Multiview Absolute Pose Using 3D – 2D Perspective Line Correspondences and Vertical Direction

Nora Horanyi\(^1\) and Zoltan Kato\(^{1,2}\)

\(^1\) Institute of Informatics, University of Szeged, P.O. Box 652, H-6701, Szeged, Hungary
\(^2\) Department of Mathematics and Informatics, J. Selye University, Komarno, Slovakia,

horanyi@inf.u-szeged.hu, kato@inf.u-szeged.hu

Abstract

In this paper, we address the problem of estimating the absolute pose of a multiview calibrated perspective camera system from 3D - 2D line correspondences. We assume, that the vertical direction is known, which is often the case when the camera system is coupled with an IMU sensor, but it can also be obtained from vanishing points constructed in the images. Herein, we propose two solutions, both can be used as a minimal solver as well as a least squares solver without reformulation. The first solution consists of a single linear system of equations, while the second solution yields a polynomial equation of degree three in one variable and one systems of linear equations which can be efficiently solved in closed-form. The proposed algorithms have been evaluated on various synthetic datasets as well as on real data. Experimental results confirm state of the art performance both in terms of quality and computing time.

1. Introduction

Absolute pose estimation consists in determining the position and orientation of a camera with respect to a 3D world coordinate frame. It is a fundamental building block in various computer vision applications, such as visual odometry, simultaneous localization and mapping (SLAM), image-based localization and navigation, augmented reality. The problem has been extensively studied yielding various formulations and solutions. Most of the approaches focus on the single camera pose estimation using point correspondences. However, modern applications, especially in vision-based localization and navigation for robotics and autonomous vehicles, it is often desirable to use multi-camera systems which covers large field of views and provides direct 3D measurements. This problem is known as non-perspective absolute pose estimation as such a camera system may be modeled as a virtual camera with many projection centers.

The absolute pose estimation of a perspective camera from \(n\) 2D-3D point correspondences is known in the literature as the Perspective n Point (PnP) problem, which has been widely studied in the last few decades \([6, 13, 14, 8]\). Various solutions have been developed for both large \(n\) as well as for the \(n = 3\) minimal case (see [8] for a recent overview).

The use of line correspondences, known as the Perspective n Line (PnP) problem, has also been investigated in the last two decades, yielding robust and efficient solutions (see [24] for a detailed overview). Mirzae et al. \([17]\) construct a polynomial system of equations from line correspondences to solve for the camera orientation. The system consists of three \(5^{th}\) order equations and one cubic equation with four unknowns, which yields 40 candidate solutions. They also develop an algorithm for perspective pose estimation from three or more line correspondences \([16]\), where the problem is formulated as a non-linear least-squares and solved as an eigenvalue problem using the Macaulay matrix without a need for initialization. Unfortunately, this algorithm yields 27 solutions, which makes it difficult to identify the correct solution in practical applications.

The minimal case of \(n = 3\) line correspondences is particularly important as its solution is the basis for dealing with the general PnP problem. It has been shown in \([5]\), that P3L leads to an 8\(^{th}\) order polynomial, which is higher than the 4\(^{th}\) order polynomial of a P3P problem. While the use of point and line correspondences are widespread, there are pose estimation methods relying on other type of correspondences, e.g. set of regions [21, 20] or silhouettes. However, such approaches are typically computationally more expensive hence they cannot be used as real-time solvers.

Recently, due to increasing popularity of multi-camera systems in e.g. autonomous driving [11] and UAVs, the problem of multiview perspective pose estimation has been
addressed. Solutions to the PnP or PnL problem cover only single-view perspective cameras, hence new methods are needed to efficiently deal with the multiview PnP (NPnP) problem [3, 8, 11, 12].

In this work, we deal with multiview absolute pose estimation from 3D–2D perspective line correspondences (also known as the NPnP problem) with known vertical direction. While several point-based methods exist [3, 8], little work has been done on using line correspondences. One notable work is the minimal NP3L solver of Lee [10], which deals with 6 DOF pose parameter estimation. Today, the vast majority of modern cameras, smart phones, UAVs, and camera mounted mobile platforms are equipped with cheap and precise inertial measurement unit (IMU). Such devices provide the vertical direction from which one can calculate 2 rotation angles, thus reducing the free parameters from 6 to 4. The accuracy of the up-vector is typically between 0.5° – 0.1° [1]. While robust minimal solutions based on point correspondences exist [1, 9, 12], none of these methods work for line correspondences.

