Computer Vision and Image Understanding 161 (2017) 130–144

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Absolute pose estimation from line correspondences using direct linear transformation☆

Bronislav Přibyl*, Pavel Zemčík, Martin Čadík

Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Božetěchova 2, 612 66, Brno Czech Republic

ARTICLE INFO

Article history:
Received 25 August 2016
Revised 7 February 2017
Accepted 2 May 2017
Available online 8 May 2017

2010 MSC:
68T45

Keywords:
Camera pose estimation
Perspective-n-Line
Line correspondences
Direct linear transformation

ABSTRACT

This work is concerned with camera pose estimation from correspondences of 3D/2D lines, i. e. with the Perspective-n-Line (PnL) problem. We focus on large line sets, which can be efficiently solved by methods using linear formulation of PnL. We propose a novel method "DLT-Combined-Lines" based on the Direct Linear Transformation (DLT) algorithm, which benefits from a new combination of two existing DLT methods for pose estimation. The method represents 2D structure by lines, and 3D structure by both points and lines. The redundant 3D information reduces the minimum required line correspondences to 5. A cornerstone of the method is a combined projection matrix estimated by the DLT algorithm. It contains multiple estimates of camera rotation and translation, which can be recovered after enforcing constraints of the matrix. Multiplicity of the estimates is exploited to improve the accuracy of the proposed method. For large line sets (10 and more), the method is comparable to the state-of-the-art in accuracy of orientation estimation. It achieves state-of-the-art accuracy in estimation of camera position and it yields the smallest reprojection error under strong image noise. The method achieves top-3 results on real world data. The proposed method is also highly computationally effective, estimating the pose of 1000 lines in 12 ms on a desktop computer.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

Absolute pose estimation is the task of determining the relative position and orientation of a *camera* and an *object* to each other in 3D space. It has many applications in computer vision: 3D reconstruction, robot localization and navigation, visual servoing, and augmented reality are just some of them. The task can be formulated either as object pose estimation (with respect to camera coordinate frame) or as camera pose estimation (with respect to object or world coordinate frame). The latter formulation is used in this paper.

To estimate the camera pose, correspondences between known real world features and their counterparts in the image plane of the camera are needed. The features can be e. g. points, lines, or combinations of both (Kuang and Astrom, 2013). The task has been solved using point correspondences first (Fischler and Bolles, 1981; Lowe, 1987). This is called the *Perspective-n-Point* (PnP) problem and it still enjoys attention of researchers (Ferraz et al., 2014; Lepetit et al., 2009; Valeiras et al., 2016). Camera pose can also be estimated using line correspondences, which is called the *Perspective-n-Line* (PnL) problem. The PnP approach has been studied first, as points are easier to handle mathematically than lines. PnP however is limited only to cases with enough distinctive points, i. e. mainly to well textured scenes. Conversely, the PnL approach is suitable for texture-less scenes, e. g. for man-made and indoor environments. Moreover, line features are more stable than point features and are robust to (partial) occlusions.

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), optionally, iteratively refine the solution, e. g. by minimizing reprojection error. 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. Task (iii), on the other hand, requires solving a problem with high number of lines. In some applications, the correspondences are already known and thus only task (iii) is to be solved.

In recent years, versatile PnL methods have been developed which are suitable for both of these tasks. Remarkable progress has been achieved (Ansar and Daniilidis, 2003; Mirzaei and Roumeliotis, 2011; Zhang et al., 2013), mainly in accuracy of the methods, in their robustness to image noise, and in their effectiveness. These

☆ Matlab code and supplementary material are available at http://www.fit.vutbr.cz/~ipribyl/DLT-based-PnL/.

* Corresponding author.
E-mail addresses: ipribyl@fit.vutbr.cz (B. Přibyl), zemcik@fit.vutbr.cz (P. Zemčík), cadik@fit.vutbr.cz (M. Čadík).

http://dx.doi.org/10.1016/j.cviu.2017.05.002
1077-3142/© 2017 Elsevier Inc. All rights reserved.

methods are outperformed in task (iii) however, by LPnL methods – methods based on a linear formulation of the PnL problem (Přibyl et al., 2015; Xu et al., 2016). LPnL methods are superior in terms of both accuracy and computational speed in camera pose estimation from many (∼ tens to thousands) line correspondences. The oldest LPnL method is that proposed by Hartley and Zisserman (2004, p. 180), followed recently by the method of Přibyl et al. (2015). Even more recently, Xu et al. (2016) introduced a series of LPnL methods generated by the use of Cartesian or barycentric coordinates, and by alternating whether the solution is retrieved in closed form or by optimization. As we show in this paper, space for improving accuracy of the methods still exists.

In this paper, we introduce a novel method based on linear formulation of the PnL problem, which is a combination of the DLT-Lines method of Hartley and Zisserman (2004) and the DLT-Plücker-Lines method of Přibyl et al. (2015). The former represents the 3D structure by 3D points, while the latter represents it by 3D lines parameterized by Plücker coordinates. The proposed method exploits the redundant representation of 3D structure by both 3D points and 3D lines, which leads to the reduction of the minimum required line correspondences to 5. A cornerstone of the method is a combined projection matrix recovered by the DLT algorithm. It contains multiple estimates of camera orientation and translation, enabling a more accurate estimation of the final camera pose. The proposed method achieves state-of-the-art accuracy for large line sets under strong image noise, and it performs comparably to state-of-the-art methods on real world data. The proposed method also keeps the common advantage of LPnL methods – being very fast.

The rest of this paper is organized as follows. We present a review of related work on PnL in Section 2. Then we introduce mathematical notation and Plücker coordinates of 3D lines, and show how points and lines transform and project onto the image plane in Section 3. In Section 4, we explain the application of DLT algorithm to the PnL problem in general, and we describe the existing methods DLT-Lines and DLT-Plücker-Lines. In Section 5, we propose the novel method DLT-Combined-Lines. We evaluate the performance of the proposed method using simulations and real-world experiments in Section 6, and we conclude in Section 7.

2. Related work

The task of camera pose estimation from line correspondences has been receiving attention for more than a quarter of century. Some of the earliest works are those by Dhome et al. (1989) and Liu et al. (1990). They introduce two different ways to deal with the PnL problem – algebraic and iterative approaches – both of which have different properties and thus also different uses. A specific subset of algebraic approaches are the methods based on linear formulation of the PnL problem.

2.1. 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. (1990), the authors attempted to estimate the camera position and orientation separately developing a method called R_then_T. Later on, Kumar and Hanson (1994) 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. (2016) 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. David et al. (2003) and Zhang et al. (2012). 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.

2.2. 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. (1989) and Chen (1990). 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 (Chen, 1990). 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 (2003) 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 – Navab and Faugeras, 1993) 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 (2011) 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 computational 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. (2013). 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. (2016) 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. This algorithm is compared to other state-of-the-art methods in Section 6. 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.

2.3. 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 (Hartley and Zisserman, 2004). 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 (2004, p. 180). Following the terminology of Silva et al. (2012), 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, Přibyl et al. (2015) developed a DLT method, 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. (2016) introduced a new set of methods exploiting the linear formulation of the PnL problem. The authors were inspired by the state-of-the-art PnP solver working on the same principle (Ferraz et al., 2014). Similarly to DLT-Lines, the new methods act on 3D *points* and 2D lines. The methods of Xu et al. (2016) 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[1], or it is estimated from an "effective nullspace" (Lepetit et al., 2009) 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 (Hartley and Zisserman, 2004, 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 (Lepetit et al., 2009) 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.

In this paper, we exploit the common properties of DLT-Lines and DLT-Plücker-Lines methods and we combine them into a new method **DLT-Combined-Lines**. As a result, the minimal number of line correspondences required by the proposed method is reduced to 5, position of the camera is estimated more accurately under strong image noise than by the existing most accurate method (LPnL_Bar_ENull), and the method yields lower reprojection error. Accuracy of orientation estimates is similar to the state-of-the-art method. The proposed method also benefits from the common advantage of all LPnL methods – being very fast.

3. Transformations of points and lines

In this section, we introduce notation, define coordinate systems, and also define parameterization of 3D lines using Plücker coordinates. Then, we review how points and lines are transformed in Euclidean space, and how they project onto the image plane using central projection.

3.1. Notation and coordinate systems

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 ($\mathtt{t}$, $\mathtt{D}$), the identity matrix is denoted by I. 2D entities are denoted by lower case letters (x, $\mathbf{l}$, $\mathtt{t}$), 3D entities by upper case letters (X, $\mathbf{L}$, $\mathtt{D}$). No formal distinction between coordinate vectors and physical entities is made. Equality of up to a non-zero scale factor is denoted by $\approx$, transposition by $^\top$, Euclidean norm of a vector by $||.||$, Kronecker product by $\otimes$, vectorization of a matrix in column-major order by "vec(.)", and the skew symmetric 3×3 matrix associated with the cross product by $[.]_\times$, i. e. $[\mathbf{a}]_\times\mathbf{b}=\mathbf{a}\times\mathbf{b}$. Transformation matrices acting on points and lines are distinguished by a dot and a bar, respectively ($\dot{\mathtt{D}}$, $\bar{\mathtt{D}}$).

Let us now define the coordinate systems: a world coordinate system and a camera coordinate system. Both systems are right-handed. The camera X-axis goes right, the Y-axis goes up and the Z-axis goes behind the camera, so that the points placed in front of the camera have negative Z coordinates in the camera coordinate system. 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. Camera pose is thus parameterized by T_1, T_2, T_3, A, B, Γ.

In the following sections, a pinhole camera with known intrinsic parameters is assumed.

