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

Mariana Ribeiro dos Reis do Vale Martins
mariana.v.martins@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2019

Abstract

The brain is a highly complex system and even a simple task like looking at an object is still not fully understood. The eye has six extra-ocular muscles, but only needs two degrees of freedom to orient the fovea at any far position in the visual environment. How does the brain determine which muscles to contract? Bernstein theorized that it tries to find the optimal control solution, but which cost is minimized: energy, speed, accuracy? Building a robotic eye can prove valuable to identify the fundamental mechanisms behind these questions. A working mechanical model of a biologically inspired eye has been built in a previous project. An indispensable aspect of this prototype is determining its orientation with high accuracy, which is currently done by an IMU. However, this method suffers from significant inaccuracy, due to drift. Hence, we investigated whether adding a RGB camera to estimate the eye's 3D orientation could significantly improve accuracy and precision of the measurements. The camera position on the prototype is such that when rotating the eye, there is also a translation associated to the movement, which is a fixed function of the rotation. Despite the long literature on orientation estimation using a camera with naturalistic visual features, there is not much literature concerning this constraint. We compare two adapted state-of-the art algorithms, and present a novel algorithm that optimally makes use of this constraint. We validated the algorithms with a simulator and on the real eye prototype, showing promising results when compared to the IMU.
Keywords: camera orientation, rotation estimation, epipolar geometry, procrustes, back projection

1. Introduction

Even after years of research, the brain remains to this day a mysterious information-processing biological system. 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. 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. As it seems, the brain gradually tries to find the optimal control solution for a certain task, which was theorized by Nikolai Bernstein[1], yet it's unknown which cost is minimized: energy, speed or accuracy. The main topic of the ORIENT research project for the Lisbon team[2] 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 control and learning algorithms. This robot will hopefully contribute to a better understanding of the eye's control system.

So far, a working mechanical model of a biologically inspired eye with six muscles has been built in a previous work [1], shown in Figure 1. Currently, it is using an Inertial Measurement Unit (IMU) to estimate the eye's orientation. Although these units are good on acceleration and velocity measurements, they tend to have a significant position drift when determining the orientation. This work focuses on studying whether the addition of a camera to the system, could lead to a more precise and stable estimate for the orientation of the eye.

The eye model will eventually rely on accurate camera output, so to foster a solid neuroscientifically inspired study with this model, it's indispensable to have the best possible estimates of the camera's orientation, as well as to consider the computational speed versus accuracy trade off for real-time operations. The accuracy and computational time mainly depend on the complexity of detecting correspondences between consecutive images and on the algorithm responsible for calculating the camera's orientation difference between those images. There is a long literature on estimating the orientation of a camera from natural visual features. However, the current prototype, has a peculiarity. There is a trans-

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

[2] http://www.mbfys.ru.nl/ johnvo/OrientWeb/orient.html

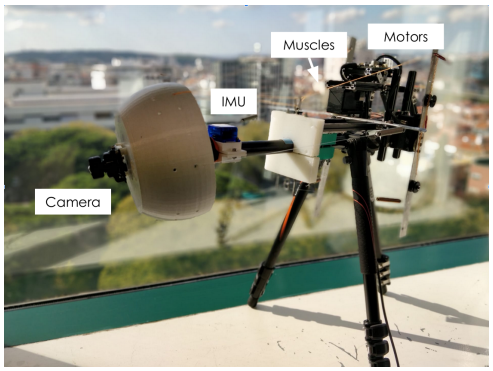

Figure 1: The current mechanical eye model. It is composed of an IMU, seen on the middle of the image, and connected to a white eye ball enclosure mounted on a spherical joint, in turn connected to six elastics, representing the eye muscles, and finally controlled by three motors that pull at those elastics (paired by the aluminum strips). The three motors are controlled by the computer. The embedded camera is seen on the left.

lation movement associated to the camera movement that may be defined as a function of the rotation and the length from the camera's center to the rotational center (spherical joint) that holds the camera. That length is referred to as baseline. Hence, the main goal is to identify the algorithm that does the job best under this constraint. Beyond that, this work also aims to (i) understand how much accuracy can be gained from using the constraint versus the classical unconstrained or rotation-only approaches, (ii) how the algorithms behave in the real world and (iii) evaluate the improvement of using the camera over the IMU.