In this paper, we propose two new solutions to the NPnP problem with known vertical direction. Both algorithms can be used as a minimal NP3L solver with 3 line correspondences suitable for hypothesis testing like RANSAC. Furthermore, the same algorithms can be used without re-formulation for n > 3 lines as well as for classical single-view PnP problems. The performance and robustness of the proposed methods have been evaluated on large synthetic datasets as well as on real data.

2. Perspective Projection of Lines

Given a calibrated camera P and 3D lines Li in the world coordinate frame, the projection of the lines are 2D lines li in the image plane. The perspective camera matrix P = K[R|t] consists of the internal calibration matrix K and the camera pose [R|t] w.r.t. the world coordinate frame. A homogeneous 3D point X is mapped by P into a homogeneous 2D image point x’ as [7]

\[ x' \cong PX = K[R|t]X, \]

where ‘\cong’ denotes the equivalence of homogeneous coordinates, i.e. equality up to a non-zero scale factor. Since we assume a calibrated camera, we can multiply both sides of (1) by K⁻¹ and work with the equivalent normalized image

\[ x = K^{-1}x' \cong K^{-1}PX = [R|t]X. \]  

The above equation is the starting point of absolute perspective pose estimation [8, 14, 13, 9]: given a set of 3D–2D point correspondences (xi ↔ Xi), one can recover the 3D rigid body transformation (R, t) : W → C acting between the world coordinate frame W and the camera coordinate frame C.

Unlike points, 3D lines may have various representations in the projective space [18, 2]. Plücker line coordinates are popular as they are complete (i.e. every 3D line can be represented) and allow for a linear projection model similar to (2) [7, 2, 10, 19, 25, 15]. However, Plücker coordinates are not minimal because a 3D line has 4 degrees of freedom but its Plücker coordinate is a homogeneous 6-vector. Furthermore, transforming a Plücker line between coordinate frames using a standard homogeneous 4 × 4 rigid motion matrix

\[ M = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \]

is indirect, i.e. two points of the line have to be transformed and then form their Plücker coordinates again. In [2], a 6×6 3D line motion matrix representation is proposed for the transformation, which allows for a direct and linear transformation at the price of a larger matrix. This approach is also used in [10, 19] for absolute pose estimation.

Herein, 3D lines are represented as L = (V, X), where V is the unit direction vector of the line and X is a point on the line (see Fig. 1) [22]. The projection of L in a central perspective camera is a line l in the image plane, which can also be represented as l = (v, x). Note that the point x is not necessarily the image of the 3D point X! Both L and l lie on the projection plane π passing through the camera projection center C. The unit normal to the plane π in the camera coordinate system C is denoted by n, which can be computed from the image line l = (v, x) as n = (v × x)/∥v × x∥. Since L lies also on π, its direction vector V is perpendicular to n. Hence

\[ n^T RV = n^T V^C = 0, \]

where R is the rotation matrix from the world W to the camera C frame and V^C denotes the unit direction vector of L in the camera coordinate frame. Furthermore, the vector from the camera center C to the point X on line L is also lying on π, thus it is also perpendicular to n:

\[ n^T (RX + t) = n^T X^C = 0, \]

where t is the translation from the world W to the camera C frame and X^C denotes the point X on L in the camera coordinate frame.