3.2. 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 $\dot{\mathtt{D}}\mathbf{X}$ in the camera coor-

[1] We use the term "homogeneous linear least squares" to denote solving of a homogeneous linear system $\mathsf{M}\mathbf{p}=\mathbf{0}$ for $\mathbf{p}$ which is done by minimization of $||\mathsf{M}\mathbf{p}||$ subject to $||\mathbf{p}||=1$. The correct notation, however somewhat confusing, would be a "low-rank approximation" (of M).

dinate system using a 4×4 point displacement matrix $\dot{\mathrm{D}}$, where

$$\dot{\mathrm{D}} \approx \begin{bmatrix} \mathrm{R} & -\mathrm{R}\mathbf{T} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}. \tag{1}$$

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 $(\mathrm{I}\ \mathbf{0})$. Compositing the two transformations yields the 3×4 point projection matrix

$$\dot{\mathrm{P}} \approx \begin{bmatrix} \mathrm{R} & -\mathrm{R}\mathbf{T} \end{bmatrix}. \tag{2}$$

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

$$\mathbf{x} \approx \dot{\mathrm{P}}\mathbf{X}, \tag{3}$$

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

3.3. Plücker coordinates of 3D lines

3D lines can be represented using several parameterizations in the projective space (Bartoli and Sturm, 2005). Parameterization using Plücker coordinates is complete (i. e. every 3D line can be represented) but not minimal (a 3D line has 4 degrees of freedom but Plücker coordinate is a homogeneous 6-vector). The benefit of using Plücker coordinates is in convenient linear projection of 3D lines onto the image plane.

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 can be represented using Plücker coordinates as 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

$$\begin{aligned} \mathbf{U}^\top &= (L_1\ L_2\ L_3) = (X_1\ X_2\ X_3) \times (Y_1\ Y_2\ Y_3), \\ \mathbf{V}^\top &= (L_4\ L_5\ L_6) = X_4(Y_1\ Y_2\ Y_3) - Y_4(X_1\ X_2\ X_3). \end{aligned} \tag{4}$$

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 the 4 degrees of freedom of a 3D line and its parameterization by a homogeneous 6-vector. More on Plücker coordinates can be found in Hartley and Zisserman (2004).

3.4. 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 $\bar{\mathrm{D}}$ (Bartoli and Sturm, 2004),[2] where

$$\bar{\mathrm{D}} \approx \begin{bmatrix} \mathrm{R} & \mathrm{R}[-\mathbf{T}]_\times \\ 0_{3\times3} & \mathrm{R} \end{bmatrix}. \tag{5}$$

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 Fig. 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 $\bar{\mathrm{D}}$ is needed, yielding the 3×6 line projection matrix (Faugeras and Mourrain, 1995)

$$\bar{\mathrm{P}} \approx \begin{bmatrix} \mathrm{R} & \mathrm{R}[-\mathbf{T}]_\times \end{bmatrix}. \tag{6}$$

[2] Please note that our line displacement matrix differs slightly from the matrix of Bartoli and Sturm (2004, Eq. (6)), namely in the upper right term: We have $\mathrm{R}[-\mathbf{T}]_\times$ instead of $[\mathbf{T}]_\times\mathrm{R}$ due to our different coordinate system.

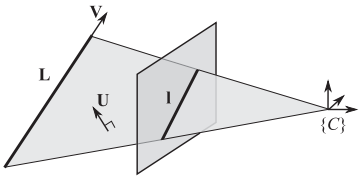

Fig. 1. 3D line projection. The 3D line $\mathbf{L}$ is parameterized by its direction vector $\mathbf{V}$ and a normal $\mathbf{U}$ of its interpretation plane, which passes through the origin of the camera coordinate system $\{C\}$. Since the projected 2D line $\mathbf{l}$ lies at the intersection 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 $\bar{\mathrm{P}}$ as

$$\mathbf{l} \approx \bar{\mathrm{P}}\mathbf{L}, \tag{7}$$

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

4. Pose estimation using DLT

We will first describe DLT methods in general in Section 4.1 and show the common steps. Then, we will describe the DLT-Lines method in Section 4.2 and the DLT-Plücker-Lines method in Section 4.3. Finally, we will briefly describe an algebraic scheme to deal with outlying line correspondences in Section 4.4.

4.1. General structure of DLT

Let us assume that we have (i) a calibrated pinhole camera and (ii) correspondences between 3D lines (or 3D points lying on those lines) and images of the lines obtained by the camera. Given these requirements, it is possible to estimate the camera pose using a DLT method. The methods have the following steps in common:

1. Input data are 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 also includes constraint enforcement in the case of noisy data, since the constraints are not taken into account during the least squares estimation.

4.1.1. 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 (Hartley, 1997). Various transformations can be used, but the optimal ones are unknown. In practice, however, the goal is to reduce large absolute values of point/line coordinates. This is usually achieved by centering the data around the origin and by scaling them so that an average coordinate has the absolute value of 1.

Specific prenormalizing transformations are proposed for each method in the following sections.

4.1.2. 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. This might be the projection of homogeneous 3D points $\mathbf{x} \approx \dot{\mathrm{P}}\mathbf{X}$ in Eq. (3), or the projection of Plücker lines $\mathbf{l} \approx \bar{\mathrm{P}}\mathbf{L}$ in Eq. (7), or any other linear system, or some combination of these. 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 so that only a zero vector remains at the right hand side (see Appendix A for details). The transformed system can be written in the form

$$\mathsf{M}\mathbf{p} = \mathbf{0} \ , \tag{8}$$

where 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. (8) is then solved for the d-vector $\mathbf{p} = \mathrm{vec}(\mathsf{P})$.

Eq. (8), however, holds only in the noise-free case. If a noise is present in the measurements, an inconsistent system is obtained:

$$\mathsf{M}\mathbf{p}' = \boldsymbol{\epsilon} \ . \tag{9}$$

Only an approximate solution $\mathbf{p}'$ may be found through minimization of a $2n$-vector of measurement residuals $\boldsymbol{\epsilon}$ in the least squares sense, subject to $||\mathbf{p}'|| = 1$.

Once the system of linear equations given by (9) is solved, e. g. by Singular Value Decomposition (SVD) of M, the estimate P' of the projection matrix P can be recovered from the d-vector $\mathbf{p}'$.

4.1.3. Extraction of pose parameters

The estimate P' of a projection matrix P obtained as a solution of (9) does not satisfy the constraints imposed on P. In fact, P has only 6 degrees of freedom – the 6 camera pose parameters T_1, T_2, T_3, A, B, Γ. P has, however, more entries: The 3×4 point projection matrix $\dot{\mathsf{P}}$ has 12 entries and the 3×6 line projection matrix $\bar{\mathsf{P}}$ has 18 entries. This means that the projection matrices have 6 and 12 independent linear constraints, respectively, see Eqs. (2) and (6). 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 P' has to be corrected first, since $\mathbf{p}'$ is usually of unit length as a minimizer of $\boldsymbol{\epsilon}$ in Eq. (9). The correct scale of P' can only be determined from the part which does not contain the translation $\mathbf{T}$. In both cases of $\dot{\mathsf{P}}$ and $\bar{\mathsf{P}}$, it is the left 3×3 submatrix P'_1 – an estimate of a rotation matrix R. We recommend a method of scale correction based on the fact that all three singular values of a proper rotation matrix should be 1, see Algorithm 1. Alternatively, the scale can also be corrected so that $\det(s\mathsf{P}'_1) = 1$, but Algorithm 1 proved more robust in practice.

Algorithm 1 Scale correction of a projection matrix.

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

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: $s\mathsf{P}'$.

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

4.2. DLT-Lines

This is the method introduced by Hartley and Zisserman (2004, p. 180). It 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: $\mathbf{l}^\top\mathbf{x} = 0$. Putting this together yields the constraint equation

$$\mathbf{l}^\top\dot{\mathsf{P}}\mathbf{X} = 0 \ . \tag{10}$$

The pose parameters are encoded in the 3×4 point projection matrix $\dot{\mathsf{P}}$, see Eq. (2). 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.

4.2.1. Prenormalization

The known quantities of Eq. (10), 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, Hartley (1998) 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 is $\sqrt{3}$ and $\sqrt{2}$, respectively. By exploiting the principle of duality (Coxeter, 2003), we suggest treating coordinates of 2D lines as homogeneous coordinates of 2D points, and then following Hartley in the prenormalization procedure – i. e. to apply translation and anisotropic scaling.

4.2.2. 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 $\mathbf{X}_i \leftrightarrow \mathbf{l}_i$, $(i = 1 \dots n, \ n \geq 11)$. $\dot{\mathsf{M}}$ is constructed as

$$\dot{\mathsf{M}}_{(i,\ :)} = \mathbf{X}_i^\top \otimes \mathbf{l}_i^\top \ , \tag{11}$$

where $\dot{\mathsf{M}}(i,\ :)$ denotes the i-th row of $\dot{\mathsf{M}}$ in Matlab notation. See Appendix A.3 for a derivation of Eq. (11). The 3D points $\mathbf{X}_i$ must be located on at least 6 different lines.

4.2.3. 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). It is recovered as $\mathbf{T} = s\mathsf{R}^\top\dot{\mathbf{P}}'_4$, completing the extraction of pose parameters.

4.3. DLT-Plücker-Lines

This is the method introduced by Přibyl et al. (2015). It exploits the linear projection of 3D lines parameterized using Plücker coordinates onto the image plane, as described in Section 3.3. The constraint equation defines the formation of 2D lines $\mathbf{l}$ as projections of 3D lines $\mathbf{L}$, as defined in Eq. (7):

$$\mathbf{l} \approx \bar{\mathsf{P}}\mathbf{L} \ . \tag{12}$$

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

4.3.1. Prenormalization

The known quantities of Eq. (12), i. e. the Plücker coordinates of 3D lines, and the coordinates of 2D lines, need to be prenormalized. Since the homogeneous Plücker coordinates of a 3D line $\mathbf{L}$ cannot be treated as homogeneous coordinates of a 5D point (because of the bilinear constraint, see Section 3.3), we suggest the following prenormalization: Translation and scaling, which can be applied through the line similarity matrix (Bartoli and Sturm, 2004), affects only the $\mathbf{U}$ part of $\mathbf{L}$. Therefore, the $\mathbf{V}$ parts are adjusted first by multiplying each $\mathbf{L}$ by a non-zero 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 (Aftab et al., 2015). Finally, anisotropic scaling is applied so that the average magnitude of $\mathbf{U}$ matches the average magnitude of $\mathbf{V}$.

Prenormalization of 2D lines can be carried out in the same way as in the case of the DLT-Lines method, see Section 4.2.

