Camera Pose Estimation from Lines using Plücker Coordinates

Bronislav Přibyl
ipribyl@fit.vutbr.cz
Pavel Zemčík
zemcik@fit.vutbr.cz
Martin Čadík
cadik@fit.vutbr.cz

Graph@FIT, CPhoto@FIT
Department of Computer Graphics and Multimedia
Faculty of Information Technology
Brno University of Technology
Božetěchova 2
612 66 Brno
Czech Republic

Abstract

Correspondences between 3D lines and their 2D images captured by a camera are often used to determine position and orientation of the camera in space. In this work, we propose a novel algebraic algorithm to estimate the camera pose. We parameterize 3D lines using Plücker coordinates that allow linear projection of the lines into the image. A line projection matrix is estimated using Linear Least Squares and the camera pose is then extracted from the matrix. An algebraic approach to handle mismatched line correspondences is also included. The proposed algorithm is an order of magnitude faster yet comparably accurate and robust to the state-of-the-art, it does not require initialization, and it yields only one solution. The described method requires at least 9 lines and is particularly suitable for scenarios with 25 and more lines, as also shown in the results.

1 Introduction

Camera pose estimation is the task of determining the position and orientation of a camera in 3D space and it has many applications in computer vision, cartography, and related fields. Augmented reality, robot localization, navigation, or 3D reconstruction are just a few of them. To estimate the camera pose, correspondences between known real world features and their counterparts in the image plane of the camera have to be learned. The features can be *e.g.* points, lines, or combinations of both [16]. The task has been solved using *point correspondences* first [12, 20]. This is called the *Perspective-n-Point* (PnP) problem and it still enjoys attention of the scientific community [11]. Camera pose can also be estimated using *line correspondences*, which is called the *Perspective-n-Line* (PnL) problem. A remarkable progress in solving PnL has been achieved in the last years [5, 21, 25], particularly thanks to the work of Mirzaei and Roumeliotis [21] and more recently to the work of Zhang *et al.* [25]. Both of the methods are accurate, cope well with noisy data, and they are more efficient than the previously known methods. Computational efficiency is a critical aspect for many applications and we show that it can be pushed even further.

We propose an efficient solution to the PnL problem which is substantially faster yet accurate and robust compared to the state-of-the-art [21, 25]. The idea is to parameterize the 3D lines using Plücker coordinates [4] to allow using Linear Least Squares to estimate the

projection matrix. The camera pose parameters are then extracted from the projection matrix by posterior constraint enforcement.

The proposed method (**i**) is more than the order of magnitude faster than the state-of-the-art [21, 25], (**ii**) yields only one solution of the PnL problem, and (**iii**) similarly to the state-of-the-art, copes well with image noise, and is initialization-free. These advantages make the proposed method particularly suitable for scenarios with many lines. The method needs 9 lines in the minimal case, so it is not practical for a RANSAC-like framework because it would result in increased number of iterations. To eliminate this, we involve an alternative algebraic scheme to deal with mismatched line correspondences.

The rest of this paper is organized as follows. We present a review of related work in Section 2. Then we state the basics of parameterizing 3D lines using Plücker coordinates in Section 3, show how the lines are projected onto the image plane and how we exploit it to estimate the camera pose. We evaluate the performance of our method using simulations and real-world experiments in Section 4, and conclude in Section 5.

2 Related work

The task of camera pose estimation from line correspondences is receiving attention for more than two decades. Some of the earliest works are the ones of Liu *et al*. [18] and Dhome *et al*. [10]. They introduce two different ways to deal with the PnL problem which can be tracked until today – algebraic and iterative approaches.

The *iterative approaches* consider pose estimation as a Nonlinear Least Squares problem by iteratively minimizing specific cost function, which usually has a geometrical meaning. Earlier works [18] attempted to estimate the camera position and orientation separately while the latter ones [7, 9, 17] favour simultaneous estimation. The problem is that majority of iterative algorithms do not guarantee convergence to the global minimum; therefore, without an accurate initialization, the estimated pose is often far from the true camera pose.

The *algebraic approaches* estimate the camera pose by solving a system of (usually polynomial) equations, minimizing an algebraic error. Dhome *et al*. [10] and Chen [6] solve the minimal problem of pose estimation from 3 line correspondences whereas Ansar and Danilidis [2] work with 4 or more lines. Their algorithm has quadratic computational complexity depending on the number of lines and it may fail if the polynomial system has more than 1 solution. More crucial disadvantage of these methods is that they become unstable in the presence of image noise and must be plugged into a RANSAC or similar loop.

