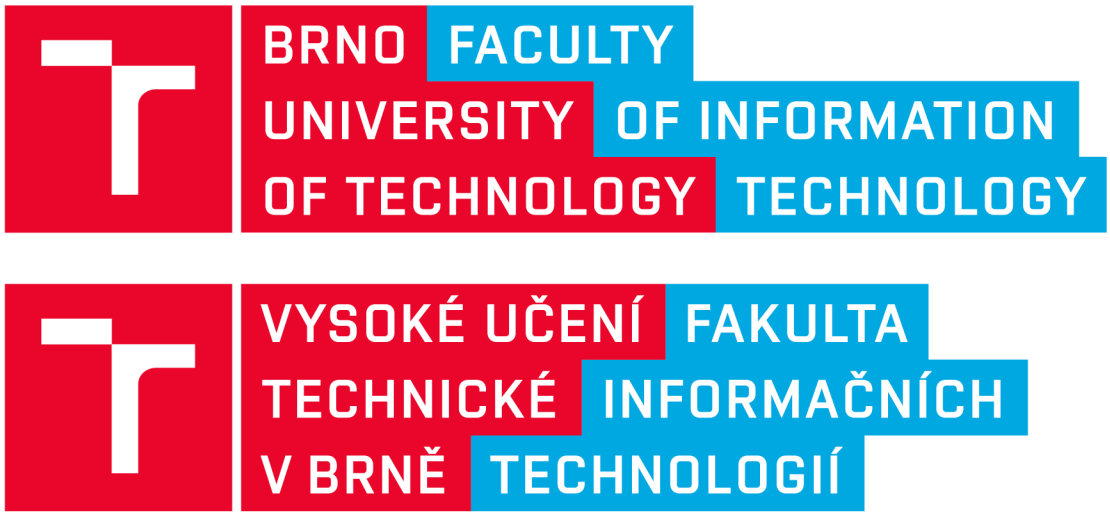

Camera Pose Estimation from Lines using Direct Linear Transformation

Odhad pózy kamery z přímek pomocí přímé lineární transformace

PH. D. THESIS
DISERTAČNÍ PRÁCE

AUTHOR
AUTOR

Ing. Bronislav Přibyl

SUPERVISOR
ŠKOLITEL

prof. Dr. Ing. Pavel Zemčík

BRNO 2017

Abstract

The thesis is concerned with camera pose estimation from correspondences of 3D/2D lines, i. e. with the Perspective-n-Line (PnL) problem. Attention is focused on large line sets which can be efficiently solved by methods using linear formulation of PnL. Up to date, methods working only with point-line correspondences were known. Motivated by this, two novel methods based on the Direct Linear Transformation (DLT) algorithm are proposed: DLT-Plücker-Lines working with line-line correspondences and DLT-Combined-Lines working with both point-line and line-line correspondences. In the latter case, the redundant information reduces the minimum of required line correspondences to 5 and improves accuracy of the method. The methods were extensively evaluated and compared to several state-of-the-art PnL methods in various conditions including simulated and real-world data. DLT-Combined-Lines achieves results similar to or better than state-of-the-art, while it is still highly efficient. In addition, the thesis introduces a unifying framework for DLT-based pose estimation methods, within which the proposed methods are presented.

Abstrakt

Disertační práce se zabývá odhadem pózy kamery z korespondencí 3D a 2D přímek, tedy tzv. perspektivním problémem n přímek (angl. Perspective-n-Line, PnL). Pozornost je soustředěna na případy s velkým počtem čar, které mohou být efektivně řešeny metodami využívajícími lineární formulaci PnL. Dosud byly známy pouze metody pracující s korespondencemi 3D bodů a 2D přímek. Na základě tohoto pozorování byly navrženy dvě nové metody založené na algoritmu přímé lineární transformace (angl. Direct Linear Transformation, DLT): Metoda DLT-Plücker-Lines pracující s korespondencemi 3D a 2D přímek a metoda DLT-Combined-Lines pracující jak s korespondencemi 3D bodů a 2D přímek, tak s korespondencemi 3D přímek a 2D přímek. Ve druhém případě je redundantní 3D informace využita k redukci minimálního počtu požadovaných korespondencí přímek na 5 a ke zlepšení přesnosti metody. Navržené metody byly důkladně testovány za různých podmínek včetně simulovaných a reálných dat a porovnány s nejlepšími existujícími PnL metodami. Metoda DLT-Combined-Lines dosahuje výsledků lepších nebo srovnatelných s nejlepšími existujícími metodami a zároveň je značně rychlá. Disertační práce také zavádí jednotný rámec pro popis metod pro odhad pózy kamery založených na algoritmu DLT. Obě navržené metody jsou definovány v tomto rámci.

Contents

List of Abbreviations

AOR	Algebraic Outlier Rejection
ASPnL	Accurate Subset based PnL (algorithm)
BA	Bundle Adjustment
CGR	Cayley-Gibbs-Rodriguez (parameterization)
DLT	Direct Linear Transformation
DoF	Degree of Freedom
LBD	Line Band Descriptor
LEHF	Line-based Eight-directional Histogram Feature
LICF	Line Intersection Context Feature
LOI	Line-based Orthogonal Iteration (sometimes also LBOI)
LPnL	Linear PnL
LSD	Line Segment Detector
MLESAC	Maximum Likelihood Estimation SAmple Consensus
MSLD	Mean-Standard deviation Line Descriptor
OI	Orthogonal Iteration
P3L	Perspective-3-Line (problem)
PnL	Perspective-n-Line (problem)
PnP	Perspective-n-Point (problem)
POSIT	Pose from Orthography and Scaling with ITeration
RANSAC	RANdom SAmple Consensus
RPnL	Robust PnL (algorithm)
RPnP	Robust PnP (algorithm)
SIFT	Scale Invariant Feature Transform
SVD	Singular Value Decomposition

Chapter 1

Introduction

Computers take part in our lives, and that part is increasing as computers get faster, smaller, easier to use and more powerful. If a camera is connected to a computer, it is given a chance to "see", enhancing its capabilities. The computer does not see in fact; it just gets a meaningless mosaic of pixels. In order to give a meaning to the pixels (i. e. to s e e), the computer must be given instructions for interpreting the pixel values or it must be able to learn them. People who prepare such instructions or teach computers to learn them deal with *computer vision*.

The goal of computer vision is to allow computers to see. To see like humans perhaps, or even better. This is a very ambitious goal and it is still too far from being true due to its complexity. However, some tasks have already been solved. Computers are able, for example, to find specific objects in images, to recognize human faces, to localize a robot using on-board cameras, or to reconstruct 3D objects, or even whole cities, from multiple images.

Accomplishing of many tasks in computer vision is achieved through the exploitation of *features*. Features are interesting parts of an image or a scene in this context. Depending on an application, the features can be points, lines, curves, regions, more complicated structures, or combinations of them. If features in a scene are captured by a camera, they can be used to infer various geometric relations: Either between objects of the scene, or between the scene and the camera. By exploiting the geometric relations, it is possible to reconstruct a 3D scene, to localize and navigate a mobile robot, to operate a robotic arm (solely on the basis of visual information) or to augment user's view with additional information, to give an example. A fundamental underlying task of each of these applications is *pose estimation* – the task of determining the relative position and orientation of a camera and an object to each other in 3D space[1].

While pose estimation methods utilizing point features have been in focus of researchers for some time and they are thus relatively mature, pose estimation methods utilizing line features lag behind. However, points and lines carry a complementary information about a scene and it is thus desirable to make use of both. Points have an exact location, whereas the "location" of a line along its direction is inherently unknown. On the other hand, lines are more robust primitives because they can be broken or partially occluded, but they are still visible and they can be exploited. Recent state-of-the-art methods are efficient and accurate, but they utilize lines only in the image space. In the 3D space, just *point* features are used (exploiting the fact that if 3D points lie on a 3D line,

[1] The problem of absolute *pose estimation* is also known as the problem of *absolute orientation* or *exterior orientation* in photogrammetry.

their projections must coincide with projection of that line in the image). That means only point-line correspondences are used and the potential of line-line correspondences is wasted, although line-line correspondences may carry stronger geometric information about a scene than point-line correspondences.

The goal of the thesis is to improve accuracy and robustness of current state-of-the-art on pose estimation from lines by incorporating 3D lines and thus also the line-line correspondences directly into the pose estimation process, which will be experimentally proved. The thesis studies the linear formulation of pose estimation from lines, which is especially suitable for scenarios with large sets of lines. The Direct Linear Transformation (DLT)-based formulation, which was used to exploit only point-line correspondences so far, is of special interest. The thesis contributes to the state-of-the-art by formulating two new methods for pose estimation, which are built upon the DLT and make use of line-line correspondences. A secondary contribution of the thesis is a unifying view on the DLT-based methods for pose estimation from lines.

Although the work presented in the thesis is my own, it has been influenced by many discussions with Pavel Zemčík and Martin Čadík. They also both collaborated with me on writing our joint papers.

This work is organized into six chapters. In Chapter 2, basic concepts are introduced upon which the thesis is build. In Chapter 3, a review of related work and state-of-the-art of pose estimation from line correspondences is presented. In Chapter 4, the state-of-the-art is critically analyzed and two new methods – DLT-Plücker-Lines and DLT-Combined-Lines – are proposed and presented in a unifying framework, which relates the proposed methods with the existing method for pose estimation, DLT-Lines. In Chapter 5, performance of the proposed methods is benchmarked and compared to the state-of-the-art using simulations and real-world experiments. Finally, the thesis is concluded in Chapter 6 by summarizing its key points and by suggesting future research. The core of the thesis is constituted by Chapters 4 and 5.

Chapter 2

Basic Concepts

Since mathematical notation and related concepts vary in literature, they way how they are used in the thesis is defined in this chapter. The mathematical notation is introduced first. Then, camera model is introduced. An introduction to parameterization of 3D lines using Plücer coordinates follows. After that, projection of points and lines onto the image plane is derived in the context of the used camera model and line parameterization. Finally, a method of solving a homogeneous system of linear equations is introduced.

2.1 Notation

Scalars are typeset in italics (x, X), vectors are typeset in bold ($\mathbf{l}$, $\mathbf{L}$). All vectors are thought of as being column vectors unless explicitly transposed. Matrices are typeset in sans-serif fonts (t, D), the identity matrix is denoted by I and the zero matrix by $\mathsf{0}$. 2D entities are denoted by lower case letters (x, $\mathbf{l}$, t), 3D entities by upper case letters (X, $\mathbf{L}$, D). Some of the symbols used in this work are organized in the following table.

	scalar	vector	matrix
2D	$a-h$, $j-n$, q, s, x, y, δ, ϵ, ε, π, σ	$\mathbf{l}$, $\mathbf{p}$, $\mathbf{t}$, $\mathbf{u}$, $\mathbf{x}$, $\boldsymbol{\epsilon}$	t
3D	E, L, S, T, X, Y, Z, A, B, Γ, Δ, Σ, Θ	$\mathbf{0}$, $\mathbf{E}$, $\mathbf{L}$, $\mathbf{N}$, $\mathbf{T}$, $\mathbf{U}$, $\mathbf{V}$, $\mathbf{X}$, $\mathbf{Y}$	$\mathsf{0}$, I, D, K, L, M, P, R, U, V, W, Z, $\mathsf{\Sigma}$

No formal distinction between coordinate vectors and physical entities is made. Transformation and projection matrices acting on *points* and *lines* are distinguished by a *dot* and a *bar*, respectively ($\dot{\mathsf{D}}$, $\dot{\mathsf{P}}$, $\bar{\mathsf{D}}$, $\bar{\mathsf{P}}$).
Operators and functions are denoted as follows.

- Equality of up to a nonzero scale factor is denoted by $\approx$,
- transposition by $^\top$,
- ℓ_2 norm (Euclidean norm) of a vector by $\|.\|$,
- ℓ_1 norm of a vector by $\|.\|_1$,
- Kronecker product by $\otimes$,
- vectorization of a matrix in column-major order by vec(.) ,
- the skew symmetric matrix associated with the cross product by $[.]_\times$,
  i. e. $[\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b}$.

Finally, the following two functions are defined. The first one is $\text{mean}_\circ(.)$ – the mean of all atomic elements of its argument. In the case of a *vector*, the result is straightforward:

$$\text{mean}_\circ(\mathbf{a}) = \frac{\sum_{i=1}^{n} a_i}{n} \ . \tag{2.1}$$

In the case of a *matrix*, the result is the mean of all matrix entries (not just of column/row vectors):

$$\text{mean}_\circ(\mathsf{M}) = \text{mean}_\circ(\text{vec}(\mathsf{M})) \ . \tag{2.2}$$

In the case of a *set*, the elements of the set are concatenated into a single vector or matrix first, the function is evaluated after the concatenation

$$\text{mean}_\circ(\{\mathbf{X}_i\}) = \text{mean}_\circ((\mathbf{X}_1^\top \ \mathbf{X}_2^\top \dots \mathbf{X}_n^\top)^\top) \ , \tag{2.3}$$

where $i = 1 \dots n$.

The second function is $\text{mean}_{|\circ|}(.)$ – the mean of *absolute values* of all atomic elements of its argument. It acts on vectors, matrices and sets in the same way as the function $\text{mean}_\circ(.)$ does.

2.2 Camera Model

A camera with central perspective projection is assumed, where 3D points and lines project onto an image plane which does not coincide with the center of projection. This is called a *pinhole camera* model [18]. The model is parameterized using two sets of parameters: extrinsic and intrinsic parameters.

Extrinsic parameters encode the position and orientation – i. e. the *pose* – of a camera in space. A transition from the world to the camera coordinate system is realized through a translation followed by a rotation The translation is parameterized using a 3×1 translation vector $\mathbf{T} = (T_1 \ T_2 \ T_3)^\top$, which represents the position of the camera in the world coordinate system. The rotation is parameterized using a 3×3 rotation matrix R describing the orientation of the camera in the world coordinate system by means of three consecutive rotations along the three axes Z, Y, X by respective Euler angles Γ, B, A. The pose of a camera thus has 6 Degrees of Freedom (DoF): T_1, T_2, T_3, A, B, Γ.

The task of pose estimation can be alternatively formulated as object pose estimation (w. r. t. the camera coordinate system). In this work, however, the earlier formulation is adopted, i. e. estimation of the pose of a camera (w. r. t. the object or world coordinate system). The two formulations are equivalent.

Intrinsic parameters describe how the (physical) coordinates of 2D points in the image plane map to its image coordinates (in pixels). Such mapping can be expressed by an upper-triangular 3×3 camera calibration matrix K.

Putting together both the extrinsic parameters (R, $\mathbf{T}$) and the intrinsic parameters (K), a 3D

point $\mathbf{X}$ can be related to its projection $\mathbf{u}$ in the image by the equation

$$\mathbf{u} \approx \mathsf{K}\,[\mathsf{R}\ \text{-}\mathsf{R}\mathbf{T}]\,\mathbf{X}\ . \tag{2.4}$$

Both $\mathbf{X}$ and $\mathbf{u}$ are expressed in homogeneous coordinates.

When the camera is intrinsically calibrated, i. e. when K is known, the image coordinates $\mathbf{u}$ can be converted into the normalized image coordinates $\mathbf{x} = \mathsf{K}^{-1}\mathbf{u}$. The projection $\mathbf{x}$ of a 3D point $\mathbf{X}$ in the normalized image plane can then be computed directly

$$\mathbf{x} \approx [\mathsf{R}\ \text{-}\mathsf{R}\mathbf{T}]\,\mathbf{X}\ . \tag{2.5}$$

In the rest of this work, a pinhole camera with known intrinsic parameters is assumed, i. e. coordinates of 2D points and lines are the *normalized image coordinates*.

2.3 Plücker Coordinates

Plücker Coordinates are the only linear over-parameterization with *linear* projection function. Moreover, it is a complete parameterization using "only" 6 DoF.

Given two distinct 3D points $\mathbf{X} = (X_1\ X_2\ X_3\ X_4)^\top$ and $\mathbf{Y} = (Y_1\ Y_2\ Y_3\ Y_4)^\top$ in homogeneous coordinates, a line joining them in projective 3-space is a homogeneous 6-vector $\mathbf{L} \approx (\mathbf{U}^\top\ \mathbf{V}^\top)^\top = (L_1\ L_2\ L_3\ L_4\ L_5\ L_6)^\top$, where

$$\begin{aligned} \mathbf{U}^\top &= (L_1\ L_2\ L_3) = (X_1\ X_2\ X_3)\ \times\ (Y_1\ Y_2\ Y_3)\ , \\ \mathbf{V}^\top &= (L_4\ L_5\ L_6) = X_4(Y_1\ Y_2\ Y_3)\ -\ Y_4(X_1\ X_2\ X_3)\ . \end{aligned} \tag{2.6}$$

The $\mathbf{V}$ part encodes direction of the line while the $\mathbf{U}$ part encodes position of the line in space. In fact, $\mathbf{U}$ is a normal of an interpretation plane – a plane passing through the line and the origin. As a consequence, $\mathbf{L}$ must satisfy a bilinear constraint $\mathbf{U}^\top\mathbf{V} = 0$. Existence of this constraint explains the discrepancy between 4 DoF of a 3D line and its parameterization by a homogeneous 6-vector. More on Plücker coordinates can be found e. g. in [18].

2.4 Projection of Points and Lines

The way, how transformations of points and lines are made, depends on the chosen parameterization. In the following, 3D lines are assumed to be parameterized using Plücker coordinates and 3D points are assumed to be parameterized using homogeneous coordinates.

Transformation of a Point. A homogeneous 3D point $\mathbf{X} = (X_1\ X_2\ X_3\ X_4)^\top$ in the world coordinate system is transformed to a point $\dot{\mathsf{D}}\mathbf{X}$ in the camera coordinate system using a 4×4 point displacement matrix

$$\dot{\mathsf{D}} \approx \begin{bmatrix} \mathsf{R} & \text{-}\mathsf{R}\mathbf{T} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}\ . \tag{2.7}$$

Projection of a Point. After 3D points are transformed into the camera coordinate system, they can be projected onto the normalized image plane using the 3×4 canonical camera matrix $(\mathsf{I}\ \mathbf{0})$. Compositing the two transformations yields the 3×4 point projection matrix

$$\dot{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & -\mathsf{R}\mathbf{T} \end{bmatrix} . \tag{2.8}$$

A 3D point $\mathbf{X}$ is then projected using the point projection matrix $\dot{\mathsf{P}}$ as

$$\mathbf{x} \approx \dot{\mathsf{P}}\mathbf{X} , \tag{2.9}$$

where $\mathbf{x} = (x_1\ x_2\ x_3)^\top$ is a homogeneous 2D point in the normalized image plane.