2. Background and State of the Art

2.1. Representation of rotations

A 3D rotation can be represented in several ways, the most common are Euler angles and rotation matrices. Euler angles are three angles that define three rotations applied successively to three axes. Considering two 3D orthogonal right-handed coordinate systems, their relative orientation can be described by a 3×3 rotation matrix, R, which is parameterized by the Euler angles. There are many axes sequence conventions for Euler angles, in this work the convention will be ZYX, which corresponds to rotating around the torsional, vertical and horizontal axes. Considering a sequence of counter-clockwise rotations around Z axis by θ_Z, resulting in the coordinate system Z'Y'X', around Y axis by θ_Y, resulting in Z"Y"X" and around X axis by θ_X, resulting in Z"'Y"'X"'. The initial coordinate system is ZYX and the final coordinate system is Z"'Y"'X"', thus the rotation matrix can be defined as a multiplication (i.e. a serial order) of the following three rotation matrices, R_Z, R_Y and R_X, noting that the order by which they are multiplied matters (rotations are non-commutative).

$$R_z(\theta_Z) = \begin{bmatrix} \cos\theta_Z & -\sin\theta_Z & 0 \\ \sin\theta_Z & \cos\theta_Z & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (1)$$

$$R_y(\theta_Y) = \begin{bmatrix} \cos\theta_Y & 0 & \sin\theta_Y \\ 0 & 1 & 0 \\ -\sin\theta_Y & 0 & \cos\theta_Y \end{bmatrix}, \quad (2)$$

$$R_x(\theta_X) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_X & -\sin\theta_X \\ 0 & \sin\theta_X & \cos\theta_X \end{bmatrix}, \quad (3)$$

2.2. Camera Model

It's possible to establish a relationship between a real point in space, $M_W = [X_W\ Y_W\ Z_W]^T$, and a point in an image in pixels, $\mathbf{m} = [u\ v]^T$, through the camera model as

$$\lambda\widetilde{\mathbf{m}} \sim K[R \mid \mathbf{t}]\widetilde{M_W}, \quad (4)$$

where $\widetilde{\mathbf{m}}^T = [\mathbf{m}^T 1]$ and $\widetilde{\mathbf{M_W}}^T = [\mathbf{M_W}^T 1]$ are the, so called, homogeneous representations of $\mathbf{m}$ and $\mathbf{M_W}$, respectively. The rotation R and the translation $\mathbf{t}$ bring the point M_W, represented in the world reference frame to the camera's reference frame. λ is a scale factor proportional to the depth of point M in space. K are the intrinsic parameters (focal length, f, meter to pixel scalings, s_x and s_y, skew, s and frame translation, c_x and c_y) defined as

$$K = \begin{bmatrix} fs_x & s & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

that bring the point from the image plane to the digital image. (Figure 2).

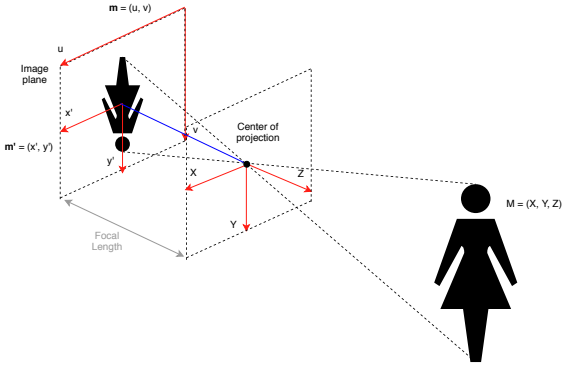

Figure 2: Pinhole Camera Model. The girl defined by M is projected onto the image plane defined by $\mathbf{m}$ through the light rays passing by the camera's pinhole. The focal length, f, is the length of the blue line. The digital image's coordinate system (u, v) is defined on top right corner of the image plane, whereas the image plane is defined on the center.

