University of Szeged

Institute of Informatics

MASTER'S THESIS

Nora Horanyi

2017

University of Szeged

Institute of Informatics

Absolute pose estimation using 3D-2D line correspondences and vertical direction

Master's Thesis

Prepared:
Nora Horanyi
Info-bionics Engineering, M.Sc.
Student

Supervisor:
Zoltan Kato
Professor

Szeged
2017

Contents

Task specification

The main goal of the thesis work is to estimate the absolute pose (position and orientation) of a calibrated single or multiview camera system using 3D-2D line correspondences. The algorithm also use additional sensor data (*e.g.* GPS, IMU) but the pose estimation is based on images. In general the problem consist of 6 DOF, however if the vertical direction is available it decrease to 4. The estimated pose can be used for visual odometry if the camera system is mounted on a moving platform.

Abstract

Pose estimation is a fundamental building block of various vision applications, e.g. visual odometry, image-based localization and navigation, fusion, and augmented reality. Herein, we are interested in absolute pose estimation, which consists in determining the position and orientation of a camera with respect to a 3D world coordinate frame.

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. Not only classical image-based techniques, such as Structure from Motion (SfM) provide 3D measurements of a scene, but modern range sensors (e.g. Lidar, Kinect) record 3D structure directly. Thus the availability of 3D data is also becoming widespread, hence methods to estimate absolute pose of a set of cameras based on 2D measurements of the 3D scene received more attention.

Since modern cameras are frequently equipped with various location and orientation sensors, we assume that the vertical direction of the camera system (*e.g.* a gravity vector) is available.

In this work, we will discuss the problem of absolute pose estimation in case of a generalized camera using straight lines, which are common in urban environment. The only assumption about the imaging model is that 3D straight lines are projected via projection planes determined by the line and camera projection directions, i.e. correspondences are given as a 3D world line and its projection plane. Therefore we formulate the problem in terms of 4 unknowns using 3D line – projection plane correspondences which yields a closed form solution.

As an important special case, we address the problem of estimating the absolute pose of a multiview calibrated perspective camera system from 3D - 2D line correspondences. Herein, we propose two solutions: 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. All of the solutions can be used as a minimal solver as well as a least squares solver without reformulation. Experimental results confirm state of the art performance both in terms of quality and computing time.

Keywords: absolute pose estimation, vertical direction, line correspondences, generalized camera, multiview camera system

Chapter 1

Introduction

Pose estimation is a fundamental building block of various vision applications, *e.g.* visual odometry [32], image-based localization and navigation [16], fusion [39], and augmented reality [2]. Herein, we are interested in absolute pose estimation, which consists in determining the position and orientation of a camera with respect to a 3D world coordinate frame.

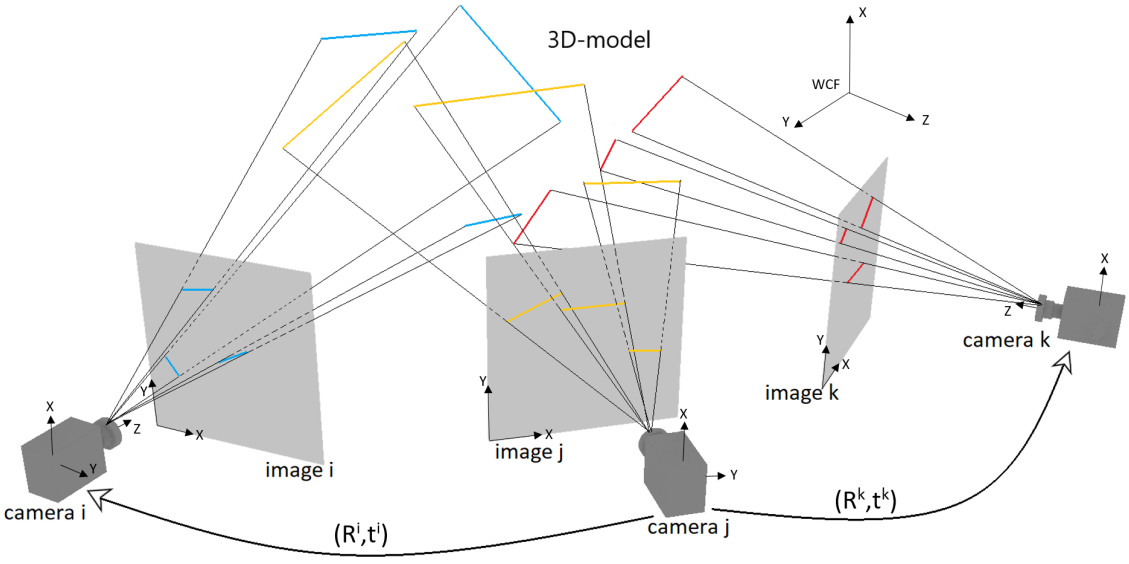


Figure 1.1. Representation of coordinate systems and the 3D-2D lines correspondences in case of a multi-camera system. The corresponding 3D-2D lines from each camera are shown with the same color.

In Fig. 1.1 we show a perspective multi-camera system, the line correspondences and the coordinate systems. The right-handed 3D coordinate system attached to a camera is called the camera coordinate system. Z axis is the main projection ray, which is the

optical axis of the camera. The perpendicular plane to the Z axis, is the projection plane called the image plane, and the XY plane of the camera coordinate system is the principal plane which is parallel to the image plane. Pixels are given in the 2D generalized image coordinate system whose origin is typically in one corner of the picture. The 3D points seen by the cameras are given in a general 3D euclidean coordinate system, called the world coordinate frame (WCF).

In this study, we assume that our camera is fully calibrated, so their intrinsic parameters are known which can be obtained by standard camera calibration methods (*e.g.* Matlab Calibration Toolbox). As a result we get the camera calibration matrix, $\mathbf{K}$ which has the following form [15]:

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.1}$$

where f_x and f_y are the focal lengths, x_0 and y_0 is the principal point offset, and s is the axis skew. This camera calibration matrix is essential for perspective projection as presented in Section 2.3.

The goal of camera pose estimation is to calculate the extrinsic parameters $[\mathbf{R}|\mathbf{t}]$ of the cameras, which is a 3×4 matrix: $\mathbf{R}$ represents a 3×3 rotation matrix that defines the camera orientation with angles α, β, γ and $\mathbf{t}$ is a translation vector which acts between the world and camera coordinate frames.

$$\mathbf{R} = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

Computer-vision applications focus on estimating this $[\mathbf{R}|\mathbf{t}]$ matrix which corresponds to the euclidean transformation from a world coordinate system to the camera coordinate

system.

$$[\mathbf{R}|\mathbf{t}] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \end{bmatrix} \tag{1.3}$$

From these above discussed parameters we can construct the $\mathbf{P}$ camera matrix [15]:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}], \tag{1.4}$$

where $\mathbf{K}$ is the internal calibration matrix, and $[\mathbf{R}|\mathbf{t}]$ is the camera pose. This camera matrix fully describes the projection between 3D points given in the 3D world coordinate systems into the 2D image coordinate system.

Many computer vision applications rely on the presence of stable and representative features in the image. Thus, feature matching is a fundamental task in computer vision and has been widely studied in the past decades. Concerning feature matching, among point, line, region features, point matching has received much attention and various approaches have been proposed *e.g.* SIFT [26], SURF [5]. These approaches first construct a local descriptor to describe the neighborhood distribution of a point, then the point matching is conducted by comparing local descriptors. Through local descriptors, point matching becomes robust to changes of illumination, affine transformation, scale as well as some extent of viewpoint changes [10].

However, most key-points which are popularly used for matching are not localized at edges fail to capture geometrical and structural information about the scene. In contrast, lines supply sufficient geometric information about the scene. Therefore, line matching [13] is both desirable and more reliable in many applications. In this work we only use line correspondences to estimate the absolute pose of the camera or a multi-camera system.

Absolute pose estimation has been extensively studied yielding various formulations and solutions. Most of the approaches focus on a single perspective camera pose estimation using n 2D–3D point correspondences, known as the *Perspective n Point* (PnP) problem [24, 25, 20]. It has been widely studied for both large n as well as for the $n = 3$ minimal case (see [20] for a recent overview). Using line correspondences yields the *Perspective n Line* (PnL) problem (see [43] for a detailed overview). The minimal case of

$n = 3$ line correspondences is particularly important as its solution is the basis for dealing with the general PnL problem. It has been shown in [9], that P3L leads to an 8^{th} order polynomial, which is higher than the 4^{th} polynomial of a P3P problem.

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 [33, 7, 23]. Not only classical image-based techniques, such as Structure from Motion (SfM) [38] provide 3D measurements of a scene, but modern range sensors (*e.g.* Lidar, Kinect) record 3D structure directly. Thus the availability of 3D data is also becoming widespread, hence methods to estimate absolute pose of a set of cameras based on 2D measurements of the 3D scene received more attention [20, 7, 22].

1.1 Generalized absolute pose estimation

In this work, we deal with generalized absolute pose estimation from 3D–2D line correspondences (also known as the gPnL problem) with known vertical direction. While several point-based methods exist [7, 20], little work has been done on using line correspondences for generalized pose. One notable work is the minimal multiview NP3L solver of Lee [22], which deals with full 6 DOF pose parameter estimation. Today, the vast majority of modern cameras, smartphones, UAVs, and camera mounted mobile platforms are equipped with a 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 this *up-vector* is typically between $0.5^\circ - 0.02^\circ$ [1]. While robust minimal solutions based on point correspondences exist for perspective cameras with known vertical direction [1, 21], none of these methods use line correspondences nor generalized non-perspective cameras.

We propose a novel solution to the gPnL problem with known vertical direction. The only assumption about our generalized camera [14] is that projection rays of a 3D line fall into *coplanar* subsets yielding a pencil of projection planes. In the first part of this thesis, our algorithm can be used as a minimal gP3L solver with 3 line correspondences suitable for hypothesis testing like RANSAC [11]. Furthermore, the same algorithm can be used without reformulation for $n > 3$ lines as well as for classical single-view PnL

problems. The performance and robustness of the proposed method have been evaluated on large synthetic datasets as well as on real data.

Important special cases of such a camera include pushbroom, stereo and multiview perspective camera systems [7, 20, 22], perspective camera moving along a trajectory [7, 23, 8, 41, 44], as well as other non-perspective cameras with central omnidirectional [3, 12, 28, 36] or orthographic projection.

1.1.1 Multiview perspective absolute pose estimation

The most common example of the generalized cameras are those widely applied camera systems where N perspective cameras are rigidly assembled. Such a camera system can be treated as one generalized camera with N distinct projection center. In the second part of this thesis, we investigated in more detail the gPnL problem in case of a multiview perspective camera system. For perspective cameras, Mirzaei *et al.* [31] 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 [30], 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 PnL problem. It has been shown in [9], 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 [40, 39] or silhouettes. However, such approaches are typically computationally more expensive hence they cannot be used as real-time solvers.

Herein, we propose two new solutions to the NPnL problem with known vertical direction. Both NPnLup 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 reformulation for $n > 3$ lines as well as for classical single-view PnL problems. The performance and robustness of the proposed methods have been evaluated on large synthetic as well as on real datasets.

Chapter 2

Methodology

2.1 Generalized camera model

While the dominant imaging model in computer vision is perspective projection, recently much more complex vision sensors have been introduced in various application areas [14]. Regardless of its specific design, an imaging system maps incoming rays of light from the scene onto pixels on the detector. For a perspective imaging system, all the incoming light rays are projected directly onto to the detector plane through a single point, namely, the projection center of the perspective system. This is not true in an arbitrary system, *e.g.* a camera system could be comprised of multiple individual perspective or non-perspective imaging systems, each with its own imaging optics and image detector.

If we are not placing any restrictions on the properties of the imaging system, then all rays can be expressed using its Plücker coordinates defined in the generalized camera coordinate frame $\mathcal{C}$ [33, 7]. A Plücker coordinate is a homogeneous 6-vector composed of a pair of 3-vectors $(\mathbf{v}, \mathbf{m})$, where $\mathbf{v}$ is the unit direction vector of the line and $\mathbf{m} = \mathbf{v} \times \mathbf{p}$ is a vector whose direction is perpendicular to the plane containing the line and the origin, and $\mathbf{p}$ is an arbitrary point on the line [33, 7]. Obviously, $\mathbf{v} \cdot \mathbf{m} = 0$.

Thus any 3D point $\mathbf{X}$ in the camera frame $\mathcal{C}$ lying on the camera projection ray $(\mathbf{v}, \mathbf{m})$ can be written as

$$\mathbf{X} = \mathbf{v} \times \mathbf{m} + d\mathbf{v}, \quad d \in \mathbb{R}. \tag{2.1}$$

Since $\mathbf{v}$ is a unit vector, the point $(\mathbf{v} \times \mathbf{m})$ is the point on the ray closest to the ori-

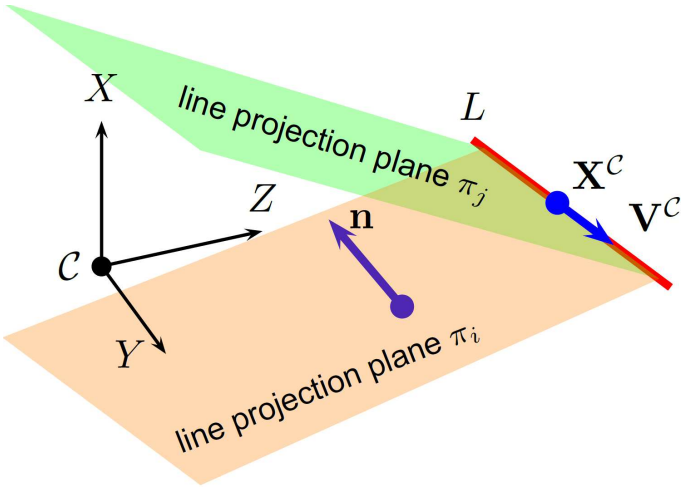


Figure 2.1. Projection of a 3D line L by a generalized camera.

gin and d is the (signed) distance from that point [33]. The (2.1) expression defines the 3D point - ray correspondence for a generalized camera. While this is sufficient to estimate the absolute pose with respect to a world coordinate frame $\mathcal{W}$, rays are usually not directly available in an imaging system, it has to be computed from recorded pixels. Following [14], we can represent the mapping of rays to pixels through so called ray pixels or *raxel*s, which are conveniently arranged on a so called *ray surface*. For example, central cameras (both perspective and non-perspective) are often represented by a unit sphere [3, 12, 28, 36]. Many imaging systems have a special ray surface, called *caustic*, which is defined as the envelope of the incoming rays (*i.e.* incoming rays are tangent to this surface). As argued by [14], caustics are the logical place to locate raxels. Note that in the perspective case, the caustic is a point (the projection center).