Transformation of a Line. A 3D line parameterized using Plücker coordinates can be transformed from the world into the camera coordinate system using the 6×6 line displacement matrix[1]

$$\bar{\mathsf{D}} \approx \begin{bmatrix} \mathsf{R} & \mathsf{R}[-\mathbf{T}]_\times \\ \mathsf{0}_{3\times3} & \mathsf{R} \end{bmatrix} . \tag{2.10}$$

Projection of a Line. After 3D lines are transformed into the camera coordinate system, their projections onto the image plane can be determined as intersections of their interpretation planes with the image plane; see Figure 2.1 for illustration. The normal $\mathbf{U}$ of an interpretation plane is identical to the image line $\mathbf{l}$ in the coordinate system of the camera, hence only $\mathbf{U}$ needs to be computed when projecting $\mathbf{L}$, and only the upper half of $\bar{\mathsf{D}}$ is needed, yielding the 3×6 line projection matrix [13]

$$\bar{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & \mathsf{R}[-\mathbf{T}]_\times \end{bmatrix} . \tag{2.11}$$

The line projection matrix in Eq. (2.11) can also be achieved by compositing the two transformations defined by the line displacement matrix $\bar{\mathsf{D}}$ (2.10) and by the 3×6 canonical camera matrix $(\mathsf{I}\ \mathsf{0})$.

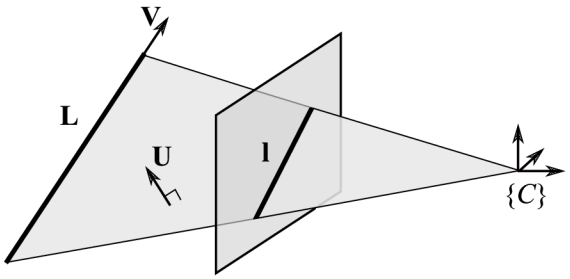

Figure 2.1: 3D line projection. The 3D line $\mathbf{L}$ is parameterized by its direction vector $\mathbf{V}$ and a normal $\mathbf{U}$ of its interpretation plane, which passes through the origin of the camera coordinate system $\{C\}$. Since the projected 2D line $\mathbf{l}$ lies at the intersection of the interpretation plane and the image plane, it is fully defined by the normal $\mathbf{U}$.

A 3D line $\mathbf{L}$ is then projected using the line projection matrix $\bar{\mathsf{P}}$ as

$$\mathbf{l} \approx \bar{\mathsf{P}}\mathbf{L} , \tag{2.12}$$

[1] Please note that our line displacement matrix differs slightly from the matrix of Bartoli and Sturm [4, Eq. (6)], namely in the upper-right term: We have $\mathsf{R}[-\mathbf{T}]_\times$ instead of $[\mathbf{T}]_\times\mathsf{R}$ due to different coordinate system.

where $\mathbf{l} = (l_1\ l_2\ l_3)^\top$ is a homogeneous 2D line in the normalized image plane.

2.5 Solving a Homogeneous System of Linear Equations

Methods presented in this work often solve a homogeneous system of linear equations, which can be described by the matrix equation

$$\mathsf{M}\mathbf{x} = \mathbf{0} \ . \tag{2.13}$$

If the system has m equations and n unknowns, then the measurement matrix M containing coefficients of the equations is $m \times n$, and the vector of unknowns $\mathbf{x}$ has n entries. The trivial solution $\mathbf{x} = \mathbf{0}$ is not of interest, hence the desired solution must be constrained, typically

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \|\mathsf{M}\mathbf{x}\|^2 \\ \text{s.t.} \quad & \|\mathbf{x}\| = 1 \ . \end{aligned} \tag{2.14}$$

Eq. (2.13) holds only in an ideal (noise-free) case. If equation coefficients in M are perturbed by noise, an inconsistent system is obtained

$$\mathsf{M}\mathbf{x}' = \boldsymbol{\epsilon} \ , \tag{2.15}$$

where $\mathbf{x}'$ is only an approximate solution and $\boldsymbol{\epsilon}$ is an m-vector of measurement residuals.

In an ideal case (2.13) and assuming $m \geq n$, M has rank $n-1$ and $\mathbf{x}$ is the right nullspace of M of rank 1. However, in a noisy case (2.15), M has full rank n, thus its nullspace must have rank 0. This implies nonexistence of an exact solution. Still, an approximate solution may be found in a least-squares sense. If a rank deficient matrix M' is found

$$\begin{aligned} \arg\min_{\mathsf{M}'} \quad & \|\mathsf{M}' - \mathsf{M}\|^2 \\ \text{s.t.} \quad & \operatorname{rank}(\mathsf{M}') = \operatorname{rank}(\mathsf{M}) - 1 \ , \end{aligned} \tag{2.16}$$

then, the approximate solution $\mathbf{x}'$ of the system (2.15) is the right nullspace of M' of rank 1, i. e.

$$\mathsf{M}'\mathbf{x}' = \mathbf{0} \ . \tag{2.17}$$

Remark 2.1: In the rest of the thesis, the above-described way of solving a homogeneous linear system $\mathsf{M}\mathbf{x} = \mathbf{0}$ will be referred to as "*homogeneous linear least squares*". Although a mathematically correct term would be "low-rank approximation" (of M), the former designation was chosen due to its analogy to the term "linear least squares", which designates solving of a linear system $\mathsf{M}\mathbf{x} = \mathbf{b}$.

Chapter 3

Pose Estimation from Lines

Points are the most commonly used features, not only for pose estimation. It is so because points are the simplest geometric primitives, easy to represent mathematically and easy to handle in a space of any dimension [18]. A substantial amount of research has been dedicated to point features and their applications in computer vision. Lines, on the other hand, are more difficult to represent, especially in spaces of dimension 3 and higher. This was naturally reflected in less research effort dedicated to line features.

Nevertheless, points and lines carry a complementary information about a scene and it is thus desirable to make use of both. Points have an exact location, whereas the "location" of a line along its direction is inherently unknown. On the other hand, lines are a more robust type of a primitive, because they can be broken or partially occluded, but they are still visible and they can be exploited. Additionally, lines provide stronger structural information about a scene than points, see Figure 3.1. Lines are especially useful and sometimes indispensable in situations where point features are unreliable. This might be caused, for example, by a lack of texture or presence of repetitive patterns, see Figure 3.2. Such conditions are typical for man-made environments – wiry structures, streets, facades of buildings, corridors, rooms etc. Lines are often abundant in such environments [27].

The task of camera pose estimation from lines has a finite number of solutions for 3 and more lines. However, in the minimal case of 3 lines, solutions of the Perspective-3-Line (P3L) problem are multiple: up to 8 solutions may exist [6]. The ambiguity is removed by adding one or more lines and thus the PnL problem has a unique solution for $n \geq 4$ [33]. Having said that, special configurations

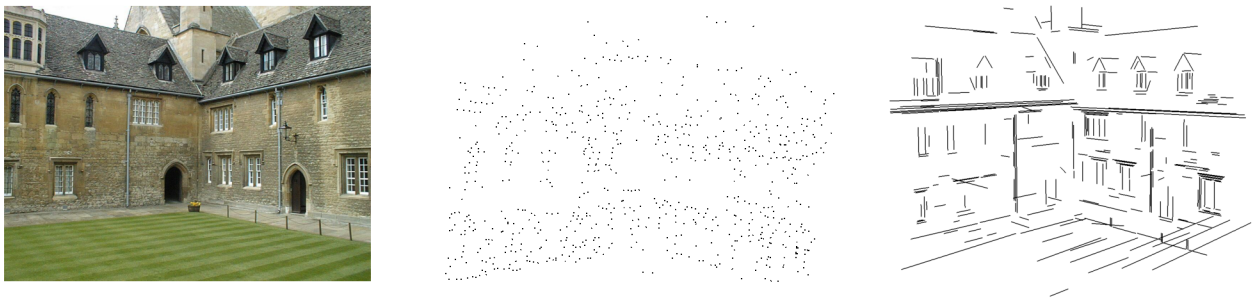

Figure 3.1: Representation of a building (on the *left*) using points (*center*) and lines (*right*).[1]

[1] The data is a courtesy of [32].

of lines must not be forget, for which the PnL problem has an infinite number of solutions even for $n \geq 4$. Such cases are termed singular configurations (e. g. a set of parallel lines, in which case, it is impossible to locate the camera along the lines). Generally, methods for pose estimation are known to be prone to singular and sometimes also to quasi-singular configurations of lines [28].

The PnL problem has been receiving attention for more than a quarter of century. Some of the earliest works are the ones of Dhome et al. from 1989 [12] and Liu et al. from 1990 [22]. They introduce two different ways to deal with the PnL problem: *iterative* and *algebraic*[2] approaches. As the names suggest, the algebraic methods solve PnL by minimizing an algebraic error in "one step", while the iterative methods iteratively minimize a nonlinear error function, which usually has a geometric meaning. Both approaches have different properties and thus also different use. A specific subset of algebraic approaches are the methods based on linear formulation of the PnL problem.

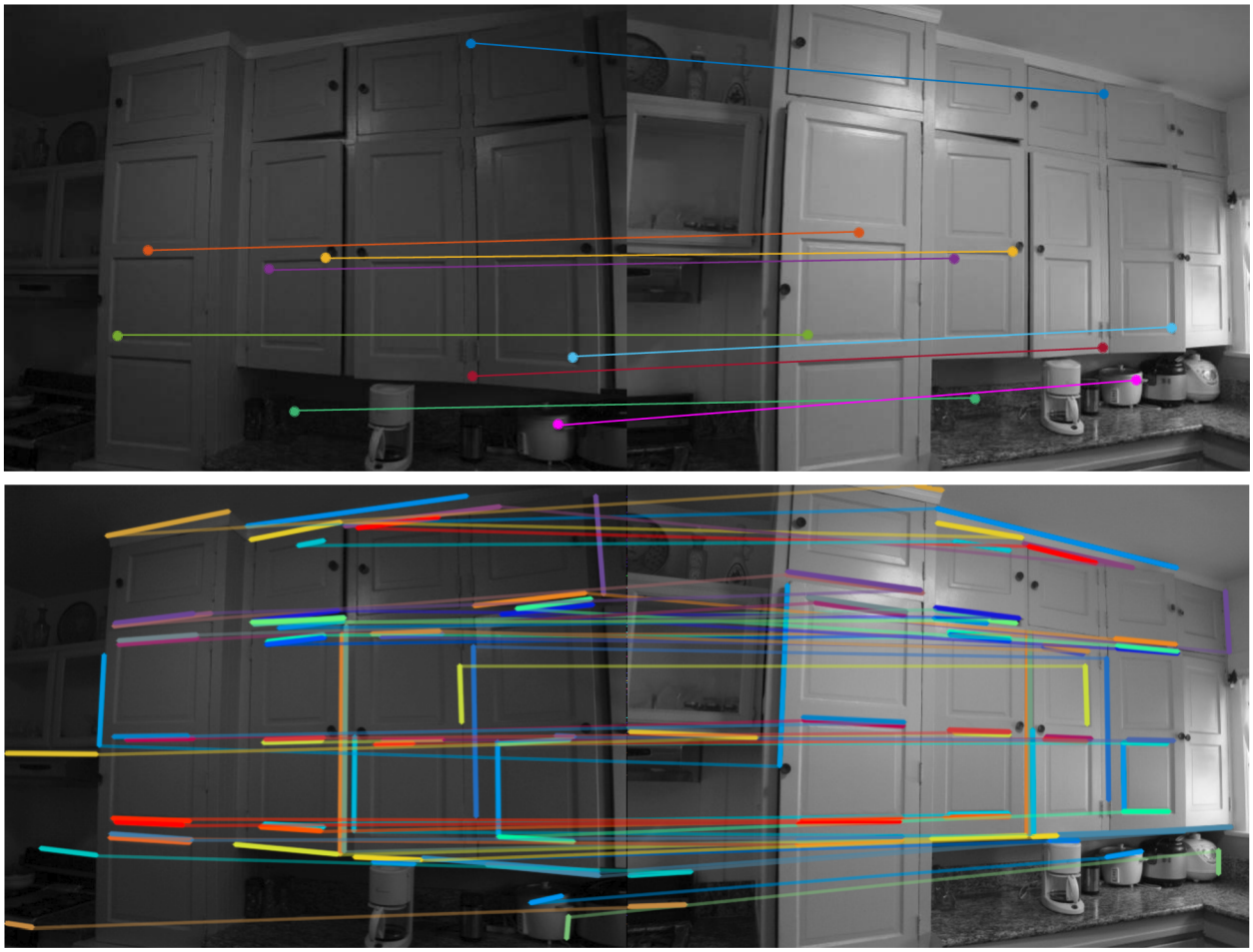

Figure 3.2: Point matches (*top*) and line matches (*bottom*) in a pair of images of a low-texutre scene. Only 9 matches were found using points, while 54 matches were found using lines.[3]

[2]Sometimes also called *non-iterative* approaches.
[3]The images and line matches are a courtesy of [34].

3.1 Iterative Methods

The iterative approaches consider pose estimation as a nonlinear least squares problem by iteratively minimizing specific error function, which usually has a geometrical meaning. In the early work of Liu et al. [22], the authors attempted to estimate the camera position and orientation separately developing a method called R_then_T. Later on, Kumar and Hanson [20] introduced a method called R_and_T for simultaneous estimation of camera position and orientation, and proved its superior performance to R_then_T. Recently, Zhang et al. [37] proposed two modifications of the R_and_T algorithm exploiting the uncertainty properties of line segment endpoints. Several other iterative methods are also capable of *simultaneous* estimation of pose parameters and line correspondences, e. g. [11, 36]. They pose an orthogonal approach to the common RANSAC-based correspondence filtering and consecutive separate pose estimation.

Iterative algorithms suffer from two common major issues when not initialized accurately: They converge slowly, and more severely, the estimated pose is often far from the true camera pose, finding only a local minimum of the error function. This makes iterative approaches suitable for final refinement of an initial solution, provided by some other algorithm.

3.2 Algebraic Methods

The algebraic approaches estimate the camera pose by solving a system of (usually polynomial) equations, minimizing an algebraic error. Their solutions are thus not necessarily geometrically optimal; on the other hand, no initialization is needed.

Among the earliest efforts in this field are those of Dhome et al. [12] and Chen [6]. Both methods solve the minimal problem of pose estimation from 3 line correspondences in a closed form. Solutions of the P3L problem are multiple: up to 8 solutions may exist [6]. Unfortunately, neither method is able to exploit more measurements to remove the ambiguity, and both methods are sensitive to presence of image noise.

Ansar and Daniilidis [3] developed a method that is able to handle 4 or more lines, limiting the number of possible solutions to 1. Lifting is employed to convert a polynomial system to linear equations in the entries of a rotation matrix. This approach may, however, fail in cases of singular line configurations (e. g. lines in 3 orthogonal directions [28]) as the underlying polynomial system may have multiple solutions. The algorithm has quadratic computational complexity ($O(n^2)$, where n is the number of lines), which renders it impractically slow for processing higher numbers of lines. The method also becomes unstable with increasing image noise, eventually producing solutions with complex numbers.

Recently, two major improvements of algebraic approaches have been achieved. First, **Mirzaei** and Roumeliotis [26] proposed a method, which is more computationally efficient ($O(n)$), behaves more robustly in the presence of image noise, and can handle the minimum of 3 lines, or more. A polynomial system with 27 candidate solutions is constructed and solved through the eigendecomposition of a multiplication matrix. Camera orientations having the least square error are considered to be the optimal ones. Camera positions are obtained separately using linear least squares. A weakness with this algorithm is that it often yields multiple solutions. Also, despite its linear com-

putational complexity, the overall computational time is still high due to slow construction of the multiplication matrix, which causes a high constant time penalty: 78 ms / 10 lines.

The second recent improvement is the Robust PnL (**RPnL**) algorithm of Zhang et al. [35]. Their method works with 4 or more lines and is more accurate and robust than the method of Mirzaei and Roumeliotis. An intermediate model coordinate system is used in the method of Zhang et al., which is aligned with a 3D line of longest projection. The lines are divided into triples, for each of which a P3L polynomial is formed. The optimal solution of the polynomial system is selected from the roots of its derivative in terms of a least squares residual.

The RPnL algorithm was later modified by Xu et al. [33] into the Accurate Subset based PnL (**ASPnL**) algorithm, which acts more accurately on small line sets. However, it is very sensitive to outliers, limiting its performance on real-world data. A drawback of both RPnL and ASPnL is that their computational time increases strongly for higher number of lines – from 8 ms / 10 lines to 630 – 880 ms / 1000 lines.

3.3 Methods based on Linear Formulation of PnL

A specific subset of algebraic methods are methods exploiting a linear formulation of the PnL problem (LPnL). Generally, the methods solve a system of linear equations, the size of which is directly proportional to the number of measurements. The biggest advantage of LPnL methods is their computational efficiency, making them fast for both low and high number of lines.

The most straightforward way to solve LPnL is the Direct Linear Transformation (DLT) algorithm [18]. It transforms the measured line correspondences into a homogeneous system of linear equations, whose coefficients are arranged into a measurement matrix. The solution then lies in the nullspace of the matrix. A necessary condition to apply any DLT method on noisy data is to prenormalize the input in order to ensure that the entries of the measurement matrix are of equal magnitude. Otherwise, the method will be oversensitive to noise and it will produce results arbitrarily far from the true solution.

The first DLT method for solving PnL is the method of Hartley and Zisserman [18, p. 180]. Following the terminology of Silva et al. [29], we call the method **DLT-Lines**. It does not act directly on 3D lines, but rather on 3D *points* lying on 3D lines (for example line endpoints). It exploits the fact that if a 3D line and a 3D point coincide, their projections also must coincide. The DLT-Lines method requires at least 6 line correspondences.

Recently, we introduced a new DLT method [II], which acts on 3D *lines* directly The lines are parameterized using Plücker coordinates, hence the name of the method is **DLT-Plücker-Lines**. The method yields more accurate estimates of camera orientation than DLT-Lines at the cost of a bit larger reprojection error and slightly lower computational efficiency. Also, the minimum number of lines required is 9.

Even more recently, Xu et al. [33] introduced a new set of methods exploiting the linear formulation of the PnL problem. The authors were inspired by the state-of-the-art Perspective-n-Point (PnP) solver working on the same principle [14]. Similarly to DLT-Lines, the new methods act on 3D *points* and 2D lines. The methods of Xu et al. [33] can be categorized by two criteria. Firstly, by parameterization of 3D points (either by Cartesian or by barycentric coordinates – this

is denoted in the method's names by "DLT" and "Bar", respectively). Secondly, by the manner in which a solution is obtained from the nullspace. The solution is either an exact rank-1 nullspace computed in closed form using homogeneous linear least squares, or it is estimated from an "effective nullspace" [21] of a dimension 1 – 4 (higher dimensions typically occurring under the presence of noise). This is denoted in the method's names by "LS" and "ENull", respectively. All the following methods require at least 6 line correspondences, although the effective null space solver (ENull) is sometimes able to recover the correct solution of an underdetermined system defined by 4 or 5 lines. The four LPnL methods of Xu et al. are the following:

LPnL_DLT_LS parameterizes 3D points using Cartesian coordinates, and it uses homogeneous linear least squares to recover the solution: entries of the rotation matrix and translation vector. This is exactly the same algorithm as DLT-Lines [18, p. 180], so we use the name *DLT-Lines* to refer to the method in the rest of the paper.

LPnL_DLT_ENull parameterizes 3D points using Cartesian coordinates, and it uses the effective nullspace solver [21] to recover the solution: entries of the rotation matrix and translation vector. It achieves higher accuracy than DLT-Lines.