3. Multi-view Projection

Let us now investigate the case when the 3D lines are viewed by N perspective cameras. We assume that the cameras are fully calibrated, i.e. their intrinsics K^i as well as their relative pose (R^i, t^i) : C^i → C with respect to a common camera coordinate frame C are known. The common coordinate frame C is often attached to one of the cameras (e.g. the left camera in a stereo setup), but a multi-camera system may have a coordinate frame detached from
the cameras (e.g. the centroid of the mounting frame, or the
IMU device). Therefore the absolute pose of the camera
system \((R, t)\) is defined as the rigid transformation
between \(W\) and \(C\), while individual camera frames \(C_i\)
are related to the world coordinate frame via the sequence of
rigid transformations

\[
\forall i : M_{W \rightarrow i} = \begin{bmatrix} R_i & t^i \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix}^T.
\]

In fact, the whole camera system can be regarded as a gen-
eralized non-perspective camera with \(N\) projection center \([3]\).
In such a non-central camera, each 3D line \(L\) has up to \(N\)
images \(l^i, i = 1, \ldots, N\), where \(N\) is the number of cameras
(or projection centers). Given a pair of corresponding image
lines \(\{l^i, l^j\}\) and their projection plane normals \(\{n^i, n^j\}\),
the unit direction vector \(V^C\) of \(L\) can be expressed in the
camera frame \(C\) as

\[
V^C = \frac{R^{T} n^i \times R^{jT} n^j}{\|R^{T} n^i \times R^{jT} n^j\|},
\]

which yields the following relation

\[
V^C = RV \Rightarrow (RV) \times (R^{T} n^i \times R^{jT} n^j) = 0 \quad (8)
\]

Thus a natural approach to our absolute pose estimation
problem would be to reconstruct 3D line directions in the
camera frame using e.g. (7) and then solving (8) for \(R\).
While this is a very attractive, simple, and geometrically
intuitive approach, the quality of such a pose estimate
would be critically dependent on the reconstruction ac-
curacy, which is known to be quite poor for practically
important setups like narrow baseline stereo \([5]\). In general,
(8) becomes numerically unstable whenever \(R^{T} n^i\) and
\(R^{jT} n^j\) are nearly parallel, which is often the case for
narrow baseline and near-parallel principal axes. Furthermore,
having a noisy estimate of the normals would severely
deteriorate the accuracy of their cross product introducing large
ears in a system of equations constructed from (8).

Essentially (8) states that \(V^C\) is perpendicular to both \(n^i\)
and \(n^j\), yielding the multi-view form of (4):

\[
\forall i \text{ where } L \text{ is visible: } n^{iT} R^i V^C = n^{iT} R^i R V = 0 \quad (9)
\]

While this equation is mathematically equivalent to (8), it
is numerically more favorable as it provides separate
equations for each normal thus avoiding multiplication of noisy
\(n^i\) measurements. Similarly, (5) can be written for the
multi-camera case as:

\[
\forall i \text{ where } L \text{ is visible: } n^{iT} (R X^C + t^i) =
\]

\[
n^{iT} (R^i (R X + t) + t^i) = 0 \quad (10)
\]

### 3.1. Using the Vertical Direction

Let us now have a closer look at the parameterization of \(R\) assuming that the vertical direction is available. This
knowledge can be obtained from e.g. an inertial measure-
ment unit (IMU), which mainly consists of accelerometers
capable to measure the earths gravity vector. Alternatively,
one can obtain the same information from a calibrated im-
age by detecting a vanishing point (in a man-made envi-
ronment, we can get the vertical vanishing point, but the
knowledge of any other direction would also do) \([7]\). In
the following, we discuss the mathematical representation
of this information in \(R\).

Assuming that the camera coordinate system is a stan-
dard right-handed system with the \(X\) axis pointing up (see
Fig. 1), the coordinates of the world vector \((1, 0, 0)^T\) are
known in the camera coordinate frame \(C\). Given this up-
vector, we can compute rotation \(R_u\) around \(Y\) and \(Z\) axes,
which aligns the world \(X\) axis with the camera \(X\) axis:

\[
R_u = R_Z(\gamma) R_Y(\beta) =
\begin{bmatrix}
\cos(\gamma) & -\sin(\gamma) & 0 \\
\sin(\gamma) & \cos(\gamma) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos(\beta) & 0 & \sin(\beta) \\
0 & 1 & 0 \\
-\sin(\beta) & 0 & \cos(\beta)
\end{bmatrix}
\]

