TÉCNICO LISBOA

Artificial Eye Orientation Estimation with Camera-Based Measurements of Natural Visual Features

Adding a camera to obtain an unbiased real-time estimate for the three-dimensional orientation of a humanoid eye

Mariana Ribeiro dos Reis do Vale Martins

Report for

Introduction to the Research in Electrical and Computer Engineering

Advisor(s)/Supervisor(s): Prof./Dr. Alexandre Bernardino
Prof./Dr. José Santos-Víctor
Prof./Dr. John van Opstal

January 2019

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Even after years of research, the brain remains to this day a mysterious information-processing biological system. For example, from many of its puzzling abilities, it is still not completely understood how the brain determines which muscles to control in order to fulfill a particular movement task. This task could be something as simple as grasping an object, or even before that, looking at (and identifying) that object.

The required number of degrees of freedom to perform a particular movement is typically much smaller than the ones made available by the muscular apparatus, thus yielding a redundant control problem with infinitely many possibilities. This aspect of motor control has become known as the "Degrees of Freedom Problem" (DOF)[1], and was first articulated by Nikolai Bernstein in its current form. For example, the eye has six extra-ocular muscles (6 DOF), but to point the eye in any given direction, only two coordinates are needed (the azimuth and the elevation angles). Bernstein theorized that the brain gradually tries to find the optimal control solution for a certain task, to constrain the DOF to the required motor space, which would result in consistent performance. Indeed, when different subjects perform the same task, their muscular activations are remarkably similar. As it will be seen below, the eye orientation thus appears to be constrained by the so-called Donders' law.

Moreover, navigating through an environment in which multiple stimuli compete for attention, and being equipped with multiple sensory and motor systems to do so, severely complicates the challenges for the brain to rapidly select the optimal plan for a response to achieve a particular goal [1]. This leads to several questions: How does the normal brain decide which signals are "goals", and which are "distractors" in those complex environments? How is the plan to reach the goal truly defined? And how does this differ for sensory-impaired brains?

When focusing on audiovisual stimuli, it is important to apply the scientifically acquired concepts from neuroscience and psychophysics to an approximate model of the human sensory-motor system (eyes, head, and ears), which is the main topic of this research project. The idea is to create an autonomous humanoid eye-head robot with foveal vision, realistic auditory inputs, three-dimensional nested eye and head motor systems, and rapid sensory-motor feedback controls and learning algorithms. So far, a working mechanical model of a biologically inspired eye with six muscles has been built in a previous project [2], and is shown in Figure 1.1.

This model was constructed with Donders' and Listing's Laws in mind. Donders' law states that the 3D orientation of the eye, when looking in a specific direction, is always the same. Listing propounded

[1] N. Bernstein, "The Coordination and Regulation of Movements. Oxford : Pergamon Press," 1967.

that the law could be further constrained by the discovery that the rotation axes corresponding to all possible eye orientations all lie in a common plane, called Listing's plane. The component of the rotation axis normal to this plane quantifies the eye's torsional orientation component, and results to be close to zero in these conditions. The normal to Listing's Plane is directed into the so-called *primary direction*, and points about 15 to 30 degrees upwards relative to the straight-ahead viewing direction (head fixed). It is still not well understood which aspects of the ocular motor system determine the primary orientation (and hence, Listing's law).

Whether the rotation axis of eye orientation is programmed by the brain, fully determined by the mechanical properties of the eye muscles, or by both factors, is still not known for sure, and is highly debated in the literature. Yet, implementing this system in a humanoid robot might help at understanding this problem better. Note that for active eye-head movements, for vergence eye movements (near viewing), for vestibular eye movements (head rotations), and for eye-movements under tilted head orientations, Listing's law no longer holds true, which provides a strong counter argument against the eye muscle (pulley) hypothesis. [3]

The current prototype uses an Inertial Measurement Unit (IMU) sensor to estimate the eye's pose. Although IMU units are good on acceleration and velocity measurements, they tend to have a significant position drift when determining the orientation (see Figure 1.2), which is of course a considerable problem when the eye's orientation needs to be established with better than 0.5 deg resolution. Therefore, this project's focus is to study the addition of a camera to the system, in order to determine the orientation of the eye in a more reliable manner for this model. This will hopefully contribute to a better understanding of the eye's control system, and consequently help at restoring impaired vision, which was, after all, Franciscus Donders' ultimate desire.

Figure 1.1: The current mechanical eye model. It is composed of an IMU sensor, seen on the left, and connected to a supportive green eye mounted on a global joint, in turn connected to six elastics, representing the eye muscles, and finally controlled by three motors that pull at those elastics (paired by the aluminum strips). The three motors are controlled by the computer, which is seen lying on the table.

1.2 Problem definition

As the eye model will eventually rely on accurate camera output, and to foster solid neuroscientifically inspired study with this model, it's indispensable to have the best possible estimates of the camera's

translation and orientation (and relative image transformations). As well as that, the computational speed versus accuracy trade off will also have to be dealt with. The objective of this study is to approximate the current model to a biological eye as well as possible. Thus, determining the instantaneous orientation of the eye in 3D when the gaze direction changes suddenly, which happens, for example, during rapid saccadic eye movements, should be calculated at the appropriate speed.

The aim of this work is to find the accurate camera poses within the desired time constraints. The accuracy and computational time mainly depend on the complexity of feature detection, on feature matching, and on the algorithm responsible for calculating the pose, making each of these factors an important sub-topic for this study.

This project intends to go through a detailed analysis of the advantages and disadvantages of using the camera over the IMU, and what may be gained by the combination of both.

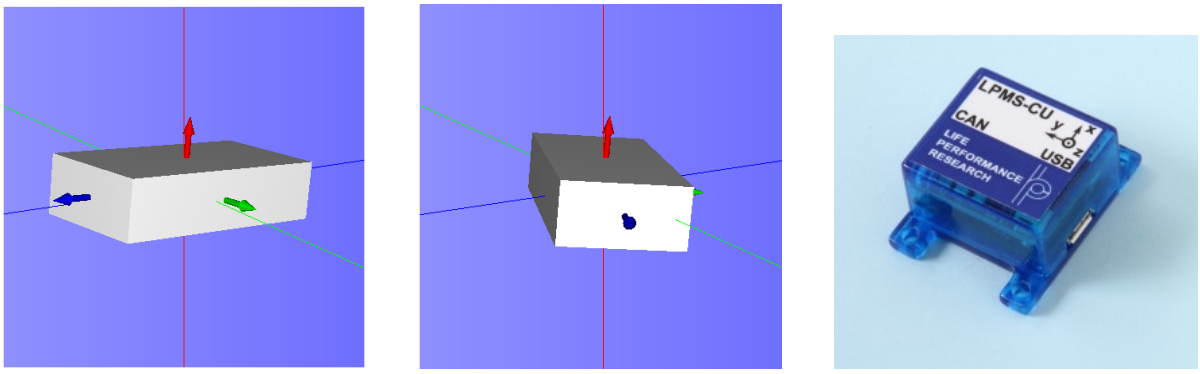

Figure 1.2: IMU sensor's drift. The two images on the left represent the IMU position on 3D space, between them it can be observed that, in 80 seconds, the IMU drifts about 60 degrees around the torsional axis. The coordinate system of the IMU sensor is defined on the image on the right, where the axis in red is the torsional component, z, green is y, and blue is x.

1.3 Report Outline

The next chapters of this report are structured as follows. Chapter 2 gives insight on the initial but essential concepts necessary to launch this project, along with an overview of the state of the art on some of the matters described. Chapter 3 explains the methods that will be experimented and later on compared, in order to accomplish the project's goal. Chapter 4 describes the preliminary work on some of the project's subjects that have taken place during the last months. And finally, chapter 5 defines the master thesis development plan for the months ahead, with the respective Gantt chart.

Chapter 2

Background and State of the art

2.1 Camera Model

The first step in this project is to understand and apply the pinhole camera model. Digital cameras are equipped with a sensor (mostly charge-coupled devices (CCD) and CMOS') that transforms light into discrete electrical signals. When taking a picture, some of the light reflecting on the object is directed towards the camera pinhole (tiny aperture on the camera), forming a 180º inverted image of the object on the sensor. All the light rays entering the camera converge into the pinhole and diverge on the other side (with a unique one-to-one correspondence). Therefore the pinhole is also named the **center of projection**, principal point, or focal point.

In cameras the aperture size and light-exposure time can be controlled. A smaller aperture produces a sharper image but needs a longer exposure time to collect enough light, whereas a larger pinhole allows more light to be captured, but at the expense of producing blurred images.

The image of the object is projected onto the so-called **image plane**. When moving this plane closer to the pinhole (zoom out), the projected object gets smaller, and vice-versa. The distance between the plane and the aperture is the **focal length**, or focal distance. The view-angle of a camera depends on the focal distance, and on the sensor size. The image resolution depends solely on the number of pixels that fit on the sensor.

A useful coordinate system, called the **camera reference system**, is centered at the pinhole, and has its z-axis perpendicular to the image plane. The line defined between the center of projection and the center of the image plane is called the **optical axis**.

Figure 2.1 summarizes the concepts explained above, and will help to understand the upcoming details. As may be observed, a correspondence between the real world (3D) points $M = [X\ Y\ Z]^T$ and the points projected onto the image plane $\mathbf{m}' = [x'\ y']^T$ can be established by the following triangular similarity as

$$x' = \frac{f \cdot X}{Z},\ y' = \frac{f \cdot Y}{Z}, \tag{2.1}$$

where f is the focal length (in meters), (x', y') are the projected points (in meters) on the image plane, and (X, Y, Z) (in meters) are the 3D points of the object.

In an ideal case, where the camera is normalized, i.e., $f = 1$ meter, the perspective projection can

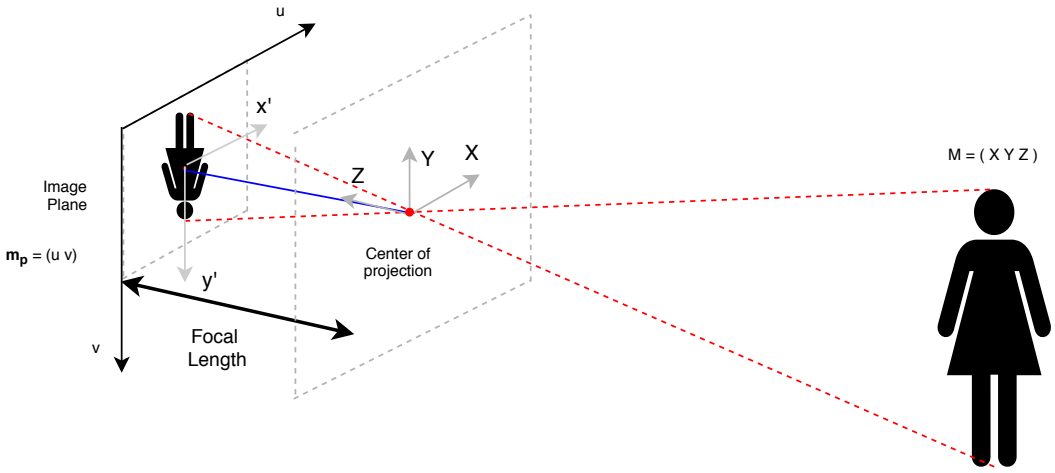

Figure 2.1: Pinhole Camera Model. The girl defined by the generic set of points M is projected onto the image plane (defined by the set of generic points $\mathbf{m_p}$) through the light rays passing through the camera's pinhole, called the center of projection. The focal length is the distance from the center of the image plane to the center of projection. The digital image's coordinate system (u, v) defined by the camera is not centered on the image plane. The optical axis is the blue line.

be defined, using homogeneous coordinates (represented by "$\sim$"), as

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \Leftrightarrow \lambda \widetilde{\mathbf{m}} = [I\ 0] \widetilde{M} = M. \tag{2.2}$$

with $\lambda = Z$, where $\mathbf{m}$ represents $\mathbf{m}'$ normalized.