Recently, two major improvements of algebraic approaches have been achieved. First, Mirzaei and Roumeliotis [21] proposed a method which is both efficient (linear computational complexity depending on the number of lines) and robust in the presence of image noise. The cases with 3 or more lines can be handled. 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 the Linear Least Squares. Nonetheless, the problem of this algorithm is that it often yields multiple solutions.

The second recent improvement is the work of Zhang *et al*. [25]. 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. A drawback of

this algorithm is that the computational time increases strongly for higher number of lines.

In this paper, we propose an algebraic solution to the PnL problem which is an order of magnitude faster than the two described state-of-the-art methods yet it is comparably accurate and robust in the presence of image noise.

3 Pose estimation using Plücker coordinates

Let us assume that we have (**i**) a calibrated pinhole camera and (**ii**) correspondences between 3D lines and their images obtained by the camera. The 3D lines are parameterized using Plücker coordinates (Section 3.1) which allows linear projection of the lines into the image (Section 3.2). A line projection matrix can thus be estimated using Linear Least Squares (Section 3.3). The camera pose parameters are extracted from the line projection matrix (Section 3.4). An outlier rejection scheme must be employed in cases where line mismatches occur (Section 3.5). For the pseudocode of our algorithm, please refer to Appendix A in the supplementary material [22]. An implementation of our algorithm in Matlab is also provided.

Let us now define the coordinate systems: a world coordinate system $\{W\}$ and a camera coordinate system $\{C\}$, both 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 situated in front of the camera have negative z coordinates in $\{C\}$. A homogeneous 3D point $\mathbf{A}^W = (a_x^W\ a_y^W\ a_z^W\ a_w^W)^\top$ in $\{W\}$ is transformed into a point $\mathbf{A}^C = (a_x^C\ a_y^C\ a_z^C\ a_w^C)^\top$ in $\{C\}$ as

$$\mathbf{A}^C = \begin{pmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \mathbf{A}^W , \tag{1}$$

where $\mathbf{R}$ is a 3×3 rotation matrix describing the orientation of the camera in $\{W\}$ by means of three consecutive rotations along the three axes z, y, x by respective angles γ, β, α. $\mathbf{t} = (t_x\ t_y\ t_z)^\top$ is a 3×1 translation vector representing the position of the camera in $\{W\}$.

Let us now assume that we have a calibrated pinhole camera (i.e. we know its intrinsic parameters), which observes a set of 3D lines. Given $n \geq 9$ 3D lines $\mathbf{L}_i$ $(i = 1 \ldots n)$ and their respective projections $\mathbf{l}_i$ onto the normalized image plane, we are able to estimate the camera pose. We parameterize the 3D lines using Plücker coordinates.

3.1 Plücker coordinates of 3D lines

3D lines can be represented using several parameterizations in the projective space [4]. 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 $\mathbf{A} = (a_x\ a_y\ a_z\ a_w)^\top$ and $\mathbf{B} = (b_x\ b_y\ b_z\ b_w)^\top$ in homogeneous coordinates, a line joining them can be represented using Plücker coordinates as a homogeneous 6-vector $\mathbf{L} = (\mathbf{u}^\top\ \mathbf{v}^\top)^\top = (L_1\ L_2\ L_3\ L_4\ L_5\ L_6)^\top$, where

$$\begin{aligned} \mathbf{u}^\top &= (L_1\ L_2\ L_3) = (a_x\ a_y\ a_z) \times (b_x\ b_y\ b_z) \\ \mathbf{v}^\top &= (L_4\ L_5\ L_6) = a_w(b_x\ b_y\ b_z) - b_w(a_x\ a_y\ a_z) , \end{aligned} \tag{2}$$

'$\times$' denotes a vector cross product. 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

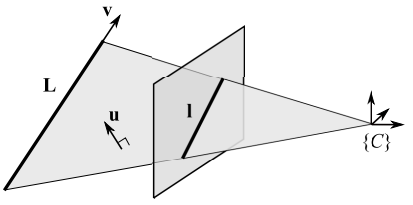

Figure 1: 3D line projection. The 3D line **L** is parameterized by its direction vector **v** and a normal **u** of its interpretation plane, which passes through the origin of the camera coordinate system $\{C\}$. Since the projected 2D line **l** lies at the intersection of the interpretation plane and the image plane, it is fully defined by the normal **u**.

constraint $\mathbf{u}^\top\mathbf{v} = 0$. Existence of this constraint explains the discrepancy between 4 degrees of freedom of a 3D line and its parameterization by a homogeneous 6-vector. More on Plücker coordinates can be found in [15].