4.3.2. Linear estimation of the line projection matrix

The line projection matrix $\bar{\mathsf{P}}$ and its estimate $\bar{\mathsf{P}}'$ are $3\times\ 6$, so the corresponding measurement matrix $\bar{\mathsf{M}}$ has 18 columns. The number of its rows depends on the number m of line-line correspondences $\mathbf{L}_j \leftrightarrow \mathbf{l}_j$, $(j = 1 \dots m,\ m \geq 9)$. By exploiting Eq. (12), each correspondence generates three rows of $\bar{\mathsf{M}}$ (Matlab notation is used to index the matrix elements):

$$\bar{\mathsf{M}}_{(3j-2:3j,\ :)} = \mathbf{L}_j^\top \ \otimes\ [\mathbf{l}_j]_\times \ . \tag{13}$$

The line measurement matrix $\bar{\mathsf{M}}$ is thus $3m\times\ 18$.[3] See Appendix A.1 for a derivation of Eq. (13).

4.3.3. 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. $\mathsf{R}[-\mathbf{T}]_\times$, see Eq. (6)). Since this is the structure of the essential matrix (Longuet-Higgins, 1981), we propose the algorithm of Tsai and Huang (1984) to decompose it, as outlined in Algorithm 3. This completes the extraction of pose parameters.

The variable $q = (\Sigma_{1,1} + \Sigma_{2,2})/2$ in Algorithm 3 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 $\pm\ 1$ term in Step 4 of Algorithm 3 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 $[\mathbf{T}]_\times = -\mathsf{R}^\top s\bar{\mathsf{P}}'_2$. However, our experiments showed that the alternative ways are less robust when dealing with image noise. Therefore, we have chosen the solution described in Algorithm 3.

4.4. Algebraic outlier rejection

In practice, mismatches of lines (i. e. outlying correspondences) often occur, which degrades the performance of camera pose estimation. The RANSAC algorithm is commonly used to identify and remove outliers; however, as the LPnL methods work with 5 or more line correspondences, they cannot compete with the minimal (P3L) methods when plugged into a RANSAC-like framework due to the increased number of iterations required.

[3] Note that only two of the three rows of $\bar{\mathsf{M}}$ defined by Eq. (13) are needed for each line-line correspondence, because they are linearly dependent. $\bar{\mathsf{M}}$ would be only $2m\times\ 18$ in this case.

Algorithm 3 Extraction of pose parameters from the estimate $\bar{\mathsf{P}}'$ of a line projection matrix (inspired by Tsai and Huang, 1984).

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: $\mathsf{Z} \leftarrow \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, $\mathsf{W} \leftarrow \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$,
$q \leftarrow (\Sigma_{1,1} + \Sigma_{2,2})/2$
4: Compute 2 candidate solutions (A, B):
$\mathsf{R}_\mathrm{A} \leftarrow \mathsf{U}\mathsf{W}\ \mathrm{diag}(1\ 1\ \pm1)\mathsf{V}^\top$, $[\mathbf{T}]_{\times\mathrm{A}} \leftarrow q\mathsf{V}\mathsf{Z}\ \mathsf{V}^\top$
$\mathsf{R}_\mathrm{B} \leftarrow \mathsf{U}\mathsf{W}^\top\mathrm{diag}(1\ 1\ \pm1)\mathsf{V}^\top$, $[\mathbf{T}]_{\times\mathrm{B}} \leftarrow q\mathsf{V}\mathsf{Z}^\top\mathsf{V}^\top$
5: Accept the physically plausible solution, so that the scene lies in front of the camera.
$\mathsf{R} \leftarrow \mathsf{R}_\mathrm{A}$, $\mathbf{T} \leftarrow \mathbf{T}_\mathrm{A}$ or
$\mathsf{R} \leftarrow \mathsf{R}_\mathrm{B}$, $\mathbf{T} \leftarrow \mathbf{T}_\mathrm{B}$.
Output: R, $\mathbf{T}$.

For this reason, an alternative scheme called Algebraic Outlier Rejection (AOR, Ferraz et al., 2014) may be used instead. It is an iterative approach integrated directly into the pose estimation procedure (specifically, into solving Eq. (9) in Section 4.1 in form of iteratively reweighted least squares). Incorrect correspondences are identified as outlying based on the residual ϵ_i of the least squares solution in Eq. (9). Correspondences with residuals above a predefined threshold ϵ_max are assigned zero weights, which effectively removes them from processing in the next iteration, and the solution is recomputed. This is repeated until the error of the solution stops decreasing.

The strategy for choosing ϵ_max may be arbitrary, but our experiments showed that the strategy $\epsilon_\mathrm{max} = \mathrm{Q}_j(\epsilon_1, \dots, \epsilon_n)$ has a good tradeoff between robustness and the number of iterations. $\mathrm{Q}_j(\cdot)$ denotes the jth quantile, where j decreases following the sequence $(0.9,\ 0.8,\ \dots\ ,\ 0.3)$ and then it remains constant 0.25 until error of the solution stops decreasing. This strategy usually leads to approximately 10 iterations.

Remark 1. It is important *not* to prenormalize the data in this case because it will impede the identification of outliers. Prenormalization of inliers should be done just before the last iteration.

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

5. DLT-Combined-Lines

In this section, we introduce the novel method DLT-Combined-Lines. It is a combination of DLT-Lines and DLT-Plücker-Lines, exploiting the redundant representation of 3D structure in the form of both 3D points and 3D lines. The 2D structure is represented by 2D lines. The outcome is a higher accuracy of the camera pose estimates, smaller reprojection error, and lower number of lines required.

The central idea is to merge two systems of linear equations, which share some unknowns, into one system. The unknowns are entries of the point projection matrix $\dot{\mathsf{P}}$ and the line projection matrix $\bar{\mathsf{P}}$. The two systems defined by Eqs. (10) and (12) 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 $\bar{\mathsf{P}}$:

$$\left.\begin{array}{l}\dot{\mathsf{P}} \approx [\ \mathsf{R} \quad -\mathsf{R}\mathbf{T}\ ] \\ \bar{\mathsf{P}} \approx [\ \mathsf{R} \quad \mathsf{R}[-\mathbf{T}]_\times\ ]\end{array}\right\} \quad \ddot{\mathsf{P}} \approx \begin{bmatrix}\mathsf{R} & -\mathsf{R}\mathbf{T} & \mathsf{R}[-\mathbf{T}]_\times\end{bmatrix} \tag{14}$$

We call the matrix a "combined projection matrix", because it allows us to write projection equations for point-line, line-line, and even point-point correspondences, as follows:

$$\mathbf{l}^\top \ddot{\mathsf{P}} \begin{pmatrix}\mathbf{X}^\top & 0 & 0 & 0\end{pmatrix}^\top = 0, \tag{15}$$

$$\mathbf{l} \approx \ddot{\mathsf{P}} \begin{pmatrix}\mathbf{U}^\top & 0 & \mathbf{V}^\top\end{pmatrix}^\top, \tag{16}$$

$$\mathbf{x} \approx \ddot{\mathsf{P}} \begin{pmatrix}\mathbf{X}^\top & 0 & 0 & 0\end{pmatrix}^\top. \tag{17}$$

These equations can then be used to estimate $\ddot{\mathsf{P}}$ linearly from the correspondences, as shown in detail in Section 5.1.

Higher accuracy of pose estimates using the proposed method stems from the fact that the left-most R in $\ddot{\mathsf{P}}$ is determined by twice as many equations, and also from the fact that $\ddot{\mathsf{P}}$ contains multiple estimates of R and $\mathbf{T}$. This is further investigated in Section 5.3.

The other benefit is that the method requires only 5 lines (and 10 points across them) – less than DLT-Plücker-Lines and even less then DLT-Lines. To explain why, we first define the following matrices: 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 $\ddot{\mathsf{P}}_3$.

$$\ddot{\mathsf{P}} = \begin{bmatrix}\mathsf{R} & -\mathsf{R}\mathbf{T} & \mathsf{R}[-\mathbf{T}]_\times\end{bmatrix} = \begin{bmatrix}\ddot{\mathsf{P}}_1 & \ddot{\mathbf{P}}_2 & \ddot{\mathsf{P}}_3\end{bmatrix} \tag{18}$$

$\ddot{\mathsf{P}}$ 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 Eq. (15) and each line-line correspondence generates 2 independent linear Eq. (16). 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 4 1/2) 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 $(n+2m) \geq 20$. In such minimal cases, the points must be distributed equally among the lines, i. e. each point or two must lie on a different line; otherwise, the system of equations would be underdetermined.

We now proceed with the description of the algorithm. Please notice that the prenormalization procedure will be described in Section 5.2, i. e. *after* the definition of a measurement matrix in Section 5.1, because prenormalization is strongly motivated by its structure.

5.1. 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. The number of its rows depends on the number of n point-line correspondences $\mathbf{X}_i \leftrightarrow \mathbf{l}_i$, $(i = 1 \ldots n)$, and on the number of m line-line correspondences $\mathbf{L}_j \leftrightarrow \mathbf{l}_j$, $(j = n+1 \ldots n+m)$. The minimal values of n and m depend on each other and must satisfy the inequality $(n+2m) \geq 20$. Each point-line correspondence (15) leads to one row of $\ddot{\mathsf{M}}$, and each line-line correspondence (16) gives rise to three rows of $\ddot{\mathsf{M}}$ (Matlab notation is used to index the matrix elements):

$$\ddot{\mathsf{M}}_{(i,\ :)} = (\mathbf{X}_i^\top \ \ 0 \ \ 0 \ \ 0) \otimes \mathbf{l}_i^\top, \tag{19}$$

$$\ddot{\mathsf{M}}_{(3j-n-2\ :\ 3j-n,\ :)} = (\mathbf{U}_j^\top \ \ 0 \ \ \mathbf{V}_j^\top) \otimes [\mathbf{l}_j]_\times. \tag{20}$$

The combined measurement matrix $\ddot{\mathsf{M}}$ is thus $(n+3m) \times 21$.[4] See Appendix A for derivations of Eqs. (19) and (20). The combined measurement matrix $\ddot{\mathsf{M}}$ can also be constructed by stacking and aligning $\dot{\mathsf{M}}$ and $\bar{\mathsf{M}}$:

$$\ddot{\mathsf{M}} = \begin{bmatrix}\dot{\mathsf{M}}_{n\times 12} & 0_{n\times 9} \\ \bar{\mathsf{M}}_{(:,1:9)} & 0_{3m\times 3} \quad \bar{\mathsf{M}}_{(:,10:18)}\end{bmatrix} \tag{21}$$

Remark 2. 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 either $\dot{\mathsf{M}}$ 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. (18).)