However, in reality the image plane reference system is usually different from the digital image one, as shown in Figure 2.1, thus necessitating to translate it by $(c_x\ c_y)$ pixels. Also, the focal length is not normalized to one meter. Furthermore, the digital image obtained is actually expressed in pixel units, whereas the camera model is described in metric units, requiring the points to be converted by scalings (s_x, s_y). And finally, due to sensor errors, there might be a deformation skew, s, associated to the camera coordinate system, which means that it may not have exactly perpendicular axes. All these aspects are taken into consideration through the so-called **intrinsic parameters** representation, which is described as,

$$K = \begin{bmatrix} fs_x & s & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

resulting in,

$$\begin{aligned} u &= fs_x x' + sy' + c_x \\ v &= fs_y y' + c_y \end{aligned}, \tag{2.4}$$

where (u, v) are the shifted image coordinates in pixels, denoted by $\mathbf{m_p} = [u\ v]^T$ and (x', y') are in metric units, denoted earlier by $\mathbf{m}'$.

On top of this, the world frame, where object M is located, might at an arbitrary position and orientation, so it is necessary to convert it to the camera coordinate frame. This conversion consists on expressing the origin of the world coordinate frame in camera coordinates, through a translation $\mathbf{t}$, and the rotation between the two frames, through a rotation matrix, R. These are called the **extrinsic pa-**

rameters. Combining this, the full perspective projection, or camera model, is defined as

$$\begin{array}{l} \lambda \widetilde{\mathbf{m}}_{\mathbf{p}} \sim K[R|t]\widetilde{M} \\ \lambda \widetilde{\mathbf{m}}_{\mathbf{p}} \sim P\widetilde{M} \end{array}, \tag{2.5}$$

where P is a 3×4 matrix, named **camera matrix**, and λ is a scale factor. $\mathbf{m_p}$ may be expressed as,

$$u = \frac{\mathbf{p_1}^T \widetilde{M}}{\mathbf{p_3}^T \widetilde{M}},\ v = \frac{\mathbf{p_2}^T \widetilde{M}}{\mathbf{p_3}^T \widetilde{M}}, \tag{2.6}$$

where $\mathbf{p_i}^T$ denotes the row i of P.

In practice, the camera model contains several nonlinear effects, generically denoted as "distortion", since real cameras have lenses instead of a pinhole. The lens allows all the emitted rays of light to be refracted into one converging point, the focal point. Although the focal length will now increase to be the sum of the distance from the lens to the converging point, and from that point to the image plane, the camera model explained above still applies, but the distortions must be accounted for. These distortions are incorporated in (2.7). Radial distortion is modeled by the k_i coefficients on the left-hand side of the polynomial transformation in (2.7) (taken from Zernike's model of aberrations), and the tangential distortion (which occurs when the image plane is not parallel to the camera lens) is defined by the p_j coefficients in the right-hand term.[1]

$$\begin{array}{l} x'_d = x' \ \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + (2p_1 x' y' + p_2(r^2+2x'^2)) \\ y'_d = y' \ \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + (p_1(r^2+2y'^2) + 2p_2 x' y') \end{array}, \tag{2.7}$$

where $r^2 = x'^2 + y'^2$ and x'_d and y'_d, are the distorted coordinates of the image point.

Summarizing all the concepts, to obtain a digital camera image from the 3D world corresponding points, the steps are:

1. Project the 3D points onto the image plane using $(R, \mathbf{t})$: $M = [X\ Y\ Z]$ to $\mathbf{m}' = [x'\ y']^T$;
2. Distort the points to compensate for what happens in reality by using (2.7): $\mathbf{m}' = [x'\ y']^T$ to $\mathbf{m_d}' = [x'_d\ y'_d]^T$;
3. Transform the image plane coordinates into image pixel coordinates using the intrisic parameters: $\mathbf{m_d}' = [x'\ y']^T$ to $\mathbf{m_p} = [u\ v]^T$.

On the following sections, $\mathbf{m_p}$ will be noted just as $\mathbf{m}$ for simplicity.

2.2 Orthogonal Procrustes Problem

After having established the camera model, it is now time to discuss some potential methods of determining image transformations, as needed for this project. The first of these methods is called the Orthogonal Procrustes Problem. Procrustes, a son of Poseidon in Greek mythology, made his victims fit in a wonderful all-fitting bed, either by stretching their limbs, or by cutting them off. Ultimately, he was fitted into his own bed by Theseus. Similar to the analysis held here, there are 3 elements to Procrustes's problem: M_1, the bed, M_2, the unlucky adventurer, and T, the fitting treatment. The solution to the problem is a matrix, T, that minimizes

$$\|M_1 - TM_2\|^2, \tag{2.8}$$

[1] Born, Max; Wolf, Emil. Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light. Chapter 5.

which transforms M_2 into M_1.

Concerning the project's goal, M_1 and M_2 are clouds of 3D points observed by the camera obtained when looking at the same scene from two different points of view after a rotation, defined as R. Then the function to minimize is

$$\|M_1 - RM_2\|^2 \tag{2.9}$$

that through the Frobenius norm can be expanded as

$$\|M_1 - RM_2\|^2 = \operatorname{trace}(M_1^T M_1 + M_2^T M_2) - 2\operatorname{trace}(M_2^T M_1 R), \tag{2.10}$$

This means that minimizing (2.8) with respect to R is equivalent to maximizing the second term of (2.10). By applying singular value decomposition (SVD) to $M_2^T M_1$, the latter term can be further simplified into

$$\operatorname{trace}(M_2^T M_1 R) = \operatorname{trace}(U\Sigma V^T R) = \operatorname{trace}(\Sigma V^T R U) = \operatorname{trace}(\Sigma H) = \sum_{i=1}^{N} \sigma_i h_{ii}. \tag{2.11}$$

The singular values of σ_i are all non-negative, and so the expression becomes maximum when $h_{ii} = 1$ for $i = 1, 2, ..., N$, since H, a product of orthogonal matrices, is an orthogonal matrix itself, thus having its maximal value when $H = I$. This results in $I = V^T R U$, and so $R = VU^T$.

2.2.1 Translation

Often, M_1 and M_2 may be associated with different origins. Hence, it is customary to consider better fits of the configurations by allowing shape-preserving translations of the origin. When translating M_1 and M_2 by t_1 and t_2, respectively, (2.8) becomes

$$\left\|(M_1 - \mathbf{1}\mathbf{t_1}^T) - R(M_2 - \mathbf{1}\mathbf{t_2}^T)\right\|, \tag{2.12}$$

which can also be written as

$$\left\|M_1 - RM_2 - \mathbf{1}\mathbf{t}^T\right\|, \tag{2.13}$$

where $\mathbf{t}^T = \mathbf{t_1}^T R - \mathbf{t_2}^T$. The expression is minimized when $\mathbf{t}^T$ corresponds to the column-means of $M_1 R - M_2$, therefore a simple way of effecting this translation is to remove the column-means of M_1 and M_2, separately, before minimizing. This is called centering. The data then becomes expressed in terms of the mean deviations. [4]

2.3 Epipolar Geometry

Epipolar geometry provides an alternative yet powerful tool to obtain image transformations. It describes the relation between two views of a single scene through a 3x3 singular, non-invertible, matrix called the **essential matrix**, E, if the camera matrix is known, or the **fundamental matrix**, F, otherwise. These matrices are singular because they express an underconstrained relationship between a point in one image and its possible location in the other image, which is not unique due to depth ambiguity. The essential matrix is the product of a rotation matrix and a skew-symmetric matrix, whose determinant is zero, as it will be shown below. If a point in the (3D) world, M, is projected as point $\mathbf{m_1}$ in the first view, and point $\mathbf{m_2}$ in the second, then those points satisfy the relation

$$\tilde{\mathbf{m}}_\mathbf{2}^T F \tilde{\mathbf{m}}_\mathbf{1} = 0. \tag{2.14}$$

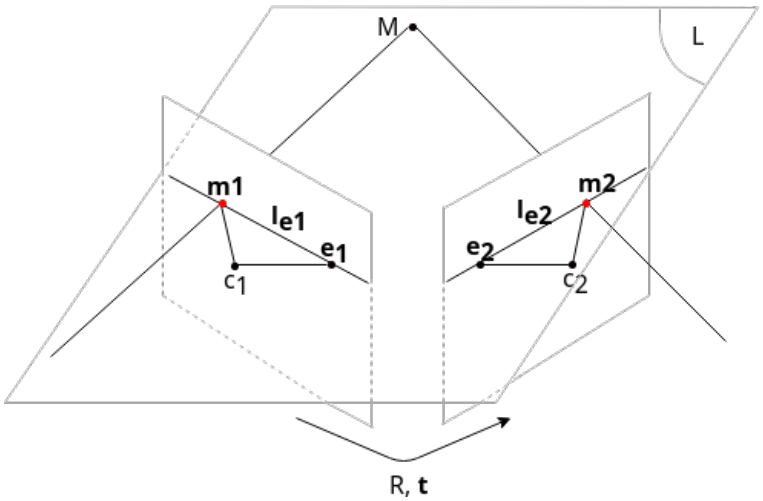

Figure 2.2: Epipolar geometry. The two small planes correspond to the image planes of two different views by the camera, separated by a rotation R and a translation $\mathbf{t}$, of the same scene, with centers of projection given by c_1 and c_2, respectively. M is the fixed point gazed at in the real world. $\mathbf{m_1}$ and $\mathbf{m_2}$ are the projections of point M into the respective image planes, whose rays are lying on the plane L. $\mathbf{l_{e1}}$ and $\mathbf{l_{e2}}$ are the epipolar lines, that also lie on the plane and $\mathbf{e_1}$ and $\mathbf{e_2}$ are the epipoles. The line defined by the centers is called the baseline.

In Figure 2.2, c_1 and c_2 are the centers of projection of the first and second views of the scene from one camera that has been displaced by $(R, \mathbf{t})$ (or by two cameras staring at the same scene). Given the point on the first view, $\mathbf{m_1}$, its correspondence to the second view is constrained to a line, which is called the epipolar line, $\mathbf{l_{e2}}$. This line is defined as the intersection of the plane L and the second view's image plane. Furthermore, plane L is delineated by the centers of projection of both views and $\mathbf{m_1}$. The epipoles, $\mathbf{e_1}$ and $\mathbf{e_2}$, are born from the intersection of the line $c_1 - c_2$ (baseline) with the epipolar lines, $\mathbf{l_{e1}}$ and $\mathbf{l_{e2}}$.