3.2 Projection of 3D lines

3D lines can be transformed from the world coordinate system $\{W\}$ into the camera coordinate system $\{C\}$ using the 6×6 line motion matrix $\mathbf{T}$ [3] as

$$\mathbf{L}^C = \mathbf{T}\mathbf{L}^W \quad . \tag{3}$$

The line motion matrix is defined as

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{R}[-\mathbf{t}]_\times \\ \mathbf{0}_{3\times 3} & \mathbf{R} \end{pmatrix} \quad , \tag{4}$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $[\mathbf{t}]_\times$ is a 3×3 skew-symmetric matrix constructed from the translation vector $\mathbf{t}$[1]. After 3D lines are transformed into the camera coordinate system, their projections onto the image plane can be determined as intersections of their interpretation planes with the image plane; see Figure 1 for illustration.

Recall from Eq. (2) that coordinates of a 3D line consist of two 3-vectors: **u** (normal of an interpretation plane) and **v** (direction of a line). Since **v** is not needed to determine the projection of a line, only **u** needs to be computed. Thus, when transforming a 3D line according to Eq. (3) in order to calculate is projection, only the upper half of $\mathbf{T}$ is needed, yielding the 3×6 line projection matrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{R} & \mathbf{R}[-\mathbf{t}]_\times \end{pmatrix} \quad . \tag{5}$$

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

$$\mathbf{l}^C \approx \mathbf{P}\mathbf{L}^W \quad , \tag{6}$$

where $\mathbf{l}^C = (l_x^C\ l_y^C\ l_w^C)^\top$ is a homogeneous 2D line in the normalized image plane and '$\approx$' denotes an equivalence of homogeneous coordinates, i.e. equality up to multiplication by a scale factor.

3.3 Linear estimation of the line projection matrix

As the projection of 3D lines is defined by Eq. (6), the problem of camera pose estimation resides in estimating the line projection matrix $\mathbf{P}$, which encodes all the six camera pose parameters t_x, t_y, t_z, α, β, γ.

We solve this problem using the Direct Linear Transformation (DLT) algorithm, similarly to Hartley [14] who works with points. The system of linear equations (6) can be transformed into a homogeneous system

$$\mathbf{M}\mathbf{p} = \mathbf{0} \tag{7}$$

[1] Please note that our line motion matrix differs slightly from the matrix of Bartoli and Sturm [3, Eq. (6)], namely in the upper right term: We have $\mathbf{R}[-\mathbf{t}]_\times$ instead of $[\mathbf{t}]_\times\mathbf{R}$ due to different coordinate system.

by transforming each equation of (6) so that only a 0 remains at the right-hand side. This forms a $2n \times 18$ measurement matrix $\mathbf{M}$ which contains coefficients of equations generated by correspondences between 3D lines and their projections $\mathbf{L}_i \leftrightarrow \mathbf{l}_i$ $(i = 1 \dots n,\ n \geq 9)$. For details on construction of $\mathbf{M}$, please refer to Appendix B in the supplementary material [22].

The DLT then solves (7) for $\mathbf{p}$ which is a 18-vector containing the entries of the line projection matrix $\mathbf{P}$. Eq. (7), however, holds only in the noise-free case. If a noise is present in the measurements, an inconsistent system is obtained.

$$\mathbf{M}\hat{\mathbf{p}} = \boldsymbol{\varepsilon} \tag{8}$$

Only an approximate solution $\hat{\mathbf{p}}$ may be found through minimization of a $2n$-vector of measurement residuals $\boldsymbol{\varepsilon}$ in the least squares sense on the right hand side of Eq. (8).

Since DLT algorithm is sensitive to the choice of coordinate system, it is crucial to prenormalize the data to get properly conditioned $\mathbf{M}$ [13]. Thanks to the principle of duality [8], coordinates of 2D lines can be treated as homogeneous coordinates of 2D points. The points should be translated and scaled so that their centroid is at the origin and their average distance from the origin is equal to $\sqrt{2}$.

The Plücker coordinates of 3D lines cannot be treated as homogeneous 5D points because of the bilinear constraint (see Section 3.1). However, the closest point to a set of 3D lines can be computed using the Weiszfeld algorithm [1] and the lines can be translated so that the closest point is the origin.

Once the system of linear equations given by (8) is solved in the least squares sense, *e.g.* by Singular Value Decomposition (SVD) of $\mathbf{M}$, the estimate $\hat{\mathbf{P}}$ of the 3×6 line projection matrix $\mathbf{P}$ can be recovered from the 18-vector $\hat{\mathbf{p}}$.