2.1.1 Line projection

While point projection of a generalized camera is governed by (2.1), the projection of lines becomes much more complex. For points, we assume (according to physical laws) that they are projected by *straight* lines and the image of a point will be a set of points. However lines may have a very complex projection in a generalized camera as opposed to a well defined line in the perspective case. Therefore we make an additional assumption about our camera: the projection rays of a 3D line L can be divided into *coplanar* subsets yielding a pencil of projection planes π_i^L with L being the axis of the pencil (see Fig. 2.1).

For such a given projection plane π_i^L, one image of the line L becomes a general curve on the ray surface determined by the intersection of π_i^L with the ray surface. Common examples include multiview central cameras (see Sec. 2.2.2), or linear pushbroom imaging which may be thought of as a projective image in one direction and an orthographic image in the other direction.

2.2 Generalized absolute pose

Herein, 3D lines will be represented by their unit direction vector $\mathbf{v}$ and by an arbitrary point $\mathbf{X}$ on it: $L = (\mathbf{v}, \mathbf{X})$. The projection planes π_i^L are given by their unit normal $\mathbf{n}_i$ and the signed distance d_i from the origin, which is also known as the Hessian normal form: $\pi_i^L = (\mathbf{n}_i, d_i)$. The sign of d_i determines the side of the plane on which the origin is located. If $d_i > 0$, it is in the half-space determined by the normal direction $\mathbf{n}_i$, and if $d_i < 0$, it is in the other half-space. Note that the (signed) point-plane distance from a homogeneous 3D point $\mathbf{X} = (X_1, X_2, X_3, 1)$ to a plane π_i^L is given by $\pi_i^L \cdot \mathbf{X}$, which should be 0 for points on the plane. If the point $\mathbf{X}$ is in the half-space determined by the normal direction $\mathbf{n}_i$, then the distance is positive; if it is in the other half-space, then the distance is negative.

Since L lies on π_i^L, its direction vector $\mathbf{v}$ is perpendicular to $\mathbf{n}_i$:

$$\forall i : \mathbf{n}_i \cdot \mathbf{v}^{\mathcal{C}} = \mathbf{n}_i \cdot \mathbf{R}\mathbf{v} = 0 \tag{2.2}$$

where $\mathbf{R}$ is the rotation matrix from the world $\mathcal{W}$ to the camera frame $\mathcal{C}$ and $\mathbf{v}^{\mathcal{C}}$ denotes the unit direction vector of L in the camera frame $\mathcal{C}$. Furthermore, the point $\mathbf{X}$ on line L also lies on π_i^L, hence

$$\forall i : \pi_i^L \cdot (\mathbf{X}^{\mathcal{C}}, 1) = \mathbf{n}_i \cdot (\mathbf{R}\mathbf{X} + \mathbf{t}) + d_i = 0 \tag{2.3}$$

where $\mathbf{t}$ is the translation from the world $\mathcal{W}$ to the camera $\mathcal{C}$ frame and $\mathbf{X}^{\mathcal{C}}$ denotes the point on L in the camera coordinate system $\mathcal{C}$.

The absolute pose of our generalized camera is defined as the rigid transformation $(\mathbf{R}, \mathbf{t})$ acting between $\mathcal{W}$ and $\mathcal{C}$, and the equations (2.2)–(2.3) provide constraints for

computing the pose using 3D line and projection plane correspondences. Note that both equations have a geometric meaning: (2.2) expresses the $\cos$ of the angle between the plane normal and the line direction; while (2.3) gives the signed distance of the point on the line and its projection plane. Hence minimizing the squared error of these equations would actually minimize the geometric error.

2.2.1 Known vertical direction

Before formulating our solution, let us see the parametrization of $\mathbf{R}$ when the vertical direction is available. This information is typically obtained from an IMU (Inertial Measurement Unit), which consists of accelerometers capable to measure the Earth's gravity center. The accuracy of an IMU's *up-vector* is typically between $0.5^\circ - 0.02^\circ$ [1]. However, similar information can be obtained from a calibrated image by detecting a vanishing point (in man-made environment, one can get the vertical vanishing point) [15]. In fact, the knowledge of any direction would lead to a similar formulation of the problem.

Assuming that the camera coordinate system $\mathcal{C}$ is a standard right handed system with the X axis pointing *up* (see Fig. 2.1), the world unit vector $(1,0,0)^\top$ is known in the camera coordinate frame $\mathcal{C}$. Given this *up-vector*, we can compute rotation $\mathbf{R}_v$ around Y and Z axes, which aligns the world X axis with the camera X axis:

$$\mathbf{R}_v = \mathbf{R}_Z(\gamma)\mathbf{R}_Y(\beta) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{2.4}$$

The only unknown parameter in the rotation matrix $\mathbf{R}$ is the rotation $\mathbf{R}_X(\alpha)$ around the vertical X axis:

$$\mathbf{R}_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{2.5}$$

thus the $\mathcal{W} \to \mathcal{C}$ rotation $\mathbf{R}$ has the following form:

$$\mathbf{R} = \mathbf{R}_v \mathbf{R}_X = \mathbf{R}_v \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{2.6}$$

Unfortunately, the above factorization gives only the $\cos$ and $\sin$ of the rotation angle which, when plugged into (2.2)–(2.3), yields trigonometric equations in α but linear in $\mathbf{t}$. In order to eliminate $\sin(\alpha)$ and $\cos(\alpha)$, we can use the substitution $q = \tan(\alpha/2)$ [21, 1], for which $\cos(\alpha) = (1-q^2)/(1+q^2)$ and $\sin(\alpha) = 2q/(1+q^2)$. Therefore

$$(1+q^2)\mathbf{R}_X(q) = \begin{bmatrix} 1+q^2 & 0 & 0 \\ 0 & 1-q^2 & -2q \\ 0 & 2q & 1-q^2 \end{bmatrix}. \tag{2.7}$$

Note that q becomes numerically unstable for α close to $180°$. However, with the known vertical direction we can always make a rough initialization of the camera pose, thus the $\alpha = 180°$ degenerate case excluded as it would correspond to a camera looking in the opposit direction of the true pose. Substituting $\mathbf{R}_X(q)$ into (2.2) yields a quadratic equation in the single unknown q:

$$\begin{aligned} \forall i: \quad & a_i q^2 + b_i q + c_i = 0, \text{ with} \\ a_i &= (\mathbf{n}_i^\top \mathbf{R}_v) \begin{bmatrix} v_1 \\ -v_2 \\ -v_3 \end{bmatrix} \\ b_i &= 2(v_2(\mathbf{n}_i^\top \mathbf{R}_v^1) - v_3(\mathbf{n}_i^\top \mathbf{R}_v^2)) \\ c_i &= (\mathbf{n}_i^\top \mathbf{R}_v)\mathbf{v}, \end{aligned} \tag{2.8}$$

where $\mathbf{v} = (v_1, v_2, v_3)^\top$ is the unit direction vector of L, while $\mathbf{R}_v^1$ and $\mathbf{R}_v^2$ denote the first and second column vector of $\mathbf{R}_v$, respectively. Once a solution is obtained for $\mathbf{R}_X$, it can

be substituted into (2.3) yielding the following linear equation in $\mathbf{t} = (t_1, t_2, t_3)^\top$:

$$\begin{aligned} \mathbf{n}_i^\top(\mathbf{R}_v\mathbf{R}_X\mathbf{X}) + \mathbf{n}_i^\top\mathbf{t} + d_i &= 0 \\ n_{i,1}t_1 + n_{i,2}t_2 + n_{i,3}t_3 + \mathbf{n}_i^\top(\mathbf{R}\mathbf{X}) + d_i &= 0 \\ \begin{bmatrix}\mathbf{n}_i^\top & \mathbf{n}_i^\top(\mathbf{R}\mathbf{X}) + d_i\end{bmatrix}\begin{bmatrix}\mathbf{t} \\ 1\end{bmatrix} &= 0 \end{aligned} \tag{2.9}$$

where $\mathbf{X}$ is a point on the 3D line L and $\mathbf{n}_i$ is one projection plane of L in the generalized camera. Given a set of 3D lines and their projection planes, each such pair provides one linear equation of the form (2.9) yielding a homogeneous linear system in $\mathbf{t}$ whose matrix has rows $[\mathbf{n}_i^\top, \mathbf{n}_i^\top(\mathbf{R}\mathbf{X}) + d_i]$.

Number of correspondences

For each 3D line L and each corresponding projection plane π_i^L, we get one quadratic equation of the form (2.8) and one linear equation of the form (2.9). Thus having N_L distinct projections generates N_L equations for L. The total number of equations is thus $n = \sum_{\ell=1}^{M} N_\ell$ where M is the total number of 3D lines visible (*i.e.* having at least one projection plane) in our generalized camera.

The minimal case of the linear system (2.9) consists of three pairs of L and π^L, each one providing one equation in $\mathbf{t}$, which is easily solved. Although the minimal case for (2.8) would be only one pair of L and π^L, we always need 3 pairs of (L, π^L) because of $\mathbf{t}$. Therefore we only address the least squares solution of (2.8). We note, however, that based on one (L, π^L) pair, the rotation parameter q could be obtained as a direct solution of the quadratic equation (2.8). The reason why we not proceed in this way is because the least squares formulation also leads a direct solution, hence computationally it is close to this minimal solution, while the additional constraints ensure an increased numerical stability. As a consequence, the formulation of our method is identical for the minimal as well as least squares cases. Minimal solvers are typically employed for robust pose estimation within RANSAC [11] in order to maximize the probability of picking an all-inlier sample, hence reducing the number of iterations.

Efficient solution: gPnLup

In the non-minimal case, both (2.8) and (2.9) becomes overdetermined and can be solved in the least squares sense [17]. Let us emphasize, that the least squares solution minimizes the *geometric error* of the solution as (2.8) expresses the $\cos$ of the angle between the plane normal and the line direction; while (2.9) gives the signed distance of the point $\mathbf{X}$ on the line and its projection plane. The squared error of (2.8) becomes a quartic polynomial in q:

$$\sum_{i=1}^{n} \left(a_i^2 q^4 + 2a_i b_i q^3 + (2a_i c_i + b_i^2) q^2 + 2b_i c_i q + c_i^2\right), \tag{2.10}$$

whose minima is found by computing the roots of its derivative

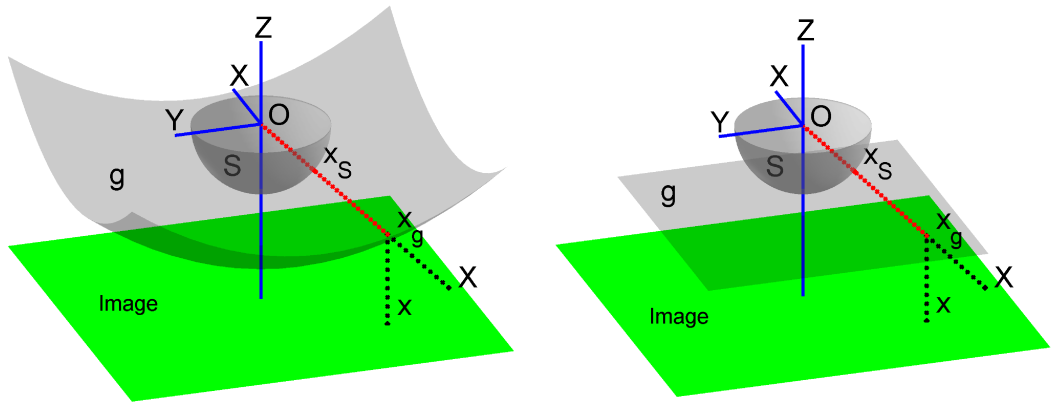
$$\sum_{i=1}^{n} \left(4a_i^2 q^3 + 6a_i b_i q^2 + (4a_i c_i + 2b_i^2) q + 2b_i c_i\right) \tag{2.11}$$

The roots of this third order polynomial are computed in a closed form. In general, there are maximum 3 roots, at least one of them being real. In case of multiple solutions, each real root is back-substituted into (2.9) to compute the corresponding $\mathbf{t}$. The final solution is then selected by checking whether lines are consistently placed w.r.t. the camera system, or by evaluating the reprojection error of each solution and choosing the one with minimal error.

2.2.2 Multiview central cameras

One common example of a generalized camera is a set of central cameras [20, 7, 33, 23]. Let us have a closer look at this important special case, when 3D lines are viewed by N central cameras. Note that a central camera may or may not be perspective! Even when a camera has a single effective viewpoint, its projection model may include non-linear distortions, like in the case of central omnidirectional cameras [3, 12, 19, 37, 39]. Herein, we will consider an arbitrary mixture of perspective and non-perspective central cameras and derive equations for (2.8) and (2.9) for this important special case.

A unified model for central omnidirectional cameras was proposed by Geyer and Daniilidis [12], which represents central panoramic cameras as a projection onto the sur-



omnidirectional perspective

Figure 2.2. Spherical representation of central projection cameras.

face of a unit sphere $\mathcal{S}$. The camera coordinate system is in the center of $\mathcal{S}$, and the Z axis is the optical axis of the camera which intersects the image plane in the *principal point*. This formalism has been adopted and models for the internal projection function have been proposed by Micusik [28, 27] and subsequently by Scaramuzza [37] who derived a general polynomial form $g(\|\mathbf{x}\|) = a_0 + a_2\|\mathbf{x}\|^2 + a_3\|\mathbf{x}\|^3 + a_4\|\mathbf{x}\|^4$ which has 4 parameters representing the internal calibration parameters (a_0, a_2, a_3, a_4) of the camera (only 4 parameters as a_1 is always 0 [37]). Thus the nonlinear (but symmetric) distortion of central omnidirectional optics is represented by placing this rotationally symmetric g surface between the image plane and the unit sphere $\mathcal{S}$ [37] (see Fig. 2.2). Knowing the internal calibration of the camera allows us to work directly with spherical image points $\mathbf{x}_\mathcal{S} \in \mathcal{S}$ using the bijective mapping of image points $\mathbf{x} \mapsto \mathbf{x}_\mathcal{S}$ composed of 1) lifting the image point $\mathbf{x}$ onto the g surface by an orthographic projection

$$\mathbf{x}_g = \begin{bmatrix} \mathbf{x} \\ a_0 + a_2\|\mathbf{x}\|^2 + a_3\|\mathbf{x}\|^3 + a_4\|\mathbf{x}\|^4 \end{bmatrix} \tag{2.12}$$

and then 2) centrally projecting the lifted point $\mathbf{x}_g$ onto the surface of the unit sphere $\mathcal{S}$:

$$\mathbf{x}_\mathcal{S} = \frac{\mathbf{x}_g}{\|\mathbf{x}_g\|} \tag{2.13}$$

Similarly, the image points of a perspective camera can be represented on $\mathcal{S}$ by the bijective mapping $\mathbf{x} \mapsto \mathbf{x}_\mathcal{S}$: $\mathbf{x}_K = \mathbf{K}^{-1}\mathbf{x}$ and $\mathbf{x}_\mathcal{S} = \mathbf{x}_K/\|\mathbf{x}_K\|$ (see Fig. 2.2). Thus the

projection of a calibrated central camera is fully described by means of unit vectors $\mathbf{x}_{\mathcal{S}}$ in the half space of $\mathbb{R}^3$. A 3D world point $\mathbf{X}$ is projected into $\mathbf{x}_{\mathcal{S}} \in \mathcal{S}$ by a simple central projection taking into account the pose:

$$\mathbf{x}_{\mathcal{S}} = \frac{\mathbf{R}\mathbf{X} + \mathbf{t}}{\|\mathbf{R}\mathbf{X} + \mathbf{t}\|} \tag{2.14}$$