2.3.1 Projective geometry concepts

Homogenous representation of lines

A line in a plane is represented by $ax + by + c = 0$, which can also be defined by a vector $(a, b, c)^T$. For any non-zero constant, k, the vectors $(a, b, c)^T$ and $k(a, b, c)^T$ represent the same line and are equivalent. An equivalence class of vectors is known as an homogenous vector.

Point lying on a line

A point $\mathbf{m} = (x, y)^T$ lies on a line $\tilde{\mathbf{l}} = (a, b, c)^T$, if and only if $ax + by + c = 0$, which can be written as $(x, y, 1)(a, b, c)^T = \tilde{\mathbf{m}}^T\tilde{\mathbf{l}} = \tilde{\mathbf{l}}^T\tilde{\mathbf{m}} = 0$.

Line joining points

A line passing through any two points $\mathbf{m}$ and $\mathbf{m}'$ can be defined by the cross-product of those points as $\tilde{\mathbf{l}} = \tilde{\mathbf{m}} \times \tilde{\mathbf{m}}'$.

Cross product's skew symmetric representation

The previous cross product, $\tilde{\mathbf{l}} = \tilde{\mathbf{m}} \times \tilde{\mathbf{m}}'$, can be written as $\tilde{\mathbf{l}} = [\tilde{\mathbf{m}}]_\times \tilde{\mathbf{m}}'$, where $[\tilde{\mathbf{m}}]_\times$ is defined as

$$[\tilde{\mathbf{m}}]_\times = \begin{bmatrix} 0 & -m_3 & m_2 \\ m_3 & 0 & -m_1 \\ -m_2 & m_1 & 0 \end{bmatrix}. \tag{2.15}$$

2.3.2 Deducing the fundamental matrix

With the above concepts, the following conclusions may be drawn:

1. There is an homographic mapping via L plane from the points of the first to the second view, characterized by $\tilde{\mathbf{m}}_\mathbf{2} = H_M \tilde{\mathbf{m}}_\mathbf{1}$;
2. Because $\mathbf{m_2}$ lies on $\tilde{\mathbf{l}}_\mathbf{2}$, then $\tilde{\mathbf{m}}_\mathbf{2}^T \tilde{\mathbf{l}}_\mathbf{2} = 0$;
3. Since $\mathbf{e_2}, \mathbf{m_2} \in \tilde{\mathbf{l}}_\mathbf{2}$, then $\tilde{\mathbf{l}}_\mathbf{2} = [\tilde{\mathbf{e}}_\mathbf{2}]_\times \tilde{\mathbf{m}}_\mathbf{2}$.

From conclusions 1) and 3),

$$\tilde{\mathbf{l}}_\mathbf{2} = [\tilde{\mathbf{e}}_\mathbf{2}]_\times H_M \tilde{\mathbf{m}}_\mathbf{1} \tag{2.16}$$

can be deduced. Using conclusion 2) and (2.16)

$$\tilde{\mathbf{m}}_\mathbf{2}^T \tilde{\mathbf{l}}_\mathbf{2} = 0 = \tilde{\mathbf{m}}_\mathbf{2}^T [\tilde{\mathbf{e}}_\mathbf{2}]_\times H_M \tilde{\mathbf{m}}_\mathbf{1} = \tilde{\mathbf{m}}_\mathbf{2}^T F \tilde{\mathbf{m}}_\mathbf{1} \tag{2.17}$$

F is an homogeneous matrix of rank-2 (since $[\tilde{\mathbf{e}}_\mathbf{2}]_\times$ is rank-2 and H_M is rank-3) with 7 degrees of freedom, and $det(F) = 0$.

2.3.3 Relationship between the fundamental matrix and camera motion

Algebraically, having M expressed in the first camera's reference frame, under the camera model studied above, (2.18) can be deduced.

$$\lambda_1 \tilde{\mathbf{m}}_\mathbf{2} = K_1 [I\ \mathbf{0}] \tilde{M},\ \lambda_2 \tilde{\mathbf{m}}_\mathbf{2} = K_2 [R\ \mathbf{t}] \tilde{M} \tag{2.18}$$

In the case of only one camera in two different views: $K_1 = K_2$. For simplicity, the focal distance will be considered one meter, thus $K = I$ and the scale factors λ_2 and λ_1 will be dropped. Now by eliminating $\tilde{M}$, the previous equation becomes

$$\tilde{\mathbf{m}}_\mathbf{2} = R \tilde{\mathbf{m}}_\mathbf{1} + \mathbf{t}, \tag{2.19}$$

and because the cross product of two vectors is orthogonal to them both,

$$\tilde{\mathbf{m}}_\mathbf{2}^T \cdot (\tilde{\mathbf{m}}_\mathbf{2} \times \mathbf{t}) = 0 \tag{2.20}$$

$$(\tilde{\mathbf{m}}_\mathbf{2}^T \cdot ((R \tilde{\mathbf{m}}_\mathbf{1} + \mathbf{t}) \times \mathbf{t}) = 0, \tag{2.21}$$

the fundamental matrix, F, can be determined by

$$\begin{aligned} \tilde{\mathbf{m}}_\mathbf{2}{}^T [\mathbf{t}]_\times R \tilde{\mathbf{m}}_\mathbf{1} = 0 \\ \tilde{\mathbf{m}}_\mathbf{2}{}^T F \tilde{\mathbf{m}}_\mathbf{1} = 0. \end{aligned} \tag{2.22}$$

If the intrinsic parameters are not the identity matrix, then

$$\begin{aligned} \tilde{\mathbf{m_2}}^T K_2^{-T} E K_1^{-1} \tilde{\mathbf{m_1}} = 0 \\ F = K_2^{-T} E K_1^{-1}, \end{aligned} \tag{2.23}$$

where E is the essential matrix.

2.3.4 Estimating the fundamental matrix

In order to find the fundamental matrix, and consequently the transformation from one view to another, having a set of matched image points, $\mathbf{m_{i1}} = [u_1 \ v_1]^T$ and $\mathbf{m_2} = [u_{i2} \ v_2]^T$, the epipolar equation (2.14) can be written as

$$\begin{aligned} & \begin{bmatrix} u_2 & v_2 & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \\ = & \begin{bmatrix} f_{11}u_1u_2 + f_{12}v_1u_2 + f_{13}u_2 + f_{21}u_1v_2 + f_{22}v_1v_2 + f_{23}v_2 + f_{31}u_1 + f_{32}v_1 + f_{33} \end{bmatrix} \\ = & \begin{bmatrix} u_1u_2 \ v_1u_2 \ u_2 \ u_1v_2 \ v_1v_2 \ v_2 \ u_1 \ v_1 \ 1 \end{bmatrix} \begin{bmatrix} f_{11} \ f_{12} \ f_{13} \ f_{21} \ f_{22} \ f_{23} \ f_{31} \ f_{32} \ f_{33} \end{bmatrix}^T \\ = & \ \mathbf{u} \cdot \mathbf{f} \\ = & \ 0 \end{aligned} \tag{2.24}$$

For n sets of points the expression becomes

$$Uf = 0, \tag{2.25}$$

where $U = [\mathbf{u_1}, \cdots, \mathbf{u_n}]^T$.

This linear homogeneous equation, and the rank-2 constraint over F, that restrains the matrix to 7 degrees of freedom (since F is also defined up a scalar factor), will permit its unique identification.

The estimation of F may be done using only 7 point matches, or by using 8 or more if enough data points are available. In the latter case, the solution produced is in general unique, and there are several techniques to obtain it. In Zhang 1996's review on the issue [5], the conclusion was that linear techniques are usually sensitive to noise and not very stable, because they ignore the constraints of F and the minimization criterion is not physically meaningful. However, the results could be improved by using normalized data points instead of pixel coordinates.

Nevertheless, non-linear optimization techniques seem to yield better results to the estimation problem of F. Three nonlinear algorithms are available: (i) the distances between points and their corresponding epipolar lines, (ii) the gradient-weighted epipolar errors, and (iii) the distances between points and the reprojections of their corresponding points reconstructed in space. The third one is the most time-consuming method, thus not recommended. The first seems to give the worst results out of the three. Therefore, the second algorithm has been proposed to give the best results in the least amount of computational time.

Note that outliers, such as a wrongly matched pairs of points between the two views of the scene, or points with large location errors, could severely affect the precision of the estimation of F. The reason for this is that all methods are least-square techniques that assume the noise which corrupts the data has zero mean. Hence, more robust techniques have been proposed that are less sensitive to outliers, the most popular being M-estimators, and the least-median-of-squares (LMedS). The LMedS method solves a nonlinear minimization problem yielding the smallest value for the median of squared residuals

computed over the entire data set, and proves to be very robust both against false matches, and to erroneous locations, whereas M-estimators is less robust to false matches. LMedS must be solved by a search in the space of possible estimates generated from the data, which is too time-consuming. Thus, as a pragmatic approach is to only test a random sample of the data.

2.3.5 Factorization of the essential matrix

Once the fundamental matrix is obtained through one of those methods, assuming the intrinsic parameters are known, which is the case for this project, where the setup is only one camera, the essential matrix may be obtained by $E = K^T F K$ as seen on (2.23). From here, it's possible to retrieve the camera matrix, $P = [R \mid \mathbf{t}]$ from the first to the second view, since $E = [\mathbf{t}]_\times R$.

A $3x3$ matrix is an essential matrix if and only if two of its singular values are equal, and the third is zero. This is easily proven by the decomposition of $E = SR$, where S is a skew-symmetric matrix, as follows. Considering matrices

$$\mathrm{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathrm{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.26}$$

where W is orthogonal and Z is skew-symmetric, S may be written as[2] $S = kUZU^T$, where U is orthogonal and $Z = \operatorname{diag}(1,1,0)W$, up to sign. Thus, $S = U\operatorname{diag}(1,1,0)WU^T$, up to scale, and

$$E = SR = U\operatorname{diag}(1,1,0)\left(WU^T R\right) = U\operatorname{diag}(1,1,0)V^T, \tag{2.27}$$

proving the initial statement.

Because the singular values have to be equal, the SVD of E is not unique, in fact

$$E = U\operatorname{diag}(1,1,0)V^T, \tag{2.28}$$

or

$$E = U\operatorname{diag}(1,1,0)(-V)^T. \tag{2.29}$$

Considering (2.28), because

$$\begin{aligned} ZW &= \operatorname{diag}(1,1,0) \\ \text{and } ZW^T &= -\operatorname{diag}(1,1,0), \end{aligned} \tag{2.30}$$

$E = SR$ may be decomposed into two forms,

$$S = -UZU^T, \quad R = UW^TV^T, \tag{2.31}$$

or

$$S = UZU^T, \quad R = UWV^T. \tag{2.32}$$

The matrix R on (2.31) is a rotation, since it is orthogonal,

$$R^TR = \left(UW^TV^T\right)^T UW^TV^T = VWU^TUW^TV^T = I, \tag{2.33}$$

[2] A proof of this is given in Result A4.1 of R. Hartley and A. Zisserman,Multiple View Geometry in Computer Vision [6]

and

$$\det\left(UW^TV^T\right) = \det(U)\det\left(W^T\right)\det\left(V^T\right) = \det(W)\det\left(UV^T\right) = 1, \tag{2.34}$$

which are enough conditions to prove it. Furthermore, S is skew-symmetric because,

$$-S^T = \left(UZU^T\right)^T = UZ^TU^T = -UZU^T = S. \tag{2.35}$$

The same applies to (2.32). [7] A proof that these are the only solutions is given in Result 9.18 of R. Hartley and A. Zisserman [6].

