Lappeenranta University of Technology
School of Industrial Engineering and Management
Degree Program in Computer Science

Bachelor's Thesis

Lauri Rapo

FEATURE-BASED CAMERA POSE ESTIMATION

Examiners: Prof., D.Sc. (Tech.) Lasse Lensu

Supervisor: Prof., D.Sc. (Tech.) Lasse Lensu

ABSTRACT

Lappeenranta University of Technology
School of Industrial Engineering and Management
Degree Program in Computer Science

Lauri Rapo

Feature-based camera pose estimation

Bachelor's Thesis

2016

29 pages, 10 figures, 2 tables.

Examiners: Prof., D.Sc. (Tech.) Lasse Lensu

Keywords: computer vision, traffic sign, localization, feature extraction, feature tracking, pose estimation

The estimating of the relative orientation and position of a camera is one of the integral topics in the field of computer vision. The accuracy of a certain Finnish technology company's traffic sign inventory and localization process can be improved by utilizing the aforementioned concept. The company's localization process uses video data produced by a vehicle installed camera. The accuracy of estimated traffic sign locations depends on the relative orientation between the camera and the vehicle. This thesis proposes a computer vision based software solution which can estimate a camera's orientation relative to the movement direction of the vehicle by utilizing video data. The task was solved by using feature-based methods and open source software. When using simulated data sets, the camera orientation estimates had an absolute error of 0.31 degrees on average. The software solution can be integrated to be a part of the traffic sign localization pipeline of the company in question.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Business and Management
Tietotekniikan koulutusohjelma

Lauri Rapo

Kameran asennon arvioiminen kuvapiirteiden perusteella

Kandidaatintyö

2016

29 sivua, 10 kuvaa, 2 taulukkoa.

Tarkastajat: Prof., TkT Lasse Lensu

Hakusanat: konenäkö, liikennemerkki, paikantaminen, kuvapiirteiden tunnistus, kuvapiirteiden seuranta, asennontunnistus

Kameran suhteellisen asennon ja sijainnin arvioiminen on yksi olennaisista aiheista konenäön alalla. Erään suomalaisen teknologiayrityksen liikennemerkkien paikantamis- ja inventointiprosessin tarkkuutta voidaan parantaa hyödyntämällä edellä mainittua konseptia. Yrityksen paikantamisprossessi hyödyntää videoita, joita tuotetaan ajoneuvoon kiinnitetyllä kameralla. Liikennemerkin arvioidun sijainnin tarkkuus riippuu tämän kameran asennosta suhteessa ajoneuvoon. Tämä kandidaatintyö esittää konenäköön perustuvan ohjelmistoratkaisun, joka pystyy arvioimaan kameran asentoa suhteessa ajoneuvon liikesuuntaan hyödyntämällä kameran videodataa. Ratkaisussa hyödynnettiin kuvapiirteisiin perustuvia menetelmiä ja vapaan lähdekoodin ohjelmistoja. Kun ohjelmistoratkaisun testauksessa käytettiin simuloitua dataa, absoluuttinen virhe kameran asennolle oli keskimäärin 0,31 astetta. Ohjelmistoratkaisu voidaan integroida osaksi kyseessä olevan yrityksen liikennemerkkien inventoinnin prosessia.

CONTENTS

ABBREVIATIONS

FAST	Features from accelerated segment test.
FOE	Focus of expansion.
GPS	Global positioning system.
LIDAR	Light detection and ranging.
RANSAC	Random sample consensus.
SfM	Structure from motion.
SLAM	Simultaneous localization and mapping.
SVD	Singular value decomposition.

1 INTRODUCTION

1.1 Background

Vionice Ltd. is a Finnish technology company that specializes in utilizing computer vision for information production, asset management and service solutions. The inventory and localization of traffic signs is one of the principal services provided by the company. The outline of the localization process is as follows: Video data is produced by using a smartphone camera. The smartphone is attached to a vehicle's dashboard or windshield with a stand. Video and corresponding Global positioning system (GPS) data are uploaded to a server. The traffic signs are then extracted from the video using computer vision. The location of each detected traffic sign is determined by means of perspective projection and GPS data utilization. The traffic sign detection and localization processes in question are described in [1].

Since the heights of most traffic signs are known beforehand, the distance between a traffic sign and the camera can be triangulated. The GPS location of a traffic sign is calculated based on its estimated distance, its location in the frame and the GPS location of the camera. The accuracy of perspective projection results depends on the relative orientation between the camera and the vehicle (i.e., the direction of motion). If the exact orientation of the camera is unknown, the calculated locations of the traffic signs will be inaccurate. This work focuses on correcting these errors in the triangulation phase and was done in collaboration with Vionice.

Table 1 shows a computational example of how the divergence of the camera orientation affects the calculated location of a traffic sign. For example, if we denote the camera's horizontal angle divergence as α and the distance between a traffic sign and the camera as d, the corresponding triangulation error e can be calculated with the equation $e = d\tan(\alpha)$. Figure 1 presents the scenario from top view and Figure 2 from the side. Point C denotes the location of the camera and V denotes the heading of the vehicle in Figures 1 and 2. In Figure 1, α denotes the relative horizontal angle between the camera heading and vehicle heading. Respectively in Figure 2, β denotes the relative vertical angle between the camera heading and vehicle heading.

These aforementioned inaccuracies can be corrected by estimating the angle divergence by human vision and then manually correcting the heading for each video. Nonetheless, this method is not very practical if a large amount of video data needs

Table 1. Traffic sign distance and the corresponding errors (in meters) caused by camera angle divergence.

Distance (m)	10	20	30	40	50
Angle					
1°	0.17	0.35	0.52	0.70	0.87
2°	0.35	0.70	1.05	1.40	1.75
3°	0.52	1.05	1.57	2.11	2.62
4°	0.70	1.40	2.10	2.80	3.50
5°	0.87	1.75	2.62	3.50	4.37
6°	1.05	2.10	3.15	4.20	5.26
7°	1.23	2.46	3.68	4.91	6.14
8°	1.41	2.81	4.22	5.62	7.03
9°	1.58	3.17	4.75	6.34	7.92
10°	1.76	3.53	5.29	7.05	8.82

to be processed. This problem presents the need for an automated process that can resolve the orientation divergence automatically.