The only unknown parameter in the rotation matrix \(R\) is the
rotation \(R_X(\alpha)\) around the vertical \(X\) axis:

\[
R_X(\alpha) =
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\alpha) & -\sin(\alpha) \\
0 & \sin(\alpha) & \cos(\alpha)
\end{bmatrix}
\]

thus the \(W \rightarrow C\) rotation \(R\) has the following form:

\[
R = R_u R_X(\alpha)
\]

With this parameterization, the equations (9) and (10) have
the form for all camera \(i\) where \(L\) is visible:

\[
n^{iT} R^i R_u R_X(\alpha) V = 0 \quad (14)
\]

\[
n^{iT} (R^i (R_u R_X(\alpha ) X + t) + t^i) = 0 \quad (15)
\]
4. Efficient Solutions

We aim to compute $R_X(\alpha)$ and $t$ using the equations in (14) - (15). We have 4 unknowns: the rotation angle $\alpha$ and the translation components $t_1, t_2, t_3$. Although each 3D-2D line correspondence $L \leftrightarrow l$ provides 2 equations, only one contains $t$. Therefore we need at least 3 line correspondences. In the following, we propose two solutions. Both of them use the fact that the images are normalized (i.e. image points are multiplied by the inverse of $K'$ as in (2)); the relative pose $(R', t')$ of each camera is known w.r.t. the common camera frame $C'$; and the vertical direction is known, i.e. the rotation matrix $R_v$ of (11) is available.

4.1. Linear Solution: NPlnLpL

The equations (14) - (15) are linear in $t$, but $R_X(\alpha)$ is defined in terms of $\cos(\alpha)$ and $\sin(\alpha)$. Letting these trigonometric functions of $\alpha$ to be two separate unknowns $c$ and $s$ [14, 24, 26], respectively, one can linearize (14) - (15) by substituting

$$\begin{pmatrix} R' R_v \end{pmatrix} R_X(\alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 1 & 0 & 0 \\ r_{21} & r_{22} & r_{23} & 0 & c & -s \\ r_{31} & r_{32} & r_{33} & 0 & s & c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \begin{bmatrix} a_{1q}^2 + b_{1q}^2 + c_{1q} = 0, \quad i = 1 \ldots n, \end{bmatrix}$$

into (14) - (15). Stacking these pairs of equations for $n$ correspondences, we get $2n$ homogeneous linear equations with unknowns $p = (c, s, t_1, t_2, t_3, 1)^T$. Hence we have to solve an $Ap = 0$ system of equations in the least squares sense, which is straightforward using SVD of $A$. Note that the elements of the $2n \times 6$ matrix $A$ are expressed in terms of $n'$, $R' R_v$, $V$, $X$, and $t'$ using (14) and (15). Since $c$ and $s$ are estimated as separate unknowns, they may not satisfy the trigonometric constraint $c^2 + s^2 = 1$. Thus they should be normalized before constructing $R_X(\alpha)$:

$$\cos(\alpha) = \frac{c}{\sqrt{c^2 + s^2}} \quad \text{and} \quad \sin(\alpha) = \frac{s}{\sqrt{c^2 + s^2}} \quad (17)$$

At the price of higher computational complexity, a somewhat more sophisticated normalization involves an additional 3D registration step [14, 24, 26], which aligns the 3D world $\{X_i\}$ and camera $\{X'_i\}$ point sets using a standard 3D registration scheme [23].

A major drawback of this linear solution is that orthonormality constraint on $R_X(\alpha)$ has been ignored, thus the solution can be quite far from a rigid body transformation for noisy input data. In spite of this, experiments show that our linear solver represents a good tradeoff between accuracy and computing time, yielding quite stable pose estimates under moderate noise levels.