3.4 Estimation of the camera pose

The 3×6 estimate $\hat{\mathbf{P}}$ of the line projection matrix $\mathbf{P}$ obtained as a least squares solution of Eq. (8) does not satisfy the constraints imposed on $\mathbf{P}$. In fact, $\mathbf{P}$ has only 6 degrees of freedom – the 6 camera pose parameters t_x, t_y, t_z, α, β, γ. It has, however, 18 entries suggesting that it has 12 independent linear constraints, see Eq. (5). The first six constraints are imposed by the rotation matrix $\mathbf{R}$ that must satisfy the orthonormality constraints (unit-norm and mutually orthogonal rows). The other six constraints are imposed by the skew-symmetric matrix $[\mathbf{t}]_\times$ (three zeros on the main diagonal and antisymmetric off-diagonal elements). We propose the following method to extract the camera pose parameters from the estimate $\hat{\mathbf{P}}$.

First, the scale of $\hat{\mathbf{P}}$ has to be determined, since $\hat{\mathbf{p}}$ is usually of unit length as a minimizer of $\boldsymbol{\varepsilon}$ in Eq. (8). The correct scale of $\hat{\mathbf{P}}$ can be determined from its left 3×3 submatrix $\hat{\mathbf{P}}_1$ which is an estimate of the rotation matrix $\mathbf{R}$. Since the determinant of an orthonormal matrix must be equal to 1, $\hat{\mathbf{P}}$ has to be scaled by a factor $s = 1/\sqrt[3]{\det \hat{\mathbf{P}}_1}$ so that $\det(s\hat{\mathbf{P}}_1) = 1$.

Second, the camera pose parameters can be extracted from $s\hat{\mathbf{P}}_2$, the scaled right 3×3 submatrix of $\hat{\mathbf{P}}$. The right submatrix is an estimate of a product of an orthonormal and a skew-symmetric matrix ($\mathbf{R}[-\mathbf{t}]_\times$) which has the same structure as the essential matrix [19] used in multi-view computer vision. Therefore, we use a method for the decomposition of an essential matrix into a rotation matrix and a skew-symmetric matrix (see [15, p. 258]) as follows: Let $s\hat{\mathbf{P}}_2 = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ be the SVD of the scaled 3×3 submatrix $s\hat{\mathbf{P}}_2$, and let

$$\mathbf{Z} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{9}$$

Two possible solutions (A and B) exist for the estimate $\hat{\mathbf{R}}$ of the rotation matrix and estimate $[\hat{\mathbf{t}}]_\times$ of the skew-symmetric matrix:

$$\begin{aligned} \hat{\mathbf{R}}_\mathrm{A} &= \mathbf{U}\mathbf{W}\ \mathrm{diag}(1\,1\,\pm 1)\mathbf{V}^\top, \quad & [\hat{\mathbf{t}}]_{\times\mathrm{A}} &= \sigma\mathbf{V}\mathbf{Z}\ \mathbf{V}^\top \\ \hat{\mathbf{R}}_\mathrm{B} &= \mathbf{U}\mathbf{W}^\top\mathrm{diag}(1\,1\,\pm 1)\mathbf{V}^\top, \quad & [\hat{\mathbf{t}}]_{\times\mathrm{B}} &= \sigma\mathbf{V}\mathbf{Z}^\top\mathbf{V}^\top \end{aligned}, \tag{10}$$

where $\sigma = (\Sigma_{1,1} + \Sigma_{2,2})/2$ is an average of the first two singular values of $s\hat{\mathbf{P}}_2$ (a properly constrained essential matrix has the first and second singular values equal to each other and the third one is zero). The ± 1 term in Eq. (10) denotes either 1 or -1 which has to be put on the diagonal so that $\det\hat{\mathbf{R}}_\mathrm{A} = \det\hat{\mathbf{R}}_\mathrm{B} = 1$.

The correct solution A or B is chosen based on a simple check whether 3D lines are in front of the camera or not. Extraction of the components t_x, t_y, t_z of the translation vector from the skew symmetric matrix $[\mathbf{t}]_\times$ and also extraction of the rotation angles α, β, γ from the rotation matrix $\mathbf{R}$ are straightforward. This completes the pose estimation procedure.

Alternative ways of extracting the camera pose parameters from $s\hat{\mathbf{P}}$ also exist, *e.g.* computing the closest rotation matrix $\hat{\mathbf{R}}$ to the left 3×3 submatrix of $s\hat{\mathbf{P}}_1$ and then computing $[\hat{\mathbf{t}}]_\times = -\hat{\mathbf{R}}^\top s\hat{\mathbf{P}}_2$. However, our experiments showed that the alternative ways are less robust to image noise. Therefore, we have chosen the solution described in this section.