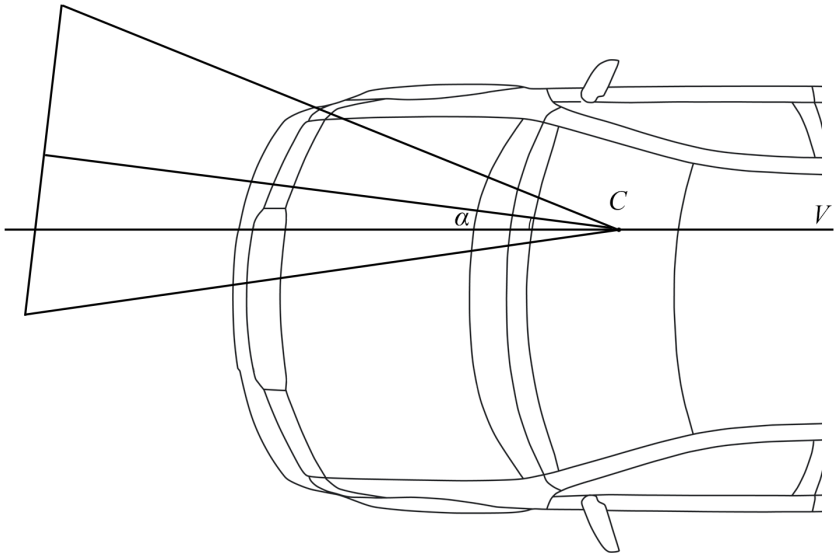

Figure 1. Top view of the scenario.

1.2 Approach

In essence, the camera orientation divergence estimation problem can be formulated as the estimation of the camera pose, which is the relative orientation and position of the camera. If the relative position and orientation of the camera are estimated between two consecutive frames, the camera orientation divergence can

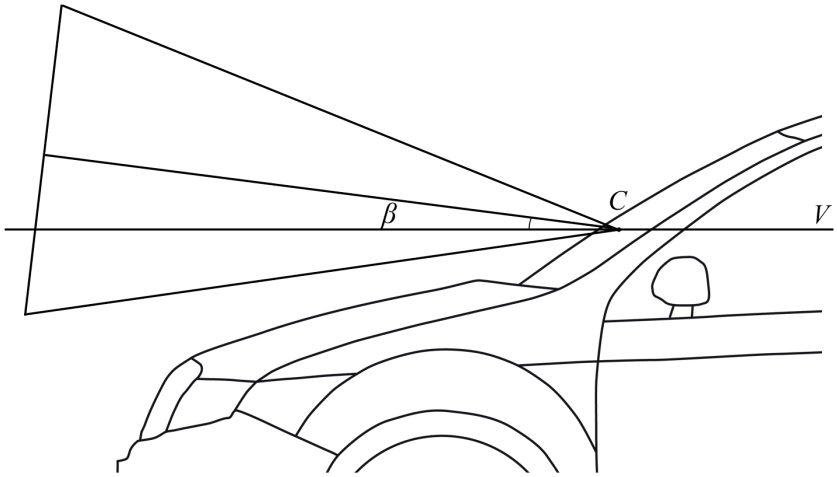

Figure 2. Side view of the scenario.

be approximated by calculating the orientation difference between the direction of motion and orientation. A more accurate estimate for the orientation difference can be achieved by calculating the average orientation difference of all frames. There are various approaches for estimating a camera's pose. Different methods for camera pose estimation include e.g. Light detection and ranging (LIDAR) systems [2], direct methods [3], depth sensing RGB-D cameras [4] or feature-based methods [5].

The task was approached by utilizing open-source software that is suitable for this problem. In this case the chosen basis was Mono-vo [6], a monocular feature-based visual odometry implementation created by Avi Singh. Mono-vo is open source and utilizes OpenCV 3.0 [7], a BSD-licensed computer vision software library. Mono-vo was chosen because it is relatively compact which makes it manageable and convenient from the viewpoint of software development. Also, Mono-vo is MIT-licensed, which means it can be used freely in commercial applications. The application served as a basis and was further developed to provide a practical solution specifically to the camera orientation divergence estimation problem.

Mono-vo suited the prerequisites of the use case of this thesis well, since it was supposed that any additional devices besides a smartphone (monocular camera) were assumed unavailable. Mono-vo utilizes Features from accelerated segment test (FAST) algorithm for feature detection, Lucas–Kanade method for optical flow estimation and feature tracking, and Nistér's five-point algorithm for estimating the essential matrix, which is directly related to the camera pose. For convenience, these aforementioned methods are also utilized in the final software solution for

the problem in question and are described in Section 2. All of these methods are implemented in the OpenCV library.

1.3 Objectives and restrictions

The objective of this bachelor's thesis was to describe a software solution which was developed to automatically determine the camera orientation divergence for any given video. The specific objectives for the method implementation are the following:

1. The software solution can process any given video that has been recorded by a dashboard/windshield attached smartphone.
2. The accuracy of the orientation estimation is at least 1 degree for both horizontal and vertical axes.

The specific assumptions and restrictions for the software implementation are the following:

1. For each video, it is assumed that the corresponding focal length and horizontal field of view are known.
2. It is assumed that the angle divergences remain constant during each individual video.

1.4 Structure of the thesis

Section 2 outlines the necessary theory and the chosen methods related to the problem of camera pose estimation. Section 3 discusses the practical implementation of the software solution to the camera orientation divergence estimation problem. Section 4 contains the experiments and results. The results are discussed and the thesis is concluded in Section 5.

2 CAMERA POSE ESTIMATION

The feature-based estimation of camera pose consists of three components:

1. The extraction of image features.
2. The motion tracking of features in consecutive frames.
3. The estimation of the essential matrix, from which the camera pose can be extracted.

The execution of these components requires that the intrinsic parameters of the camera are known. The intrinsic parameters are depicted in Section 2.1.2 and are part of the pinhole camera model, which is depicted in Section 2.1. The FAST algorithm for image feature extraction is described in Section 2.3 and the Lucas–Kanade method for feature tracking in Section 2.4. The essential matrix is presented in Section 2.2, its estimation using Nistér's five-point algorithm in Section 2.2.1 and the extraction of the camera pose in Section 2.2.2.

