TÉCNICO LISBOA

Determining the orientation of a RGB camera embedded on an artificial eye

My Subtitles

Mariana Ribeiro dos Reis do Vale Martins

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Advisor(s)/Supervisor(s): Prof./Dr. Alexandre J. M. Bernardino
Prof./Dr John van Opstal
Prof./Dr José A. R. dos Santos Vitor

Examination Committee

Chairperson: Prof./Dr. Lorem Ipsum
Supervisor: Prof./Dr. Alexandre J. M. Bernardino
Members of the Committee: Prof./Dr. Lorem Ipsum
Prof./Dr. Lorem Ipsum

October 2019

Acknowledgments

Abstract

Your abstract goes here.

Keywords: my keywords

Resumo

O teu resumo aqui...

Keywords: as tuas palavras chave

Contents

List of Tables

List of Figures

Nomenclature

Camera Model

$\mathbf{m}$	Digital image point (in pixels)
u, v	Digital image horizontal and vertical coordinates (in pixels)
$\mathbf{m}'$	Image plane point (in meters)
x', y'	Image plane horizontal and vertical coordinates (in meters)
$\mathbf{m}'_\mathbf{d}$	Image plane point (in meters) distorted
x'_d, y'_d	Image plane horizontal and vertical coordinates (in meters distorted)
$\mathbf{M}$	3D point (in meters)
X, Y, Z	3D point coordinates (in meters), Z is perpendicular to the image plane, X is horizontal and Y is vertical
K	Intrinsics Matrix
P	Projection Matrix
λ	Depth
f	Focal Length
s	Skew
c_x, c_y	Camera's principal point offset
s_x, s_y	Pixel scalings

Epipolar Geometry

E	Essential Matrix
F	Fundamental Matrix

Representation of the eye's orientation

R	Rotation Matrix
$\mathbf{t}$	Translation vector

Acronyms

FLANN Fast Library for Approximate Nearest Neighbour. 11

GRAT Gradient-based technique. vii, viii, 16, 23

IMU Inertial Measurement Unit. viii, xiii, 4–6, 25, 27

MBPE Minimiztion of the back projection error. viii, 24

OPPR Orthogonal Procrustes Problem. 11, 12

RANSAC RANdom SAmple Consensus. vii, 18

RGB Red, Green, Blue. 5, 12

SIFT Scale Invariant Feature Transform. viii, xiii, 10, 11, 31–33

SURF Speeded Up Robust Features. viii, xiii, 10, 11, 33

SVD Singular Value Decomposition. i, 16, 17

Glossary

gaussian blur The Gaussian blur is an image-blurring filter that uses a Gaussian function. It produces a smoother and less noisy image.. 11, 31

homogeneous coordinates In computer vision, an extra dimension is added to the coordinates of a point that makes perspective projection transformations easier to compute. For a point $[x\ y]^T$, any three numbers, $[a_1\ a_2\ a_3]^T$ for which $\frac{a_1}{a_3} = x$ and $\frac{a_2}{a_3} = y$ are homogenous coordinates. This three new coordinates are represented by "$\sim$", e.g. $\widetilde{a} = [a_1\ a_2\ a_3]^T$. vii, 9, 13

singular value decomposition Singular Value Decomposition (SVD) is the factorization of a matrix A into the product of three matrices $U\Sigma V^T$ where the columns of U and V are orthonormal and Σ is a diagonal matrix with positive real entries. 11

skew symmetric matrix A skew-symmetric matrix is a square matrix whose transpose equals its negative. These matrices can be used to represent cross products as matrix multiplications. vii, 12, 14, 16, 17

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) to estimate the eye's orientation. Although IMU units are good on acceleration and velocity measurements, they tend to have a significant position drift when determining the orientation, as seen in 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, 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. TO CHANGE

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

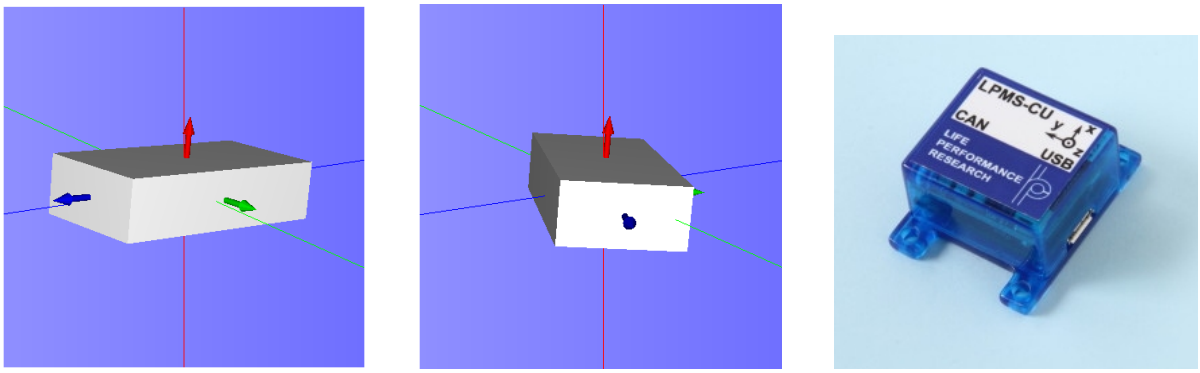

Figure 1.2: IMU sensor's drift. The two images on the left represent the IMU position on 3D space, between them it can be observed that, in 80 seconds, the IMU drifts about 60 degrees around the torsional axis. The coordinate system of the IMU is defined on the image on the right, where the axis in red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

orientation. 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 accuracy and computational time mainly depend on the complexity of detecting correspondences between consequent images and on the algorithm responsible for calculating the camera's orientation difference between those images. The latter being the focus of this thesis.

There is a long literature on what are the best algorithms to do so. However, the current prototype, seen on figure 1.2, has a particularity. There is a translation movement associated to the camera movement that may be defined as a function of the rotation and the fixed length of the camera's support. To that length, we call baseline and is observable on the left of figure 1.3, as is its effect on the right.

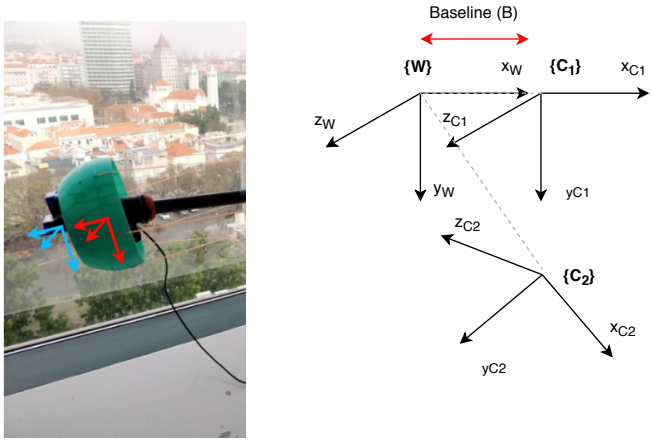

Figure 1.3: Camera's support baseline. On the left, it shows where the camera's center is, blue coordinate system, versus where the eye's rotation center is, red coordinate system, hence creating a baseline length.On the right, it shows how this length affects the rotation movement. TO CHANGE

Besides this, there is also another condition to the system. There is no depth information provided because the camera is only RGB. Hence, the problem is finding the best algorithm to estimate the orientation of a RGB camera with a baseline length associated to the rotational movement.

