Camera Pose Estimation Based on PnL With a Known Vertical Direction

Louis Lecrosnier, Rémi Boutteau, Pascal Vasseur, Xavier Savatier, Friedrich Fraundorfer

Camera pose estimation based on PnL with a known vertical direction

Louis Lecrosnier[1] , Rémi Boutteau[1], Pascal Vasseur [2], Xavier Savatier[1], Friedrich Fraundorfer [3]

***Abstract*—In this paper, we address the problem of camera pose estimation using 2D and 3D line features, also known as PnL (Perspective-n-Line) with a known vertical direction.**

The minimal number of line correspondences required to estimate the complete camera pose is 3 (P3L) in the general case, yielding to a minimum of 8 possible solutions. Prior knowledge of the vertical direction, such as provided by common sensors (e.g. Inertial Measurement Unit, or IMU), reduces the problem to a 4 Degree of Freedom (DoF) problem and outputs a single solution. We benefit this fact to decouple the remaining rotation estimation and the translation estimation and we present a two-fold contribution: (1) we present a linear formulation of the PnL problem in Plücker lines coordinates with a known vertical direction, including a Gauss-Newton-based orientation and location refinement to compensate IMU sensor noise. (2) we propose a new efficient RANdom SAmple Consensus (RANSAC) scheme for both feature pairing and outliers rejection based solely on rotation estimation from 2 line pairs. This greatly diminishes the computational cost compared to a RANSAC3 or RANSAC4 scheme.

We evaluate our algorithms on synthetic data and on our own real dataset. Experimental results show state of the art results in term of accuracy and runtime, when facing 2D noise, 3D noise and vertical direction sensor noise.

***Index Terms*—Computer Vision for Other Robotic Applications, Sensor Fusion, Localization**

I. INTRODUCTION

Camera pose estimation consists in determining the position and the orientation of a camera with respect to a reference frame. This process requires known correspondences between real world features and their projection onto the image plane. When these features are points, we refer to the well-studied Perspective-n-Point (PnP) problem [1] [2] [3] [4] [5]. In the case of line features, we are facing the challenging, more recent and less studied Perspective-n-Line (PnL) problem. A recent review of the latter methods is presented in [6].

Once the 2D and 3D lines extraction process is completed, feature pairing is not a simple task: lines lack effective descriptors, and descriptor-based pose estimation methods are computationally more expensive.

Manuscript received: February, 22, 2019; Revised April, 24, 2019; Accepted June, 26, 2019.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers comments. This work was carried out as part of the COPTER research project, and is co-funded by the European Union and the Région Normandie. Europe is involved in Normandy with the European Regional Development Fund (ERDF).

[1]Authors are with Normandie Univ, UNIROUEN, ESIGELEC, IRSEEM, 76000 Rouen, France

[2]Author is with Normandie Univ, UNIROUEN, UNIHAVRE, INSA Rouen, LITIS, 76000 Rouen, France

[3]Author is with Institute for Computer Graphics and Vision, Graz University of Technology, 8010 Graz, Austria, and German Aerospace Center (DLR), Remote Sensing Technology Institute, Germany

When considered, feature pairing is often tackled with a RAndom SAmple Consensus method (RANSAC) [7] using 3 or more lines correspondences. This method can be computationally expensive, regarding the number of pairs required in the process. For n 2D lines and m 3D lines, we have $6.\mathrm{C}_3^m.\mathrm{C}_3^n$ possible combinations for 3 pairs of lines and only $2.\mathrm{C}_2^m.\mathrm{C}_2^n$ combinations for 2 pairs of lines, i.e. for a 20 2D and 3D lines dataset, we have 7,797,600 possible line combinations for 3 pairs of lines and 72,200 for 2 pairs of lines. We easily understand that reducing from 3 to 2 the number of pairs required in the pairing process, and thus the number of combinations to evaluate, can greatly improve the speed of the overall process. Feature pairing is computationally expensive. Performing this task online is challenging for embedded CPUs. Increasing the overall speed of the process could leverage this issue.

Many modern vision-based systems are equipped with relatively cheap and accurate inertial measurement units (IMU), which provide useful information about the system orientation, i.e. give a prior knowledge on two of the three rotations of the camera. This vertical direction vector, or *up-vector*, can be accurate from 0.5° for the cheapest embedded sensors to 0.02° for less affordable sensors [8]. Some PnP and PnL methods rely on the provided *up-vector* to reduce the number of potential pose solutions, to reduce the general problem complexity in case of minimal formulation, or to increase performance [8] [9] [10] [11].

In this paper, our first contribution is a Linear-formulation-based PnL with a known vertical direction. Our formulation relies on the Plücker representation of lines inspired by [12] and [5], but we benefit the known *up-vector* of the system to reduce the PnL problem from 6 DoF to 4 DoF, and to a unique solution. We also propose a pose refinement process based on a Gauss-Newton method to sequentially optimize the estimated rotation and translation.

Our second contribution is a new RANSAC2 algorithm capable of either line-pairing or outliers rejection by simply changing its inputs. With our PnL formulation, the translation estimation requires at least three lines to be solved but the remaining rotation $\mathbf{R}$ can be computed with only 2 pairs of lines. Thereby, we present a RANSAC2 scheme that is solely based on the rotation estimation. Our approach doesn't rely on a prior pairing, and takes advantage of our PnL formulation to proceed simultaneously and efficiently to feature-pairing and pose estimation.

We bring forward a study of the robustness of our PnL formulation and our outliers removal module with both synthetic and real data.

II. Related Work

All PnL methods formulate the pose estimation problem as a set of linear or polynomial equations and minimize algebraic or geometric error (such as reprojection error).

[1] and [13] introduced PnL research in 1989. They formulated the problem with a minimum of three 2D-3D line correspondences, known as P3L. [1] were the first to propose a closed-form solution to the PnL problem with a polynomial approach. [13] stated that in the general case, at least 8 solutions exist for the PnL problem.