Now regarding the translation, since $S\mathbf{t} = [\mathbf{t}]_\times\mathbf{t} = 0$, it follows that $t = U(0,0,1)^T = u_3$, the last column of U, which can be explained by the following. Assuming $S\mathbf{t} = UZU^T\mathbf{t} = 0$, $ZU^T\mathbf{t}$ should be equal to zero to satisfy the equation. Because Z is an orthogonal matrix,

$$ZU^T\mathbf{t} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T\mathbf{t} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} = 0, \tag{2.36}$$

where s is any non-zero constant, such as 1. Therefore, $\mathbf{t} = U[0\ 0\ 1]^T = u_3$, as U is orthogonal as well. Finally, due to the lack of uniqueness of the SVD, there are 4 possible solutions for the camera matrix,

$$P = \left[UWV^T|+u_3\right] \quad \text{or} \quad \left[UWV^T|-u_3\right] \quad \text{or} \quad \left[UW^TV^T|+u_3\right] \quad \text{or} \quad \left[UW^TV^T|-u_3\right], \tag{2.37}$$

that are represented by (a), (b), (c) and (d), respectively, on Figure 2.3.

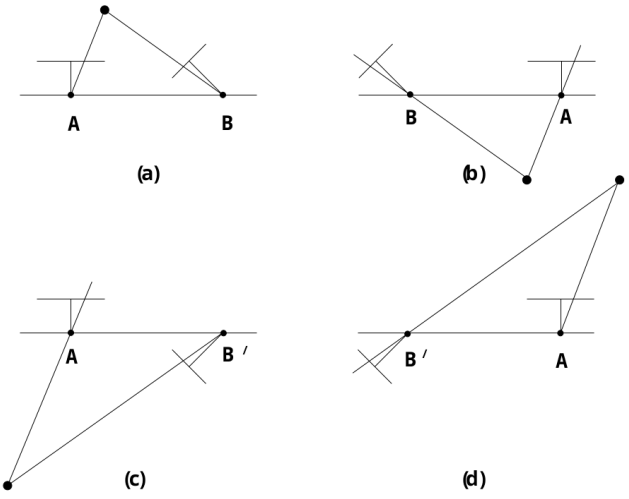

Figure 2.3: Four possible solutions retrieved from E. A and B are the first and second camera center positions when looking at the same scene. Between the right side, (a) and (c), and the left side, (b) and (d), figures, there is a translation direction inversion. Between top, (a) and (b), and bottom, (c) and (d), figures, there is a rotation of 180 degrees around the baseline. [6]

The real world point is only in front of both cameras in one of the four solutions, thus, it is enough to use a single point to decide which of the different solutions produces the correct camera matrix. [6]

To summarize, the steps for obtaining the projective transformation from epipolar geometry are:

1. Determining the fundamental matrix, F, using one of the algorithms described on section 2.3.4;
2. Obtain the essential matrix, E, from the intrinsic parameters (which is possible in this case);
3. Retrieve the four possible solutions for $P = [R\ \mathbf{t}]$ by doing a SVD decomposition on E;

4. Choose the feasible solution.

2.4 Feature Detection and Matching

Apart from a methods to determine image transformations from one view of a scene to another, described till now, it is also necessary to have solid detection and trust-worthy correspondence of the points in each view that are going to be used for computing that transformation.

Two main approaches exist to the problem of feature detection and matching. The first method finds features in one image and matches these features in the other image(s). The second method, which is more suitable for points in the near space, independently detects features in all images and subsequently matches them on the basis of local correspondences.

It might not be appropriate to extract features from an image having high detail (i.e. high spatial bandwidth) on the finest stable scale possible, because when matching those points with the view of the same scene at a different (coarser) scale those details may no longer be detectable. Therefore, it's important to extract features at a variety of scales to ensure their invariance.

Besides scale, preserving rotation and orientation invariance is also a concern. One way to deal with this problem is to define a descriptor. This is a scaled and oriented patch around the chosen point with the local orientation and scale, from which the dominant orientation can be extracted to guarantee its invariance. [8]

There exist a number of algorithms that can do the job. None of these algorithms, however, is optimal for all possible images, as each of them is optimized for particular computer vision applications, with performance depending heavily on the environmental properties (illumination, image quality, contrast, ...). A comprehensive survey on the state of the art of feature detection and description [9] (by Salahat E. and Qasaimeh M. in 2017), proposes that ideal features should have the following properties:

- Distinctiveness: the gradient variations surrounding the point should be sufficiently large;
- Locality: to avoid obstruction and deformation between two views of a scene;
- Quantity: enough features to describe the view's content;
- Accuracy: features should be accurate (what is accurate?) enough to be detected independently of image scale, shape or pixel location;
- Efficiency: be detected fast enough for real-time systems;
- Repeatability: a high percentage of features should be detected in both views of the overlapping regions in the scene;
- Invariance: deformative effects on the features, due to scaling, rotation or translation of the scene's view, are minimized;
- Robustness: features should be less sensitive to deformations due to noise, blur, compression, and so on.

The review paper also presents an overview on the recent algorithms proposed on this matter, comparing them in terms of the metrics defined above. The analysis in that article suggests that MSER (Maximally stable extremal regions) and SIFT (Scale Invariant Feature Transform) algorithms enhance performance on computational complexity, accuracy and execution time. From the scale and rotation invariant algorithms, SURF (Speeded Up Robust Features), proved to be faster than SIFT, although not as robust.

MSER[3] regions are areas of uniform intensity outlined by contrasting backgrounds. They are constructed by trying multiple thresholds on the input image. The ones that remain unchanged in shape over the range of thresholds are the potential features to be selected as areas of interest. The centroids of these areas can subsequently be used to create a feature descriptor for the matching step.

2.4.1 SIFT - Scale Invariant Feature Transform

The SIFT algorithm consists of a local feature detector and local histogram-based descriptor. At first, SIFT takes the original image, and generates progressively blurred images through a Gaussian filter as $L(u, v, \sigma) = G(u, v, \sigma) * I(u, v)$, where (u, v) are the pixel coordinates, L is the blurred image, I is the original image, G is the Gaussian blur, and finally σ is the amount of blur. Then, it re-sizes the original image to half its size, and generates the blurred images again, and repeats. Each new re-sizing of the previous image is scaled by an octave, and each octave can be blurred several times. This process ensures the scale invariance of potential features.

The set of blurred images obtained, then goes through the Difference of Gaussians (DoG) calculation, to find points of interest, or features, later on. To that end, the images from each octave (scale) are subtracted from each other pairwise, as illustrated in Figure 2.4. Mathematically, what happens is obtaining $D(u, v, \sigma) = L(u, v, k\sigma) - L(u, v, \sigma)$, where k is a constant multiplicative (scaling) factor.

Then, the search for local maxima/minima on the resulting images takes place. Each point is compared to its eight neighbors in the current image and nine neighbors in the image of one scale above and below, as Figure 2.5 shows. The points are only selected if they are larger than all of its neighbors or smaller than all of them.

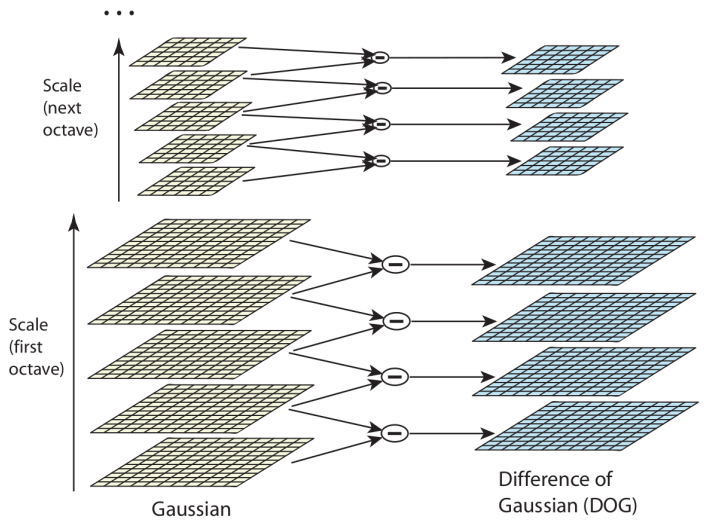

Figure 2.4: Difference of Gaussians (DoG) by octave. To find points of interest (features), the Gaussian-blurred images at each scale (= octave) are subtracted pairwise. This process is computationally more efficient than computing the Laplacian of the Gaussians. [10]

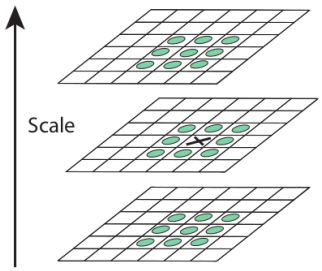

Figure 2.5: Search for the maxima and minima of the DoG. After the DoG, each point is compared with its eight neigbours, and with the 9 corresponding neighbours of the images of the scale above and below. [10]

The location of this maxima/minima is an approximation since it almost never lies on the exact pixel but somewhere between pixels. Thus, the next step is to search for the feature's exact location by using a Taylor expansion (up to the quadratic terms) of the scale-space function $D(u, v, \sigma)$, shifted from the

[3] J. Matas, O. Chum, M. Urban and T. Pajdla, "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions," 2002.

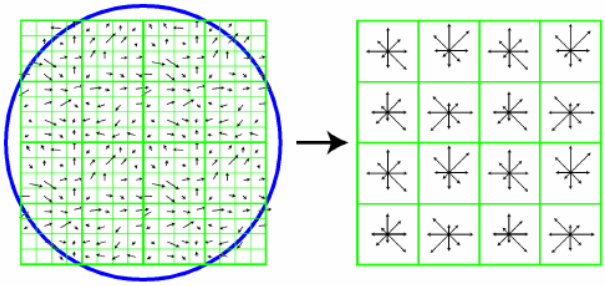

Figure 2.6: SIFT feature descriptor. A window of 16×16 squares is centered around the point of interest. The blue circle represents the Gaussain window. In each 4×4 sub-region, a histogram of orientations is determined, represented on the image on the right. In total there are $8 \times 16 = 128$ values to describe the feature. [10]

origin, according to

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}x. \tag{2.38}$$

D and its derivatives are determined at each point by using $\mathbf{x} = (u, v, \sigma)^T$ as an offset.