2.1 Pinhole camera model

The mathematical model which a few of the following concepts are built upon is the pinhole camera model. The pinhole camera model, which is also known as the perspective camera model, describes the mathematical relationship between 3D-points in the real world and 2D-points on the image plane. The pinhole camera model is a simplification of how an actual camera functions [8].

A typical camera is a hollow enclosure with a small hole (aperture), from which light can enter. A light-collecting lens is positioned in front of the aperture and a light-sensitive surface is positioned on the opposite side of the hollow. In an ideal pinhole camera, the aperture is a single point with no lenses. The pinhole camera model does not take lens distortions and other physical side-effects into account. This is why the model can be used only as an approximation for the relationship between the real world and the image [8].

In Figure 3, we can see an overview of the pinhole camera model. The focal plane F is at distance f (focal length) away from the image plane I. The pinhole C is located in the center of the focal plane F. Light that enters through the pinhole projects on the image plane and forms an inverted image. Objects in the 3D space are projected to a 2D space. Such linear transformation is referred as perspective

projection. The relationship between the 3D coordinates and the 2D coordinates can be written as

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where $x = U/S$ is the horizontal and $y = V/S$ the vertical coordinate in the image plane [8].

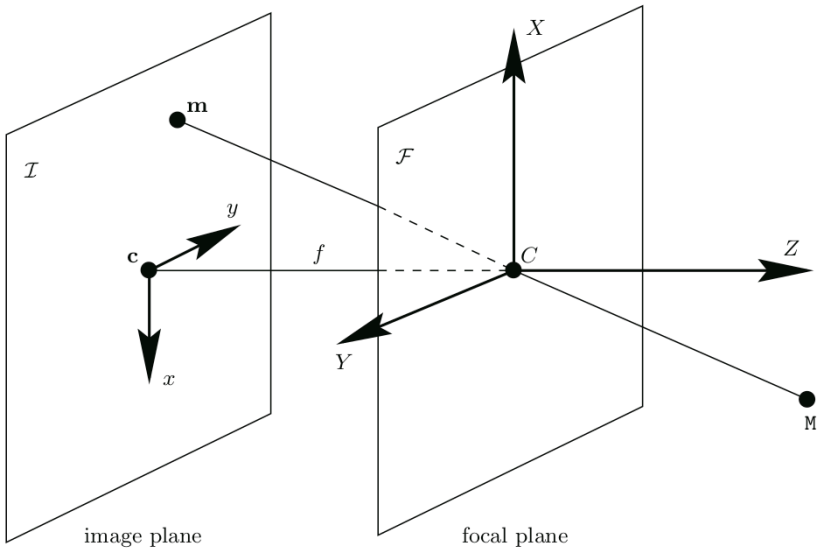

Figure 3. The pinhole camera model [8].

The pinhole camera model can be extended by including the extrinsic and intrinsic parameters of the camera. The extrinsic parameters are used to describe the transformation between the camera coordinates and world coordinates. The intrinsic parameters are used to describe the internal properties of the camera, such as the focal length. In the context of computer vision, the camera parameters need to be known in order to understand the relationship between the camera and the surrounding environment. However, some parameters such as lens distortion can be ignored if the quality of the camera is high. Lens distortions can be estimated and corrected if necessary [9].

2.1.1 Extrinsic parameters

The extrinsic parameters are used to transform from the camera coordinate system to a world coordinate system [9]. The transformation is achieved by rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^{3\times1}$. Thus a point in the world coordinate system, $\mathbf{M}$, can be expressed in camera coordinates, $\mathbf{m}$, as follows:

$$\mathbf{m} = \mathbf{R}\mathbf{M} + \mathbf{t} \tag{2}$$

The extrinsic parameters, or the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, are also referred as the camera pose [10]. The rotation matrix $\mathbf{R}$ can be transformed to Euler angles (yaw, pitch and roll) as follows [11]:

$$\begin{cases} \phi_1 = \text{atan2}(R_{32}, R_{33}) \\ \phi_2 = -\text{asin}(R_{31}) \\ \phi_3 = \text{atan2}(R_{21}, R_{11}) \end{cases} \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3}$$

2.1.2 Intrinsic parameters

The values that the intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ contains are directly tied to the physical properties of the camera [9]. The matrix takes the focal length, skewness and principal point into account. Generally, the principal point (the origin of the image plane) is not located at the center of the image plane [9]. Also the axes of the real image may not be orthogonal (skewness). The intrinsic matrix $\mathbf{K}$ is defined in Equation 4, where θ denotes the angle between the horizontal and vertical image axes. f_u and f_v denote the focal length in pixel units. The horizontal and vertical coordinates of the principal point are represented by u_0 and v_0 respectively.

$$\mathbf{K} = \begin{bmatrix} f_u & f_u\text{cot}(\theta) & u_0 \\ 0 & f_v/\text{sin}(\theta) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

If we assume that the image axes are orthogonal, the angle θ is $\pi/2$. Thus the matrix takes the form:

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

When matrices $\mathbf{K}$ and $\mathbf{R}$ and vector $\mathbf{t}$ are known, the camera parameters are known. By incorporating both intrinsic and extrinsic parameters, the following projection matrix can be written:

$$\mathbf{x} = \mathbf{K} \left[\mathbf{R} \;\middle|\; \mathbf{t} \right] \mathbf{X} \tag{6}$$

where $\mathbf{x}$ is a homogeneous representation of the corresponding 2D point on the image and $\mathbf{X}$ is a homogeneous representation of the 3D point which is mapped to $\mathbf{x}$ by the camera [9].