Remark 3. The method can easily be extended to point-point correspondences (17) 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

$$\ddot{\mathsf{M}}_{(3k-n-m-2\ :\ 3k-n-m,\ :)} = (\mathbf{X}_i^\top \ \ 0 \ \ 0 \ \ 0) \otimes [\mathbf{x}_i]_\times \ , \tag{22}$$

two of which are linearly independent. See Appendix A.2 for a derivation of Eq. (22).

5.2. Prenormalization

Prenormalization of 2D lines is rather complicated. The problem is that a 2D line $\mathbf{l}$ is in the direct form and on the opposite side from the line projection matrix $\bar{\mathsf{P}}$ in Eq. (16), and it is in the transposed form and on the same side as the point projection matrix $\dot{\mathsf{P}}$ in Eq. (15). 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. We thus suggest not prenormalizing 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 4.1: to center the data around the origin by translation, and to scale them so that 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 ($\mathbf{V}$ and X_4) must be adjusted beforehand: Each 3D line and each 3D point is normalized by multiplication by a non-zero scale factor, so that $||\mathbf{V}|| = \sqrt{3}$, and $X_4 = 1$. Note

[4] Note that only two of the three rows of $\ddot{\mathsf{M}}$ defined by Eq. (20) are needed for each line-line correspondence, because they are linearly dependent. Our experiments showed that using all three rows brings no advantage, so we use only two of them in practice. In this case, $\ddot{\mathsf{M}}$ is only $(n+2m) \times 21$.

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.[5] Although the translation is intuitively correct (it results in zero mean 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 minimize the average magnitude of $(X_1\,X_2\,X_3)^\top$ and $\mathbf{U}$. 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, X_4 and $\mathbf{V}$ are equal, i. e. $\overline{|X_1|} + \overline{|L_1|} = \overline{|X_2|} + \overline{|L_2|} = \overline{|X_3|} + \overline{|L_3|} = \overline{|X_4|} + \overline{(|L_4|, |L_5|, |L_6|)}$. This also ensures that the corresponding blocks of the measurement matrix $\ddot{\mathrm{M}}$ 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 $\dot{\mathrm{M}}$ and $\bar{\mathrm{M}}$ should be scaled as stated in Remark 2 above.

The effects of individual stages of prenormalization on accuracy of the proposed method are experimentally evaluated in Section 6.3.

5.3. Extraction of pose parameters

First, the scale of $\ddot{\mathrm{P}}'$ is corrected using Algorithm 1, yielding $s\ddot{\mathrm{P}}'$. The estimates of R and $\mathbf{T}$ are doubled in $s\ddot{\mathrm{P}}'$, which can be exploited to estimate the camera pose more robustly. In the following, we use the definitions of submatrices $\ddot{\mathrm{P}}_1$, $\ddot{\mathbf{P}}_2$, and $\ddot{\mathrm{P}}_3$ from Eq. (18). The first estimate of R is in the direct form in $s\ddot{\mathrm{P}}'_1$, 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 $-\mathrm{R}$. It can be recovered as $\mathbf{T}_2 = -\mathrm{R}_1^\top\, s\ddot{\mathbf{P}}'_2$. The second estimates of R and $\mathbf{T}$ are in the form of an essential matrix in $s\ddot{\mathrm{P}}'_3$, from which they can be extracted using Algorithm 3, 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 $\mathbf{T}_3$. We hypothesize this is because $\ddot{\mathbf{P}}'_2$ has no redundant DoF, contrary to $\ddot{\mathrm{P}}'_3$, 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 direction, a suitable interpolation between them can produce a more accurate position estimate

$$\mathbf{T} = k \cdot \mathbf{T}_2 + (1 - k) \cdot \mathbf{T}_3 \quad . \tag{23}$$

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.[6] Interpolating between R_1 and R_3 yields an orientation R "compatible" with $\mathbf{T}$:

$$\mathrm{R} = \mathrm{R}_1 \cdot \exp(k \cdot \log(\mathrm{R}_1^\top \mathrm{R}_3)) \quad . \tag{24}$$

Here, "exp" and "log" denote matrix exponential and matrix logarithm, respectively. The whole pose extraction procedure is summarized in Algorithm 4.

[5] Another possible translation is to center the 3D lines using the Generalized Weiszfeld algorithm (Aftab et al., 2015). However, our experiments showed that the two possible translations yield nearly identical robustness of the method. We thus suggest to translate the 3D structure to the centroid of points, because its computation is cheaper.

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

Algorithm 4 Extraction of pose parameters from the estimate $\ddot{\mathrm{P}}'$ of a combined projection matrix.

Input: An estimate $\ddot{\mathrm{P}}'$ of a line projection matrix $\ddot{\mathrm{P}}$.
Input: Corrective scale factor s.
1: $\left[\ddot{\mathrm{P}}'_1 \quad \ddot{\mathbf{P}}'_2 \quad \ddot{\mathrm{P}}'_3\right] \leftarrow \ddot{\mathrm{P}}'$ // divide into submatrices
2: Extract R_1 from $\ddot{\mathrm{P}}'_1$ using Algorithm 2.
3: $\mathbf{T}_2 = -\mathrm{R}_1^\top s\ddot{\mathbf{P}}'_2$
4: Extract R_3, $\mathbf{T}_3$ from $\ddot{\mathrm{P}}'_3$ using Algorithm 3.
5: $\mathrm{R} = \mathrm{R}_1 \cdot \exp(k \cdot \log(\mathrm{R}_1^\top \mathrm{R}_3))$
$\mathbf{T} = k \cdot \mathbf{T}_2 + (1 - k) \cdot \mathbf{T}_3$
Output: R, $\mathbf{T}$.

6. Experimental results

The accuracy of pose estimates, the computational efficiency of the methods, and their robustness to image noise and to outliers were measured. Accuracy of pose estimates was expressed in terms of position error and orientation error of the camera, and in terms of reprojection error of the lines, as each error measure suits different applications. For example, robot localization requires a minimal position error, visual servoing requires both position and orientation error to be small, while augmented reality applications prioritize a small reprojection error.

The proposed algorithm was evaluated and compared with the following state-of-the-art methods:

1. **Ansar**, the method by Ansar and Daniilidis (2003), implementation from Xu et al. (2016), results shown in black ▶.
2. **Mirzaei**, the method by Mirzaei and Roumeliotis (2011), results shown in red ●.
3. **RPnL**, the method by Zhang et al. (2013), results shown in dark blue ◆.
4. **ASPnL**, the method by Xu et al. (2016), results shown in light blue ◆.
5. **LPnL_Bar_LS**, the method by Xu et al. (2016), results shown in teal ★.
6. **LPnL_Bar_ENull**, the method by Xu et al. (2016), results shown in blue-green ✸.
7. **DLT-Lines**, the method by Hartley and Zisserman (2004, p. 180), our implementation, results shown in purple ▲.
8. **DLT-Plücker-Lines**, the method by Přibyl et al. (2015), our implementation, results shown in green ▼.
9. **DLT-Combined-Lines**, the proposed method, results shown in orange ■.

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

First, we evaluate accuracy, robustness, and efficiency of the methods using synthetic lines. Then, we evaluate accuracy of pose estimates using rendered images and real data.

6.1. 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 endpoints were used. A virtual pinhole camera with image size of 640 × 480 pixels and focal length of 800 pixels was