1.3 Objectives

As mentioned until now, the main objective of this thesis is to develop an algorithm to determine the orientation of the camera in its particular context that works as accurately and fast as possible. Beyond that, this work also aims to:

- Determine how much accuracy may be gained from using the baseline constraint versus the usual approach;
- Understand how different algorithms behave in the real world;
- And evaluate the improvement of using the camera over the IMU.

1.4 Outline

The next chapters of this document are structured as follows.

- Chapter 2 gives insight on essential concepts necessary to understand this work, along with an overview of the state of the art on the thesis' topic.
- Chapter 3 explains the methodology used to accomplish the objectives. It describes a novel algorithm to obtain the camera's orientation, how state of the art algorithms were implemented and how all of them were compared against each other.
- Chapter 4 presents the results gathered from comparing various estimation algorithms between each other, and with the IMU.
- Chapter 5 draws conclusions on which algorithm is best to use on the current prototype according to their behavior on the real world. Enlightens the reader on how this work attempts to contribute to science and on future work that shall be done.

Chapter 2

Background and State of the Art

2.1 Camera Model

To understand this work, it's important to first grasp the essential concepts on 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 helps at understanding the upcoming details. It's possible to establish a relationship between a real point in space and a point in an image in pixels, through the camera model as follows.

1. **Passing from the world frame to the camera reference frame**

   An arbitrary real point in space $M_W = [X_W \; Y_W \; Z_W]^T$ may be represented in the world reference frame, so it needs to be be converted into the camera reference frame, becoming M.

   This conversion consists on expressing the origin of the world reference frame on the camera frame, through a translation, $\mathbf{t}$, and a rotation, that can be represented by a matrix, R. These are called the **extrinsic parameters**.

$$M = [X \; Y \; Z]^T = R \, [X_W \; Y_W \; Z_W]^T + \mathbf{t} \tag{2.1}$$

2. **Project the 3D points into the image plane reference frame**

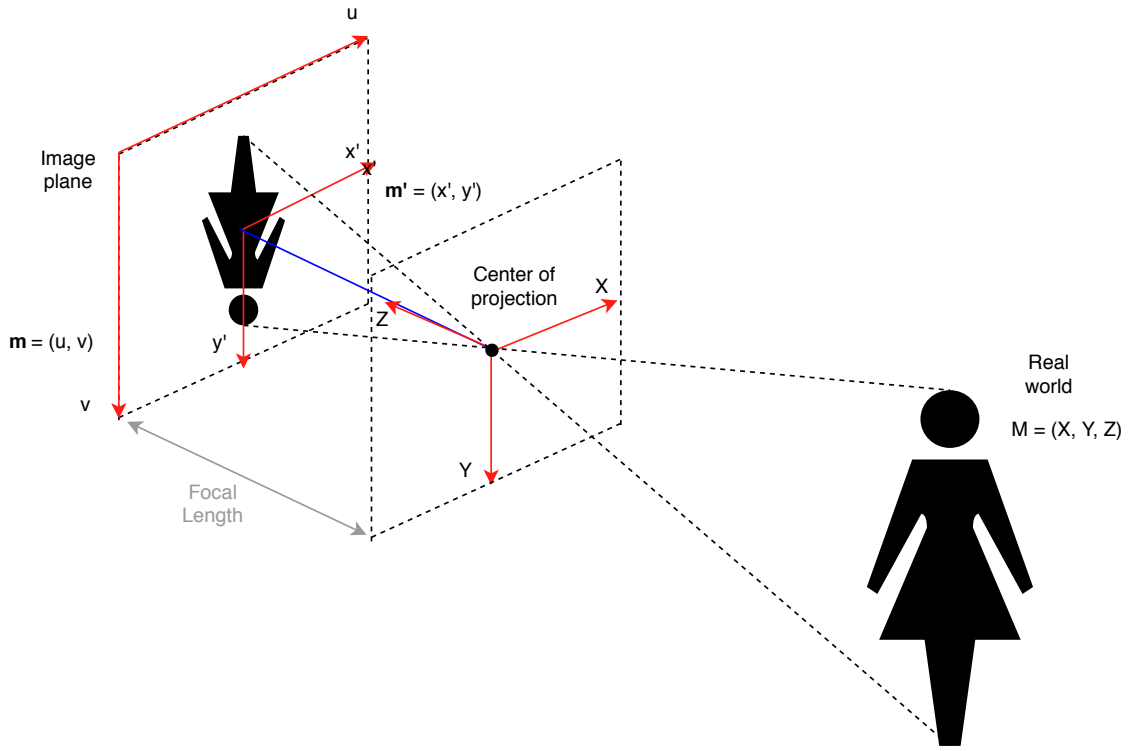

Figure 2.1: Pinhole Camera Model. The girl defined by the generic set of points M is projected onto the image plane (defined by the set of generic points $\mathbf{m_p}$) through the light rays passing through the camera's pinhole, called the center of projection. The focal length is the distance from 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.

As may be observed in the figure, a correspondence between the real world 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

$$x' = f\frac{X}{Z},\ y' = f\frac{Y}{Z}, \tag{2.2}$$

where f is the focal length in meters, and x', y', X, Y and Z are also represented in meters.

3. **Transform image plane points into pixel coordinates**

The digital image obtained is actually expressed in pixel units, whereas till now everything was described in metric units, requiring the points to be converted by scalings (s_x, s_y), as

$$u = s_x x',\ v = s_y y', \tag{2.3}$$

where (u, v) are the scaled image coordinates in pixels, denoted by $\mathbf{m} = [u\ v]^T$.

Moreover, the image plane center may not be coincide with the center of the digital image reference frame, as shown in Figure 2.1, thus the image points also have to be translated by $(c_x\ c_y)$ pixels, which is camera's principal point offset.

$$u = s_x x' + c_x,\ v = s_y y' + c_y. \tag{2.4}$$

Furthermore, due to sensor errors, there might be a deformation skew, s, associated to the camera

reference frame, which means that it may not have exactly perpendicular axes.

$$u = s_x x' + sy' + c_x,\ v = s_y y' + c_y. \tag{2.5}$$

Combining all the parameters, it yields the so-called **intrinsic parameters** matrix defined as,

$$K = \begin{bmatrix} fs_x & s & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.6}$$

Hence, using homogeneous coordinates, $\mathbf{m}$ may be expressed as,

$$\begin{array}{l} \widetilde{\mathbf{m}} = KR\frac{M}{Z} + \mathbf{t} \\ \lambda\widetilde{\mathbf{m}} = KRM + \mathbf{t} \end{array}, \tag{2.7}$$

where $\lambda = Z$. Or, more commonly,

$$\begin{array}{l} \lambda\widetilde{\mathbf{m}} \sim K[R \mid \mathbf{t}]\widetilde{M} \\ \lambda\widetilde{\mathbf{m}} \sim P\widetilde{M} \end{array}, \tag{2.8}$$

where P is a 3×4 matrix, named **camera matrix**.