2.2 Epipolar geometry

Epipolar geometry describes the geometric relationship between two camera systems [12]. In the context of a monocular camera system, we can treat two camera systems as two consecutive frames from a single camera. The relation is encompassed in the essential matrix $\mathbf{E} \in \mathbb{R}^{3\times3}$ in the case of a calibrated camera or in the fundamental matrix $\mathbf{F} \in \mathbb{R}^{3\times3}$ in the case of an uncalibrated camera. The motivation for considering epipolar geometry is the fact that if the essential matrix or the fundamental matrix is known, the camera pose can be recovered from these aforementioned matrices [13]. Essentially, the search for the camera pose is the search for either of these matrices. If a 3D point $\mathbf{X}$ is projected as $\mathbf{x}$ in the first view, and as $\mathbf{x}'$ in the second, the points are related to each other as follows [13]:

$$\mathbf{x'}^{\top} \mathbf{F} \mathbf{x} = 0 \tag{7}$$

In Figure 4a, we can see that the camera centers, the image points $\mathbf{x}$ and $\mathbf{x}'$, and the world point $\mathbf{X}$ reside on the same plane π. The plane π is referred as an epipolar plane [13]. If the image points $\mathbf{x}$ and $\mathbf{x}'$ are re-projected back to a world point, the projection lines intersect at $\mathbf{X}$. Assuming that only the image point $\mathbf{x}$ is known, we can then consider how the corresponding point $\mathbf{x}'$ is constrained. The plane π

is determined by the line connecting the camera centers of the two views and the projection line connecting $\mathbf{x}$ and $\mathbf{X}$. It is known that $\mathbf{x}'$ lies on the plane π. Thus $\mathbf{x}'$ lies on the intersection of line $\mathbf{l}'$ and plane π on the image plane of the second view. Line $\mathbf{l}'$ is referred as the epipolar line corresponding to $\mathbf{x}$ [13].

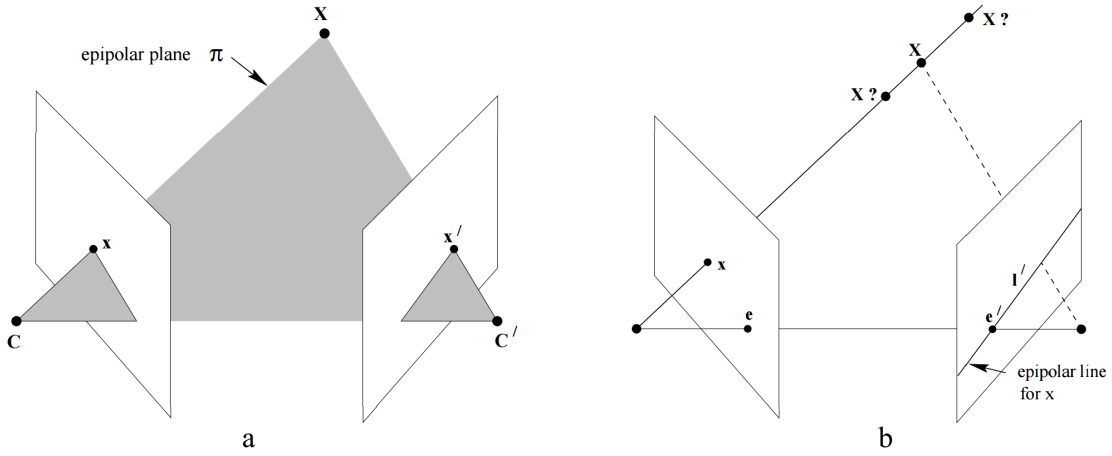

Figure 4. Point correspondence geometry. (a) The point **C** denotes the camera center of the first view and **C′** of the second. The camera centers, the image points **x** and **x′**, and the world point **X** lie on the epipolar plane π. (b) The image point **x** projected back to a 3D space creating a projection ray. The 3D point **X** lies on this projection ray and **x′** lies on the epipolar line **l′** [13].

In a sense, the fundamental matrix is a generalization of the essential matrix. The essential matrix has only five degrees of freedom, whereas the fundamental matrix has seven [13]. The essential matrix also has additional properties and constraints. The essential matrix can be defined as

$$\mathbf{E} = \mathbf{K'}^{\top}\mathbf{F}\mathbf{K} \tag{8}$$

where $\mathbf{K}'$ and $\mathbf{K}$ are the intrinsic calibration matrices of the two views [13]. If the intrinsic calibration matrices are known, and $\mathbf{x}$ and $\mathbf{x}'$ (homogeneous form) are premultiplied by the intrinsic calibration matrices, the epipolar constraint in Equation 7 can be expressed as [14]

$$\mathbf{x'}^{\top}\mathbf{E}\mathbf{x} = 0. \tag{9}$$

2.2.1 Nistér's five-point algorithm

Nistér's five-point algorithm [14] is a method for estimating the essential matrix or the fundamental matrix which are related to a pair of camera views. The algorithm estimates the essential matrix or the fundamental matrix from a set of five corresponding image feature points. In order to estimate these matrices, the algorithm computes the coefficients of a tenth degree closed-form polynomial and then finds its roots.

All of the five point correspondences follow the epipolar constraint described in Equation 9. The constraint can also be expressed as $\tilde{\mathbf{q}}^\top \tilde{\mathbf{E}} = 0$ where

$$\tilde{\mathbf{q}} \equiv \begin{bmatrix} q_1q'_1 & q_2q'_1 & q_3q'_1 & q_1q'_2 & q_2q'_2 & q_3q'_2 & q_1q'_3 & q_2q'_3 & q_3q'_3 \end{bmatrix}^\top \tag{10}$$

$$\tilde{\mathbf{E}} \equiv \begin{bmatrix} E_{11} & E_{12} & E_{13} & E_{21} & E_{22} & E_{23} & E_{31} & E_{32} & E_{33} \end{bmatrix}^\top \tag{11}$$

and q and q' denote the homogeneous presentations of the corresponding image points. The subscripts in Equation 10 denote the vector elements. A 5×9 matrix is formed when the vector $\tilde{\mathbf{q}}^\top$ is stacked for all five point correspondences. The right nullspace of this matrix is formed by vectors $\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, \tilde{\mathbf{Z}}, \tilde{\mathbf{W}}$. These vectors are computed by QR-factorization. The vectors correspond to four 3×3 matrices $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W}$. The essential matrix is of the form

$$\mathbf{E} = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z} + \mathbf{W} \tag{12}$$

where x, y and z are scalars to be solved [14].

Using the following constraints

$$\mathbf{E}\mathbf{E}^\top\mathbf{E} - \frac{1}{2}\text{trace}(\mathbf{E}\mathbf{E}^\top)\mathbf{E} = 0 \qquad \det(\mathbf{E}) = 0 \tag{13}$$

and by applying Gauss-Jordan elimination with partial pivoting, equation system A can be formed as seen in Figure 5a [14].