A decade later, [14] reduced the number of solutions using 4 or more line-correspondences, and employed polynomial lifting to convert a polynomial system to a linear system of the elements from the rotation matrix. [15] proposed more recently a PnL solution using a minimum of 3 line-correspondences, more robust to noise, but that spans 23 solutions. [16] presented the RLPnL algorithm, requiring at least 4 line correspondences. This algorithm uses subsets of 3 line correspondences and identifies a solution in the derivatives of a 16^{th} order polynomial. [6] proposed a modified RLPnL method named ASPnL (Accurate Subset based PnL), that is more efficient on small noisy datasets, but is very sensitive to outliers. ASPnL estimates the complete pose in homogeneous coordinates, and includes a pose refinement module based on a Gauss-Newton optimization.

[5] [17] proposed a PnL algorithm based on the DLT (Direct Linear Transform) algorithm of [18], using the Plücker line representation. While we use the same coordinates parameterization, we solve the PnL problem differently. They recover rotation and translation by solving a homogeneous system with singular value decomposition, where we rely on a linear least squares solver. Their method is efficient and accurate with a high number of lines (up to a few thousands), but is inaccurate for small datasets [6].

On the topic of PnL with a known vertical direction (or *up-vector*), [19] proposed two PnL algorithms relying on modified Plücker coordinates, known as NPnLUpL and NPnLUpC (N-camera PnL Up-vector Linear and Cubic). The first one solves the orientation and position from a single system of linear equations producing a unique solution, while the later recovers the pose from a cubic polynomial, thus yielding 3 solutions. To perform accurately these algorithms require a calibrated multi-camera setup, and known correspondences between 2D-lines. These assumptions are not suitable for a lightweight robotic application (*eg.* UAV with a monocular camera).

Several outliers rejection schemes exist within the litterature. A first category is based on the PnP outliers rejection scheme by [4], and tends to minimize an algebraic error [6] [17]. These methods struggle with small and noisy datasets, but handle efficiently large datasets.

On the other hand, RANSAC methods using 3 or 4 line correspondences are more efficient with smaller datasets, but tend to have an unsuitable runtime for real-time applications with an increasing number of lines in the dataset [6].

Diminishing the number of line correspondences required for the pairing process is necessary to reduce the runtime of the latter methods. For this reason, we address in this paper a novel RANSAC-based pairing process relying on two 2D-3D line correspondences and a linear formulation of the PnL problem with the Plücker line representation.

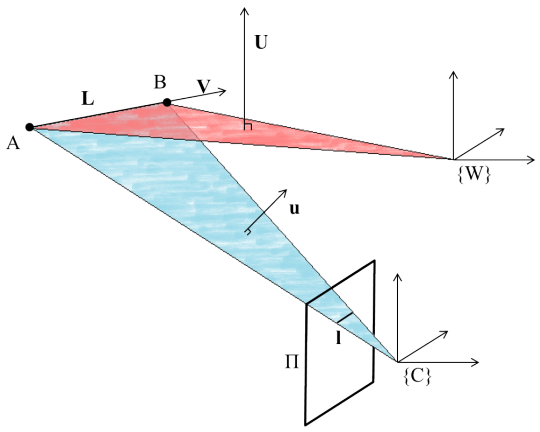


Fig. 1: The 3D line **L** is parameterized by its direction **V** and a normal **U** intersecting **L** and the origin of the world frame $\{\mathrm{W}\}$. **l** is the projection of **L** onto the image plane Π in the camera frame $\{\mathrm{C}\}$.

III. The Perspective-N-Line problem

We assume in this section that we have a Z-axis-facing calibrated pinhole camera, with an associated IMU (inertial measurement unit) giving us the vertical direction of the camera (i.e. two of the three rotations are known).

A. 3D Line parameterization

As seen in Fig.1, Plücker coordinates define a 3D line **L** in the world frame $\{\mathrm{W}\}$ from its direction **V** and a plane **U** passing through the line and the origin, in the form of a homogeneous 6-vector. Given two distinct 3D points in homogeneous coordinates $\mathbf{A} = (a_1, a_2, a_3, a_4)^T$ and $\mathbf{B} = (b_1, b_2, b_3, b_4)^T$, we define the homogeneous 6-vector $\mathbf{L} = (\mathbf{U}^T, \mathbf{V}^T)^T$ such as:

$$\mathbf{L} = (L_1, L_2, L_3, L_4, L_5, L_6)^T \tag{1}$$

where

$$\begin{aligned} \mathbf{U} &= (a_1, a_2, a_3) \times (b_1, b_2, b_3) \\ \mathbf{V} &= a_4.(b_1, b_2, b_3) - b_4.(a_1, a_2, a_3), \end{aligned} \tag{2}$$

and '$\times$' being the cross-product. It is important to note that **L** must satisfy the bilinear constraint $\mathbf{U}^T.\mathbf{V} = 0$.

Let $\mathbf{L}^W$ be the expression of **L** in the world frame $\{W\}$. $\mathbf{L}^W$ can be expressed in the camera frame $\{C\}$ with the line motion matrix $\mathbf{M}_{6\times6}$ as:

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

with

$$\mathbf{M} = \begin{pmatrix} \mathbf{R} & -\mathbf{R}.\mathbf{T}_{[\times]} \\ \mathbf{0}_{3\times3} & \mathbf{R} \end{pmatrix}. \tag{4}$$

'$[\times]$' represents the skew symmetric matrix of a 3-vector. $\mathbf{T} = (t_x, t_y, t_z)^T$ is a translation vector and **R** is a 3×3 rotation matrix.

B. Line projection

The projection of the 3D line $\mathbf{L}$ onto the image plane Π is referred as $\mathbf{l}$. $\mathbf{l}$ is defined by the intersection of Π and the projection plane $\mathbf{u}$ passing through the 3D line $\mathbf{L}$ and the origin of the camera frame $\{C\}$. For this reason, the 3-vector $\mathbf{u}$ is sufficient to fully represent $\mathbf{l}$. Let $\mathbf{l}^C$ be the expression of $\mathbf{l}$ in the camera frame. We obtain $\mathbf{l}^C$ by projecting $\mathbf{L}^W$ with the upper part of the line motion matrix $\mathbf{M}$ from equation (4), which is a projection matrix $\mathbf{P}$, with

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

and

$$\mathbf{l}^C \sim \mathbf{P}.\mathbf{L}^W, \tag{6}$$

where $\mathbf{l}^C = (l_1, l_2, l_3)^T$. We use here the same parameterization as [5], derived from [12]. This framework provides a linear projection, that can easily be used as input for a linear least squares pose solver. Note that the equality (6) is true up to a non-zero scale factor.

