direction in the camera frame.

- For a line triplet, a quadratic equation system about the line direction parameter is derived by using the feature of the Manhattan world that lines should be orthogonal or parallel to each other. The solutions of the system are exactly the orthogonal vanishing points. Further, they are completely consistent with the previous theoretical analysis on the multiplicity of solutions [MR11b; Che91].

- Moreover, when the intrinsic parameters of the perspective camera are unknown, we adjust our algorithm to estimate the vanishing points together with the focal length of the camera. For the central catadioptric camera, Geyer and Daniilidis [GD01] showed that the camera can be calibrated by the projection of lines. Hence, we suggest to calibrate the catadioptric camera in advance by using the calibration tool implemented by Barreto and Araujo [BA05].

- To measure the consistency between lines and vanishing points, the method based on the orthogonal constraint and the method based on the collinear constraint are introduced and evaluated. Besides, to refine the results of the RANSAC process, a simple iterative approach and the Maximum Likelihood Estimator are compared as well.

We validate the performance of the proposed algorithm on several publicly available databases: the York Urban Database (YUD) [DEE08], the Eurasian Cities Database (ECD) [TBKL12], the CoSy Localization Database (Cold) [PC09] and the Rawseeds Database [Cer+09]. The experiments demonstrate its efficiency and robustness. Compared to the state-of-the-art methods, the biggest advantage of the proposed algorithm is its efficiency. The maximum computing time of the C++ implementation is less than five milliseconds per image without sacrificing the accuracy for both calibrated and uncalibrated cameras. In the end, the proposed algorithm

is employed in two typical applications: single view reconstruction and robot navigation.

The rest of the chapter is organized as follows. After a review of the related work in Sec.6.2, in Sec.6.3 we present the method to estimate three orthogonal vanishing points from a line triplet in a Manhattan world. For uncalibrated cameras, the solutions are given in Sec.6.4. The RANSAC based line classification algorithm, the consistency measure methods and the refinement approaches are introduced in Sec.6.5 followed by the experimental evaluation and the applications in Sec.6.6. Finally, the conclusion is drawn in Sec.6.7.

6.2 Related work

Since Coughlan and Yuille [CY99] considered the problem of estimating the vanishing points of lines in a Manhattan world which imposes regularities on the image statistics, a considerable amount of work has been reported to address this problem.

A Bayesian approach to group edges was proposed in the early work of Coughlan and Yuille [CY99]. It performs a one-dimensional exhaustive search over a single camera angle based on a probabilistic classification of each edge. An improved version with a course-to-fine search over 3D camera orientation was reported in their latter work [CY03]. Gallagher [Gal02] improved the Bayesian based algorithm with a more precise prior probability model learned from the ground truth data. Schindler and Dellaert [SD04] replaced the stochastic search with a continuous optimization approach by using the expectation maximization (EM) algorithm. The EM-based methods [SD04; KZ02; WV07] assume that deviations of Manhattan edges and lines from the expected orientations are normally distributed which is often a non-valid assumption. Denis et al. [DEE08] built a more accurate statistical model from a training set by using edges rather than dense gradient maps. Nieto and Salgado [NS11] employed the EM algorithm to estimate the vanishing points and their converging lines simultaneously. These approaches suffer from two common drawbacks: Firstly, they require an initial estimation of the vanishing points and are sensitive to initialization. The typical initialization methods, such

as the Hough transform [LMLK94; TVGPM98] or the heuristic clustering [WV07; ALC05; Foe10], cannot guarantee to produce a sufficiently accurate initialization. Secondly, they are iterative in nature which makes them computationally too expensive. The reported best time performance of the EM-based methods in the comparison experiments of [DEE08] is about a few seconds per image.

To guarantee an optimal solution, Rother [Rot00] proposed an exhaustive search over vanishing point hypotheses obtained from all possible line intersections which enforces the orthogonality of the vanishing points. However, its computational complexity is $O(n^5)$, which is prohibitive, where n is the number of the extracted lines. Bazin et al. [BDVK12] also suggested a quasi-exhaustive search which samples the rotation space to determine the rotation maximizing the number of clustered lines. This method depends on the sampling rate and has to process a large number of samples for fine sampling or large search space. In their latter work [BSP12], they formulated the vanishing points estimation task as a consensus set maximization problem over the rotation search space which is divided into intervals. They employed the branch-and-bound algorithm [HK09] to solve this problem. It takes more than ten seconds to find the best solution. Recently, Tretyak et al. [TBKL12] presented an optimization based parsing framework to model the man-made scene as a composition of geometric primitives spanning different layers from low level (edges) through mid-level (lines and vanishing points) to high level (the zenith and the horizon). Due to the high dimension of their objective function, the whole process of their algorithm takes tens of seconds.

To improve the efficiency of the vanishing point estimation algorithm, Tardif [Tar09] presented a non-iterative solution for simultaneously estimating the vanishing points in an image given a set of sparse edges based on J-Linkage which is similar to RANSAC. This method does not enforce orthogonality of the vanishing points when generating hypotheses. Mirzaei and Roumeliotis [MR11b] introduced a robust and efficient RANSAC-based line classifier that employs an optimal estimator to generate hypotheses for all three orthogonal points from triplets of line observations. Although their approach achieves highly remarkable performance, the polynomial system solver employed by the optimal estimator is sophisticated which results in 40 solutions with large number of multiplicities.

Besides, the proposed algorithm in [MR11b] is difficult to generalize to uncalibrated cameras. The processing time of the aforementioned two approaches is around hundreds of millisecond per image.

The existing approaches seldom utilize the characteristic of the Manhattan world well. Most of them first build the algorithm for lines in general configurations then apply the Manhattan world assumption. [CDR99] is an early work which estimates the orthogonal vanishing points from the user-labeled parallel and perpendicular lines. Since user interaction is required, its applicability is limited. We notice that only very recently, two other works [WH12; BP12] together with our previous work [ZK12c], inherently enforce the orthogonality of the vanishing points by directly incorporating the orthogonal or parallel constraints into the model estimation step of the RANSAC procedure. The great performance gain is achieved by applying the constraints in the Manhattan world as demonstrated in these papers. The 3-line configuration addressed in [BP12] is a special case of ours, i. e., two lines are parallel and the third line is orthogonal to them. In [WH12], three orthogonal vanishing points as well as the camera focal length are estimated based on the relationship between vanishing points and the image of the absolute conic (IAC). If the projections of two parallel lines remain parallel in the image plane, then their cross product will be zero. If a hypothesis generated in [WH12] includes such kind of parallel lines, then the solution will be wrong. Although it will not affect the accuracy of the RANSAC algorithm in general (because it only returns the best hypothesis), it is a theoretical flaw. Besides, it is difficult to generalize the framework in [WH12] to the omnidirectional images.

The vanishing point is actually the direction of the corresponding lines in the camera frame. Based on this fact, we derive the solutions for three lines in all possible spatial configurations in the Manhattan world. In the following, we first consider three lines in a Manhattan world for a calibrated camera. The endpoints of lines extracted in the image are backward projected onto a unit sphere by Eq.(2.25) in advance. A brief introduction of the unifying camera model can be found in Sec.2.3.2.

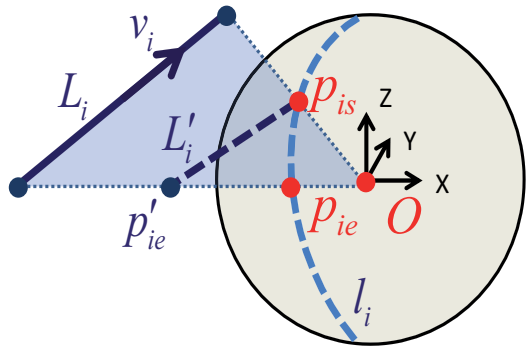


Figure 6.2. Illustration of the line projection on the unit sphere.

6.3 Line triplet in a Manhattan world

As discussed in Sec.4.3.2, for a line triplet in space, there are three possible configurations for their directions: $Rank(\boldsymbol{V}) = 3$, $Rank(\boldsymbol{V}) = 2$ and $Rank(\boldsymbol{V}) = 1$. Further, if the space is under the Manhattan world assumption, then the three corresponding configurations are as follows: (i) three lines are orthogonal to each other (corresponding to case 1.(a) in Sec.4.3.2); (ii) two lines are parallel, and the third line is orthogonal to them (corresponding to case 2.(a) in Sec.4.3.2); (iii) all of them are parallel to each other (corresponding to $Rank(\boldsymbol{V}) = 1$). These relationships between lines are invariant under Euclidean transformation. Note that, for the third configuration, since all three lines are parallel in space, only one vanishing point in the Manhattan world can be determined up to sign while the other two vanishing points have infinite solutions. This configuration is degenerated, so we consider only the first two in the sequel.

Following the framework in Sec.4.3.2, we extend the approach designed for the perspective camera to the central catadioptric camera. Consider three lines L_1, L_2 and L_3 with directions $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ respectively. Their projections on the unit sphere are portion of three great circles $\boldsymbol{l}_1$, $\boldsymbol{l}_2$ and $\boldsymbol{l}_3$ with endpoints $\boldsymbol{p}_{is}(x_{is}, y_{is}, z_{is})$ and $\boldsymbol{p}_{ie}(x_{ie}, y_{ie}, z_{ie})$, $i = 1, 2, 3$ as shown in Fig.6.2. For each line in space, it should lie on the projection plane

passing through its projection and the sphere center O. In this plane, we choose a line L_i' passing through the point $\boldsymbol{p}_{is}$ that is parallel with L_i. The line L_i' will intersect with the ray $O\boldsymbol{p}_{ie}$ at the point $\boldsymbol{p}_{ie}'(\lambda_i x_{ie}, \lambda_i y_{ie}, \lambda_i z_{ie})$. Since $L_i \parallel L_i'$, the direction of the spatial line in the sphere frame can be parametrized as:

$$\boldsymbol{v}_i = \boldsymbol{p}_{is} - \boldsymbol{p}_{ie}' = [x_{is} - \lambda_i x_{ie},\ y_{is} - \lambda_i y_{ie},\ z_{is} - \lambda_i z_{ie}]^T, \quad i = 1, 2, 3. \quad (6.1)$$

Similar to Eq.(4.11), λ_i is the parameter whose geometrical meaning is the depth ratio of the spatial endpoints in the sphere frame because $d(\boldsymbol{p}_{ie}')/d(\boldsymbol{p}_{is}) = \lambda_i/1$. Here, the depth $d()$ of a point is its distance to the sphere center. Now we consider the first two configurations separately. For each configuration, we will derive a quadratic to solve the line directions and the vanishing points.

Configuration 1: Three lines are orthogonal to each other: $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$, $\boldsymbol{v}_1 \perp \boldsymbol{v}_3$ and $\boldsymbol{v}_2 \perp \boldsymbol{v}_3$. In the sphere frame, we have

$$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = 0,\ \boldsymbol{v}_1 \cdot \boldsymbol{v}_3 = 0,\ \boldsymbol{v}_2 \cdot \boldsymbol{v}_3 = 0. \quad (6.2)$$

By substituting Eq.(6.1) into Eq.(6.2), we get three second order polynomials:

$$\begin{cases} (x_{1s} - \lambda_1 x_{1e})(x_{2s} - \lambda_2 x_{2e}) + (y_{1s} - \lambda_1 y_{1e})(y_{2s} - \lambda_2 y_{2e}) + \\ \qquad (z_{1s} - \lambda_1 z_{1e})(z_{2s} - \lambda_2 z_{2e}) = 0 \\ (x_{1s} - \lambda_1 x_{1e})(x_{3s} - \lambda_3 x_{3e}) + (y_{1s} - \lambda_1 y_{1e})(y_{3s} - \lambda_3 y_{3e}) + \\ \qquad (z_{1s} - \lambda_1 z_{1e})(z_{3s} - \lambda_3 z_{3e}) = 0 \\ (x_{2s} - \lambda_2 x_{2e})(x_{3s} - \lambda_3 x_{3e}) + (y_{2s} - \lambda_2 y_{2e})(y_{3s} - \lambda_3 y_{3e}) + \\ \qquad (z_{2s} - \lambda_2 z_{2e})(z_{3s} - \lambda_3 z_{3e}) = 0 \end{cases}. \quad (6.3)$$

After the variable resultant and substitution of λ_2, λ_3, similar to Eq.(4.16), we get a quadratic about λ_1 as:

$$g(\lambda_1) = w_2 \lambda_1^2 + w_1 \lambda_1 + w_0 = 0. \quad (6.4)$$

λ_1 is computed by solving this quadratic equation (6.4), then the remaining two parameters λ_2 and λ_3 are linearly computed from Eq.(6.3).

For a calibrated camera, the vanishing point associated with a line is the direction of the line in the camera frame [MR11b] (in our case, the sphere frame). Since three lines are orthogonal to each other, we get three orthogonal vanishing points from the line directions in the Manhattan world as:

$$\boldsymbol{VP} = [\bar{v}_1, \bar{v}_2, \bar{v}_3], \tag{6.5}$$

where $\bar{v}_i$ is the normalized vector of v_i.

Configuration 2: Two lines are parallel, and the third line is orthogonal to them. For this configuration, there are three possible situations: $v_1 \parallel v_2$, $v_1 \parallel v_3$ or $v_2 \parallel v_3$. Supposing $v_1 \parallel v_2$, then in the sphere frame, v_1 and v_2 should equal up to scale which yields:

$$\begin{cases} (x_{1s} - \lambda_1 x_{1e})(z_{2s} - \lambda_2 z_{2e}) = (x_{2s} - \lambda_2 x_{2e})(z_{1s} - \lambda_1 z_{1e}) \\ (y_{1s} - \lambda_1 y_{1e})(z_{2s} - \lambda_2 z_{2e}) = (y_{2s} - \lambda_2 y_{2e})(z_{1s} - \lambda_1 z_{1e}) \\ (x_{1s} - \lambda_1 x_{1e})(y_{2s} - \lambda_2 y_{2e}) = (x_{2s} - \lambda_2 x_{2e})(y_{1s} - \lambda_1 y_{1e}) \end{cases} . \tag{6.6}$$

By using the variable resultant, a similar quadratic about λ_1 can be generated from the first two rows of Eq.(6.6):

$$g'(\lambda_1) = w'_2 \lambda_1^2 + w'_1 \lambda_1 + w'_0 = 0. \tag{6.7}$$

Note that if $z_{1e} \neq 0$ and $z_{2e} \neq 0$, then $\lambda_1 = z_{1s}/z_{1e}$ must be one root of Eq.(6.7) because $\lambda_1 = z_{1s}/z_{1e}$, $\lambda_2 = z_{2s}/z_{2e}$ is obviously a solution of the first two rows of Eq.(6.6). However, in general this solution does not satisfy the third row of Eq.(6.6). So $\lambda_1 = z_{1s}/z_{1e}$ is a trivial root of Eq.(6.7) and is discarded. If it does satisfy the third row of Eq.(6.6), then it is kept which defines a vanishing point at infinity. Actually, in this case, we have $w'^2_1 - 4w'_2 w'_0 = 0$, i. e., $\lambda_1 = z_{1s}/z_{1e}$ is the double root of Eq.(6.7). This happens when the camera is perspective (or with a planar mirror, i. e., the parameter ξ in Tab.2.2 equals zero) and the projections of the parallel lines remain parallel in the image plane.