In equation system A, a dot denotes a scalar value and $[N]$ denotes a polynomial of

A	x^3	y^3	x^2y	xy^2	x^2z	x^2	y^2z	y^2	xyz	xy	x	y	1
$\langle a \rangle$	1	.	.	.	.	.	.	.	.	.	[2]	[2]	[3]
$\langle b \rangle$		1	.	.	.	.	.	.	.	.	[2]	[2]	[3]
$\langle c \rangle$			1	.	.	.	.	.	.	.	[2]	[2]	[3]
$\langle d \rangle$				1	.	.	.	.	.	.	[2]	[2]	[3]
$\langle e \rangle$					1						[2]	[2]	[3]
$\langle f \rangle$						1					[2]	[2]	[3]
$\langle g \rangle$							1				[2]	[2]	[3]
$\langle h \rangle$								1			[2]	[2]	[3]
$\langle i \rangle$									1		[2]	[2]	[3]
$\langle j \rangle$										1	[2]	[2]	[3]

a)

B	x	y	1
$\langle k \rangle$	[3]	[3]	[4]
$\langle l \rangle$	[3]	[3]	[4]
$\langle m \rangle$	[3]	[3]	[4]

b)

Figure 5. Equation systems A (a) and B (b) [14].

degree N in the variable z. Afterwards, the following equations are defined:

$$\begin{cases} \langle k \rangle \equiv \langle e \rangle - z\langle f \rangle \\ \langle l \rangle \equiv \langle g \rangle - z\langle h \rangle \\ \langle m \rangle \equiv \langle i \rangle - z\langle j \rangle. \end{cases} \tag{14}$$

These equations are accommodated into a 3×3 matrix B, which contains z polynomials. Matrix B is illustrated in Figure 5b. The vector $[x\ y\ 1]^\top$ is a nullvector to B, thus the determinant of B vanishes. The determinant is the tenth degree polynomial

$$\langle n \rangle \equiv \det(B). \tag{15}$$

The next step is the computation of the real roots of $\langle n \rangle$. One way to accomplish this is using Sturm-sequences [15] to bracket the roots, followed by a root-polishing scheme [14]. Using the equation system B, the variables x and y can be found for each root z. The essential matrix can then be obtained from Equation 12 [14].

In practice, when estimating the essential matrix, more than five point correspondences are used [14]. To produce a more accurate estimate, multiple samples of five point correspondences are used. An estimate for the essential matrix is calculated for each sample. To eliminate outliers and to obtain a more robust estimate, the results are processed by the Random sample consensus (RANSAC) [16] [17] method.

2.2.2 Pose extraction from the essential matrix

The camera matrices can be recovered, when the essential matrix is known [13]. In the case of an essential matrix, the camera matrices can be recovered up to scale. However, there are four possible solutions. We can assume the camera matrix of the first view as

$$\mathbf{P} = \left[\mathbf{I} \,\middle|\, \mathbf{0}\right]. \tag{16}$$

If we factorize the essential matrix by Singular value decomposition (SVD) $\mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, the camera matrix of the second view can be determined as follows [13]:

$$\mathbf{P}' = \left[\mathbf{U}\mathbf{W}\mathbf{V}^\top \,\middle|\, \pm\mathbf{u_3}\right] \text{ or } \left[\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \,\middle|\, \pm\mathbf{u_3}\right] \tag{17}$$

where $\mathbf{u_3}$ is the third column vector from the matrix $\mathbf{U}$ and where

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{18}$$

Thus the rotation matrix $\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^\top$ or $\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$, and translation vector $\mathbf{t} = \pm\mathbf{u_3}$. To determine which of the camera matrices is the correct one can be deduced by testing if a single point is in front of both of the cameras.

2.3 FAST feature extraction

Features from accelerated segment test (FAST) [18] is a corner detection algorithm, which can be used to extract features from image frames. FAST is very computationally efficient compared to most other feature detection methods and was designed real-time applications in mind e.g. Simultaneous localization and mapping (SLAM).

Suppose we have a corner candidate point p. Next, the candidate corner is surrounded by a Bresenham circle of 16 pixels. The point p is classified as a corner if a

set of n contiguous pixels exists in the circle which all have a brighter intensity than the candidate pixel I_p plus a threshold t, or a darker intensity than $I_p - t$ [18]. All of this is illustrated in Figure 6.

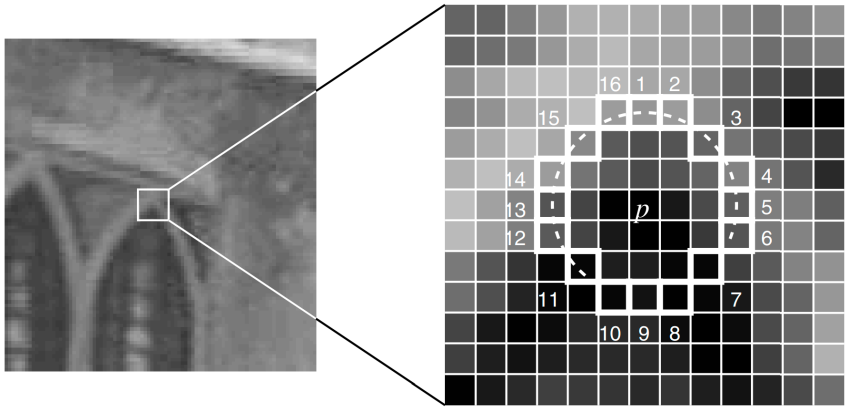

Figure 6. The pixel at p is the center of a candidate corner. The highlighted pixels that lie on the circle are used in the corner classification [18].

Multiple detected corners at adjacent locations can be pruned with indirect non-maximal suppression. A score function V is computed for all the detected corners. There are multiple ways to define the score function V. One way is to define V as the sum of absolute intensity difference between p and the surrounding contiguous arc of pixels [18]. When we consider two features at adjacent locations, their calculated V values are assessed. The feature with the lower V value is discarded, which means that only the most distinctive corners are selected.