C. Pose estimation with a known vertical direction

Let $\mathbf{l}_i = (l_{i1}, l_{i2}, l_{i3})^T$ be the 2D projection of the 3D line $\mathbf{L}_i = (L_{i1}, L_{i2}, L_{i3}, L_{i4}, L_{i5}, L_{i6})^T$, $\mathbf{l}_i$ and $\mathbf{L}_i$ being the i^{th} element of a n pairs of lines dataset. We have $\mathbf{L}_i = (\mathbf{U}_i^T, \mathbf{V}_i^T)^T$.

Solving the complete P3L problem requires estimating three rotations ρ, θ, ψ and three translations t_x, t_y, and t_z. IMU sensors provide a prior knowledge on two out of the three rotations, which reduces the problem to a 4 DoF problem.

By definition, $\mathbf{L}_i^C$ lies on the projection plane of the 2D line $\mathbf{l}_i$, leading to the constraint

$$(\mathbf{R}_{CW}.\mathbf{V}_i^W)^T.\mathbf{l}_i^C = 0, \tag{7}$$

where R_{CW} is the rotation matrix composed with the remaining rotation to be determined, such as

$$\begin{aligned} \mathbf{R}_{CW} &= \mathbf{R_z}.\mathbf{R_y}.\mathbf{R_x} \\ &= \begin{pmatrix} c_z & -s_z & 0 \\ s_z & c_z & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} c_y & 0 & s_y \\ 0 & 1 & 0 \\ -s_y & 0 & c_y \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_x & -s_x \\ 0 & s_x & c_x \end{pmatrix} \end{aligned} \tag{8}$$

where $c_x = \cos(\rho)$, $s_x = \sin(\rho)$, $c_y = \cos(\theta)$, $s_y = \sin(\theta)$, $c_z = \cos(\psi)$, $s_z = \sin(\psi)$. Prior knowledge on the vertical direction provides the values for ρ and θ, i.e. provides the product matrix $\mathbf{R}_y.\mathbf{R}_x$, meaning that only two unknowns c_z and s_z remain. For two line correspondences, we obtain a linear system that can be solved for ψ, with a unique solution, through a linear least squares solver.

In a noiseless case, we directly obtain $\mathbf{R}_{CW}$. However, when proceeding with real data, we often obtain a badly scaled rotation matrix $\tilde{\mathbf{R}}$, that requires further refinement. Having recovered c_z and s_z, we can recompose a matrix $\tilde{\mathbf{R}}$ with equation (8). Since c_z and s_z are estimated separately, they might not satisfy the trigonometric constraint $c_z^2 + s_z^2 = 1$. We enforce this constraint with a singular value decomposition of the recomposed $\tilde{\mathbf{R}}$ matrix, such as

$$\begin{aligned} \mathbf{U}.\mathbf{\Sigma}.\mathbf{V}^* &= \tilde{\mathbf{R}} \\ \mathbf{R_{CW}} &= \mathbf{U}.\mathbf{V}^{*T} \end{aligned} \tag{9}$$

In order to estimate the translation vector $\mathbf{T}_{CW}$, we reformulate (6) into

$$\mathbf{l}^C_{[\times]}.\mathbf{P}.\mathbf{L}^W = \mathbf{0}_{3\times 1}. \tag{10}$$

For a 2D/3D line correspondence, we obtain from (10) a system of three equations. However, for only one line correspondence, this system is rank deficient [18]. We need to stack the equations from at least three 2D/3D line correspondences to obtain a rank 3 linear system and recover the three remaining unknowns t_x, t_y and t_z. We rearrange this linear system such as

$$\mathbf{M}.\mathbf{T}_{CW} = \mathbf{N}, \tag{11}$$

each column of $\mathbf{M}$ being respectively expressed in term of t_x, t_y and t_z. $\mathbf{T}_{CW}$ is the translation matrix such as $\mathbf{T}_{CW} = (t_x, t_y, t_z)^T$, and $\mathbf{N}$ contains all terms independent from t_x, t_y and t_z. We finally solve (11) using a linear least squares method, i.e.

$$\mathbf{T}_{CW} = (\mathbf{M}^T.\mathbf{M})^{-1}\mathbf{M}^T.\mathbf{N}. \tag{12}$$

We refer to this method as VPnL_LS (Vertical Perspective-n-Lines).

D. Gauss-Newton optimization for the rotation and the translation

In our PnL formulation, the rotation and translation are sequentially estimated. We propose rotation and translation refinement methods that can also be sequentially applied to the estimated pose.

a) Rotation optimization: Our PnL method relies on the known *up-vector* to estimate the complete pose. However, the *up-vector* is susceptible to sensor error. For this reason, we propose a rotation optimization scheme to refine the estimated camera orientation. Let $\mathbf{A}_i^W$ and $\mathbf{B}_i^W$ be two 3D points in the world frame defining the i^{th} 3D line $\mathbf{L_i}$ of a n line dataset. Let $\mathbf{l_i}$ be the projection of $\mathbf{L_i}$ onto the image plane. $\mathbf{V_i}^W$ and $\mathbf{V_i}^C$ are respectively the 3D line directions in the world and camera frames. For this set of n lines, we express the constraint (7) as a set of n functions $\mathbf{f} = (f_1, ..., f_n)$ depending on the three rotations ρ, θ and ψ defined as in (8), so that

$$f_i(\mathbf{R}_{CW}) = (\mathbf{R}_{CW}.\mathbf{V}_i^W)^T.\mathbf{l}_i = 0. \tag{13}$$