2.3. Orthogonal Procrustes Problem

To estimate the difference in orientation, R, of a camera between two time instances t_1 and t_2 , there are several algorithms available in the literature. If the camera undergoes a pure rotation, a possible algorithm is using the solution to the Orthogonal Procrustes Problem (OPPR). This algorithm finds the optimal linear transformation between two 3D clouds of points, by minimizing the cost

$$\|M_2 - RM_1\|^2, \tag{6}$$

where M_1 into M_2 are the 3D coordinates of points in the environment measured in the reference frame of the camera at time t_1 and t_2, respectively. This problem can be solved by applying Singular Value Decomposition (SVD) to $M_1 M_2^T$, obtaining $R = VU^T$. [2]

However, in order to reconstruct the 3D point clouds from the images, it is necessary to have the depth of the point, λ, as seen on section 2.2, but this information is unavailable when using a single camera with unknown motion. Assuming pure rotation, any depth provides the same information, so it's possible to define an artificial depth by projecting the 2D points in a sphere with sufficient radius compared to the focal length, as follows

$$\|(\lambda \mathbf{m_1'})\| = \|(\lambda x_1', \lambda y_1', \lambda)\| = r \tag{7}$$

where $m_1' = [x_1' \; y_1']$ is a point on the image plane before rotating the camera, r is the radius of the sphere and λ then becomes

$$\lambda = \frac{r}{\sqrt{x_1'^2 + y_1'^2 + 1}}. \tag{8}$$

Because OPPR only works for pure rotation, the translation associated (baseline constraint) is simply ignored.

2.4. Epipolar Geometry

Epipolar geometry 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 intrinsic parameters, K are known, or the fundamental matrix, F, otherwise. It is expressed as

$$\tilde{\mathbf{m}}_\mathbf{2}^T F \tilde{\mathbf{m}}_\mathbf{1} = 0, \tag{9}$$

where m_1 and m_2 are pixel points in the image before and after rotating, respectively [3]. From the fundamental matrix, F, it's possible to extract the rotation R and the translation $\mathbf{t}$, through a somewhat complex factorization that yields

$$F = K^{-T}[\mathbf{t}]_\times R K^{-1}, \tag{10}$$

where the essential matrix is $E = [\mathbf{t}]_\times R$. The fundamental matrix itself can be estimated through several methods. In Zhang's 1996 review on the issue [4], 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. Three nonlinear algorithms are mentioned in the review paper: (i) determines the maximum likelihood estimation of the fundamental matrix, (ii) minimizes the symmetric epipolar error, and (iii) minimizes the first order geometric error. The first one is the most time-consuming. From the other two promising approaches, the second 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) finds the maximum likelihood estimation of F by minimizing the distances between the points in the image and its re-projections. Having M_1 as a point in space, the corresponding points in the images before and after rotating are $\widetilde{\mathbf{m}}_1$ and $\widetilde{\mathbf{m}}_2$, respectively, obtained through the camera model. The re-projections of $\widetilde{\mathbf{m}}_1$ and $\widetilde{\mathbf{m}}_2$ are given by the projection functions $h_1(M_1, \mathbf{f})$ and $h_2(M_1, \mathbf{f})$, respectively, explained in Section 5.5 of Zhang 1996 [4], where $\mathbf{f}$ is the vectorized fundamental matrix, $[f_{11}\ f_{12}\ f_{13}\ f_{21}\ f_{22}\ f_{23}\ f_{31}\ f_{32}\ f_{33}]^T$.

Algorithm (ii) minimizes the epipolar error symmetrically, which is the distance from a point to its epipolar line[3].The epipolar line of the first image is defined as $\widetilde{\mathbf{l}}_{e_1} = F\widetilde{\mathbf{m}}_2$ and the second's as $\widetilde{\mathbf{l}}_{e_2} = F\widetilde{\mathbf{m}}_1$, the quantity to minimize is given by

$$\min_{\mathbf{F}} \sum_i \left(d^2\left(\mathbf{m}_{2i}, \mathbf{l}_{e_2}\right) + d^2\left(\mathbf{m}_{1i}, \mathbf{l}_{e_1}\right)\right), \tag{11}$$

where $d(\mathbf{p}, \mathbf{l}) = \frac{ap_1 + b_2 + c}{\sqrt{a^2+b^2}}$, with $\widetilde{\mathbf{l}} = (a, b, c)^T$ and $\mathbf{p} = (p_1, p_2)^T$, is the distance of a point to a line.