2.4 Lucas–Kanade method

The Lucas–Kanade method is a technique of optical flow estimation and can be used for tracking feature points in consecutive image frames [19]. Optical flow is a vector field that describes the apparent motion of objects in the image plane. Suppose we have two consecutive image frames. By estimating the optical flow between these two frames, one can approximate how each pixel has moved. A matching criterion like the optical flow constraint is needed to estimate the optical flow field:

$$I(x + u(x, y), y + v(x, y), t + 1) - I(x, y, t) = 0. \tag{19}$$

In Equation 19, I denotes the image sequence and $w = (u, v)^\top$ the optical flow field. This optical flow constraint is also known as the brightness constancy constraint [20]. The optical flow constraint is often linearized to the form

$$I_x u + I_y v + I_t = 0 \tag{20}$$

where the subscripts denote partial derivatives [19]. Using the optical flow constraint, an equation can be constructed for each pixel. There are two unknown variables for each pixel, thus the problem is underconstrained. One way two solve this problem is to use the Lucas–Kanade method: Instead of considering each pixel as a single entity, a neighborhood of a fixed size is determined for each pixel. If the chosen neighborhood size is large enough, a sufficient amount of information is collected to determine the flow [19]. The Lucas–Kanade method was originally proposed in [21].

The particular implementation of the Lucas–Kanade method, that is included in the OpenCV library [22] and is utilized in the software implementation, includes a pyramidal approach for motion tracking [23]. The pyramidal approach allows the tracking of large pixel motions. The image is in a sense divided in to L_m layers. The first and topmost layer L_0 encompasses the whole image in full resolution. The layer L_1 below it includes a portion that is half the width and height of the full image. Each time we progress to a lower layer, the portion decreases by dividing the width and height of the previous layer by two. First, the optical flow and affine transformation are computed at the lowest level L_m of the pyramid. The computation results are then propagated to the layer above it $L_m - 1$. The computations are executed again and then propagated to the upper layer until the top layer is reached.

3 PRACTICAL IMPLEMENTATION

The first step in the software implementation is to load the first frame of the video. The feature points of the frame are extracted with the FAST algorithm. The subsequent frame is then loaded. The features which were detected in the first frame are tracked to the second frame using the Lucas–Kanade method. The features for which the feature tracking fails or that which move outside the frame are removed. Next, the essential matrix is estimated by using Nistér's five-point algorithm which utilizes the tracked features.

After the essential matrix is estimated, the rotation matrix and the translation vector are extracted. The rotation matrix is decomposed into Euler angles. By using the Euler angles, a vector representation is created for the camera heading. The angles between the camera heading vector and the translation vector are calculated. Then the calculated values are tested if the values are within the camera's horizontal and vertical fields of view. If the test is passed, these angle values are stored for later use.

Next, the total number of remaining features is checked. If the number of features is below a certain threshold, a redetection of features is executed. Whether the aforementioned redetection is executed or not, the next frame is loaded and the cycle starts over. When all of the used frames are processed, the final values for the camera orientation divergence are determined by calculating the mean of all of the previously computed values. A general outline of the software implementation is illustrated in Figure 7.

In order to reduce the total computation time of the software implementation, a frame skipping feature was implemented: Every nth cycle, the next loaded frame jumps ahead s frames. The variable s can be defined for example, as the number of total frames used in the processing divided by 10. When using the frame skipping feature, the software implementation evaluates the camera orientation divergence from smaller sections. By utilizing smaller sections which span the entire video, we can lessen the effects of the occasional non-ideal conditions in the frames. A section of a video where the vehicle is not moving is an example of a non-ideal scenario.

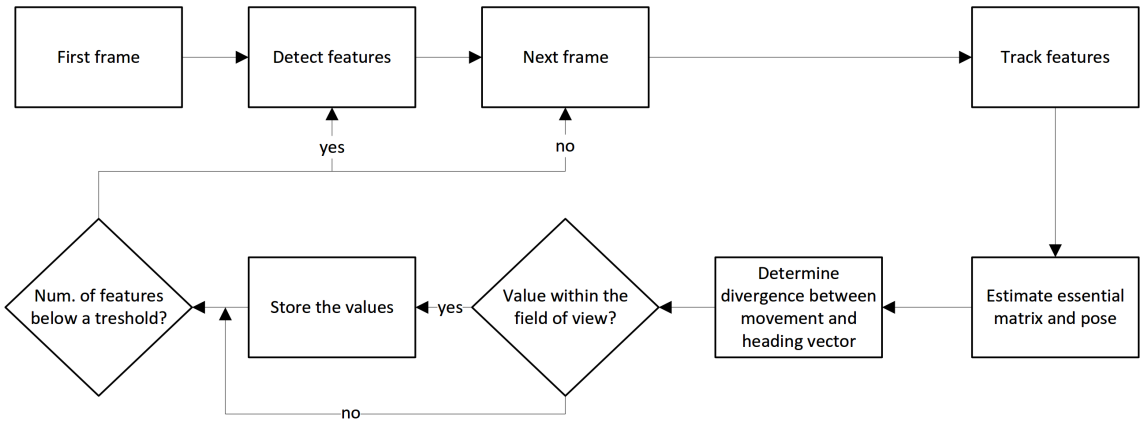

Figure 7. The outline of the software implementation.

4 EXPERIMENTS AND RESULTS

4.1 Datasets and evaluations

In order to evaluate the performance of the implementation, one should capture video data for which the angle divergences are known accurately beforehand. Ideally, the best way to go about the data collection would be capturing real-world video with a smartphone camera attached to a vehicle. However, in practice, it is very difficult to measure the camera orientation in relation to the vehicle beforehand. This challenge led to an alternative solution: The test data were created by rendering videos from a virtual 3D environment. In this 3D environment, the camera pose could be manipulated easily and accurately beforehand.

The test data collection phase was carried out as follows: The test data includes videos where the angle divergences include all possible combinations of values [0, 5, 10, 15] (degrees) for both horizontal and vertical axes. The number of different test videos is thus 16. After the videos that contain these 16 scenarios are ran through the software implementation, the given values are compared to the exactly known values.

The test data were generated with Blender, an open-source 3D graphics software product. The design chosen for the generated test data was a continuous tube. The generated route contains a variety of gradual and sharp turns in an attempt to provide some real-world characteristics to the test data. A checkerboard-like texture was applied to the tube. This kind of texture provides plenty of features for the FAST algorithm to detect. The generated data does not correlate with real videos very well, but it allows performance evaluation in a nearly ideal test scenario. The frame skipping feature described in Section 3 was not used when testing the generated test data, since it was appropriate that all of the different conditions (i.e., the various turns) would be included in the calculations. Figure 8 shows the overview of the modeled route, and Figure 9 shows a rendered frame from the interior of the route.