Having said this, 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 (2.9). Radial distortion is modeled by the k_i coefficients on the left-hand side of the polynomial transformation in (2.9) (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]

$$\begin{array}{l} x'_d = x' \ \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + (2p_1x'y' + p_2(r^2+2x'^2)) \\ y'_d = y' \ \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + (p_1(r^2+2y'^2) + 2p_2x'y') \end{array}, \tag{2.9}$$

where $r^2 = x'^2 + y'^2$ and x'_d and y'_d, are the distorted coordinates of the image point, which should then be converted into digital image coordinates as shown previously.

2.2 Feature detection and matching

As explained before on section 1.2, it is necessary to gather corresponding features in consecutive images taken before and after a rotation, and to use an algorithm to estimate the orientation using those features. The latter, also referred to as keypoints or interest points, are spatial locations of an image that "stand out", allowing it to be identifiable. This points should be such that even after rotating, translating, shrinking or distorting the image, they can be found.

It is absolutely necessary to have a solid detection and trust-worthy correspondence of image features (matching) when doing an "eye" movement to eventually estimate the its current orientation. 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. The second method, which is more

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

suitable for points in the near space, independently detects features in both 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 features with another image 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. [4]

There exist a number of algorithms that can do the job. However, none of these algorithms is optimal for all 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 [5] (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 the two images.
- Quantity: enough features to describe the content;
- Accuracy: features should be 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 images on the overlapping regions;
- Invariance: deformative effects on the features, due to scaling, rotation or translation 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 Maximally stable extremal regions (MSER) and Scale Invariant Feature Transform (SIFT) algorithms enhance performance on computational complexity, accuracy and execution time. From the scale and rotation invariant algorithms, Speeded Up Robust Features (SURF), proved to be faster than SIFT, although not as robust.

MSER (by Matas J. et al in 2002)[2] 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.

SIFT (by Lowe, David G. et al in 2004) [6] algorithm detects image features using a Gaussian filter by generating increasingly blurred images, and subtracting each image to each other. This is done in several scales in order to provide scale invariance. Afterwards, a descriptor for each feature at a certain

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

scale is created and includes information about the orientation and gradient magnitude around the point, which also grants rotation invariance.

SURF (by Bay, Herbert et al in 1999) [7] uses an integral image, which is an intermediate representation of the original image where the value of a location is the sum of all pixels of a rectangular region formed around that location. This is called box filter and serves as an approximation to the gaussian blur, speeding up the process in relation to SIFT.

A more comprehensive explanation on both SIFT and SURF can be found at appendix A.1 and A.2, respectively. Any of these 3 algorithms mentioned seems promising to use as feature detector in this work.

2.2.1 Matching step

Regarding the matching of feature descriptors, one of the most common search algorithms used is the Nearest Neighbor Search. Fast Library for Approximate Nearest Neighbour (FLANN) (by M. Muja et al in 2009) [3] 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.

2.3 Orthogonal Procrustes Problem and Lack of Depth

Now concerning the algorithm used to estimate the orientation, one of the potential candidates is the Orthogonal Procrustes Problem (OPPR). The latter is able to find out the optimal linear transformation between two 3D clouds of points.

Procrustes, the 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. This is similar to the analysis held here, there are 3 elements to Procrustes's problem: M_2, the bed, M_1, the unlucky adventurer, and T, the fitting treatment. The solution to the problem is a matrix, T, that minimizes

$$\|M_2 - TM_1\|^2, \tag{2.10}$$

which transforms M_1 into M_2. M_1 and M_2 are clouds of 3D points obtained from the images took from the perspectives before and after a rotation, defined as R. Hence, the function to minimize is

$$\|M_2 - RM_1\|^2 \tag{2.11}$$

that through the Frobenius norm can be expanded as

$$\|M_2 - RM_1\|^2 = \operatorname{trace}(M_2^T M_2 + M_1^T M_1) - 2\operatorname{trace}(M_2^T R M_1). \tag{2.12}$$

This means that minimizing (2.10) with respect to R is equivalent to maximizing the second term of (2.12). By applying singular value decomposition (SVD) to $M_1 M_2^T$, the latter term can be further simplified into

$$\operatorname{trace}(M_2^T R M_1) = \operatorname{trace}(M_1 M_2^T R) = \operatorname{trace}(U\Sigma V^T R) = \operatorname{trace}(\Sigma V^T R U) = \operatorname{trace}(\Sigma H) = \sum_{i=1}^{N} \sigma_i h_{ii}. \tag{2.13}$$

[3] M. Muja and D. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration,", 2009.

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 $I = V^T RU$, obtaining $R = VU^T$ as the rotation of the eye.

However, as mentioned in section 1.2, the prototype has a translation associated to the eye movement, thus the rotation derived from OPPR is only an approximate estimation when ignoring this particularity.

When M_1 and M_2 are associated with different origins, it is customary to consider better fits of the configurations by allowing shape-preserving translations of the origin. When translating the points clouds by t_1 and t_2, respectively, (2.10) becomes

$$\left\|(M_2 - \mathbf{1}\mathbf{t_2}^T) - R(M_1 - \mathbf{1}\mathbf{t_1}^T)\right\|^2, \tag{2.14}$$

which can also be written as

$$\left\|M_2 - RM_1 - \mathbf{1}\mathbf{t}^T\right\|^2, \tag{2.15}$$

where $\mathbf{t}^T = \mathbf{t_2}^T - R\mathbf{t_1}^T$. The expression is minimized when $\mathbf{t}^T$ corresponds to the column-means of $M_2 - RM_1$, therefore a simple way of effecting this translation is to remove the column-means of M_1 and M_2, separately, before minimizing, which is called centering. The data then becomes expressed in terms of the mean deviations. [8] This would solve the issue with the translation component, if it weren't for the lack of depth information that makes centering inapplicable.