We derive all f_i equations into a jacobian matrix $\mathbf{J_R}$ such as

$$\mathbf{J}_R = \begin{pmatrix} \frac{\partial f_1}{\partial \rho} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial \psi} \\ \dots & \dots & \dots \\ \frac{\partial f_n}{\partial \rho} & \frac{\partial f_n}{\partial \theta} & \frac{\partial f_n}{\partial \psi} \end{pmatrix}. \tag{14}$$

We then iteratively refine the rotation from a step s to the next step $s+1$ with

$$\boldsymbol{\beta}^{s+1} = \boldsymbol{\beta}^s - (\mathbf{J}_\mathbf{R}^T.\mathbf{J_R})^{-1}.\mathbf{J}_\mathbf{R}^T.\mathbf{f}, \tag{15}$$

and $\boldsymbol{\beta} = [\rho, \theta, \psi]^T$.

b) Translation optimization: Any 3D point $\mathbf{A_i}$ lying on the 3D line $\mathbf{L_i}$ can be projected onto the 2D line $\mathbf{l_i}$. We express this constraint similarly to (13) in a set of n functions $\mathbf{g} = (g_1, ..., g_n)$ to refine the translation estimation in

$$g_i(\mathbf{T}_{CW}) = (\mathbf{R}_{CW}.\mathbf{A}_i^W + \mathbf{T}_{CW})^T.\mathbf{l}_i = 0. \quad (16)$$

We also express the jacobian matrix of (16), which we use as in (15) to optimize the translation. When coupled to our PnL algorithm, these two optimizations are referred as VPnL_GN.

Because of the non-linear nature of the functions $\mathbf{f}$ and $\mathbf{g}$, the Gauss-Newton algorithm does not guarantee convergence to the global minimum with a random initialization [20]. However, convergence to optimal parameters can be achieved with an initialization close to the global minimum. We observed that both our methods generally converge within 2 iterations, when initialized with our linear least square solver VPnL_LS. For this reason, we stop the algorithm when the refined parameters reach a stationary value over two consecutive iterations, i.e. convergence is attained, or when a maximum of 20 iterations is reached, i.e. the parameters are extremely close to a local minimum and evolution becomes almost stationary.

E. Line pairing and outliers rejection with RANSAC2

Algorithm 1 RANSAC2 Pairing

procedure PAIRING
  input:
  $\mathbf{l_i}$: normalized 2D line, $i = 1..n$
  $\mathbf{V_j}$: normalized direction of the 3D line, $j = 1..m$
  criterion : loop break threshold
  N : maximum number of iteration
  x : size of each set of data between each quantile
  output:
  inliers
  internal variables:
  $\mathbf{R_{CW}}$: Rotation matrix World $\rightarrow$ Camera
  $\boldsymbol{\varepsilon}$: error vector such as $\mathbf{l_i}^T.\mathbf{R_{CW}}.\mathbf{V_j}$
  ε : first quantile of $\boldsymbol{\varepsilon}$
  Algorithm :
  do
    Random sample two 2D-3D pairs
    Estimate $\mathbf{R_{CW}}$ with a linear least square solver
    Measure $\boldsymbol{\varepsilon} = \mathbf{l_i}^T.\mathbf{R_{CW}}.\mathbf{V_j}$, $\forall\ i = 1..n,\ j = 1..m$
    Split the error vector $\boldsymbol{\varepsilon}$ into $\frac{m.n}{x+1}$ quantiles
    Extract ε
  while $\varepsilon >$ criterion AND iterations counter $< N$
  Return inliers for $\boldsymbol{\varepsilon}_i <$ criterion

Algorithm 1 presents the line pairing process with outliers rejection. First, a random sample of two 2D-3D pairs of lines is used to estimate a rotation matrix $\mathbf{R}$ from the world frame to the camera frame. An error vector $\boldsymbol{\varepsilon}$ is then computed for all possible pairs of lines. We split $\boldsymbol{\varepsilon}$ into $(m.n)/(x+1)$ quantiles, $m.n$ being the number of unique 2D/3D line combinations and x number of lines separated by each quantile. We observe the value of the first quantile of the error vector. A low value for the first quantile of the error vector implies that a subset of x lines generated a low error according to our error function. In this case, we can assume that the estimated rotation is correct, and that the line producing an error value under our threshold are inliers. If not, we iterate until this condition is met, or the maximum authorized number iterations is obtained. In the latter case, we still return the pairs that generated the smallest error over all iterations. Note here that the number of iterations N is defined as

$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}, \quad (17)$$

where p is the probability of selecting a random sample containing only inliers (usually set to 0.99), ε the percentage of outliers in the data, and s the number of data required for the algorithm to operate (i.e. $s = 2$ here).

Regarding the number of quantiles $(m.n)/(x+1)$, experiments suggest that x should be set close to 35% the theoretical number of inliers in the dataset.

IV. RESULTS

A. Synthetic data

In this section, we evaluate quantitatively the performance and robustness of the different steps of our method.

a) Synthetic dataset setup: For the entire synthetic dataset, we simulate a monocular perspective camera setup with a 640×480 pixel resolution, a focal length of 655 pixels, and no radial and tangential distortion.

We generate a set of random 2D lines, each being at least 70 pixels long and defined initially by two endpoints on the image plane. These lines are given in the camera frame $\{C\}$. We then express these lines in metric coordinates, and add a random depth from 1 to 3 meters to each endpoint. Finally we express these 3D points in the world frame $\{W\}$ by applying the transform $\mathbf{Tr}_{CW} = [\mathbf{R}_{CW}(\rho, \theta, \psi)|\mathbf{T}_{CW}]$ with ρ, θ, ψ being random rotations between 0 rad and 2π rad, and $\mathbf{T}_{CW}$ being a random translation between 0 to 5m around the camera optical center.

We evaluate the robustness of our method to 2D and 3D noises. A normally distributed noise is added on both 2D endpoints of the lines so that each line endpoint is displaced of a given distance σ in pixels. This modifies both the position and direction of the line on the image plane. We add a gaussian 3D noise to the endpoints of the 3D lines so that the displacement of each 3D endpoint caused by the noise has a σ_{3D}mm norm.