Following the line projection model outlined in Section 2.1.1, a 3D line L is centrally projected by a single projection plane $\pi_L = (\mathbf{n}, d)$ onto the surface $\mathcal{S}$. Since the camera projection center is also on π_L, d becomes zero and thus π_L is uniquely determined by its unit normal $\mathbf{n}$. The image of L is the intersection of the *ray surface* $\mathcal{S}$ and π_L, which is a *great circle*, while a particular line segment becomes a *great circle segment* on the unit sphere $\mathcal{S}$. When we have N central cameras, a 3D line L has up to N images, one in each camera. These cameras may be rigidly assembled into a multi-camera system or they might originate from a single camera moving along a trajectory [33, 7, 23, 2] – in either case, they form a generalized camera with known relative poses $\mathcal{C} \rightarrow \mathcal{C}_i$ with respect to the common camera coordinate frame $\mathcal{C}$. This is obtained by *e.g.* calibration of the multi-camera system [33] or by tracking the pose of a moving camera using visual-inertial-odometry [16, 7] or visual-odometry [32, 7]. Thus individual cameras are related to $\mathcal{C}$ via $(\mathbf{R}^i, \mathbf{t}^i) : \mathcal{C} \rightarrow \mathcal{C}_i$ and the projection plane $\pi^{\mathcal{C}^i} = (\mathbf{n}^{\mathcal{C}^i}, 0)$ of L in camera $\mathcal{C}^i$ is given in the generalized camera frame $\mathcal{C}$ as

$$\pi_i^L = (\mathbf{n}_i, d_i) \text{ with } \mathbf{n}_i = \mathbf{R}^{i\top}\mathbf{n}^{\mathcal{C}^i} \text{ and } d_i = \mathbf{n}^{\mathcal{C}^i} \cdot \mathbf{t}^i \tag{2.15}$$

Substituting the above expressions for $\mathbf{n}_i$ and d_i into (2.8)–(2.9) gives the equations for multiview central cameras which will be explained in details in section 2.3.

2.3 Multiview perspective cameras

Perspective Projection of Lines

In case we use multi-camera systems which covers large field of views, this problem is known as multiview absolute pose estimation as such a camera system may be modeled

as a generalized camera with N projection centers. An important difference w.r.t. 2.2.2, is that here we only use perspective cameras!

Given a calibrated camera $\mathbf{P}$ and 3D lines L_i in the world coordinate frame, the projection of the lines are 2D lines l_i in the image plane. The perspective camera matrix $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ consists of the internal calibration matrix $\mathbf{K}$ and the camera pose $[\mathbf{R}|\mathbf{t}]$ w.r.t. the world coordinate frame. A homogeneous 3D point $\mathbf{X}$ is mapped by $\mathbf{P}$ into a homogeneous 2D image point $\mathbf{x}'$ as [15]

$$\mathbf{x}' \cong \mathbf{P}\mathbf{X} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}, \tag{2.16}$$

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 (2.16) by $\mathbf{K}^{-1}$ and work with the equivalent normalized image

$$\mathbf{x} = \mathbf{K}^{-1}\mathbf{x}' \cong \mathbf{K}^{-1}\mathbf{P}\mathbf{X} = [\mathbf{R}|\mathbf{t}]\mathbf{X}. \tag{2.17}$$

The above equation is the starting point of absolute perspective pose estimation [20, 25, 24, 21]: given a set of 3D–2D point correspondences $(\mathbf{x}_i \leftrightarrow \mathbf{X}_i)$, one can recover the 3D rigid body transformation $(\mathbf{R}, \mathbf{t}) : \mathcal{W} \rightarrow \mathcal{C}$ acting between the world coordinate frame $\mathcal{W}$ and the camera coordinate frame $\mathcal{C}$.

Unlike points, 3D lines may have various representations in the projective space [34, 4]. 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 (2.17) [15, 4, 22, 35, 44, 29]. 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

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.18}$$

is indirect, *i.e.* two points of the line have to be transformed and then form their Plücker coordinates again. In [4], 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

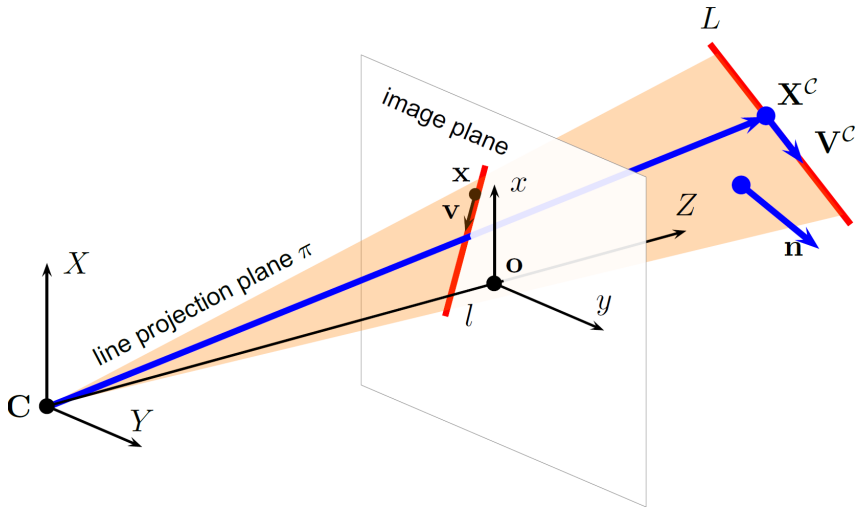


Figure 2.3. Perspective projection of a 3D line $L \rightarrow l$.

matrix. This approach is also used in [22, 35] for absolute pose estimation.

Herein, 3D lines are represented as $L = (\mathbf{V}, \mathbf{X})$, where $\mathbf{V}$ is the unit direction vector of the line and $\mathbf{X}$ is a point on the line (see Fig. 2.3) [41]. The projection of L in a central perspective camera is a line l in the image plane, which can also be represented as $l = (\mathbf{v}, \mathbf{x})$. Note that the point $\mathbf{x}$ is *not* necessarily the image of the 3D point $\mathbf{X}$! Both L and l lie on the projection plane π passing through the camera projection center $\mathbf{C}$. The unit normal to the plane π in the camera coordinate system $\mathcal{C}$ is denoted by $\mathbf{n}$, which can be computed from the image line $l = (\mathbf{v}, \mathbf{x})$ as $\mathbf{n} = (\mathbf{v} \times \mathbf{x})/\|\mathbf{v} \times \mathbf{x}\|$. Since L lies also on π, its direction vector $\mathbf{V}$ is perpendicular to $\mathbf{n}$. Hence, similar to (2.2)

$$\mathbf{n}^\top \mathbf{R}\mathbf{V} = \mathbf{n}^\top \mathbf{V}^\mathcal{C} = 0, \tag{2.19}$$

where $\mathbf{R}$ is the rotation matrix from the world $\mathcal{W}$ to the camera $\mathcal{C}$ frame and $\mathbf{V}^\mathcal{C}$ denotes the unit direction vector of L in the camera coordinate frame. Furthermore, the vector from the camera center $\mathbf{C}$ to the point $\mathbf{X}$ on line L is also lying on π, thus it is also perpendicular to $\mathbf{n}$. Yielding a similar formula to (2.3):

$$\mathbf{n}^\top (\mathbf{R}\mathbf{X} + \mathbf{t}) = \mathbf{n}^\top \mathbf{X}^\mathcal{C} = 0, \tag{2.20}$$

where $\mathbf{t}$ is the translation from the world $\mathcal{W}$ to the camera $\mathcal{C}$ frame and $\mathbf{X}^\mathcal{C}$ denotes the point $\mathbf{X}$ on L in the camera coordinate frame.

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 $\mathbf{K}^i$ as well as their relative pose $(\mathbf{R}^i, \mathbf{t}^i) : \mathcal{C}^i \rightarrow \mathcal{C}$ with respect to a common camera coordinate frame $\mathcal{C}$ are known. The common coordinate frame $\mathcal{C}$ 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 $(\mathbf{R}, \mathbf{t})$ is defined as the rigid transformation acting between $\mathcal{W}$ and $\mathcal{C}$, while individual camera frames $\mathcal{C}^i$ are related to the world coordinate frame via the sequence of rigid transformations

$$\forall i : \mathbf{M}^{\mathcal{W}\rightarrow i} = \begin{bmatrix} \mathbf{R}^i & \mathbf{t}^i \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{2.21}$$

In fact, the whole camera system can be regarded as a generalized non-perspective camera with N projection center [7]. In such a non-central camera, each 3D line L has up to N images $l^i, i = 1 \dots N$, 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 $(\mathbf{n}^i, \mathbf{n}^j)$, the unit direction vector $\mathbf{V}^{\mathcal{C}}$ of L can be expressed in the camera frame $\mathcal{C}$ as

$$\mathbf{V}^{\mathcal{C}} = \frac{\mathbf{R}^{i\top}\mathbf{n}^i \times \mathbf{R}^{j\top}\mathbf{n}^j}{\|\mathbf{R}^{i\top}\mathbf{n}^i \times \mathbf{R}^{j\top}\mathbf{n}^j\|}, \tag{2.22}$$

which yields the following relation

$$\mathbf{V}^{\mathcal{C}} = \mathbf{R}\mathbf{V} \Rightarrow (\mathbf{R}\mathbf{V}) \times (\mathbf{R}^{i\top}\mathbf{n}^i \times \mathbf{R}^{j\top}\mathbf{n}^j) = \mathbf{0} \tag{2.23}$$

Thus a natural approach to our absolute pose estimation problem would be to reconstruct 3D line directions in the camera frame using *e.g.* (2.22) and then solving (2.23) for $\mathbf{R}$. 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 accuracy, which is known to be quite poor for practically important setups like narrow baseline stereo [8]. In general, (2.23) becomes numerically unstable whenever $\mathbf{R}^{i\top}\mathbf{n}^i$ and $\mathbf{R}^{j\top}\mathbf{n}^j$ 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 errors in a system of equations constructed from (2.23).

Essentially (2.23) states that $\mathbf{V}^{\mathcal{C}}$ is perpendicular to both $\mathbf{n}^i$ and $\mathbf{n}^j$, yielding the multi-view form of (2.19):

$$\forall i \text{ where } L \text{ is visible: } \mathbf{n}^{i\top}\mathbf{R}^i\mathbf{V}^{\mathcal{C}} = \mathbf{n}^{i\top}\mathbf{R}^i\mathbf{R}\mathbf{V} = 0 \tag{2.24}$$

While this equation is mathematically equivalent to (2.23), it is numerically more favorable as it provides *separate* equations for each normal thus avoiding multiplication of noisy $\mathbf{n}^i$ measurements. Similarly, (2.20) can be written for the multi-camera case as:

$$\forall i \text{ where } L \text{ is visible: } \mathbf{n}^{i\top}(\mathbf{R}^i\mathbf{X}^{\mathcal{C}} + \mathbf{t}^i) = \mathbf{n}^{i\top}(\mathbf{R}^i(\mathbf{R}\mathbf{X} + \mathbf{t}) + \mathbf{t}^i) = 0 \tag{2.25}$$

With the previously presented parametrization (2.6), the equations (2.24) and (2.25) have the form for all camera i where L is visible:

$$\mathbf{n}^{i\top}\mathbf{R}^i\mathbf{R}_v\mathbf{R}_X(\alpha)\mathbf{V} = 0 \tag{2.26}$$

$$\mathbf{n}^{i\top}(\mathbf{R}^i(\mathbf{R}_v\mathbf{R}_X(\alpha)\mathbf{X} + \mathbf{t}) + \mathbf{t}^i) = 0 \tag{2.27}$$

Efficient Solutions

We aim to compute $\mathbf{R}_X(\alpha)$ and $\mathbf{t}$ using the equations in (2.26) - (2.27). We have 4 unknowns: the rotation angle α 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 $\mathbf{t}$. Therefore we need at least 3 line correspondences. In the following, we propose two solutions [18]. Both of them use the fact that the images are normalized (*i.e.* image points are multiplied by the inverse of $\mathbf{K}^i$ as in (2.17)); the relative pose $(\mathbf{R}^i, \mathbf{t}^i)$ of each camera is known w.r.t. the common camera frame $\mathcal{C}$; and the vertical direction is known, *i.e.* the rotation matrix $\mathbf{R}_v$ of (2.4) is available.

Linear Solution: NPnLupL The equations (2.26) - (2.27) are linear in $\mathbf{t}$, but $\mathbf{R}_X(\alpha)$ is defined in terms of $\cos(\alpha)$ and $\sin(\alpha)$. Letting these trigonometric functions of α to be two separate unknowns c and s [25, 43, 45], respectively, one can linearize (2.26) - (2.27) by substituting

$$(\mathbf{R}^i\mathbf{R}_v)\mathbf{R}_X(\alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \tag{2.28}$$

into (2.26) - (2.27). Stacking these pairs of equations for n correspondences, we get $2n$ homogeneous linear equations with unknowns $\mathbf{p} = (c, s, t_1, t_2, t_3, 1)^\top$. Hence we have to solve an $\mathbf{Ap} = \mathbf{0}$ system of equations in the least squares sense, which is straightforward using SVD of $\mathbf{A}$. Note that the elements of the $2n \times 6$ matrix $\mathbf{A}$ are expressed in terms of $\mathbf{n}^i$, $\mathbf{R}^i\mathbf{R}_v$, $\mathbf{V}$, $\mathbf{X}$, and $\mathbf{t}^i$ using (2.26) and (2.27). 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 $\mathbf{R}_X(\alpha)$:

$$\cos(\alpha) = \frac{c}{\sqrt{c^2+s^2}} \quad \text{and} \quad \sin(\alpha) = \frac{s}{\sqrt{c^2+s^2}} \tag{2.29}$$

At the price of higher computational complexity, a somewhat more sophisticated normalization involves an additional 3D registration step [25, 43, 45], which aligns the 3D world $\{\mathbf{X}_i\}$ and camera $\{\mathbf{X}_i^c\}$ point sets using a standard 3D registration scheme [42].

A major drawback of this linear solution is that orthonormality constraint on $\mathbf{R}_X(\alpha)$ 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.