Nevertheless, in order to obtain the 3D point clouds from the RGB images, taken before and after rotation, it is necessary to have the depth λ that corresponds to the Z coordinate in the real world, as seen on section 2.1. The depth can be defined by projecting the 2D points in a sphere with enough radius compared to the focal length, as follows

$$\|(\lambda \mathbf{m_1'})\| = \|(\lambda x_1', \lambda y_1', \lambda)\| = r \tag{2.16}$$

where r is the radius of the sphere and λ then becomes

$$\lambda = \frac{r}{\sqrt{x'^2 + y'^2 + 1}}. \tag{2.17}$$

2.4 Epipolar Geometry

Epipolar geometry provides an alternative yet powerful tool to obtain image transformations using various different approaches. It describes the relation between two images, before and after a transformation, 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 under-constrained 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 real world, M, is projected as a point $\mathbf{m_1}$ in the first image, and point $\mathbf{m_2}$ in the second, then those points satisfy the epipolar relation

$$\tilde{\mathbf{m}}_\mathbf{2}^T F \tilde{\mathbf{m}}_\mathbf{1} = 0, \tag{2.18}$$

which can be easily deduced with projective geometry. In Figure 2.2, c_1 and c_2 are the centers of projection before and after the camera has been displaced by $(R, \mathbf{t})$. Given the point $\mathbf{m_1}$ on the first

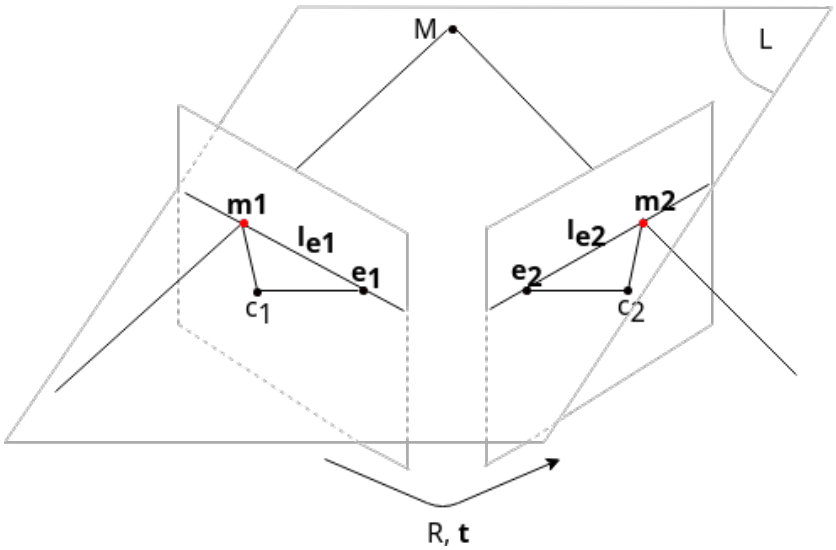

Figure 2.2: Epipolar geometry. The two small vertical planes correspond to the image planes after the rotation and translation of the camera, R and $\mathbf{t}$, respectively. c_1 and c_2 are the centers of projection before and after the transformation. M is the fixed point gazed at in the real world. $\mathbf{m_1}$ 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.

image, its correspondence, $\mathbf{m_2}$, in the second image is constrained to a line, the epipolar line, $\mathbf{l_{e2}}$. The latter is defined as the intersection of the plane L and the second image plane. Furthermore, plane L is delineated by the two centers of projection and $\mathbf{m_1}$. The epipoles, $\mathbf{e_1}$ and $\mathbf{e_2}$, are born from the intersection of the line $c_1 - c_2$ with the epipolar lines, $\mathbf{l_{e1}}$ and $\mathbf{l_{e2}}$.

2.4.1 Projective geometry concepts

Now, before one may understand the upcoming deductions, it is important to first understand the following concepts.

Representation of lines in homogeneous coordinates

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 $\mathbf{p} = (p_1, p_2)^T$ lies on a line $\tilde{\mathbf{l}} = (a, b, c)^T$, if and only if $ap_1 + bp_2 + c = 0$, which can be written as $(p_1, p_2, 1)(a, b, c)^T = \tilde{\mathbf{p}}^T\tilde{\mathbf{l}} = \tilde{\mathbf{l}}^T\tilde{\mathbf{p}} = 0$.

Line joining points

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

Cross product's skew symmetric matrix representation

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

$$[\mathbf{p}]_\times = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}. \tag{2.19}$$

2.4.2 Deducing the fundamental matrix algebraically

Having said this,

1. there is an homographic mapping via L plane from the points of the first to the second image, characterized by $\tilde{\mathbf{m_2}} = H_M \tilde{\mathbf{m_1}}$;
2. because $\mathbf{m_2}$ lies on $\tilde{\mathbf{l}}_{\mathbf{e_2}}$, then $\tilde{\mathbf{m}}_\mathbf{2}^T \tilde{\mathbf{l}}_{\mathbf{e_2}} = 0$;
3. and since $\mathbf{e_2}, \mathbf{m_2} \in \tilde{\mathbf{l}}_{\mathbf{e_2}}$, then $\tilde{\mathbf{l}}_{\mathbf{e_2}} = [\tilde{\mathbf{e}}_\mathbf{2}]_\times \tilde{\mathbf{m}}_\mathbf{2}$.

From 1) and 3),

$$\tilde{\mathbf{l}}_{\mathbf{e_2}} = [\tilde{\mathbf{e}}_\mathbf{2}]_\times H_M \tilde{\mathbf{m}}_\mathbf{1} \tag{2.20}$$

can be deduced and using conclusion 2) and (2.20),

$$\tilde{\mathbf{m}}_\mathbf{2}^T \tilde{\mathbf{l}}_{\mathbf{e_2}} = 0 = \tilde{\mathbf{m}}_\mathbf{2}^T [\tilde{\mathbf{e}}_\mathbf{2}]_\times H_M \tilde{\mathbf{m}}_\mathbf{1} = \tilde{\mathbf{m}}_\mathbf{2}^T F \tilde{\mathbf{m}}_\mathbf{1}, \tag{2.21}$$

where F, the fundamental matrix, 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 $det(F) = 0$.

2.4.3 Deducing the fundamental matrix from camera motion

Another way of obtaining this relation is through the camera motion. If the point in the real world is expressed in the eye's perspective before rotating, M_1, under the camera model studied in section 2.2, (2.22) and ((2.23) are true.

$$\lambda_1 \tilde{\mathbf{m}}_\mathbf{1} = K[I\ \mathbf{0}]\tilde{M}_1, \tag{2.22}$$

$$\lambda_2 \tilde{\mathbf{m}}_\mathbf{2} = K[R\ \mathbf{t}]\tilde{M}_1 \tag{2.23}$$

For now, for the sake of simplicity, the intrinsics matrix will be considered the identity, $K = I$, and the scale factors λ_1 and λ_2 will be dropped. Hence, by eliminating $\tilde{M}_1$, the previous equations become

$$\tilde{\mathbf{m}}_\mathbf{2} = R\tilde{\mathbf{m}}_\mathbf{1} + \mathbf{t}, \tag{2.24}$$

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

$$\tilde{\mathbf{m}}_\mathbf{2}^T \cdot (\tilde{\mathbf{m}}_\mathbf{2} \times \mathbf{t}) = 0 \tag{2.25}$$

$$\text{and } (\tilde{\mathbf{m}}_\mathbf{2}^T \cdot ((R\tilde{\mathbf{m}}_\mathbf{1} + \mathbf{t}) \times \mathbf{t}) = 0. \tag{2.26}$$

Therefore, the fundamental matrix, F, can be determined by

$$\begin{aligned} \tilde{\mathbf{m}_\mathbf{2}}^T [\mathbf{t}]_\times R \tilde{\mathbf{m}}_\mathbf{1} = 0 \\ \tilde{\mathbf{m}_\mathbf{2}}^T F \tilde{\mathbf{m}}_\mathbf{1} = 0. \end{aligned} \tag{2.27}$$

When the intrinsic parameters are not the identity matrix, then

$$\begin{aligned} \tilde{\mathbf{m}_2}^T K^{-T} E K^{-1} \tilde{\mathbf{m}_1} = 0 \\ \text{and finally } F = K^{-T} E K^{-1}, \end{aligned} \tag{2.28}$$

where E is the essential matrix.