Minimizing the epipolar relation $\sum_i (\widetilde{\mathbf{m_i}}'^T F \widetilde{\mathbf{m_i}})^2$ 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 is to determine the fundamental matrix the way it's given by algorithm (iii) that minimizes:

$$\min_F \sum_i \frac{(\widetilde{\mathbf{m}}_{\mathbf{2i}}^T F \widetilde{\mathbf{m}}_{1i})^2}{\sigma_i^2}, \tag{12}$$

where σ_i^2 is the variance given by

$$\sigma_i^2 = \sigma[l_{e1i_x}^2 + l_{e1i_y}^2 + l_{e2i_x}^2 + l_{e2i_y}^2]. \tag{13}$$

This corresponds to minimizing the derivative of the epipolar relation, thus is also called Gradient-Based Technique (GRAT), or the Sampson error. Because multiplying each term by a constant makes no difference, σ can be dropped.

[3] See section 2.2 of Zhang 1996. [4]

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

2.5. Robust Estimation

Wrongly matched pairs of points between the two images, or points with large location errors, can severely affect the precision of the rotation estimation. 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.

In computer vision, it is very common to estimate the parameters of a model from image data. Robust estimation eliminates noise from the data. Points that dont conform to a model are called outliers and are eliminated.

One technique of robust estimation, is RANdom SAmple Consensus (RANSAC), first introduced by Martin A. Fischler and Robert C. Bolles in 1981 [5], where a small set of inliers is used to find a model and test all the other points against it. In this manner, it's possible to discover which points fit the model or not, and if they dont consider them outliers. The final model is the one that has the most inliers.

However, it is necessary to first define the said model. In the present study, the model could be obtained with the OPPR applied to the point matches. More specifically, 3 point matches would suffice to estimate the rotation between images that, applied to the remaining matches, could define which ones are 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 constraint. 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 eliminated. This corresponds to image sections that would be more affected by translation and might prove beneficial to eliminate them. [6]

3. Methods

Three methods will be used to estimate the orientation and compared against each other, OPPR, MBPE and GRAT. The first was described in 2.3, MBPE will be presented in this section and as for GRAT some alterations to what was explained on section 2.4 will be presented. As well as that, the algorithms GRAT and MBPE will use the fact that, in the particular geometry of this system the translation will be derived from the baseline constraint.

3.1. Translation derivation

Because the current eye prototype is a coupled system, the translation can be obtained through the knowledge of the rotation and the baseline length associated. This length is defined as the distance from the center of the camera's sensor to the center of rotation, as shown on Figure 3.

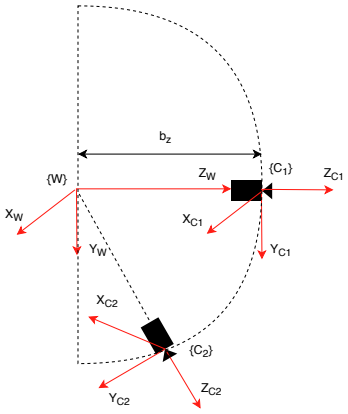

Figure 3: The rotation and the baseline length define the translation. W is the world reference frame centered on the rotation center, C_1 is the reference frame of the camera in the first position and C_2 is its frame on the second position, after rotating. b_z is the baseline length on the Z (torsional) axis.