4.2. Cubic Polynomial Solution: NPlnLpC

Another way to eliminate $\cos(\alpha)$ and $\sin(\alpha)$ is to use the substitution $q = \tan(\alpha/2)$ [9, 1], for which $\cos(\alpha) = (1 - q^2)/(1 + q^2)$ and $\sin(\alpha) = 2q/(1 + q^2)$. Therefore

$$\begin{pmatrix} (1 + q^2) R_X(q) \end{pmatrix} = \begin{bmatrix} 1 + q^2 & 0 & 0 \\ 0 & 1 - q^2 & -2q \\ 0 & 2q & 1 - q^2 \end{bmatrix}$$

Substituting $R_X(q)$ into (14) yields a quadratic equation in the single unknown $q$. Since we have $n \geq 3$ line correspondences, we obtain $n$ quadratic equations in $q$:

$$a_i q^2 + b_i q + c_i = 0, \quad i = 1 \ldots n, \quad (19)$$

where the coefficients $a_i, b_i, c_i$ are computed in terms of $n', R' R_v$, and $V$ using (14). We will solve this nonlinear system of equations in terms of least square residual. The squared error of the equations is a quartic function in $q$

$$\begin{pmatrix} \sum_{i=1}^{n} \begin{pmatrix} a_i q^4 + 2a_i b_i q^3 + (2a_i c_i + b_i^2) q^2 + 2b_i c_i q + c_i^2 \end{pmatrix} \end{pmatrix},$$

whose minima is found by computing the roots of its derivative

$$\begin{pmatrix} \sum_{i=1}^{n} \begin{pmatrix} 4a_i q^3 + 6a_i b_i q^2 + (4a_i c_i + 2b_i^2) q + 2b_i c_i \end{pmatrix} = 0 \end{pmatrix}$$

Such a third order polynomial equation can be solved in a closed form and results in maximum 3 solutions, at least one of them being real. For each real root, we have to determine $t$ by back substituting each possible $R_X(q)$ into (15), which yields a simple linear system of equations in $t$:

$$n'^T (X' + R't) = 0, \quad (22)$$

with $X' = R' R_v R_X(q) X + t'$. The final solution is selected by checking that lines are in front of the camera system, or simply by evaluating the reprojection error of each solution and selecting the one with minimal error. In the non-perspective case, the reprojection error characterizes the difference between the observed image line $l'$ and the reprojected image line $l'$ for all cameras [22]:

$$\begin{pmatrix} \epsilon = \sum_{i=1}^{N} \sum_{j=1}^{n} \begin{pmatrix} \hat{n}_j^T (A^T B A) \hat{n}_j \end{pmatrix}, \quad \text{with} \quad A = \begin{bmatrix} a_i' \\ b_i' \end{bmatrix}, \quad \text{and} \quad B = \frac{|l_j'|}{3(n_j^T + n_j^T)} \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad (23) \end{pmatrix}$$

where $|l_j'|$ denotes the length of the image line with the 2D homogeneous endpoints $a_i' = (a_{i1}' a_{i2}' 1)^T$ and $b_i' = (b_{i1}' b_{i2}' 1)^T$; $\hat{n}_j'$ is the reprojection plane normal in camera $i$ computed from the corresponding 3D line $L_j = (X_j, V_j)$ as

$$\hat{n}_j' = (R' (R' X_j + t) + t') \times (R' R V_j). \quad (24)$$

2745
Figure 2: Comparison of the rotational errors w.r.t. the baseline for 30 lines in case of a single-view and stereo system configurations.

<table>
<thead>
<tr>
<th>n lines</th>
<th>NPnLupL</th>
<th>NPnLupC</th>
<th>RPnL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run time (s)</td>
<td>0.0009</td>
<td>0.0013</td>
<td>0.0088</td>
</tr>
<tr>
<td>3 lines</td>
<td>NP3LupL</td>
<td>NP3LupC</td>
<td>UP3P</td>
</tr>
<tr>
<td>Run time (s)</td>
<td>0.0004</td>
<td>0.0005</td>
<td>0.0063</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the run times of different methods w.r.t number of lines.

The main advantage of this solution is that the trigonometric constraint on $C$ is explicitly taken into account, thus we expect an increased robustness under noisy observations. However, the estimation of $C$ and $t$ is decoupled, which may lead to slightly less accurate solutions as any error in $C$ is directly propagated into the linear system of $t$. Furthermore, computational complexity is slightly higher than the purely linear solver.

Finally, Table 1 shows the typical runtime for all tested methods in case of 3 and $n$ lines.