3.5 Rejection of mismatched lines

In practice, mismatches of lines (i.e. outlying correspondences) often occur, which degrades the performance of camera pose estimation. RANSAC algorithm is commonly used to identify and remove outliers; however, as our method works with 9 and more line correspondences, it is unsuitable for use in a RANSAC-like framework because the required number of correspondences leads to increased number of iterations.

For this reason, we use an alternative scheme called Algebraic Outlier Rejection (AOR) recently proposed by Ferraz *et al*. [11]. It is an iterative approach integrated directly into the pose estimation procedure (specifically, into solving Eq. (8) in Section 3.3) in form of Iteratively Reweighted Least Squares. Wrong correspondences are identified as outlying based on the residual ε_i of the least squares solution in Eq. (8). 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 $\varepsilon_\mathrm{max} = \mathrm{Q}_j(\varepsilon_1,\ldots,\varepsilon_n)$ 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, ... , 0.3) for the first 7 iterations and then it remains constant 0.25. This strategy usually leads to approximately 10 iterations.

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.

4 Experimental evaluation

Accuracy, robustness, and efficiency of the proposed algorithm were evaluated and compared with the state-of-the-art methods. The following methods were compared:

1. **Mirzaei**, the method by Mirzaei and Roumeliotis [21] (results shown in red ■),
2. **Zhang**, the method by Zhang *et al.* [25] (results shown in blue ■),
3. **ours**, the proposed method (results shown in green ■).

Both simulations using synthetic lines and experiments using the real-world imagery are presented.

4.1 Synthetic lines

Monte Carlo simulations with synthetic lines were performed under the following setup: at each trial, n 3D line segments were generated by randomly placing segment endpoints inside a cube 10^3 m large which was centered at the origin of $\{W\}$. A virtual pinhole camera with image size of 640×480 pixels and focal length of 800 pixels was placed randomly in the distance of 25 m from the origin. The camera was then oriented so that it looked directly at the origin, having all 3D line segments in its field of view. The 3D line segments were projected onto the image plane. Coordinates of the 2D endpoints were then perturbed with independent and identically distributed Gaussian noise with standard deviation of σ_p pixels. 1000 trials were carried out for each combination of n, σ_p parameters.

Accuracy and robustness of each method was evaluated by measuring the estimated and true camera pose while varying n and σ_p similarly to [21]. The position error $\Delta\tau = ||\hat{\mathbf{t}} - \mathbf{t}||$ is the distance from the estimated position $\hat{\mathbf{t}}$ to the true position $\mathbf{t}$. The orientation error $\Delta\Theta$ was calculated as follows. The difference between the true and estimated rotation matrix ($\mathbf{R}^\top\hat{\mathbf{R}}$) is converted to axis-angle representation ($\mathbf{e}$, θ) and the absolute value of the difference angle $|\theta|$ is considered as the orientation error.