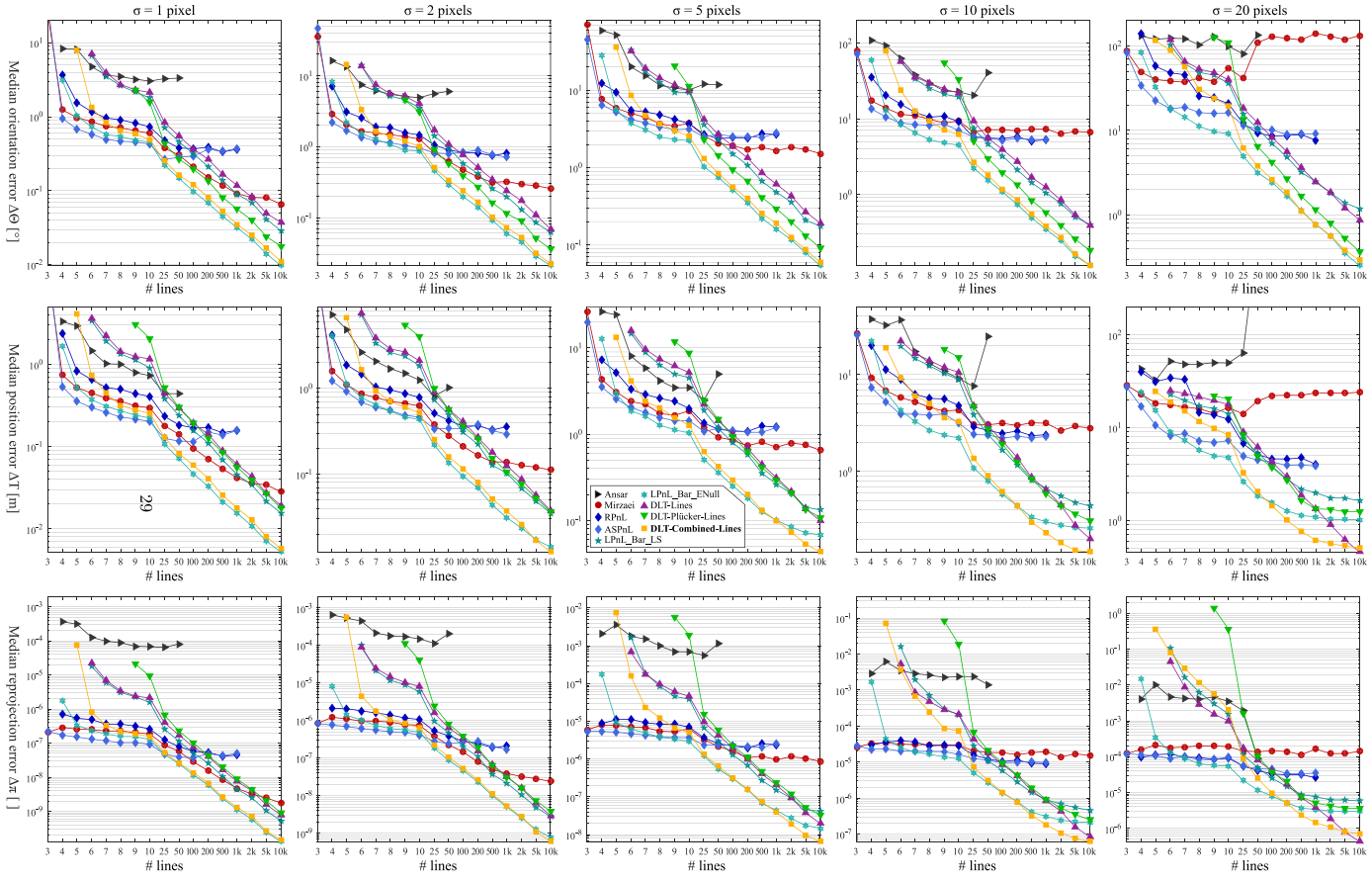

Fig. 2. Experiments with synthetic lines. Median errors in estimated camera pose as a function of the number of lines, computed from 1000 trials. Orientation errors ($\Delta\Theta$, *top row*), position errors (ΔT, *middle row*) and reprojection errors ($\Delta\pi$, *bottom row*) are depicted for various levels of image noise (σ = 1 px – 20 px, *from left to right*). All vertical axes are logarithmic.

placed randomly at 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 m, σ parameters.

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 Mirzaei and Roumeliotis (2011). The *position error* $\Delta T = ||\mathbf{T}' - \mathbf{T}||$ is the distance from the estimated position $\mathbf{T}'$ to the true position $\mathbf{T}$. The *orientation error* $\Delta\Theta$ was calculated as follows. The difference between the true and estimated rotation matrix ($R^\top R'$) is converted to axis-angle representation ($\mathbf{E}$, Θ) and the absolute value of the difference angle $|\Theta|$ is considered as the orientation error. The *reprojection error* $\Delta\pi$ is an integral of squared distance between points on the image line segment and the projection of an infinite 3D line according to Taylor and Kriegman (1995), averaged[7] over all individual lines.

The results showing accuracy of the methods and their robustness to image noise are depicted in Fig. 2. 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 impractical memory requirements). In the following text, the method names are often augmented with their plot marks to ease referencing into result charts.

Our results show high sensitivity to noise of **Ansar** ▶. Even under slight image noise σ = 1 px, 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** ★, the accuracy of which is close to **ASPnL** ◆. It even outperforms **ASPnL** in the case of medium and strong image noise (σ = 5–20 px), see Fig. 2.

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 the errors of the non-LPnL methods (**Mirzaei** ●, **RPnL** ◆, **ASPnL** ◆) do not fall below a certain level. This gets more obvious with increasing levels of noise, see Fig. 2. Each of the LPnL methods also eventually reaches its limit, as it can be seen in the bottom right area of Fig. 2. However, the accuracy limits of non-LPnL methods lag behind the limits of LPnL methods.

DLT-Lines ▲ and **LPnL_Bar_LS** ★ behave nearly identically, the latter being slightly more accurate. The only difference between the two is the latter's use of barycentric coordinates, to which the slight improvement in results can probably be attributed. However, **DLT-Lines** proves to be more accurate in position estimation and reprojection under strong image noise. **DLT-Plücker-Lines** ▼ per-

[7] Please note that Taylor and Kriegman (1995) defined the reprojection error as a *sum* over all individual lines, which makes it dependent on the number of lines.

forms comparably with the two aforementioned methods for 25 or 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 the right part of Fig. 2.

The distributions of errors of the individual methods over all 1000 trials are provided in the supplementary material.

6.1.1. Quasi-singular line configurations

Methods for pose estimation are known to be prone to singular or quasi-singular configurations of 3D primitives. Therefore, the robustness of the methods to quasi-singular line configurations was also evaluated. The setup from Section 6.1 was used with the number of lines fixed to $m = 200$, and standard deviation of image noise fixed to $\sigma = 2$ px.

6.1.2. Limited number of line directions

Lines were generated in three different scenarios: 2 random directions, 3 random directions, and 3 orthogonal directions. The methods of **Ansar** and **Mirzaei** do not work in either case. **RPnL** and **ASPnL** do work, but are susceptible to failure. **DLT-Plücker-Lines** and **DLT-Combined-Lines** do not work in the case of 2 directions, work unreliably in the case of 3 directions, and begin to work flawlessly if the 3 directions are mutually orthogonal. **DLT-Lines, LPnL_Bar_LS** and **LPnL_Bar_ENull** work well in all cases.

6.1.3. Near-planar line distribution

Lines were generated inside a cube spanning 10^3 m, but the cube was progressively flattened until it became a plane. Nearly all methods start to degrade their accuracy when flatness of the "cube" reaches a ratio of 1:10 and perform noticeably worse at the ratio of 1:100. **Mirzaei**, all three **DLT-based** methods and **LPnL_Bar_LS** mostly stop working. **RPnL** and **ASPnL** do work, but often fail. The only fully working method is **LPnL_Bar_ENull**.

6.1.4. Near-concurrent line distribution

Lines were generated randomly, but an increasing number of lines were forced to intersect at a random point inside the cube until all lines were concurrent. **Mirzaei, RPnL, ASPnL** and **LPnL_Bar_LS** degrade their accuracy progressively, although **ASPnL** and **LPnL_Bar_LS** are reasonably accurate even in the fully concurrent case. The **DLT-based** methods work without any degradation as long as 3 or more lines are non-concurrent. **LPnL_Bar_ENull** also works without degradation in the fully concurrent case.

To sum up, behavior of the proposed method, **DLT-Combined-Lines**, is inherited from its two predecessor methods **DLT-Lines** and **DLT-Plücker-Lines**. Their accuracy is degraded:

- If the lines tend to be *planar* (flatness ≈ 1:10 or more).
- If there are fewer than 3 *non-concurrent* lines.
- If the lines are organized into *3 or less directions* (DLT-Lines works in this case, but DLT-Plücker-Lines and DLT-Combined-Lines work only if the 3 directions are orthogonal).

For the sake of brevity, charts depicting errors in the quasi-singular cases are part of the supplementary material.

6.2. Real data

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 Fig. 3, and their characteristics are summarized in Table 1. The Timberframe House dataset contains rendered images, while the rest contain 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 exteriors of various buildings. The Building Blocks dataset is the most challenging because many line segments lie on the common plane of a chessboard.

Table 1
Image sequences used in the experiments with real data.

Sequence	Source	Abbreviation	#images	#lines
Timberframe House	MPI[a]	TFH	72	828
Building Blocks	MPI[a]	BB	66	870
Street	MPI[a]	STR	20	1841
Model House	VGG[b]	MH	10	30
Corridor	VGG[b]	COR	11	69
Merton College I	VGG[b]	MC1	3	295
Merton College II	VGG[b]	MC2	3	302
Merton College III	VGG[b]	MC3	3	177
University Library	VGG[b]	ULB	3	253
Wadham College	VGG[b]	WDC	5	380

[a] MPI dataset http://resources.mpi-inf.mpg.de/LineReconstruction/
[b] VGG dataset http://www.robots.ox.ac.uk/~vgg/data/data-mview.html

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 sequence. The mean errors achieved by all methods on individual datasets are given in Table 2 and visualized in Fig. 4.

On sequences with small number of lines (MH 30, COR 69), the results of non-LPnL and LPnL methods are comparable. Conversely, on sequences with high number of lines (177–1841), the non-LPnL methods are usually less accurate than the LPnL methods. **Ansar** ▶ was run only on the MH sequence containing 30 lines, because other sequences caused it to exceed available memory. It achieves poor performance. **Mirzaei** ● yields usually the least accurate estimate on sequences with high number of lines. On other sequences, it performs comparably to the other methods. A slightly better accuracy is achieved by **RPnL** ◆, but it also has trouble on sequences with high number of lines (TFH, BB, STR). The related method **ASPnL** ◆ mostly performs better than **RPnL** with an exception of sequences with many lines – BB and STR. Nevertheless, **ASPnL** yields the most accurate pose estimates on MH and COR. This agrees with the findings of Xu et al. (2016), who state that **ASPnL** is suitable rather for small line sets.

The most accurate results on each sequence 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 2. **LPnL_Bar_LS** ★ and **DLT-Lines** ▲ also achieve top-3 accuracy, although it happens less frequently. **DLT-Plücker-Lines** ▼ is the least accurate LPnL method on real 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.

6.3. Effect of prenormalization

The prenormalization procedure of the proposed method was described in Section 5.2. It has five stages, which are labeled (i) – (v). To show how the individual stages contribute to the overall accuracy of the method, it was executed on both synthetic lines and real data, while the prenormalization stages were activated one after another. The experimental setup for synthetic lines was the same as in Section 6.1.

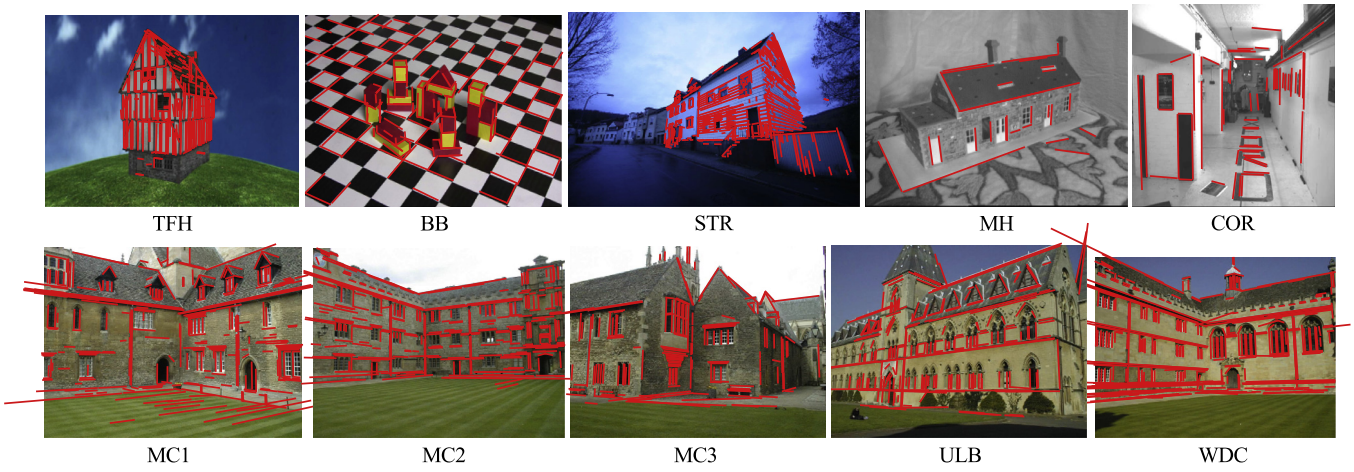

Fig. 3. Example images from used datasets. The images are overlaid with reprojections of 3D line segments using the camera pose estimated by the proposed method DLT-Combined-Lines.