The translation can be derived from using frame to frame transformations[4]. Having the world reference frame, W, set at the center of rotation, the transformation from the world to the first position of the camera, C_1, is

$$ {}^{C_1}T_W = \begin{bmatrix} I & -\mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{14} $$

where $\mathbf{b}$ is the baseline length expressed in each axis as $\mathbf{b} = [b_X \; b_Y \; b_Z]$. The transformation from the world reference frame to the second position of the camera, C_2, is

$$ {}^{C_2}T_W = \begin{bmatrix} R & -\mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{15} $$

where R is the rotation. Hence, the transformation from the first position of the camera to the second can be obtained as

$$ {}^{C_2}T_{C_1} = {}^{C_2}T_W \; {}^{W}T_{C_1} = \begin{bmatrix} R & -\mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} I & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R & R\mathbf{b} - \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{16} $$

[4]Consult Chapter 2 of Introduction to ROBOTICS mechanics and control [7] on Spatial descriptions and transformations

And finally, the translation ends up as

$$\mathbf{t}(R, \mathbf{b}) = R\mathbf{b} - \mathbf{b} = (R - I)\mathbf{b}. \quad (17)$$

3.2. Minimization of the backprojection error (MBPE)

This algorithm tries to minimize the error between the real image points, $\tilde{\mathbf{m}}_\mathbf{1}$ and $\tilde{\mathbf{m}}_\mathbf{2}$, and its back projections, $\tilde{\mathbf{m}}^\mathbf{r}_\mathbf{1}$ and $\tilde{\mathbf{m}}^\mathbf{r}_\mathbf{2}$, estimating the variables R and $\mathbf{t}$ directly instead of using the fundamental matrix, F, as an intermediate like it was described on section 2.4. Three parameters are enough to define the rotation (see section 2.1) and consequently the translation, $\mathbf{t}(R, \mathbf{b})$, as is dependent.

The back projections of the points can be obtained by

$$\lambda_2 \tilde{\mathbf{m}}^\mathbf{r}_\mathbf{2} = K[R\ \mathbf{t}]K^{-1}\lambda_1 \tilde{\mathbf{m}}_\mathbf{1} \quad (18)$$

$$\lambda_1 \tilde{\mathbf{m}}^\mathbf{r}_\mathbf{1} = K[R^T\ -R^T\mathbf{t}]K^{-1}\lambda_2 \tilde{\mathbf{m}}_\mathbf{2}, \quad (19)$$

where K are the known intrinsic parameters of the camera and λ_1 and λ_2, the depth of the corresponding 3D points, is unknown. Therefore, it has to be estimated for each point, making a total of $3 + n$ variables to estimate in the minimization. The cost function to this algorithm is then

$$\min_{\theta, \lambda^r_{11}, \ldots, \lambda^r_{1n}} \sum_{i=1}^{n} [(u^r_{1i} - u_{1i})^2 + (v^r_{1i} - v_{1i})^2 + (u^r_{2i} - u_{2i})^2 + (v^r_{2i} - v_{2i})^2] \quad (20)$$

where $[u_{1i}\ v_{1i}] = \mathbf{m_{1i}}$ are the pixel points in the image before rotating, $[u_{2i}\ v_{2i}] = \mathbf{m_{2i}}$ are the pixel points after rotating, $[u^r_{1i}\ v^r_{1i}] = \mathbf{m^r_{1i}}$ and $[u^r_{2i}\ v^r_{2i}] = \mathbf{m^r_{2i}}$ are its corresponding back projections, $\lambda^r_{11}, ..., \lambda^r_{1n}$ are the estimated depths and $\theta = [\theta_Z\ \theta_Y\ \theta_X]$ are Euler angles used to determine the rotation.

The back projections should be obtained by

$$\tilde{\mathbf{m}}^\mathbf{r}_\mathbf{1} = \frac{KR(\theta)^T(\lambda^r_2 K^{-1}\tilde{\mathbf{m}}_\mathbf{2}) - R(\theta)^T\mathbf{t}(R(\theta), \mathbf{b}))}{\lambda^r_1}$$

$$\tilde{\mathbf{m}}^\mathbf{r}_\mathbf{2} = \frac{KR(\theta)(\lambda^r_1 K^{-1}\tilde{\mathbf{m}}_\mathbf{1}) + \mathbf{t}(R(\theta), \mathbf{b}))}{\lambda^r_2}.$$

In order to start running the algorithm it's necessary to give it initialization parameters, λ^r_1 and λ^r_2 can be acquired from the projection of the image points in a 3D sphere, explained in section 2.3, and the Euler angles, θ, can be given by OPPR as it's a fast and light-weight approximation.