After solving λ_1, we can calculate λ_2 from Eq.(6.6). Then λ_3 can be computed by the constraints $v_1 \cdot v_3 = 0$. Since v_1 is parallel to v_2, they correspond to the same vanishing point in the Manhattan world. The second vanishing point is v_3. The third vanishing point in the Manhattan world

is determined by the cross product of v_1 and v_3 under the orthogonality constraint, i. e.,

$$\boldsymbol{VP} = [\bar{v_1},\ \bar{v_3},\ \overline{v_1 \times v_3}]. \tag{6.8}$$

6.3.1 The multiplicity of solutions

It is worth to investigate the multiplicity of vanishing point solutions from a line triplet. One exciting feature of our approach compared to [MR11b] is that our approach only generates the geometrically meaningful solutions. The details of the analysis are as follows.

For configuration one, it is obvious that at most two real roots can be generated from Eq.(6.4). Consequently, there are at most two distinct solutions of vanishing points. For configuration two, as mentioned above, only one non-trivial root is obtained from Eq.(6.7) in general or a double root $\lambda_1 = z_{1s}/z_{1e}$ is obtained, so only one distinct solution is generated.

In [MR11b], an algebraic geometry algorithm is employed to solve a three-variable and fifth-order polynomial system which yields 40 possible solutions. So the analysis of multiplicity is more complicated. Comparatively, our result is quite clear and simple. Besides, it is consistent with previous conclusions: In [MR11b], it is concluded that at most two distinct solutions for the orthogonal vanishing points can be obtained from observations of more than three lines passing through at least two vanishing points; In [Che91], it is pointed out that in order to have a finite number of solutions for the camera's orientation, three or more lines must be observed and at least one of the 3D lines must be nonparallel with the others. In a Manhattan world this corresponds to observing three lines that pass through at least two (out of three) vanishing points. In that case, up to eight solutions for the camera's orientation may exist. For each set of estimated vanishing points, a rotation matrix $\boldsymbol{R}$ can be obtained by $\boldsymbol{R} = \boldsymbol{VP}/det(\boldsymbol{VP})$ where $det()$ means the determinant of a matrix. Based on the symmetric property of line directions as presented in Sec.4.3.2, it is easy to verify that $\boldsymbol{R'} = \boldsymbol{RM}$ for

$$\boldsymbol{M} \in \left\{ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right\}$$

is also a valid orientation solution.

Now we summarize the results for line triplet in the Manhattan world: (i) If three lines are orthogonal to each other, then there are two distinct solutions for the orthogonal vanishing points which correspond to eight solutions of the camera orientation. (ii) If two lines are parallel, and the third line is orthogonal to them, then there is one distinct solution for the orthogonal vanishing points which is corresponding to the non-trivial root of Eq.(6.7) in general or corresponding to the double root of Eq.(6.7) when the camera is perspective (or with a planar mirror) and the projections of two parallel spatial lines remain parallel in the image plane. The solution of vanishing points will generate four solutions of the camera orientation. (iii) If three lines are parallel to each other, then only one vanishing point is determined, for the rest of two vanishing points, there are infinite solutions, also for the camera orientation.

6.4 In the case of uncalibrated camera

As stated in the introduction of this chapter (Sec.6.1), Geyer and Daniilidis [GD01] showed that the central catadioptric camera can be calibrated from the projections of a few lines in a single image without the knowledge of the scene. The reason is that the projection of a line in the omnidirectional image is generally a conic section which enforces five constraints (for hyperbolic or elliptical mirrors) or three constraints (for parabolic mirrors). The number of constraints enforced on the projection of a line is larger than the number of unknown parameters to specify the projection plane normal. In contrast, for perspective cameras, the projection of a line is a segment in the image which only enforces two constraints on the unknown 3D line. If there is no extra knowledge of the scene, then it is impossible to calibrate the perspective camera from the line projections. Therefore, in this work, we only address the case of uncalibrated perspective camera by using the knowledge of the Manhattan world.

Moreover, as pointed out by Cipolla [CDR99] and Liebowitz [Lie01], in practice the principal point is very sensitive to image noise. This is because for a zero skew camera with unit aspect ratio, the principal point is the orthocenter of the triangle formed by the three orthogonal vanishing

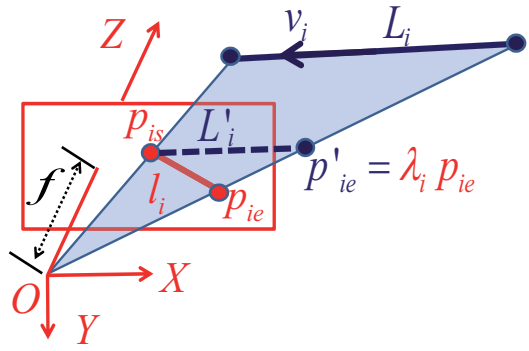


Figure 6.3. Illustration of the line projection for a perspective camera with unknown focal length.

points in which at least one of them is often far from the image center (their corresponding lines in the image plane are close to parallel). It is common to set the principal point to the image center under the assumption of zero skew and unit aspect ratio [WH12; TBKL12; CDR99].

Now the only unknown intrinsic parameter of the perspective camera is the focal length f. Since f is unknown, one cannot backward project the line endpoints onto the unit sphere. In this case, we consider the perspective image plane instead of the sphere. Fig.6.3 illustrates the projection of a line into the image plane for a perspective camera with unknown focal length. The projection of the spatial line L_i with direction $\boldsymbol{v}_i$ in the image plane is $\boldsymbol{l}_i$ with endpoints $\boldsymbol{p}_{is}(x_{is}, y_{is}, f)$ and $\boldsymbol{p}_{ie}(x_{ie}, y_{ie}, f)$. Similar to Fig.6.2, the line L'_i passing through $\boldsymbol{p}_{is}$ is parallel with L_i in space. The line direction in the camera frame is parametrized as:

$$\boldsymbol{v}_i = \boldsymbol{p}_{is} - \boldsymbol{p}'_{ie} = [x_{is} - \lambda_i x_{ie}, y_{is} - \lambda_i y_{ie}, f - \lambda_i f]^T . \tag{6.9}$$

Four lines are required to estimate the focal length and three orthogonal vanishing points. There are four configurations for four spatial lines in the Manhattan world: (i) two lines are parallel and mutually orthogonal with the other two lines; (ii) four lines are drawn in two orthogonal groups,

each group includes a parallel line pair; (iii), three lines are parallel and the fourth line is orthogonal to them; (iv) all of them are parallel to each other. The last two configurations are not admissible to solve the problem [WH12]. In the following we propose the solutions of the first and second line configurations.

Configuration 1: Two lines are parallel and mutually orthogonal with the other two lines. There are six possible situations in this configuration. Without loss of generality, we assume that $\boldsymbol{v}_1 \parallel \boldsymbol{v}_2 \perp \boldsymbol{v}_3 \perp \boldsymbol{v}_4$. Similar to Eq.(6.6), for $\boldsymbol{v}_1 \parallel \boldsymbol{v}_2$, we have:

$$\begin{cases} (x_{1s} - \lambda_1 x_{1e})(f - \lambda_2 f) = (x_{2s} - \lambda_2 x_{2e})(f - \lambda_1 f) \\ (y_{1s} - \lambda_1 y_{1e})(f - \lambda_2 f) = (y_{2s} - \lambda_2 y_{2e})(f - \lambda_1 f) \\ (x_{1s} - \lambda_1 x_{1e})(y_{2s} - \lambda_2 y_{2e}) = (x_{2s} - \lambda_2 x_{2e})(y_{1s} - \lambda_1 y_{1e}) \end{cases}. \tag{6.10}$$

Based on Eq.(6.10), λ_1 and λ_2 can be solved from the first two rows. One of the solutions $\lambda_1 = \lambda_2 = 1$ is discarded in general except when $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ in the image plane remain parallel, i. e., $\lambda_1 = \lambda_2 = 1$ is the double root of Eq.(6.10). Similar to Eq.(6.2), from $\boldsymbol{v}_1 \perp \boldsymbol{v}_3 \perp \boldsymbol{v}_4$, we have

$$\begin{cases} (x_{1s} - \lambda_1 x_{1e})(x_{3s} - \lambda_3 x_{3e}) + (y_{1s} - \lambda_1 y_{1e})(y_{3s} - \lambda_3 y_{3e}) + \\ \qquad (f - \lambda_1 f)(f - \lambda_3 f) = 0 \\ (x_{1s} - \lambda_1 x_{1e})(x_{4s} - \lambda_4 x_{4e}) + (y_{1s} - \lambda_1 y_{1e})(y_{4s} - \lambda_4 y_{4e}) + \\ \qquad (f - \lambda_1 f)(f - \lambda_4 f) = 0 \\ (x_{3s} - \lambda_3 x_{3e})(x_{4s} - \lambda_4 x_{4e}) + (y_{3s} - \lambda_3 y_{3e})(y_{4s} - \lambda_4 y_{4e}) + \\ \qquad (f - \lambda_3 f)(f - \lambda_4 f) = 0 \end{cases}. \tag{6.11}$$

If $\lambda_1 = 1$ is the double root, then we can directly compute λ_3 and λ_4 from the first two rows of Eq.(6.11) respectively. Then solve the focal length f from the third row. Note that if either $\lambda_3 = 1$ or $\lambda_4 = 1$ which means that two vanishing points lies at infinity in the image plane (i. e., the image plane is parallel with two Manhattan directions), then f is unsolvable. In practice, this never happens for four lines in configuration 1 because it is impossible to observe the projections of lines corresponding to three Manhattan directions simultaneously when the image plane is parallel with two of them.

Table 6.1. The number of solutions for four lines with uncalibrated cameras. η is the number of Manhattan directions which are parallel to the image plane. ' - ' means the situation never happens in practice. '∞' means there are infinite number of solutions.

	Configuration 1		Configuration 2	
	#VPs	#Focal	#VPs	#Focal
$\eta = 0$	2	2	1	1
$\eta = 1$	1	1	∞	∞
$\eta = 2$	-		1	∞

If $\lambda_1 \neq 1$, then from the first two rows of Eq.(6.11), we get λ_3 and λ_4 which are functions of f^2. Substituting them into the third row with some straightforward manipulation, a quartic equation about f can be derived:

$$g(f) = \alpha_2 f^4 + \alpha_1 f^2 + \alpha_0 = 0, \tag{6.12}$$

in which α_0, α_1 and α_2 can be calculated from λ_1 and the endpoints in the image plane. From Eq.(6.12), it is easy to infer that in general, at most two real solutions $f > 0$ can be obtained. Substituting f back into the first two rows of Eq.(6.11), λ_3 and λ_4 can then be linearly calculated. Then the three vanishing points are just the normalization of $\boldsymbol{v}_1, \boldsymbol{v}_3$ and $\boldsymbol{v}_4$.

Configuration 2: Four lines are drawn in two orthogonal groups, each group includes a parallel line pair. There are three possible situations in this configuration. Without loss of generality, we assume that $\boldsymbol{v}_1 \parallel \boldsymbol{v}_2 \perp \boldsymbol{v}_3 \parallel \boldsymbol{v}_4$. First according to $\boldsymbol{v}_1 \parallel \boldsymbol{v}_2$, we can compute λ_1 and λ_2 by using Eq.(6.10). λ_3 and λ_4 can be computed similarly. Again, if none of them equals to 1, then f is easily solved from the first row of Eq.(6.11). After solving λ_1, λ_3 and f, the vanishing points are just the normalization of $\boldsymbol{v}_1$, $\boldsymbol{v}_3$ and their cross product. If both λ_1 and λ_3 equal to 1, then the first two vanishing points are the normalization of $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$. The third vanishing point is $[0, 0, 1]^T$. f is unsolvable in this situation. If only one of them equals to 1, then the third vanishing points and the focal length are unsolvable. This happens in practice when the image plane is parallel with one Manhattan direction.

The number of the vanishing point solutions and the focal length

solutions for four lines with uncalibrated camera is summarized in Tab.6.1. Our result is consistent with the result in [WH12]. However, the solution of vanishing points and the focal length in [WH12] does not distinguish the situation when the projections of parallel spatial lines remain parallel in the image plane.

6.5 Classification of lines

Like other approaches to estimate vanishing points [Tar09; MR11b; WV07], the proposed algorithm is RANSAC-based.

For central catadioptric cameras, since generally the line projections in the omnidirectional image are conic sections, we randomly sample line triplets from the image as hypotheses. The hypotheses are tested by assuming the following four situations: $L_1 \perp L_2 \perp L_3$ (configuration 1), $L_1 \parallel L_2 \perp L_3$, $L_1 \parallel L_3 \perp L_2$ and $L_2 \parallel L_3 \perp L_1$ (configuration 2). At most five solutions of vanishing points $(2 + 1 + 1 + 1)$ will be computed for each hypothesis by using the approach presented in Sec.6.3.

For perspective cameras, an improved RANSAC algorithm is designed to speed up the hypothesis test by the following observation from the YUD database: for at least one of the Manhattan directions, a large number of its corresponding lines are close to parallel in the image plane. Based on this observation, we first compute the histogram of image line directions ranging in $[-90°, 90°]$. According to the direction histogram, we then group image lines for two cases: (a) most of the image lines included in the largest bin correspond to the same Manhattan direction; in this case, groupA1 includes lines in the largest bin and groupA2 includes the remainder lines; (b) the corresponding Manhattan directions of image lines in the largest bin are randomly distributed; in this case groupB includes all the image lines.

If the perspective camera is calibrated, then for case (a), the hypotheses are generated by two steps: first randomly picking two lines from groupA1, then sampling a line from groupA2. The hypotheses are tested by assuming that two lines from groupA1 are parallel and the line from groupA2 is orthogonal to them. Then the approach presented in Sec.6.3 is employed to estimate the vanishing points. Only one solution will be

computed for each hypothesis in this case. For case (b), three lines are sampled from groupB to form a hypothesis. For each hypothesis, the two non-degenerated line configurations presented in Sec.6.3 are tested, which generate at most five solutions of vanishing points.

If the perspective camera is uncalibrated, then for case (a), the hypotheses are generated by first randomly picking two lines from groupA1, then sampling two lines from groupA2. The hypotheses are tested by assuming that two lines from groupA1 are parallel and the other two lines are either orthogonal or parallel mutually but orthogonal to the first two lines. For case (b), four lines are randomly sampled and all nine admissible situations (six situations in configuration 1 and three situations in configuration 2 in Sec.6.4) are tested.