LPnL_Bar_LS parameterizes 3D points using barycentric coordinates, which depend on the position of 4 arbitrarily chosen control points. Position of the control points with respect to camera is solved using homogeneous linear least squares. Alignment of the 4 camera- and world-referred control points defines the camera pose. Accuracy of the method is similar to DLT-Lines.

LPnL_Bar_ENull parameterizes 3D points using barycentric coordinates. Position of the 4 control points with respect to camera is solved using the effective nullspace solver. Alignment of the 4 camera- and world-referred control points defines the camera pose. The method is even more accurate than LPnL_Bar_LS.

3.4 Handling Mismatched Correspondences

In practice, mismatches of lines (i. e. outlying correspondences) often occur, which degrades the performance of camera pose estimation or even impedes it. It is thus necessary to identify and filter out mismatched correspondences and work preferably with correct matches.

RANSAC-based

The RANdom SAmple Consensus (RANSAC) algorithm [15] is commonly used to identify and remove outliers. It is a hypothesize-and-test scheme, where random samples are drawn from a set of data points, model parameters (i. e. hypotheses) are computed from the samples, and consensus of other data points is tested. This is repeated until a hypothesis with sufficient consensus is found or an iteration limit is exceeded.

A correct hypothesis is generated only if all data points in the sample are inliers. Since the chance of drawing an outlier-free sample depends not only on the fraction of inliers in the data but also on the size of the sample, it is desirable to use a minimal model. A non-minimal model can also be used, but, on average, more iterations are needed to obtain a correct hypothesis with the same probability as when using a minimal model.

In the context of pose estimation from lines, the data points are usually tentative line correspondences, the model parameters are parameters of a camera pose, and the consensus may be quantified e. g. by reprojection error of corresponding lines. The minimal number of line correspondences required to determine a camera pose is 3, but methods working with 4 line correspondences are also being used to generate hypotheses.

The RANSAC scheme can handle any percentage of outliers in theory as long as at least one outlier-free sample can be found. RANSAC is nondeterministic due to the use of random sampling. However, dozens of different RANSAC modifications have been introduced [25] eliminating various drawbacks of the original algorithm, e. g. [8, 9, 30].

Algebraic Outlier Rejection

As the LPnL methods work with 5 and more line correspondences, they cannot compete with the minimal (P3L) methods when plugged into a RANSAC-like framework due to an increased number of iterations.

This motivated an alternative scheme called Algebraic Outlier Rejection (AOR, [14]). It is an iterative approach integrated directly into the pose estimation procedure. Specifically, it is integrated into solving of the homogeneous linear system (2.13). Each line correspondence is assigned a weight, and the weights are arranged on the main diagonal of a square matrix W. This yields a homogeneous system of weighted linear equations

$$\mathsf{WM}\mathbf{x} = \mathbf{0} \ . \tag{3.1}$$

At the beginning, all weights are initialized to 1, conservatively assuming that all line correspondences are inliers. An approximate least-squares solution $\mathbf{x}$' of the system (3.1) is computed by Singular Value Decomposition (SVD) of $\mathsf{M}^\top\mathsf{WM}$, and a residual vector $\boldsymbol{\epsilon}$ of the solution is computed as

$$\boldsymbol{\epsilon} = \mathsf{M}\mathbf{x}' \ . \tag{3.2}$$

An algebraic error ε of each line correspondence is computed from the residual vector $\boldsymbol{\epsilon}$ as a norm of a sub-vector of corresponding residuals. E. g., for a case with 2 equations per line correspondence, the algebraic error of the i-th correspondence is $\varepsilon_i = \|(\epsilon_{2i-1}\ \epsilon_{2i})\|$. All correspondences are then assigned new weights

$$w_i = \begin{cases} 1 & \text{if } \varepsilon_i \leq \max(\varepsilon_{\max}, \delta_{\max}) \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{3.3}$$

and the whole procedure is repeated until convergence of the solution $\mathbf{x}'$. The constants $\varepsilon_{\max}$ and $\delta_{\max}$ are predefined thresholds. The strategy for choosing $\varepsilon_{\max}$ may be arbitrary but the authors [14] recommend $\varepsilon_{\max} = \mathrm{Q}_{25}(\varepsilon_1, \ \ldots, \ \varepsilon_n)$ which is the algebraic error of the correspondence that is at the boundary of the 25th percentile. The function is used as a robust estimator to reject correspondences with largest errors. The other threshold, $\delta_{\max}$, needs to be reached to consider a specific correspondence as an outlier. Its purpose is to avoid unnecessary rejections of inlier correspondences in an outlier-free case, and to achieve faster convergence.

The authors claim the break-down point to reach 60 % when applied to the PnP problem, and the process to usually converge in less than 5 iterations.

Chapter 4

Pose Estimation from Lines using Direct Linear Transformation

This chapter contains the majority of contributions of the thesis. First, the state-of-the-art is critically analyzed and the resolution is outlined. Then, two novel DLT-based methods for pose estimation from line correspondences are introduced and related to one existing DLT-based method. The methods are formulated within a novel unifying framework for DLT-based PnL methods.

4.1 Analysis of the State-of-the-Art

Pose estimation from line correspondences is a fundamental task required for many applications of computer vision – 3D reconstruction of a scene, localization and navigation of a robot, operation of a robotic arm solely on the basis of visual information, or augmentation of user's view with additional information, for example.

When estimating camera pose "from scratch", the following pipeline is typically used:

(i) Obtain tentative feature correspondences,
(ii) filter out outliers,
(iii) compute a solution from all inliers, and
(iv) iteratively refine the solution, e. g. by minimizing reprojection error (optionally).

Task (i) is usually carried out by appearance-based or geometry-based matching of lines. Task (ii) is usually carried out by iterative solving of a problem with a minimal number of line correspondences (i. e. P3L) in a RANSAC loop. Tasks (iii) and (iv), on the other hand, require solving a PnL problem with potentially high number of lines, which might be a time-consuming task. It is thus of interest to solve the task using an efficient algorithm.

As presented in the previous chapter, methods for solving PnL can be categorized as either iterative or algebraic. The iterative algorithms [7, 11, 20, 22, 36, 37] need initialization. This makes them suitable only for final refinement (iv) of an initial solution, which must be provided by some other algorithm. The initial solution (iii) may be provided by an algebraic algorithm [3, 6, 12, 26, 33, 35]. Among these, the methods of Chen [6] and Dhome et al. [12] are able to exploit only 3 line correspondences, thus they cannot be used in scenarios with more lines. The algorithm of Ansar and Daniilidis [3] overcomes the limitation of fixed number of lines, allowing to

use 4 and more lines. However, it has a quadratic computational complexity in the number of lines, which renders it impractically slow even for scenarios with dozens of lines. Mirzaei and Roumeliotis [26] eliminated the computational burden by introducing a method with linear computational complexity. Nonetheless, its runtime is still high due to a slow construction of a multiplication matrix, causing a high constant time penalty: it takes 78 ms to process 10 lines. Another drawback of the method is that it often yields multiple solutions. The shortcomings of [26] have been overcome by Zhang et al. [35] in their RPnL algorithm: it always yields a single solution and it takes 8 ms to compute a pose of 10 lines. However, the computational time increases strongly for higher number of lines: it takes 880 ms to process 1000 lines. The related method ASPnL of Xu et al. [33] inherits the attributes of RPnL. Alhough ASPnL is more accurate on small line sets, its runtime follows the characteristic of RPnL.

The non-LPnL algebraic methods only have been discussed so far. Nevertheless, in tasks involving a high number of lines, the non-LPnL methods are outperformed by the LPnL methods: by DLT-Lines of Hartley and Zisserman [18] and by the methods of Xu et al. [33]. These state-of-the-art methods are efficient and accurate *especially* in scenarios with high number of lines. Interestingly enough, they do not exploit all available information: They only utilize points in 3D space, but 3D lines remain unused. This means only point-line correspondences are used and the potential of line-line correspondences is unexploited, leaving a promising room for research and improvement.

The thesis aims for better accuracy and robustness than the state-of-the-art by introducing a new linear method for pose estimation. The method shall utilize line-line correspondences and keep the advantage of being fast which LPnL methods have in common. The goal is elaborated in the rest of this work and it is verified experimentally using both synthetic and real-world data.

The attention is focused on methods based on the DLT. First, a unifying framework for all DLT-based PnL methods is presented in Section 4.2. Then, all three DLT-based PnL methods are formulated within the framework. The methods are:

DLT-Lines of Hartley and Zisserman [18, p. 180], exploiting point-line correspondences only – Section 4.3.

DLT-Plücker-Lines of ours [II], exploiting line-line correspondences only – Section 4.4.

DLT-Combined-Lines of ours [I], exploiting both point-line and line-line correspondences – Section 4.5.

4.2 Common Structure of DLT Methods

In this section, the novel unifying framework for DLT-based PnL methods is introduced. Given the point-line or line-line correspondences, the camera pose can be estimated using a PnL method. The DLT-based PnL methods have the following steps in common:

1. Input data is prenormalized to achieve good conditioning of the linear system.
2. A projection matrix is estimated using homogeneous linear least squares, and the effect of prenormalization is reverted.

3. The pose parameters are extracted from the estimated projection matrix. This includes also constraint enforcement in the case of noisy data, since the constraints are not taken into account during the least-squares estimation.

Prenormalization

Since the DLT algorithm is sensitive to the choice of coordinate system, it is crucial to prenormalize the data to get a properly conditioned measurement matrix M [16]. Various transformations can be used, but the optimal ones are unknown. In practice, however, the goal is to reduce large values of point/line coordinates. This is usually achieved by centering the data around the origin and by scaling them s. t. an average coordinate has the absolute value of 1 (which means the average distance to the origin shall equal to $\sqrt{2}$ and $\sqrt{3}$ in the 2D and 3D case, respectively). Specific prenormalizing transformations are proposed for each method in the following sections.

Linear Estimation of a Projection Matrix

As a starting point, a system of linear equations needs to be constructed, which relates (prenormalized) 3D entities with their (prenormalized) image counterparts through a projection matrix, denoted P. The relation might be the projection of homogeneous 3D points $\mathbf{x} \approx \dot{\mathsf{P}}\mathbf{X}$ in Eq. (2.9), or the projection of Plücker lines $\mathbf{l} \approx \bar{\mathsf{P}}\mathbf{L}$ in Eq. (2.12), or other linear system, or a combination of those. The problem of camera pose estimation now resides in estimating the projection matrix P, which encodes all the six camera pose parameters T_1, T_2, T_3, A, B, Γ.

The system of linear equations is transformed into a homogeneous system of linear equations (see Appendix A of the thesis for details), i. e. a system having only a zero vector at the right-hand side.

$$\mathsf{M}\mathbf{p} = \mathbf{0} \tag{4.1}$$

M is a measurement matrix containing coefficients of equations generated by correspondences between 3D entities and their image counterparts. Each of the n correspondences gives rise to a number of independent linear equations (usually 2), and thus to the same number of rows of M. The number of columns of M equals d, which is the number of entries contained in P. The size of M is thus $2n \times d$. Eq. (4.1) is then solved for the d-vector $\mathbf{p} = \mathrm{vec}(\mathsf{P})$.

As mentioned in Section 2.5, Eq. (4.1) holds only in a noise-free case. If a noise is present in the measurements, an inconsistent system is obtained:

$$\mathsf{M}\mathbf{p}' = \boldsymbol{\epsilon} \ . \tag{4.2}$$

Only an approximate solution $\mathbf{p}'$ may be found through minimization of a $2n$-vector of measurement residuals $\boldsymbol{\epsilon}$ in a least-squares sense s. t. $\|\mathbf{p}'\| = 1$.

Once the system of linear equations given by Eq. (4.2) is solved, the estimate P' of the projection matrix P can be recovered from the d-vector $\mathbf{p}'$.

Extraction of Pose Parameters

The estimate P' of a projection matrix P obtained as a solution of the system (4.2) does not satisfy the constraints imposed on P. In fact, a projection matrix P has only 6 DoF – the 6 camera pose parameters T_1, T_2, T_3, A, B, Γ. It has, however, more entries: The 3×4 point projection matrix

$$\dot{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & -\mathsf{R}\mathbf{T} \end{bmatrix} \tag{4.3}$$

has 12 entries and the 3×6 line projection matrix

$$\bar{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & \mathsf{R}[-\mathbf{T}]_\times \end{bmatrix} \tag{4.4}$$

has 18 entries. This means that the projection matrices have 6 and 12 independent linear constraints, respectively.

The first six constraints are imposed by the rotation matrix R that must satisfy the orthonormality constraints (unit-norm and mutually orthogonal rows). The other six constraints in the case of $\bar{\mathsf{P}}$ are imposed by the skew-symmetric matrix $[-\mathbf{T}]_\times$ (three zeros on the main diagonal and antisymmetric off-diagonal elements).

In order to extract the pose parameters, the *scale* of an estimate P' of a projection matrix P has to be corrected first, since $\mathbf{p}'$ is usually of unit length as a minimizer of $\boldsymbol{\epsilon}$ in Eq. (4.2). The correct scale of P' can only be determined from the part which does *not* contain the translation $\mathbf{T}$. In both cases of $\dot{\mathsf{P}}$ (4.3) and $\bar{\mathsf{P}}$ (4.4), it is the left 3×3 submatrix – let us denote it P'_1 – an estimate of a rotation matrix R. A method of scale correction is recommended based on the fact that all three singular values of a proper rotation matrix should be 1. See Algorithm 1.

Algorithm 1: Scale correction of a projection matrix.

Input: An estimate P' of a projection matrix, possibly wrongly scaled and without fulfilled constraints.

1. $\mathsf{P}'_1 \leftarrow$ left 3×3 submatrix of P'
2. $\mathsf{U}\Sigma\mathsf{V}^\top \leftarrow \mathrm{SVD}(\mathsf{P}'_1)$
3. $s \leftarrow 1/\mathrm{mean}(\mathrm{diag}(\Sigma))$

Output: $s\mathsf{P}'$.

Alternatively, the scale can also be corrected so that $\det(s\mathsf{P}'_1) = 1$, but Algorithm 1 proved to be more robust in practice.

Further steps in the extraction of pose parameters differ in each method, they are thus described separately in the following sections.

4.3 DLT-Lines

DLT-Lines is the method by Hartley and Zisserman [18, p. 180]. In the following text, the method is put into context using the unifying framework of the previous section. DLT-Lines exploits the fact that a 3D point $\mathbf{X}$ lying on a 3D line $\mathbf{L}$ projects such that its projection $\mathbf{x} = \dot{\mathsf{P}}\mathbf{X}$ must also lie

on the projected line: $\mathbf{l}^\top \mathbf{x} = 0$, see Figure 4.1. Putting this together yields the constraint equation

$$\mathbf{l}^\top \dot{\mathsf{P}} \mathbf{X} = 0 \ . \tag{4.5}$$

The pose parameters are encoded in the 3×4 point projection matrix $\dot{\mathsf{P}}$, see Eq. (2.8). Since $\dot{\mathsf{P}}$ has 12 entries, at least 6 lines are required to fully determine the system, each line with 2 or more points on it.

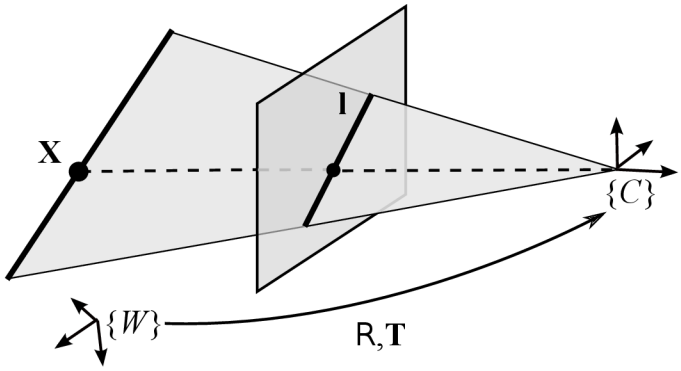

Figure 4.1: A point $\mathbf{X}$ lying on a 3D line projects s. t. its projection must lie on the image line $\mathbf{l}$ – a projection of the 3D line.

Prenormalization

The known quantities of Eq. (4.5), i. e. the coordinates of 3D points and 2D lines, need to be prenormalized. In the case of the DLT-based pose estimation from *points* [17], Hartley suggests to translate and scale both 3D and 2D points so that their centroid is at the origin and their average distance from the origin equals to $\sqrt{3}$ and $\sqrt{2}$, respectively.

By exploiting the principle of duality [10], it is suggested to treat coordinates of 2D lines as homogeneous coordinates of 2D points, and then to follow Hartley in the prenormalization procedure – i. e. to apply translation to the origin and then anisotropic scaling.

Linear Estimation of the Point Projection matrix

The point projection matrix $\dot{\mathsf{P}}$ and its estimate $\dot{\mathsf{P}}'$ are 3×4, so the corresponding measurement matrix $\dot{\mathsf{M}}$ is $n \times 12$, where n is the number of point-line correspondences $\mathbf{X}_i \leftrightarrow \mathbf{l}_i$, $(i = 1 \dots n,\ n \geq 12)$. $\dot{\mathsf{M}}$ is constructed as

$$\dot{\mathsf{M}}_{(i,\,:)} = \mathbf{X}_i^\top \otimes \mathbf{l}_i^\top \ , \tag{4.6}$$

where $\dot{\mathsf{M}}_{(i,\,:)}$ denotes the i-th row of $\dot{\mathsf{M}}$ in the Matlab notation. See Appendix A.3 of the thesis for a derivation of Eq. (4.6). The 3D points $\mathbf{X}_i$ must be located on at least 6 different lines.

Extraction of Pose Parameters

First, the scale of $\dot{\mathsf{P}}'$ is corrected using Algorithm 1, yielding $s\dot{\mathsf{P}}'$. Then, the left 3×3 submatrix of $s\dot{\mathsf{P}}'$ is taken as the estimate R' of a rotation matrix. A nearest rotation matrix R is found in the sense of the Frobenius norm using Algorithm 2.

Algorithm 2: Orthogonalization of a 3×3 matrix.

Input: A 3×3 estimate R' of a rotation matrix R.

1. $\mathsf{U}\Sigma\mathsf{V}^\top \leftarrow \mathrm{SVD}(\mathsf{R}')$
2. $d \leftarrow \det(\mathsf{U}\mathsf{V}^\top)$
3. $\mathsf{R} \leftarrow d\mathsf{U}\mathsf{V}^\top$

Output: R.

Please, note that Algorithms 1 and 2 can be combined and executed at once.

The remaining pose parameter to recover is the translation vector $\mathbf{T}$, which is encoded in the fourth column $\dot{\mathbf{P}}'_4$ of $\dot{\mathsf{P}}'$, see Eq. (2.8). It is recovered as $\mathbf{T} = \mathsf{R}^\top s\dot{\mathbf{P}}'_4$, completing the extraction of pose parameters.

4.4 DLT-Plücker-Lines

DLT-Plücker-Lines is a novel method, which was published in [II]. It exploits the *linear* projection of 3D lines parameterized using Plücker coordinates onto the image plane, as described in Section 2.3. A benefit of this method is higher accuracy of camera orientation estimates compared to DLT-Lines.