2.4.4 Estimating the fundamental matrix

In such wise, the question now is how to determine F, and consequently the rotation, given two images with matching points. With the last-mentioned referred to as $\mathbf{m_1} = [u_1 \ v_1]^T$ and $\mathbf{m_2} = [u_2 \ v_2]^T$, the epipolar equation (2.18) can then be written as

$$\begin{aligned} & \begin{bmatrix} u_2 & v_2 & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \\ = & \begin{bmatrix} f_{11}u_1u_2 + f_{12}v_1u_2 + f_{13}u_2 + f_{21}u_1v_2 + f_{22}v_1v_2 + f_{23}v_2 + f_{31}u_1 + f_{32}v_1 + f_{33} \end{bmatrix} \\ = & \begin{bmatrix} u_1u_2 \ v_1u_2 \ u_2 \ u_1v_2 \ v_1v_2 \ v_2 \ u_1 \ v_1 \ 1 \end{bmatrix} \begin{bmatrix} f_{11} \ f_{12} \ f_{13} \ f_{21} \ f_{22} \ f_{23} \ f_{31} \ f_{32} \ f_{33} \end{bmatrix}^T \\ = & \ \mathbf{u} \cdot \mathbf{f} \\ = & \ 0 \end{aligned} \tag{2.29}$$

For n sets of points the expression becomes

$$Uf = 0, \tag{2.30}$$

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. Hence, 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 [9], 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 mentioned in the review paper: (i) a minimization of the epipolar error, (ii) a minimization of the re-projection errors (iii) and a gradient-based technique. The second one is the most time-consuming, thus not recommended. From the other two most promising approaches, the first seems to give the worst results and the last algorithm has been proposed to give the best results in the least amount of computational time.

Algorithm (i) minimizes the distances between points and their corresponding epipolar lines,

$$\min_{\mathbf{F}} \sum_i \left(d^2(\widetilde{\mathbf{m}}_{2i}, \widetilde{\mathbf{l}}_{e_2 i}) + d^2(\widetilde{\mathbf{m}}_{1i}, \widetilde{\mathbf{l}}_{e_1 i}) \right), \tag{2.31}$$

where $i = 1, ..., n$ with n being all the point matches and for a generic point in the first image

$$d(\widetilde{\mathbf{m}}_1, \widetilde{\mathbf{l}}_{e_1}) = \frac{\widetilde{\mathbf{m}}_1^T \widetilde{\mathbf{l}}_{e_1}}{\sqrt{l_{e_{1_x}}^2 + l_{e_{1_y}}^2}}. \tag{2.32}$$

Algorithm (ii) minimizes

$$\min_{\mathbf{F}} \sum_i \left(\|\widetilde{\mathbf{m}}_{1i} - \mathbf{h}_1(\mathbf{f}, M_i)\|^2 + \|\widetilde{\mathbf{m}}_{2i} - \mathbf{h}_2(\mathbf{f}, M_i)\|^2 \right), \tag{2.33}$$

where M_i with $i = 1, ..., n$ are the n points in space, while $\mathbf{h}_1(\mathbf{f}, M_i)$ and $\mathbf{h}_2(\mathbf{f}, M_i)$ are the functions that project the points in space into the first and second image plane, respectively, given the fundamental matrix represented as a vector by $\mathbf{f}$.

2.4.5 Gradient-based technique

Minimizing $\sum_i (\widetilde{\mathbf{m_i}}'^T F \widetilde{\mathbf{m_i}})^2$ doesn't yield a good result because the variance of each i term is not the same and the least-squares technique produces an optimal solution if each term has the same variance. So one possibility, given by algorithm (iii), is

$$\min_F \sum_i \frac{(\widetilde{\mathbf{m}}_{\mathbf{2i}}^T F \widetilde{\mathbf{m}}_{1i})^2}{\sigma_i^2}, \tag{2.34}$$

where $i = 1, ..., n$ and σ_i^2 is the variance given by

$$\sigma_i^2 = \sigma[l_{e1i_x}^2 + l_{e1i_y}^2 + l_{e2i_x}^2 + l_{e2i_y}^2]. \tag{2.35}$$

Because multiplying each term by a constant makes no difference, σ can be dropped. σ_i is what originates the name as it is the epipolar relation gradient.

2.4.6 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 (2.28). From here, it's possible to retrieve the camera matrix, $P = [R \mid \mathbf{t}]$, 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. Considering the matrices

$$\mathrm{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathrm{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.36}$$

where W is orthogonal and Z is a skew symmetric matrix, S may be written as $S = kUZU^T$[4], where U is orthogonal and $Z = \mathrm{diag}(1, 1, 0)W$, up to sign. Thus, $S = U\,\mathrm{diag}(1, 1, 0)WU^T$, up to scale, and

$$E = SR = U\,\mathrm{diag}(1, 1, 0)\left(WU^T R\right) = U\,\mathrm{diag}(1, 1, 0)V^T, \tag{2.37}$$

proving the initial statement.

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

$$E = U\,\mathrm{diag}(1, 1, 0)V^T, \tag{2.38}$$

[4]A proof of this is given in Result A4.1 of R. Hartley and A. Zisserman,Multiple View Geometry in Computer Vision [4]

or

$$E = U\,\mathrm{diag}(1,1,0)(-V)^T. \tag{2.39}$$

Considering (2.38), because

$$\begin{aligned} ZW &= \mathrm{diag}(1,1,0) \\ \text{and } ZW^T &= -\,\mathrm{diag}(1,1,0), \end{aligned} \tag{2.40}$$

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

$$S = -UZU^T, \quad R = UW^TV^T, \tag{2.41}$$

or

$$S = UZU^T, \quad R = UWV^T. \tag{2.42}$$

The matrix R on (2.41) is a rotation, since it is orthogonal,

$$R^TR = \left(UW^TV^T\right)^T UW^TV^T = VWU^TUW^TV^T = I, \tag{2.43}$$

and

$$\det\left(UW^TV^T\right) = \det(U)\det\left(W^T\right)\det\left(V^T\right) = \det(W)\det\left(UV^T\right) = 1, \tag{2.44}$$

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

$$-S^T = \left(UZU^T\right)^T = UZ^TU^T = -UZU^T = S. \tag{2.45}$$

The same applies to (2.42).[5] A proof that these are the only solutions is given in Result 9.18 of R. Hartley and A. Zisserman [4].