The number of generated hypotheses τ_{hyp} is selected by the adaptive algorithm for determining the number of RANSAC samples presented in [HZ04]. In practice, there should also be an empirical minimum and maximum iterative threshold (in our work, $40 \leqslant \tau_{hyp} \leqslant 100$). All the solutions generated by hypotheses are employed to classify the lines and the one resulting in the largest number of inliers is the winner of the RANSAC procedure. Here, an inlier is a line whose consistency measure ε with one of the vanishing points is smaller than a threshold t_ε. In the following, we discuss two most popular consistency measure methods of a line with respect to a vanishing point.

Consistency Measure 1 (CM1): The first method to measure the consistency of a line with respect to a vanishing point is based on the constraint that the vanishing point of a line in space should lie on the projection plane of the line, hence, the plane normal should be orthogonal to the vanishing point [MR11b; ZK12c]. ε_{1ij} is defined as the scalar product between the normal of projection plane $\boldsymbol{n}_i$ and the vanishing point $\boldsymbol{vp}_j$ in the camera frame (or the sphere frame):

$$\varepsilon_{1ij} = \boldsymbol{n}_i \cdot \boldsymbol{vp}_j. \tag{6.13}$$

Consistency Measure 2 (CM2): The second method is based on the constraint that the vanishing point of a line in the image plane should lie on the projection of the line. An ideal line is created by passing through the vanishing point in the image plane and the middle point of the image

line [WH12; Tar09]. ε_{2ij} is defined as the distance of a endpoint $\boldsymbol{p}_i$ to the ideal line $\hat{\boldsymbol{l}}_i$ passing through the vanishing point $\boldsymbol{vp}_j$ and the middle point $\boldsymbol{m}_i$ of the image line:

$$\varepsilon_{2ij} = d(\boldsymbol{p}_i, \hat{\boldsymbol{l}}_i) = d(\boldsymbol{p}_i, \boldsymbol{vp}_j \times \boldsymbol{m}_i), \tag{6.14}$$

in which $d()$ is a distance function from a point to a line and $\times$ denotes the cross product of two vectors.

CM1 is more suitable for the unifying camera model but it ignores the length of line projections in the image. **CM2** measures the consistency in the image which is usually preferred because that is where the uncertainty originates [HZ04]. However, **CM2** is only suitable for the perspective camera. In Sec.6.6.1, we experimentally compare **CM1** and **CM2** on the YUD database.

6.5.1 Refinement of the classification

Generally, the result of the RANSAC process can be refined by some optimization strategies to improve the accuracy. In [ZK12c], the result is refined by using only the inliers. An iterative method (e. g., the Newton-Raphson Method or the Levenberg-Marquardt algorithm) can be employed to optimize the objective function $J = \sum_{j=1}^{3} \sum_{i=1}^{\#C_j} \varepsilon_{ij}^2$ where $\#C_j$ is the number of inlier lines in class C_j. Here, ε_{ij} can be either ε_{1ij} as in Eq.(6.13) or ε_{2ij} as in Eq.(6.14). We denote this method as **Iter** for later reference.

In [WH12], all the extracted lines in the image are used to refine the result and a probability likelihood model is built. Let $\Psi = \{f,\ \boldsymbol{VP}\}$ be the set of parameters. The likelihood of a line segment corresponding to a Manhattan direction is modeled as:

$$P(\boldsymbol{l}_i|\Psi) = \sum_{j=1}^{3} \theta_j P(\boldsymbol{l}_i|\boldsymbol{vp}_j(\Psi)) + \theta_4 P(\boldsymbol{l}_i|O), \tag{6.15}$$

where $P(\boldsymbol{l}_i|\Psi)$ is the likelihood of a line segment corresponding to a particular vanishing point and $P(\boldsymbol{l}_i|O)$ models the contribution of the outlier process. The coefficients $\Theta = \{\theta_j|\theta_j > 0 \wedge \sum_{j=1}^{4} \theta_j = 1\}$ are the priors on the fraction of the line segments corresponding to one vanishing

point or being an outlier. Θ can be computed from the classification results of RANSAC process. Assuming conditional independence between line segments, Ψ can be estimated by using a maximum likelihood estimator of the form:

$$\Psi^{\star} = arg \max_{\Psi} \sum_{i} log\ P(\boldsymbol{l}_i|\Psi). \tag{6.16}$$

Now, what is missing in this likelihood model is the probability $P(\boldsymbol{l}_i|\boldsymbol{vp}_j(\Psi))$ and $P(\boldsymbol{l}_i|O)$. Wildenauer and Hanbury [WH12] created a training dataset and found that a Cauchy distribution is closer to the empirical distribution than the zero-mean Gaussian distribution. Thus, $P(\boldsymbol{l}_i|\boldsymbol{vp}_j(\Psi))$ is modeled as:

$$P(\boldsymbol{l}_i|\boldsymbol{vp}_j(\Psi)) = \frac{1}{\pi}\left(\frac{\sigma}{\varepsilon_{2ij}^2 + \sigma^2}\right), \tag{6.17}$$

in which σ is the scale parameter. The outlier process $P(\boldsymbol{l}_i|O)$ is modeled as a small constant component. Finally, the BFGS algorithm [NW06] is employed to maximize the log-likelihood in Eq.(6.16). The Maximum Likelihood Estimator is abbreviated as **MLE**. In Sec.6.6.2, we experimentally compare the accuracy and efficiency of **Iter** and **MLE**.

6.6 Experiments and applications

In the following experiments, four publicly available databases are tested for different purposes. The YUD [DEE08] database includes 102 calibrated perspective images from indoor and outdoor Manhattan worlds. The labeled ground truth lines as well as their corresponding vanishing points are given. The ECD [TBKL12] database includes 103 uncalibrated perspective images of outdoor urban scenes in which a significant proportion of images violates the Manhattan assumption. The zenith and horizon of the scenes are given. For catadioptric cameras, we choose an uncalibrated omnidirectional image sequence from the Cold [PC09] database taken in the office and a calibrated image sequence from the Rawseeds [Cer+09] database taken on the campus. Two more sequences taken in our laboratory with a calibrated perspective camera are also tested to demonstrate

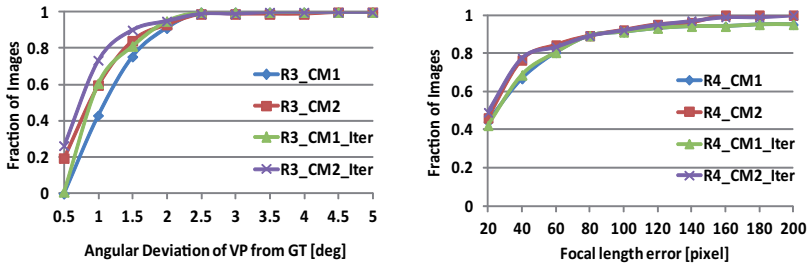
the performance of the proposed algorithm in real applications.

The LSD line detector [GJMR10] is employed to extract lines in the perspective image and the approach in [BDVK10] is utilized to extract lines in the omnidirectional image.

For calibrated cameras, the 3-line RANSAC algorithm is denoted as **R3** which only estimates the vanishing points. For uncalibrated cameras, the 4-line RANSAC algorithm is denoted as **R4** which estimates both the vanishing points and the camera focal length. To capture the statistical character of the RANSAC based algorithms, we run 100 times for each RANSAC based algorithm in the following experiments. The cumulative histogram is used to present the performance of algorithms on the databases. In each cumulative histogram graph, the vertical axis is always the fraction of images in the tested database which have smaller errors than the values given in the horizontal axis. All the experiments are performed on a 3.4GHz Intel(R) Core4 processor with 8GB of RAM.

6.6.1 Comparison of the consistency measures

In this experiment, we test the performance of algorithms with different consistency measures (**CM1** and **CM2**). The **MLE** refinement method requires different prior models for **CM1** and **CM2**. It will be hard to judge the performance difference between **CM1** and **CM2** because the performance also depends on the parameters of the prior models in the **MLE** algorithm. Therefore, in this comparison, only **Iter** is chosen in the refinement step for both calibrated camera and uncalibrated camera. We take the extracted lines from the YUD database as input. In the case of the vanishing point estimation for calibrated cameras, we evaluate the mean angular derivation between the estimated vanishing points and the ground truth offered in the YUD database. Fig.6.4(a) shows the cumulative histogram of the mean angular derivation. **R3_CM1** means the 3-line RANSAC algorithm using **CM1** as consistency measure. **R3_CM1_Iter** means the refinement of **R3_CM1** by the **Iter** method. **R3_CM2** and **R3_CM2_Iter** can be explained similarly. For uncalibrated cameras, we compare the accuracy of the estimated camera focal length. Fig.6.4(b) shows the cumulative histogram of the estimated focal length error. From Fig.6.4, it can be seen that in terms of accuracy, **CM2** is slightly better



(a) For calibrated cameras (b) For uncalibrated cameras

Figure 6.4. Comparison of the consistency measures: (a) The cumulative histogram of the mean angular derivation between the estimated vanishing points and the ground truth offered in the YUD database; (b) The cumulative histogram of the estimated focal length error using images in the YUD database. The ground truth of the focal length is 675 pixels. The extracted lines are taken as input of the algorithms.

than **CM1** for both calibrated and uncalibrated cameras. However, **CM1** can be used for the unifying camera model while **CM2** is only suitable for the perspective images. Besides, **CM1** is about two times faster to be evaluated than **CM2**.

One interesting observation from Fig.6.4 must be pointed out here: even without the refinement, the results of the RANSAC process have had high accuracy already. In Fig.6.4(a), the mean angular derivations of **R3_CM1** and **R3_CM2** are less than 3 degrees for all images in the dataset. In Fig.6.4(b), there are more than 80% of images with focal length error smaller than 60 pixels. The image size is 640×480 pixels in the YUD database. This demonstrates the benefit of applying Manhattan world characteristics in the RANSAC process. The benefit will be further demonstrated in the following experiments.

6.6.2 Comparison of the refinement methods

To compare the performance of different refinement algorithms, in this experiment we fix the consistency measure method as **CM2**. To discover

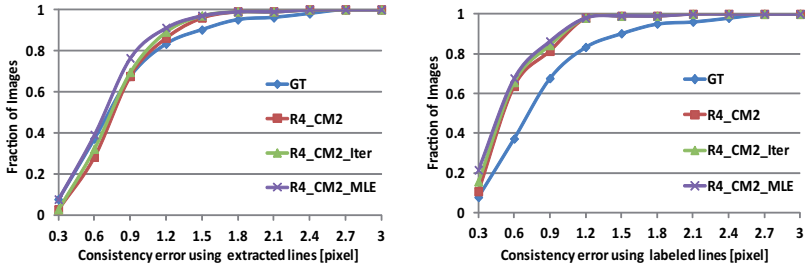


Figure 6.5. The cumulative histogram of the RMS consistency error of the estimated vanishing points and the focal length with respect to the labeled ground truth lines in the YUD database. (a) The vanishing points and the camera focal length are estimated from the extracted lines. (b) The ground truth lines in the database are taken as input of the algorithms. **CM2** is chosen as the consistency measure method (uncalibrated camera).

the influence of the different percentage of non-Manhattan lines, the extracted lines (which include a large amount of lines corresponding to non-Manhattan directions) and the ground truth lines in the YUD database are taken as input respectively. We assume the camera is uncalibrated, so **R4** is chosen in the RANSAC process. We compare the consistency error of the estimated vanishing points and the focal length with respect to the labeled lines (corresponding to three orthogonal directions). The Root Mean Square (RMS) of the consistency error is computed across all the ground truth lines in the image.

Fig.6.5 shows the cumulative histogram of the RMS of consistency errors. **GT** denotes the consistency error of the ground truth vanishing points and the focal length offered in the database. **R4_CM2** is the result of 4-line RANSAC process. **R4_CM2_Iter** and **R4_CM2_MLE** are the results after refining. One can see that the consistencies of tested algorithms are even better than that of **GT**. Similar observation is reported in the state-of-the-art work [MR11b; WH12; ZK12c]. This is due to the sub-optimal two-stage process which is employed to estimate the ground truth of three orthogonal vanishing points in the YUD database. When using the

Table 6.2. The computing time of the refinement step with Matlab.

	Using Extracted Lines			Using Labeled Lines		
	Min	Mean	Max	Min	Mean	Max
Iter(s)	0.034	0.123	0.481	0.027	0.113	0.590
MLE(s)	0.095	0.275	0.570	0.052	0.189	1.175

extracted lines, **R4_CM2_MLE** performs slightly better than **R4_CM2_-Iter** as shown in Fig.6.5(a), because **MLE** refines the RANSAC results by considering all the extracted lines with a well trained prior model while **Iter** only refines the inlier lines classified by the RANSAC process. When using the ground truth lines, **R4_CM2_MLE** and **R4_CM2_Iter** perform almost the same as shown in Fig.6.5(b), because in this case the inlier set classified by the RANSAC process almost includes all the ground truth lines.

Tab.6.2 reports the computing time of the refinement step by using **Iter** and **MLE** respectively. For each image, we run 100 times to get the average computing time. There are 102 images in the database. The minimum, mean and maximum of the average computing time are given. If the efficiency of the algorithm is also taken into consideration, then we prefer **Iter** to **MLE** because the latter takes almost double time with little accuracy improvement. Another shortcoming of **MLE** is that the prior model used in the algorithm should be learned in advance for different databases and/or different consistency measure methods.

One may notice that the **R4_CM2_MLE** algorithm presented here is very close to **R4L+MLE_fR** in [WH12]. Indeed, both of them use **MLE** to refine the 4-line RANSAC results and choose **CM2** as the consistency measure method. They should have similar performance. However, the biggest difference between them is that they use different approaches to solve the 4-line hypothesis. As pointed out in Sec.6.2 and detailed in Sec.6.4, there is a theoretical flaw of the approach in [WH12]. Besides, our framework solves the direction of lines in the camera frame instead of estimating vanishing points in the image plane directly, which is easily extended to the unifying camera model. We will further discuss the superiority of our framework in Sec.6.7.

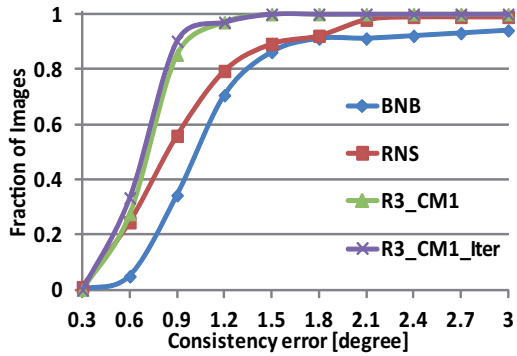


Figure 6.6. The cumulative histogram of the RMS consistency error of vanishing points with respect to the labeled ground truth lines in YUD. The vanishing points are estimated from the extracted lines. **CM1** is chosen as the consistency measure method (calibrated camera).