The formation of a 2D line $\mathbf{l}$ as a projection of a 3D line $\mathbf{L}$ is defined by the constraint equation (2.12)

$$\mathbf{l} \approx \bar{\mathsf{P}}\mathbf{L} \ , \tag{4.7}$$

as illustrated in Figure 4.2. The pose parameters are encoded in the 3×6 line projection matrix $\bar{\mathsf{P}}$, see Eq. (2.11). Since $\bar{\mathsf{P}}$ has 18 entries, at least 9 lines are required to fully determine the system.

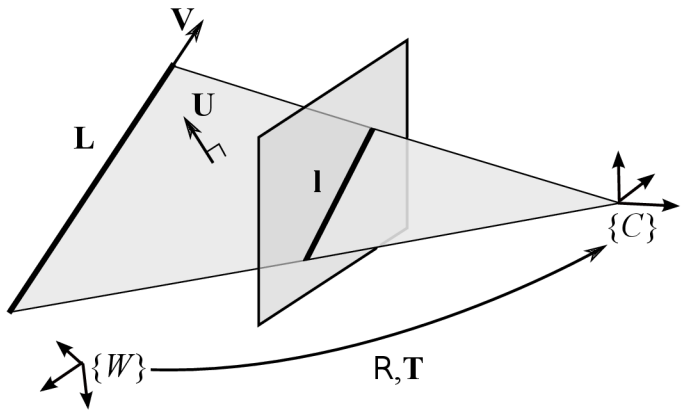

Figure 4.2: A 3D line $\mathbf{L}$ parameterized using Plücker coordinates is defined by a normal $\mathbf{U}$ of its interpretation plane and by its direction vector $\mathbf{V}$. Its projection is denoted $\mathbf{l}$.

Prenormalization

The known quantities of Eq. (4.7) need to be prenormalized, i. e. the Plücker coordinates of 3D lines $\mathbf{L}$, and the coordinates of 2D lines $\mathbf{l}$. Since the homogeneous Plücker coordinates of a 3D line $\mathbf{L}$ cannot be simply treated as homogeneous coordinates of a 5D point (because of the bilinear constraint, see Section 2.3), the following prenormalization is suggested.

Translation and scaling is applied in this case as well. However, both translation and scaling affect only the $\mathbf{U}$ part of each $\mathbf{L}$, and not the $\mathbf{V}$ part. Therefore, the $\mathbf{V}$ parts are adjusted first by multiplying each $\mathbf{L}$ by a nonzero scale factor so that $\|\mathbf{V}\| = \sqrt{3}$. Then, translation is applied to minimize the average magnitude of $\mathbf{U}$. Since $\|\mathbf{U}\|$ decreases with the distance of $\mathbf{L}$ from the origin, it is feasible to translate the lines so that the sum of squared distances from the origin is minimized. This can be efficiently computed using the Generalized Weiszfeld algorithm [1]. Finally, anisotropic scaling is applied so that the average magnitude of all $\mathbf{U}$ parts matches the average magnitude of all $\mathbf{V}$ parts. Both translation and scaling of lines is achieved by premultiplying them by a 6×6 line similarity matrix [4]. The procedure is summarized in Algorithm 3.

Algorithm 3: Prenormalization of 3D lines parameterized by Plücker coordinates.
Note: See Section 2.1 for the definition of function $\text{mean}_{|\circ|}(.)$.

Input: A set of m 3D lines $\{\mathbf{L}_j\}$, $j = 1 \dots m$.

1. For all lines do: $\mathbf{L}_j = \dfrac{\sqrt{3}}{\|\mathbf{V}_j\|} \cdot \mathbf{L}_j$

2. $\mathbf{T} \leftarrow$ Generalized_Weiszfeld_Algorithm$(\{\mathbf{L}_j\})$ ◁ Aftab et al. [1]

3. For all lines do: $\mathbf{L}_j = \begin{bmatrix} \mathsf{I} & [\text{-}\mathbf{T}]_\times \\ \mathsf{0} & \mathsf{I} \end{bmatrix} \mathbf{L}_j$ ◁ translation

4. $S_\text{X} \leftarrow \dfrac{\text{mean}_{|\circ|}(\{\mathbf{V}_j\})}{\text{mean}_{|\circ|}(\{L_{j,1}\})}$, $S_\text{Y} \leftarrow \dfrac{\text{mean}_{|\circ|}(\{\mathbf{V}_j\})}{\text{mean}_{|\circ|}(\{L_{j,2}\})}$, $S_\text{Z} \leftarrow \dfrac{\text{mean}_{|\circ|}(\{\mathbf{V}_j\})}{\text{mean}_{|\circ|}(\{L_{j,3}\})}$

5. For all lines do: $\mathbf{L}_j = \begin{bmatrix} S_\text{X} & & & \\ & S_\text{Y} & & \mathsf{0} \\ & & S_\text{Z} & \\ & \mathsf{0} & & \mathsf{I} \end{bmatrix} \mathbf{L}_j$ ◁ scaling

Output: A set of m prenormalized 3D lines $\{\mathbf{L}_j\}$, $j = 1 \dots m$.

Prenormalization of 2D lines can be carried out in the same way as in the case of the DLT-Lines method, see Section 4.3.

Linear Estimation of the Line Projection Matrix

The line projection matrix $\bar{\mathsf{P}}$ and its estimate $\bar{\mathsf{P}}'$ are 3×6, so the corresponding measurement matrix $\bar{\mathsf{M}}$ has 18 columns. The number of its rows depends on m, the number of line-line correspondences $\mathbf{L}_j \leftrightarrow \mathbf{l}_j$, $(j = 1 \dots m,\ m \geq 9)$. By exploiting Eq. (4.7), each correspondence generates three rows of $\bar{\mathsf{M}}$ (Matlab notation is used to index the matrix elements):

$$\bar{\mathsf{M}}_{(3j-2\,:\,3j,\ :)} = \mathbf{L}_j^\top \ \otimes\ [\mathbf{l}_j]_\times \ . \tag{4.8}$$

The line measurement matrix $\bar{\mathsf{M}}$ is thus $3m \times 18$. Note that only two of the three rows of $\bar{\mathsf{M}}$ defined by Eq. (4.8) are needed for each line-line correspondence, because they are linearly dependent. $\bar{\mathsf{M}}$ will be only $2m \times 18$ in this case. See Appendix A.1 of the thesis for a derivation of Eq. (4.8).

Extraction of Pose Parameters

First, the scale of $\bar{\mathsf{P}}'$ is corrected using Algorithm 1, yielding $s\bar{\mathsf{P}}'$. Then, the camera pose parameters are extracted from the right 3×3 submatrix of $s\bar{\mathsf{P}}'$, which is an estimate of a skew-symmetric matrix premultiplied by a rotation matrix (i. e. $\mathsf{R}[\text{-}\mathbf{T}]_\times$, see Eq. (2.11)). Since $s\bar{\mathsf{P}}'$ has the structure of the essential matrix [23], the algorithm of Tsai and Huang [31] is proposed to decompose $s\bar{\mathsf{P}}'$, as outlined in Algorithm 4. This completes the extraction of pose parameters.

The variable $q = (\Sigma_{1,1} + \Sigma_{2,2})/2$ in Algorithm 4 is an average of the first two singular values of $s\bar{\mathsf{P}}'_2$ to approximate the singular values of a properly constrained essential matrix, which should be $(q, q, 0)$. The ±1 term in Step 4 of Algorithm 4 denotes either +1 or -1 which has to be put on the diagonal so that $\det(\mathsf{R}_\mathrm{A}) = \det(\mathsf{R}_\mathrm{B}) = 1$.

Alternative ways of extracting the camera pose parameters from $s\bar{\mathsf{P}}'$ exist, e. g. computing the closest rotation matrix R to the left 3×3 submatrix of $s\bar{\mathsf{P}}'_1$ and then computing $[\mathbf{T}]_\times = \text{-}\mathsf{R}^\top s\bar{\mathsf{P}}'_2$. However, our experiments showed that the alternative ways are less robust to image noise. Therefore, the solution described in Algorithm 4 was chosen.

Algorithm 4: Extraction of pose parameters from the estimate $\bar{\mathsf{P}}'$ of a line projection matrix, inspired by [31].

Input: An estimate $\bar{\mathsf{P}}'$ of a line projection matrix $\bar{\mathsf{P}}$.

Input: Corrective scale factor s.

1. $\bar{\mathsf{P}}'_2 \leftarrow$ right 3×3 submatrix of $\bar{\mathsf{P}}'$
2. $\mathsf{U}\Sigma\mathsf{V}^\top \leftarrow \mathrm{SVD}(s\bar{\mathsf{P}}'_2)$
3. $$\mathsf{Z} \leftarrow \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathsf{W} \leftarrow \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$
   $q \leftarrow (\Sigma_{1,1} + \Sigma_{2,2})/2$
4. Compute 2 candidate solutions (A, B):
   $\mathsf{R}_\mathrm{A} \leftarrow \mathsf{U}\mathsf{W}\ \mathrm{diag}(1\ 1\ \pm1)\mathsf{V}^\top, \quad [\mathbf{T}]_{\times\mathrm{A}} \leftarrow q\mathsf{V}\mathsf{Z}\ \mathsf{V}^\top$
   $\mathsf{R}_\mathrm{B} \leftarrow \mathsf{U}\mathsf{W}^\top\mathrm{diag}(1\ 1\ \pm1)\mathsf{V}^\top, \quad [\mathbf{T}]_{\times\mathrm{B}} \leftarrow q\mathsf{V}\mathsf{Z}^\top\mathsf{V}^\top$
5. Accept the physically plausible solution, so that the scene lies in front of the camera.
   $\mathsf{R} \leftarrow \mathsf{R}_\mathrm{A}, \quad \mathbf{T} \leftarrow \mathbf{T}_\mathrm{A}$ or
   $\mathsf{R} \leftarrow \mathsf{R}_\mathrm{B}, \quad \mathbf{T} \leftarrow \mathbf{T}_\mathrm{B}$.

Output: R, $\mathbf{T}$.

4.5 DLT-Combined-Lines

DLT-Combined-Lines is a novel method published in [I]. It is a combination of DLT-Lines and DLT-Plücker-Lines methods, exploiting the redundant representation of 3D structure in the form of both 3D points and 3D lines, see Figure 4.3. The 2D structure is represented by 2D lines. The primary benefit of the method is a higher accuracy of the camera pose estimates and smaller reprojection

error, the secondary benefit is the lower number of required lines.

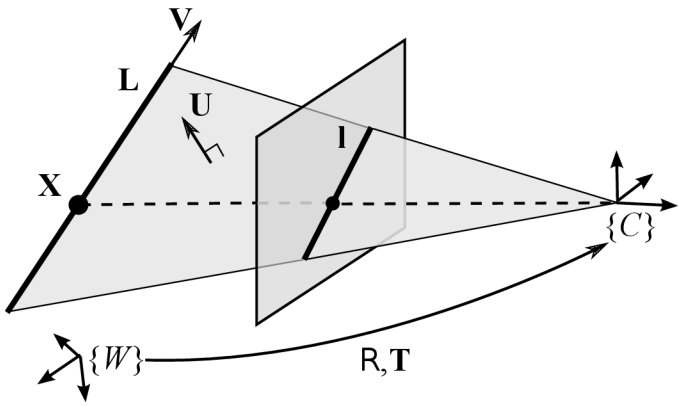

Figure 4.3: A 3D line **L** is parameterized by both Plücker coordinates of the line (i. e. the normal **U** of its interpretation plane and its direction vector **V**) and a point **X** lying on the line. **L** may be parameterized by many such points. Projection of the point **X** must lie on the projection **l** of the line **L**.

The central idea of the method is to merge two systems of linear equations, which share some unknowns, into one system. The unknowns are entries of the point projection matrix $\dot{\mathsf{P}}$ used in DLT-Lines and the line projection matrix $\bar{\mathsf{P}}$ used in DLT-Plücker-Lines. The two systems defined by Eq. (4.5) and (4.7) can be merged so that the set of unknowns of the resulting system is formed by the union of unknowns of both systems. It can be observed that the shared unknowns reside in the left 3×3 submatrices of $\dot{\mathsf{P}}$ and $\bar{\mathsf{P}}$. If unknowns of the resulting system are arranged in a feasible manner, a new 3×7 matrix $\ddot{\mathsf{P}}$ can be constructed, which is a "union" of $\dot{\mathsf{P}}$ and $\bar{\mathsf{P}}$:

$$\left.\begin{aligned} \dot{\mathsf{P}} &\approx [\ \mathsf{R} \quad \text{-}\mathsf{R}\mathbf{T}\ ] \\ \bar{\mathsf{P}} &\approx [\ \mathsf{R} \quad \mathsf{R}[\text{-}\mathbf{T}]_\times] \end{aligned}\right\} \quad \ddot{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & \text{-}\mathsf{R}\mathbf{T} & \mathsf{R}[\text{-}\mathbf{T}]_\times \end{bmatrix} \tag{4.9}$$

The matrix is called a *combined projection matrix*, because it allows to write the projection equations for point-line, line-line, and even point-point correspondences, as follows:

$$\mathbf{l}^\top \ddot{\mathsf{P}} \begin{pmatrix} \mathbf{X}^\top & 0 & 0 & 0 \end{pmatrix}^\top = 0 \ , \tag{4.10}$$

$$\mathbf{l} \approx \ddot{\mathsf{P}} \begin{pmatrix} \mathbf{U}^\top & 0 & \mathbf{V}^\top \end{pmatrix}^\top \ , \tag{4.11}$$

$$\mathbf{x} \approx \ddot{\mathsf{P}} \begin{pmatrix} \mathbf{X}^\top & 0 & 0 & 0 \end{pmatrix}^\top \ . \tag{4.12}$$

These equations can then be used to estimate $\ddot{\mathsf{P}}$ linearly from the correspondences.

A secondary benefit of the method is that it requires only 5 lines (and 10 points on them) – less then DLT-Plücker-Lines and even less then DLT-Lines. To explain why, the following matrices are defined first: the left-most 3×3 submatrix of $\ddot{\mathsf{P}}$ is denoted $\ddot{\mathsf{P}}_1$, the middle 3×1 submatrix (column vector) is denoted $\ddot{\mathbf{P}}_2$, and the right-most 3×3 submatrix is denoted $\ddot{\mathsf{P}}_3$.

$$\ddot{\mathsf{P}} = \begin{bmatrix} \mathsf{R} & \text{-}\mathsf{R}\mathbf{T} & \mathsf{R}[\text{-}\mathbf{T}]_\times \end{bmatrix} = \begin{bmatrix} \ddot{\mathsf{P}}_1 & \ddot{\mathbf{P}}_2 & \ddot{\mathsf{P}}_3 \end{bmatrix} \tag{4.13}$$

$\ddot{\mathsf{P}}$ has 21 entries, but since it encodes the camera pose, it has only 6 DoF. This means it has 14 nonlinear constraints (homogeneity of the matrix accounts for the 1 remaining DoF). Ignoring the nonlinear constraints, which are not taken into account during the least-squares estimation, $\ddot{\mathsf{P}}$ has 20 DoF. Each point-line correspondence generates 1 independent linear equation (4.10) and each line-line correspondence generates 2 independent linear equations (4.11). Since $\ddot{\mathbf{P}}_2$ is determined only by point-line correspondences and since it has 3 DoF, at least 3 3D points are required to fully determine it. An analogy holds for $\ddot{\mathsf{P}}_3$: since it is determined only by line-line correspondences and since it has 9 DoF, at least 5 (in theory 4½) 3D lines are required to fully determine it. The required number of m line-line correspondences and n point-line correspondences is thus $m = 9$, $n = 3$, or $m = 5$, $n = 10$, or something in between satisfying the inequality $(n + 2m) \geq 20$, see Table 4.1. In such minimal cases, the points must be distributed equally among the lines, i. e. each point or a pair of points must lie on a different line; otherwise, the system of equations would be under-determined.

Table 4.1: Minimal numbers of line-line and point-line correspondences required for the DLT-Combined-Lines method.

point-line	$n =$	3	4	5	6	7	8	9	10
line-line	$m =$	9	8	8	7	7	6	6	5

Let us proceed with the description of the algorithm. Please notice that the prenormalization procedure will be unusually described *after* the definition of a measurement matrix, because prenormalization is strongly motivated by its structure.

Linear Estimation of the Combined Projection Matrix

The combined projection matrix $\ddot{\mathsf{P}}$ and its estimate $\ddot{\mathsf{P}}'$ are 3×7, so the combined measurement matrix $\ddot{\mathsf{M}}$ has 21 columns. Number of its rows depends on n – the number of point-line correspondences $\mathbf{X}_i \leftrightarrow \mathbf{l}_i$, $(i = 1 \dots n)$, and on m – the number of line-line correspondences $\mathbf{L}_j \leftrightarrow \mathbf{l}_j$, $(j = n + 1 \dots n + m)$. The minimal values of n and m depend on each other and are given in Table 4.1. Each point-line correspondence (4.10) leads to one row of $\ddot{\mathsf{M}}$, and each line-line correspondence (4.11) gives rise to three rows of $\ddot{\mathsf{M}}$ (Matlab notation is used to index the matrix elements):

$$\ddot{\mathsf{M}}_{(i,\ :)} = (\mathbf{X}_i^\top \ \ 0 \ \ 0 \ \ 0) \otimes \mathbf{l}_i^\top \ , \tag{4.14}$$

$$\ddot{\mathsf{M}}_{(3j-n-2\ :\ 3j-n,\ :)} = (\mathbf{U}_j^\top \ \ 0 \ \ \mathbf{V}_j^\top) \otimes [\mathbf{l}_j]_\times \ . \tag{4.15}$$

The combined measurement matrix $\ddot{\mathsf{M}}$ is thus $(n + 3m) \times 21$. Note that only two of the three rows of $\ddot{\mathsf{M}}$ defined by Eq. (4.15) are needed for each line-line correspondence, because they are linearly dependent. Our experiments showed that using all three rows brings no advantage, so only two of them are used in practice. In this case, $\ddot{\mathsf{M}}$ is only $(n + 2m) \times 21$. See Appendix A of the thesis for derivations of Eq. (4.14) and (4.15).

The combined measurement matrix $\ddot{\mathsf{M}}$ can also be constructed by stacking and aligning the

point measurement matrix $\dot{\mathsf{M}}$ and the line measurement matrix $\bar{\mathsf{M}}$:

$$\ddot{\mathsf{M}} = \begin{bmatrix} \dot{\mathsf{M}}_{n\times 12} & & \mathsf{0}_{n\times 9} \\ \bar{\mathsf{M}}_{(:,\ 1:9)} & \mathsf{0}_{3m\times 3} & \bar{\mathsf{M}}_{(:,\ 10:18)} \end{bmatrix} . \tag{4.16}$$

Remark 4.1: It is advisable to scale both $\dot{\mathsf{M}}$ and $\bar{\mathsf{M}}$ so that the sums of squares of their entries are equal. (If they were not, it would negatively affect the scales of those parts of the solution $\ddot{\mathbf{p}} = \mathrm{vec}(\ddot{\mathsf{P}})$, which are determined exclusively by $\dot{\mathsf{M}}$ or $\bar{\mathsf{M}}$, but not by both of them. These are the entries 10-12 and 13-21 of $\ddot{\mathbf{p}}$, which contain estimates of translation. See the middle and right part of $\ddot{\mathsf{P}}$ in Eq. (4.13).)