3.3. Gradient-Based Tecnhique (GRAT)

Regarding what was said on section 2.4, instead of obtaining R and $\mathbf{t}$ through the factorization of the fundamental matrix, F, they can be estimated rather than estimating F. Furthermore, given that $\mathbf{t}$ depends on R, like before only 3 parameters have to be estimated. This constraint can be used in combination with the most promising epipolar method explained on section 2.4, GRAT. A similar approach is taken by Vasconcelos F. et al [8] to calibrate a camera using multiple sets of pairwise correspondences, in section "8.2 Bundle Adjustment". The cost function for this algorithm is then

$$\min_\theta \sum_i \frac{(\tilde{\mathbf{m}}^T_\mathbf{2i} F \tilde{\mathbf{m}}_{1i})^2}{\sigma_i^2} \text{ with} \quad (21)$$

$$F = K^{-T}[\mathbf{t}]_\times R(\theta) K^{-1} \text{ and} \quad (22)$$

$$\sigma_i^2 = [l^2_{e1i_x} + l^2_{e1i_y} + l^2_{e2i_x} + l^2_{e2i_y}], \quad (23)$$

where σ_i^2 is the variance, $l_{e1i} = F\tilde{\mathbf{m}}_2$ and $l_{e2i} = F\tilde{\mathbf{m}}_1$ are the epipolar lines for each point i, θ are the Euler angles and $R(\theta)$ is a rotation matrix obtained through 1. Once again, the initialization of θ can be done using OPPR.

As opposed to MBPE, in GRAT the depths of the 3D points are not necessary. The explicit representation of 3D points is avoided by minimizing the perpendicular distances between point correspondences and their epipolar lines [8] [9].

4. Implementation

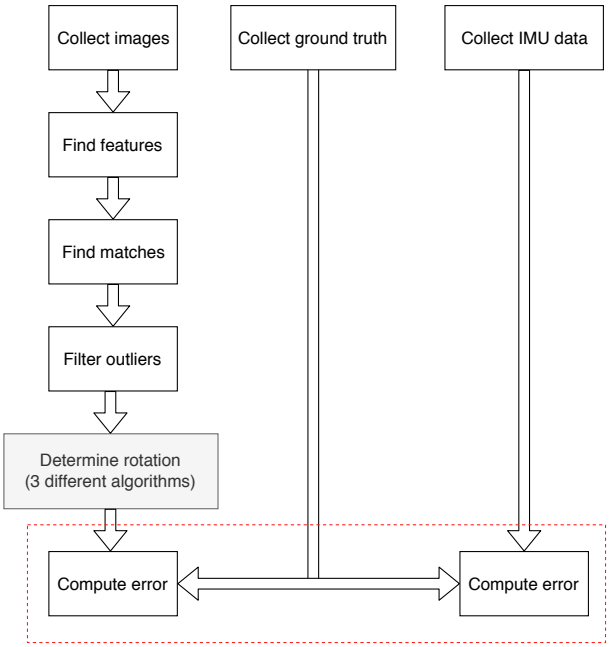

Figure 4: Flow Diagram. The RGB camera collects two images, before and after rotation. In each image, features are detected and matched between them. Some matches might be false or have too much noise, so they are filtered out. Using the matches information, the executed rotation is determined using an estimation algorithm. The three algorithms are tested. Finally, the estimated rotation is compared against the ground truth. The IMU's orientation output is also compared against the ground truth to evaluate its performance in relation to the camera.

Figure 4 shows the flow of the procedures that determine the eye rotation. It specifies what is needed to determine the rotation of the camera for real data sets. However, in order to evaluate the three algorithms in a controlled environment, a simulator was implemented which generates image point matches/correspondences with known rotations. These points may contain additive Gaussian noise, or even false matches to test the performance of the outlier filtering and the resilience of the estimator.

Both the simulator and the procedures to deal with real data were implemented in Matlab[5] and can be found in the thesis github repository[6]. A C++ library was also created to deal with the data on the real system for convenience and computational speed, and can also be found at the repository[7].