Cubic Polynomial Solution: NPnLupC Another way to eliminate $\cos(\alpha)$ and $\sin(\alpha)$ from $\mathbf{R}_X$ was already introduced in Section 2.2.1 which results in (2.7). Substituting this $\mathbf{R}_X(q)$ into (2.26) yields a quadratic equation in the single unknown q. Since we have $n \geq 3$ line correspondences, we obtain n quadratic equations in q:

$$a_i q^2 + b_i q + c_i = 0, \quad i = 1 \dots n, \tag{2.30}$$

where the coefficients a_i, b_i, c_i are computed in terms of $\mathbf{n}^i$, $\mathbf{R}^i\mathbf{R}_v$, and $\mathbf{V}$ using (2.26):

$$\begin{aligned} \forall i: \quad & a_i q^2 + b_i q + c_i = 0, \text{ with} \\ a_i &= (\mathbf{n}_i^\top \mathbf{R}^i \mathbf{R}_v) \begin{bmatrix} V_1 \\ -V_2 \\ -V_3 \end{bmatrix} \\ b_i &= 2(V_2(\mathbf{n}_i^\top (\mathbf{R}^i\mathbf{R}_v)^1) - V_3(\mathbf{n}_i^\top (\mathbf{R}^i\mathbf{R}_v)^2)) \\ c_i &= (\mathbf{n}_i^\top \mathbf{R}^i \mathbf{R}_v)\mathbf{V}, \end{aligned} \tag{2.31}$$

where $\mathbf{V} = (V_1, V_2, V_3)^\top$ is the unit direction vector of L, while $(\mathbf{R}^i\mathbf{R}_v)^1$ and $(\mathbf{R}^i\mathbf{R}_v)^2$ denote the first and second column vector of $\mathbf{R}^i\mathbf{R}_v$, respectively. Once a solution is obtained for $\mathbf{R}_X$, it can be backsubstituted into (2.3) yielding the following linear equation in $\mathbf{t} = (t_1, t_2, t_3)^\top$:

$$\begin{aligned} \mathbf{n}_i^\top (\mathbf{R}^i(\mathbf{R}_v\mathbf{R}_X\mathbf{X} + \mathbf{t}) + \mathbf{t_i}) &= 0 \\ \begin{bmatrix} \mathbf{n}_i^\top & \mathbf{n}_i^\top(\mathbf{R}^i\mathbf{R}\mathbf{X}) + \mathbf{n}_i^\top \mathbf{t_i} \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix} &= 0 \end{aligned} \tag{2.32}$$

where $\mathbf{X}$ is a point on the 3D line L and $\mathbf{n}_i$ is one projection plane normal of L in the N-view perspective camera system. Given a set of 3D lines and their projection plane normals, each such pair provides one linear equation of the form (2.9) yielding a homogeneous linear system in $\mathbf{t}$ whose matrix has rows $[\mathbf{n}_i^\top, \mathbf{n}_i^\top(\mathbf{R}^i\mathbf{R}\mathbf{X}) + \mathbf{n}_i^\top \mathbf{t_i}]$.

We will solve (2.30) nonlinear system of equations in terms of least square residual. The squared error of the equations is a quartic function in q, formulated the same way as (2.10), whose minima is found by computing the roots of its derivative. This results in an equation of the same form as (2.11), with a_i, b_i, c_i coefficients given (2.31).

Solving this third order polynomial equation in a closed form results in maximum 3 solutions, at least one of them being real. For each real root, we have to determine a $\mathbf{t}$ by back substituting each possible $\mathbf{R}_X(q)$ into (2.32), which yields a simple linear system of equations in $\mathbf{t}$.

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 N-view perspective case, the reprojection error characterizes the difference between the observed image line l^i and the reprojected image line $\hat{l}^i$ for all cameras [41]:

$$\epsilon = \sum_{i=1}^{N} \sum_{j=1}^{n} \hat{\mathbf{n}}_j^{i\top} (\mathbf{A}^\top \mathbf{B} \mathbf{A}) \hat{\mathbf{n}}_j^i, \text{ with}$$
$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_j^i \\ \mathbf{b}_j^i \end{bmatrix}, \text{and } \mathbf{B} = \frac{|l_j^i|}{3(n_{j1}^{i2} + n_{j2}^{i2})} \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad (2.33)$$

where $|l_j^i|$ denotes the length of the image line with the 2D homogeneous endpoints $\mathbf{a}_j^i = (a_{j1}^i, a_{j2}^i, 1)^\top$ and $\mathbf{b}_j^i = (b_{j1}^i, b_{j2}^i, 1)^\top$; $\hat{\mathbf{n}}_j^i$ is the reprojection plane normal in camera i computed from the corresponding 3D line $L_j = (\mathbf{X}_j, \mathbf{V}_j)$ as

$$\hat{\mathbf{n}}_j^i = (\mathbf{R}^i(\mathbf{R}\mathbf{X}_j + \mathbf{t}) + \mathbf{t}^i) \times (\mathbf{R}^i \mathbf{R} \mathbf{V}_j). \quad (2.34)$$

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

Chapter 3

Implementation

The summary of the proposed algorithms for absolute pose estimation are presented in Algorithm 1, 2 and 3.

Algorithm 1 Summary of the gPnLup method (Section2.2.1)

Input: Structure (detailed in Fig. 3.1)

Output: The orientation and the position of camera in world coordinate frame

1: Normal vector calculation
2: Constructing $\mathbf{R}_v = \mathbf{R}_Z(\gamma)\mathbf{R}_Y(\beta)$ rotation matrix with the known vertical direction (Equation 2.4)
3: Calculation of the coefficients and solving the nonlinear system of equations (Section 2.2.1) in terms of least squares residual for q
4: Construct $\mathbf{R}_X(\alpha)$ matrix with each real q solution
5: Determine the translational components by solving a linear system of equations (Equation 2.9)
6: Select the solution with the smallest backprojection error

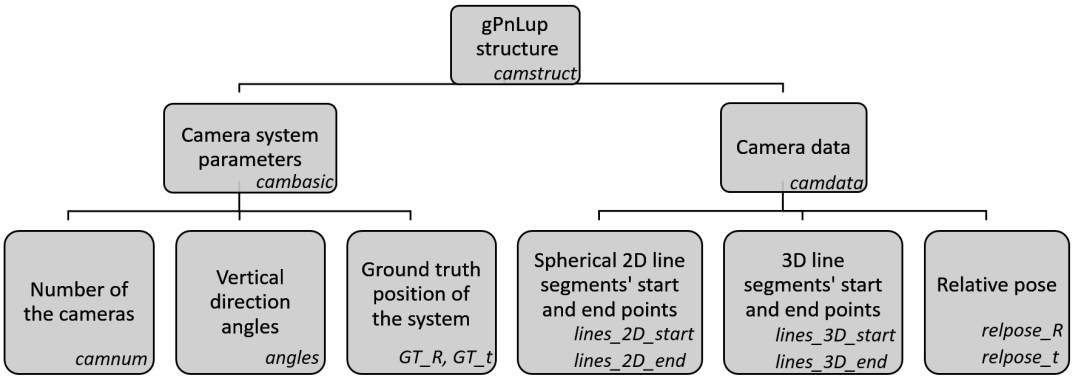


Figure 3.1. Organization of the input structure of gPnLup. Note that we only used the ground truth pose of the camera system for evaluating the accuracy of our methods. In each box we also give the actual name of the variable in the structure.

Algorithm 2 Summary of the NPnLupL method (Section2.3)

Input: Start point and end point of the image lines
Vertical direction rotation angle around Y and Z axis in radian
Direction vector of the 3D line in the world frame
Point of the 3D line in the world frame
Relative position of the camera w.r.t. the camera coordinate system

Output: The orientation and the position of camera in world coordinate frame

1: Normal vector calculation
2: Constructing $\mathbf{R}_v = \mathbf{R}_Z(\gamma)\mathbf{R}_Y(\beta)$ rotation matrix (Equation 2.4)
3: Solving the homogeneous linear equations (Section 2.3) with unknowns $\mathbf{p} = (c, s, t_1, t_2, t_3, 1)^\top$ using SVD
4: Normalization of the estimated c and s separate unknowns (Equation 2.29), construction of $\mathbf{R}_X(\alpha)$
5: Select the solution with the smallest backprojection error

Algorithm 3 Summary of the NPnLupC method (Section2.3)

Input: Start point and end point of the image lines
Vertical direction rotation angle around Y and Z axis in radian
Direction vector of the 3D line in the world frame
Point of the 3D line in the world frame
Relative position of the camera w.r.t. the camera coordinate system

Output: The orientation and the position of camera in world coordinate frame

1: Normal vector calculation
2: Constructing $\mathbf{R}_v = \mathbf{R}_Z(\gamma)\mathbf{R}_Y(\beta)$ rotation matrix (Equation 2.4)
3: Calculation of the coefficients and solving the nonlinear system of equations (Equation 2.30) in terms of least squares residual for q
4: Construct $\mathbf{R}_X(\alpha)$ matrix with each real q solution
5: Determine the translational components by solving a linear system of equations (Equation 2.32)
6: Select the solution with the smallest backprojection error

The proposed methods were implemented in Matlab. Hereby we present only the most complex one, the gPnLup algorithm and in addition some explanations to better understand the functionality. The other methods work under the same principles.

This implementation of gPnLup can work with multiview central camera system (See in Section 2.2.2). In case of these systems the natural input is the endpoints of the line segments and the relative pose between the cameras. This was necessary as line extraction methods usually provide us line segments. This algorithm is prepared for the case when normal vectors are not provided, thus it calculates $\mathbf{n}$ from the endpoints of the line segments as $\mathbf{n} = (\mathbf{x_{start}} \times \mathbf{x_{end}})/\|\mathbf{x_{start}} \times \mathbf{x_{end}}\|$. The unit direction vector of the line

in the world coordinate frame was computed similarly to the normal vector using the end points of the 3D line segments.

$\mathbf{R}_v$ is calculated with the known angles β and γ as described in Equation 2.4, where β is the rotation angle around Y and γ is the rotation angle around Z axis.

The a_i, b_i, c_i coefficients (2.8) were derived with Maple and we generated Matlab code with the CodeGeneration package. After constructing the third order equation (2.11) in q, it was solved and the real roots were selected. Thereafter, substituting the selected q into (2.5) we determine the rotation matrix $\mathbf{R}_X$ of the estimated absolute pose. This was implemented in Matlab as follows:

```
q = roots([a b c d]);
q = q(imag(q)==0);
Rx = [q^2+1,0,0;0,-q^2+1,-2*q;0,2*q,-q^2+1]/(1+q^2);
R = Rr * Rz * Ry * Rx;
```

For the translational components we had to solve a homogenous linear system of equations (2.9) in $\mathbf{t} = (t_1, t_2, t_3)^\top$ with singular value decomposition (SVD). As the last step after evaluating the reprojection error of each solution we chose the one with the smallest error. Our output is an absolute pose of the camera or camera system w.r.t. the world coordinate frame. To call the gPnLup function, use:

```
[pose_c] = gPnLup(camstruct)
```

The output of the algorithm is the estimated absolute pose of the camera or camera system whose structure is similar to (1.3):

$$\texttt{pose\_c} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The input of gPnLup is a structure whose representation is shown in Fig. 3.1. Note that the algorithm expects normalized coordinates on the unit sphere such as [20]. If our camera is a calibrated perspective camera and we have a matrix $\mathbf{I}$ of size 3xn which

camstruct.camdata

Fields	relpose_R	relpose_t	lines_3D_start	lines_3D_end	lines_2D_start	lines_2D_end
1	[0.2284 -0.1373 -...	[27.0203;0.1...	5x3 double	5x3 double	5x3 double	5x3 double
2	[1 0 0;0 1 0;0 0 1]	[0;0;0]	4x3 double	4x3 double	4x3 double	4x3 double
3	[0.9777 0.0320 0....	[-5.3758;0.1...	4x3 double	4x3 double	4x3 double	4x3 double
4	[1.0000 3.4694e-1...	[-7.1054e-1...	7x3 double	7x3 double	7x3 double	7x3 double

Figure 3.2. An example of the camstruct.camdata field of the input structure.

contains the corresponding 2D measurements with homogenous coordinate representation in the image plane then we transform the measurements using **K** calibration matrix (1.1) as follows:

```
temp = inv(K) * I;
I_norms = sqrt(sum(temp.*temp));
I_normalized = temp ./ repmat(I_norms,3,1);
```

In the omnidirectional camera case the extracted lines are already provided with spherical representation by the automatic line extraction toolbox of Bermudez [6].

In Fig. 3.2 we show an example of the `camstruct.camdata` for the Fig. 4.10 real data test case. Here, we can see that we used 4 cameras in our system, the second camera was choosen as the center of the camera system thus its relative position is set to zero. The relative pose of the other cameras are available as the system is fully calibrated. For this least squares estimation of the pose, we used 20 lines and each 2D line has its corresponding 3D line.

All the presented methods of this thesis are available in Annex with an example structure which corresponds to the real data presented in Fig. 4.10 and in Fig. 4.12. For the evaluation of our absolute pose estimation methods we used Matlab R2017a.

Chapter 4

Experimental Results

For the quantitative evaluation of our pose estimation algorithms with line correspondences, we generated various benchmark datasets of 3D–2D line pairs. Each dataset has 1000 samples. 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 unit cube. Then we applied a random translation of $0-1$unit and rotation of $0°, \ldots, 45°$ around the Z axis and $20°, \ldots, 60°$ around the vertical X axis to place the planes. All the parameters were inspired by common urban structural properties, in which environment the real data experiments were performed too.

The synthetic 2D images of the 3D lines were generated with virtual omnidirectional and perspective cameras. We applied the same rotation range of $-20°, \ldots, 20°$ around all three axes and random displacement of $0-3$ units. The known intrinsic parameters of the perspective cameras are $\mathbf{K}$: focal length $f_x = 846.1251$, $f_y = 846.1424$, and the principal point $\mathbf{o}$ which was set to the center of the image. In case of the omnidirectional camera, we used a virtual camera to generate omnidirectional images with size of 3.6MPx. The camera parameters were taken from a real 8mm Fish-eye camera calibration, calibrated with the calibration toolbox of Scaramuzza [37].

Separate datasets were generated by the combination of the perspective and omnidirectional cameras into a generalized camera system. In case of 3 camera systems, we used four different constructions which contains 0,1,2 or 3 omnidirectional cameras. For this setup, datasets were generated with random translation 0.4.

For our tests with multiview perspective cameras we generated datasets with perspec-

tive cameras for single, standard stereo, and multiview 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 unit. For the three-camera setup, datasets were generated with random translations corresponding to 0.05, 0.15, and 0.4 baselines. For all the generated multiview systems, we applied random relative rotation between the cameras around the Y and Z axis in the range of $-5^\circ, \ldots, 5^\circ$, and around the X axis $15^\circ, \ldots, 25^\circ$.