6.6.3 Comparison with the state-of-the-art methods

In this section, we compare the proposed algorithms with two groups of the state-of-the-art methods respectively. One group is for calibrated cameras and another group is for uncalibrated cameras. The algorithms for catadioptric cameras are compared separately.

Calibrated perspective camera

First, for calibrated cameras, there are two most recently reported algorithms featuring remarkable performance. **RNS** [MR11b] is a RANSAC based approach which solves the 3-line hypothesis by a polynomial solver. **BNB** [BSP12] employs a branch-and-bound procedure to globally optimize the rotation matrix formed by three orthogonal vanishing points. Since both algorithms choose **CM1** as the consistency measure, we compare them with **R3_CM1** and **R3_CM1_Iter**. The source codes of **RNS** and **BNB** are by the courtesy of their authors and the parameters are set as recommended values. The RMS consistency errors of the estimated vanishing points with respect to the labeled lines in the YUD database are

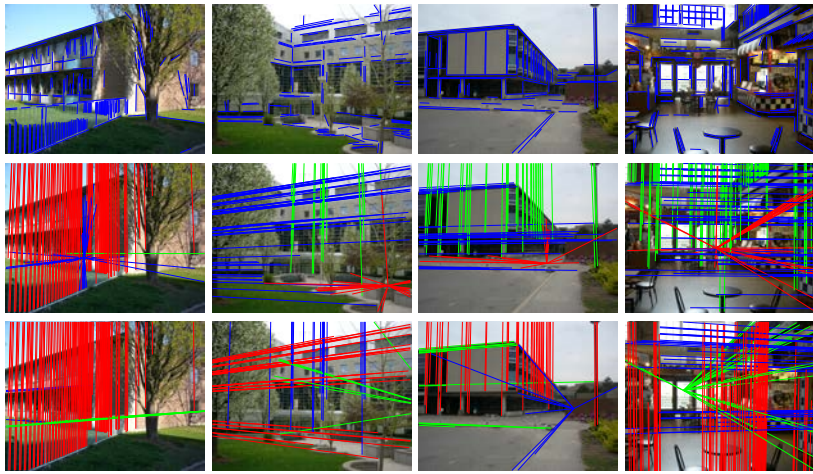


Figure 6.7. Some example images for which **BNB** fails to estimate one of the Manhattan directions. The first row shows the example images with extracted lines. The second row shows the line classification results of **BNB**. The third row shows the typical results of **R3_CM1_Iter** for these images.

compared as shown in Fig.6.6. Surprisingly, the global optimal approach **BNB** performs even worse than the RANSAC based approaches. The **BNB** algorithm is designed to maximize the number of inliers by searching the parameter space of rotation matrix. When there are many lines corresponding to non-Manhattan directions, the solution of rotation corresponding to the largest number of inliers may not be the best solution for the Manhattan directions. This can be seen from Fig.6.7 which shows some images for which **BNB** fails to estimate one of the Manhattan directions. Some typical results of **R3_CM1_Iter** for these images are also presented in Fig.6.7. The results of **R3_CM1_Iter** on the complete YUD database are downloadable on the website[1]. Our algorithms also perform better than **RNS** because of two reasons: (i) a better approach to solve the 3-line hypothesis (Sec.6.3); (ii) a better approach to generate and test the hypotheses (Sec.6.5).

[1] `http://sdrv.ms/15oWX9F`

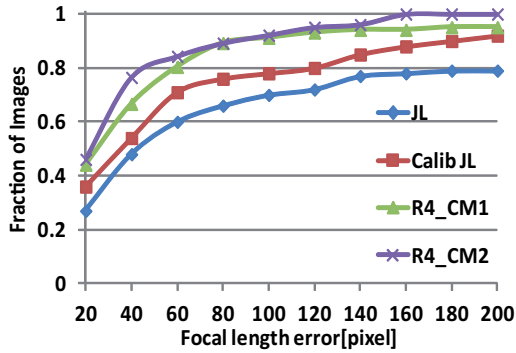


Figure 6.8. The cumulative histogram of the estimated focal length error using images in the YUD database. The ground truth of the focal length is 675 pixels. The extracted lines are taken as input of the algorithms (uncalibrated camera).

Uncalibrated perspective camera

For uncalibrated cameras, the comparison with the work in [WH12] has been detailed in Sec.6.6.2. In this experiment, we first compare our algorithms with the well established pipeline approach **JL** proposed by Tardif [Tar09]. **JL** is a J-Linkage based algorithm which estimates a set of vanishing points in the image. When the camera intrinsic parameters are known, the **Calib JL** algorithm is presented to find the most orthogonal triplet from the estimated vanishing point set. The source code of **JL** and **Calib JL** without the EM refinement function are released by the author. Hence, we compare them to **R4_CM1** and **R4_CM2** without the refinement step. The accuracy of the estimated focal length is evaluated as shown in Fig.6.8. The performance of **R4_CM1** and **R4_CM2** are even better than **Calib JL** which utilizes the knowledge of camera intrinsic parameters. The reason is that the vanishing point set estimated by **JL** does not enforce the orthogonality and **Calib JL** only searches the most orthogonal triplet from this defective candidate set.

Recently, Tretyak et al. [TBKL12] presented a parsing framework for the geometric analysis of an image taking in a man-made environment. The **GeoParsing** algorithm models the scene as a composition of geometric

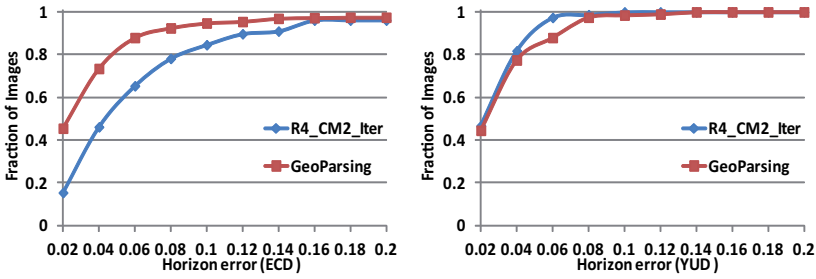


Figure 6.9. The cumulative histograms of the horizon error on the ECD and YUD databases (uncalibrated camera).

primitives spanning different layers from low level (edges) through mid-level (lines segments, lines and vanishing points) to high level (the zenith and the horizon). A more challenging ECD database is collected with a significant proportion of images violating the Manhattan assumption. In their experiment, the first 25 images are held out for training parameters. Then they verify the accuracy of **GeoParsing** by evaluating the horizon error which is defined as the maximum Euclidean distance between the estimated and the ground truth horizon within the image domain [$0 < x <$ image width], divided by the image height. **GeoParsing** chooses **CM2** as consistency measure method. We follow their experiment and test our **R4_CM2_Iter** on the ECD and YUD databases. The cumulative histograms of the horizon error are reported in Fig.6.9. It can be seen that for the challenging ECD database, **GeoParsing**, which is designed for the general scene, performs better than **R4_CM2_Iter**, which is designed for the Manhattan scene. As expected, **R4_CM2_Iter** performs better on the YUD database. Fig.6.10 shows the zenith and horizon estimation examples of **R4_CM2_Iter** on the ECD database. Note that, although it fails to estimate the horizon in some images, the algorithm can successfully detect the orthogonal structures as shown in the last column of Fig.6.10. The results on the complete ECD database are downloadable on the website[2].

[2] `http://sdrv.ms/15oY2yn`

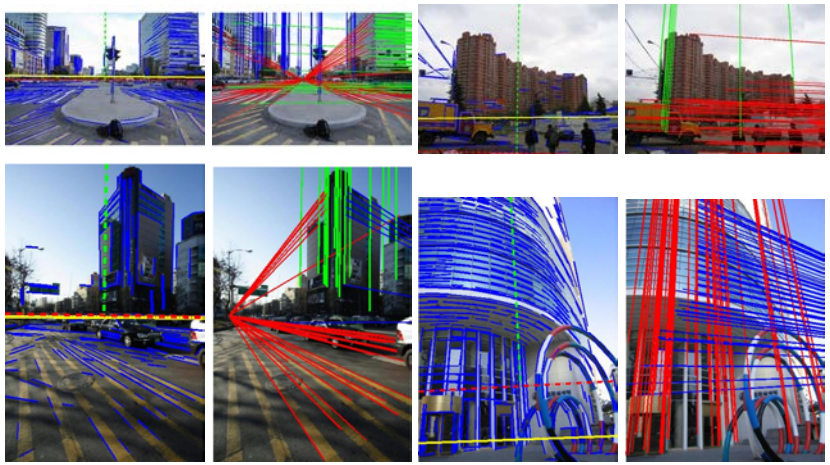


Figure 6.10. The zenith and horizon estimation examples of **R4_CM2_Iter** on the ECD database. The left two columns show the successful estimations and the right two columns show some failures. For each image pair, the first image shows the original image with extracted lines, the zenith and horizon imposed. The second image shows the line classification results. The ground truth horizon offered in the ECD database is depicted as solid yellow. The estimated horizon and zenith are drawn in dashed red and green lines respectively.

Catadioptric camera

There are only a few algorithms estimating vanishing points in the omnidirectional images. Among them, the 3-line RANSAC algorithm in [BP12] estimates vanishing points from three lines in which two lines are parallel and the third is orthogonal to them. The algorithm is only a special case of ours. In this section, our algorithm is compared to the rotation Sampling Approach (**SA**) proposed in [BDVK12] and the **BNB** algorithm proposed in [BSP12]. Since **CM2** is not suitable to measure the consistency in the omnidirectional image, we choose **CM1** as consistency measure method and test **R3_CM1**. The experiment is conducted on the image sequence from the Cold database which includes 1000 frames. The intrinsic parameters of the catadioptric camera are estimated by the calibration tool of

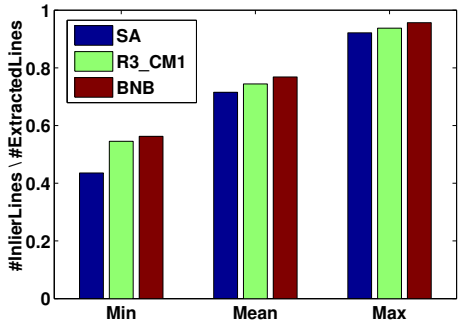


Figure 6.11. The clustered inlier ratios on the Cold database (catadioptric camera).

Barreto and Araujo [BA05] in advance.

The efficiency of **SA** is dependent upon the search space and the sampling rate. For example, if the search space is $\pm\beta$ degrees in each direction of the 3D space and the sampling rate is κ degrees, then the number of tested rotation hypotheses will be $(2\beta/\kappa + 1)^3$. In this experiment, we set $\beta = 10^\circ$ and $\kappa = 1^\circ$ corresponding to about 8000 rotation hypotheses in **SA**. For the **R3_CM1** algorithm, the maximum number of hypotheses is 100 as introduced in Sec.6.5. The **SA** algorithm is initialized by the result of **R3_CM1** for the first frame, then the rotation estimate of **SA** for the previous frame in the sequence is taken as current initial value. Since the ground truth of vanishing points and the line labels are not available in the database, the performance is evaluated by comparing the fraction of clustered inlier lines. All three algorithms use the same threshold $t_\varepsilon = 0.03$ to cluster the inliers by the **CM1** method. Fig.6.11 shows the minimum, mean and maximum inlier ratios clustered by **SA**, **R3_CM1** and **BNB** on the Cold database. It can be seen that **R3_CM1** tests a much smaller number of hypotheses than **SA** while clusters more inliers. The number of inliers clustered by **R3_CM1** is also close to **BNB** which detects the maximum number of inliers for each frame by the exhaustive global optimization.

The performance of our algorithm on the Rawseeds database will be presented in the navigation application (Sec.6.6.4). Some qualitative results

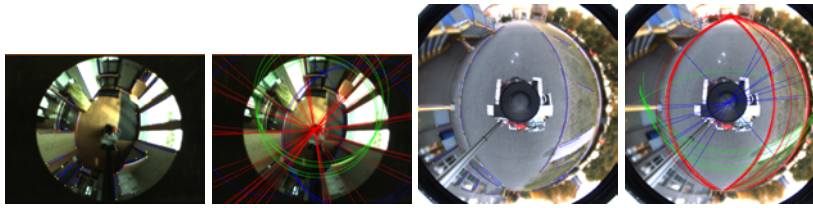


Figure 6.12. Example of vanishing point estimation and line classification results for omnidirectional images from the Cold database and the Rawseeds database.

are presented in Fig.6.1 and Fig.6.12 which show the example images with line classification results.

Computing Time

In the previous comparisons, the time performance of algorithms are not reported because the source codes of **JL**, **Calib JL** and **GeoParsing** are implemented as mex files (the mix of Matlab and C++ functions). It is hard to judge the time performance of algorithms with different software platforms. **RNS** and **BNB** are implemented with Matlab. In our experiment, **RNS** spends about double times than the Matlab version of **R3_CM1_Iter** to test the same number of hypotheses. The global optimal algorithm **BNB** spends a few hours to do a single run on the complete YUD database in our experiment. The computing time of **JL** and **Calib JL** is about hundreds of milliseconds per image. **GeoParsing** takes tens of seconds to analyze the geometry of a single image.

In order to demonstrate the efficiency of our algorithms, we report the computing time of the C++ versions of **R3_CM1** and **R3_CM1_Iter** on the YUD, Cold and Rawseeds database for calibrated cameras. We also report the computing time of **R4_CM1** and **R4_CM1_Iter** on the YUD and ECD database for uncalibrated cameras as given in Tab.6.3. The 3-line RANSAC based algorithms are more efficient for catadioptric cameras because there are less extracted lines in the omnidirectional images than in the perspective images. As expected, the 4-line RANSAC based algorithms for uncalibrated cameras take longer time than the 3-line based algorithms

Table 6.3. The minimum, mean and maximum computing time per image of the C++ versions on the databases in milliseconds. (platform: a 3.4GHz Intel(R) Core4 processor with 8GB of RAM)

3-line RANSAC for calibrated cameras									
	Perspective camera			Catadioptric camera					
	YUD			Cold			Rawseeds		
	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
R3_CM1 (ms)	0.199	0.541	1.074	0.074	0.271	0.350	0.100	0.241	0.359
R3_CM1_Iter (ms)	0.238	0.594	1.151	0.087	0.318	0.425	0.197	0.331	0.442
4-line RANSAC for uncalibrated perspective cameras									
	YUD			ECD					
	Min	Mean	Max	Min	Mean	Max			
R4_CM1 (ms)	0.366	0.966	2.001	0.496	1.583	3.653			
R4_CM1_Iter (ms)	0.673	1.746	3.156	1.082	2.370	4.932			

for calibrated cameras. The computing time on the challenging ECD database is longer than on the YUD database because more hypotheses are tested on the ECD database to find a good solution.