4.1. Real system Procedures

4.1.1 Eye prototype setup

Figure 5 shows the current eye prototype The distance from the center of rotation to the camera, defined previously as baseline length, is $[0\ 0\ 53.7] \pm 3\ mm$. This setup was fixed $5\ m$ away from furthest object in the scene.

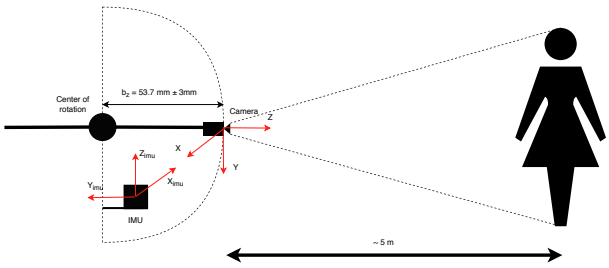

Figure 5: Eye prototype scheme used on the experiments. The camera is displaced from the center of rotation by b_z

4.1.2 Collect images

The camera was calibrated and the images were collected using an uEye LE USB3 camera in grayscale before and after a certain rotation.

4.1.3 Collect ground truth

Using the same camera in the same positions, two images were taken with a chessboard on the scene, which is used to determine the ground truth through its regular pattern. This images are then loaded into a Matlab algorithm[8] that determines the ground truth. It's important to note that there is error associated to the ground truth that can heavily affect the outcome of the experiments.

4.1.4 Feature detection and matching

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 them to be identifiable. These points should be such that even after rotating, translating, shrinking or distorting the image, they can be found. There exist a multitude of algorithms that can do the job. Salahat E. and Qasaimeh M. in 2017 [10], presents an overview on the recent algorithms, comparing them in terms of distinctiveness, locality, quantity, accuracy, efficiency, repeatability, invariance and robustness. 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. Because SURF is faster and more accessible due to patenting concerns regarding SIFT, it will be be used on this work.

Regarding the matching of feature descriptors, one of the most common search algorithms used, that will also be used here, is the Nearest Neighbor Search. Fast Library for Approximate Nearest Neighbour (FLANN) [11] 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.

4.1.5 Filter outliers

As referred on section 2.5, robust estimation is essential to guarantee an accurate estimation by eliminating outliers that can interfere with the estimation. On that note, RANSAC with OPPR is used to filter the matches.

4.1.6 Determine rotation

To estimate the rotation, the three different methods OPPR, MBPE and GRAT were used. For the last two, it is necessary to initialize them with OPPR as mentioned on section 3, and for OPPR and MBPE it is necessary to project the points onto a sphere with a large enough radius. OPPR is easily implemented using SVD. However, the others are non-linear unconstrained minimization problems, thus a non-linear solver is necessary. For that, Matlab's `fminsearch`[9] was chosen. This function is an adaptation of the Nelder-Mead simplex algorithm, described by Lagarias et al. [12].

4.1.7 Compute error

The estimation error is calculated as

$$error = ||\theta_{gt} - \theta_{est}||, \tag{24}$$

[5] https://www.mathworks.com/products/matlab.html
[6] https://github.com/Mrrvm/Orient/tree/master/Matlab
[7] https://github.com/Mrrvm/Orient/tree/master/C++
[8] https://www.mathworks.com/help/vision/ref/detectcheckerboardpoints.html
[9] https://www.mathworks.com/help/matlab/ref/fminsearch.html

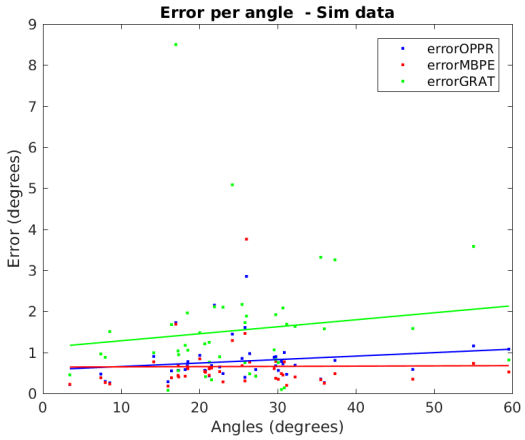

Figure 6: Experiment 1 - Error as function of amplitude (in degrees) under simulation for saccade amplitudes with $\sigma^2 = 15\,\text{deg}$

Method	Mean	Standard Deviation
OPPR	$0.77\,\text{deg}$	$0.49\,\text{deg}$
MBPE	$0.65\,\text{deg}$	$0.60\,\text{deg}$
GRAT	$1.53\,\text{deg}$	$1.43\,\text{deg}$

Table 1: Experiment 1 mean error and standard deviation.

where θ_{gt} are the ground truth Euler angles and θ_{est} are the Euler angles of the estimation.