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%, 8%, 10% and 20%, 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 (see Fig. 4.8): $[-20, +20]$ 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.

Following the structure of chapter 2, we will present our results on central non-perspective camera systems and on multiview perspective cameras separately on the above mentioned synthetic datasets. The evaluation of the algorithms were done in two scenarios: 1) as a least square solver where all of the 3D–2D line pairs are used (about 30 line pairs per sample - this will be denoted by n); 2) as a minimal solver when only the minimal 3 line pairs are used (this will be denoted by 3).

4.1 Central non-perspective camera systems

In this section, we evaluate the accuracy of our gPnLup algorithm introduced in Section 2.2.1, in case of central camera systems which contain omnidirectional as well as perspective cameras. First, we study the limits of our algorithm in various scenarios and then we compare its performance in different noisy cases with state of the art methods [20] [22].

Fig. 4.1 shows the sensitivity of our gPnLup algorithm in the least square case with n lines for the composition of the camera systems. Results show that for n lines, our solver is slightly more accurate when we have less omnidirectional camera in our sys-

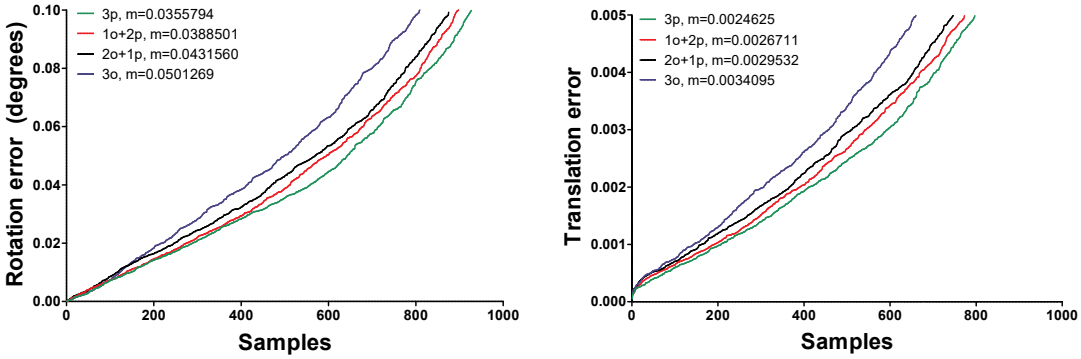


Figure 4.1. Efficiency of our gPnLup method in case of different camera system configurations (o:omnidirectional camera, p:perspective cameras, m:median error value). The left plot presents the rotation errors w.r.t. different camera system configurations, while the plot in the right shows translation errors.

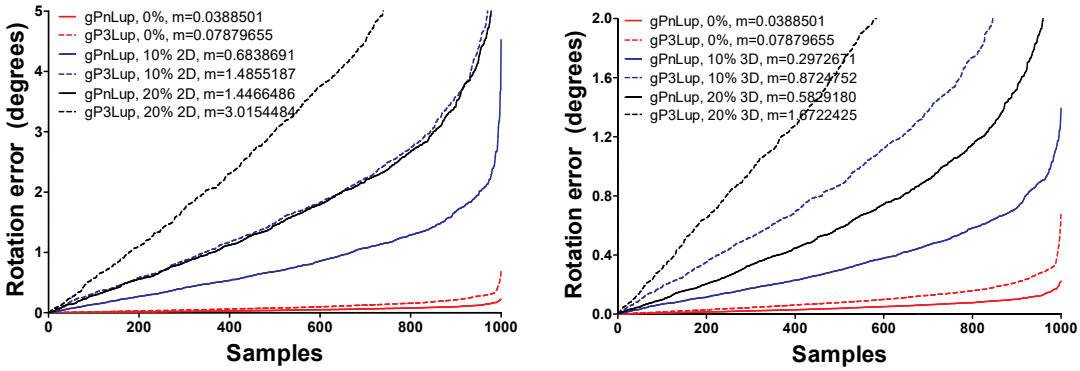


Figure 4.2. Comparison of our gPnLup method in case of varying line numbers and varying noise levels (10% and 20% noise level,m:median error value). The left plot indicates the efficiency of our minimal ($n = 3$) and least square solutions in case of 2D noise, while the plot in the right shows the results in case of 3D noise. In both of these cases we used one omnidirectional and two perspective cameras in our camera system.

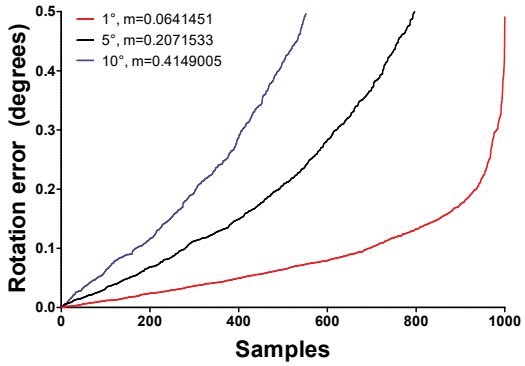


Figure 4.3. Comparison of the effect of various vertical direction errors on our gPnLup algorithm in case of $1°$, $5°$ and $10°$ using one omnidirectional and two perspective cameras.

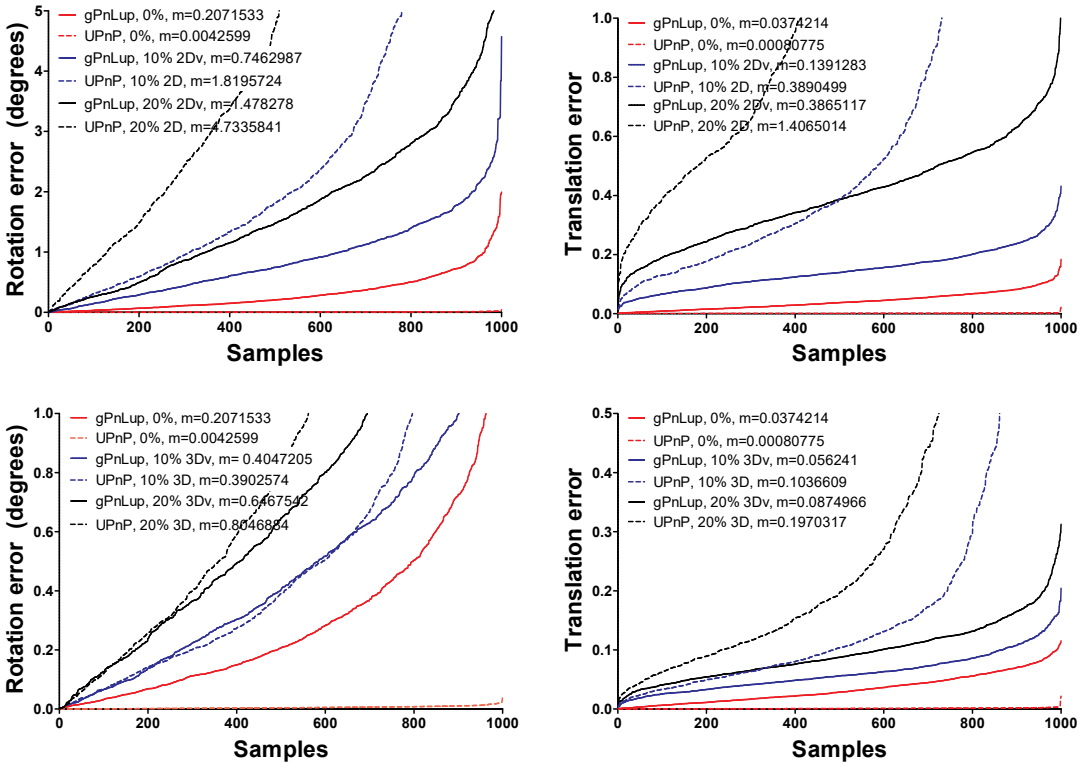


Figure 4.4. Comparison of the UPnP method with our gPnLup method in case of varying noise levels (2D: 10% and 20% 2D noise, 3D: 10% and 20% 3D noise, 2Dv: 10% and 20% 2D noise with 0.5° vertical noise, 3Dv: 10% and 20% 3D noise with 0.5° vertical noise, m:median error value). The first row presents the rotation and translation errors w.r.t. different 2D noise levels, while the second row shows errors w.r.t. 3D noise levels. In all of these cases we used one omnidirectional and two perspective cameras in our camera system.

tem due to their effective lower resolution. However, overall the method performs quite well independently of the type of construction, having a median rotation error less than 0.0501° in all test cases. Hereafter we use the 1 omnidirectional + 2 perspective camera configuration for our comparisons.

Next, we compare the performance in the minimal and n-line cases. Fig. 4.2 shows the rotational and translation error in case of n lines as well as 3 lines. Obviously, the algorithm performs better for n lines, but overall the estimates are quite accurate both in case of 2D and 3D noise.

Thereafter, we evaluate the noise sensitivity of gPnLup in comparison with other algorithms. Since to the best of our knowledge, there is no prior method for generalized pose estimation from line correspondences and known vertical direction, we compare our gPnLup method with the point-based non-perspective UPnP [20] algorithm of Kneip *et al.* .

In Fig. 4.3, we show the robustness of our algorithm in case of 1°, 5° and 10° vertical direction noise levels. Since for gPnLup the vertical direction is available, to have a fair comparison with the other state of the art methods we added a $\pm 0.5^\circ$ random noise to the vertical direction (this is above the typical noise level of a low quality IMU [1]).

With the noisy vertical direction we run comparative tests on the 10% and 20% noisy datasets both in case of 2D and 3D noise types. The results of these experiments are shown in Fig. 4.4, where we show error plots compared with UPnP [20]. UPnP is slightly more accurate for noiseless input, but it is consistently outperformed by our methods in every noisy case. The translation errors show us the same tendency, in the noisy cases our algorithm performs better. We note here that because the rotation and translation is decoupled, any error in the rotation is directly propagated into the linear system of the translation. Note as well, that in the 3 cameras case, UPnP has more than 150 point pairs to work with (we used 2 points per line), which is double than the number of correspondences for our algorithm. Finally, the typical runtime of our method was 9.8ms, while it was 8.8ms for UPnP [20] in case of n lines. CPU time of our method is slightly higher than UPnP [20], however our algorithm was implemented in MATLAB, while UPnP is implemented in C++.

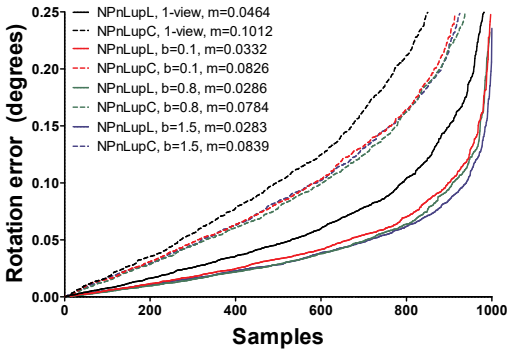


Figure 4.5. Comparison of the rotational errors of our NPnLup solvers w.r.t. the baseline for 30 lines in case of a single-view and stereo system configurations.

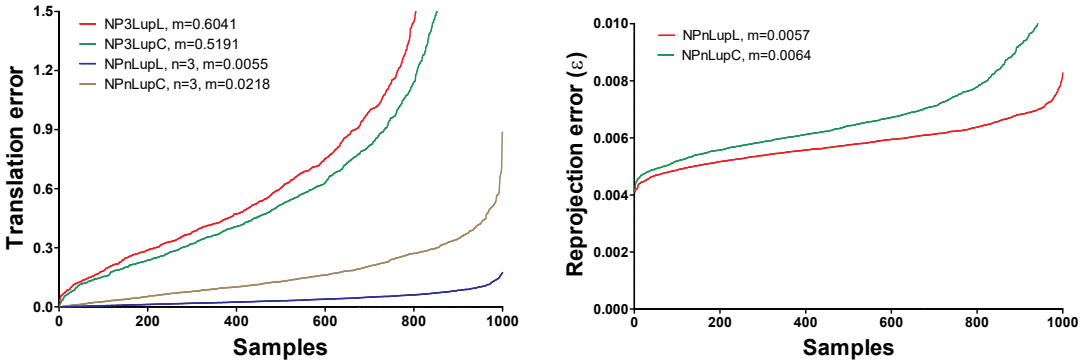


Figure 4.6. Left plot shows the translation errors in case of 3 camera for both n and 3 lines. The right plot shows the mean re-projection errors ϵ of the proposed NPnLup algorithms for n lines.

4.2 Multiview perspective cameras

Opposite to the previous section, here we only have perspective cameras in our system. We run similar tests in case of NPnLupL and NPnLupC algorithms Section 2.3 on multi-view perspective cameras. Test cases include various baselines, minimal and n-line cases as well as possible minimal case scenarios. Then, we perform comparative evaluation with state of the art methods [45] [20] [22], keeping the same principles in mind as in Section 4.1.

For better understanding of the behavior of our algorithms, we evaluate their sensitivity for the baseline in case of a standard stereo setup (parallel optical axes, only horizontal translation between cameras), which is the most challenging configuration as projection

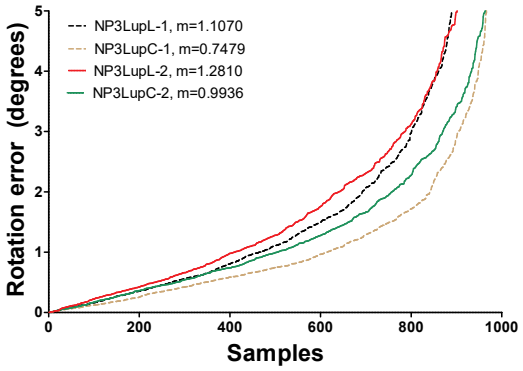


Figure 4.7. Comparison of two possible minimal case scenarios in a 3-camera system. Dashed lines: one 3D line and its corresponding 2D lines from each camera; continuous lines: a different 3D–2D line pair for each camera.