Unless otherwise specified, all tests are run with 20 2D-3D correspondences, and we run 2000 tests per noise step. Our formulation requires at least three pairs of lines to work, but is compatible with a higher number. For this reason, we evaluate the impact of the number of lines on our algorithms.

We refer to different metrics to evaluate our algorithms. Since the vertical direction is assumed known, the rotation error refers to the absolute angular difference between the estimated angle ψ_{est} and the initially applied angle ψ. The translation error is the Euclidean distance between the estimated location of the camera and the real camera location divided by the norm of the original translation vector, and is expressed in percent.

In the synthetic dataset, 2D and 3D lines are expressed from points. For this reason, we can express the reprojection error as the mean Euclidean distance between all 2D points in pixel coordinates and the corresponding 3D points projected on the image plane using the estimated pose.

For the pairing algorithm, we include the recall rate, i.e. the number of inliers recovered over the total number of inliers. We also show the precision rate, i.e. the number of inliers recovered over the number of pairs recovered. Both these metrics are evaluated with an increasing outliers rate.

b) Linear PnL with Plücker coordinates formulation: We begin our tests with the pose estimation method. We present the robustness to 2D and 3D gaussian noises in Fig. 2, and we evaluate the impact of the number of lines as input in Fig. 3.

We compare both our algorithms with the ASPnL algorithm from [6], NPnLUpL and NPnLUpC from [19]. While the two latter are designed for multiview pose estimation, they accept a single view configuration as input. ASPnL estimates the 6 DoF pose, and both our algorithm and NPnLUpL consider a 4 DoF problem. For a fair comparison between these algorithms, we introduce a 0.5° constant noise on one of the two rotations of the *up-vector*, simulating this way a common noise from a low-cost orientation sensor.

In term of rotational error, results in Fig. 2 show that when facing 2D noise, both our algorithm perform similarly. With noise on the *up-vector*, we outperform NPnLUpL and NPnLUpC for small 2D noises, but our methods tend to similar results facing heavy 2D noises. All these four algorithms outperform ASPnL when confronting a moderate to high 2D noise. Regarding small 3D noise, our two algorithms outperform ASPnL, NPnLUpC and NPnLUpL. From 100mm to higher noise, our VPnL_LS is outperformed by NPnLUpC and NPnLUpL, but VPnL_GN competes with the two latter.

Regarding translation error, ASPnL performs better for small 2D and 3D noise, because not suffering from errors on the *up-vector*, but for medium to high noise, our VPnL_GN algorithm outperforms the others, and greatly improves the translation error when facing 3D noise.

Fig. 3 presents how the number of lines impacts the pose estimation in presence of noisy 2D or 3D lines. Here, 2D noise on line endpoints is set to 10 pixels and 3D noise is set to 100mm. Noise on the *up-vector* is set to 0.5 °. We vary the number of lines from 4 to 40. The result shows that our algorithms VPnL_LS and VPnL_GN systematically outperform all other methods, independently of the dataset size. Note here that NPnLUpC recovers the rotation with a similar accuracy than our two algorithms, but as a median translation error above 100% in our test case, for any dataset size we tested.

We also show that in presence of 2D noise, all methods tend to perform only slightly better when using more than 25 lines, while increasing the computational complexity. In case of 3D noise, we observe a similar phenomenon, apart from ASPnL whose performances decrease when using more than 17 lines.

c) RANSAC2 as outliers removal module: We present here the results of our RANSAC2 algorithm used as outliers

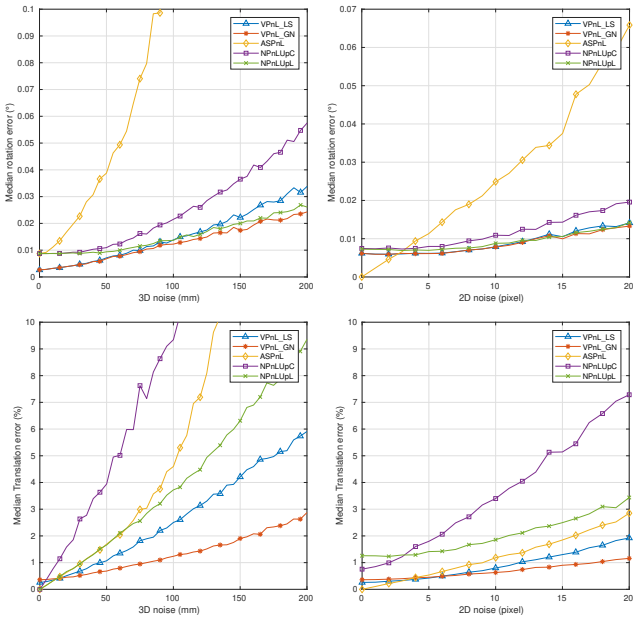


Fig. 2: Rotation and translation errors in case of 2D and 3D noises. VPnL_LS is our PnL formulation with a known vertical direction and a linear least squares solver only. VPnL_GN is the same method with Gauss-Newton optimization on the complete pose. ASPnL refers to [6] method, and NPnLUpL is the PnL linear formulation of [19] workin on a single-camera setup. Left column evaluates robustness against 2D noise, right column evaluates robustness against 3D noise.