Apart from removing low-contrast sub-pixels just determined, it's also important to remove edges since the DoG has a strong response to them (delta functions). An unwanted peak in the DoG will have a large gradient perpendicular to the edge but a small gradient along the edge direction, and because of that property it can be easily detected and eliminated. This is accomplished by computing a Hessian matrix on the point.

Finally, to have a rotation invariant feature, it's necessary to create an oriented patch around the point. For each image, $L(u, v)$, at a certain scale, the gradient magnitude, m and orientation, θ, are determined as

$$\begin{aligned} m(u,v) &= \sqrt{(L(u+1,v) - L(u-1,v))^2 + (L(u,v+1) - L(u,v-1))^2} \\ \theta(u,v) &= \tan^{-1}((L(u,v+1) - L(u,v-1))/(L(u+1,v) - L(u-1,v))) \end{aligned} . \tag{2.39}$$

This information is used to create a histogram of orientations. Peaks on the histogram correspond to dominant directions of local gradients. Local peaks that are within 80% of the highest peak are used to create more features with that orientation.

For the descriptor of the feature, a 16×16 window around the point of interest is set. For each 4×4 region, the orientations are put into an 8 bin histogram (each bin with a 45 degree range). This can be seen through Figure 2.6. The length of each arrow corresponds to the sum of the gradient magnitudes near that direction within the region. Doing this for the 16 regions, there will be 128 values describing the feature. [10]

2.4.2 SURF - Speedup Robust Features

SURF starts from an integral image, which is an intermediate representation of the original image. The entry of an integral image $I_\Sigma(\mathbf{x})$ at a location $\mathbf{x} = (u, v)$ represents the sum of all pixels in the input image I of a rectangular region formed by the point $\mathbf{x}$ and the origin as

$$\mathrm{I}_\Sigma(\mathbf{x}) = \sum_{\mathrm{i}=0}^{\mathrm{i}\leq \mathrm{u}} \sum_{\mathrm{j}=0}^{\mathrm{j}\leq \mathrm{v}} \mathrm{I}(\mathrm{i},\mathrm{j}), \tag{2.40}$$

allowing a faster computation with box-type convolution filters.

The applied box filter is an approximation of Gaussian second-order derivatives. Instead, of having to

apply the Gaussian filter iteratively to the output image and to different scales as seen on SIFT, this can be done directly with the box filter by up-scaling it and altering its masks in parallel. Figure 2.7 makes a comparisson between the Gaussian second-order derivatives, Laplacians L_{yy} and L_{xy}, and a box filter applied on a point on the y and xy-direction, D_{yy} and D_{xy}.

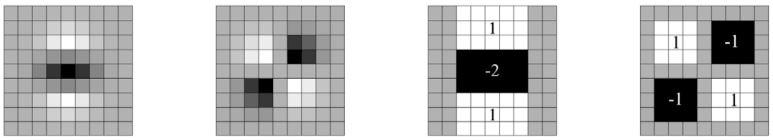

Figure 2.7: Comparison between Gaussian second order derivatives (from the left, the first image in y-direction - L_{yy} - and second image in xy-direction - L_{xy}) and the corresponding images (from the left, the third image in y-direction - D_{yy} - and fourth image in xy-direction - D_{xy}) using a box filter. [11]

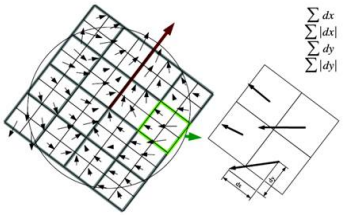

Figure 2.8: SURF feature descriptor. The sampling window is rotated towards the dominant orientation. The vector $(\sum d_x, \sum d_y, \sum_x |d_x|, \sum |d_y|)$ is generated per each region through the Haar-wavelet responses in x and y direction. [12]

The Hessian matrix at point $\mathbf{x}$ and scale σ is defined as

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}, \tag{2.41}$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second-order derivative $\frac{\partial^2}{\partial \mathbf{x}^2} g(\sigma)$ with the image I at point $\mathbf{x}$, and similarly for $L_{xy}(\mathbf{x}, \sigma)$ and $L_{yy}(\mathbf{x}, \sigma)$, which are weighted on the calculation of the determinant by $w = \frac{|L_{xy}(1.2)|_F |D_{yy}(9)|_F}{|L_{yy}(1.2)|_F |D_{xy}(9)|_F} \approx 0.9$. The determinant is then defined as

$$\det\left(\mathcal{H}_{approx}\right) = D_{xx} D_{yy} - \left(w D_{xy}\right)^2, \tag{2.42}$$

and is used to find the local change around a point. Points where the determinant is maximal are chosen as features.

SURF's feature descriptor was designed to be scale and rotation invariant. The descriptor is sampled over a window that is proportional to the image scale, so that when a scaled version of the feature in another image is found, the descriptor will be sampled relatively, satisfying the scale-invariance requirement.

To obtain rotation invariance, the sampling window is rotated towards the dominant orientation. The window size is $20s$, where s is the scale at which the point was detected, and is divided into a quadratic grid with 4$\times$4 square sub-regions. For each region, the Haar-wavelet responses are computed in the x and y directions. The descriptor is the sum of the x responses over its four quadrants, sum of the absolute values of the x responses and similarly for y, hence it may be defined as $(\sum d_x, \sum d_y, \sum_x |d_x|, \sum |d_y|)$. Having 4 values per region and 16 regions, yields a total of 64 values to describe the feature. Figure 2.8 helps at visualizing the process.

The dominant orientation, here lightly and carelessly explained, is captured through the calculation of Haar-wavelet responses over a circular region around the point of interest with a radius of $6s$. Those responses are then summed within a sliding orientation window covering an angle of $\frac{\pi}{3}$. [11] [12] [13]

2.4.3 Matching step

Regarding the matching of feature descriptors, one of the most common search algorithms used is the Nearest Neighbor Search. Fast Approximate Nearest Neighbour (FLANN)[4] provides a library for performing this kind of searches in high-dimensional spaces. It contains a collection of algorithms that the authors found to work best for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset. Pyramid Match Kernel[5] also seems to be successful approach to the matching problem. It forms a partial matching between two sets of feature vectors with linear time complexity.

2.5 Representation of eye orientations in 3 dimensions

The eye is a rotating body (no translations), and thus any orientation can be described by a unique series of three rotations around each of the axis defined in three dimensional space (roll axis, x, pitch axis, y, and yaw axis, z, respectively):

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix},\ R_y(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix},\ R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.43}$$

where each angle is defined in counter-clockwise direction around each axis. An arbitrary rotation may then be defined as some multiplication (i.e. a serial order) of those three, noting that the order by which they are multiplied matters (rotations are non-commutative).

Instead of using multiple rotations done after each other, Euler's theorem states that the orientation of the rotating body can also be parametrized by a single rotation with an angle, ρ, about an axis in 3D, $\hat{\mathbf{n}} = (n_1, n_2, n_3)$. That rotation is denoted as $R(\hat{\mathbf{n}}, \rho)$ and is given by

$$R(\hat{\mathbf{n}}, \rho) = \begin{pmatrix} \cos\rho + n_1^2(1-\cos\rho) & n_1 n_2(1-\cos\rho) - n_3\sin\rho & n_1 n_3(1-\cos\rho) + n_2\sin\rho \\ n_1 n_2(1-\cos\rho) + n_3\sin\rho & \cos\rho + n_2^2(1-\cos\rho) & n_2 n_3(1-\cos\rho) + n_2\sin\rho \\ n_1 n_3(1-\cos\rho) - n_2\sin\rho & n_2 n_3(1-\cos\rho) + n_1\sin\rho & \cos\rho + n_3^2(1-\cos\rho) \end{pmatrix}. \tag{2.44}$$

2.5.1 Rotation axis and angle

There are multiple ways by which to obtain the axis and angle of the rotation matrix. The following one uses the fact that a vector, $\mathbf{r}$, parallel to the rotation axis, $\hat{\mathbf{n}}$, is necessarily an eigenvector of the rotation matrix with the eigenvalue $\lambda = 1$, as described by

$$R\mathbf{r} = \mathbf{r}. \tag{2.45}$$

Rewriting (2.45) as $(R - I)\mathbf{r} = 0$, it can be shown that

$$0 = R^T 0 + 0 = R^T(R-I)\mathbf{r} + (R-I)\mathbf{r} = \left(R^T R - R^T + R - I\right)\mathbf{r} = \left(I - R^T + R - I\right)\mathbf{r} = \left(R - R^T\right)\mathbf{r}. \tag{2.46}$$

[4] M. Muja and D. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration,", 2009.
[5] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Efficient Learning with Sets of Features", 2005

Because $(R-R^T)$ is a skew-symmetric matrix, $\mathbf{r}$ may be chosen such that $[\mathbf{r}]_\times = \left(R - R^T\right)$ (this notation is described in (2.15)), and (2.46) then becomes a cross-product of vector $\mathbf{u}$ with itself,

$$\left(R - R^{\mathrm{T}}\right)\mathbf{r} = [\mathbf{r}]_\times \mathbf{r} = \mathbf{r} \times \mathbf{r} = 0, \tag{2.47}$$

obtaining

$$\mathbf{r} = \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix}, \tag{2.48}$$

as the rotation axis. This does not work if R is symmetric, to do so, it is necessary to diagonalize the matrix and find the eigenvector which corresponds to the eigenvalue of 1.

The rotation angle can be obtained through $\|\mathbf{r}\| = 2\sin\rho$, or through a more direct method by computing the trace of $R(\hat{\mathbf{n}}, \rho)$, defined as

$$\mathrm{Tr}\, R(\hat{\mathbf{n}}, \rho) = 1 + 2\cos\rho. \tag{2.49}$$

2.5.2 Rotation vector in head-fixed coordinates

There are other ways to represent a rotation that may be more appropriate for this project, such as the next one, which comes from Neuroscience. Here, there is a rotation vector, $\mathbf{r}$, directed along the rotation axis, $\hat{\mathbf{n}}$, which length varies with the amount of rotation, ρ, around it. A description of this vector is given by

$$\mathbf{r} = \tan(\frac{\rho}{2})\hat{\mathbf{n}}, \tag{2.50}$$

that can be obtained through the previous rotation matrix (2.48) as

$$\begin{aligned} \cos\rho &= 1/2 \cdot (R_{11} + R_{22} + R_{33} - 1) \\ n_x &= \sin\rho\,(R_{32} - R_{23})\,/2 \\ n_y &= \sin\rho\,(R_{13} - R_{31})\,/2 \\ n_z &= \sin\rho\,(R_{21} - R_{12})\,/2. \end{aligned} \tag{2.51}$$

Note that in this format, the units are given in half-radians.

2.5.3 Quaternions

Quaternions are 4-dimensional complex algebraic objects that are related to a rotation around an axis, $\hat{\mathbf{n}}$, just like the representation before, by an angle ρ as follows,

$$q = \cos(\rho/2) + i\sin(\rho/2)\hat{n} \equiv q_0 + \mathbf{i}\cdot\mathbf{q}, \tag{2.52}$$

where q_0 and $\mathbf{q}$ are the scalar and vectorial parts of the quaternion, respectively, and $\mathbf{i}$ is the complex 3D vector. For the description of eye movements, only the vectorial part is necessary. Quaternions are related to rotation vectors by $\mathbf{r} = \mathbf{q}/q_0$. [14]

2.5.4 Coordinate system

The three-dimensional coordinate systems typically used in Neuroscience and Computer Vision use different conventions, as shown on Figure 2.9. For the remainder of this section, the neuroscience

convention will be used to explain the next concepts.