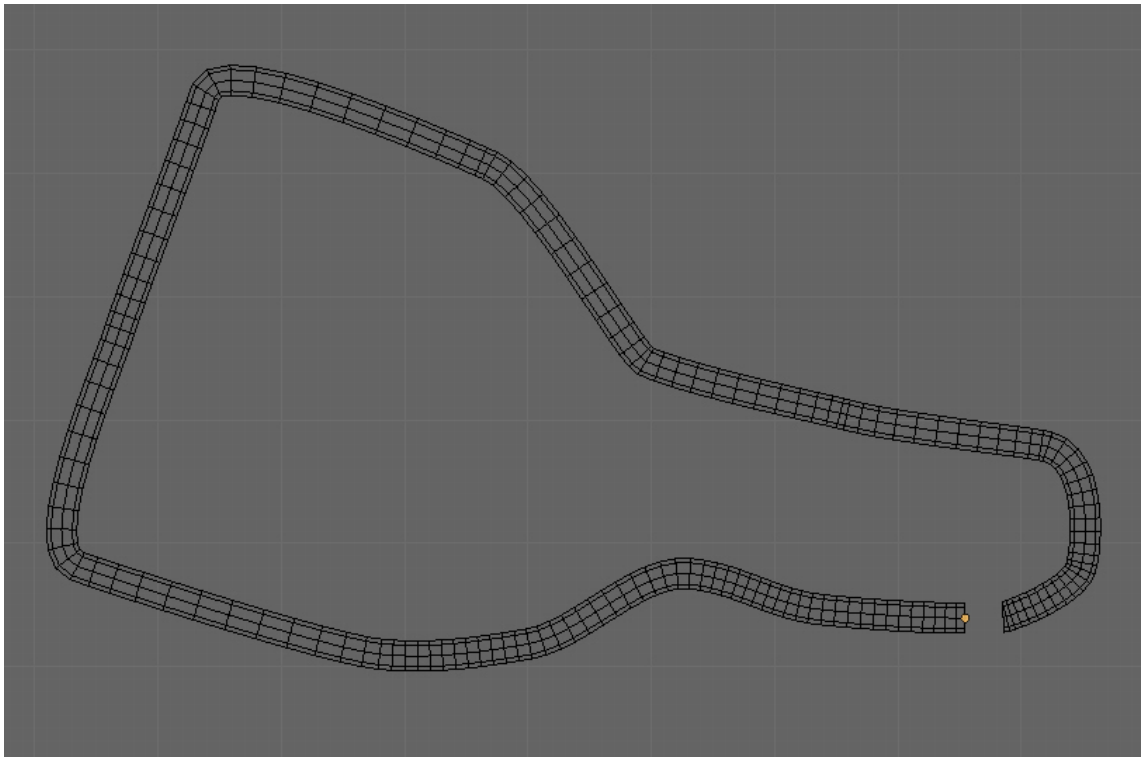

Figure 8. The overview of the modeled route. The orange dot presents the starting point of the route.

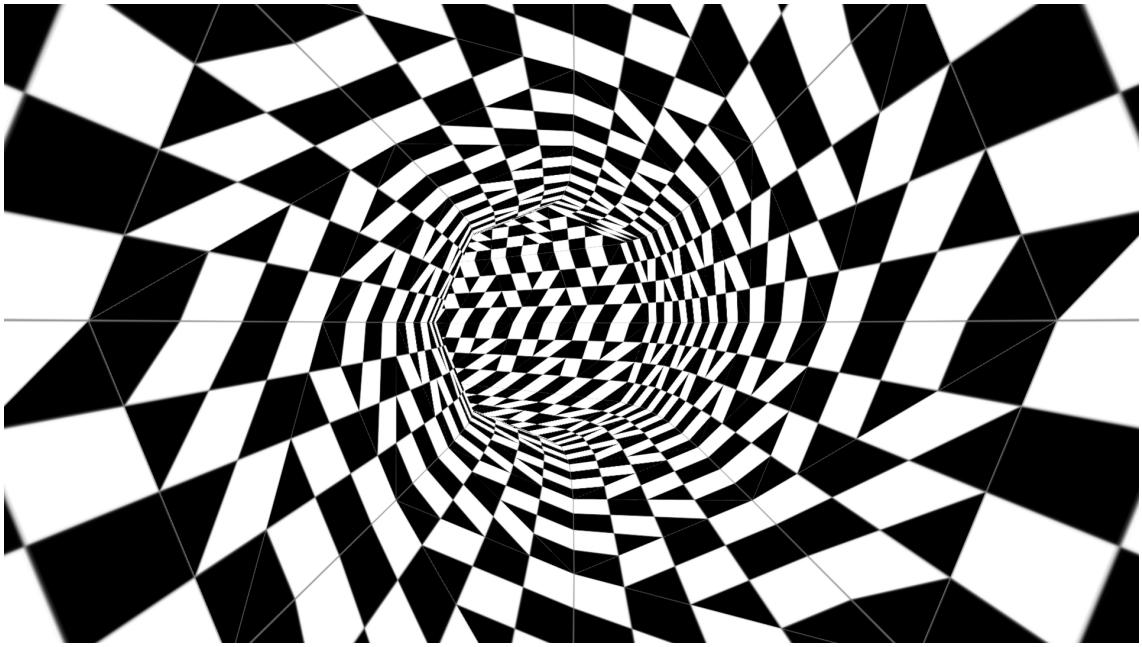

Figure 9. A rendered frame from the interior of the route

4.2 Experiment results

Table 2 shows all of the test scenarios, and the absolute and relative errors for each scenario. α denotes the horizontal angle divergence and β the vertical angle divergence. When it comes to real-world videos, in Figure 10 we can see that the estimated camera orientation divergence is quite close to the actual direction of motion or the Focus of expansion (FOE).

Table 2. Test results for all test scenarios.
(* Relative error cannot be calculated when the real value is zero.)