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,

$$ZU^T\mathbf{t} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T\mathbf{t} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} = 0, \tag{2.46}$$

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,

$$P = \left[UWV^T|+u_3\right] \quad \text{or} \quad \left[UWV^T|-u_3\right] \quad \text{or} \quad \left[UW^TV^T|+u_3\right] \quad \text{or} \quad \left[UW^TV^T|-u_3\right], \tag{2.47}$$

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

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. [4]

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 2.4.4;
2. Obtain the essential matrix, E, from the intrinsic parameters (which is possible in this case);
3. Retrieve the four possible solutions for $P = [R \mid \mathbf{t}]$ by doing a SVD decomposition on E;

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

4. Choose the feasible solution.

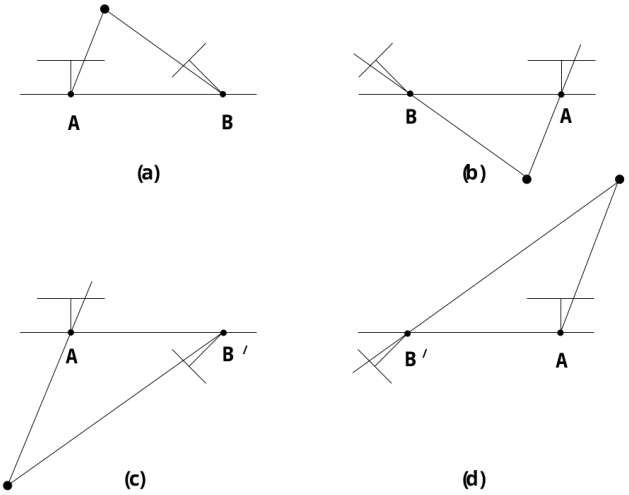

Figure 2.3: Four possible solutions retrieved from the essential matrix E. The points A and B correspond to the centers of projection before and after rotating. Between the right side, (a) and (c), and the left side, (b) and (d), figures, there is a translation direction inversion. Between top, (a) and (b), and bottom, (c) and (d), figures, there is a rotation of 180 degrees around the baseline. [4]

2.4.7 Pure rotation

As a disadvantage of epipolar geometry, if the eye's movement is not constrained by a translation component, the epipolar relation will not work, since $[\mathbf{t}]_\times$ would be a $3x3$ matrix of zeros and, consequently, $E = [\mathbf{t}]_\times R$ would yield the same, making it unfeasible to retrieve the rotation.

When doing feature detection and matching, wrongly matched pairs of points between the two images, 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, it is relevant to look into techniques of robust estimation.

2.5 Robust Estimation and Rejection of Image Sections

In computer vision, it is very common to estimate the parameters of a model from image data. Robust estimation eliminates non-gaussian noise from the data. Points that don't conform to a model are called outliers and are eliminated.

RANdom SAmple Consensus

One technique of robust estimation, is RANdom SAmple Consensus (RANSAC), first introduced by Martin A. Fischler and Robert C. Bolles in 1981 [11], where a small set of inliers is used to find the model and test all the other points against it. In this manner, it's possible to discover which points fit or not the model, and if they don't consider them outliers. The final model is the one that has more inliers.

However, it is necessary to have a way of defining the said model. In the case of this work, the model could be obtained through Orthogonal Procrustes Problem applied to the point matches. More specifically, 3 point matches would be enough to estimate the rotation between images that applied

to the remaining matches, could define which ones were outliers or inliers, leading to more matching accuracy.

Besides the accuracy, there is another interesting effect produced using this technique:

1. Points that are closer to the camera are the ones that are more affected by the baseline/translation component, mentioned in section 1.2. The ones further way are a closer fit to pure rotations.
2. OPPR derives pure rotations as explained in section 2.3.

Therefore by using RANSAC and OPPR together, points matches closer to the camera, are naturally be eliminated. This corresponds to image sections that would be more affected by translation and might prove beneficial. [10]

To summarize RANSAC works as following:

- selection of a small random sample, ”maybe inliers”;
- fit of a model to that sample;
- test of the model for the rest of the points matches. The ones that fit are ”also inliers”;
- if ”maybe inliers” + ”also inliers” are enough points for a model to be considered good, then this inlier point matches are potential outcome;
- reprat all the steps for as many tries as desired in order to gather the biggest number of inliers.

2.6 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):

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}, \; R_y(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}, \; R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.48}$$

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

$$R(\hat{\mathbf{n}}, \rho) = \begin{pmatrix} \cos\rho + n_1^2(1-\cos\rho) & n_1 n_2(1-\cos\rho) - n_3\sin\rho & n_1 n_3(1-\cos\rho) + n_2\sin\rho \\ n_1 n_2(1-\cos\rho) + n_3\sin\rho & \cos\rho + n_2^2(1-\cos\rho) & n_2 n_3(1-\cos\rho) + n_2\sin\rho \\ n_1 n_3(1-\cos\rho) - n_2\sin\rho & n_2 n_3(1-\cos\rho) + n_1\sin\rho & \cos\rho + n_3^2(1-\cos\rho) \end{pmatrix}. \tag{2.49}$$

2.6.1 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

$$R\mathbf{r} = \mathbf{r}. \tag{2.50}$$

Rewriting (2.50) as $(R - I)\mathbf{r} = 0$, it can be shown that

$$0 = R^T 0 + 0 = R^T(R - I)\mathbf{r} + (R - I)\mathbf{r} = \left(R^T R - R^T + R - I\right)\mathbf{r} = \left(I - R^T + R - I\right)\mathbf{r} = \left(R - R^T\right)\mathbf{r}. \tag{2.51}$$

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 (2.19)), and (2.51) then becomes a cross-product of vector $\mathbf{u}$ with itself,

$$\left(R - R^{\mathrm{T}}\right)\mathbf{r} = [\mathbf{r}]_\times \mathbf{r} = \mathbf{r} \times \mathbf{r} = 0, \tag{2.52}$$

obtaining

$$\mathbf{r} = \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix}, \tag{2.53}$$

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

$$\operatorname{Tr} R(\hat{\mathbf{n}}, \rho) = 1 + 2\cos\rho. \tag{2.54}$$

2.6.2 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

$$\mathbf{r} = \tan(\frac{\rho}{2})\hat{\mathbf{n}}, \tag{2.55}$$

that can be obtained through the previous rotation matrix (2.53) as

$$\begin{aligned} \cos\rho &= 1/2 \cdot (R_{11} + R_{22} + R_{33} - 1) \\ n_x &= \sin\rho\,(R_{32} - R_{23})\,/2 \\ n_y &= \sin\rho\,(R_{13} - R_{31})\,/2 \\ n_z &= \sin\rho\,(R_{21} - R_{12})\,/2. \end{aligned} \tag{2.56}$$

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

2.6.3 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,

$$q = \cos(\rho/2) + i\sin(\rho/2)\hat{n} \equiv q_0 + \mathbf{i} \cdot \mathbf{q}, \tag{2.57}$$

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 $\mathbf{r} = \mathbf{q}/q_0$. [**?**]

2.6.4 Coordinate system

The three-dimensional coordinate systems typically used in Neuroscience and Computer Vision use different conventions, as shown on Figure 2.4. For the remainder of this section, the neuroscience convention will be used to explain the next concepts.