5. Experimental Results

For the quantitative evaluation of our non-perspective pose estimation algorithm with line correspondences, we generated various benchmark datasets of 3D-2D line pairs. 3D lines were generated by placing three 2D planes in the 3D Euclidean space and about 10 lines were placed on each of these planes, whose size was normalized into a 1m$^3$ cube. Then the planes were placed relative to each other with a random translation of 0—1 unit and rotation of $0^\circ, \ldots, 45^\circ$ around the Z axis and $20^\circ, \ldots, 60^\circ$ around the vertical X axis. This arrangement of the 3D lines was motivated by common urban structural properties, in which environment the real data experiments were performed too.

Figure 3: Upper plot shows the translation errors in case of 3 camera for both $n$ and 3 lines. The plot below shows the mean re-projection errors $e$ of the proposed algorithms for $n$ lines.

Figure 4: Comparison of two possible minimal case scenarios in a 3-camera system. Dashed lines: one 3D line and its corresponding 2D lines from each camera; connected lines: a different 3D–2D line pair for each camera.

The synthetic 2D images of the 3D lines were generated with a camera system being rotated in the range of $-20^\circ, \ldots, 20^\circ$ around all three axis and random displacement of 9 units. All cameras had identical intrinsic param-
eters $K$: focal length $f_x = 800, f_y = 800$, and the principal point $o$ was set to the center of the $1024 \times 768$ image plane. Thus random 2D projections were obtained with the so defined random camera matrix $P = KR[\mathbf{t}]$. Separate datasets were generated for single camera, standard stereo, and multi-camera systems emulated with 3 cameras. In case of the standard stereo setup, where the right camera is only horizontally translated, we used three different baselines of 0.1, 0.8 and 1.5. For the three-camera setup, datasets were generated with random translations corresponding to 0.05, 0.15, and 0.4 baselines and random relative rotation between the cameras around the Y and Z axis in range of $-5^\circ, \ldots, 5^\circ$, and around the X axis $15^\circ, \ldots, 25^\circ$. Our algorithms were implemented in Matlab and run on a standard desktop computer.

The algorithms were done in two scenarios: either all of the 3D–2D line pairs are used (about 30 line pairs per sample - this will be denoted by $n$) or only the minimum number line pairs are used (3 line pairs - this will be denoted by 3). We also compare our results with state of the art methods. Since to the best of our knowledge, there is no prior method for non-perspective pose estimation from line correspondences and known vertical direction, we compare our method with the line-based single view RPNL algorithm [26] of Zhang et al.; the point-based non-perspective UPnP [8] of Kneip et al.; and the line-based non-perspective minimal solver NP3L [10] of Lee.

First, we evaluate the sensitivity of our algorithms with $n$ lines for the baseline in case of the standard stereo setup (parallel optical axes, only horizontal translation between cameras), which is the most challenging configuration as projections planes are nearly parallel for narrow baselines. Fig. 2 shows, that for $n$ lines, the linear solver is more accurate, but overall both methods perform quite well independently of the baseline length, having a median rotation error less than $0.11^\circ$ in all samples.

Next, we compare the performance in the minimal and $n$-line cases. Fig. 3 shows the translation error in case of $n$ lines as well as 3 lines. Obviously, the algorithms perform better for $n$ lines, but overall the estimates are quite accurate. Note also that the cubic solver outperforms the linear one in the minimal case. In Fig. 4, we compare two possible minimal case scenarios in a 3-camera system: 1) one 3D line and its corresponding 2D lines from each camera; 2) a different 3D–2D line pair for each camera. The first case is useful when 3D lines are limited but there is no occlusion. The second scenario corresponds to occlusions, when not all 3D lines are visible from all cameras. The accuracy of our algorithms are not influenced by these differences.

5.1. Robustness

In order to evaluate the sensitivity of our algorithms to line measurement noise, we add random noise to the generated test cases in the following way: The 2D lines are corrupted with additive random noise on one endpoint of the line and the direction vector of the line. The amount of noise is 5% and 8%, meaning that a random number is added to each coordinate up to the specified percentage of the actual coordinate value. This corresponds to a quite high noise rate: $[-20, +20]$ pixels (4 pixels mean and 12 pixels standard deviation) for the 5% case and $[-30, +30]$ pixels (4 pixels mean and 20 pixels standard deviation) for the 8% case.