Remark 4.2: The method can easily be extended to point-point correspondences (4.12) by adding extra rows to $\ddot{\mathsf{M}}$. Each of the p point-point correspondences $\mathbf{X}_k \leftrightarrow \mathbf{x}_k$, $(k = n+m+1 \ldots n+m+p)$ generates three rows

$$\ddot{\mathsf{M}}_{(3k-n-m-2\ :\ 3k-n-m,\ :)} \quad = \quad (\mathbf{X}_i^\top \ \ 0 \ \ 0 \ \ 0) \otimes [\mathbf{x}_i]_\times \ , \tag{4.17}$$

two of which are linearly independent. See Appendix A.2 of the thesis for a derivation of Eq. (4.17).

Prenormalization

Prenormalization of 2D lines is rather complicated in this case. The problem is that a 2D line $\mathbf{l}$ is in the direct form and on the opposite side than the line projection matrix $\bar{\mathsf{P}}$ in Eq. (4.11), and it is in the transposed form and on the same side like the point projection matrix $\dot{\mathsf{P}}$ in Eq. (4.10). Thus, when undoing the effect of a prenormalizing 2D transformation t, the inverse transformation is t^{-1} for $\bar{\mathsf{P}}$, and $\mathsf{t}^\top$ for $\dot{\mathsf{P}}$. Since both $\dot{\mathsf{P}}$ and $\bar{\mathsf{P}}$ are parts of $\ddot{\mathsf{P}}$, both inverse transformations must be identical ($\mathsf{t}^\top = \mathsf{t}^{-1}$). However, this only holds for a 2D rotation, which is practically useless as a prenormalizing transformation. It is thus suggested not to prenormalize 2D lines at all.

Prenormalization of 3D points and 3D lines is also nontrivial, because transformations of 3D space affect the coordinates of points and lines differently. However, it can be achieved by pursuing the goal from the beginning of Section 4.2: to center the data around the origin by translation, and to scale them s. t. an average coordinate has the absolute value of 1.

Please note that translation and scaling affects only the $\mathbf{U}$ part of a 3D line $\mathbf{L}$, and only the $(X_1\ X_2\ X_3)^\top$ part of a 3D point $\mathbf{X}$. Therefore, (i) the unaffected parts of $\mathbf{L}$ and $\mathbf{X}$ (i. e. $\mathbf{V}$ and X_4) must be adjusted beforehand: Each 3D line and each 3D point is normalized by multiplication by a nonzero scale factor, so that $\|\mathbf{V}\| = \sqrt{3}$, and $X_4 = 1$. Note that this adjustment does *not* change the spatial properties of 3D points/lines. Then, (ii) translation is applied to center the 3D points around the origin[1]. Although the translation is intuitively correct (it results in zero mean

[1] Another possible translation is to center the 3D lines using the Generalized Weiszfeld algorithm [1] as it is done in Algorithm 3. However, our experiments showed that the two possible translations yield nearly identical robustness of the method. It is thus suggested to translate the 3D structure to the centroid of points, because its computation is cheaper.

of 3D points), it is not optimal in terms of entries of the measurement matrix (joint zero mean of $(X_1\ X_2\ X_3)^\top$ and $\mathbf{U}$). Therefore, (iii) another translation is applied to achieve a joint zero mean of all $(X_1\ X_2\ X_3)^\top$ and $\mathbf{U}$. The translation can be easily computed in closed form using Algorithm 6. Finally, (iv) anisotropic scaling is applied so that the average magnitudes of all X_1 and L_1, X_2 and L_2, X_3 and L_3, and X_4 and $\mathbf{V}$ are equal, i. e.

$$\begin{aligned} &\text{mean}_{|\circ|}(\{X_{i,1}\}) + \text{mean}_{|\circ|}(\{L_{j,1}\}) = \\ &= \text{mean}_{|\circ|}(\{X_{i,2}\}) + \text{mean}_{|\circ|}(\{L_{j,2}\}) = \\ &= \text{mean}_{|\circ|}(\{X_{i,3}\}) + \text{mean}_{|\circ|}(\{L_{j,3}\}) = \\ &= \text{mean}_{|\circ|}(\{X_{i,4}\}) + \text{mean}_{|\circ|}(\{\mathbf{V}_j\}) \ . \end{aligned} \tag{4.18}$$

This ensures that also the corresponding blocks of the combined measurement matrix $\ddot{\mathsf{M}}$ will have equal average magnitude. The very last step of prenormalization (v) is not applied to the input primitives, but to the measurement matrix after its construction. Its point- and line-related parts $\dot{\mathsf{M}}$ and $\bar{\mathsf{M}}$ should be scaled as stated in Remark 4.1 above. The whole prenormalization is summarized in Algorithm 5.

Extraction of Pose Parameters

The estimates of a rotation matrix R and a translation vector $\mathbf{T}$ are multiple in the combined projection matrix $\ddot{\mathsf{P}}$ (4.13). Moreover, the left-most R is determined by twice as many equations. This can be exploited to estimate the camera pose more robustly. In the following text, the definitions of submatrices $\ddot{\mathsf{P}}_1$, $\ddot{\mathbf{P}}_2$, and $\ddot{\mathsf{P}}_3$ from Eq. (4.13) are used.

First, the scale of the estimated combined projection matrix $\ddot{\mathsf{P}}'$ is corrected using Algorithm 1, yielding $s\ddot{\mathsf{P}}'$. The first estimate of R is in the direct form in $s\ddot{\mathsf{P}}'_1$, from which it can be extracted using Algorithm 2, yielding R_1. The first estimate of $\mathbf{T}$ is in $s\ddot{\mathbf{P}}'_2$, premultiplied by -R. It can be recovered as $\mathbf{T}_2 = \text{-}\mathsf{R}_1^\top s\ddot{\mathbf{P}}'_2$. The second estimates of R and $\mathbf{T}$ are in the form of an essential matrix in $s\ddot{\mathsf{P}}'_3$, from which they can be extracted using Algorithm 4, yielding R_3 and $\mathbf{T}_3$.

Now, the question is how to combine R_1, R_3, and $\mathbf{T}_2$, $\mathbf{T}_3$. Our experiments showed that R_1 is usually more accurate than R_3, probably because it is determined by twice as many equations (generated by both line-line and point-line correspondences). The experiments also showed that $\mathbf{T}_2$ is usually more accurate than $\mathbf{T}_3$. This is probably because $\ddot{\mathbf{P}}'_2$ has no redundant DoF, contrary to $\ddot{\mathsf{P}}'_3$, which has 3 redundant DoF. However, the estimates can be combined so that the result is even more accurate. Since the error vectors of $\mathbf{T}_2$ and $\mathbf{T}_3$ tend to have opposite directions, a suitable interpolation between them can produce more accurate position estimate

$$\mathbf{T} = k \cdot \mathbf{T}_2 + (1 - k) \cdot \mathbf{T}_3 \ . \tag{4.19}$$

The value of k should be between 0 and 1. Based on grid search, an optimal value of 0.7 has been found (the error function has a parabolic shape).

Regarding the rotation estimates, the grid search discovered R_1 is indeed more accurate than R_3. However, R_1 is not fully 'compatible' with $\mathbf{T}$ in terms of reprojection error[2]. Interpolating between

[2] As an example, imagine a camera located left to its ground truth position and oriented even more left.

Algorithm 5: Prenormalization of 3D points and 3D lines in DLT-Combined-Lines.
Note: See Section 2.1 for the definition of function $\text{mean}_\circ(.)$.

Input: A set of n 3D points $\{\mathbf{X}_i\}$, $i = 1 \dots n$.
Input: A set of m 3D lines $\{\mathbf{L}_j\}$, $j = n+1 \dots n+m$.

1. For all points $\mathbf{X}_i$ and lines $\mathbf{L}_j$ do:

$$\mathbf{X}_i = \frac{\mathbf{X}_i}{X_{i,4}} \ , \qquad \mathbf{L}_j = \frac{\sqrt{3}}{\|\mathbf{V}_j\|} \cdot \mathbf{L}_j$$

2. $\mathbf{T}_1 \leftarrow \text{mean}(\{\mathbf{X}_{(1:3,\ i)}\})$ ◁ centroid of points
3. For all points $\mathbf{X}_i$ and lines $\mathbf{L}_j$ do:

$$\mathbf{X}_i = \begin{bmatrix} \mathsf{I} & -\mathbf{T}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_i \ , \qquad \mathbf{L}_j = \begin{bmatrix} \mathsf{I} & [-\mathbf{T}_1]_\times \\ \mathsf{0} & \mathsf{I} \end{bmatrix} \mathbf{L}_j$$ ◁ first translation

4. $\mathbf{T}_2 \leftarrow \arg\min_{\mathbf{T}} \left(\text{mean}_\circ(\ \{\mathbf{X}_{(1:3,\ i)} - \mathbf{T}\} \cup \{\mathbf{U}_j - \mathbf{T} \times \mathbf{V}_j\}\ )\right)^2$ ◁ use Algorithm 6
5. For all points $\mathbf{X}_i$ and lines $\mathbf{L}_j$ do:

$$\mathbf{X}_i = \begin{bmatrix} \mathsf{I} & -\mathbf{T}_2 \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_i \ , \qquad \mathbf{L}_j = \begin{bmatrix} \mathsf{I} & [-\mathbf{T}_2]_\times \\ \mathsf{0} & \mathsf{I} \end{bmatrix} \mathbf{L}_j$$ ◁ second translation

6. $$S_\text{X} \leftarrow \frac{\text{mean}_{|\circ|}(\{X_{i,4}\}) + \text{mean}_{|\circ|}(\{\mathbf{V}_j\})}{\text{mean}_{|\circ|}(\{X_{i,1}\}) + \text{mean}_{|\circ|}(\{L_{j,1}\})}$$

$$S_\text{Y} \leftarrow \frac{\text{mean}_{|\circ|}(\{X_{i,4}\}) + \text{mean}_{|\circ|}(\{\mathbf{V}_j\})}{\text{mean}_{|\circ|}(\{X_{i,2}\}) + \text{mean}_{|\circ|}(\{L_{j,2}\})}$$

$$S_\text{Z} \leftarrow \frac{\text{mean}_{|\circ|}(\{X_{i,4}\}) + \text{mean}_{|\circ|}(\{\mathbf{V}_j\})}{\text{mean}_{|\circ|}(\{X_{i,3}\}) + \text{mean}_{|\circ|}(\{L_{j,3}\})}$$

7. For all points $\mathbf{X}_i$ and lines $\mathbf{L}_j$ do:

$$\mathbf{X}_i = \begin{bmatrix} S_\text{X} & & & \\ & S_\text{Y} & & \mathbf{0} \\ & & S_\text{Z} & \\ \mathbf{0}^\top & & & 1 \end{bmatrix} \mathbf{X}_i \ , \qquad \mathbf{L}_j = \begin{bmatrix} S_\text{X} & & & \\ & S_\text{Y} & & 0 \\ & & S_\text{Z} & \\ 0 & & & \mathsf{I} \end{bmatrix} \mathbf{L}_j$$ ◁ scaling

Output: A set of n prenormalized 3D points $\{\mathbf{X}_i\}$, $i = 1 \dots n$.
Output: A set of m prenormalized 3D lines $\{\mathbf{L}_j\}$, $j = n+1 \dots n+m$.

Algorithm 6: Finding a translation $\mathbf{T}_2$ of 3D points $\{\mathbf{X}_i\}$ and 3D lines $\{\mathbf{L}_j\}$ s. t. the mean of $\{\mathbf{X}_{(1:3,\ i)}\} \cup \{\mathbf{U}_j\}$ will be zero after the translation.

Input: A set of n 3D points $\{\mathbf{X}_i\}$, $i = 1 \dots n$.
Input: A set of m 3D lines $\{\mathbf{L}_j\}$, $j = n+1 \dots n+m$.

1. $a \leftarrow n + 2m \quad ,$

$$b \leftarrow \sum_j L_{j,1} + \sum_i X_{i,1} \quad , \qquad c \leftarrow \sum_j L_{j,2} + \sum_i X_{i,2} \quad , \qquad d \leftarrow \sum_j L_{j,3} + \sum_i X_{i,3} \quad ,$$

$$e \leftarrow \sum_j L_{j,4} \quad , \qquad f \leftarrow \sum_j L_{j,5} \quad , \qquad g \leftarrow \sum_j L_{j,6}$$

2. $$T_X \leftarrow -\frac{a^2 b + b e^2 - a c g + a d f + c e f + d e g}{a(a^2 + e^2 + f^2 + g^2)}$$

$$T_Y \leftarrow -\frac{a^2 c + c f^2 + a b g - a d e + b e f + d f g}{a(a^2 + e^2 + f^2 + g^2)}$$

$$T_Z \leftarrow -\frac{a^2 d + d g^2 - a b f + a c e + b e g + c f g}{a(a^2 + e^2 + f^2 + g^2)}$$

Output: Translation $\mathbf{T}_2 = (T_X\ T_Y\ T_Z)^\top$.

R_1 and R_3 yields an orientation R 'compatible' with $\mathbf{T}$:

$$\mathsf{R} = \mathsf{R}_1 \cdot \exp(k \cdot \log(\mathsf{R}_1^\top \mathsf{R}_3)) \quad . \tag{4.20}$$

Here, 'exp' and 'log' denote matrix exponential and matrix logarithm, respectively. The whole pose extraction procedure is summarized in Algorithm 7.

Algorithm 7: Extraction of pose parameters from the estimate $\ddot{\mathsf{P}}'$ of a combined projection matrix.

Input: An estimate $\ddot{\mathsf{P}}'$ of a line projection matrix $\ddot{\mathsf{P}}$.
Input: Corrective scale factor s.

1. $\left[\ \ddot{\mathsf{P}}'_1 \quad \ddot{\mathbf{P}}'_2 \quad \ddot{\mathsf{P}}'_3\ \right] \leftarrow \ddot{\mathsf{P}}'$ ◁ divide into submatrices
2. Extract R_1 from $\ddot{\mathsf{P}}'_1$ using Algorithm 2.
3. $\mathbf{T}_2 = \text{-}\mathsf{R}_1^\top s \ddot{\mathbf{P}}'_2$
4. Extract R_3, $\mathbf{T}_3$ from $\ddot{\mathsf{P}}'_3$ using Algorithm 4.
5. $\mathsf{R} = \mathsf{R}_1 \cdot \exp(k \cdot \log(\mathsf{R}_1^\top \mathsf{R}_3))$ ◁ interpolate
   $\mathbf{T} = k \cdot \mathbf{T}_2 + (1-k) \cdot \mathbf{T}_3$

Output: R, $\mathbf{T}$.

4.6 Algebraic Outlier Rejection

To deal with outliers, the DLT-based methods can be equipped with an Algebraic Outlier Rejection module. The AOR scheme, developed originally for a PnP method, was described in Section 3.4. However, our experiments showed that its application to DLT-based LPnL methods requires a

different setting.

The difference is in the strategy for choosing the threshold $\varepsilon_{\max}$. The authors [14] recommend $\varepsilon_{\max} = \mathrm{Q}_{25}(\varepsilon_1, \ldots, \varepsilon_n)$ which is the algebraic error of the correspondence that is at the boundary of the 25th percentile. However, our experiments showed that the strategy is not robust enough for LPnL methods. A slightly different strategy is thus suggested with a good trade-off between robustness and the number of iterations: At the beginning, line correspondences with error up to the 90th percentile are accepted. In further iterations, the percentile number is progressively decreased until it reaches 25. The strategy is thus $\varepsilon_{\max} = \mathrm{Q}_p(\varepsilon_1, \ldots, \varepsilon_n)$, where $\mathrm{Q}_p(.)$ denotes the p-th percentile and p decreases following the sequence 90, 80, ..., 30. Then, it remains constant 25 until error of the solution stops decreasing. This strategy usually leads to approximately 10 iterations.

Remark 4.3: It is important *not* to prenormalize the data before using AOR because it will impede the identification of outliers. Prenormalization of *inliers* should be done just before the last iteration.

Compared to RANSAC, the greatest benefit of this approach is a low runtime *independent* of the fraction of outliers. On the other hand, the break-down point is roughly between 40 % and 70 % of outliers, depending on the underlying LPnL method, whereas RANSAC, in theory, can handle any fraction of outliers.

4.7 Summary

Although the three above described DLT-based PnL methods share a common basis, they differ in certain details. Their properties are summarized in Table 4.2. All three methods work exclusively with lines in the image space. In the scene space, however, DLT-Lines works with points, DLT-Plücker-Lines works with lines, and DLT-Combined-Lines works with both points and lines. The question is whether utilization of 3D lines, i.e. line-line correspondences, does improve the accuracy and robustness of camera pose estimation while preserving the efficiency of DLT-based methods.

The most important difference is in the projection matrices. The line projection matrix $\bar{\mathsf{P}}$ of DLT-Plücker-Lines encodes the rotation matrix R in a form of an essential matrix having only 3 redundant DoF. This is a promise of a more accurate estimation of camera orientation compared to DLT-Lines, where R is encoded in a direct form having 6 redundant DoF. The same holds for the combined projection matrix $\ddot{\mathsf{P}}$ of DLT-Combined-Lines. Moreover, $\ddot{\mathsf{P}}$ contains multiple estimates of both R and $\mathbf{T}$. A suitable combination of the estimates may further increase the accuracy of the final pose.

Prenormalization of the inputs of the methods pursues a common goal of having the data centered around the origin with a unit average absolute value of the coordinates. This goal is motivated by a good condition of the resulting linear system. Generally, it can be achieved by applying translation and scaling to the inputs. In the case of DLT-Combined-Lines, it is more complicated due to different effects of the transformations on coordinates of points and lines in the 3D space. Prenormalization of image lines is futile in this case as it is restricted to rotations only.

In principle, the methods could also be extended to estimate the pose of an uncalibrated camera,

Table 4.2: Comparison of the DLT-based LPnL methods.

		DLT-Lines	DLT-Plücker-Lines	DLT-Combined-Lines
Input	2D (image)	2D lines	2D lines	2D lines
	- prenormalization	translation (in dual space) scaling (in dual space)	translation (in dual space) scaling (in dual space)	—
	3D (scene)	3D points	3D lines	3D points + 3D lines
	- prenormalization	translation scaling	multiplication by a constant translation scaling	multiplication by a constant translation translation scaling
Minimum of lines		6	9	$5 \begin{cases} \text{5 lines + 10 points} \\ \vdots \\ \text{9 lines + 3 points} \end{cases}$
specification		12 points, 2 on each line	—	$m+n, \quad \text{s.t. } (2m+n) \geq 20$
Projection matrix		$\dot{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & -\mathsf{R}\mathbf{T} \end{bmatrix}_{3\times 4}$	$\bar{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & \mathsf{R}[-\mathbf{T}]_\times \end{bmatrix}_{3\times 6}$	$\ddot{\mathsf{P}} \approx \begin{bmatrix} \mathsf{R} & -\mathsf{R}\mathbf{T} & \mathsf{R}[-\mathbf{T}]_\times \end{bmatrix}_{3\times 7}$
Constraint equations		$\mathbf{l}^\top \dot{\mathsf{P}} \mathbf{X} = 0$	$\mathbf{l} \approx \bar{\mathsf{P}} \mathbf{L}$	$\mathbf{l}^\top \ddot{\mathsf{P}} \begin{pmatrix} \mathbf{X}^\top & 0 & 0 & 0 \end{pmatrix}^\top = 0$ $\mathbf{l} \approx \ddot{\mathsf{P}} \begin{pmatrix} \mathbf{U}^\top & 0 & \mathbf{V}^\top \end{pmatrix}^\top$

i. e. to estimate both extrinsic and intrinsic parameters of a camera. The corresponding projection matrix $\dot{\mathsf{P}}$, $\bar{\mathsf{P}}$ or $\ddot{\mathsf{P}}$ would be premultiplied by the upper-triangular 3×3 camera calibration matrix K in this case, so the number of unknowns of the resulting linear system and also the number of DoF of the projection matrix would grow from 6 up to 11 (depending on the number of intrinsic parameters). According to preliminary experiments, robustness of all three methods drops considerably in this case, making them useless for practical applications. A better choice would be a method tailored specifiacaly for estimation of parameters of an uncalibrated camera in this case, such as [5].