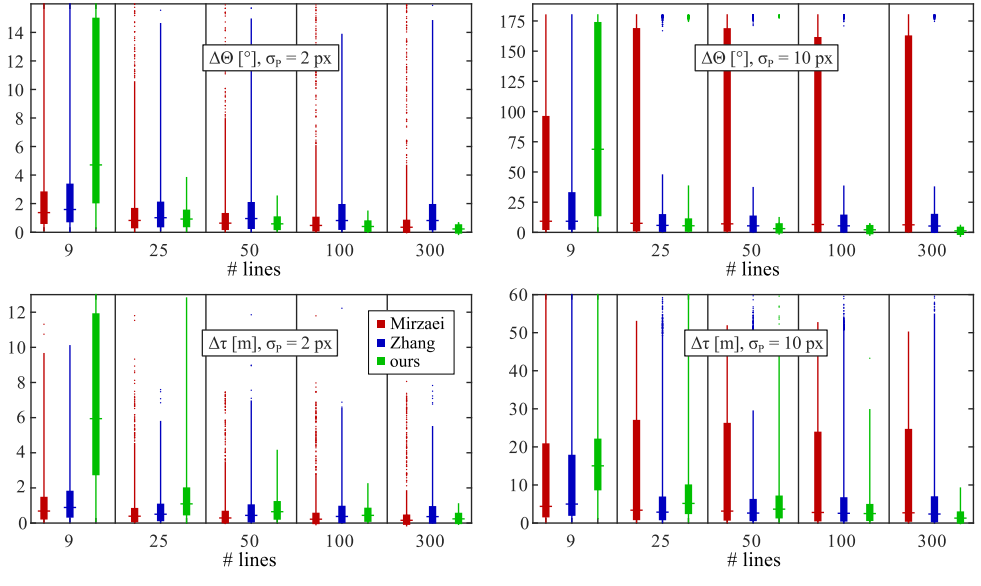

Figure 2: The distribution of orientation errors ($\Delta\Theta$, **top**) and position errors ($\Delta\tau$, **bottom**) in estimated camera pose as a function of the number of lines. Two levels of Gaussian noise are depicted: with standard deviation of $\sigma_p = 2$ px (**left**) and with $\sigma_p = 10$ px (**right**). Each box depicts the median (*dash*), interquartile range - IQR (*box body*), minima and maxima in the interval of $10\times$ IQR (*whiskers*) and outliers (*isolated dots*).

As illustrated in Figure 2, 25 lines are generally enough for our method to be on par with the state-of-the-art in terms of accuracy. 50 and more lines are usually exploited better by our method. As the number of lines grows, our method becomes even more accurate than the others. It should be noted that the orientation error decreases more rapidly than the position error with the number of lines. Our method is outperformed by the others in the minimal case of 9 lines. However, as soon as more lines are available, the results of our approach rapidly improve. This fact is a matter of chosen parameterization. Plücker coordinates of 9 lines are just enough to define all 18 entries of the line projection matrix $\mathbf{P}$ in Eq. (5). More lines bring redundancy into the system and compensate for noise in the measurements. However, even 9 lines are enough to produce an exact solution in a noise-free case.

All the three methods sometimes yield an improper estimate with exactly opposite orientation. This can be observed as isolated dots particularly in Figure 2 (top, right). Furthermore, the method of Mirzaei sometimes produced an estimate where the camera is located in between the 3D lines and it has random orientation. This happened more frequently in the presence of stronger image noise, as it is apparent from increased red bars in Figure 2 (right). The robustness of Mirzaei's method is thus much lower compared to our method and Zhang's method. However, the method of Zhang sometimes produced a degenerate pose estimate very far from the correct camera position when the 3D lines projected onto a single image point (this phenomenon cannot be seen in Figure 2 as such estimates are out of scale of the plots). The proposed method does not suffer from any of these two issues and is more robust in cases with 50 and more lines.

4.2 Real images

The three methods were also tested using real-world images from the VGG Multiview Dataset[2]. It contains indoor and outdoor image sequences of buildings with extracted 2D line segments, their reconstructed positions in 3D, and camera projection matrices. Each method was run on the data and the estimated camera poses were used to reproject the 3D lines onto the images to validate the results.

The proposed algorithm performs similarly or better than Zhang's method while Mirzaei's method behaves noticeably worse, as it can be seen in Figure 3 and Table 1. Detailed results with all images from the sequences are available as supplementary material [22].

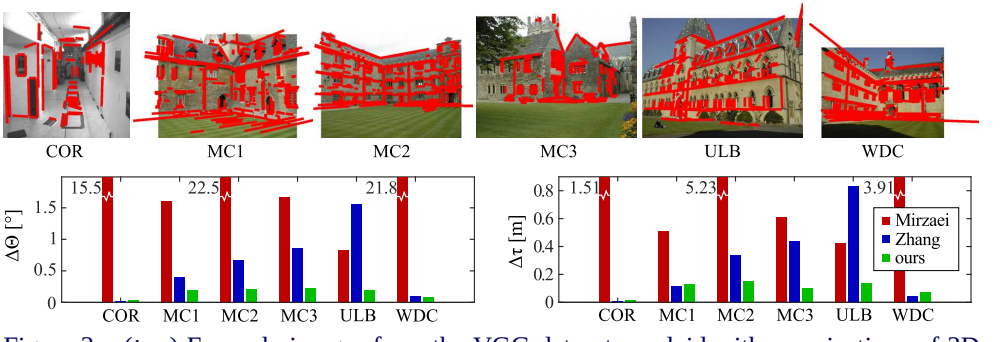

Figure 3: (**top**) Example images from the VGG dataset overlaid with reprojections of 3D line segments using our estimated camera pose. (**bottom**) Average camera orientation error $\Delta\Theta = |\theta|$ and average position error $\Delta\tau = ||\hat{\mathbf{t}} - \mathbf{t}||$ in individual image sequences.