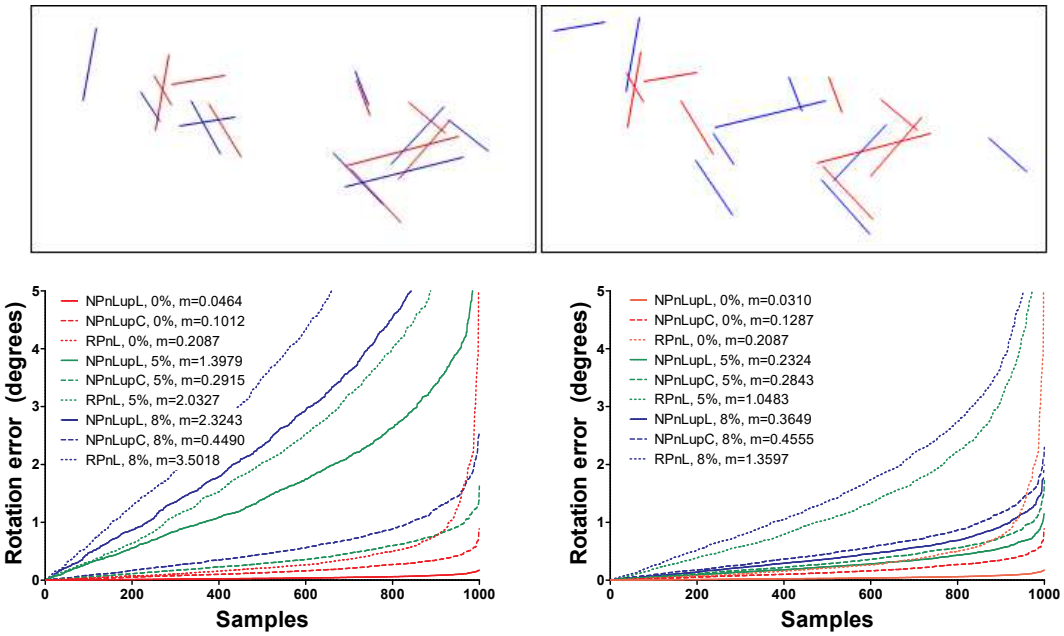


Figure 4.8. Comparison of the effect of various noise types (2D and 3D) and noise levels (5% and 8%) in a 3-camera system configuration. The first row illustrates a sample of our noisy synthetic data with 5% (left) and 8% noise level (right). Red lines are the originals while blue ones are the noisy lines. In the second row plots compare various state-of-the-art solvers' rotation errors w.r.t. different 2D and 3D noise levels.

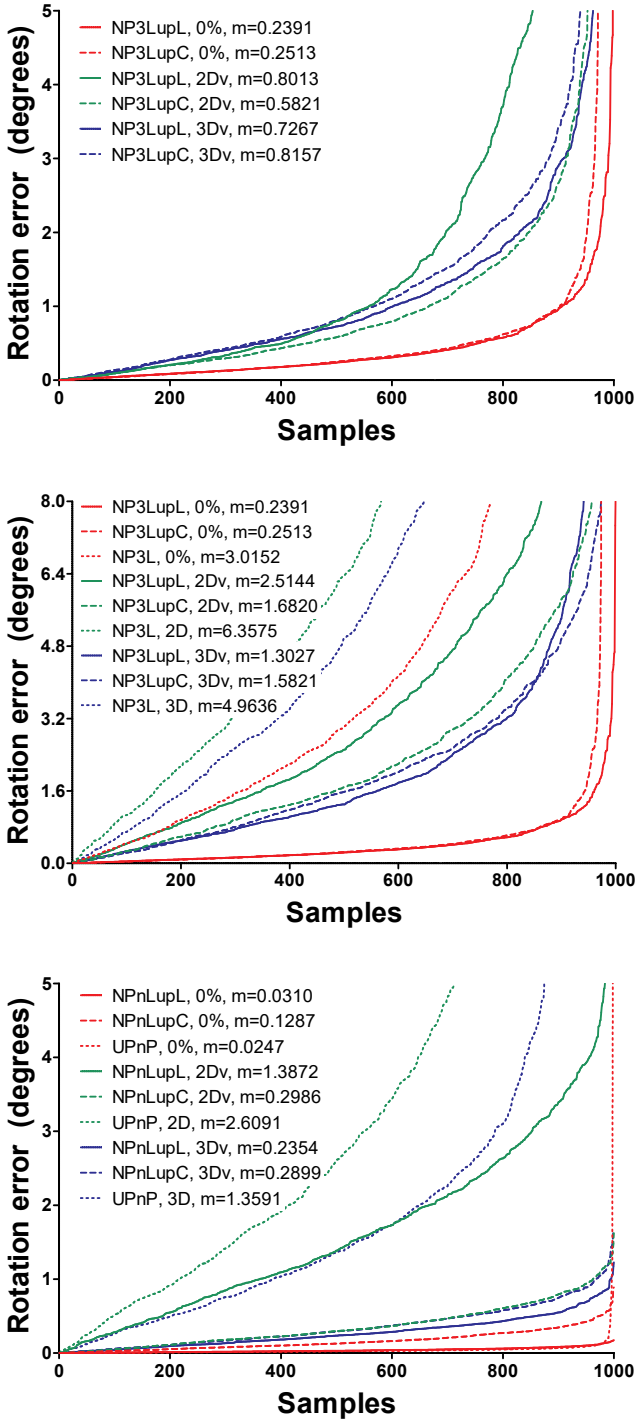


Figure 4.9. Comparison of various configurations and methods w.r.t varying line numbers and varying noise levels(2D: 5% 2D noise, 2Dv: 5% 2D noise with 0.5° vertical noise, 3D: 5% 3D noise, 3Dv: 5% 3D noise with 0.5° vertical noise, m:median error value). The first plot indicates the efficiency of our 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.

planes are nearly parallel for narrow baselines. Fig. 4.5 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° in all cases.

Then we compare the performance in the minimal and n-line cases. Fig. 4.6 shows the translation error in case of n lines as well as 3 lines. 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. 4.7, 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 form all cameras. The accuracy of our algorithms are not influenced by these differences.

After investigating the behavior of our methods, we made comparison with state of the art methods. Since to the best of our knowledge, there is no prior method for multiview perspective pose estimation from line correspondences and known vertical direction, we compare our NPnLup methods with the line-based single view RPnL algorithm [45] of Zhang *et al.* ; the point-based non-perspective UPnP [20] method of Kneip *et al.* ; and the line-based non-perspective minimal solver NP3L [22] of Lee.

In Fig. 4.8, we compared the robustness of the proposed algorithms in a 3-camera system. NPnLupL outperforms NPnLupC and RPnL at 0% and for all 3D noise levels, but for 2D noise NPnLupC 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.

The comparative results of the experiments are shown in Fig. 4.9, where we show error plots for the standard stereo setup minimal case, 3-camera system minimal case (compared with NP3L [22]), as well as 3-camera case with n-lines (compared with UPnP [20]. NP3L [22] is consistently outperformed by our methods, while for the UPnP [20] method our NPnLupC outperforms it in the noisy cases. Finally, Table 4.1 shows the typical runtime for all tested methods in case of 3 and n lines. Our algorithms were implemented in Matlab and run on a standard desktop computer.

n lines	NPnLupL	NPnLupC	RPnL
Run time (s)	0.0009	0.0013	0.0088

3 lines	NP3LupL	NP3LupC	UP3P	NP3L
Run time (s)	0.0004	0.0005	0.0063	0.0298

Table 4.1. Comparison of the run times of different methods w.r.t number of lines.

4.3 Real Data

Besides synthetic datasets, we evaluated the proposed methods also on the Komarom real test cases, where 2D perspective images were captured with a Canon 5D DSLR camera, while the 2D omnidirectional images were taken with a Canon EF 8-15mm f4L fisheye lens. The 3D point cloud was captured with a Riegl VZ400 Lidar scanner with an angular resolution of 0.05°.

The ground truth pose for Fig. 4.10 and Fig. 4.11 datasets was calculated using special Lidar markers placed on the building facades. The 3D location of these markers are read by the Lidar scanner, and their corresponding 2D locations are manually selected in each camera image. The role of these markers is two-fold: First, using these high quality 3D-2D point correspondences, UPnP [20] was used to calculate the reference pose for each camera image. Second, given an estimated pose, we can compute the forward projection of the 2D marker points onto the 3D surfaces and compute the (metric) error in 3D space. For the reference pose, the maximum of this *forward projection* error was 0.1205m, while the mean was 0.0677m. We detected 2D lines on the perspective images using the OpenCV LSD detector [13], while on the omnidirectional images we used the automatic line extraction toolbox of Bermudez [6]. 3D lines were extracted in a similar way on Riegl's own 2D RGB images used for colorizing the Lidar scan.

In Fig. 4.10, we have a calibrated camera system consisting of 3 perspective and 1 omnidirectional cameras. The evaluation of the pose obtained by our gPnLup algorithm is shown in Table 4.2. To characterize the accuracy, we compute the error from the markers' location.

Furthermore, we prepared a comparative test where we compare our NP3LupC result

	Fig. 4.10	Fig. 4.11	
	gPnLup	NPnLupC	NP3L [22]
Rotation error (deg)	1.1972	1.0216	4.5029
Translation error	0.8797	0.9088	3.0037
Forward projection error(m)	0.2407	0.2904	0.5901

Table 4.2. Comparison of the maximal rotational, translational and forward projection error of various methods on the real datasets shown in Fig. 4.10 and Fig. 4.11.

with the NP3L minimal solver of Lee [22] on the Komarom dataset. The MATLAB implementation of NP3L provided by the author of [22] works with three perspective cameras and a total of three line correspondences, hence in Fig. 4.11 we show results for such a camera system. As it is shown in Table 4.2, our algorithm outperforms NP3L in spite of the fact, that the input vertical direction for our algorithm had a 1.198° deviation from the ground truth. It is thus fair to say that NPnLupC provides state-of-the-art estimates under real conditions.

To evaluate our NPnLup methods with perspective camera system we also used the Kolozsnema real dataset presented in Fig. 4.12. 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 4.3 in comparison with UPnP [20].

RPnL is not included in these tests because it works only for a single camera while in these test case we had a multiview camera system of three cameras. As mentioned above about NP3L [22], we could not run it with n-lines. Evaluation on synthetic data already shown the performance of our method compared to RPnL and NP3L. The purpose of this test was to show that our line-based method is able to provide state-of-the-art estimates under real conditions, just like the point-based UPnP. Although our translation error is approximately two times larger than for UPnP, its rotation error is almost four times bigger than ours. Furthermore, UPnP was using two times more correspondences (two endpoints of each line) than our method. It is thus fair to say, that both methods perform pretty well, as the errors are almost negligible (see Table 4.3).

Figure 4.10. Lidar laser scan for testing our gPnLup pose estimation algorithm with a 3-perspective-1-omnidirectional multi-view camera system. The extracted 2D lines are shown on the 2D images, while on the Lidar scan (second row) red dots are the estimated positions and green dots are the real location of the markers in metric 3D space.

Fig. 4.12	NPnLupL	NPnLupC	UPnP
Rotation error (deg)	0.0166	0.0176	0.0498
Translation error	0.0402	0.0319	0.0119

Table 4.3. Comparison of the maximal rotational and translational error of various methods on the real data presented in Fig. 4.12.

Figure 4.11. Lidar laser scan for testing our NP3LupC pose estimation algorithm in minimal case with 3-perspective camera system. The extracted 2D lines are shown on the 2D images (first row). On the Lidar scan (second row), red dots are the estimated positions of our minimal solver, blue dots are the estimated positions of NP3L [22], and green dots are the real location of the markers in the 3D metric space.

Figure 4.12. Lidar laser scan for testing our NPnLup pose estimation algorithms with a 3-camera system. 2D detected lines are shown next to the 3D point cloud whose colors are the same as the corresponding camera.

Chapter 5

Conclusions

We proposed a direct least squares solution to the gPnL problem from line correspondences with known vertical direction. The only assumption about our generalized camera is that 3D lines project through projection planes. Many practically important camera setup corresponds to this model: stereo and multiview central camera systems composed of perspective and/or non-perspective (*e.g.* omnidirectional) cameras, or a camera (system) moving along a trajectory.

For the important special case with only perspective cameras, the so called NPnL problem, we proposed a linear and a cubic solution using line correspondences with known vertical direction. The minimal number of line correspondences has been discussed for various common camera configurations. 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.

All methods can be used as a minimal solver (*e.g.* within RANSAC) as well as a general least squares solver without reformulation. The methods work for camera system with one or more cameras as well. The proposed methods have been evaluated on synthetic and real datasets. Comparative tests confirm state of the art performance both in terms of quality and computing time.

Appendix

Kamera-rendszer helyzetének meghatározása 3D-2D egyenes-megfeleltetések és vertikális irány alapján

Diplomamunka Tartalmi Összefoglalója

A helyzet meghatározási probléma alapját képzi számos vizuális információn alapuló technikának pl.: vizuális odometria, kép alapú lokalizáció és navigáció, fúzió és kibővített valóság. Mi a kamerarendszer helyzetének meghatározásával foglalkozunk, ami magában foglalja a helyzet illetve orientáció becslését a 3D világ koordináta rendszerhez viszonyítva.

Modern alkalmazások, főleg a robotikában és önjárójárművekben alkalmazott kép alapú lokalizáció és navigáció esetén sokszor elengedhetetlen, hogy olyan kamerarendszer álljon rendelkezésünkre melynek nagy a látószöge. Nem csak klasszikus kép alapú technikák segítségével, mint a Structure from Motion (SfM) technikával tudunk 3D mérésekkel szolgálni az adott helyről, de modern szenzorokkal (pl.: Lidar, Kinect) 3D struktúrákat követlenül tudunk felvenni. Emiatt a 3D adatok egyre szélesebb körben elérhetőek így a helyzetbecslő algoritmusok, melyek együttesen használják a kamerarendszer 2D méréseit a rendelkezésre álló 3D adattal előtérbe kerültek.

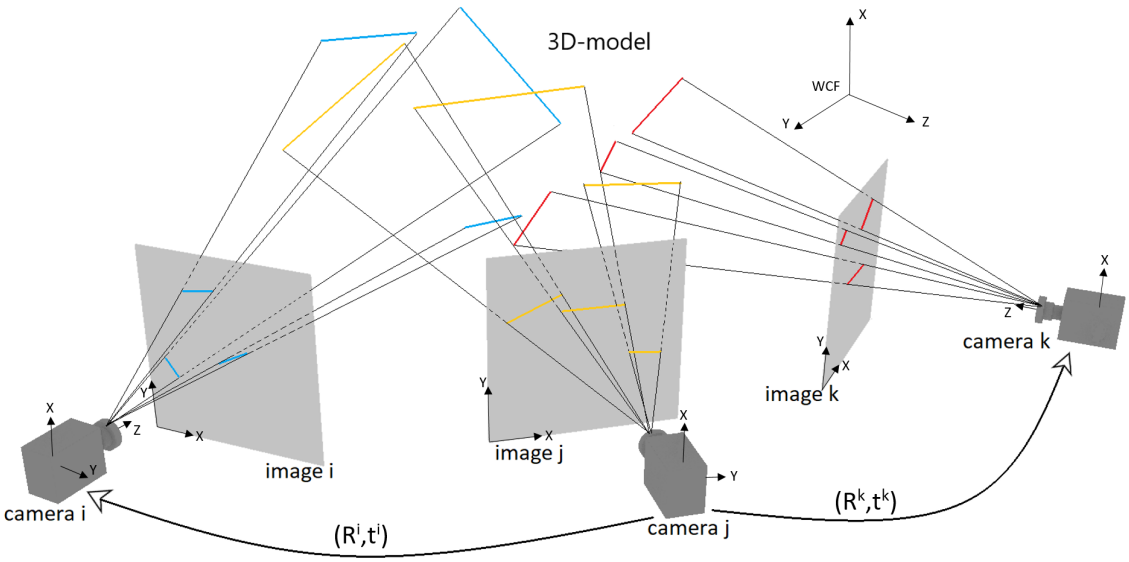
Miután a modern kamerák rendszerint fel vannak szerelve különböző helyzet és orientáció szenzorokkal, azt feltételezzük, hogy a vizsgált kamerarendszer vertikális iránya (pl.: gravitációs vektor) számunkra ismert.