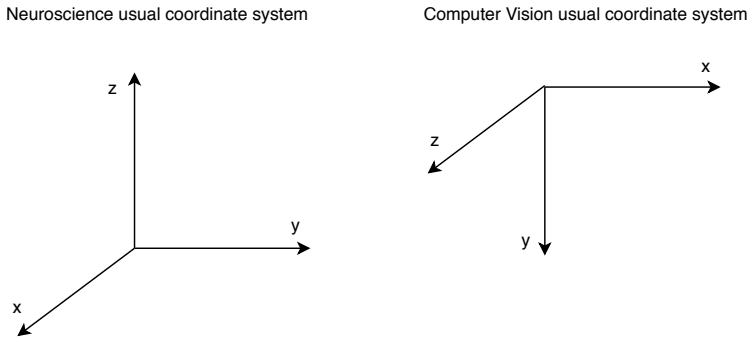

Figure 2.9: The neuroscience and computer vision 3 dimensional coordinate systems normally used are different; both use the right-hand rule. In Neuroscience, torsion is along the frontal visual x-axis, vertical angles along the inter-aural y-axis, and horizontal angles along the bottom-up z-axis.

2.5.5 Donders' and Listing's laws

As described in the beginning of this section, the orientation of a freely rotating rigid body can be quantified by a horizontal angle, θ, a vertical angle, ϕ, and a torsional angle, ψ (around the z, y and x, axis, respectively), relative to a starting position (the primary direction) for which all three angles are zero. Thus, 3 degrees of freedom.

However, as referred before, Donders' law states that the orientation of the eye, when looking in a specific direction at infinity, is always the same; in other words, as described by Donders: "with the head erect and looking at infinity, any gaze direction has a unique torsional angle, regardless the path followed by the eye to get there". Thus, the eye has 2 degrees of freedom for pointing.

Moreover, the 3D orientations of the eye can be uniquely described by head-fixed Cartesian coordinate system of single-axis rotations (section 2.5.2) that bring the eye from the primary position to the current orientation. Described in this way, Listing's law holds that the rotation axis corresponding to all possible eye orientations are confined to the yz-plane, called Listing's plane, in neuroscience. The primary position of the eye is an unique position from which any other eye position can be reached through rotation around an axis lying in Listing's plane. This hypothetical rotation axis is defined by a vector $\mathbf{r} = (r_x, r_y, r_z)$, where r_x, the torsional component perpendicular to the Listing's plane, is zero. Figure 2.10 illustrates that this law is very precise, by showing 3500 different eye orientations made by a head-restrained monkey, where the left image is the frontal view on the rotation axis, representing the gazed points coordinates of the horizontal, r_y, and vertical, r_z, components of the axis, and the right-hand plot is a side view, representing the coordinates expressed by the horizontal and torsional, r_x, components. A plane is clearly seen around $r_x = 0$ with little deviation ($\sigma_x \approx 0.6$ deg).

However, if the initial eye position is not the primary position, it should be noted that the dynamic angular velocity axis, ω, that rotates the eye from one orientation in the Listing's plane to another, e.g. during a saccadic eye movement, is **not** confined to the plane itself. It will typically have a torsional component, $\omega_x \neq 0$, as it depends on the initial eye position, $\mathbf{r_1} = (0, r_{1y}, r_{1z})$ and the saccade difference vector in Listing's Plane, $\mathbf{d_{12}} \equiv \mathbf{r_2} - \mathbf{r_1} = (0, d_{12y}, d_{12z})$ that carries the eye to the final eye position, resulting in

$$\omega_x = r_{1y} \cdot d_{12z} - r_{1z} \cdot d_{12y}. \tag{2.53}$$

This expression can be reached, using the representation on section 2.5.2, through the following:

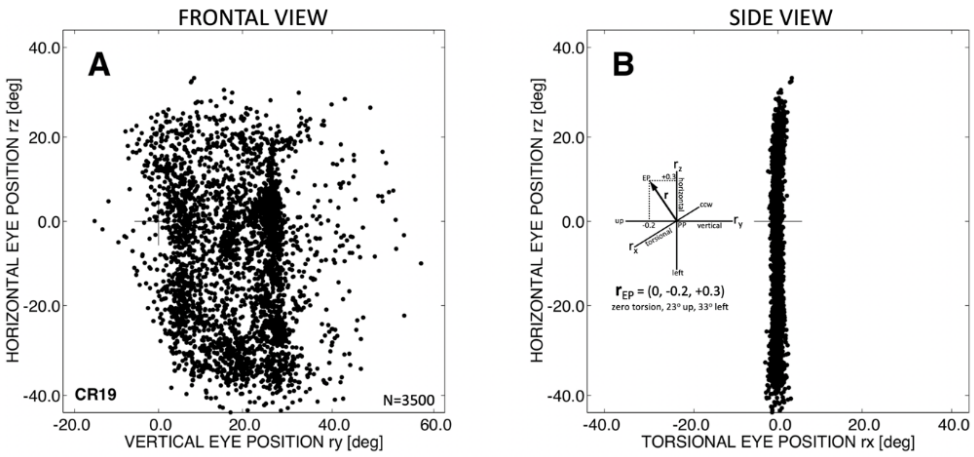

Figure 2.10: Listing's law, illustrated for 3500 eye orientations made by a head-restrained monkey, freely looking around in the laboratory room. Eye fixations are represented by the head-fixed Cartesian components of the 3D rotation axes, expressed in half-radians, and calculated as $angle = 2 \cdot atan(component)$. [3]

1. When rotating the eye, on the conditions that assure the existence of Listing's plane, from one position, $\mathbf{r_1}$, to another, $\mathbf{r_2}$, the resulting rotation axis will be $\mathbf{r_{12}} = \mathbf{r_1} \cdot \mathbf{r_2^{-1}}$;

2. A rotation vector can be obtained by

$$\mathbf{r_{21}} = \frac{\mathbf{r_2} - \mathbf{r_1} + \mathbf{r_1} \times \mathbf{r_2}}{1 + \mathbf{r_1} \cdot \mathbf{r_2}} \approx \mathbf{r_2} - \mathbf{r_1} + \mathbf{r_1} \times \mathbf{r_2} \equiv \mathbf{d_{12}} + \mathbf{r_1} \times \mathbf{d_{12}}, \tag{2.54}$$

   where $\mathbf{r_1} \cdot \mathbf{r_2} << 1$ for angles smaller then 30 degrees which is normally the case on eye rotations.

3. Expanding the previous equation to

$$\mathbf{r_{21}} = \begin{pmatrix} 0 \\ \mathbf{d_{12y}} \\ \mathbf{d_{12z}} \end{pmatrix} + \begin{pmatrix} 0 \\ \mathbf{r_{1y}} \\ \mathbf{r_{1z}} \end{pmatrix} \times \begin{pmatrix} 0 \\ \mathbf{d_{12y}} \\ \mathbf{d_{12z}} \end{pmatrix}, \tag{2.55}$$

   $\mathbf{r_{12x}} = \mathbf{r_{1y}} \cdot \mathbf{d_{12z}} - \mathbf{r_{1z}} \cdot \mathbf{d_{12y}}$ is obtained and corresponds to the angular velocity component $\omega_\mathbf{x}$.

[3] [14]

Chapter 3

Method and Proposal

In this section, the Computer Vision convention of 3D body orientation (referred in section 2.5) is used for the sake of clarity.

The three main aspects that constrain the estimation accuracy of the pose transformation between views and the computational time in which this can be accomplished, are

- the feature detection in both images of the respective views;
- the matching of corresponding features;
- and the computation of the camera orientation through the set of feature pairs.

Hence, the project will be divided into sub-projects that optimize precision/accuracy and speed for each of these aspects. In this section, the state of the art described previously is taken into consideration, and a comparison between the various techniques outlined for each of the three aspects, as well as, a novel method for computing the camera orientation that is applicable by exploiting the particular kinematics of the eye model, is proposed. The final aim is to study the benefits/disadvantages of using a camera instead of the current IMU sensor, and to understand how they could be best used in a complementary manner to reach better estimates for the eye's orientation.

Regarding the camera orientation, two methods were presented in the state of the art, Orthogonal Procrustes Problem and Epipolar Geometry. The first one may be suitable enough for pure rotations. However, as stated in section 1.3, in the current eye model, the camera is not fixed to the center of the eye, so there is also a translation component associated with the transformation. In order to know the exact rotation, this issue could be solved by eliminating the translation, as it was described in subsection 2.2.1. Yet, the camera that is being utilized on the project, does not provide depth information. Therefore, to use the Procrustes, it's necessary to reconstruct the 3D coordinates of observed points and this is only possible if the transformation is a pure rotation, when the projection of 3D points in the image does not depend on their depth. In such cases, epipolar geometry would have to be relied on. Epipolar geometry actually demands the existence of a translation component in order to work, since the transformation is taken from the essential matrix, $E = [\mathbf{t}]_{\times} R$, that depends on both the translation and the rotation. For that reason, a novel method that exploits the particular eye assembly, where the translation can be written as a function of the rotation is proposed and will compared with the other two.

The first step to the propounded technique is to find the relationship between the camera orientation and the camera translation. To do so, the diagram of Figure 3.1 is considered. Let the World reference frame W have its origin in the center of rotation of the eye and be aligned with the eye at the rest position. The transformation from the world, W, to the camera's reference frame on the first view of the scene,

C_1, shown in the figure, is defined by

$$ {}^{W}_{C_1}T = \begin{bmatrix} I & \mathbf{b} \\ 0 & 1 \end{bmatrix}, \tag{3.1} $$

where $\mathbf{b} = [b_x\ b_y\ b_z]^T$ is the baseline. Now, by applying a rotation R to the eye around the origin on the world reference frame,

$$ {}^{W}_{C_2}T = \begin{bmatrix} R & R\mathbf{b} \\ 0 & 1 \end{bmatrix}, \tag{3.2} $$

is the transformation from the world to the second-view's camera reference frame, C_2.

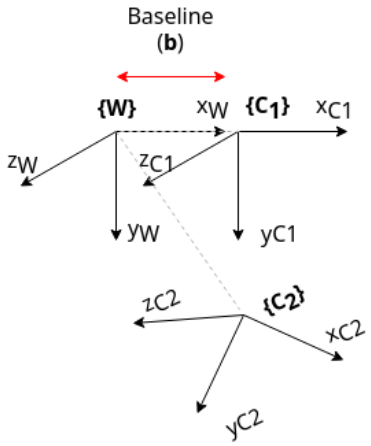

Figure 3.1: Hypothetical coordinate reference frames of the world W, the camera looking at a scene the first view C_1 and the camera looking at the same scene through the second view C_2 after the transformation. Baseline, $\mathbf{b}$ is represented as a translation in the x direction only, but in reality it's a vector defined by $[b_x\ b_y\ b_z]^T$. In this case the rotation from W to C_1 is the identity.