[2] http://www.robots.ox.ac.uk/~vgg/data/data-mview.html

Sequence	# lines	# imgs.	Mirzaei $\Delta\Theta$	Mirzaei $\Delta\tau$	Zhang $\Delta\Theta$	Zhang $\Delta\tau$	ours $\Delta\Theta$	ours $\Delta\tau$
Corridor	69	11	15.510°	1.510 m	**0.029°**	**0.008 m**	0.034°	0.013 m
Merton College I	295	3	1.610°	0.511 m	0.401°	**0.115 m**	**0.195°**	0.128 m
Merton College II	302	3	22.477°	5.234 m	0.676°	0.336 m	**0.218°**	**0.151 m**
Merton College III	177	3	1.667°	0.608 m	0.859°	0.436 m	**0.223°**	**0.101 m**
University Library	253	3	0.837°	0.423 m	1.558°	0.833 m	**0.189°**	**0.138 m**
Wadham College	380	5	21.778°	3.907 m	0.103°	**0.047 m**	**0.086°**	0.072 m

Table 1: Results of the methods on the VGG dataset in terms of average camera orientation error $\Delta\Theta = |\theta|$ and average position error $\Delta\tau = ||\hat{\mathbf{t}} - \mathbf{t}||$. Best results are in bold.

4.3 Efficiency

Efficiency of each method was evaluated by measuring runtime on a desktop PC with a quad core Intel i5 3.33 GHz CPU. Matlab implementations downloaded from the websites of the respective authors were used. As it can be seen in Table 2 and Figure 4, our method significantly outperforms the others in terms of speed. Computational complexity of all evaluated methods is linearly dependent on the number of lines. However, the absolute numbers differ substantially. Mirzaei's method is slower than Zhang's method for up to cca 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. However, Zhang's method is slower than Mirzaei's for more than 200 lines. Our method is the fastest no matter how many lines are processed; it is approximately one order of magnitude faster than both competing methods. The linear computational complexity of our method is only achieved due to the prenormalization of input data and subsequent SVD of the $2n \times 18$ measurement matrix $\mathbf{M}$; all the other computations are performed in constant time.

# lines	9	100	1000
Mirzaei	72.0	79.5	168.2
Zhang	8.7	42.1	899.4
ours	**3.2**	**3.8**	**28.5**

Table 2: Runtimes in milliseconds for varying number of lines, averaged over 1000 runs.

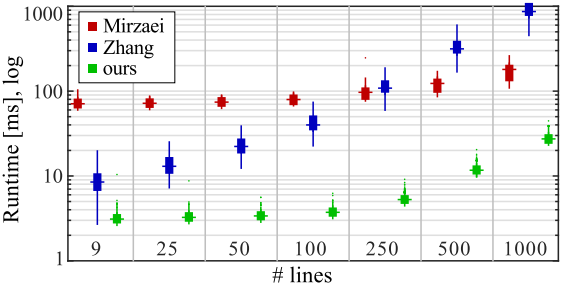

Figure 4: The distribution of runtimes as a function of the number of lines. Logarithmic vertical axis. Meaning of the boxes is the same as in Figure 2.

4.4 Robustness to outliers

As a practical requirement, robustness to outlying correspondences was also tested. The experimental setup was the same as in Section 4.1, using $n = 500$ lines which endpoints were perturbed with slight image noise with $\sigma_p = 2$ pixels. The image lines simulating outlying correspondences were perturbed with an aditional extreme noise with $\sigma_p = 100$ pixels. The methods of Mirzaei and Zhang were plugged into a MLESAC (an improved version of

RANSAC) [24] framework, generating camera pose hypotheses from 3 and 4 randomly selected line correspondences, respectively. The inlying correspondences were identified based on the line reprojection error [23]. No heuristics for early hypothesis rejection was utilized, as it can also be incorporated into the Algebraic Outlier Rejection scheme, *e.g.* by weighting the line correspondences. The proposed method with AOR was set up as described in Section 3.5.

While the RANSAC-based approaches can theoretically handle any percentage of outliers, the proposed method with AOR has a break-down point at about 30 % of outliers, as depicted in Figure 5. However, for the lower percentage of outliers, our method is more accurate and 5-7× faster.

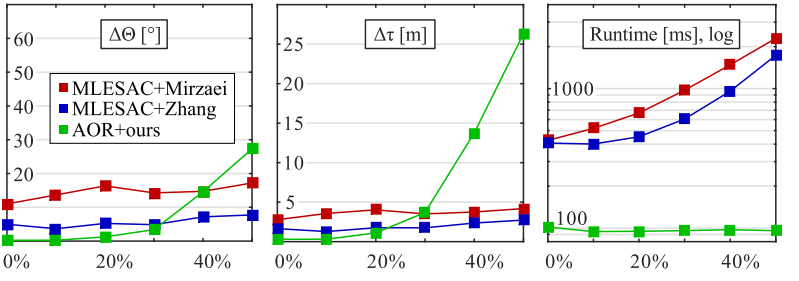

Figure 5: Camera pose errors (**left**, **center**) and runtime (**right**) depending on the percentage of outliers. $n = 500$ lines, $\sigma_p = 2$ pixels, averaged over 1000 runs.