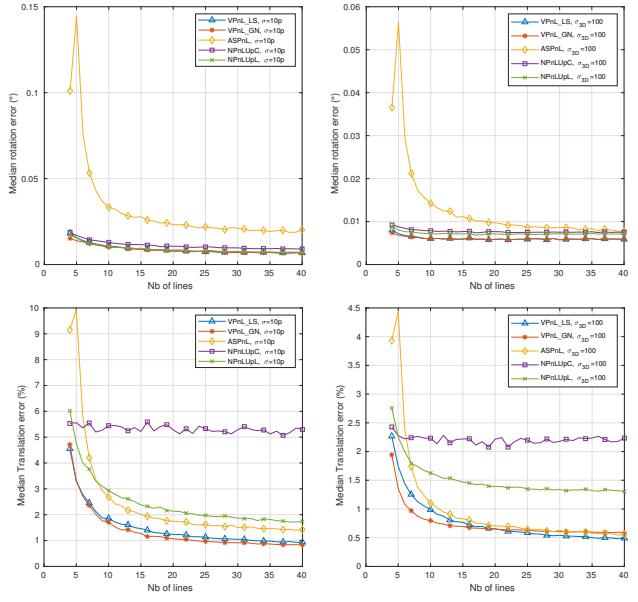


Fig. 3: Impact of the number of lines on the rotation and translation errors for 2D and 3D noise. Comparison between ASPnL, VPnL_LS, VPnL_GN, NPnLUpL and NPnLUpC. Left column evaluates the impact of the number of lines with 2D line endpoints having a 10 pixel noise. Right columns evaluates the impact of the number of lines with a 100mm 3D noise on all 3D line endpoints.

removal module. We evaluate the recall and precision rates of this outliers rejection module, as well as the runtime.

We combine our RANSAC2 algorithm with both our methods VPnL_LS and VPnL_GN to evaluate the overall runtime. We compare our outliers removal algorithm to RANSAC3, RANSAC4 and RLPNL_Enull from [6]. RANSAC3 and RANSAC4 propose a combined outliers rejection and pose estimation scheme. RANSAC3 is P3L-based, and RANSAC4 is ASPnL-based. RLPnL_Enull is a notably outlier-resistant pose estimation algorithm.

For these tests, we imposed a 0.5° noise on the *up-vector* used in our methods, and used a set of 40 lines with a fixed 5 pixels 2D noise and 50mm for the 3D noise. We measured the runtime for each method for a range of outliers from 0% to 60%, and run 2000 tests for each outlier step.

We finally computed the mean runtime over the entire outlier range, 2D and 3D noise for each method, since preliminary results showed that for 40 lines, all methods tend to similar runtimes independently of the outlier rate. For each outlier step, we also computed the mean recall and precision rates. All our runtime results are mesured on a laptop with an Intel Core i5-6440HQ CPU running at 2.6 GHz with a single threaded matlab R2017a script, on an Ubuntu 18.04 operating system.

Runtime results in table I show that even with our rotation and translation optimization enabled, our RANSAC2 algorithm competes with the non-RANSAC RLPnL_Enull method with a 3.2 ms runtime, while our RANSAC2+VPnL_LS is more than 10 times faster with a 0.21 ms runtime. RANSAC3 and RANSAC4 have a respective runtime of 341.8 ms and 1,666.7 ms, which is significantly slower than our two methods. For a fair comparison, we forced the number or iterations for RANSAC3, RANSAC4 and RANSAC2 to be 10 times the number of pairs of lines set as input, i.e. 400 iterations here.

RANSAC3 and RANSAC4 find solutions to the PnL problem in the roots of an 8^{th} degree polynomial, which is computationally more expensive than the linear formulation used in both our methods and in RLPnL_Enull. This could explain the important runtime difference.

The outliers removal module in RLPnL_Enull relies in the analysis of the nullspace of a homogeneous linear system and a Gauss-Newton scheme to find an optimal solution, while our RANSAC2 algorithm simply relies on a linear least squares method to solve a 4-lines linear system in a RANSAC scheme. This explains the runtime difference of the two methods, and also why our RANSAC2 coupled with our Gauss-Newton optimization tends to the same runtime than RLPnL_Enull.

TABLE I: Mean runtime for combined outliers rejection and pose estimation methods. For the RANSAC-based methods, the maximal number of iterations is fixed to 400. Results are calculated for 2000 tests per outlier step.

Method	Runtime(ms)
RANSAC2 (ours) + VPnL_LS	0.21
RANSAC2 (ours) + VPnL_GN	3.25
RLPnL_Enull	2.90
RANSAC3	341.80
RANSAC4	1,666.7

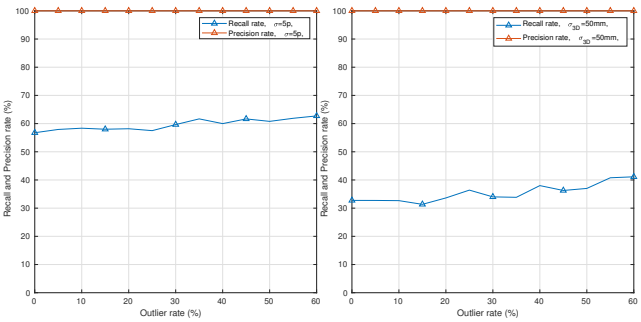


Fig. 4: Evaluation of RANSAC2 as outliers removal module. Recall and precision rates with 5 pixels 2D noise and 50mm 3D noise. Constant 0.5° noise on the *up-vector*. 2000 tests per outlier step.

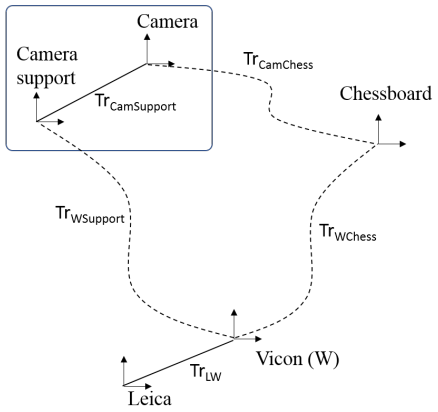


Fig. 5: Experimental setup for our dataset. The camera and camera-support are rigidly attached, but only the camera support pose is initially known in the *Vicon* frame. Ground truth for the camera pose is provided by a calibration step.

Fig. 4 shows the recall and precision rate of our RANSAC2 algorithm. We observe a mean recall rate of about 65% in the 2D noise case, and 45% in the 3D noise case. The mean precision rate is 100% in both cases, independently of the outlier rate.

We observe here that while our algorithm can be improved in term of recall rate, we obtain a perfect precision. This means that all inlier propositions returned by our algorithm are valid, and thus lead to a pose estimation with an accuracy that is shown in Fig. 2 and Fig. 3.