In Fig. 5, we compared the robustness of the proposed algorithms in a 3-camera system. NPNL performs better than the other two. RPNL is consistently outperformed by our solvers in all cases. Of course, we know the vertical direction, hence RPNL has to solve a more difficult task. However, if an IMU is available, then it is clearly worth to use this vertical information instead of relying on purely visual data.
run our algorithms on the 5% noisy datasets. The comparative results of these experiments are shown in Fig. 6, where we show error plots for the standard stereo setup minimal case, 3-camera system minimal case (compared with NP3L [10]), as well as 3-camera case with n-lines (compared with UPnP [8]. NP3L [10] is consistently outperformed by our methods, while for UPnP [8] our NPnP/LC outperforms it in the noisy cases. Note, however, that in the maximal case when we have 3 cameras this algorithm has more than 150 point pairs to work with (we used 2 points per line).

5.2. Real Data

In Fig. 7 and Fig. 8, we show two real datasets. 2D images were captured with a standard Canon DSLR camera while the 3D point cloud was captured with a Riegl VZ400 Lidar scanner with an angular resolution of 0.05°. Each camera location is shown in the Lidar coordinate system as well as the 3D lines used for pose estimation. The corresponding 3D-2D lines are shown with the same color as the camera. Pose estimation errors are shown in Table 2 in comparison with UPnP [8].

Table 2: Comparison of the maximal rotational and translational error of various methods on the real data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rotation error (deg)</th>
<th>Translation error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 7</td>
<td>0.0177</td>
<td>0.0191</td>
</tr>
<tr>
<td></td>
<td>0.0356</td>
<td>0.0262</td>
</tr>
<tr>
<td>Fig. 8</td>
<td>0.0166</td>
<td>0.0176</td>
</tr>
<tr>
<td></td>
<td>0.0402</td>
<td>0.0319</td>
</tr>
</tbody>
</table>

To have a fair comparison with the other methods, we also added a ±0.5° random noise to the vertical direction (this is a typical noise level of a low quality IMU), and then

Figure 6: Comparison of various configurations and methods w.r.t. varying line numbers and varying noise levels (2D: 5% 2D noise, 2Dv: 5% 2D noise with 0.5° vertical noise, 3D: 5% 3D noise, 3Dv: 5% 3D noise with 0.5° vertical noise, m: median error value). The plot on the top indicates the efficiency of our minimal solutions (n = 3) in standard stereo configuration. The middle plot compares the NP3L minimal solver with three cameras. The last plot compares the UPnP and our least square solvers with three cameras.
Figure 7: Lidar laser scan for testing our pose estimation algorithms with 4-camera system. 2D detected lines are shown next to the 3D point cloud which colors are the same to their corresponding camera.

Figure 8: Lidar laser scan for testing our pose estimation algorithms with 3-camera system. 2D detected lines are shown next to the 3D point cloud which colors are the same to their corresponding camera.

6. Conclusion

We proposed a linear and a cubic solutions to the NPNL problem from line correspondences with known vertical direction. Both method can be used as a minimal solver (e.g., within RANSAC) as well as a general least squares solver. The methods work for single- and multi-view camera systems without reformulation. The minimal number of line correspondences has been discussed for various common camera configurations. The proposed methods have been evaluated on synthetic and real datasets. While the linear solver is computationally more efficient, it is more sensitive to noise and low number of correspondences, while the cubic solver is much more robust at the price of a slightly increased CPU time. Both methods compare favorably to state of the art approaches.

Acknowledgement

This work was partially supported by the NKFI-6 fund through project K120366; the Agence Universitaire de la Francophonie (AUF) and the Romanian Institute for Atomic Physics (IFA), through the AUF-RO project NETASSIST; the Re- search & Development Operational Programme for the project “Modernization and Improvement of Technical Infrastructure for Research and Development of J. Selvye University in the Fields of Nanotechnology and Intelligent Space”, ITMS 26210120042, co-funded by the European Regional Development Fund.
References