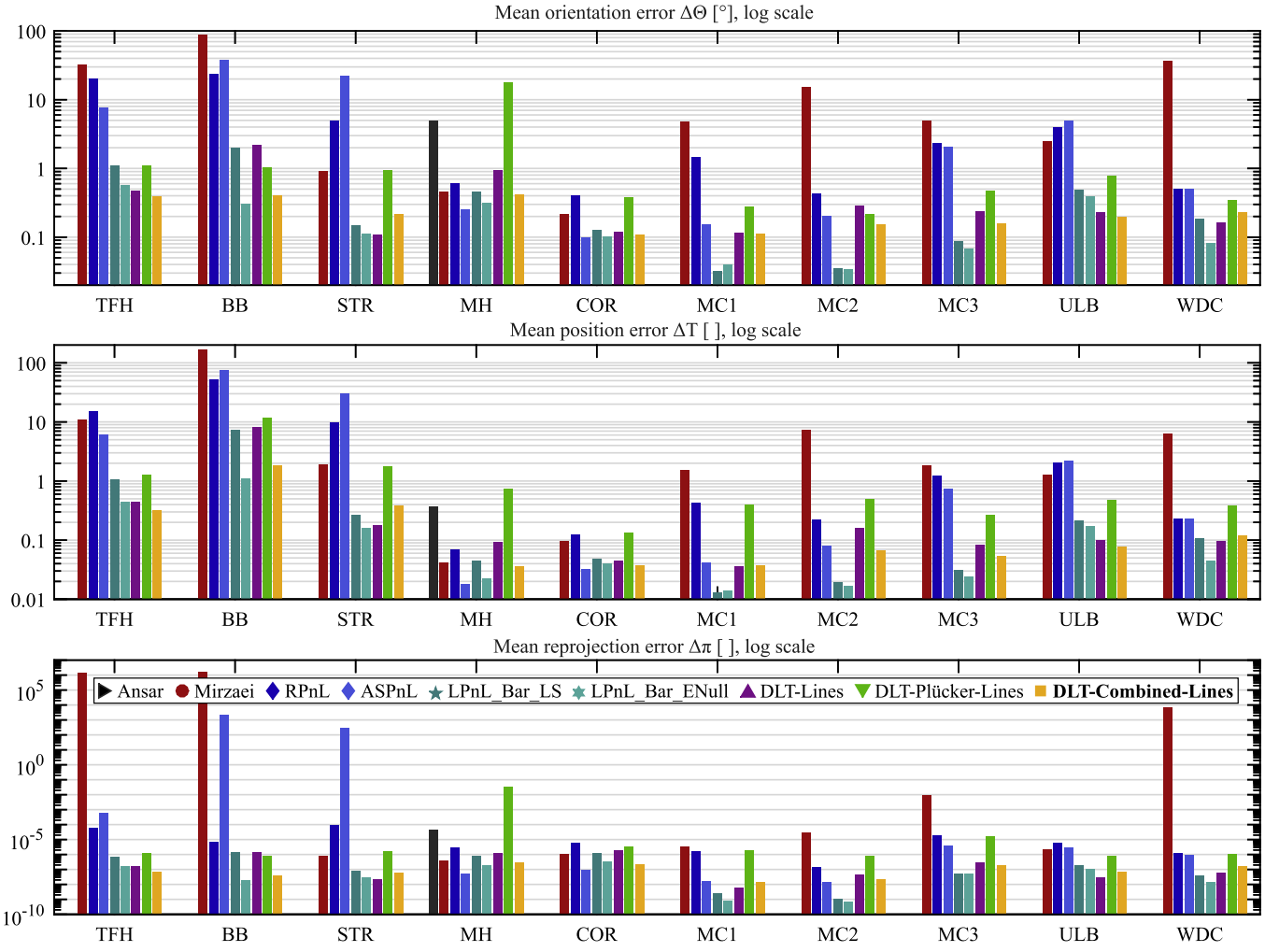

Fig. 4. Experiments with real data. Mean orientation errors ($\Delta\Theta$, *top*), position errors (ΔT, *middle*) and reprojection errors ($\Delta\pi$, *bottom*) on individual image sequences. All vertical axes are logarithmic.

The results are shown in Table 3, which contains the median errors in estimated camera pose for each group of active stages. Each row also shows a relative improvement in accuracy with respect to the previous row.

The measurements show that stages (i) and (ii), i. e. multiplication of each 3D point/line by a constant and translation of the data to be centered around the origin, have the biggest impact on accuracy of the method. Stage (v), i. e. scaling of some parts of the measurement matrix, proves to be important mainly in the case of real data. Stages (iii) and (iv), i. e. the second translation and anisotropic scaling, have a minor positive impact on accuracy.

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

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

Table 3
Effect of prenormalization stages on the accuracy of the proposed method.

Prenormalization stages	Synthetic lines: Median of abs. error $\Delta\Theta$ [°]	ΔT [m]	$\Delta\pi$ [ ]	Synthetic lines: Improvement $\Delta\Theta$	ΔT	$\Delta\pi$	Real data: Median of abs. error $\Delta\Theta$ [°]	ΔT [ ]	$\Delta\pi$ [ ]	Real data: Improvement $\Delta\Theta$	ΔT	$\Delta\pi$
none	0.92	444.48	1.11e−2	–	–	–	1.25	0.21	2.40e−6	–	–	–
(i)	0.93	373.35	1.50e−2	−1%	16%	−35%	0.38	0.14	1.42e−7	69%	34%	94%
(i), (ii)	0.01	0.48	3.17e−6	98%	100%	100%	0.22	0.09	6.25e−8	42%	33%	56%
(i) – (iii)	0.01	0.48	3.15e−6	0%	0%	0%	0.22	0.09	6.26e−8	0%	0%	0%
(i) – (iv)	0.01	0.48	3.08e−6	0%	0%	2%	0.23	0.09	6.33e−8	−1%	0%	−1%
(i) – (v)	0.01	0.64	1.47e−6	2%	−34%	52%	0.21	0.07	6.93e−8	8%	21%	−9%

6.4. Speed

Efficiency of each method was evaluated by measuring runtime on a desktop PC with a quad core Intel i5 3.33 GHz CPU and 10 GB of RAM. The experimental setup was the same as in Section 6.1, varying the number of lines.

As it can be seen in Fig. 5 and Table 4, the only method with $O(m^2)$ computational complexity in the number of lines m is **Ansar** ▶. The space complexity of the implementation used is apparently also quadratic. We were unable to execute it for as few as 100 lines due to lack of computer memory. All other tested methods have $O(m)$ computational complexity. The runtimes however

Table 4
Runtimes in milliseconds for varying number of lines, averaged over 1000 trials.

# lines	10	100	1000
Ansar	4.1	–	–
Mirzaei	77.9	84.2	155.2
RPnL	8.8	41.3	879.5
ASPnL	8.7	29.5	630.2
LPnL_Bar_LS	1.1	1.2	2.3
LPnL_Bar_ENull	5.2	5.3	6.7
DLT-Lines	1.0	1.2	2.7
DLT-Plücker-Lines	3.0	3.6	8.2
DLT-Combined-Lines	3.7	4.6	12.1

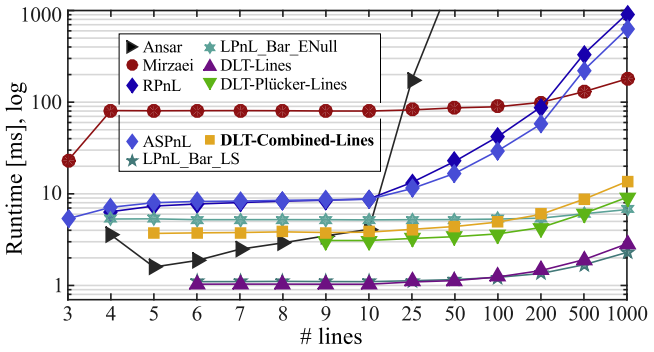

Fig. 5. Runtimes as a function of the number of lines, averaged over 1000 trials. Logarithmic vertical axis.

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. **RPnL** is slightly faster for $m < 10$ lines, while **ASPnL** is faster for greater numbers of lines. Runtimes of both methods rise steeply with increasing number of lines, reaching 630.2 ms on 1000 lines for **ASPnL**. 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 is due to 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, see Table 4. 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 double size must be additionally decomposed compared to the competing methods, see Eq. (21). The computationally demanding part of **LPnL_Bar_ENull** is the effective null space solver carrying out Gauss-Newton optimization.

6.5. Robustness to outliers

As a practical requirement, robustness to outlying correspondences was also tested. The experimental setup was the same as in Section 6.1, using $m = 500$ lines having endpoints perturbed with slight image noise ($\sigma = 2$ pixels). The image lines simulating outlying correspondences were perturbed with an additional extreme noise with $\sigma = 100$ pixels. The fraction of outliers varied from 0% to 80%.