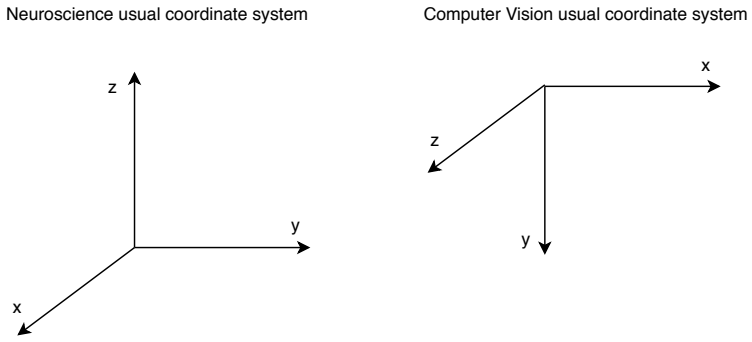

Figure 2.4: The neuroscience and computer vision 3 dimensional coordinate systems normally used are different; both use the right-hand rule. In Neuroscience, torsion is along the frontal visual x-axis, vertical angles along the inter-aural y-axis, and horizontal angles along the bottom-up z-axis.

2.6.5 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 2.6.2) 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 2.5 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.

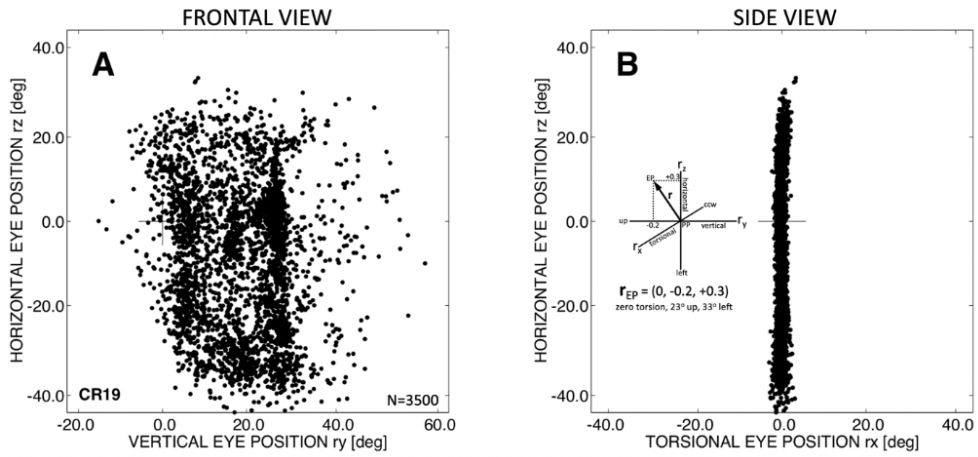

Figure 2.5: Listing's law, illustrated for 3500 eye orientations made by a head-restrained monkey, freely looking around in the laboratory room. Eye fixations are represented by the head-fixed Cartesian components of the 3D rotation axes, expressed in half-radians, and calculated as $angle = 2 \cdot atan(component)$. [3]

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

$$\omega_x = r_{1y} \cdot d_{12z} - r_{1z} \cdot d_{12y}. \tag{2.58}$$

This expression can be reached, using the representation on section 2.6.2, through the following:

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

$$\mathbf{r_{21}} = \frac{\mathbf{r_2} - \mathbf{r_1} + \mathbf{r_1} \times \mathbf{r_2}}{1 + \mathbf{r_1} \cdot \mathbf{r_2}} \approx \mathbf{r_2} - \mathbf{r_1} + \mathbf{r_1} \times \mathbf{r_2} \equiv \mathbf{d_{12}} + \mathbf{r_1} \times \mathbf{d_{12}}, \tag{2.59}$$

   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

$$\mathbf{r_{21}} = \begin{pmatrix} 0 \\ \mathbf{d_{12y}} \\ \mathbf{d_{12z}} \end{pmatrix} + \begin{pmatrix} 0 \\ \mathbf{r_{1y}} \\ \mathbf{r_{1z}} \end{pmatrix} \times \begin{pmatrix} 0 \\ \mathbf{d_{12y}} \\ \mathbf{d_{12z}} \end{pmatrix}, \tag{2.60}$$

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

[3] [**?**]

Chapter 3

Methodology

3.1 Approach

3.1.1 Feature detection and matching

3.1.2 Filtering matches

3.1.3 Baseline derivation

The transformation from the World to the Camera View 1 is

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

and the transformation from the World to the Camera View 2 is

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

Hence, the transformation from Camera View 1 to Camera View 2 is

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

Thus, the translation is a function of the rotation and the baseline on the following way

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

3.1.4 Gradient-based technique

SAY THAT EPIPOLAR IS GOOD CUZ NO NEED TO DETERMINE Z, reference PAMI paper

3.1.5 Minimiztion of the back projection error

This method is a bundle adjustment of the previous one. It uses the rotation matrix obtained with OPPr and it tries to tune it to obtain a rotation and a translation dependent on the former, through

$$\min_{R,Z_{e11},\dots,Z_{e1N}} \sum_{i=1}^{N} [(u_{e1i} - u_{1i})^2 + (u_{e2i} - u_{2i})^2 + (v_{e1i} - v_{1i})^2 + (v_{e2i} - v_{2i})^2]$$

with $Z_{e1init} = \frac{1}{\sqrt{u_{1i}^2 + v_{1i}^2 + 1}}$ and $R_{init} = R_{oppr}$,

where u_{1i} and v_{1i} are the image points of the Camera View 1, u_{2i} and v_{2i} are the image points of the Camera View 2, u_{e1i}, v_{e1i}, u_{e2i} and v_{e2i} are the corresponding image points estimations and Z_{e1i} is the depth of the Camera View 1.
The image point estimations are obtained the following way

$$\mathbf{m_{e1}} = \frac{KR^T(Z_{e2}K^{-1}\mathbf{m_2}) - R^T t)}{Z_{e1}}$$
$$\mathbf{m_{e2}} = \frac{KR(Z_{e1}K^{-1}\mathbf{m_1}) + t)}{Z_{e2}}.$$

The depth is initialized by projecting the image points in a sphere.

3.2 Implementation

3.2.1 Simulator

3.2.2 Real world

Chapter 4

Results and Discussion

4.1 Simulator

4.1.1 Data collection

4.1.2 Variable baseline

4.1.3 Variable depth

4.1.4 Variable noise

4.1.5 Orientation error

4.2 Real data

4.2.1 Data collection

4.2.2 Orientation error

Per axis using woodstand

In the eye prototype

4.2.3 Effect of baseline

4.2.4 Effect of RANSAC

4.2.5 Improvement over Inertial Measurement Unit

Chapter 5

Conclusion

Conclusion here...