The line detection process will take a few milliseconds which is not included in the reported computing time to better demonstrate the efficiency of our vanishing point estimation algorithms. If the **MLE** is used to refine the RANSAC process, then the computing time is slightly longer than **Iter**. If **CM2** is chosen as the consistency measure method, then the computing time of the corresponding algorithms will be doubled.

6.6.4 Example of applications

There is a wide range of applications involving the vanishing point detection, such as image rectification [LZ98], image segmentation [WL11], scene understanding [FMRM10], single view reconstruction [LHK09], navigation [BDVK12; Kes+10] and so on. To demonstrate the performance of our algorithms in practical applications, we apply **R3_CM1_Iter** to the applications of single view reconstruction and navigation as examples.

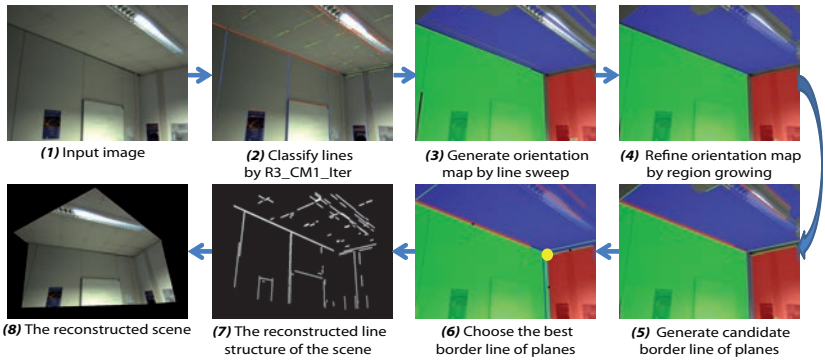


Figure 6.13. The framework of the single view reconstruction of a corner scene.

Single view reconstruction

With the knowledge of the scene (Manhattan world assumption), it is possible to reconstruct the 3D structure of the scene up to scale from a single view. In [LHK09], a general indoor world model under the Manhattan world assumption is designed. A vanishing point estimation and line classification algorithm is employed to generate the orientation map. Then the model hypotheses are evaluated according to the orientation map. Following this idea, we apply **R3_CM1_Iter** to an image of a room corner and reconstruct the scene. The framework is presented in Fig.6.13. The details of the third step about line sweep can be found in Sec.5.3 of [LHK09]. To accelerate the speed of plane segmentation, we refine the orientation map by a region growing algorithm which fills the unlabeled pixels with the label of their neighbor pixels within a certain range as shown in the fourth step. The border line of regions are chosen as the candidate border of planes as shown in the fifth step. For the scene of a room corner, the border of three planes should intersect at the corner. The consistency between the plane hypothesis and the refined orientation map is evaluated by counting the number of pixels with the same classification. The hypothesis resulting in the largest consistency score is chosen as the border line of planes. After the plane segmentation, each pixel in the image can be converted to its corresponding 3D plane, hence the line

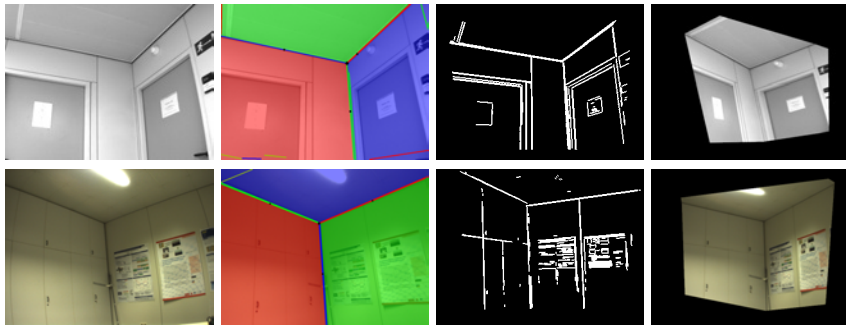


Figure 6.14. Single view reconstruction results of two corner scenes. The first column shows the input images. The second column shows the segmented planes. The last two columns show the reconstructed line structures and the reconstructed scenes, respectively.

structures and the scene are easily reconstructed. In Fig.6.14 we show two more single view reconstruction results of the corner scenes.

The reconstruction of a room corner is really a simple example. However, it can be integrated into some more complicated systems. For example, it is a good way to initialize the hand-held monocular SLAM system operating in the indoor Manhattan environments [ZK11].

Navigation

Attitude estimation is a fundamental step for various navigation tasks. The estimated vanishing points of **R3_CM1_Iter** are the Manhattan directions in the camera frame which equal to the camera attitude in the Manhattan world frame. To qualitatively illustrate the attitude estimation accuracy, we take an image sequence containing 1260 frames in our lab with a hand-held perspective camera. Since the ground truth of the camera attitude is not available, in Fig.6.15 we only shows the estimated attitude of the camera represented in Euler angles. The estimated camera attitude is used to animate a flying airplane in the world frame. Fig.6.16 shows an instant frame in the video. The video of the complete sequence is available on

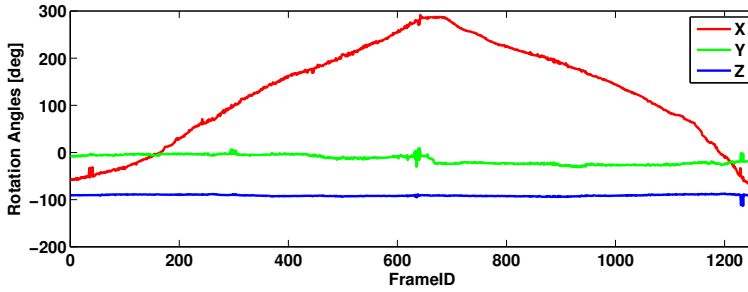


Figure 6.15. The estimated attitude of the hand-held camera moving in the indoor Manhattan world.

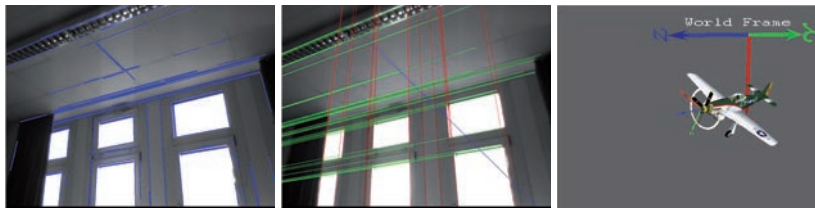


Figure 6.16. Illustration of animation using estimated attitude. The first image shows the original image with extracted lines imposed. The second image shows the line classification results. The estimated camera attitude in the Manhattan world is depicted by the airplane as shown in the animated image.

the website[3]. From Fig.6.15 and the video, it can be seen that the camera mainly rotates around the X axis of the world frame smoothly. The larger jitters around frame 640 and frame 1220 are because the corresponding images contains more non-Manhattan lines.

One may notice that, for each individual frame, three orthogonal directions are estimated, but the order of x, y and z directions is unknown. There are six possible permutations. In our implementation, the order of x, y and z is defaulted as vp_1, vp_2 and vp_3 in the first frame. Then

[3] http://sdrv.ms/15oZAsj

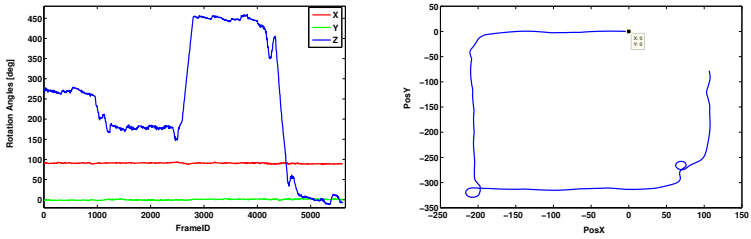


Figure 6.17. Estimated attitude and trajectory. The first plot shows the estimated attitude in Euler angle representation. The second plot shows the estimated robot trajectory.

for the rest of the frames, we choose the permutation which is closest to the previous attitude under the assumption of only small rotation between consecutive frames. There is one risk that, if the error of estimated vanishing points is too large in one frame, then the order of directions may be not consistent with the previous one because of the wrong permutation. We keep using this approach in our experiment to better demonstrate the performance of the RANSAC based **R3_CM1_Iter** algorithm. From Fig.6.15 and the attitude of the animated airplane in the video, it can be seen that the order of directions (x in red, y in green, and z in blue) is consistently estimated in the complete sequence, which means the error of the estimated vanishing points is limited in a small error bound. Of course, one can reduce the risk of breaking the consistency by determining the permutation based on a set of previous frames.

We also conduct the experiment on an outdoor omnidirectional image sequence from the Rawseeds database containing a few thousands frames. The sequence is taken by a robot which is driven with constant velocity $\boldsymbol{\nu}^c$ around a building. Given the attitude of the robot $\boldsymbol{R}_c^w$, the trajectory of the robot in the world $\boldsymbol{t}^w$ at frame k can be calculated as:

$$\boldsymbol{t}_k^w = \boldsymbol{t}_{k-1}^w + \boldsymbol{R}_{c\,k-1}^w * \boldsymbol{\nu}^c. \tag{6.18}$$

Since the robot moves along the building, the GPS signal is often blocked by the building. The ground truth trajectory measured by the GPS for

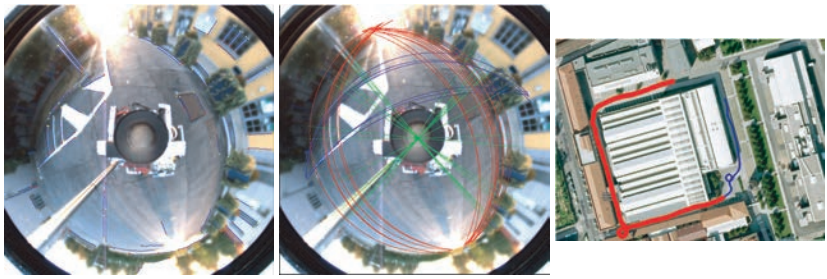


Figure 6.18. Illustration of trajectory estimation. The first image shows the image with extracted lines imposed. The second image shows the line classification results. The estimated trajectory of the robot denoted in red is imposed into the Google satellite map.

this sequence is rather fragmented. In our experiment, we simply set $\boldsymbol{v}^c = [1, 0, 0]^T$ and the trajectory is estimated up-to-scale. In Fig.6.17, we shows the estimated attitude of the robot platform and its trajectory. It can be seen that the robot platform rotates around the Z axis of world frame with fixed angle in X and Y directions when driving on the campus. To better demonstrate the accuracy of results, we impose the trajectory into the Google satellite map of the building area by similarity transformation. Fig.6.18 shows an instant frame in the sequence. The video of the complete sequence is available on the website[4]. It can be seen that shape of the trajectory is well estimated.

In the end, to quantitatively evaluate the accuracy of the estimated attitude by the **R3_CM1_Iter** algorithm, we mount a perspective camera on a pan-tilt platform which offers the ground truth of the camera attitude. We rotate the platform in our lab and record 50 frames. The movement of the platform is first a 360° sweep in pan direction with a fixed tilt angle, then another sweep with an increased tilt angle and so on. The estimated attitude and the ground truth are shown in Fig.6.19. To see the difference more easily, the errors in pan and tilt directions are also presented in this figure. It can be seen that the error bound for this sequence is less than 4 degrees. Doubtless, the error bound is also dependent upon the

[4] http://sdrv.ms/15oZP6u

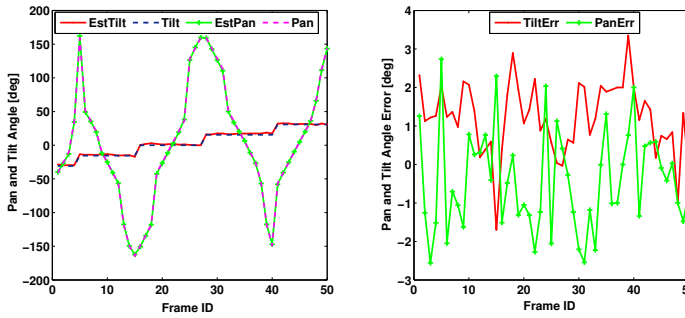


Figure 6.19. The movement of the pan-tilt platform and the estimated attitude errors in pan and tilt directions.

resolution of the image and the structure of the scene. In this experiment, the image resolution is 640×480 and our lab is subject to the Manhattan world assumption.

6.7 Summary and discussion

The knowledge of the environment enforces some constraints on the image primitives. For scenes under the Manhattan world assumption, we find that lines have the best information content to describe the scene structures. There is no direct constraint on points in the Manhattan world because their information level is too low. Planes have a higher level than lines. In the Manhattan world, they are subject to the parallel or orthogonal constraints, however, it is computationally expensive to segment planes in the image. Lines are easy to extract and are subject to the parallel or orthogonal constraints.

In this work, by applying the constraints on lines, we develop a framework to estimate the vanishing points of three orthogonal Manhattan directions. Due to the novel representation of line directions, our framework is well suited for the unifying camera model. It is also easily generalized to estimate the focal length and the vanishing points simultaneously for

uncalibrated cameras. A RANSAC based approach is proposed and the winner of the RANSAC process is the one resulting in the largest number of inlier lines. To determine the inlier lines with respect to the vanishing points, two consistency measure methods (**CM1** and **CM2**) are introduced and evaluated. **CM1** is faster to compute and is more suitable for the omnidirectional images, while **CM2** achieves slightly higher accuracy. To refine the RANSAC results, two refinement methods (**Iter** and **MLE**) are presented and compared. **Iter**, using only the inlier lines, is faster to compute and does not need the prior model from the training set, while **MLE**, using both inlier and outlier lines, achieves slightly higher accuracy. The comparisons with the state-of-the-art algorithms demonstrate the superiority of our approach which has high accuracy and efficiency. The example applications further demonstrate the performance of our approach. The implementation of our approach is available on our website[5].

One interesting observation from the experiment on the ECD database is that, although the framework of our algorithm is based on the Manhattan world assumption, the algorithm is able to detect a triplet of vanishing points which is close to orthogonal and dominant in the image when the Manhattan assumption is violated slightly.