Now, the transformation from the first view to the second view of the scene can be represented by ${}^{C_1}_{C_2}T = {}^{C_1}_{W}T\ {}^{W}_{C_2}T = ({}^{W}_{C_1}T)^T\ {}^{W}_{C_2}T$, which is equivalent to

$$ {}^{C_1}_{C_2}T = \begin{bmatrix} I & -\mathbf{b} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R & R\mathbf{b} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R & R\mathbf{b} - \mathbf{b} \\ 0 & 1 \end{bmatrix}. \tag{3.3} $$

Thus, the translation component can be characterized as a function of the rotations described by the camera as

$$ \mathbf{t} = (R - I)\mathbf{b}. \tag{3.4} $$

This constraint can be used to minimize the distances between the matching pairs of image points, initialized by the rotation matrix determined with the Orthogonal Procrustes Problem. This proposed technique will be called **Minimization of the Back Projection Error**, and may be defined as

$$ \underset{R,\mathbf{b},Z_i}{\text{minimize}} \quad J(\mathbf{m_{i1}}, \mathbf{m_{i2}}, R, \mathbf{b}, Z_i), \tag{3.5} $$

where Z_i is the depth of the points when looking at the scene from the first view, which is an unknown quantity, since the camera doesn't gather any direct depth information, $\mathbf{m_{i1}} = [u_{1i}\ v_{i1}]^T$ and $\mathbf{m_{i2}} = [u_{i2}\ v_{i2}]^T$ are the matched points in the images taken from both views and index i refers to a generic point varying from 1 to N pairs of matched points.

To understand the objective function, two real world points, $M_1 = [X_1\ Y_1\ Z_1]^T$ and $M_2 = [X_2\ Y_2\ Z_2]^T$ stared at from the first and second view of the scene, respectively, are considered. These points are

related by R and $\mathbf{t}$ as

$$\begin{aligned} M_2 &= RM_1 + \mathbf{t}, \\ M_1 &= R^T M_2 - R^T \mathbf{t}. \end{aligned} \tag{3.6}$$

Considering that the intrinsic parameters matrix of the camera is the identity (focal length of one meter), the real world points can be obtained from the image points through

$$\begin{aligned} M_1 &= Z_1 \widetilde{\mathbf{m}}_\mathbf{1}, \\ M_2 &= Z_2 \widetilde{\mathbf{m}}_\mathbf{2}. \end{aligned} \tag{3.7}$$

Using both (3.6) and (3.7), the points may be back projected to retrieve the image points on the complementary image planes as

$$\begin{aligned} Z_1 \widetilde{\mathbf{m}}'_\mathbf{1} &= R^T Z_2 \widetilde{\mathbf{m}}_\mathbf{2} - R^T \mathbf{t}, \\ Z_2 \widetilde{\mathbf{m}}'_\mathbf{2} &= R Z_1 \widetilde{\mathbf{m}}_\mathbf{1} + \mathbf{t}. \end{aligned} \tag{3.8}$$

where $\widetilde{\mathbf{m}}'_\mathbf{1}$ and $\widetilde{\mathbf{m}}'_\mathbf{2}$ represent the estimation of $\widetilde{\mathbf{m}}_\mathbf{1}$ and $\widetilde{\mathbf{m}}_\mathbf{2}$. Resulting in the following values for $\widetilde{\mathbf{m}}'_\mathbf{1} = [u'_1\ v'_1]$,

$$\begin{aligned} u'_1 &= \frac{Z_2 \mathbf{r_1} \widetilde{\mathbf{m}}_\mathbf{2} - \mathbf{r_1 t}}{Z_1}, \\ v'_1 &= \frac{Z_2 \mathbf{r_2} \widetilde{\mathbf{m}}_\mathbf{2} - \mathbf{r_2 t}}{Z_1}, \end{aligned} \tag{3.9}$$

and in the values

$$\begin{aligned} Z_2 &= Z_1 \mathbf{r_3}^T \widetilde{\mathbf{m}}_\mathbf{1} - \mathbf{t_3}, \\ u'_2 &= \frac{Z_1 \mathbf{r_1}^T \widetilde{\mathbf{m}}_\mathbf{1} + \mathbf{t_1}}{Z_1 \mathbf{r_3}^T \widetilde{\mathbf{m}}_\mathbf{1} + \mathbf{t_3}}, \\ v'_2 &= \frac{Z_1 \mathbf{r_2}^T \widetilde{\mathbf{m}}_\mathbf{1} + \mathbf{t_2}}{Z_1 \mathbf{r_3}^T \widetilde{\mathbf{m}}_\mathbf{1} + \mathbf{t_3}}, \end{aligned} \tag{3.10}$$

for $\widetilde{\mathbf{m}}'_\mathbf{2} = [u'_2\ v'_2]$ and Z_2, where $\mathbf{r_j}^T$ represents the row j of R. To note that Z_2 is expressed by Z_1. Finally, remembering that the translation is a function of the rotation (3.4), the objective function to minimize for i point matches can be defined by,

$$J(\mathbf{m_{1i}}, \mathbf{m_{2i}}, R, \mathbf{b}, Z_{1i}) = \sum_{i=1}^{N} \left[(u'_{1i} - u_{1i})^2 + (u'_{2i} - u_{2i})^2 + (v'_{1i} - v_{1i})^2 + (v'_{2i} - v_{2i})^2 \right]. \tag{3.11}$$

Chapter 4

Preliminary Work

4.1 Specifications

4.1.1 Hardware

The IMU sensor on the current eye model is LP-Research Motion Sensor CAN bus and USB version (LPMS-CU)[1] being used through the USB 2.0 Bus.

The camera handled was an IDS (Image Developing Systems) uEye[2] model UI-3271LE-C-HQ-VU.

The computer being used to gather the data is an Intel NUC Kit NUC6i7KYK[3] running on Windows 10 OS.

4.1.2 Software

OpenCV (Open Source Computer Vision) Library[4] was used for the programs implemented, using C++. It was mandatory to use the IDS uEye software libraries to handle the cameras and the LPMS software[5] library LPSensor version 1.3.4 (!) to work with the IMU sensor.

All the work developed regarding software is available at https://github.com/Mrrvm/IEEC-prework.

The code developed to capture images from the camera takes an average of 0.12s to execute, including allocating temporary memory, freezing the video to take the picture, passing it to an OpenCV structure necessary later on and freeing the memory allocated.

To calibrate the camera using a 4mm lens, one set of 65 chessboard (of 8x8 squares with 20mm square length) images was taken for model 3271LE-C-HQ-VU. It calibrated with 96% of images of the set, according to the number of chessboards detected.

4.2 Comparison of rotations around each axis between the IMU sensor and the camera

An initial experiment to compare the accuracy of the eye rotation obtained from the camera against the the one obtained from the IMU, was implemented as follows.

[1] http://lp-research.com/lpms-cu/
[2] https://www.prolinx.co.jp/supplier/IDS/uEye_Manual_En/index.html?hw_modellbezeichnungen.html
[3] https://www.intel.com/content/www/us/en/products/docs/boards-kits/nuc/nuc-kit-nuc6i7kyk-features-configurations.html
[4] https://opencv.org/
[5] https://bitbucket.org/lpresearch/openmat/downloads/

4.2.1 Methodology

Undistorting the images

Two images were taken before and after the eye rotation and their pixel coordinates were undistorted using the function *undistort()* from the OpenCV Geometric Image Transformations Library[6]. This function takes into consideration the intrinsic parameters, and returns the images undistorted in pixel coordinates.

Obtaining point matches

SURF[7] algorithm was implemented in order to detect the image's points of interest. Using FLANN[8], the point matches were determined. The minimal distance between both images features was calculated, to further enhance the matching. That distance is the euclidean distance between two descriptors $\mathbf{a}$ and $\mathbf{b}$, and represents the score of similarity between them, as

$$\sqrt{\sum_{i=1}^{64} |a_i - b_i|^2} \tag{4.1}$$

for SURF's 64 dimensions (see section 2.4.2). If the pair of points distances less than double the minimal distance or small arbitrary value (0.02) in the event that the distance is very small, it's defined as a good feature to keep [15].

Determining the artificial 3D points

Since the camera provides no depth information, the 3D points had to be generated artificially using a sphere projection with radius 1, as explained next. Firstly, the undistorted pixel coordinates of the obtained interest points were transformed into image plane points through the intrinsic parameters matrix, K, as $K^{-1}\widetilde{\mathbf{m}}_{\mathbf{p}} = \widetilde{\mathbf{m}}'$. Then, the value for the depth, λ, when the norm of $\widetilde{\mathbf{m}}'$ is 1, was determined by

$$\|(\lambda x, \lambda y, \lambda)\| = 1, \lambda = \frac{1}{\sqrt{x^2 + y^2 + 1}}, \tag{4.2}$$

and applied to the image points. This process was done for both images.

Implementing the Orthogonal Procrustes Problem

The Orthogonal Procrustes problem was implemented using the SVD class[9] from OpenCV as described on section 2.2.

In order to test this implementation, the structure on Figure 4.1 was used to provoke rotations around each axis, torsional, horizontal and vertical. Because neither the camera nor the sensor are centered at the joint of the structure, there is a translation associated to the rotations. The experiments are described below.

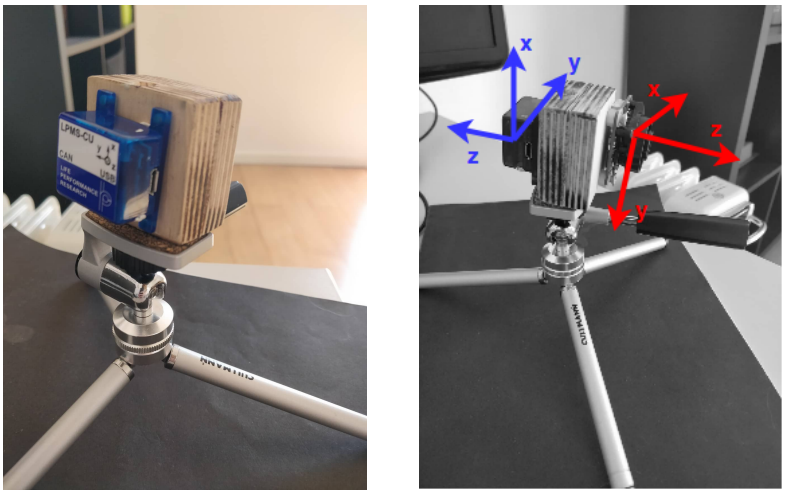

Figure 4.1: Structure used for making the experiments. The image on the right shows the different coordinate systems of the IMU sensor and the camera.

4.2.2 Experiments and results

For all tables, θ indicates the rotation angle magnitude, θ_{Cam} is the absolute rotation angle obtained by the camera using the Procrustes problem without considering translation and θ_{IMU} is the absolute rotation angle determined by the IMU sensor.

Rotation around the torsional axis

Table 4.1: Rotation angle obtained when rotating the structure around the torsional axis To obtain 90^o, a paper strip was placed vertically and the structure was rotated until the strip appeared horizontally on the image.