Munkánk során a kamerarendszer helyzetének becslését általánosított kamerák esetén egyenes vonalak alapján számítottuk, melyek gyakoriak a városi környezetben. Az egyetlen feltételezés a képalkotó rendszerről, hogy a 3D egyenesek vetítő síkkal vetülnek le, melyet az egyenes és a kamera vetítési iránya határoz meg. Ezért a problémát négy ismeretlennel fejeztük ki és vezettük vissza zárt alakra, melyhez 3D egyenes - vetítősík megfeleltetéseket használtunk.

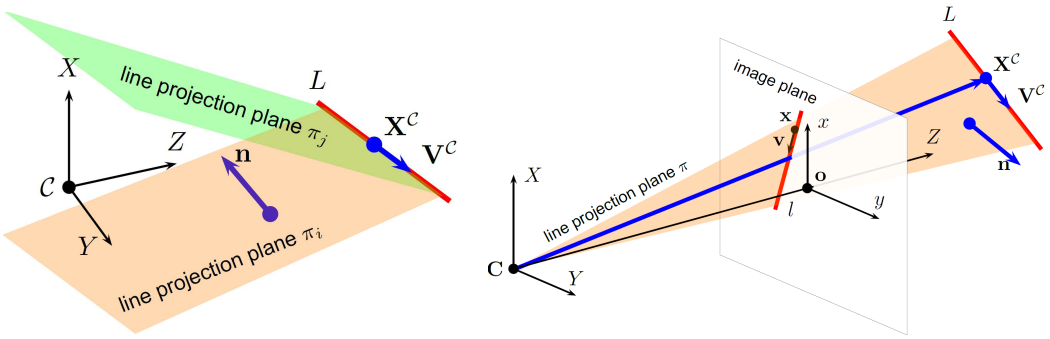
Ennek fontos speciális részesetével, a kalibrált perspektív kamerarendszer helyzetének becslésével foglalkozunk 3D-2D egyenes megfeleltetések alapján. Erre két megoldási módszert javasoltunk: az első megoldás egy lineáris egyenletrendszerhez vezet, míg a másik esetben egy harmadfokú és egy elsőfokú egyenletrendszert kell megoldanunk, mely zárt alakban hatékonyan elvégezhető.

Az alábbiakban ismertetjük a dolgozat tartalmát fejezetenkénti bontásban. A dolgozat tartalma két nemzetközi konferencián megjelent publikáción alapszik [17, 18], melyet öt fejezetben tárgyalunk.

A bevezetésben legelőször a pózbecslést és annak matematikai hátterét tárgyaljuk. Bemutatjuk a felhasznált koordináta rendszereket és azok közötti összefüggéseket (lásd Fig. 5.1), majd bevezetünk számos elengedhetetlen definíciót a kamerákra vonatkozóan, melyet későbbiekben használni fogunk.



5.1. ábra. A koordinátarendszerek és a 3D-2D vonalak megfeleltetése reprezentációja több kamerás rendszer esetén. A 3D-2D egyenes megfeleltetéseket azonos színnel jelöljük.



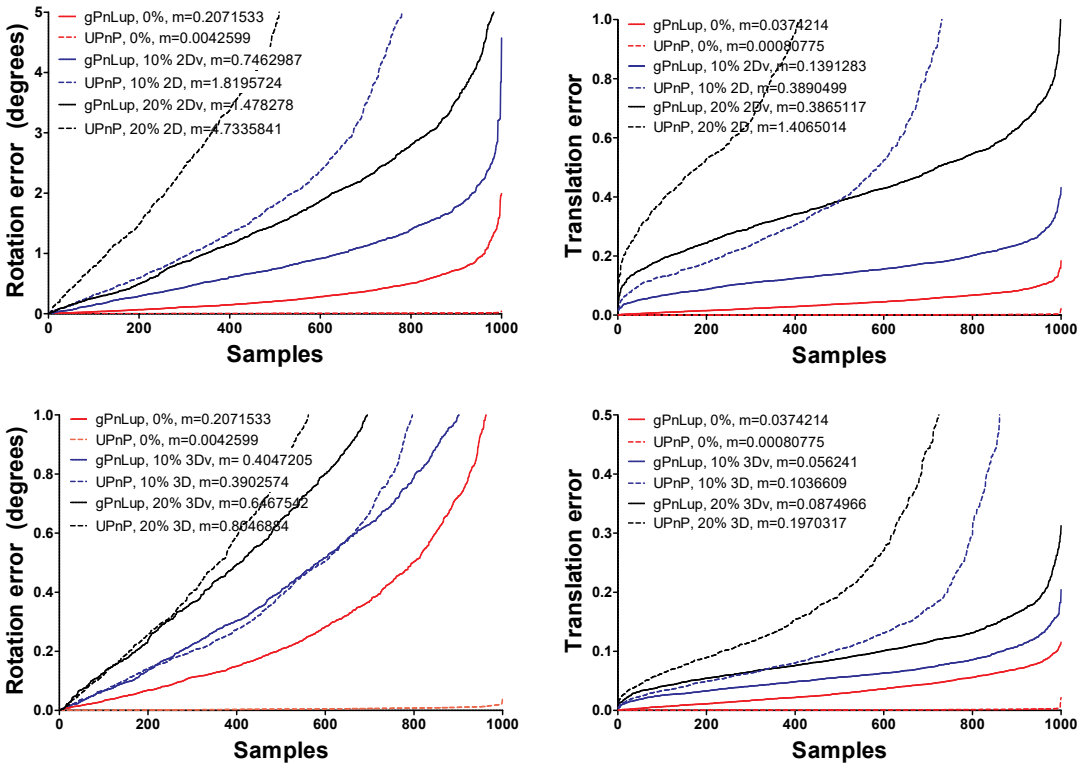
5.2. ábra. Egyenes vetítés reprezentációja általános (bal oldalon) és perspektív kamerák (jobb oldalon) esetén.

A kamera pózbecslés célja $[\mathbf{R}|\mathbf{t}]$ mátrix kiszámítása, mely egy olyan 3×4-es mátrix, ahol $\mathbf{R}$ egy 3×3-as forgató mátrix ami meghatározza a kamera orientációját α, β, γ szögekkel, míg $\mathbf{t}$ az eltolási vektor ami a világ és a kamera koordináta rendszerek között hat. Számítógépes látáson alapuló applikációk ezen $[\mathbf{R}|\mathbf{t}]$ mátrix meghatározására fókuszálnak, mely megfelel egy euklideszi transzformációnak a világ koordináta rendszerből a kamera koordináta rendszerébe.

Ezután a gyakori képi jellemzők kinyerését valamint megfeleltetését tárgyaljuk, melyek fontos szerepet játszanak számos felhasználási területen. Ebben a dolgozatban mi 3D-2D egyenes-megfeleltetések segítségével becsüljük meg a kamera-rendszer helyzetét. Módszereinket azonban az egyenes alapú eljárások (PnL) [22, 45] mellett olyan algoritmusokkal is összehasonlítottuk a negyedik fejezetben, melyek pontok alapján számolnak (PnP) [20].

Az általános bevezetés után a kamera rendszerek sajátosságát figyelembe véve megkülönböztettünk általános kamerákból álló rendszereket, valamint perspektív kamerákból álló rendszereket. Ezutóbbi a leggyakrabban használt példája az általánosított kamera-rendszereknek, ezért ezt az esetet külön alfejezetben tárgyaljuk.

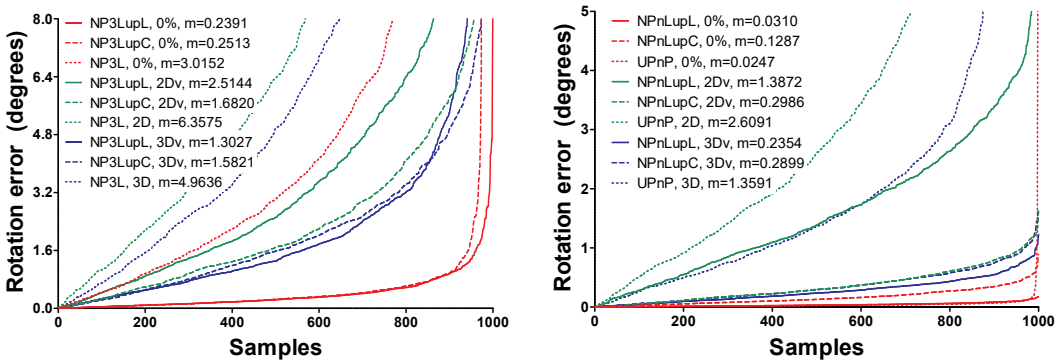
A második fejezetben az általunk javasolt algoritmusok (gPnLup (2.2.1), NPnLupC (2.3) és NPnLupL (2.3)) levezetését mutatjuk be. A bevezetéshez hasonlóan itt is külön választjuk az általánosított modellt a centrális illetve perspektív kameráktól. Mindkét kamera-rendszer esetében tárgyaljuk többek között az egyenesek vetítését, mely kulcsfontosságú az algoritmushoz használt egyenletek leveztéséhez (lásd Fig. 5.2). A módszerek az egyenes-megfeleltetések mellett feltételezik a vertikális irány ismeretét, melyet például



5.3. ábra. Az UPnP módszer összehasonlítása a mi gPnLup módszerünkkel különböző zajszintek esetén (2D: 10% és 20% 2D zaj, 3D: 10% és 20% 3D zaj, 2Dv: 10% és 20% 2D zaj $0.5°$ vertikális zajjal, 3Dv: 10% és 20% 3D zaj $0.5°$ vertikális zajjal, m:medián hiba értéke). Az első sorban a forgatási illetve az eltolási hibákat mutatjuk be különböző 2D zaj szintekre vonatkozóan, míg a második sorban a 3D zajszinteket változtatjuk a kiértékelésnél. Minden tesztesetben egy omnidirekcionális és két perspektív kamerát tartalmazott a kamera-rendszer.

egy Inerciális Mérőegység (IMU) segítségével tudunk mérni. Ez az előzetes tudás az eredetileg 6 szabadsági fokkal rendelkező helyzet meghatározási problémát 4-re csökkenti (levezetést lásd 2.2.1 fejezetben).

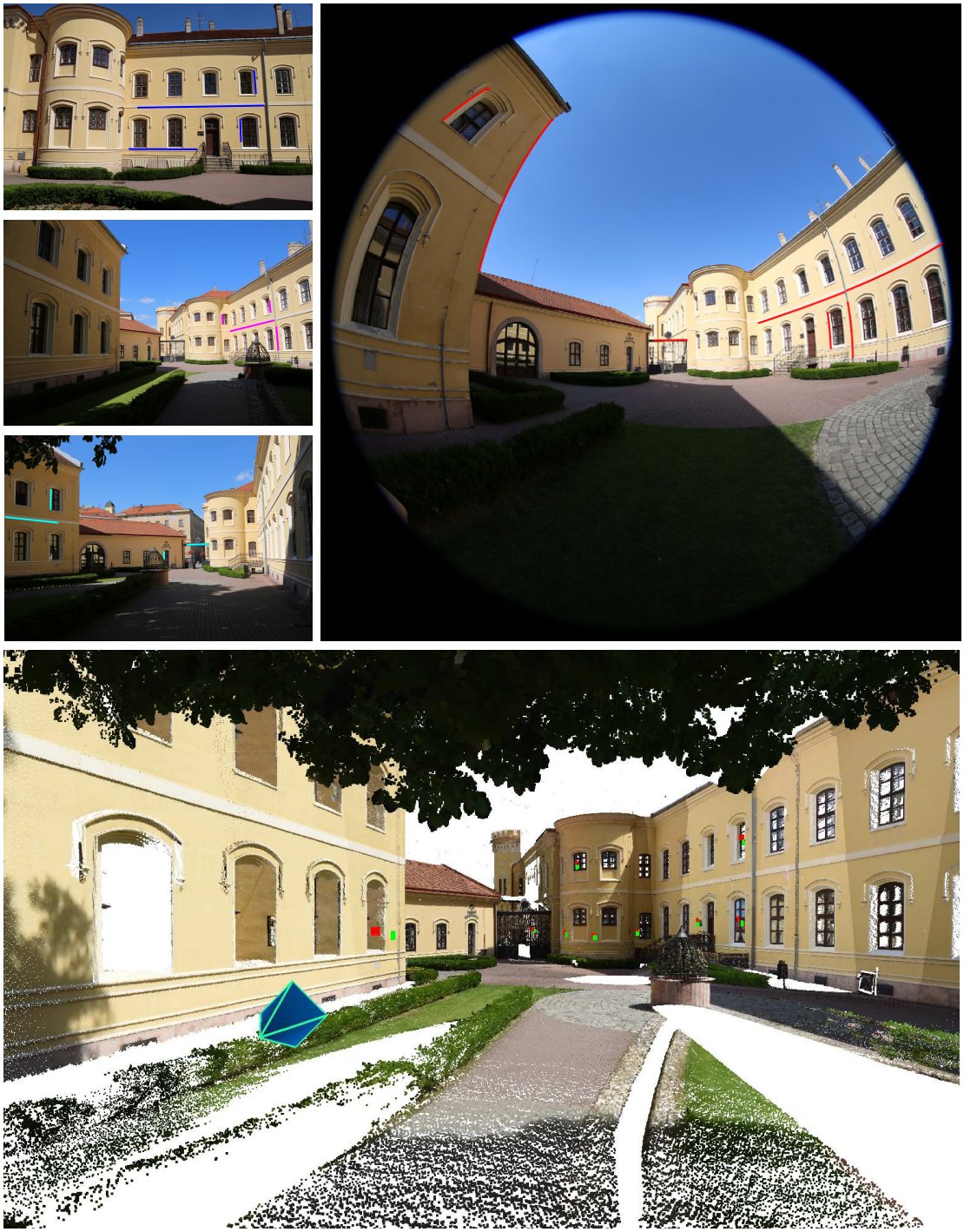
A bemutatott hatékony megoldások implementációját a harmadik fejezetben tárgyaljuk. Az algoritmusok összefoglalójain túl részletesebben csak a legkomlexebb gPnLup algoritmust mutatjuk be, mivel ennek ismeretében a másik kettő könnyen értelmezhető hasonló működési elvük miatt. Ebben a részben leírjuk, hogy hogyan kell az algoritmust meghívni, milyen input adat szükséges a számításhoz, milyen beépített Matlab függvényeket használunk fel a megoldás során valamint, hogy a kapott outputot hogyan kell értelmezni. Minden algoritmus Matlab kódja megtalálható a Mellékletben a Fig. 4.10 és Fig. 4.12 valós adatokhoz tartozó minta struktúrákkal.