One may think that the proposed algorithm is limited by the Manhattan world assumption while there are some algorithms without this limitation. Here, we want to point out that, without the knowledge of the scene, the vanishing points detected in the image offer little information. It cannot be used to calibrate the camera nor to estimate the attitude of the camera. If we do have knowledge of the scene, for example, the angle between three groups of lines (in a Manhattan world, they are right angles), then the framework of our algorithm can be easily adjusted to estimate the vanishing points of lines in such configurations. As discussed in case 2.b of Sec.4.3.2, for parallel lines, Eq.(6.6) will hold. For nonparallel lines $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ with known angle β, the term in Eq.(6.2) is adjusted as $\boldsymbol{v}_i \cdot \boldsymbol{v}_j = \|\boldsymbol{v}_i\| \, \|\boldsymbol{v}_j\| \, \cos\beta$. By combining equations of parallel and nonparallel lines, it is easy to solve the vanishing point estimation problem following our framework. If only one vanishing point is of interest in some applications, e. g., the zenith in the image or the vanishing point

[5] http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Lilian+Zhang

of the highway for a moving vehicle, then Eq.(6.6) is enough to solve the parallel line hypotheses, and the best solution is just the one with the largest number of inliers.

You can do anything, but not everything.

David Allen

Chapter 7

Conclusion

7.1 Summary

To make the achievements on line primitives be in parallel with the achievements on point primitives, this thesis is dedicated to a systematical study about line primitives and their applications in geometric computer vision. Techniques developed in this thesis are mainly built on projective geometry and photogrammetry. The thesis presents the theory and resulting algorithms in terms of 2D lines (line projections in images), 3D lines (scene structures) and the camera.

We started by matching lines extracted in the image pair as presented in Chapter 3. We built a robust and efficient line matching approach by combining the following three strategies: detecting lines in multi-scale space, constructing the Line Band Descriptor (LBD) to depict the local appearances of lines, and evaluating the pairwise geometric consistency. These strategies improve the matching performance when facing the following challenges: inaccurate locations of line endpoints, fragmentation of lines, lack of epipolar constraint, lack of distinctive appearance in low-texture scenes and instabilities for large image transformations.

After solving the line matching problem, we addressed the problem of camera pose estimation from 2D/3D line correspondences in Chapter 4 and the problem of Structure-from-Motion (SfM) based on 2D/2D line correspondences in Chapter 5. In the work of Chapter 4, we found that for 3D line primitives in special configurations, the properties of the special configurations can be employed to reduce the complexity of the camera pose estimation problem. Hence, in the last technique chapter (Chapter 6), we restricted lines in a Manhattan world and addressed the classical vanishing point estimation problem.

In Chapter 4, by investigating the symmetric property of line directions in space, we analyzed the solution of the Perspective-3-Line (P3L) problem. We presented a complete scenario for three spatial lines in all possible configurations, which unifies the previous work on the P3L problem for lines in special configurations. We also derived a general eighth order P3L polynomial, which is employed to develop the solution of the Perspective-n-Line (PnL) problem. The RPnL algorithm was proposed, including the following advantages: firstly, it stably retrieves the optimum of the solution with very little computational complexity and high accuracy; secondly, small line sets can be robustly handled to achieve highly accurate results and; thirdly, large line sets can be efficiently handled because the computational complexity of RPnL is O(n).

In Chapter 5, we addressed the SfM problem ranging from the representation of lines, their projections and the initialization procedure to the final adjustment. We developed the Cayley representation of spatial lines and derived a novel line projection function. We then employed a closed-form solution for the first image triplet, and developed an incremental initialization approach to initialize the motion and structure parameters which were further optimized by the Sparse Bundle Adjustment (SBA). An Augmented Reality (AR) system was developed by employing the algorithms proposed in Chapter 3, 4 and 5, which runs smoothly in real-time with a hand-held camera.

In Chapter 6, we addressed the problem of vanishing point estimation and line classification in a Manhattan world. Following the framework in Chapter 4, we parametrized the vanishing point in the camera frame with one unknown parameter and estimated three orthogonal vanishing points simultaneously from the hypothesis of a line triplet. The proposed algorithm thoroughly takes the advantage of the Manhattan world characteristic that lines should be orthogonal or parallel to each other. We also generalized the algorithm to estimate vanishing points in the omnidirectional images and uncalibrated perspective images. In the end, we demonstrated the performance of the proposed algorithm in two typical applications: single view reconstruction and robot navigation.

To summarize, in this thesis we focused on the geometric properties of line primitives. The proposed algorithms can be employed to solve the geometric computer vision problems related to line primitives in structured

environments. They can also be integrated into point-based systems to improve the robustness and the applicability of systems because the two types of feature primitives provide complementary information about the scene structures.

7.2 Future work

Although promising results have been obtained, our approaches still have their limitations and open questions. The questions, which can be approached directly, have been discussed in the summary section of each technique chapter. In the following we propose a long term direction for future research.

- **Simultaneous Localization and Mapping (SLAM)**: SLAM can be thought of as a chicken or egg problem: an unbiased map is needed for localization while an accurate pose estimate is needed to build that map. Localization is to answer the question: *Where am I?* In contrast to this, mapping is to answer the question: *What does the world look like?* Dependent on the application, location may refer simply to the position of the system or may also include its orientation; and the map can be either a geometrically consistent map or a topological map.

Inspired by the efficiency of the proposed algorithms, one of the major future work is to integrate all the proposed algorithms into a SLAM system which has a strong requirement on the real-time performance. The SLAM system can be initialized by our SfM algorithm or by our single view reconstruction algorithm. Then we can employ our line matching algorithm and camera pose estimation algorithm in the tracking process of the SLAM system. The sparse bundle adjustment framework can be employed to refine the map when necessary. To build a successful SLAM system, the data association and map management processes should be designed as well. Angeli et al. [AFDM08] and Cummins and Newman [CN11] offered two approaches for the loop-closure detection based on visual information. Concerning map management, promising solutions can be found in [DWDWB02; SDMK11; LFP11]. The main challenge is to integrate all the ingredients properly to make the whole system run consistently and efficiently.

The SLAM system operating in the Manhattan world is of special interest because the rotation (or the attitude) of the platform can be estimated with a limited error bound as explained in Chapter 6. Besides, in order to improve the robustness and the applicability of SLAM systems, it is worth to combine point primitives and line primitives. Furthermore, the Inertial Measurement Unit (IMU) may also be integrated into the visual SLAM system to improve the robustness of the system, especially when the image primitives are deficient in a short term, e. g., camera viewing at a white wall.

• **Understanding of structured scenes**: SLAM does not cover the complete story of a fully autonomous robot. Many questions remain open: *How does a robot know what is where? How can a robot read?* And a more challenging question: *What is happening in the scene?* These questions are within the category of scene understanding, which is a long-lasting dream of artificial intelligence research.

In Chapter 6, we proposed a vanishing point estimation algorithm and demonstrated its application in single view reconstruction which segments planes and the room corner. Under the Manhattan world assumption, we can employ the algorithm to develop an indoor scene understanding system which segments the floor, wall and ceiling surfaces as in [LHK09; FMR11; SU12]. Based on the basic segmentation, we can then further infer where the furniture and the free space are. For outdoor urban environments, we may develop a visual system to segment buildings, roads, pedestrians, vehicles, and so on. One promising tool for addressing the scene understanding problem is machine learning [Bis06], which is definitely a long term direction of our future research.

Appendix A

Detailed derivations

A.1 Computation of the angles in the 3-line junction

In Sec.4.3.2, we discussed the solution of the P3L problem for three lines in case 2.(d): No lines are parallel in space while their directions are linearly dependent and their projections form a junction as shown in Fig.A.1(a). γ_{ij} is the angle between lines $\boldsymbol{qp}'_i$ and $\boldsymbol{qp}'_j$, which can be measured in the virtual image plane Π'. β_{ij} is the angle between line directions $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, which is known in the world frame. δ is the angle $\angle \boldsymbol{p}'_0\boldsymbol{b}Q$ and δ' is the angle $\angle \boldsymbol{p}'_0\boldsymbol{bq}$. In this appendix, we give the solution of δ and δ' as in [Cag93].

In [Cag93], the projective invariants are introduced and a linear relationship between cotangent values are derived. Based on this linear relationship, in the 3-line junction case, we have

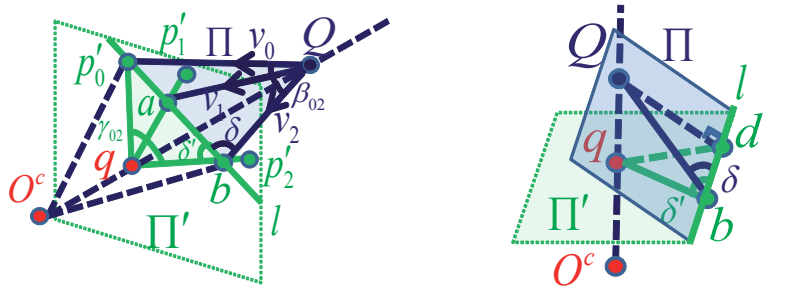
$$\cot\gamma_{12} = k\ \cot\beta_{12} + c, \quad \cot\gamma_{02} = k\ \cot\beta_{02} + c. \tag{A.1}$$

in which k and c can be computed from the cotangent values of γ_{ij} and β_{ij}. For δ' and δ, they are also subject to this linear relationship, i. e.,

$$\cot\delta' = k\ \cot\delta + c. \tag{A.2}$$

On the other hand, since $O^c\boldsymbol{q}$ is the normal of the virtual image plane, the cotangent values δ' and δ are subject to the homogeneous constraint:

$$\cot\delta' = k'\ \cot\delta. \tag{A.3}$$



(a) 3-line junction and its projection (b) Relationship between δ and δ'

Figure A.1. Illustration of angles in the 3-line junction

A short derivation of Eq.(A.3) is given in the following. We first find a point $\boldsymbol{d}$ on the intersection line $\boldsymbol{l}$ which makes $Q\boldsymbol{d}$ orthogonal to $\boldsymbol{l}$ as shown in Fig.A.1(b), then connect the points $\boldsymbol{q}$ and $\boldsymbol{d}$ in the virtual image plane. Since O^cQ is the normal of the virtual image plane, we have $O^cQ \perp \boldsymbol{l}$. Considering the plane passing through Q, $\boldsymbol{q}$ and $\boldsymbol{d}$, we have $\boldsymbol{l} \perp Q\boldsymbol{q}$ and $\boldsymbol{l} \perp Q\boldsymbol{d}$, so $\boldsymbol{l} \perp \boldsymbol{qd}$. Now, considering the right triangle $\triangle\boldsymbol{qdb}$ and $\triangle Q\boldsymbol{db}$, we have

$$\cot\delta' = \frac{|\boldsymbol{bd}|}{|\boldsymbol{qd}|}, \quad \cot\delta = \frac{|\boldsymbol{bd}|}{|Q\boldsymbol{d}|}. \tag{A.4}$$

So, the cotangent values of δ' and δ are subject to Eq.(A.3). Actually, $k' = \frac{|Q\boldsymbol{d}|}{|\boldsymbol{qd}|} = \frac{1}{\cos\phi}$ in which ϕ is the angle between plane Π and Π'. It is easy to see $|k'| \geqslant 1$.

By combining Eq.(A.2) and (A.3), we have

$$k' = k + \frac{c}{\cot\delta}. \tag{A.5}$$

For a generic line $\boldsymbol{l}_g$ passing through the point Q in plane Π, its projection in the virtual image plane Π' is $\boldsymbol{l}'_g$ passing through the point $\boldsymbol{q}$. Supposing the angle between $\boldsymbol{l}_g$ and $Q\boldsymbol{b}$ be β_g in plane Π and the angle between $\boldsymbol{l}'_g$ and $\boldsymbol{qb}$ be γ_g in plane Π', then the angle between $\boldsymbol{l}_g$ and $\boldsymbol{l}$ is $\delta - \beta_g$ and the angle between $\boldsymbol{l}'_g$ and $\boldsymbol{l}$ is $\delta' - \gamma_g$. Based on Eq.(A.2) and

(A.3), we have

$$\cot\gamma_g = k\ \cot\beta_g + c, \quad \cot(\delta' - \gamma_g) = k'\ \cot(\delta - \beta_g); \tag{A.6}$$

According to the difference formula of cotangent values $\cot(A-B) = \frac{\cot A\ \cot B+1}{\cot A - \cot B}$, we have

$$\frac{\cot\delta'\ \cot\gamma_g + 1}{\cot\delta' - \cot\gamma_g} = k'\ \frac{\cot\delta\ \cot\beta_g + 1}{\cot\delta - \cot\beta_g} \tag{A.7}$$

Since Eq.(A.6) holds for a generic line $\boldsymbol{l}_g$, we can choose it to make $\cot\beta_g = 0$, then $\cot\gamma_g = c$. Combing with Eq.(A.2) and (A.5), we can rewrite Eq.(A.7) as:

$$\frac{(k\ \cot\delta + c)\ c + 1}{k\ \cot\delta + c - c} = (k + \frac{c}{\cot\delta})\frac{1}{\cot\delta}. \tag{A.8}$$

So, a quadratic equation of $\cot\delta$ is built:

$$k\ c\ \cot^2\delta + (c^2 + 1 - k^2)\cot\delta - k\ c = 0. \tag{A.9}$$

Among the two resulting solutions for $\cot\delta$, we keep the one which satisfies the constraint $|k'| = |k + \frac{c}{\cot\delta}| \geqslant 1$. After solving δ, it is easy to compute δ' from Eq.(A.2).

A.2 Computation of Jacobian matrices

This appendix deals with the computation of Jacobian matrices and the update of parameters during the sparse bundle adjustment as introduced in Sec.5.6. Since the Jacobian matrix with respect to the camera motion parameters is nothing special and can be easily computed from the line observation model, here we only give the derivation of Jacobian matrix with respect to the line parameters. The observation is the distance $\boldsymbol{d} = (d_s, d_e)$ as defined in Eq.(5.13) and the update parameter of a line is the 4-vector $\boldsymbol{\zeta} = (\omega, s_x, s_y, s_z)^T$ as defined in Eq.(5.5).