The minimum of required lines is conditioned by the size and structure of the estimated projection matrix. It ranges from 9 lines for DLT-Plücker-Lines over 6 lines for DLT-Lines to only 5 lines for DLT-Combined-Lines.

Chapter 5

Experimental Evaluation and Applications

The goal of the thesis was to improve the accuracy and robustness of the state-of-the-art in pose estimation from lines by designing a new DLT-based method utilizing line-line correspondences. The method should also be fast comparably to other LPnL methods. Two new methods were proposed in the previous chapter: DLT-Plücker-Lines and DLT-Combined-Lines.

To verify that the goal was achieved, the newly proposed methods were tested using both synthetic and real data and their performance was compared to the state-of-the-art methods. The real data comprised building exteriors, an indoor corridor and small-scale objects on a table. The tested criteria were following.

1. The primary criterion of experiments was *accuracy* because it arguably is the primary objective of pose estimation. It was evaluated using both synthetic lines in Section 5.1 and real data in Section 5.2.
2. A secondary objective, although equally important from a practical point of view, is robustness to image noise, because noise is always present in measurements in practice. Accordingly, *robustness to image noise* was evaluated using synthetic lines in Section 5.1.
3. Since the proposed methods were also required to be fast comparably to other methods, their *speed* was measured using synthetic lines in Section 5.1.

Besides the main criteria, the following aspects were also investigated to have a more comprehensive knowledge about behavior of the proposed methods.

- Because methods for pose estimation are known to be prone to singular or quasi-singular configurations of 3D primitives in general, robustness to *quasi-singular line configurations* was examined (see Section 5.2 of the thesis).
- From an application point of view, identification and rejection of mismatched line correspondences (i. e. outliers) is a frequent scenario. Therefore, the methods were also tested for robustness and speed when plugged into an outlier rejection scheme or into a RANSAC loop using synthetic lines (see Section 5.3 of the thesis).
- Lastly, the camera poses estimated by the methods were used as an initialization for Bundle Adjustment (BA) in Section 5.2 to see how the initialization affects its convergence and runtime.

The accuracy of pose estimates is expressed in terms of position error and orientation error of the camera and in terms of reprojection error of the lines. The three error measures should cover

majority of applications for which pose estimation methods are used. For example, robot localization requires small position error, visual servoing requires both small position and orientation error, whereas augmented reality applications or BA favour small reprojection error. The error measures are defined as follows:

$\Delta\mathrm{T}$ **Position error** is the distance $\|\mathbf{T}' - \mathbf{T}\|$ from the estimated position $\mathbf{T}'$ to the true position $\mathbf{T}$.

$\Delta\Theta$ **Orientation error** was calculated as follows. The difference between the true and estimated rotation matrix ($\mathsf{R}^\top\mathsf{R}'$) is converted to axis-angle representation ($\mathbf{E}$, Θ) and the absolute value of the difference angle $|\Theta|$ is considered as the orientation error.

$\Delta\pi$ **Reprojection error** is an integral of squared distance between points on the image line segment and the projection of an infinite 3D line, averaged over all individual lines.

The proposed methods were evaluated and compared with state-of-the-art methods, which are listed below together with corresponding marks used throughout this chapter.

- ▶ **Ansar**, the method by Ansar and Daniilidis [3], implementation from [33].
- ● **Mirzaei**, the method by Mirzaei and Roumeliotis [26].
- ◆ **RPnL**, the method by Zhang et al. [35].
- ◆ **ASPnL**, the method by Xu et al. [33].
- ★ **LPnL_Bar_LS**, the method by Xu et al. [33].
- ✶ **LPnL_Bar_ENull**, the method by Xu et al. [33].
- ▲ **DLT-Lines**, the method by Hartley and Zisserman [18, p. 180] described in Section 4.3, my implementation.
- ▼ **DLT-Plücker-Lines**, our method published in [II] and described in Section 4.4.
- ■ **DLT-Combined-Lines**, our method published in [I] and described in Section 4.5.

All of the methods were implemented in Matlab. The implementations originate from the respective authors, if not stated otherwise.

5.1 Synthetic Lines

Monte Carlo simulations with synthetic lines were performed under the following setup: at each trial, m 3D line segments were generated by randomly placing $n = 2m$ line endpoints inside a cube spanning 10^3 m which was centered at the origin of the world coordinate system. For the methods which work with 3D points, the line endpoints were used. A virtual pinhole camera with image size of 640×480 pixels and focal length of 800 pixels was placed randomly in the distance of 25 m from the origin. The camera was then oriented so that it looked directly at the origin, having all 3D line segments in its field of view. The 3D line segments were projected onto the image plane. Coordinates of the 2D endpoints were then perturbed with independent and identically distributed Gaussian noise with standard deviation of σ pixels. 1000 trials were carried out for each combination of the parameters m and σ, where $m = 3$ – 10,000 lines and $\sigma = 1$, 2, 5, 10 and 20 pixels.

Accuracy and Robustness

Accuracy of pose estimation and robustness to image noise of each method was evaluated by measuring the estimated and true camera pose while varying m and σ similarly to [26].

The results showing accuracy of the methods and their robustness to image noise are depicted in Figure 5.1. For the sake of brevity, only noise levels of $\sigma = 2$ and 10 pixels are shown. The complete distribution of errors is presented in Appendix B of the thesis. Errors for each method are plotted from the minimal number of lines to 10,000 lines (or less, if the method runs too long or if it has enormous memory requirements). In the following text, the method names are typeset in bold and they are often augmented with their plot marks to ease referencing into result charts.

The results show high sensitivity to noise of **Ansar** ▶. Even under slight image noise $\sigma = 2$ pixels, the measured accuracy is poor. The other non-LPnL methods (**Mirzaei** ●, **RPnL** ♦, **ASPnL** ♦) outperform the LPnL methods for low number of lines (3 – 10), as expected. **ASPnL** is the most accurate among them. An exception is the LPnL method **LPnL_Bar_ENull** ✶, accuracy of which is close to **ASPnL**. It even outperforms **ASPnL** in the case of strong image noise ($\sigma = 10$ pixels), see Figure 5.1 (b, d, f).

For high number of lines (100 – 10,000), the LPnL methods outperform the non-LPnL ones. **LPnL_Bar_ENull** ✶ and **DLT-Combined-Lines** ■ are significantly most accurate in both orientation and position estimation, and they also yield the lowest reprojection error. With increasing number of lines, accuracy of the LPnL methods further increases, while errors of the non-LPnL methods do not fall below a certain level. This gets more obvious with increasing levels on noise. Each of the LPnL methods also eventually reaches its limit, as it can bee seen in Figure 5.1 (d, f). However, the accuracy limits of non-LPnL methods lag behind the limits of LPnL methods. Moreover, the non-LPnL methods often yield completely wrong pose estimates, as it can be seen in the distribution of errors in Figures B.1 – B.15 in Appendix B of the thesis.

DLT-Lines ▲ and **LPnL_Bar_LS** ★ behave nearly identically, the latter being slightly more accurate. The only difference between the two is the use of barycentric coordinates, which is probably the cause of the slightly better results. However, **DLT-Lines** proves to be more accurate in position estimation and reprojection under strong image noise. **DLT-Plücker-Lines** ▼ keeps up with the two aforementioned methods for 25 and more lines.

The best accuracy on many lines is achieved by the **LPnL_Bar_ENull** ✶ and **DLT-Combined-Lines** ■ methods, being the best in all criteria. While they are comparable in orientation estimation, **DLT-Combined-Lines** outperforms **LPnL_Bar_ENull** in estimation of camera position and in reprojection for many lines. The higher accuracy of **DLT-Combined-Lines** is most apparent under strong image noise, see Figure 5.1 (d, f).

The distributions of errors of the individual methods over all 1000 trials are provided in Figures B.1 – B.15 in Appendix B of the thesis.

Speed

Efficiency of each method was evaluated by measuring runtime on a desktop PC with a quad core Intel i5-661 3.33 GHz CPU and 10 GB of RAM. As it can be seen in Figure 5.2, the only method with $O(m^2)$ computational complexity in the number of lines m is **Ansar** ▶. The space complexity of the

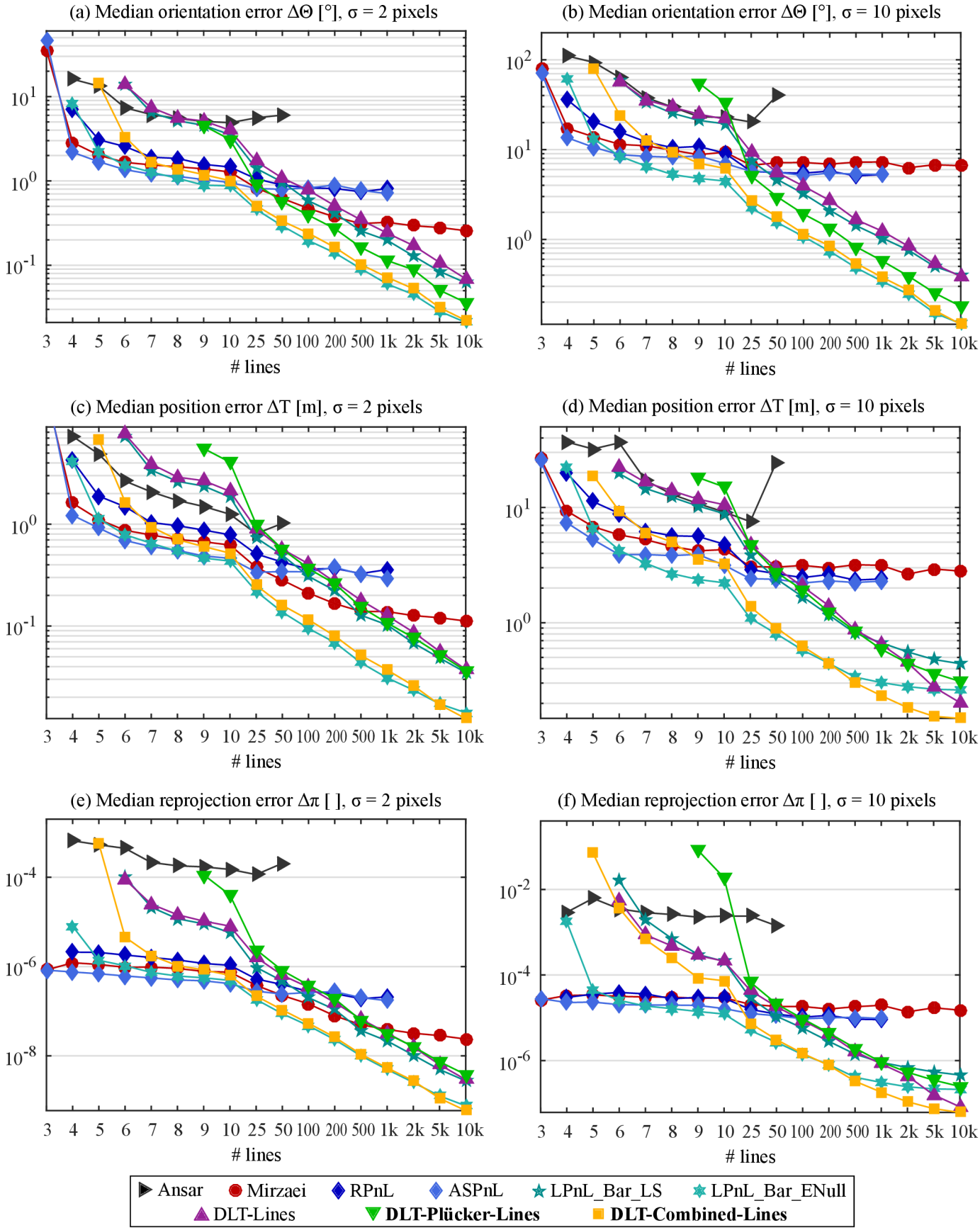

Figure 5.1: Median orientation errors (*top*), position errors (*middle*) and reprojection errors (*bottom*) as a function of the number of lines for two levels of image noise (*left*: $\sigma = 2$ pixels, *right*: $\sigma = 10$ pixels). Each data point was computed from 1000 trials.

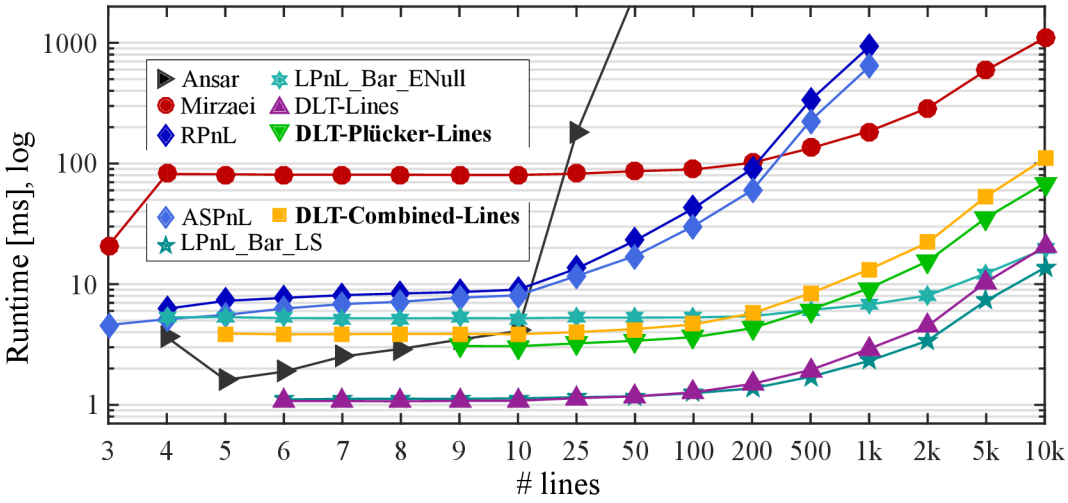

Figure 5.2: Runtimes as a function of the number of lines, averaged over 1000 trials. Logarithmic vertical axis.

used implementation is apparently also quadratic. It was not possible to execute the method already for 100 lines due to lack of computer memory. Other tested methods have $O(m)$ computational complexity. However, the runtimes differ substantially. It is apparent that the LPnL methods are significantly faster than the non-LPnL methods.

RPnL ♦ and **ASPnL** ♦, being related methods, are nearly equally fast. Runtimes of both methods rise steeply with increasing number of lines, reaching 630.2 ms on 1000 lines for **ASPnL**. The two methods were not evaluted for more lines. Runtime of **Mirzaei** ●, on the other hand, grows very slowly, spending 155.2 ms on 1000 lines. However, **Mirzaei** is slower than **RPnL** for $m < 200$ lines. This fact is caused by computation of a 120×120 Macaulay matrix in Mirzaei's method which has an effect of a constant time penalty.

The LPnL methods are one to two orders of magnitude faster than the non-LPnL methods. The fastest two are **DLT-Lines** ▲ and **LPnL_Bar_LS** ★, spending about 1 ms on 10 lines, and not more than 3 ms on 1000 lines. Slightly slower are **DLT-Plücker-Lines** ▼, **DLT-Combined-Lines** ■ and **LPnL_Bar_ENull** ✶, spending about 3 – 5 ms on 10 lines, and about 6 – 12 ms on 1000 lines. The slowdown factor for **DLT-Plücker-Lines** is the prenormalization of 3D lines. This is also the case of **DLT-Combined-Lines**, where a measurement matrix of a double size must be additionally decomposed compared to the competing methods, see Eq. (4.16). Computationally demanding part of **LPnL_Bar_ENull** is the effective null space solver carrying out Gauss-Newton optimization.

5.2 Real-World Buildings and Man-Made Objects

In this section, the proposed methods are validated on real-world data and compared to state-of-the-art methods. Ten datasets were utilized, which contain images with detected 2D line segments, reconstructed 3D line segments, and camera projection matrices. Example images from the datasets are shown in Figure 5.3. Number of images in each dataset ranged from 3 to 72 and number of lines ranged from 30 to 1841. Line correspondences are also given except for datasets Timberframe House, Building Blocks and Street in which case the correspondences were established automatically

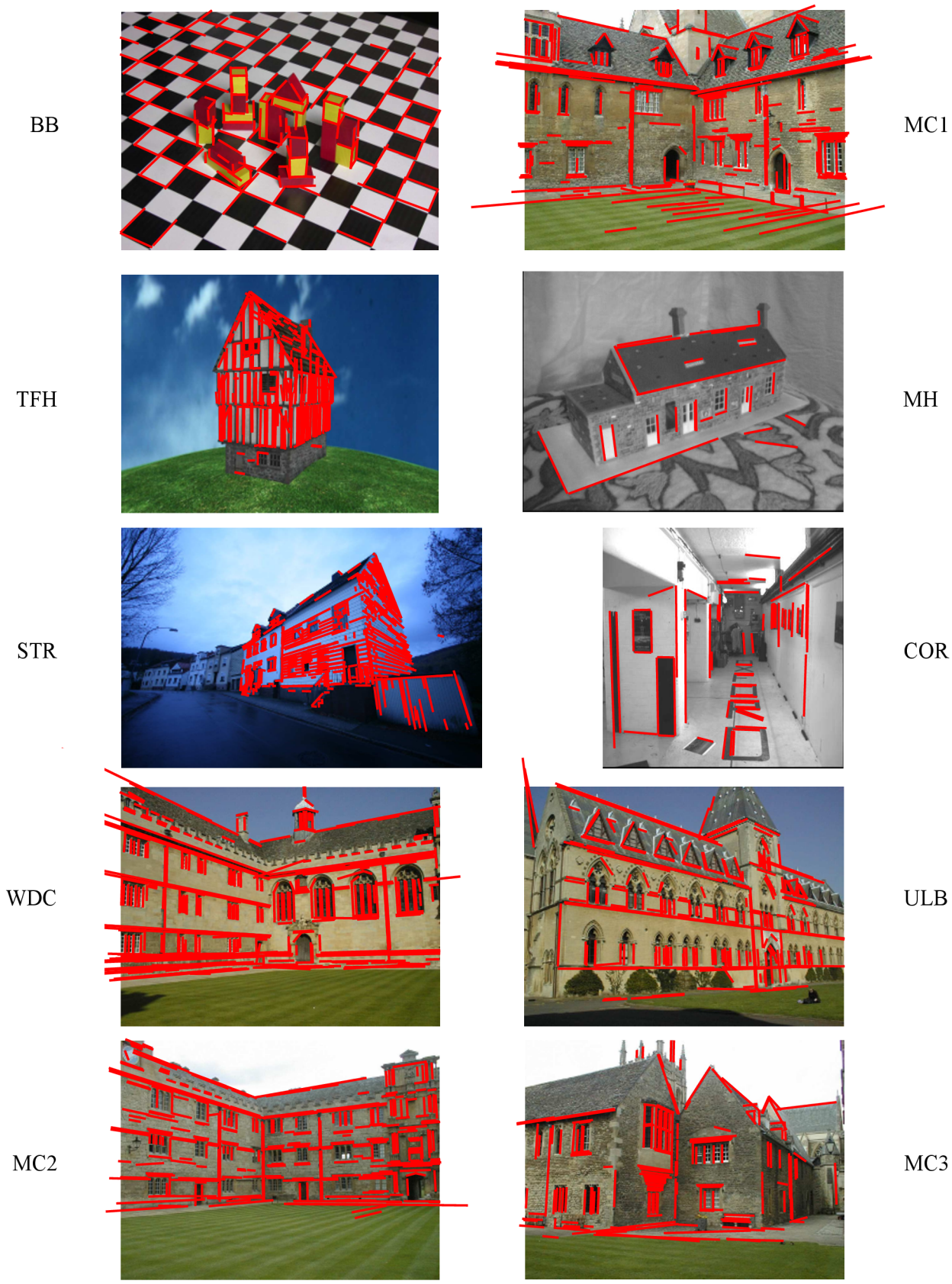

Figure 5.3: Example images from used datasets. The images are overlaid with reprojections of 3D line segments using the camera pose estimated by the proposed method DLT-Combined-Lines.

based on geometric constraints. The Timberframe House dataset contains rendered images, while the rest contains real images captured by a physical camera. The Building Blocks and Model House datasets capture small-scale objects on a table, the Corridor dataset captures an indoor corridor, and the other six datasets capture exterior of various buildings. The Building Blocks dataset is the most challenging because many line segments lie in a common plane of a chessboard.