Ansar, Mirzaei, and **RPnL** methods were plugged into a MLESAC framework (Torr and Zisserman, 2000, a generalization of RANSAC which maximizes the likelihood rather than just the number of inliers). Since **Ansar** cannot handle the final pose computation from potentially hundreds of inlying line correspondences, it is computed by **RPnL**. The probability that only inliers will be selected in some iteration was set to 99%, and the number of iterations was limited to 10,000. The inlying correspondences were identified based on the line reprojection error. No heuristic for early hypothesis rejection was utilized, as it can also be incorporated into AOR, e. g. by weighting the line correspondences. **DLT-Lines, DLT-Plücker-Lines**, and **DLT-Combined-Lines** methods were equipped with AOR, which was set up as described in Section 4.4.

The setup presented by Xu et al. (2016) was also tested: **LPnL_Bar_LS** and **LPnL_Bar_ENull** methods with AOR, and a **P3L** solver and **ASPnL** plugged into a RANSAC framework, generating camera pose hypotheses from 3 and 4 lines, respectively. The authors have set the required number of inlying correspondences to 40% of all correspondences, and limit the number of iterations to 80. When this is exceeded, the required number of inliers is decreased by a factor of 0.5, and another 80 iterations are allowed. The inlying correspondences are identified based on thresholding of an algebraic error – the residuals ϵ_i of the least squares solution in Eq. (9), where the measurement matrix $\dot{M}$ is used, defined by Eq. (11).

The tested methods are summarized in the following list (the number at the end of MLESAC/RANSAC denotes the number of lines used to generate hypotheses).

1. **Ansar + MLESAC4 + RPnL**, Ansar plugged into a MLESAC loop, the final solution computed by RPnL. Results shown in black ▶.
2. **Mirzaei + MLESAC3**, results shown in red ●.
3. **RPnL + MLESAC4**, results shown in blue ♦.
4. **P3L + RANSAC3**, the setup by Xu et al. (2016), results shown in sky blue ♦ .
5. **ASPnL + RANSAC4**, the setup by Xu et al. (2016), results shown in light blue ♦.
6. **LPnL_Bar_LS + AOR**, the setup by Xu et al. (2016), results shown in teal ★.
7. **LPnL_Bar_ENull + AOR**, the setup by Xu et al. (2016), results shown in blue-green ✶.
8. **DLT-Lines + AOR**, results shown in purple ▲.
9. **DLT-Plücker-Lines + AOR**, results shown in green ▼.
10. **DLT-Combined-Lines + AOR**, the proposed method with AOR, results shown in orange ■.

The RANSAC-based approaches can theoretically handle any percentage of outliers. This is confirmed by **Mirzaei + MLESAC3** ● and **RPnL + MLESAC4** ♦, as their accuracy does not change with respect to the fraction of outliers. What does change however, is the number of iterations (and thus also the runtime). Even then, the limit of 10,000 iterations was almost never reached. A different situation occurred when testing **Ansar + MLESAC3 + RPnL** ▶, where the iteration limit was sometimes reached even at 20% of outliers. This suggests that **Ansar** is a poor hypothesis generator, and the MLESAC framework needs to iterate more times to get a valid hypothesis.

P3L + RANSAC3 ♦ and **ASPnL + RANSAC4** ♦ have much lower runtimes, which is caused mainly by the setup limiting the number of iterations to a few hundreds. The setup has, on the other hand, a negative effect on the robustness of the method: the break-down point is only 60–70%, as it is apparent in Fig. 6a, b. This issue was not observed by Xu et al. (2016), because they tested the methods only up to 60% of outliers.

LPnL methods with AOR have constant runtimes regardless of the fraction of outliers. The fastest one is **DLT-Lines + AOR** ▲ running in 10 ms on average. The proposed method **DLT-Combined-Lines + AOR** ■ runs in 31 ms on average, and **LPnL_Bar_ENull + AOR** ✶ is the slowest one with 57 ms, see Fig. 6d. The robustness of the LPnL methods differs significantly. **DLT-Plücker-Lines + AOR** ▼ breaks-down at about 40%, but it occasionally generates wrong solutions from 30% up, We call this a "soft" break-down point. **LPnL_Bar_ENull + AOR** ✶ behaves similarly, but it yields smaller pose errors. **DLT-Lines + AOR** ▲,

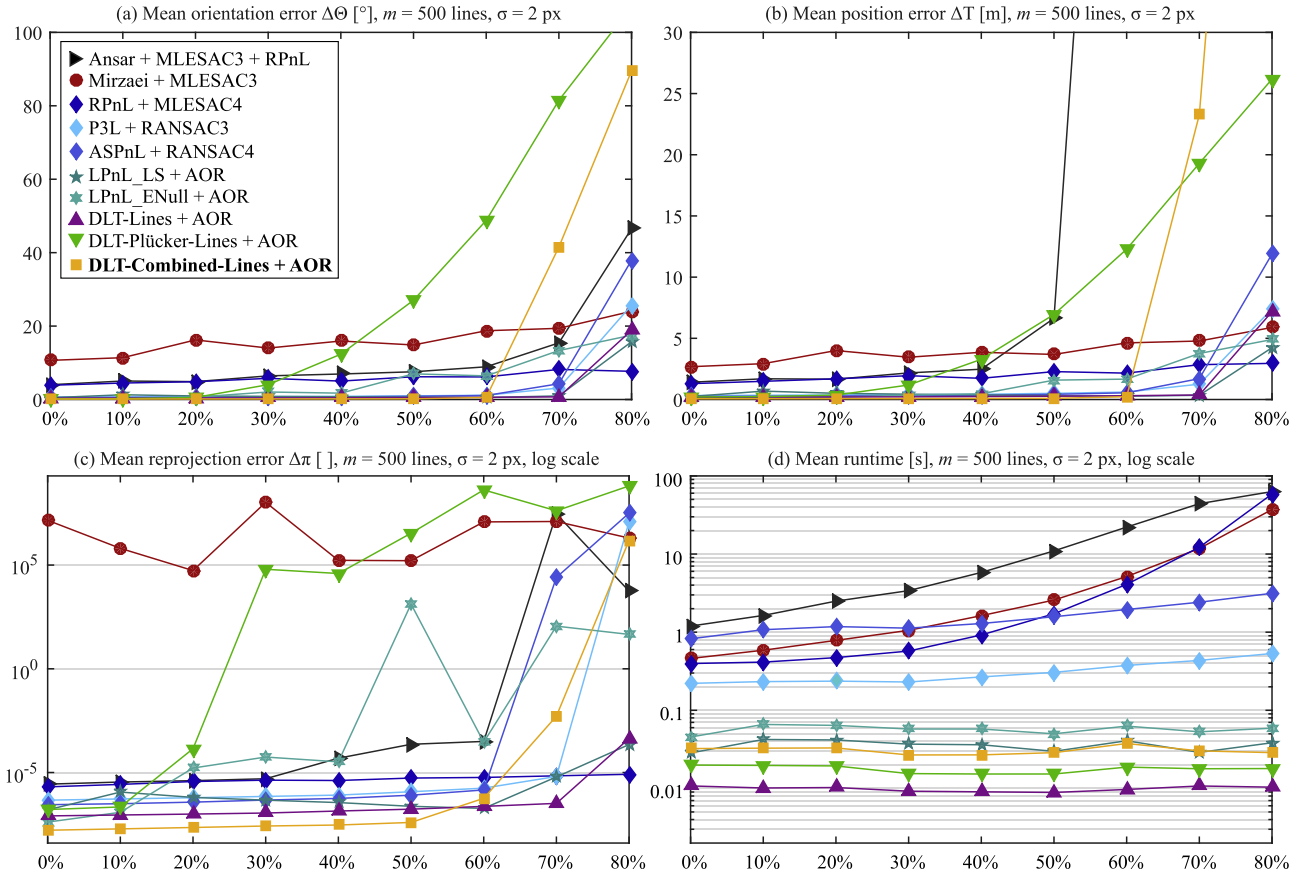

Fig. 6. Experiments with outliers. Mean camera orientation errors ($\Delta\Theta$, *a*), position errors (ΔT, *b*), reprojection errors ($\Delta\pi$, *c*) and runtimes (*d*) depending on the percentage of outliers. Averaged over 1000 trials.

LPnL_Bar_LS + AOR ★, and the proposed method **DLT-Combined-Lines + AOR** ■, on the other hand, have a "hard" break-down point at 70%, 65%, and 60%, respectively. This means they do not yield wrong solutions until they reach the break-down point.

The RANSAC-based approach is irreplaceable in cases with high percentage of outliers. Aside from this, for lower fractions of outliers, the LPnL + AOR alternatives are more accurate and 4 – 31× faster than the RANSAC-based approaches, depending on the chosen LPnL method.

The distributions of errors of the tested methods over all 1000 trials are provided in the supplementary material.

7. Conclusions

A novel algebraic method DLT-Combined-Lines is proposed to estimate camera pose from line correspondences. The method is based on linear formulation of the Perspective-n-Line problem, and it uses Direct Linear Transformation 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, leading to the reduction of the minimum of required lines to 5, and it can be easily extended to use 2D points as well. 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 methods.

For small line sets, the proposed method is not as accurate as the non-LPnL methods, which is a common attribute of all methods using linear formulation. For larger line sets, the method 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, the proposed method achieves top-3 results. It also keeps the common advantage of LPnL methods: very high computational efficiency. To deal with outliers, the proposed method is equipped with the Algebraic Outlier Rejection scheme, being able to handle up to 65% of outliers in a fraction of time required by the RANSAC-based approaches.

It is clear that none of the existing methods is universally best. We thus suggest to use **ASPnL** for small line sets ($m \leq 10$). For bigger line sets ($m > 10$), we suggest to use a LPnL method: either **LPnL_Bar_ENull** (in cases with small noise or quasi-singular line configurations) or **DLT-Combined-Lines** (in cases with many lines and/or strong noise).

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

Matlab code of the proposed method and the supplementary material are publicly available at http://www.fit.vutbr.cz/~ipribyl/DLT-based-PnL/.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in sci-

ence – LQ1602. The authors would like to thank Tomáš Werner for proofreading of the paper.

Appendix A. Derivation of a measurement matrix M from 3D/2D correspondences

Correspondences between 3D entities and their 2D counterparts are defined by equations which, in turn, generate rows of a measurement matrix M. The following derivations are made for a single 3D/2D correspondence. More correspondences lead simply to stacking the rows of M.