First according to Eq.(5.13), we can compute the Jacobian matrix of the

distance $\boldsymbol{d} = (d_s, d_e)$ with respect to the image line $\boldsymbol{\ell} = (l_x, l_y, l_z)^T$ as

$$\left[\boldsymbol{J}_{23}|_{\boldsymbol{\ell}}^{\boldsymbol{d}}\right]^T = \lambda \begin{bmatrix} x_s(l_x^2+l_y^2) - \boldsymbol{\ell}^T\boldsymbol{p}_s l_x & x_e(l_x^2+l_y^2) - \boldsymbol{\ell}^T\boldsymbol{p}_e l_x \\ y_s(l_x^2+l_y^2) - \boldsymbol{\ell}^T\boldsymbol{p}_s l_y & y_e(l_x^2+l_y^2) - \boldsymbol{\ell}^T\boldsymbol{p}_e l_y \\ (l_x^2+l_y^2) & (l_x^2+l_y^2) \end{bmatrix}, \tag{A.10}$$

where $\lambda = (l_x^2+l_y^2)^{-3/2}$, the endpoints $\boldsymbol{p}_s = [x_s, y_s, 1]^T$ and $\boldsymbol{p}_e = [x_e, y_e, 1]^T$. Then according to Eq.(5.9), compute the Jacobian matrix of the image line $\boldsymbol{\ell}$ with respect to the Plücker coordinates of the 3D line $(\boldsymbol{m}, \boldsymbol{v})$ as:

$$\boldsymbol{J}_{33}|_{\boldsymbol{m}}^{\boldsymbol{\ell}} = cof(\boldsymbol{K})\boldsymbol{R}, \quad \boldsymbol{J}_{33}|_{\boldsymbol{v}}^{\boldsymbol{\ell}} = -cof(\boldsymbol{K})\boldsymbol{R}[\boldsymbol{c}]_\times. \tag{A.11}$$

Now, based on the relationship between the Plücker coordinates and the Cayley parametrization (Eq.5.4), we can compute the Jacobian matrix of $(\boldsymbol{m}, \boldsymbol{v})$ with respect to the update parameter of the line $\boldsymbol{\zeta} = (\omega, s_x, s_y, s_z)^T$ as:

$$\boldsymbol{J}_{34}|_{\boldsymbol{\zeta}}^{\boldsymbol{m}} = [\boldsymbol{q_2},\ \omega\ \partial\boldsymbol{q_2}/\partial\boldsymbol{s}], \quad \boldsymbol{J}_{34}|_{\boldsymbol{\zeta}}^{\boldsymbol{v}} = [\boldsymbol{0},\ \partial\boldsymbol{q_1}/\partial\boldsymbol{s}], \tag{A.12}$$

where $\partial\boldsymbol{q_1}/\partial\boldsymbol{s}$ and $\partial\boldsymbol{q_2}/\partial\boldsymbol{s}$ can be easily computed from Eq.(5.3). Finally, by combining equations (A.10), (A.11) and (A.12), we can get the Jacobian matrix $\boldsymbol{J}_{24}|_{\boldsymbol{\zeta}}^{\boldsymbol{d}}$ of the observation with respect to the update parameter of the line as:

$$\boldsymbol{J}_{24}|_{\boldsymbol{\zeta}}^{\boldsymbol{d}} = \boldsymbol{J}_{23}|_{\boldsymbol{\ell}}^{\boldsymbol{d}} \left(\boldsymbol{J}_{33}|_{\boldsymbol{m}}^{\boldsymbol{\ell}}\ \boldsymbol{J}_{34}|_{\boldsymbol{\zeta}}^{\boldsymbol{m}} + \boldsymbol{J}_{33}|_{\boldsymbol{v}}^{\boldsymbol{\ell}}\ \boldsymbol{J}_{34}|_{\boldsymbol{\zeta}}^{\boldsymbol{v}}\right). \tag{A.13}$$

During each SBA iteration, we estimate the increment of parameters after the computation of Jacobian matrix. Here, we again omit the update of the camera motion, and only show the rule of line update. First, we update the Cayley parameter of a line as $\boldsymbol{\zeta}_{k+1} = \boldsymbol{\zeta}_k + \delta\boldsymbol{\zeta}$. Then according to Eq.(5.3) and Eq.(5.4), we compute the new Plücker coordinates of the 3D line $(\boldsymbol{m}_{k+1}, \boldsymbol{v}_{k+1})$.

Bibliography

[AD03] Adnan Ansar and Kostas Daniilidis. “Linear pose estimation from points or lines”. In: *TPAMI* 25 (2003), pp. 282–296.

[AF87] Nicholas Ayache and Bernard Faverjon. “Efficient registration of stereo images by matching graph descriptions of edge segments”. In: *IJCV* 1.2 (1987), pp. 107–131.

[AFDM08] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. “A fast and incremental method for loop-closure detection using bags of visual words”. In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 1027–1037.

[ALC05] D. G. Aguilera, J. Gomez Lahoz, and J. Finat Codes. “A new method for vanishing points detection in 3D reconstruction from a single view”. In: *ISPRS*. 2005.

[ANL08] James L. Crowley Amaury Negre and Christian Laugier. “Scale invariant segment detection and tracking”. In: *ISER*. 2008.

[AT11] Cuneyt Akinlar and Cihan Topal. “EDLines: a real-time line segment detector with a false detection control”. In: *Pattern Recognition Letters* 32.13 (2011), pp. 1633–1642.

[AW06] Gee Andrew and Mayol Cuevas Walterio. “Real-time model-based SLAM using line segments”. In: *ISVC*. 2006, pp. 354–363.

[BA05] J.P. Barreto and H. Araujo. “Geometric properties of central catadioptric line images and their application in calibration”. In: *TPAMI* 27.8 (2005), pp. 1327–1333.

[BDVK10] J. C. Bazin, C. Demonceaux, P. Vasseur, and I. S. Kweon. “Motion estimation by decoupling rotation and translation in catadioptric vision”. In: *CVIU* 114.2 (2010), pp. 254–273.

[BDVK12] Jean Charles Bazin, Cédric Demonceaux, Pascal Vasseur, and Inso Kweon. "Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment". In: *IJRR* 31.1 (2012), pp. 63–81.

[BETG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. "SURF: speeded up robust features". In: *CVIU* 110.3 (2008), pp. 346–359.

[BFG05] Herbert Bay, Vittorio Ferrari, and Luc Van Gool. "Wide-baseline stereo matching with line segments". In: *CVPR*. 2005, pp. 329–336.

[Bis06] Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer-Verlag New York, Inc., 2006.

[Ble09] Gabriele Bleser. "'Towards visual-inertial SLAM for mobile augmented reality'". PhD thesis. University of Kaiserslautern, 2009.

[BN98] S. Baker and S.K. Nayar. "A theory of catadioptric image formation". In: *ICCV*. 1998, pp. 35–42.

[BP12] Jean Charles Bazin and Marc Pollefeys. "3-line RANSAC for orthogonal vanishing point detection". In: *IROS*. 2012, pp. 4282–4287.

[BS04] Adrien Bartoli and Peter Sturm. "The 3D line motion matrix and alignment of line reconstructions". In: *IJCV* 57.3 (2004), pp. 159–178.

[BS05] Adrien Bartoli and Peter Sturm. "Structure from motion using lines: representation, triangulation and bundle adjustment". In: *CVIU* 100.3 (2005), pp. 416–441.

[BSP12] Jean-Charles Bazin, Yongduek Seo, and Marc Pollefeys. "Globally optimal line clustering and vanishing point estimation in Manhattan world". In: *CVPR*. 2012.

[Cag93] Vincenzo Caglioti. "The planar three line junction perspective problem with application to the recognition of polygonal patterns". In: *PR* 26.11 (1993), pp. 1603–1618.

[CDR99] R Cipolla, Tom Drummond, and D Robertson. "Camera calibration from vanishing points in images of architectural scenes". In: *BMVC*. 1999, pp. 382–392.

[Cer+09] Simone Ceriani, Giulio Fontana, Alessandro Giusti, Daniele Marzorati, Matteo Matteucci, Davide Migliore, Davide Rizzi, Domenico G. Sorrenti, and Pierluigi Taddei. "Raw seeds ground truth collection systems for indoor self-localization and mapping". In: *Autonomous Robots* 27.4 (2009), pp. 353–371.

[CH99] Stephane Christy and Radu Horaud. "Iterative pose computation from line correspondences". In: *CVIU* 73.1 (1999), pp. 137–144.

[Che91] H.H. Chen. "Pose determination from line-to-plane correspondences: existence condition and closed-form solutions". In: *TPAMI* 13.6 (1991), pp. 530–541.

[Chm05] Leszek J. Chmielewski. "Scale and rotation invariance of the evidence accumulation-based line detection algorithm". In: *CORES*. 2005, pp. 363–370.

[CKP95] W.J. Christmas, J. Kittler, and M. Petrou. "Structural matching in computer vision using probabilistic relaxation". In: *TPAMI* 17.8 (1995), pp. 749–764.

[CLK09] M. Chandraker, Jongwoo Lim, and D. Kriegman. "Moving in stereo: efficient structure and motion using lines". In: *ICCV*. 2009, pp. 1741–1748.

[CLO05] David A. Cox, John Little, and Donal OShea. Using Algebraic Geometry. Second. Springer, 2005.

[CLSF10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF: binary robust independent elementary features". In: *ECCV*. 2010.

[CN11] Mark Cummins and Paul Newman. "Appearance-only SLAM at large scale with FAB-MAP 2.0". In: *IJRR* 30.0 (2011), pp. 1100–1123.

[CSSP92] James L. Crowley, Patrick Stelmaszyk, Thomas Skordas, and Pierre Puget. “Measurement and integration of 3D structures by tracking edge lines”. In: *IJCV* 8.1 (1992), pp. 29–52.

[CY03] James M. Coughlan and Alan L. Yuille. “Manhattan world: orientation and outlier detection by bayesian inference”. In: *Neural Computation* 15.5 (2003), pp. 1063–1088.

[CY99] James M. Coughlan and A. L. Yuille. “Manhattan world: compass direction from a single image by bayesian inference”. In: *ICCV*. 1999, pp. 941–947.

[DD05] P. David and D. DeMenthon. “Object recognition in high clutter images using line features”. In: *ICCV*. 2005, pp. 1581–1588.

[DEE08] Patrick Denis, James H. Elder, and Francisco J. Estrada. “Efficient edge-based methods for estimating Manhattan frames in urban imagery”. In: *ECCV*. 2008, pp. 197–210.

[DF90] Rachid Deriche and Olivier Faugeras. “Tracking line segments”. In: *Image and Vision Computing* 8.4 (1990), pp. 261–270.

[DRLR89] M. Dhome, M. Richetin, J.-T. Lapreste, and G. Rives. “Determination of the attitude of 3D objects from a single perspective view”. In: *TPAMI* 11.12 (1989), pp. 1265–1278.

[DWDWB02] Gamini Dissanayake, StefanB. Williams, Hugh Durrant-Whyte, and Tim Bailey. “Map management for efficient simultaneous localization and mapping (SLAM)”. In: *Autonomous Robots* 12.3 (2002), pp. 267–286.

[ED06] Ethan Eade and Tom Drummond. “Edge landmarks in monocular SLAM”. In: *BMVC*. 2006, pp. 7–16.

[ERB11] Kurt Konolige Ethan Rublee Vincent Rabaud and Gary Bradski. “ORB: an efficient alternative to SIFT or SURF”. In: *ICCV*. 2011.

[FB81] MA Fischler and RC Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[FFG09] M. Farenzena, A. Fusiello, and R. Gherardi. "Structure-and-motion pipeline on a hierarchical cluster tree". In: *ICCV Workshop*. 2009, pp. 1489–1496.

[FFS10] Philip Feinsilver, Uwe Franz, and René Schott. "Computing coordinates on lie groups". In: *International Journal of Pure and Applied Mathematics* 60.4 (2010), pp. 371–381.

[FM95] Olivier D. Faugeras and B. Mourrain. "On the geometry and algebra of the point and line correspondences between N images". In: *ICCV*. 1995, pp. 951–956.

[FMR11] Alex Flint, D. Murray, and I. Reid. "Manhattan scene understanding using monocular, stereo, and 3D features". In: *ICCV*. 2011, pp. 2228–2235.

[FMRM10] Alex Flint, C. Mei, I. Reid, and D. Murray. "Growing semantically meaningful models for visual SLAM". In: *CVPR*. 2010, pp. 467–474.

[Foe10] W. Foerstner. "Optimal vanishing point detection and rotation estimation of single images of a legolandscene". In: *ISPRS*. 2010.

[FWH10] Bin Fan, Fuchao Wu, and Zhanyi Hu. "Line matching leveraged by point correspondences". In: *CVPR*. 2010, pp. 390–397.

[Gal02] Andrew C. Gallagher. "A ground truth based vanishing point detection algorithm". In: *PR* 35.7 (2002), pp. 1527–1543.

[GD01] Christopher Geyer and Kostas Daniilidis. "Catadioptric projective geometry". In: *IJCV* 45.3 (2001), pp. 223–243.

[GHTC03] Xiaoshan Gao, Xiaorong Hou, Jianliang Tang, and Hangfei Cheng. "Complete solution classification for the perspective-three-point problem". In: *TPAMI* 25.8 (2003), pp. 930–943.

[GJMR10] R.G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. "LSD: a fast line segment detector with a false detection control". In: *TPAMI* 32.4 (2010), pp. 722–732.

[GN12] Mohit Gupta and Shree K. Nayar. "Micro phase shifting". In: *CVPR*. 2012, pp. 1–8.

[God97] James Samuel Goddard. "'Pose and motion estimation from vision using dual quaternion-based extended kalman filtering'". PhD thesis. University of Tennessee, 1997.

[Hin+10] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. "Dominant orientation templates for real-time detection of texture-less objects". In: *CVPR*. 2010, pp. 2257–2264.

[HK09] Richard I. Hartley and Fredrik Kahl. "Global optimization through rotation space search". In: *IJCV* 82.1 (2009), pp. 64–79.

[HML02] Ayman F. Habib, Michel Morgan, and Young-Ran Lee. "Bundle adjustement with self-calibration using straight lines". In: *The Photogrammetric Record* 17.100 (2002), pp. 635–650.

[HR11] J.A. Hesch and Stergios I. Roumeliotis. "A direct least-squares (DLS) method for PnP". In: *ICCV*. 2011, pp. 383–390.

[HS89] Radu Horaud and Thomas Skordas. "Stereo correspondence through feature grouping and maximal cliques". In: *TPAMI* 11.11 (1989), pp. 1168–1180.

[HS97] J. Heikkila and O. Silven. "A four-step camera calibration procedure with implicit image correction". In: *CVPR*. 1997, pp. 1106–1112.

[HZ04] R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Second. Cambridge University Press, 2004.

[Kes+10] C. Kessler, C. Ascher, N. Frietsch, M. Weinmann, and G.F. Trommer. "Vision-based attitude estimation for indoor navigation using vanishing points and lines". In: *PLANS*. 2010, pp. 310–318.

[KH94] Rakesh Kumar and Allen R. Hanson. "Robust methods for estimating pose and a sensitivity analysis". In: *CVGIP: Image Understanding* 60.3 (1994), pp. 313–342.

[KL10] Hyunwoo Kim and Sukhan Lee. "Wide-baseline image matching based on coplanar line intersections". In: *IROS*. 2010, pp. 1157–1164.

[KM07] G. Klein and D. Murray. "Parallel tracking and mapping for small AR workspaces". In: *ISMAR*. 2007, pp. 225–234.

[Kra99] Steven George Krantz. Handbook of Complex Variables. A product of Birkhaeuser Boston, 1999.

[KSS11] L. Kneip, D. Scaramuzza, and R. Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *CVPR*. 2011, pp. 2969–2976.