B. Experiment on real data

a) Experimental setup: To validate our methods with real data, we created an experimental dataset composed of images of an indoor scaffolding structure and corresponding camera poses in a known 3D environment. The 3D environment is represented in the form of point cloud acquired with a *Leica ScanStation C10* LiDAR. Ground truth for the poses is provided with a millimetric accuracy by a *Vicon* motion capture system, calibrated with the 3D point cloud and the camera. 3D and 2D lines are extracted respectively from the 3D point cloud and the images of our dataset. Because

we perform the extraction process manually, we limited our experiment to 6 images acquired from various locations and orientations.

To acquire images with 6 DoF poses, we mounted a calibrated 640×480 pixel *Logitech Quickcam 4000* camera on a rigid metallic frame (see Fig.6 (a)), denoted camera support. We moved and tilted this hand-held camera-setup in the laboratory, around the scaffolding structure seen in Fig.6(d) and Fig.7. Note here that the camera support to camera transform $\mathbf{Tr_{CamSupport}}$, as seen in Fig. 5, is initially unknown.

b) Calibration: To calibrate the camera with respect to the *Vicon* system, we acquire images of a chessboard equipped with *Vicon* markers (see Fig.6 (b)) with our camera. We obtain here $\mathbf{Tr_{CamChess}}$ and $\mathbf{Tr_{WChess}}$.

For multiple images and associated poses, we can now recompose $\mathbf{Tr_{CamSupport}}$ with:

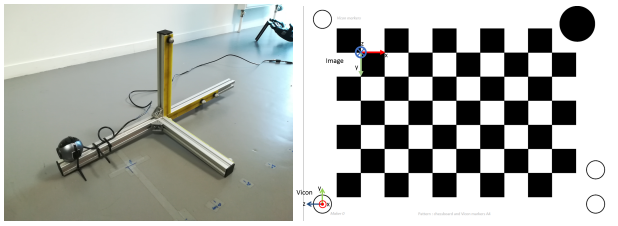
$$\mathbf{Tr_{CamSupport}} = \mathbf{Tr_{CamChess}}.\mathbf{Tr_{WChess}}^{-1}.\mathbf{Tr_{WSupport}}. \quad (18)$$

We estimate the rigid transform between the *Vicon* frame $\{W\}$ and the *Leica* frame $\{L\}$ using known point features available in the two frames. We placed several *Vicon* markers into the laboratory, and manually selected their correspondences in the point cloud. We obtained two sets of 23 3D points $\mathbf{A}_i$ and $\mathbf{B}_i$ $(i = 1..n)$ respectively in *Vicon* and *Leica* frame, linked by a rigid transform $\mathbf{Tr_{LW}}$ with

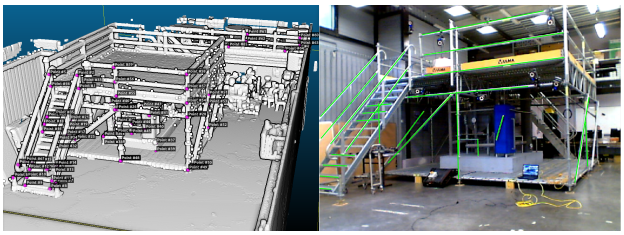
$$\mathbf{Tr_{LW}}.\mathbf{A} = \mathbf{B} \quad (19)$$

Having calibrated the complete system, we know the complete pose of the camera in the world frame.

c) Pose estimation tests and results: For each of the 6 images of our dataset, we estimated the camera pose using manually matched 2D lines from the images, and 3D lignes for the point cloud.



(a) (b)



(c) (d)

Fig. 6: a. camera setup equipped with *Vicon* markers, b. calibration target template, c. point cloud with extracted 3D line endpoints, d. image with extracted 2D lines.

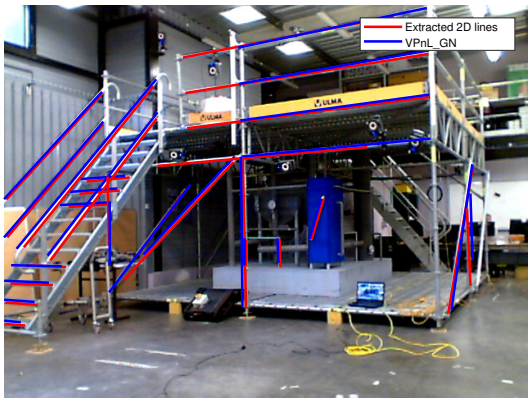


Fig. 7: Reprojection of 3D lines extracted from the *Leica* point cloud. The pose parameters are estimated with VPnL_GN only. 2D and 3D lines are paired manually.

Results are shown in table II. Regarding the rotation error, result show that ASPnL has the lowest accuracy with a median error above 5°. All methods perform similarly with an accuracy around 0.2°. This result can be explained, since ASPnL recovers the complete rotation where all the other method considered benefit the *up-vector* to only recover one rotation.

TABLE II: Mean, median and standard deviation for the translation error (%) and rotation error (°) over 6 poses. 2D and 3D lines are manually extracted and paired. The camera is displaced in a 5m radius around the *Vicon* frame origin.

	Rotation error (°)			Translation error (%)		
Method	**Med.**	**Mean**	**Std. dev.**	**Med.**	**Mean**	**Std. dev.**
VPnL_LS	0.247	0.207	0.087	2.728	3.064	1.459
VPnL_GN	0.276	0.294	0.082	2.786	3.109	1.481
NPnLUpL	0.249	0.267	0.187	3.241	3.894	2.760
NPnLUpC	0.278	0.298	0.108	5.281	27.27	50.47
ASPnL	5.494	13.64	21.72	6.767	6.889	2.702

Regarding the translation error, VPnL_LS achieves the best results with a 2.72% median translation error, followed by VPnL_GN with 2.78%. NPnLUpL and NPnLUpC are designed for a multi-camera system, which explains a greater translation error, respectively about 3.24% and 5.28%. ASPnL is here the less accurate, with 5.76 % translation error. Reprojection with the parameters estimated with VPnL_GN are shown in Fig. 7.

d) Outliers removal module tests and results: To test the robustness against outliers with real data, we select a pose from the 6 poses available in our dataset and randomly introduce 40% of outliers in the line correspondences. For a given test, each algorithm is given the same data as input. The selected pose has 25 line correspondences and the camera is placed 4.3m far from the *Vicon* frame origin. We evaluate the rotation and translation errors, as well as the runtime, for RANSAC3, RANSAC4, RLPnL_Enull, RANSAC2+VPnL_LS and RANSAC2+VPnL_GN. Results shown in table III are median values for 1000 tests. The

TABLE III: Outliers rejection and pose estimation test on real data. We show the median, mean and standard deviation for the translation error (%), rotation error (°) and runtime (ms). The camera is placed 4.3m far from the *Vicon* frame origin and tilted. For each test, 40% of the 26 line correspondences are shuffled to introduce outliers.