α	β	Abs. α error	Abs. β error	Rel. α error	Rel. β error
0°	0°	0.4524°	0.1627°	*	*
0°	5°	0.4861°	0.0665°	*	1.3300 %
0°	10°	0.3071°	0.3834°	*	3.8340 %
0°	15°	0.2031°	0.5782°	*	3.8547 %
5°	0°	0.2311°	0.1317°	4.6222 %	*
5°	5°	0.1149°	0.0823°	2.2972 %	1.6464 %
5°	10°	0.0210°	0.3273°	0.4192 %	3.2730 %
5°	15°	0.2724°	0.4797°	5.4476 %	3.1980 %
10°	0°	0.0079°	0.0566°	0.0790 %	*
10°	5°	0.0061°	0.1958°	0.0613 %	3.9156 %
10°	10°	0.2699°	0.2363°	2.6990 %	2.3630 %
10°	15°	0.5788°	0.5612°	5.7880 %	3.7413 %
15°	0°	0.1344°	0.0682°	0.8960 %	*
15°	5°	0.2646°	0.2173°	1.7640 %	4.3454 %
15°	10°	1.2534°	0.3934°	8.3560 %	3.9340 %
15°	15°	0.8164°	0.5520°	5.4427 %	3.6800 %
Average		0.3387°	0.2808°	3.1560 %	3.2596 %

Figure 10. The software implementation processing a real-world video. The black lines indicate the center of the frame. The yellow lines indicate the estimated orientation divergence which is the mean accumulated value. The red lines indicate the tracked points in the current frame.

5 CONCLUSIONS

The objective of this thesis was to present a software solution which can determine a camera's orientation relative to the direction of motion. When the implementation was experimented on using a generated test data set, the accuracy of the estimation results were more than adequate: The average absolute error for the orientation divergence was 0.3387 degrees in the horizontal axis and 0.2808 degrees in the vertical axis on average. If we average these results, the overall mean absolute error was 0.3098 degrees.

Because of practical reasons, accurate testing with real-world videos was not possible. Still, indicative performance could be measured by assessing the final locations of the traffic signs on a map and through repeatability. Nonetheless, the overall performance on real-world videos was visually evaluated. When assessing these observations, the results seemed quite accurate.

The orientation estimation accuracy could possibly have been further improved by fine-tuning the parameters of the various algorithms. However, it should be noted that the used test data was artificial. The tuning of parameters to better suit the case of generated data does not necessarily ensure better performance in the case of real-world videos. The software solution meets the given requirements and hereby can be integrated to be a part of Vionice's traffic sign inventory pipeline.

5.1 Future work

The next potential step for improving the traffic sign inventory pipeline could be abandoning triangulation based on the known heights of traffic signs. By utilizing the same or similar methods displayed in this thesis, a localization method based on the 3D reconstruction of a video sequence could be built. The system would be similar to SLAM [24] [25] and Structure from motion (SfM) [26] systems. By creating a 3D reconstruction of a video sequence and fusing it with GPS data, the locations of traffic signs could then possibly be estimated without knowing the heights of traffic signs beforehand.

References

[1] P. Hienonen, "Automatic traffic sign inventory- and condition analysis," Master's thesis, Lappeenranta University of Technology, Finland, 2014. [Online]. Available: http://urn.fi/URN:NBN:fi-fe2014092244826.

[2] C. Brenner, "Vehicle localization using landmarks obtained by a lidar mobile mapping system," *Int. Arch. Photogramm. Remote Sens*, vol. 38, pp. 139–144, 2010.

[3] D. Burschka and E. Mair, "Direct pose estimation with a monocular camera," in *Robot Vision*, Springer Berlin Heidelberg, 2008, pp. 440–453.

[4] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, IEEE, 2011, pp. 719–722.

[5] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme.," in *Intelligent Vehicles Symposium*, 2010, pp. 486–492.

[6] A. Singh, *Mono-vo*, 2015. [Online]. Available: https://github.com/avisingh599/mono-vo (visited on 12/16/2015).

[7] Itseez. (2016). Opencv, [Online]. Available: http://opencv.org/ (visited on 12/16/2015).

[8] P. Sturm, "Pinhole camera model," in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed., Springer US, 2014, pp. 610–613. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-31439-6_472.

[9] Z. Zhang, "Camera parameters (intrinsic, extrinsic)," in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed., Boston, MA: Springer US, 2014, pp. 81–85. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-31439-6_152.

[10] T. Nöll, A. Pagani, and D. Stricker, "Markerless camera pose estimation-an overview," in *OASIcs-OpenAccess Series in Informatics*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 19, 2011.

[11] G. G. Slabaugh, "Computing euler angles from a rotation matrix," *Retrieved on August*, vol. 6, no. 2000, pp. 39–63, 1999.

[12] Z. Zhang, "Essential matrix," in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed., Springer US, 2014, pp. 258–259. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-31439-6_129.

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

[14] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[15] W. Gellert, *The VNR concise encyclopedia of mathematics*. Springer Science & Business Media, 2012.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[17] D. Nistér, "Preemptive RANSAC for live structure and motion estimation," *Machine Vision and Applications*, vol. 16, no. 5, pp. 321–329, 2005.

[18] E. Rosten and T. Drummond, "Computer vision – eccv 2006: 9th european conference on computer vision, Graz, Austria, May 7-13, 2006. proceedings, part I," in, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ch. Machine Learning for High-Speed Corner Detection, pp. 430–443. [Online]. Available: `http://dx.doi.org/10.1007/11744023_34`.

[19] T. Brox, "Optical flow," in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed. Boston, MA: Springer US, 2014, pp. 564–564. [Online]. Available: `http://dx.doi.org/10.1007/978-0-387-31439-6_100214`.

[20] M. Irani and P. Anandan, "Vision algorithms: Theory and practice: International workshop on vision algorithms Corfu, Greece, September 21–22, 1999 proceedings," in, B. Triggs, A. Zisserman, and R. Szeliski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, ch. About Direct Methods, pp. 267–277. [Online]. Available: `http://dx.doi.org/10.1007/3-540-44480-7_18`.

[21] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (IJCAI)," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, 1981, pp. 674–679.

[22] OpenCV. (2014). Motion analysis and object tracking - calcopticalflowpyrlk, [Online]. Available: `http://docs.opencv.org/3.0-beta/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowpyrlk` (visited on 03/16/2016).

[23] J.-Y. Bouguet, "Pyramidal implementation of the affine Lucas–Kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[24] R. Mur-Artal, J. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[25] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 871–889. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30301-5_38.

[26] H. Ackermann, "Factorization," in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed. Boston, MA: Springer US, 2014, pp. 288–291. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-31439-6_689.