θ	θ_{Cam}	θ_{IMU}	$n_{matches}$
90^o	86.76^o	90.9^o	7

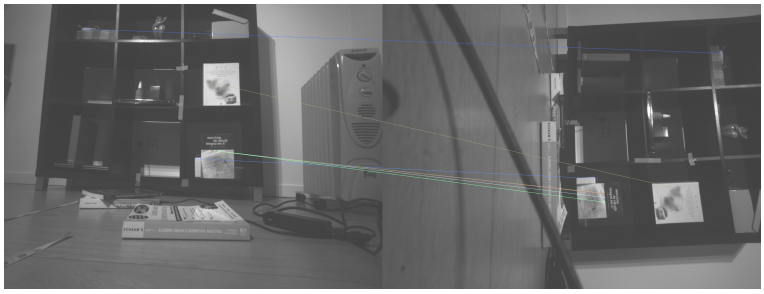

Figure 4.2: Matches, represented by colored lines, obtained for rotating 90^o around the torsional axis.

Table 4.1 shows that, the IMU sensor gave a closer approximation to the expected value. Due to the environment conditions, it wasn't possible to find more than 7 matches between the 2 images (see figure 4.2), which then lead to a poorer estimation through the camera.

[6]https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html
[7]https://docs.opencv.org/3.4/d5/df7/classcv_1_1xfeatures2d_1_1SURF.html
[8]https://docs.opencv.org/2.4/modules/flann/doc/flann_fast_approximate_nearest_neighbor_search.html
[9]https://docs.opencv.org/3.4.2/df/df7/classcv_1_1SVD.html

Rotation around the horizontal axis

Table 4.2: Rotation angle obtained when rotating the structure around the horizontal axis. To obtain 30^o, the camera was placed at 1 meter away from the grid and the camera was rotated along the vertical axis until the center of the image observes a point located $57cm$ away in the grid. To obtain 5^o, the camera was placed at 4 meters away from the grid and the center of the image was shifted by $35cm$.

θ	θ_{Cam}	θ_{IMU}	$n_{matches}$
30^o	31.75^o	10.63^o	11
5^o	5.27^o	6.12^o	27

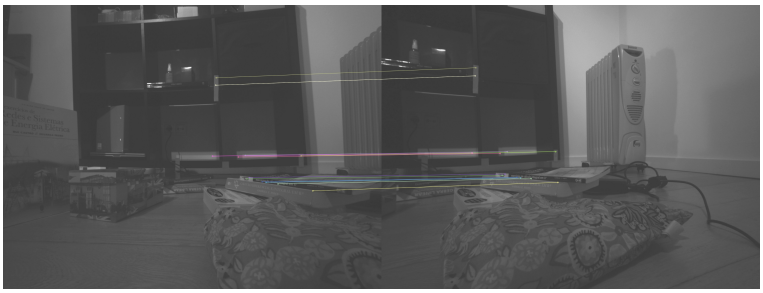

Figure 4.3: Matches, represented by colored lines, obtained for rotating 30^o around the horizontal axis.

Figure 4.4: Matches, represented by colored lines, obtained for rotating 5^o around the horizontal axis.

In table 4.2, the estimation through the IMU for the 30^o rotation is very far from what was expected, and might be caused by the fact that the IMU could not use gravity to determine the orientation since it's horizontal. Figures 4.3 and 4.4 show the matches obtained between the images for the 30^o and 5^o rotation, respectively.

Rotation around the vertical axis

Table 4.3: Rotation angle obtained when rotating the structure around the vertical axis. To obtain 19^o, the camera was placed at 1 meter away from the grid and the camera was rotated along the horizontal axis until the center of the image observes a point located $35cm$ away in the grid. To obtain 5^o, the camera was placed at 4 meters away from the grid and the center of the image was shifted by $35cm$ vertically.

θ	θ_{Cam}	θ_{IMU}	$n_{matches}$
19^o	21.37^o	16.56^o	23
5^o	5.41^o	5.35^o	28

Lastly, table 4.3, shows improvement when using the the camera to determine rotations around the vertical axis. Figures 4.5 and 4.6 show the matches obtained between the images for the 19^o and 5^o

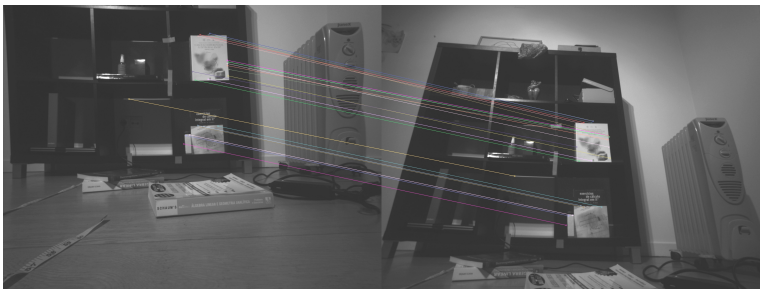

Figure 4.5: Matches, represented by colored lines, obtained for rotating 19^{o} around the vertical axis.

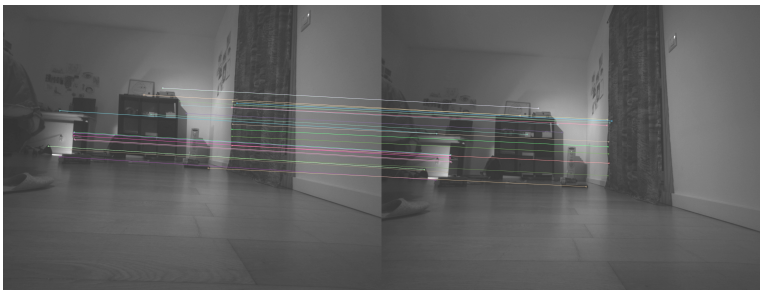

Figure 4.6: Matches, represented by colored lines, obtained for rotating 5^{o} around the vertical axis.

rotation, respectively.

Overall, except for the torsional case where there were very few matches, the camera proved to gather better approximations to reality in relation to the sensor, even considering the fact that there is translation associated to the movement. However, further experiments, with a better environment and using a different structure to hold the camera and the sensor must be done, in order to get more accurate results. Because the structure has to be moved by hand, it is very prone to error. The experiments weren't done directly on the eye model, as it still needs some adaptations to support the camera in its structure.

Chapter 5

Thesis Development Plan

As mentioned on the introduction, the IMU sensor has drift associated and since there is complementary parallel work being developed on this research topic, it is necessary to have a good working algorithm to estimate the camera transformation from view to view at reasonable speed. Therefore, out of necessity, this project, as it already has by the preleminary work, starts by determining a good computation method for the transformation between views of the camera.

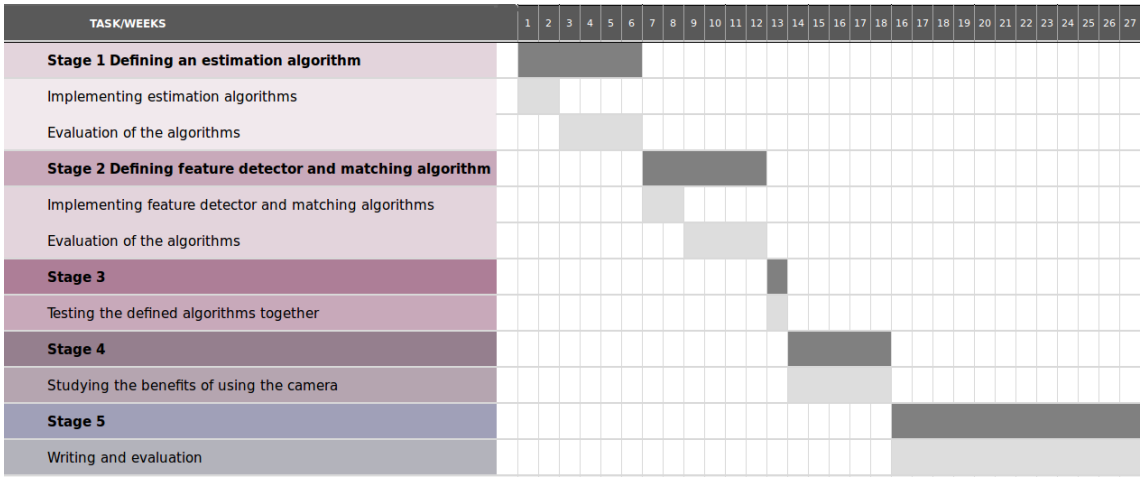

Figure 5.1: Gantt chart of the thesis development. Starts in 18^{th} February 2019.

Keeping that in mind, the development of the thesis can be separated into the following main stages,

1. Defining an estimation algorithm for the camera views transformations;
2. Defining a feature detector and matching algorithm;
3. Testing the conjunction of 1 and 2 over accuracy and speed;
4. Studying the advantages/disadvantages of the camera over the IMU sensor or potential symbiotic relation
5. Evaluation and Writing,

organized into the Gantt chart in Figure 5.1.

Bibliography

[1] J. van Opstal. http://www.mbfys.ru.nl/~johnvo/OrientWeb/orient_1.html#Abstract. Accessed on December 15th 2018.

[2] M. Lucas, "Construction and Characterization of a Biomimetic Robotic Eye Model with Three Degrees of Rotational Freedom: A Testbed for Neural Control of Eye Movements," 2017.

[3] J. van Opstal, "Editorial: 200 Years Franciscus Cornelis Donders," Donders Center for Neuroscience, Dept. of Biophysics, Radboud University Nijmegen, The Netherlands, December 2018.

[4] J. Gower and G. Dijksterhuis, *Procrustes Problems*. Oxford Statistical Science Series, 2004. Chapter 3 and 4.

[5] Z. Zhang, "Determining the Epipolar Geometry and its Uncertainty: A review," 1996.

[6] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2003. Part II: Chapter 9.

[7] C. Olsson. http://www.maths.lth.se/matematiklth/personal/calle/datorseende13/notes/forelas6.pdf. Lund Institue of Technology. Computer Vision 2013. Lecture 6.

[8] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2011. Chapter 4.

[9] E. Salahat and M. Qasaimeh, "Recent Advances in Features Extraction and Description Algorithms: A Comprehensive Survey," 2017.

[10] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," 2004.

[11] H. BayTinne, T. TuytelaarsLuc and L. Van Gool, "SURF: Speeded Up Robust Features," 2006.

[12] H. BayTinne, A. Ess, T. TuytelaarsLuc and L. Van Gool, "Speeded Up Robust Features (SURF)," 2007.

[13] D. Mistry and A. Banerjee, "Comparison of Feature Detection and Matching Approaches: SIFT and SURF," 2017.

[14] J. van Opstal, A. Berthoz , *Multisensory Control of Movement: Representaion of the eye position in three dimensions* . Oxford University Press, 1993. Chapter 2.

[15] OpenCV 3.0.0-dev documentation. https://docs.opencv.org/3.0-beta/doc/tutorials/features2d/feature_flann_matcher/feature_flann_matcher.html#feature-flann-matcher. Accessed on January 2nd 2018.