The original AOR approach applied to the PnP problem [11] has a higher break-down point at 45 %. We think it might be because the authors need to estimate a null space with only 12 entries whereas we estimate 18 entries of the nullspace $\hat{\mathbf{p}}$ in Eq. (8). The use of barycentric coordinates for parameterization of 3D points in [11] may also play a role.

5 Conclusions

In this paper, a novel algebraic approach to the Perspective-n-Line problem is proposed. The approach is substantially faster, yet equally accurate and robust compared to the state-of-the-art. The superior computational efficiency of the proposed method achieving speed-ups of more than one order of magnitude for high number of lines is proved by simulations and experiments. As an alternative to the commonly used RANSAC, Algebraic Outlier Rejection is used to deal with mismatched lines. The proposed method requires at least 9 lines, but it is particularly suitable for large scale and noisy scenarios. For very small size noisy scenarios (≤ 25 lines), the state-of-the-art performs better and we recommend to use the Zhang's method. Future work involves examination of the degenerate line configurations.

The Matlab code of the proposed method and the appendices are publicly available in the supplementary material [22].

Acknowledgements This work was supported by the Technology Agency of the Czech Republic by projects TA02030835 D-NOTAM and TE01020415 V3C. It was also supported by SoMoPro II grant (financial contribution from the EU 7 FP People Programme Marie Curie Actions, REA 291782, and from the South Moravian Region). The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

References

[1] K. Aftab, R. I. Hartley, and J. Trumpf. Lq-closest-point to affine subspaces using the generalized weiszfeld algorithm. *International Journal of Computer Vision*, February 2015. ISSN 1573-1405.

[2] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):578–589, 2003.

[3] A. Bartoli and P. Sturm. The 3d line motion matrix and alignment of line reconstructions. *International Journal of Computer Vision*, 57:159–178, 2004.

[4] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3): 416–441, 2005.

[5] K. K. Bhat and J. Heikkila. Line matching and pose estimation for unconstrained model-to-image alignment. In *International Conference on 3D Vision 2014*, volume 1, pages 155–162. IEEE, 2014.

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

[7] S. Christy and R. Horaud. Iterative pose computation from line correspondences. *Computer vision and image understanding*, 73(1):137–144, 1999.

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

[9] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. Simultaneous pose and correspondence determination using line features. In *IEEE Conference on Computer Vision and Pattern Recognition 2003*, volume 2, pages 424–431. IEEE, 2003.

[10] M. Dhome, M. Richetin, J.-T. Lapreste, and G. Rives. Determination of the attitude of 3d objects from a single perspective view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, 1989.

[11] L. Ferraz, X. Binefa, and F. Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *IEEE Conference on Computer Vision and Pattern Recognition 2014*, pages 501–508. IEEE, 2014.

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

[13] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[14] R. I. Hartley. Minimizing algebraic error. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356 (1740):1175–1192, 1998.

[15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. ISBN 0521540518.

[16] Y. Kuang and K. Astrom. Pose estimation with unknown focal length using points, directions and lines. In *IEEE International Conference on Computer Vision 2013*, pages 529–536. IEEE, 2013.

[17] R. Kumar and A. R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVGIP: Image understanding*, 60(3):313–342, 1994.

[18] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2-d to 3-d line and point correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, 1990.

[19] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[20] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, 31(3):355–395, 1987.

[21] F. M. Mirzaei and S. I. Roumeliotis. Globally optimal pose estimation from line correspondences. In *IEEE International Conference on Robotics and Automation 2011*, pages 5581–5588. IEEE, 2011.

[22] B. Přibyl, P. Zemčík, and M. Čadík. Supplementary material for camera pose estimation from lines using plücker coordinates, September 2015. URL http://www.fit.vutbr.cz/~ipribyl/pubs.php.en?id=10659.

[23] C. J. Taylor and D. Kriegman. Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11): 1021–1032, 1995.

[24] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[25] L. Zhang, C. Xu, K.-M. Lee, and R. Koch. Robust and efficient pose estimation from line correspondences. In *Asian Conference on Computer Vision 2012*, pages 217–230. Springer, 2013.