A1. Line-line correspondence

We start from Eq. (7) defining the projection of a 3D line $\mathbf{L}$ by a line projection matrix $\bar{\mathrm{P}}$ onto the image line $\mathbf{l}$

$$\mathbf{l} \approx \bar{\mathrm{P}}\mathbf{L} \ . \tag{A.1}$$

We swap its sides and pre-multiply them by $[\mathbf{l}]_\times$

$$[\mathbf{l}]_\times \bar{\mathrm{P}}\mathbf{L} \approx [\mathbf{l}]_\times \mathbf{l} \ . \tag{A.2}$$

The right-hand side is apparently a vector of zeros

$$[\mathbf{l}]_\times \bar{\mathrm{P}}\mathbf{L} = \mathbf{0} \ . \tag{A.3}$$

Using Lemma 4.3.1 of Horn and Johnson (1994), we get

$$\left(\mathbf{L}^\top \otimes [\mathbf{l}]_\times\right) \cdot \mathrm{vec}(\bar{\mathrm{P}}) = \mathbf{0} \ . \tag{A.4}$$

The left-hand side can be divided into the measurement matrix $\mathrm{M} = \mathbf{L}^\top \otimes [\mathbf{l}]_\times$ and the vector of unknowns $\bar{\mathbf{p}} = \mathrm{vec}(\bar{\mathrm{P}})$, finally yielding the homogeneous system

$$\mathrm{M}\bar{\mathbf{p}} = \mathbf{0} \ . \tag{A.5}$$

A2. Point-point correspondence

The derivation is the same as in the case of line-line correspondences, but starting from Eq. (3) defining the projection of a 3D point $\mathbf{X}$ by a point projection matrix $\dot{\mathrm{P}}$ onto the image point $\mathbf{x}$.

$$\mathbf{x} \approx \dot{\mathrm{P}}\mathbf{X} \tag{A.6}$$

$$[\mathbf{x}]_\times \dot{\mathrm{P}}\mathbf{X} \approx [\mathbf{x}]_\times \mathbf{x} \tag{A.7}$$

$$[\mathbf{x}]_\times \dot{\mathrm{P}}\mathbf{X} = \mathbf{0} \tag{A.8}$$

$$\left(\mathbf{X}^\top \otimes [\mathbf{x}]_\times\right) \cdot \mathrm{vec}(\dot{\mathrm{P}}) = \mathbf{0} \tag{A.9}$$

$$\mathrm{M}\dot{\mathbf{p}} = \mathbf{0} \tag{A.10}$$

A3. Point-line correspondence

We start from Eq. (10) relating the projection of a 3D point $\mathbf{X}$ and an image line $\mathbf{l}$

$$\mathbf{l}^\top \dot{\mathrm{P}}\mathbf{X} = 0 \ . \tag{A.11}$$

Since Eq. (A.11) already has the right-hand side equal to 0, we can directly apply Lemma 4.3.1 of Horn and Johnson (1994), and see how the measurement matrix M is generated:

$$\left(\mathbf{X}^\top \otimes \mathbf{l}^\top\right) \cdot \mathrm{vec}(\dot{\mathrm{P}}) = 0 \ , \tag{A.12}$$

$$\mathrm{M}\dot{\mathbf{p}} = 0 \ . \tag{A.13}$$

References

Aftab, K., Hartley, R.I., Trumpf, J., 2015. Lq-closest-point to affine subspaces using the generalized weiszfeld algorithm. Int. J. Comput. Vis. 114 (1), 1–15. doi:10.1007/s11263-014-0791-8.

Ansar, A., Daniilidis, K., 2003. Linear pose estimation from points or lines. IEEE Trans. Pattern Anal. Mach. Intell. 25 (5), 578–589. doi:10.1109/TPAMI.2003.1195992.

Bartoli, A., Sturm, P., 2004. The 3d line motion matrix and alignment of line reconstructions. Int. J. Comput. Vis. 57, 159–178. doi:10.1023/B:VISI.0000013092.07433.82.

Bartoli, A., Sturm, P., 2005. Structure-from-motion using lines: representation, triangulation, and bundle adjustment. Comput. Vision Image Understanding 100 (3), 416–441. doi:10.1016/j.cviu.2005.06.001.

Chen, H.H., 1990. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. In: IEEE International Conference on Computer Vision 1990. IEEE, pp. 374–378. doi:10.1109/ICCV.1990.139554.

Coxeter, H.S.M., 2003. Projective Geometry. Springer New York.

David, P., DeMenthon, D., Duraiswami, R., Samet, H., 2003. Simultaneous pose and correspondence determination using line features. In: IEEE Conference on Computer Vision and Pattern Recognition 2003, 2. IEEE, pp. 424–431. doi:10.1109/CVPR.2003.1211499.

Dhome, M., Richetin, M., Lapreste, J.-T., Rives, G., 1989. Determination of the attitude of 3d objects from a single perspective view. IEEE Trans. Pattern Anal. Mach. Intell. 11 (12), 1265–1278. doi:10.1109/34.41365.

Faugeras, O., Mourrain, B., 1995. On the geometry and algebra of the point and line correspondences between n images. In: International Conference on Computer Vision 1995. IEEE, pp. 951–956. doi:10.1109/ICCV.1995.466832.

Ferraz, L., Binefa, X., Moreno-Noguer, F., 2014. Very fast solution to the pnp problem with algebraic outlier rejection. In: IEEE Conference on Computer Vision and Pattern Recognition 2014. IEEE, pp. 501–508. doi:10.1109/CVPR.2014.71.

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24 (6), 381–395. doi:10.1145/358669.358692.

Hartley, R.I., 1997. In defense of the eight-point algorithm. IEEE Trans. Pattern Anal. Mach. Intell. 19 (6), 580–593. doi:10.1109/34.601246.

Hartley, R.I., 1998. Minimizing algebraic error. Philos. Trans. R. Soc. London, Ser. A 356 (1740), 1175–1192. doi:10.1098/rsta.1998.0216.

Hartley, R.I., Zisserman, A., 2004. Multiple View Geometry in Computer Vision, second ed. Cambridge University Press doi:10.1017/CBO9780511811685.

Horn, R., Johnson, C., 1994. Topics in Matrix Analysis. Cambridge University Press doi:10.1017/CBO9780511840371.

Kuang, Y., Astrom, K., 2013. Pose estimation with unknown focal length using points, directions and lines. In: IEEE International Conference on Computer Vision 2013. IEEE, pp. 529–536. doi:10.1109/ICCV.2013.71.

Kumar, R., Hanson, A.R., 1994. Robust methods for estimating pose and a sensitivity analysis. CVGIP 60 (3), 313–342. doi:10.1006/ciun.1994.1060.

Lepetit, V., Moreno-Noguer, F., Fua, P., 2009. Epnp: an accurate o(n) solution to the pnp problem. Int. J. Comput. Vision 81 (2), 155–166. doi:10.1007/s11263-008-0152-6.

Liu, Y., Huang, T.S., Faugeras, O.D., 1990. Determination of camera location from 2-d to 3-d line and point correspondences. IEEE Trans. Pattern Anal. Mach. Intell. 12 (1), 28–37. doi:10.1109/34.41381.

Longuet-Higgins, H.C., 1981. A computer algorithm for reconstructing a scene from two projections. Nature 293, 133–135. doi:10.1038/293133a0.

Lowe, D.G., 1987. Three-dimensional object recognition from single two-dimensional images. Artif. Intell. 31 (3), 355–395. doi:10.1016/0004-3702(87)90070-1.

Mirzaei, F.M., Roumeliotis, S.I., 2011. Globally optimal pose estimation from line correspondences. In: IEEE International Conference on Robotics and Automation 2011. IEEE, pp. 5581–5588. doi:10.1109/ICRA.2011.5980272.

Navab, N., Faugeras, O., 1993. Monocular pose determination from lines: Critical sets and maximum number of solutions. In: IEEE Conference on Computer Vision and Pattern Recognition 1993. IEEE, pp. 254–260. doi:10.1109/CVPR.1993.340981.

Přibyl, B., Zemčík, P., Čadík, M., 2015. Camera pose estimation from lines using plücker coordinates. In: Proceedings of the British Machine Vision Conference (BMVC 2015).The British Machine Vision Association and Society for Pattern Recognition, pp. 1–12. doi:10.5244/C.29.45.

Silva, M., Ferreira, R., Gaspar, J., 2012. Camera calibration using a color-depth camera: Points and lines based dlt including radial distortion. IROS Workshop in Color-Depth Camera Fusion in Robotics.

Taylor, C.J., Kriegman, D., 1995. Structure and motion from line segments in multiple images. IEEE Trans. Pattern Anal. Mach. Intell. 17 (11), 1021–1032. doi:10.1109/34.473228.

Torr, P.H.S., Zisserman, A., 2000. Mlesac: a new robust estimator with application to estimating image geometry. Comput. Vision Image Understanding 78 (1), 138–156. doi:10.1006/cviu.1999.0832.

Tsai, R.Y., Huang, T.S., 1984. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6 (1), 13–27. doi:10.1109/TPAMI.1984.4767471.

Valeiras, D.R., Kime, S., Ieng, S.-H., Benosman, R.B., 2016. An event-based solution to the perspective-n-point problem. Front. Neurosci. 10. doi:10.3389/fnins.2016.00208.

Xu, C., Zhang, L., Cheng, L., Koch, R., 2016. Pose estimation from line correspondences: a complete analysis and a series of solutions. IEEE Trans. Pattern Anal. Mach. Intell. doi:10.1109/TPAMI.2016.2582162.

Zhang, L., Xu, C., Lee, K.-M., Koch, R., 2013. Robust and efficient pose estimation from line correspondences. In: Asian Conference on Computer Vision 2012. Springer, pp. 217–230. doi:10.1007/978-3-642-37431-9_17.

Zhang, X., Zhang, Z., Li, Y., Zhu, X., Yu, Q., Ou, J., 2012. Robust camera pose estimation from unknown or known line correspondences. Appl. Opt. 51 (7), 936–948. doi:10.1364/AO.51.000936.

Zhang, Y., Li, X., Liu, H., Shang, Y., 2016. Probabilistic approach for maximum likelihood estimation of pose using lines. IET Comput. Vision 10 (6), 475–482. doi:10.1049/iet-cvi.2015.0099.