5.4. ábra. Különböző konfigurációk és módszerek összehasonlítása különböző számú egyenes-megfeleltetés illetve zajszint függvényében (2D: 5% 2D zaj, 2Dv: 5% 2D zaj 0.5° vertikális zajjal, 3D: 5% 3D zaj, 3Dv: 5% 3D zaj 0.5° vertikális zajjal, m:medián hiba értéke). Az első ábra a mi NPnLup megoldóink eredményét hasonlítja össze az NP3L minimális megoldóval (3 egyenespár figyelembe vételével) három kamera esetén. A jobb oldali ábra pedig a UPnP módszerrel hasonlítja össze a mi algoritmusainkat három kamera esetén minden egyenest felhasználva.

Miután mind az elméleti hátteret illetve a gyakorlati megvalósítás részleteit ismertettük a negyedik fejezetben bemutatjuk a módszerekkel elért eredményeket szintetikus illetve valós adaton egyaránt valamint összehasonlítjuk a hatékonyságukat a jelenlegi legkorszerűbb módszerekkel. A szintetikus tesztekhez több, ezer mintát tartalmazó szintetikus adathalmazt generáltunk mind virtuális omnidirekcionális mind perspektív kamerákkal, melynek menetét részletesen ismertetjük. Ezen kamerák kombinálásával külön adathalmazt generáltunk általánosított kamera-rendszerek helyzet-meghatározásának teszteléséhez. Az algoritmus viselkedését szélsőséges körülmények között különböző minőségű és mértékű zajjal terhelt adatokon teszteltük, melynek eredményeit részletesen bemutatjuk. Ezen tesztek eredményei hasonlóan az előző fejezetekhez a kamerák típusa szerint vannak tagolva.

A centrális nem-perspektív kamera-rendszereken végzett összehasonlító eredményeket a 5.3. ábra foglalja össze. A gPnLup algoritmust ebben az esetben a pont alapú UPnP módszerrel [20] hasonlítottuk össze mivel legjobb tudásunk szerint eddig nem állt rendelkezésre olyan korábbi módszer, amely megbecsülné az általánosított kamera helyzetét egyenes megfeleltetések illetve ismert vertikális irány alapján. Annak érdekében, hogy az összehasonlításunk megfelelő legyen egy olyan módszerrel, amelynek nem áll rendelkezésére a vertikális irány az ismert szögeket minden esetben az IMU szenzor átlagos hiba



5.5. ábra. Lidar lézer szken a gPnLup helyzetbecslő algoritmusunk tesztelésére 3-perspektív-1-omnidirekcionális kamera-rendszer esetén. A kinyert 2D egyenesek a 2D képeken vannak ábrázolva, míg a második sorban a lidar szkenen pirossal a becsült pozíciója, zölddel a valós helyzete látható a markereknek a 3D metrikus térben.

5.6. ábra. Lidar lézer szken az NPnLup helyzetbecslő algoritmusunk tesztelésére 3 kamerás rendszer esetén. 2D detektált egyenesek a kamerájuknak megfelelő színnel vannak a 3D pontfelhőn megjelenítve.

mértékével zajosítottuk meg.

Hasonló összehasonlítást végeztünk multiview perspektív kamerák esetén is, melyet a 5.4. ábrán mutatunk be. Ebben az esetben az NPnLupC és NPnLupL algoritmusainkat hasonlítottuk össze az UPnP [20] módszerrel valamint az egyenes alapú NP3L [22] algoritmussal. Az NP3L módszerhez a forráskódot a szerző Gim Hee Lee bocsájtotta rendelkezésünkre, mely csak minimális számú egyenespárral valamint csak 3 perspektív kamera esetén működik.

A szintetikus tesztek után az algoritmusainkat Fig. 5.5 és Fig. 5.6 valós adatokon is teszteltük, melyeknek eredményét a Táblázat 4.2 és Táblázat 4.3 táblázatok foglalják össze. A tesztek eredménye alátámasztja a state of the art teljesítményt mind minőség, mind a számítási idő (lásd Table 4.1) tekintetében. Minden javasolt megoldás módosítás nélkül használható minimális illetve legkisebb négyzetek megoldóként is.

Kulcsszavak: kamerarendszer helyzet meghatározása, vertikális irány, egyenes megfeleltetések, általánosított kamera, multiview kamerarendszer

Declaration

I, Nóra Horányi, student, declare that my dissertation was made at the Institute of Informatics of the University of Szeged, Department of Image Processing and Computer Graphics in order to obtain a Master degree in Info-bionics Engineering.

I declare that I have not presented this thesis for other degrees before, and that I used only my own work, and the sources mentioned (literature, tools, etc.).

I acknowledge that my diploma work will be located at the library of the Institute of Informatics of the University of Szeged, among the reference books.

Szeged, December 5, 2017 . . . . . .

signature

Acknowledgement

This work was partially supported by the "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002; 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 Research & Development Operational Programme for the project "Modernization and Improvement of Technical Infrastructure for Research and Development of J. Selye University in the Fields of Nanotechnology and Intelligent Space", ITMS 26210120042, co-funded by the European Regional Development Fund.

I would like to express my sincere gratitude to my supervisor Prof. Zoltan Kato, for the continuous support of my research, for his patience and to always be there for me. I thank my fellow labmates in Research Group on Visual Computation: László Körmöczi, Róbert Fröchlich and Hichem Abdellali for the stimulating discussions, and for all the fun we have had in the last one year. Thank you for helping me in the difficult times.

I would like to thank to my partner to encourage me to start something new, to always bring out the best in me and to believe in me. Thank you for all your support and love. I would like to thank to my mother, to my father and to my little sister all the support and the endless love that I received every day. Thank you to always stand by me and to believe in me and in my dreams. You are the most wonderful people in the world! Thank you for Manyika to always be proud of me and love me. Thank you to be my grandmother.

With the most special and sincere thanks for my only angel who protects our family and always directs my steps from above.

"It is only with the heart that one can see rightly.
What is essential is invisible to the eye."
Antoin de St Exupery

Köszönetnyilvánítás

Szeretném kifejezni őszinte hálámat Kató Zoltánnak a folyamatos támogatásáért a kutatásomban, a türelméért és hogy mindig lehetett rá számítani, amikor szükségem volt rá.

Köszönöm a munkatársaimnak az RGVC csoportban: Körmöczi Lászlónak, Fröchlich Róbertnek és Abdellali Hichemnek az ösztönző beszélgetésekért és minden boldog és vicces pillanatért az elmúlt egy évben. Köszönöm, hogy segítettetek nekem a nehéz időkben.

Köszönöm a páromnak, hogy bátorított arra, hogy valami újat kezdjek, hogy mindig kihozta belőlem a legjobbat és hogy hitt bennem. Köszönet a támogatásodért és a szeretetért amit kaptam tőled.

Szeretném megköszönni anyukámnak, apukámnak és a húgomnak a támogatást és a végtelen szeretet amit kaptam minden nap. Köszönöm, hogy mindig mellettem álltok és hogy hisztek bennem és az álmaimban. Ti vagytok a legcsodálatosabb emberek a világon!

Köszönet Manyikának, hogy mindig büszke volt rám és szeretett engem. Köszönöm, hogy a nagymamám voltál.

Külön és őszinte köszönettel az egyetlen angyalomnak, aki fentről óvja a családunkat és mindig irányt mutat nekem.

"Jól csak a szívével lát az ember.
Ami igazán lényeges, az a szemnek láthatatlan."
Antoin de St Exupery

Bibliography

[1] Cenek Albl, Zuzana Kukelova, and Tomás Pajdla. Rolling shutter absolute pose problem with known vertical direction. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 3355–3363, Las Vegas, NV, USA, June 2016.

[2] Clemens Arth, Manfred Klopschitz, Gerhard Reitmayr, and Dieter Schmalstieg. Real-time self-localization from panoramic images on mobile devices. In *Proceedings of International Symposium on Mixed and Augmented Reality*, pages 37–46, Basel, Switzerland, October 2011. IEEE Computer Society.

[3] Simon Baker and Shree K. Nayar. A Theory of Single-Viewpoint Catadioptric Image Formation. *International Journal of Computer Vision*, 35(2):175–196, 1999.

[4] Adrien Bartoli and Peter Sturm. The 3D line motion matrix and alignment of line reconstructions. *International Journal of Computer Vision*, 57(3):159–178, 2004.

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[6] J. Bermudez-Cameo, G. Lopez-Nicolas, and J. J. Guerrero. Automatic line extraction in uncalibrated omnidirectional cameras with revolution symmetry. *International Journal of Computer Vision*, 114(1):16–37, August 2015.

[7] Federico Camposeco, Torsten Sattler, and Marc Pollefeys. Minimal solvers for generalized pose and scale estimation from two rays and one point. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Proceedings of European Conference Computer Vision*, volume 9909 of *Lecture Notes in Computer Science*, pages 202–218, Amsterdam, The Netherlands, October 2016. Springer.

[8] Manmohan Krishna Chandraker, Jongwoo Lim, and David J. Kriegman. Moving in stereo: Efficient structure and motion using lines. In *International Conference on Computer Vision*, pages 1741–1748, Kyoto, Japan, October 2009.

[9] Homer Chen. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):530–541, 1991.

[10] B. Fan, F. Wu, and Z. Hu. Line matching leveraged by point correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 390–397, June 2010.

[11] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[12] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems. In *European Conference on Computer Vision*, pages 445–462, Dublin, Ireland, June 2000.

[13] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: a Line Segment Detector. *Image Processing On Line*, 2:35–55, 2012.

[14] Micheal D Grossberg and Shree Nayar. A general imaging model and a method for finding its parameters. In *International Conference on Computer Vision*, pages 108–115, 2001.

[15] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2004.

[16] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Camera-IMU-based localization: Observability analysis and consistency improvement. *The International Journal of Robotics Research*, 33(1):182–201, 2014.

[17] Nora Horanyi and Zoltan Kato. Generalized pose estimation from line correspondences with known vertical direction. In *International Conference on 3D Vision*, Qingdao, China, October 2017. IEEE.

[18] Nora Horanyi and Zoltan Kato. Multiview absolute pose using 3D - 2D perspective line correspondences and vertical direction. In *Proceedings of ICCV Workshop on Multiview Relationships in 3D Data*, Venice, Italy, October 2017. IEEE.

[19] Juho Kannala and Sami S. Brandt. A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.

[20] Laurent Kneip, Hongdong Li, and Yongduek Seo. UPnP: an optimal O(n) solution to the absolute pose problem with universal applicability. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Proceedings of European Conference Computer Vision, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 127–142, Zurich, Switzerland, September 2014. Springer.

[21] Zuzana Kukelova, Martin Bujnak, and Tomáš Pajdla. Closed-form solutions to minimal absolute pose problems with known vertical direction. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Proceedings of Asian Conference on Computer Vision, Part II*, volume 6493 of *LNCS*, pages 216–229, Queenstown, New Zealand, November 2010. Springer.

[22] Gim Hee Lee. A minimal solution for non-perspective pose estimation from line correspondences. In *Proceedings of European Conference on Computer Vision*, pages 170–185, Amsterdam, The Netherlands, October 2016. Springer.

[23] Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 2746–2753, Portland, OR, USA, June 2013.

[24] V. Lepetit, F.Moreno-Noguer, and P.Fua. EPnP: an accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2), 2009.

[25] S. Li, C. Xu, and M. Xie. A robust O(n) solution to the perspective-n-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1444–1450, 2012.

[26] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[27] Branislav Mičušík. *Two-View Geometry of Omnidirectional Cameras*. Phd thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic, June 2004.

[28] Branislav Mičušík and Tomáš Pajdla. Para-catadioptric Camera Auto-calibration from Epipolar Geometry. In *Asian Conference on Computer Vision*, volume 2, pages 748–753, Seoul, Korea South, January 2004.

[29] P. Miraldo, H. Araujo, and N. Goncalves. Pose estimation for general cameras using lines. *IEEE Transactions on Cybernetics*, 45(10):2156–2164, October 2015.

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

[31] Faraz M Mirzaei and Stergios I Roumeliotis. Optimal estimation of vanishing points in a Manhattan world. In *International Conference on Computer Vision*, pages 2454–2461, Barcelona, Spain, November 2011. IEEE, IEEE Computer Society.

[32] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition*, volume 1, pages 1–8, Washington, DC, USA, June 2004. IEEE.

[33] Robert Pless. Using many cameras as one. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2003.

[34] H. Pottmann and J. Wallner. *Computational Line Geometry*. Mathematics and Visualization. Springer, 2009.

[35] Bronislav Pribyl, Pavel Zemcík, and Martin Cadík. Camera pose estimation from lines using Plücker coordinates. In Xianghua Xie, Mark W. Jones, and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference*, pages 45.1–45.12, Swansea, UK, September 2015. BMVA Press.

[36] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion. In *International Conference on Computer Vision Systems*, pages 45–51, Washington, USA, January 2006.

[37] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A Toolbox for Easily Calibrating Omnidirectional Cameras. In *International Conference on Intelligent Robots*, pages 5695–5701, Bejing, October 2006.

[38] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3D. In *ACM SIGGRAPH*, pages 835–846, Boston, Massachusetts, 2006. ACM.

[39] Levente Tamas, Robert Frohlich, and Zoltan Kato. Relative pose estimation and fusion of omnidirectional and lidar cameras. In Lourdes de Agapito, Michael M. Bronstein, and Carsten Rother, editors, *Proceedings of the ECCV Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving*, volume 8926 of *Lecture Notes in Computer Science*, pages 640–651, Zurich, Switzerland, September 2014. Springer.

[40] Levente Tamas and Zoltan Kato. Targetless calibration of a lidar - perspective camera pair. In *Proceedings of ICCV Workshop on Big Data in 3D Computer Vision*, pages 668–675, Sydney, Australia, December 2013. IEEE, IEEE.

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

[42] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[43] 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*, 2016.

[44] Lilian Zhang and Reinhard Koch. Hand-held monocular SLAM based on line segments. In *Proceedings of the Irish Machine Vision and Image Processing Conference*, pages 7–14, Dublin, Ireland, 2011. IEEE Computer Society.

[45] Lilian Zhang, Chi Xu, Kok-Meng Lee, and Reinhard Koch. Robust and efficient pose estimation from line correspondences. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, *Proceesings of Asian Conference on Computer Vision*, volume 7726 of *Lecture Notes in Computer Science*, pages 217–230, Daejeon, Korea, November 2012. Springer.