5.1 Contributions

- Empirical study on the best method to estimate orientation with the current prototype's particular constraints
- Derivation of the baseline constraint
- An orientation estimation method with better precision than an IMU
- An open-source C++ library for similar contexts within the community
- A matlab simulator

5.2 Future Work

Bibliography

[1] J. Van Opstal. http://www.mbfys.ru.nl/~johnvo/OrientWeb/orient_1.html#Abstract. Accessed on October 9th, 2019.

[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," no. October, 2017.

[3] J. Van Opstal, "200 years Franciscus Cornelis Donders," *Strabismus*, vol. 26, no. 4, pp. 159–162, 2018.

[4] R. Hartley and A. Zisserman, "Multiview Geometry in Computer Vision," 2003.

[5] E. Salahat and M. Qasaimeh, "Recent advances in features extraction and description algorithms: A comprehensive survey," *Proceedings of the IEEE International Conference on Industrial Technology*, pp. 1059–1063, 2017.

[6] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, pp. 1–28, 2004.

[7] H. Bay, A. Ess, T. Tuytelaars, Gool, and L. Van, "Speeded-Up Robust Features (SURF) Herbert," *Revue du Praticien - Medecine Generale*, no. 465 SUPPL., pp. 44–45.

[8] D. G. Gower, J., "Procrustes Problems," *Technometrics*, vol. 47, no. 3, pp. 376–376, 2005.

[9] Z. Z., "Determining the Epipolar Geometry and its Uncertainty: A Review," *International Journal of Computer Vision*, vol. 27, no. 2, pp. 161–195, 1998.

[10] D. Burschka and E. Mair

[11] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," vol. 24, no. 6, 1981.

[12] D. Mistry and A. Banerjee, "Comparison of Feature Detection and Matching Approaches: SIFT and SURF," *GRD Journals- Global Research and Development Journal for Engineering*, vol. 2, no. March, pp. 7–13, 2017.

Appendix A

Feature Detectors

A.1 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 A.1. 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 and D is the difference.

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 A.2 shows. The points are only selected if they are larger than all of its neighbors or smaller than all of them.

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 origin, according to

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

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_g and orientation, θ, are

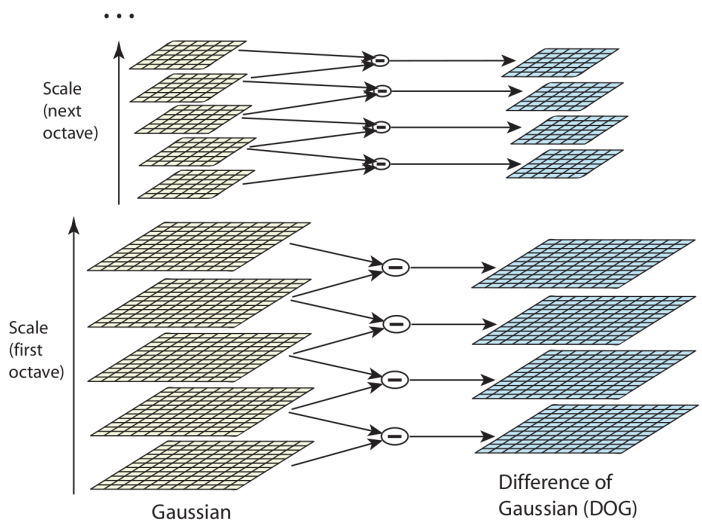

Figure A.1: Difference of Gaussians (DoG) by octave. To find points of interest (features), the Gaussian-blurred images at each scale (octaves) are subtracted pairwise. This process is computationally more efficient than computing the Laplacian of the Gaussians. [6]

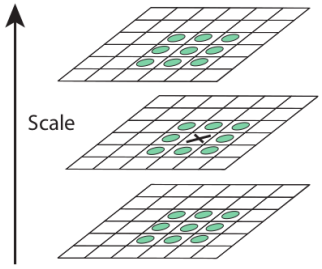

Figure A.2: Search for the maxima and minima of the DoG. After the DoG, each point is compared with its eight neigbours, and with the 9 corresponding neighbours of the images of the scale above and below. [6]

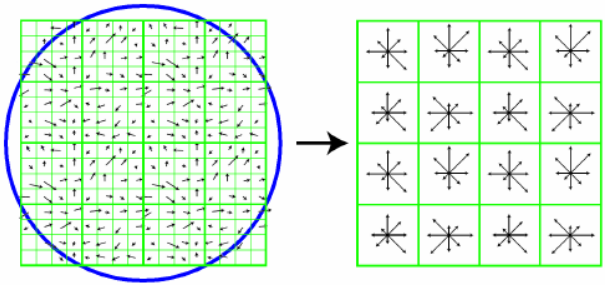

Figure A.3: SIFT feature descriptor. A window of 16×16 squares is centered around the point of interest. The blue circle represents the Gaussain window. In each 4×4 sub-region, a histogram of orientations is determined, represented on the image on the right. In total there are $8 \times 16 = 128$ values to describe the feature. [6]

determined as

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

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 A.3. 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. [6]

A.2 Speeded Up 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

$$\mathrm{I}_\Sigma(\mathbf{x}) = \sum_{\mathrm{i}=0}^{\mathrm{i}\leq \mathrm{u}} \sum_{\mathrm{j}=0}^{\mathrm{j}\leq \mathrm{v}} \mathrm{I}(\mathrm{i},\mathrm{j}), \tag{A.3}$$

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 A.4 makes a comparison 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}.

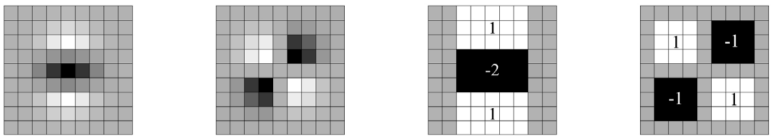

Figure A.4: Comparison between Gaussian second order derivatives (from the left, the first image in y-direction - L_{yy} - and second image in xy-direction - L_{xy}) and the corresponding images (from the left, the third image in y-direction - D_{yy} - and fourth image in xy-direction - D_{xy}) using a box filter. [7]

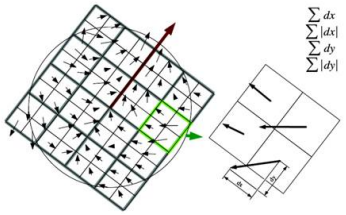

Figure A.5: SURF feature descriptor. The sampling window is rotated towards the dominant orientation. The vector $(\sum d_x, \sum d_y, \sum_x |d_x|, \sum |d_y|)$ is generated per each region through the Haar-wavelet responses in x and y direction. [7]

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

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}, \tag{A.4}$$

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

$$\det\left(\mathcal{H}_{approx}\right) = D_{xx} D_{yy} - \left(w D_{xy}\right)^2, \tag{A.5}$$

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 4×4 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 A.5 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 $\frac{\pi}{3}$. [7] [12]