Accuracy

Each PnL method was run on the data, and the errors in camera orientation, camera position and reprojection of lines were averaged over all images in each dataset. The mean errors achieved by all methods on individual datasets are given in Table 5.1 and visualized in Figure 5.4.

On datasets with small number of lines (MH: 30 lines, COR: 69 lines), the results of non-LPnL and LPnL methods are comparable, see Figure 5.4. Contrarily, on other datasets with high number of lines (177 – 1841 lines), the non-LPnL methods are usually less accurate than the LPnL methods. **Ansar**▶ was run only on the MH dataset containing 30 lines, because it ran out of memory on other datasets. It shows rather poor performance. **Mirzaei**● yields usually the least accurate estimate on datasets with high number of lines (TFH, BB, MC1, MC2, MC3, WDC). On other datasets, it performs comparably to the other methods. A slightly better accuracy is achieved by **RPnL**♦, but it also has trouble on datasets with high number of lines (TFH, BB, STR). The related method **ASPnL**♦ mostly performs better than **RPnL** with an exception of datasets with many lines (BB, STR). Nevertheless, **ASPnL** yields the most accurate pose estimates on MH and COR. This complies with the findings of Xu et al. [33], who state that **ASPnL** is suitable rather for small line sets.

The most accurate results on each dataset are predominantly achieved by the LPnL methods: Most of the top-3 results are achieved by **LPnL_Bar_ENull**★, followed by the proposed method **DLT-Combined-Lines**■, see Table 5.1. **LPnL_Bar_LS**★ and **DLT-Lines**▲ also sometimes achieve top-3 accuracy, although it happens less frequently. **DLT-Plücker-Lines**▼ is the least accurate LPnL method on real-world data, being the only LPnL method which performs slightly below expectations based on synthetic data. Results of other methods are consistent with the results achieved on synthetic lines (Section 5.1).

Bundle Adjustment

As Bundle Adjustment (BA) is commonly used as a final step in 3D reconstruction problems, it is interesting to see how its results are affected by initialization. For this purpose, BA was run on the datasets[1] and initialized using camera poses provided by the tested methods.

A line-based BA engine was preferred. Unfortunately, the only suitable engine was the one of Mičušík and Wildenauer [27], which was a commercial solution unavailable to public. Thus, it was chosen to use a more common point-based BA engine, representing 3D structure only by line segment endpoints. Similarly to [27], an implementation based on the publicly available Ceres Solver [2] was chosen. The implementation uses the Levenberg-Marquardt algorithm [24] to optimize an

[1] The Timberframe House, Building Blocks and Street datasets were excluded from the experiment because the line correspondences were not provided.

Table 5.1: Experiments with real data. Mean orientation error $\Delta\Theta$ [°], position error ΔT [] and reprojection error $\Delta\pi$ [] for each method and image dataset. The top-3 results for each dataset are typeset in bold and color-coded (**best**, **2nd-best** and **3rd-best** result).

Dataset		TFH	BB	STR	MH	COR	MC1	MC2	MC3	ULB	WDC
▶ Ansar	$\Delta\Theta$	-	-	-	4.96	-	-	-	-	-	-
	ΔT	-	-	-	0.38	-	-	-	-	-	-
	$\Delta\pi$	-	-	-	5e-05	-	-	-	-	-	-
● Mirzaei	$\Delta\Theta$	32.24	88.18	0.90	0.46	0.22	4.83	15.47	5.00	2.51	36.52
	ΔT	11.04	168.47	1.92	**0.04**	0.10	1.53	7.37	1.82	1.27	6.44
	$\Delta\pi$	1e+06	2e+06	8e-07	4e-07	1e-06	3e-06	3e-05	1e-02	2e-06	7e+03
◆ RPnL	$\Delta\Theta$	20.46	23.27	4.91	0.61	0.40	1.45	0.43	2.33	3.96	0.50
	ΔT	15.32	53.03	9.73	0.07	0.13	0.43	0.22	1.22	2.08	0.23
	$\Delta\pi$	6e-05	7e-06	9e-05	3e-06	6e-06	2e-06	1e-07	2e-05	6e-06	1e-06
◆ ASPnL	$\Delta\Theta$	7.76	37.82	22.08	**0.25**	**0.10**	0.15	0.20	2.08	4.89	0.51
	ΔT	6.11	76.61	30.47	**0.02**	**0.03**	**0.04**	0.08	0.74	2.22	0.23
	$\Delta\pi$	6e-04	2e+03	3e+02	**5e-08**	**9e-08**	2e-08	1e-08	4e-06	3e-06	1e-06
★ LPnL_Bar_LS	$\Delta\Theta$	1.10	1.98	**0.15**	0.45	0.13	**0.03**	**0.03**	**0.09**	0.49	**0.18**
	ΔT	1.05	**7.23**	**0.27**	**0.04**	0.05	**0.01**	**0.02**	**0.03**	0.22	**0.11**
	$\Delta\pi$	7e-07	1e-06	8e-08	8e-07	1e-06	**2e-09**	**1e-09**	**6e-08**	2e-07	**4e-08**
★ LPnL_Bar_ENull	$\Delta\Theta$	**0.57**	**0.30**	**0.11**	**0.32**	**0.10**	**0.04**	**0.03**	**0.07**	**0.39**	**0.08**
	ΔT	**0.45**	**1.13**	**0.16**	**0.02**	**0.04**	**0.01**	**0.02**	**0.02**	**0.18**	**0.05**
	$\Delta\pi$	**2e-07**	**2e-08**	**3e-08**	**2e-07**	**4e-07**	**8e-10**	**7e-10**	**5e-08**	**1e-07**	**2e-08**
▲ DLT-Lines	$\Delta\Theta$	**0.47**	2.18	**0.11**	0.95	0.12	0.12	0.28	0.23	**0.23**	**0.16**
	ΔT	**0.44**	8.11	**0.18**	0.09	0.05	**0.04**	0.16	0.08	**0.10**	**0.10**
	$\Delta\pi$	**2e-07**	1e-06	**2e-08**	1e-06	2e-06	**6e-09**	4e-08	3e-07	**3e-08**	**6e-08**
▼ **DLT-Plücker-Lines**	$\Delta\Theta$	1.11	**1.04**	0.93	17.58	0.38	0.28	0.22	0.48	0.77	0.34
	ΔT	1.28	11.69	1.78	0.74	0.13	0.40	0.50	0.27	0.47	0.39
	$\Delta\pi$	1e-06	**8e-07**	2e-06	3e-02	3e-06	2e-06	9e-07	2e-05	8e-07	1e-06
■ **DLT-Combined-Lines**	$\Delta\Theta$	**0.39**	**0.40**	0.22	**0.41**	**0.11**	**0.11**	**0.15**	**0.16**	**0.20**	0.23
	ΔT	**0.32**	**1.88**	0.38	**0.04**	**0.04**	**0.04**	**0.07**	**0.05**	**0.08**	0.12
	$\Delta\pi$	**7e-08**	**4e-08**	**6e-08**	**3e-07**	**2e-07**	2e-08	**2e-08**	**2e-07**	**7e-08**	2e-07

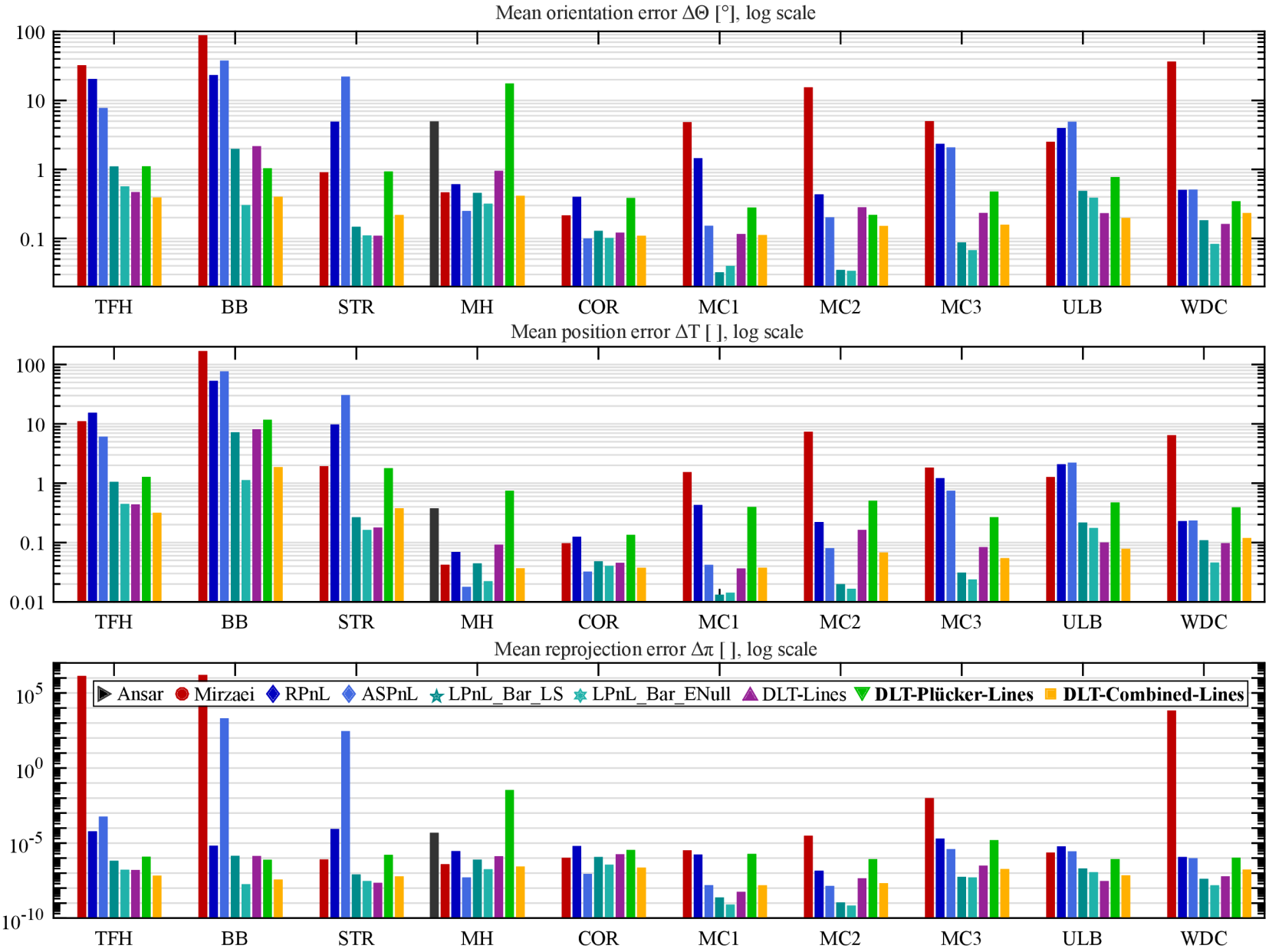

Figure 5.4: Experiments with real data. Mean orientation errors ($\Delta\Theta$,*top*), position errors (ΔT, *middle*) and reprojection errors ($\Delta\pi$, *bottom*) on individual datasets. All vertical axes are logarithmic.

objective function based on reprojection errors – the distances between observed and reprojected point positions. However, the objective function does not utilize the frequently used squared loss, but it is robustified instead by using the Huber's loss function [19], making it less sensitive to outliers. Furthermore, optimization of intrinsic camera parameters was deactivated to allow comparison to pose estimation methods, which do not take the intrinsic parameters into account. As a result, only camera poses and 3D structure were optimized.

BA was initialized using 3D structures provided by the datasets and using camera poses generated by the tested pose estimation methods. Furthermore, BA was also initialized using the ground truth camera poses provided in the datasets. The BA engine then optimized each problem. Because we wanted it to find the optimum as accurately as possible, the stopping criterion (a change in the value of an objective function between consecutive iterations) was set to 10^{-16}. After the optimization, the resulting camera poses and 3D structure were obtained. Because initialization by different camera poses may cause the resulting 3D structures to be slightly different both in shape and position in space, they were aligned by a similarity transformation. The resulting camera poses were transformed using the same transformation. After the alignment, the camera poses were compared.

All optimizations initialized by various pose estimation methods and by the ground truth poses terminated successfully by finding a minimum of the objective function. All minima had the same function value but, within the scope of each single dataset, the minima were not identical: After aligning the optimized 3D structures, the camera poses differed by a magnitude of 0.1 ° and 0.01 length unit. This is approximately the same magnitude of difference as before BA. Since a unique minimum of the objective function was not found, accuracy of the individual pose estimation methods could not be compared in relation to BA, results of which could be considered as a more accurate ground truth.

Nevertheless, it is possible to compare the rate of convergence of BA expressed in terms of runtime. Generaly, BA initialized by camera poses computed by a pose estimation method ran comparably long to the BA initialized by the ground truth camera poses (the runtimes ranged from ≈ 0.6 s for the Model House dataset to ≈ 7.5 s for the Wadham College dataset). An exceptionally long runtime was observed in the case of **RPnL** ♦ and **ASPnL** ♦ in the Merton College III dataset and in the case of **Mirzaei** ● in the Wadham College dataset. This indicated the initialization was worse.

From a practical point of view, the time spent on estimation of camera poses (i. e. initialization of BA) also counts. Therefore, the total time spent on pose estimation *and* on BA is a more appropriate measure. The used datasets contain rather a few camera poses, thus the time of pose estimation is relatively low compared to the time of BA. Even though, the differences in total runtime between individual methods are clearly visible in Figure 5.5. Apart from the exceptionally long runtimes mentioned above, it can be observed that the LPnL-based methods systematically yield lower total runtimes of pose estimation and BA compared to the non-LPnL ones. Differences can be observed even among the LPnL-based methods: The proposed method **DLT-Combined-Lines** ■ provides a speedup over its closest competitor **LPnL_Bar_ENull** ★ ranging from none (for the Wadham College dataset) to 1.27× (for the Merton College I dataset).

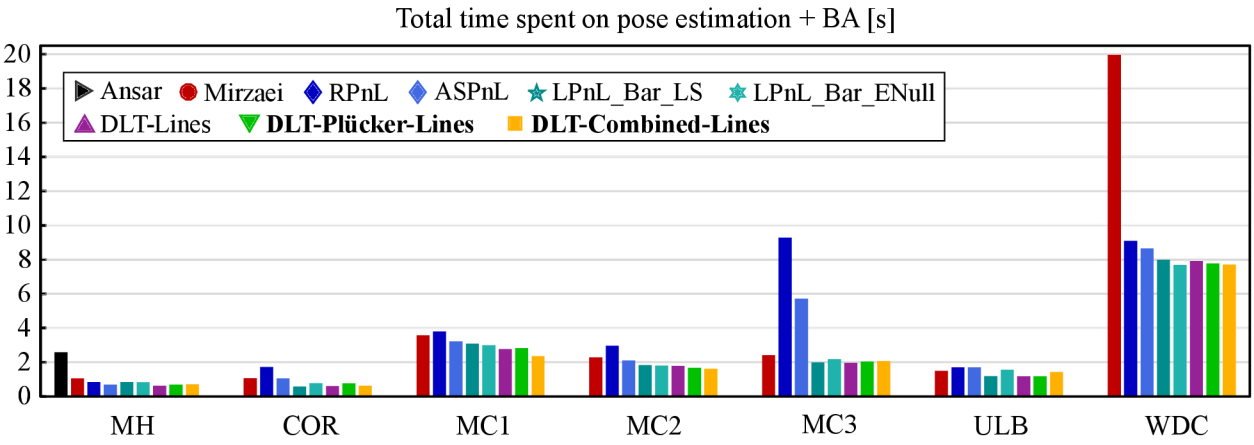

Figure 5.5: Total time spent on pose estimation and Bundle Adjustment in seconds.

5.3 Summary

As it was stated at the beginning of Chapter 4, the thesis aims for better accuracy and robustness than the state-of-the-art in pose estimation from lines by designing a new DLT-based method utilizing line-line correspondences. The method shall keep the common advantage of LPnL methods of being fast.

Two new linear methods for pose estimation were introduced which utilize line-line correspondences. First, The DLT-Plücker-Lines method which competes with the state-of-the-art in some aspects, but it does not exceed it. Second, the DLT-Combined-Lines method which does outperform the state of the art.

1. *Accuracy* – The DLT-Combined-Lines method outperforms the state-of-the-art in estimation of camera position for many lines (Section 5.1: Figure 5.1) and it is comparable to state-of-the-art in orientation estimation. The performance is confirmed also by the results on real data (Section 5.2: Table 5.1), where DLT-Combined-Lines achieves top-3 results on majority of the used datasets.
2. *Robustness to image noise* – The higher accuracy of the estimates of DLT-Combined-Lines is most apparent under strong image noise, which proves its better robustness to this disturbance (Section 5.1: Figure 5.1).
3. *Speed* – DLT-Combined-Lines does not deviate from other LPnL methods as it preserves their common advantage of being fast. A pose of 1000 lines is estimated in about 12 ms (Section 5.1: Figure 5.2).