	Rotation error (°)			**Translation error (%)**			**Runtime (ms)**		
Method	**Med.**	**Mean**	**Std. dev.**	**Med.**	**Mean**	**Std. dev.**	**Med.**	**Mean**	**Std. dev.**
VPnL_LS	0.009	0.016	0.353	12.49	16.83	45.52	1.1	2.2	0.95
VPnL_GN	0.005	0.006	0.003	12.52	16.88	46.80	4.4	3.8	1.87
RLPnL_Enull	1.48	1.454	0.917	106.6	113.4	58.10	1.8	2.4	13.91
RANSAC3	1.90	1.65	0.868	211.8	886.1	3118.6	334.4	262.8	23.67
RANSAC4	1.59	1.47	0.903	166.3	192.3	386.1	2109.6	1465.8	159.66

maximum number of iterations for the RANSAC algorithms is fixed to 500.

The result shows that when coupled, our outliers rejection and pose estimation algorithms successfully recover a valid pose in most of the cases, where RANSAC3+P3L and RANSAC4+ASPnL fail. We note that RLPnL_Enull performs better than RANSAC3 and RANSAC4 methods for a much smaller runtime, but is still greatly outperformed by our methods in term accuracy. We observe a runtime respectively two and three orders of magnitude faster when comparing VPnL_GN to RANSAC3 and RANSAC4, when RLPnL_Enull has a runtime comparable to both our methods.

Note that for this pose, VPnL_GN does not perform better than VPnL_LS, because we are in presence of low 2D and 3D noises combined with noise on the *up-vector*. This is consistent with the simulations results shown in Fig. 2.

Our experiment validates the simulation results, and proves the advantages of a known vertical direction for outliers rejection and pose estimation, in term of accuracy and runtime.

V. Conclusion

In this paper, we first present a line-based pose estimation scheme with known vertical direction. Our algorithm follows a Plücker coordinates formulation, and a unique solution is estimated with a linear least squares solver. We derive a rotation and translation optimization scheme based on the Gauss-Newton algorithm. We also present a pairing and outliers removal module based on the RANSAC2 algorithm, relying only on the rotation estimation.

Our PnL formulation is well suited for small and noisy datasets, especially in mobile robotics, where the vertical direction is often known from integrated sensors (IMU). However, the use of a linear least squares solver involves a low robustness to outliers. Our outliers removal module achieves a 100% precision rate, up to 60% of outliers, in presence of 2D and 3D noise. Its very low runtime enables its use for real-time robotic applications or as a pairing module.

Acknowledgment

The authors would like to thank Romain Rossi, Pierre Merriaux and Sophie Ladet for helpful discussions.

References

[1] 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*, vol. 11, no. 12, pp. 1265–1278, 1989.

[2] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[3] J. A. Hesch and S. I. Roumeliotis, "A direct least-squares (dls) method for pnp," in *IEEE International Conference on Computer Vision*, 2011, pp. 383–390.

[4] 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*, pp. 501–508.

[5] B. Přibyl, P. Zemčík, and M. Čadík, "Camera pose estimation from lines using plucker coordinates," *arXiv:1608.02824*, 2016.

[6] C. Xu, L. Zhang, L. Cheng, and R. Koch, "Pose estimation from line correspondences: A complete analysis and a series of solutions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1209–1222, 2017.

[7] 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*, vol. 24, no. 6, pp. 381–395, 1981.

[8] C. Albl, Z. Kukelova, and T. Pajdla, "Rolling shutter absolute pose problem with known vertical direction," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3355–3363.

[9] G. H. Lee, M. Pollefeys, and F. Fraundorfer, "Relative pose estimation for a multi-camera system with known vertical direction," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2014, pp. 540–547.

[10] N. Horanyi and Z. Kato, "Generalized pose estimation from line correspondences with known vertical direction," in *International Conference on 3D Vision (3DV)*, 2017, pp. 244–253.

[11] O. Saurer, P. Vasseur, R. Boutteau, C. Demonceaux, M. Pollefeys, and F. Fraundorfer, "Homography based egomotion estimation with a common direction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 2, pp. 327–341, 2017.

[12] A. Bartoli and P. Sturm, "The 3d line motion matrix and alignment of line reconstructions," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

[13] H. H. Chen, "Pose determination from line-to-plane correspondences: existence condition and closed-form solutions," in *IEEE International Conference on Computer Vision*, 1990, pp. 374–378.

[14] A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 578–589, 2003.

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

[16] X. Zhang, Z. Zhang, Y. Li, X. Zhu, Q. Yu, and J. Ou, "Robust camera pose estimation from unknown or known line correspondences," *Applied optics*, vol. 51, no. 7, pp. 936–948, 2012.

[17] B. Přibyl, P. Zemčík, and M. Čadík, "Absolute pose estimation from line correspondences using direct linear transformation," *Computer Vision and Image Understanding*, 2017.

[18] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[19] N. Horanyi and Z. Kato, "Multiview absolute pose using 3d–2d perspective line correspondences and vertical direction," in *IEEE International Conference on Computer Vision Workshop*, 2017, pp. 2472–2480.

[20] W. F. Mascarenhas, "The divergence of the bfgs and gauss newton methods," *Mathematical Programming*, vol. 147, no. 1-2, pp. 253–276, 2014.