[KZ02] Jana Kosecka and Wei Zhang. "Video compass". In: *ECCV*. 2002, pp. 657–673.

[LA09] M.I. A. Lourakis and A.A. Argyros. "SBA: a software package for generic sparse bundle adjustment". In: *ACM Trans. Math. Software* 36.1 (2009), pp. 1–30.

[LCT11] Jianzhuang Liu, Yu Chen, and Xiaoou Tang. "Decomposition of complex line drawings with hidden lines for 3D planar-faced manifold object reconstruction". In: *TPAMI* 33.1 (2011), pp. 3–15.

[LFP11] Jongwoo Lim, J.-M. Frahm, and M. Pollefeys. "Online environment mapping". In: *CVPR*. 2011, pp. 3489–3496.

[LH05] M. Leordeanu and M. Hebert. "A spectral technique for correspondence problems using pairwise constraints". In: *ICCV*. 2005, pp. 1482–1489.

[LH88a] Yuncai Liu and Thomas S. Huang. “A linear algorithm for motion estimation using straight line correspondences”. In: *ICPR*. 1988, pp. 213–219.

[LH88b] Yuncai Liu and Thomas S. Huang. “Estimation of rigid body motion using straight line correspondences”. In: *CVGIP* 43.1 (1988), pp. 37–52.

[LHD88] Seppo Linnainmaa, David Harwood, and Larry S. Davis. “Pose determination of a three dimensional object using triangle paris”. In: *TPAMI* 10.5 (1988), pp. 634–647.

[LHF90] Y. Liu, T.S. Huang, and O.D. Faugeras. “Determination of camera location from 2D to 3D line and point correspondences”. In: *TPAMI* 12.1 (1990), pp. 28–37.

[LHK09] David Changsoo Lee, Martial Hebert, and Takeo Kanade. “Geometric reasoning for single image structure recovery”. In: *CVPR*. 2009, pp. 2136–2143.

[LHO00] M I A Lourakis, S T Halkidis, and S C Orphanoudakis. “Matching disparate views of planar surfaces using projective invariants”. In: *Image and Vision Computing* 18.9 (2000), pp. 673–683.

[Lie01] D. Liebowitz. “‘Camera calibration and reconstruction of geometry from images’”. PhD thesis. University of Oxford, Dept. Engineering Science, 2001.

[LMLK94] E. Lutton, H. Maitre, and J. Lopez-Krahe. “Contribution to the determination of vanishing points using hough transform”. In: *TPAMI* 16.4 (1994), pp. 430–438.

[LMNF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “EPnP: an accurate O(n) solution to the PnP problem”. In: *IJCV* 81.2 (2009), pp. 155–166.

[LMSY11] Rich Lehoucq, Kristi Maschhoff, Danny Sorensen, and Chao Yang. “Arpack software”. In: `http://www.caam.rice.edu/software/ARPACK/`. 2011.

[Low04] David G. Lowe. “Distinctive image features from scale-invariant keypoints”. In: *IJCV* 60.2 (2004), pp. 91–110.

[LSH11] Marius Leordeanu, Rahul Sukthankar, and Martial Hebert. "Unsupervised learning for graph matching". In: *IJCV* (2011), pp. 1–18.

[LX11] Shiqi Li and Chi Xu. "A stable direct solution of perspective-three-point problem". In: *IJPRAI* 25.5 (2011), pp. 627–642.

[LXX12] Shiqi Li, Chi Xu, and Ming Xie. "A robust O(n) solution to the perspective-n-point problem". In: *TPAMI* 34.7 (2012), pp. 1444–1450.

[LZ98] D. Liebowitz and A. Zisserman. "Metric rectification for perspective images of planes". In: *CVPR*. 1998, pp. 482–488.

[LZOY10] Chang Liu, Feng Zhu, Jinjun Ou, and Yong Yu. "Z-shaped perspective-three-line problem's unique solution conditions". In: *Intelligent Networks and Intelligent Systems, International Workshop on*. 2010, pp. 132–135.

[Mei07] Christopher Mei. "'Laser-augmented omnidirectional vision for 3D localisation and mapping'". PhD thesis. INRIA Sophia Antipolis, Project-team ARobAS, 2007.

[MK98] Daniel Morris and Takeo Kanade. "A unified factorization algorithm for points, line segments and planes with uncertainty models". In: *ICCV*. 1998, pp. 696–702.

[MP03] Daniel Martinec and Tom Pajdla. "Line reconstruction from many perspective images by factorization". In: *CVPR*. 2003, pp. 497–502.

[MR11a] Faraz M. Mirzaei and Stergios I. Roumeliotis. "Globally optimal pose estimation from line correspondences". In: *ICRA*. 2011, pp. 5581–5588.

[MR11b] Faraz M. Mirzaei and Stergios I. Roumeliotis. "Optimal estimation of vanishing points in a Manhattan world". In: *ICCV*. 2011, pp. 2454–2461.

[MS05a] Krystian Mikolajczyk and Cordelia Schmid. "A performance evaluation of local descriptors". In: *TPAMI* 27.10 (2005), pp. 1615–1630.

[MS05b] Krystian Mikolajczyk and Cordelia Schmid. "A performance evaluation of local descriptors". In: *TPAMI* 27.10 (2005), pp. 1615–1630.

[MS08] J. Meltzer and S. Soatto. "Edge descriptors for robust wide-baseline correspondence". In: *CVPR*. 2008, pp. 1–8.

[MTM00] J. M. M. Montiel, J. D. Tardos, and L. Montano. "Structure and motion from straight line segments". In: *PR* 33.8 (2000), pp. 1295–1307.

[MZS03] K. Mikolajczyk, A. Zisserman, and C. Schmid. "Shape recognition with edge-based features". In: *BMVC*. 2003.

[NPVCL08] P. Neubert, P. Protzel, T. Vidal-Calleja, and S. Lacroix. "A fast visual line segment tracker". In: *ETFA*. 2008, pp. 353–360.

[NS11] Marcos Nieto and Luis Salgado. "Simultaneous estimation of vanishing points and their converging lines using the EM algorithm". In: *Pattern Recogn. Lett.* 32.14 (2011), pp. 1691–1700.

[NW06] Jorge Nocedal and Stephen J. Wright. Numerical optimization. Second. Springer, 2006.

[Ohw80] M.S Ohwovoriole. "'An extension of screw theory and its application to the automation of industrial assemblies'". PhD thesis. Department of Mechanical Engineering, Stanford University, 1980.

[PC09] A. Pronobis and B. Caputo. "COLD: the CoSy localization database". In: *IJRR, Special Issue on Robotic Vision* 28.5 (2009), pp. 588–594.

[PHOW04] H. Pottmann, M. Hofer, B. Odehnal, and J. Wallner. "Line geometry for 3D shape understanding and reconstruction". In: *ECCV*. 2004, pp. 297–309.

[PTVF07] WH Press, SA Teukolsky, WT Vetterling, and BP Flannery. Numerical recipes: the art of scientific computing. Cambridge Univ. Press, 2007.

[QL99] Long Quan and Zhongdan Lan. “Linear n-point camera pose determination”. In: *TPAMI* 21.8 (1999), pp. 774–780.

[QZ08] Lijuan Qin and Feng Zhu. “A new method for pose estimation from line correspondences”. In: *Acta Automatica Sinica* 34.2 (2008), pp. 130–134.

[Rob88] K.S. Roberts. “A new representation for a line”. In: *CVPR*. 1988, pp. 635–640.

[Rot00] C. Rother. “A new approach for vanishing point detection in architectural environments”. In: *BMVC*. 2000, pp. 382–391.

[SA90] Minas E. Spetsakis and John Aloimonos. “Structure from motion using line correspondences”. In: *IJCV* 4.3 (1990), pp. 171–183.

[SD04] Grant Schindler and Frank Dellaert. “Atlanta world: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments”. In: *CVPR*. 2004, pp. 203–209.

[SDMK11] Hauke Strasdat, Andrew J. Davison, J. M. M. Montiel, and Kurt Konolige. “Double window optimisation for constant time visual SLAM”. In: *ICCV*. 2011, pp. 2352–2359.

[SH96] Yongduek Seo and Ki Sang Hong. “Sequential reconstruction of lines in projective space”. In: *ICPR*. 1996, pp. 503–507.

[SKD06] G. Schindler, P. Krishnamurthy, and F. Dellaert. “Line-based structure from motion for urban environments”. In: *3DPVT*. 2006, pp. 846–853.

[SLS11] Margarita Chli Stefan Leutenegger and Roland Siegwart. “BRISK: binary robust invariant scalable keypoints”. In: *ICCV*. 2011.

[SRD06] Paul Smith, Ian Reid, and Andrew Davison. “Real-time monocular SLAM with straight lines”. In: *BMVC*. 2006, pp. 17–26.

[SU12] Alexander G. Schwing and Raquel Urtasun. "Efficient exact inference for 3D indoor scene understanding". In: *ECCV*. 2012, pp. 299–313.

[SVCCM12] Joan Sola, Teresa A. Vidal-Calleja, Javier Civera, and Jose Maria Martinez Montiel. "Impact of landmark parametrization on monocular EKF-SLAM with points and lines". In: *IJCV* 97.3 (2012), pp. 339–368.

[SZ97] Cordelia Schmid and Andrew Zisserman. "Automatic line matching across views". In: *CVPR*. 1997, pp. 666–671.

[Tar09] J.-P. Tardif. "Non-iterative approach for fast and accurate vanishing point detection". In: *ICCV*. 2009, pp. 1250–1257.

[TBKL12] Elena Tretyak, Olga Barinova, Pushmeet Kohli, and Victor Lempitsky. "Geometric image parsing in man-made environments". In: *IJCV* 97.3 (2012), pp. 305–321.

[TK95] Camillo J. Taylor and David J. Kriegman. "Structure and motion from line segments in multiple images". In: *TPAMI* 17.11 (1995), pp. 1021–1032.

[Tri96] B. Triggs. "Factorization methods for projective structure and motion". In: *CVPR*. 1996, pp. 845–851.

[TVG04] Tinne Tuytelaars and Luc Van Gool. "Matching widely separated views based on affine invariant regions". In: *IJCV* 59.1 (2004), pp. 61–85.

[TVGPM98] T. Tuytelaars, L. Van Gool, M. Proesmans, and T. Moons. "The cascaded hough transform as an aid in aerial image interpretation". In: *ICCV*. 1998, pp. 67–72.

[TW04] D. Titterton and J. Weston. Strapdown Inertial Navigation Technology. Second. The American Institute of Aeronautics and Astronautics, 2004.

[Ume91] Shinji Umeyama. "Least-squares estimation of transformation parameters between two point patterns". In: *TPAMI* 13.4 (1991), pp. 376–380.

[VF90] T. Vieville and O. Faugeras. “Feed-forward recovery of motion and structure from a sequence of 2D-lines matches”. In: *ICCV*. 1990, pp. 517–520.

[WH12] H. Wildenauer and A. Hanbury. “Robust camera self-calibration from monocular images of manhattan worlds”. In: *CVPR*. 2012, pp. 2831–2838.

[WH97] Richard C. Wilson and Edwin R. Hancock. “Structural matching by discrete relaxation”. In: *TPAMI* 19.6 (1997), pp. 634–648.

[WHA92] J. Weng, T.S. Huang, and N. Ahuja. “Motion and structure from line correspondences: closed-form solution, uniqueness, and optimization”. In: *TPAMI* 14.3 (1992), pp. 318–336.

[WL11] Guowei Wan and Sikun Li. “Automatic facades segmentation using detected lines and vanishing points”. In: *CISP*. 2011, pp. 1214–1217.

[WNY09] Lu Wang, U. Neumann, and S. You. “Wide-baseline image matching using line signatures”. In: *ICCV*. 2009, pp. 1311–1318.

[WPHB09] Dong-Min Woo, Dong-Chul Park, Seung-Soo Han, and Seunghwa Beack. “2D line matching using geometric and intensity data”. In: *AICI*. 2009, pp. 99–103.

[WT86] Wu Wen-Tsun. “Basic principles of mechanical theorem proving in elementary geometries”. In: *Journal of Automated Reasoning* 2.3 (1986), pp. 221–252.

[Wu+05] Yuanxin Wu, Xiaoping Hu, Dewen Hu, Tao Li, and Junxiang Lian. “Strapdown inertial navigation system algorithms based on dual quaternions”. In: *TAES* 41.1 (2005), pp. 110–132.

[WV07] Horst Wildenauer and Markus Vincze. “Vanishing point detection in complex man-made worlds”. In: *ICIAP*. 2007, pp. 615–622.

[WWH09] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. "MSLD: a robust descriptor for line matching". In: *PR* 42.5 (2009), pp. 941–953.

[XLT12] Tianfan Xue, Jianzhuang Liu, and Xiaoou Tang. "Example-based 3D object reconstruction from line drawings". In: *CVPR*. 2012, pp. 302–309.

[XZCK13] Chi Xu, Lilian Zhang, Li Cheng, and Reinhard Koch. "Pose estimation from line correspondences: a complete 3D line configuration analysis". In: *Submitted to IJCV* (2013).

[YH83] B. Yen and T. Huang. "Determining 3D motion and structure of a rigid body using straight line correspondences". In: *ICASSP*. 1983, pp. 118–121.

[ZDB08] Feng Zhou, H.B.-L. Duh, and M. Billinghurst. "Trends in augmented reality tracking, interaction and display: a review of ten years of ISMAR". In: *ISMAR*. 2008, pp. 193–202.

[ZK11] Lilian Zhang and Reinhard Koch. "Hand-held monocular SLAM based on line segments". In: *IMVIP*. 2011, pp. 8–15.

[ZK12a] Lilian Zhang and Reinhard Koch. "Line matching using appearance similarities and geometric constraints". In: *DAGM-OAGM*. 2012, pp. 236–245.

[ZK12b] Lilian Zhang and Reinhard Koch. "Structure and motion from line correspondences: representation, projection, initialization and sparse bundle adjustment". In: *Submitted to JVCI* (2012).

[ZK12c] Lilian Zhang and Reinhard Koch. "Vanishing points estimation and line classification in a manhattan world". In: *ACCV*. 2012, pp. 38–51.

[ZK13] Lilian Zhang and Reinhard Koch. "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency". In: *JVCI* 24.7 (2013), pp. 794–805.

[ZLK13] Lilian Zhang, Humin Lu, and Reinhard Koch. "Vanishing point estimation and line classification in a Manhattan world with a unifying camera model". In: *Submitted to IJCV* (2013).

[ZXLK12] Lilian Zhang, Chi Xu, Kok-Meng Lee, and Reinhard Koch. "Robust and efficient pose estimation from line correspondences". In: *ACCV*. 2012, pp. 217–230.

[Shu93] M. D. Shuster. "A survey of attitude representations". In: *Journal of the Astronautical Sciences* 41 (1993), pp. 439–517.