As it was proven in the experiments listed above, the criteria were fulfilled: Both accuracy and robustness improved while speed was comparable to other DLT-based methods. Thus the dissertation goal was achieved.

Beyond this goal, limits of DLT-Combined-Lines were determined when handling quasi-singular line configurations (near-planar, near-concurrent, and 2 or 3 line directions), it was shown that DLT-Combined-Lines can be used together with AOR to filter out mismatched line correspondences for up to 60 % of mismatches , and it was also shown that DLT-Combined-Lines can decrease the total time spent on pose estimation and the following BA over the state-of-the-art (Section 5.2).

Chapter 6

Conclusions

The goal of the thesis was to improve accuracy and robustness of pose estimation from lines – i. e. of the Perspective-n-Line (PnL) problem – with accent on the formulation based on the Direct Linear Transformation (DLT). The methods based on a linear formulation of PnL (LPnL) are especially suitable for scenarios with large line sets due to their efficiency and accuracy. The goal shall have been achieved by proposing a new linear method utilizing line-line correspondences and keeping the common advantage of LPnL methods of being fast.

Starting from the existing method DLT-Lines which exploits only point-line correspondences, the thesis contributes to the state-of-the-art by proposing two novel methods for pose estimation: *DLT-Plücker-Lines* which exploits line-line correspondences, and *DLT-Combined-Lines* which exploits both point-line and line-line correspondences. Another contribution of the thesis is a unifying framework for all DLT-based methods for pose estimation from lines.

The method DLT-Combined-Lines uses DLT to recover the combined projection matrix. The matrix is a combination of projection matrices used by the DLT-Lines and DLT-Plücker-Lines methods, that work with 3D points and 3D lines, respectively. The proposed method works with both 3D points and lines, which leads to a reduction of the minimum of required lines from 6 (and 9, respectively) to only 5 lines. The method can also easily be extended to use not only 2D lines but also 2D points. The combined projection matrix contains multiple estimates of camera rotation and translation, which can be recovered after enforcing constraints of the matrix. Multiplicity of the estimates leads to better accuracy compared to the other DLT-based methods.

Both novel methods are benchmarked on synthetic data and compared to several state-of-the-art PnL methods. Practical usefulness of the methods is tested on real data comprising buildings and other man-made objects. For larger line sets, DLT-Combined-Lines is comparable to the state-of-the-art method LPnL_Bar_ENull in accuracy of orientation estimation; Yet, it is more accurate in estimation of camera position and it yields smaller reprojection error under strong image noise. On real-world data, DLT-Combined-Lines achieves top-3 results in both orientation estimation, position estimation and reprojection error. When using pose estimation methods to initialize Bundle Adjustment (BA), DLT-Combined-Lines provides a speedup up to 1.27× over LPnL_Bar_ENull in the total runtime of pose estimation and BA. This also indicates the proposed method keeps the common advantage of LPnL methods: very high computational efficiency. The poses of 1000 lines are estimated in 12 ms on a contemporary desktop computer. Altogether, the proposed method DLT-Combined-Lines shows superior accuracy and robustness over its predecessors DLT-Lines and DLT-

Plücker-Lines, which make use either of point-line or line-line correspondences. DLT-Combined-Lines make use of both types of correspondences, yet it is fast. As it was proven in the experiments, the requirements were fulfilled: Both accuracy and robustness improved while speed was comparable to other DLT-based methods. Thus the dissertation goal was achieved.

Future work involves examination of the combined projection matrix to adaptively combine the multiple camera rotation and translation estimates contained in the matrix. Inspired by the work of Xu et al. [33], the proposed methods can also be combined with the effective null space solver. This might further increase accuracy of the methods.

Matlab code of the proposed methods as well as other tested methods and the experiments are made publicly available.[1]

[1] http://www.fit.vutbr.cz/~ipribyl/DLT-based-PnL/

References

[1] Khurrum Aftab, Richard I. Hartley, and Jochen Trumpf. Lq-closest-point to affine subspaces using the generalized weiszfeld algorithm. *International Journal of Computer Vision*, 114(1): 1–15, 2015. ISSN 1573-1405. DOI: 10.1007/s11263-014-0791-8.

[2] Sameer Agarwal, Keir Mierle, et al. Ceres solver. http://ceres-solver.org. Accessed: January 26, 2017.

[3] Adnan Ansar and Kostas Daniilidis. Linear pose estimation from points or lines. *Transactions on Pattern Analysis and Machine Intelligence*, 25(5):578–589, 2003. ISSN 0162-8828. DOI: 10.1109/TPAMI.2003.1195992.

[4] Adrien Bartoli and Peter Sturm. The 3d line motion matrix and alignment of line reconstructions. *International Journal of Computer Vision*, 57:159–178, 2004. ISSN 1573-1405. DOI: 10.1023/B:VISI.0000013092.07433.82.

[5] Martin Bujňák, Zuzana Kukelová, and Tomáš Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Asian Conference on Computer Vision*, pages 11–24, Queenstown, New Zealand, November 2010. Springer. DOI: 10.1007/978-3-642-19315-6_2.

[6] Homer H. Chen. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. In *International Conference on Computer Vision*, pages 374–378, Osaka, Japan, 1990. IEEE. DOI: 10.1109/ICCV.1990.139554.

[7] Stéphane Christy and Radu Horaud. Iterative pose computation from line correspondences. *Computer vision and image understanding*, 73(1):137–144, 1999. ISSN 1077-3142. DOI: 10.1006/cviu.1998.0717.

[8] Ondřej Chum and Jiří Matas. Matching with prosac-progressive sample consensus. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 220–226, San Diego, CA, USA, June 2005. IEEE. DOI: 10.1109/CVPR.2005.221.

[9] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243, Magdeburg, Germany, September 2003. Springer. ISBN 978-3-540-45243-0. DOI: 10.1007/978-3-540-45243-0_31.

[10] Harold S. M. Coxeter. *Projective Geometry*. Springer New York, 2003. ISBN 9780387406237.

[11] Philip David, Daniel DeMenthon, Ramani Duraiswami, and Hanan Samet. Simultaneous pose and correspondence determination using line features. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 424–431, Madison, WI, USA, June 2003. IEEE. DOI: 10.1109/CVPR.2003.1211499.

[12] Michel Dhome, Marc Richetin, Jean-Thierry Lapresté, and Gérard Rives. Determination of the attitude of 3d objects from a single perspective view. *Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, 1989. ISSN 0162-8828. DOI: 10.1109/34.41365.

[13] Olivier D. Faugeras and Bernard Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *International Conference on Computer Vision*, pages 951–956, Cambridge, MA, USA, June 1995. IEEE. DOI: 10.1109/ICCV.1995.466832.

[14] Luis Ferraz, Xavier Binefa, and Francesc Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *Conference on Computer Vision and Pattern Recognition*, pages 501–508, Columbus, OH, USA, June 2014. IEEE. DOI: 10.1109/CVPR.2014.71.

[15] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, 1981. ISSN 0001-0782. DOI: 10.1145/358669.358692.

[16] Richard I. Hartley. In defense of the eight-point algorithm. *Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997. ISSN 0162-8828. DOI: 10.1109/34.601246.

[17] Richard I. Hartley. Minimizing algebraic error in geometric estimation problems. In *International Conference on Computer Vision*, pages 469–476, Bombay, India, January 1998.

[18] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. ISBN 0521540518. DOI: 10.1017/CBO9780511811685.

[19] Peter J. Huber et al. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. DOI: 10.1214/aoms/1177703732.

[20] Rakesh Kumar and Allen R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVGIP: Image understanding*, 60(3):313–342, 1994. ISSN 1049-9660. DOI: 10.1006/ciun.1994.1060.

[21] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009. ISSN 1573-1405. DOI: 10.1007/s11263-008-0152-6.

[22] Yuncai Liu, Thomas S. Huang, and Olivier D. Faugeras. Determination of camera location from 2-d to 3-d line and point correspondences. *Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, 1990. ISSN 0162-8828. DOI: 10.1109/34.41381.

[23] H. Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981. ISSN 0028-0836. DOI: 10.1038/293133a0.

[24] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. ISSN 0368-4245. DOI: 10.1137/0111030.

[25] Jiří Matas. Ransac in 2011. Lecture in the tutorial on Tools and Methods for Image Registration at the Conference on Computer Vision and Pattern Recognition, 2011. URL http://www.imgfsr.com/CVPR2011/Tutorial6/RANSAC_CVPR2011.pdf.

[26] Faraz M. Mirzaei and Stergios I. Roumeliotis. Globally optimal pose estimation from line correspondences. In *International Conference on Robotics and Automation*, pages 5581–5588, Shanghai, China, May 2011. IEEE. DOI: 10.1109/ICRA.2011.5980272.

[27] Branislav Mičušík and Horst Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. *International Journal of Computer Vision*, 2016. ISSN 1573-1405. DOI: 10.1007/s11263-016-0971-9.

[28] Nassir Navab and Olivier D. Faugeras. Monocular pose determination from lines: Critical sets and maximum number of solutions. In *Conference on Computer Vision and Pattern Recognition*, pages 254–260, New York, NY, USA, June 1993. IEEE. DOI: 10.1109/CVPR.1993.340981.

[29] Manuel Silva, Ricardo Ferreira, and José Gaspar. Camera calibration using a color-depth camera: Points and lines based dlt including radial distortion. In *IROS Workshop in Color-Depth Camera Fusion in Robotics*, Vilamoura, Portugal, October 2012.

[30] Philip H. S. Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. ISSN 1077-3142. DOI: 10.1006/cviu.1999.0832.

[31] Roger Y. Tsai and Thomas S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *Transactions on pattern analysis and machine intelligence*, PAMI-6(1):13–27, 1984. ISSN 0162-8828. DOI: 10.1109/TPAMI.1984.4767471.

[32] Tomáš Werner and Andrew Zisserman. New techniques for automated architectural reconstruction from photographs. In *European conference on computer vision*, pages 541–555, Copenhagen, Denmark, May 2002. Springer Berlin Heidelberg. ISBN 978-3-540-47967-3. DOI: 10.1007/3-540-47967-8_36.

[33] Chi Xu, Lilian Zhang, Li Cheng, and Reinhard Koch. Pose estimation from line correspondences: A complete analysis and a series of solutions. *Transactions on Pattern Analysis and Machine Intelligence*, 2016. ISSN 0162-8828. DOI: 10.1109/TPAMI.2016.2582162.

[34] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013. ISSN 1047-3203. DOI: 10.1016/j.jvcir.2013.05.006.

[35] Lilian Zhang, Chi Xu, Kok-Meng Lee, and Reinhard Koch. Robust and efficient pose estimation from line correspondences. In *Asian Conference on Computer Vision*, pages 217–230, Daejeon, Korea, November 2012. Springer. DOI: 10.1007/978-3-642-37431-9_17.

[36] Xiaohu Zhang, Zheng Zhang, You Li, Xianwei Zhu, Qifeng Yu, and Jianliang Ou. Robust camera pose estimation from unknown or known line correspondences. *Applied optics*, 51(7): 936–948, 2012. ISSN 1559-128X. DOI: 10.1364/AO.51.000936.

[37] Yueqiang Zhang, Xin Li, Haibo Liu, and Yang Shang. Probabilistic approach for maximum likelihood estimation of pose using lines. *IET Computer Vision*, 10(6):475–482, 2016. ISSN 1751-9632. DOI: 10.1049/iet-cvi.2015.0099.

List of Publications

My work at FIT BUT led to the following publications, listed in reversed chronological order.

Peer-Reviewed Publications Containing the Core of the Thesis

[I] Bronislav Přibyl, Pavel Zemčík and Martin Čadík: Absolute Pose Estimation from Line Correspondences using Direct Linear Transformation, *Computer Vision and Image Understanding (CVIU)*, 2017: ISSN 1077-3142. DOI: 10.1016/j.cviu.2017.05.002.
Authorship: 70 %.

[II] Bronislav Přibyl, Pavel Zemčík and Martin Čadík: Camera Pose Estimation from Lines using Plücker Coordinates. In *Proceedings of the British Machine Vision Conference (BMVC)*, BMVA, 2015, ISBN 978-1-901725-53-7, p. 12. DOI: 10.5244/C.29.45.
Authorship: 70 %. Google Scholar citations: 7.

Other Peer-Reviewed Publications

[III] Bronislav Přibyl, Alan Chalmers, Pavel Zemčík, Lucy Hooberman and Martin Čadík: Evaluation of Feature Point Detection in High Dynamic Range Imagery. *Journal of Visual Communication and Image Representation (JVCI)*, vol. 38, no. 1, 2016: pp. 141–160, ISSN 1047-3203. DOI: 10.1016/j.jvcir.2016.02.007.
Authorship: 75 %. Google Scholar citations: 2.

[IV] Michal Seeman, Pavel Zemčík and Bronislav Přibyl: Hue Correction in HDR Tonemapping. In *Proceedings of the special session on High Dynamic Range imaging: from theory to deployment in real life applications on the European Signal Processing Conference (EUSIPCO Special Session)*, EURASIP, 2013, ISSN 2219-5491, p. 5, http://ieeexplore.ieee.org/document/6811784/.
Authorship: 20 %.

[V] Bronislav Přibyl, Alan Chalmers and Pavel Zemčík: Feature Point Detection under Extreme Lighting Conditions. In *Proceedings of the Spring Conference on Computer Graphics (SCCG)*, Comenius University in Bratislava, 2012, ISBN 978-80-223-3211-8, pp. 156–163. DOI: 10.1145/2448531.2448550.
Authorship: 85 %. Google Scholar citations: 11.

[VI] Bronislav Přibyl and Pavel Zemčík: Simple Single View Scene Calibration. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS)*, num. 6915 in LCNS, Springer Verlag, 2011, ISBN 978-3-642-23686-0, ISSN 0302-9743, pp. 748–759. DOI: 10.1007/978-3-642-23687-7.
Authorship: 50 %. Google Scholar citations: 1.

[VII] Pavel Zemčík, Bronislav Přibyl, Adam Herout and Michal Seeman: Accelerated Image Resampling for Geometry Correction. *Journal of Real-Time Image Processing (JRTIP)*, vol. 6, no. 3, 2011: p. 9, ISSN 1861-8200. DOI: 10.1007/s11554-011-0213-x.
Authorship: 40 %.

[VIII] Pavel Zemčík, Bronislav Přibyl and Adam Herout: Precise Image Resampling Algorithm. In *Proceedings of the GraVisMa Workshop*, University of West Bohemia in Pilsen, 2009, ISBN 978-80-86943-90-9, pp. 135–138. https://gaupdate.wordpress.com/2010/02/12/proc-gravisma-2009-online/.
Authorship: 40 %.

Other Works

[IX] Pavel Zemčík, Bronislav Přibyl, Martin Žádník and Pavol Korček: Fast and Energy Efficient Image Processing Algorithms using FPGA. In *Proceedings of the Workshop on Computer Vision on Low-Power Reconfigurable Architectures, Conference on Field Programmable Logic and Applications (FPL Workshops)*, IEEE, 2011, ISBN 978-0-7695-4529-5, p. 2. https://www.techfak.uni-bielefeld.de/~fwerner/fpl2011/.
Authorship: 30 %. Google Scholar citations: 1.

[X] Bronislav Přibyl and Pavel Zemčík: Fine Image Resampling Algorithm. In *Proceedings of the Central European Seminar on Computer Graphics for students (CESCG)*, Vienna University of Technology, 2010, ISBN 978-3-9502533-2-0, pp. 175–181. http://www.cescg.org/CESCG-2010/proceedings/CESCG-2010.pdf.
Authorship: 50 %.

[XI] Bronislav Přibyl and Michal Seeman: Precise Image Resampling for Optics Geometry Correction. In *Digital Technologies Workshop*, Faculty of Electrical Engineering of Žilina University, 2007, p. 6.
Authorship: 50 %.

Curriculum Vitae

Personal Data

Full Name:	Bronislav Přibyl
Born:	18th February 1986, Zlín, CZ
Address:	Botanická 823/33, 602 00 Brno, CZ
E-mail:	ipribyl@fit.vutbr.cz
Phone:	+420 777 241 447

Education

2010 Master's degree program, Intelligent systems. [Ing.]
Brno University of Technology, Faculty of Information Technology, CZ.
Master's thesis *Estimation of Object Parameters from Images.* Graduated with honors.

2008 Bachelor's degree program, Information technologies. [Bc.]
Brno University of Technology, Faculty of Information Technology, CZ.
Bachelor's thesis *Image Processing in FPGA*. Graduated with honors.

Study Visits

2012 University of Warwick, Coventry, UK. (7 months)
Research stay for *Detection of Feature Points in High Dynamic Range Images.*

Teaching

- *Computational Geometry.* 5 lectures, FIT BUT, 2013 – 2016.
- *Computer Vision.* 1 lecture, FIT BUT, 2012.
- *Image Processing.* Preparation of a supporting text, FIT BUT, 2012.
- *Computer Graphics Principles.* Laboratory exercies, FIT BUT, 2011.
- Supervision of 7 bachelor's theses, FIT BUT, 2010 – 2012.

Patents, Utility Models

2012 *Method and Device for Digital Image Correction.* Czech patent. Authors: Pavel Zemčík, Adam Herout, Michal Seeman, and Bronislav Přibyl. Registration 2010, approval 2012. Owner: Brno University of Technology.

2010 *Device for Image Resampling.* Utility model. Authors: Pavel Zemčík, Adam Herout, Michal Seeman, and Bronislav Přibyl. Registration 2010, approval 2010. Owner: Brno University of Technology.

Software

2016 *Camera Pose Estimation from Line Correspondences using DLT.* Authors: Bronislav Přibyl, Pavel Zemčík, and Martin Čadík.

2014 *Camera Pose Estimation from Line Correspondences using Plücker Coordinates.* Authors: Bronislav Přibyl, Pavel Zemčík, and Martin Čadík.

AIXM Viewer. Authors: Jozef Mlích, Bronislav Přibyl, Michal Zachariáš, and Pavel Zemčík.

2012 *Evaluation Tool for Feature Point Detection in High Dynamic Range Imagery.* Authors: Bronislav Přibyl and Pavel Zemčík.

2010 *Graph Annotation Tool.* Authors: Jan Koriťák, Aleš Láník, Jiří Zuzaňák, Bronislav Přibyl, and Pavel Zemčík.

Image Processing Server. Authors: Jiří Zuzaňák, Bronislav Přibyl, Aleš Láník, and Pavel Smrž.

Simple Single View Scene Calibration. Authors: Bronislav Přibyl and Pavel Zemčík.

Speech Tagging. Authors: Pavel Smrž, Marek Schmidt, Jiří Zuzaňák, Bronislav Přibyl, Jan Navrátil, Aleš Láník, Lukáš Burget, Tomáš Cipr, Michal Fapšo, Ondřej Glembek, František Grézl, Kamil Chalupníček, Martin Karafiát, Pavel Matějka, Petr Schwarz, and Igor Szőke.

2007 *Separable Resampling.* Authors: Bronislav Přibyl, Michal Seeman, and Pavel Zemčík.

Interests

Outdoor activities and sport (orienteering, mountaineering/climbing, mountain biking, hiking), IT and technology, puzzlehunt games.