5. Results

In this section, 6 experiments are presented. 5 of them are run under the simulator to test the estimation error according to several different aspects: the saccade amplitude, the noise, the depth to the camera and the effect of robust estimation. The last experiment is run under the real system. Finally, the computational times of the algorithms are presented.

5.1. Simulator

The baseline length of $53.7\ mm$ on the torsional axis was also used in the simulator to make the comparisons easier.

5.1.1 Experiment 1 - Saccade amplitudes generated with $\sigma^2 = 15\,\text{deg}$

Figure 6 shows the error per amplitude in degrees from 45 random saccades around all axis, simultaneously, generated in a normal distribution with variance $\sigma^2 = 15\,\text{deg}$. The lines on the graphs below represent the best fitting of the points. Table 1 presents the respective mean error and standard deviation.

5.1.2 Experiment 2 - Variable noise

To test how Gaussian noise affects the algorithms, all the same parameters as the previous section were kept and the amplitude of the saccades was set to a variance of $4\,\text{deg}$, while the added noise varies from 0 to $100\ px$. Figure 7 shows the error increasing with the pixel noise.

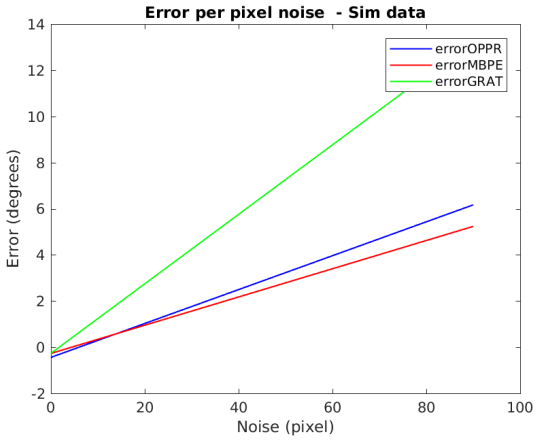

Figure 7: Experiment 2 - Error (in degrees) as function of the amount of noise in the simulated image (in pixels).

5.1.3 Experiment 3 - Variable distance

To see how the distance from the furthest object to the camera interferes with the estimation, a simulation with the following parameters was run, with a varying distance from $5\ cm$ to $10\ m$. Figure 8 exhibits the error in degrees according to the the distance to the furthest points from the camera.

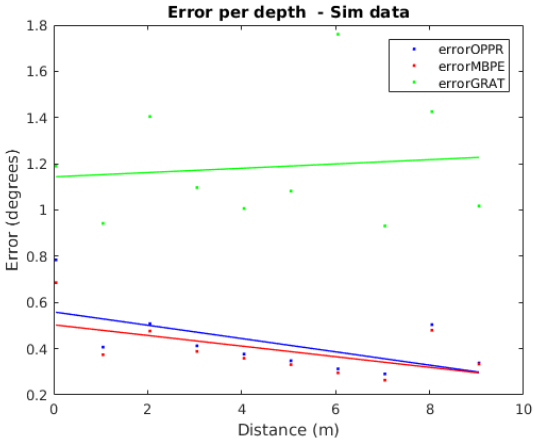

Figure 8: Experiment 3 - Error (in degrees) as function of distance to the camera in the simulated scene in meters.

5.1.4 Experiment 4 - Without robust estimation

The effect of rejecting image sections is easier to study under simulation without noise or false matches. The variance for saccade amplitude generation is now $\sigma^2 = 15\,\text{deg}$ to emphasize the effect, as bigger translations are executed. Figure 9 and Table 2 correspond to

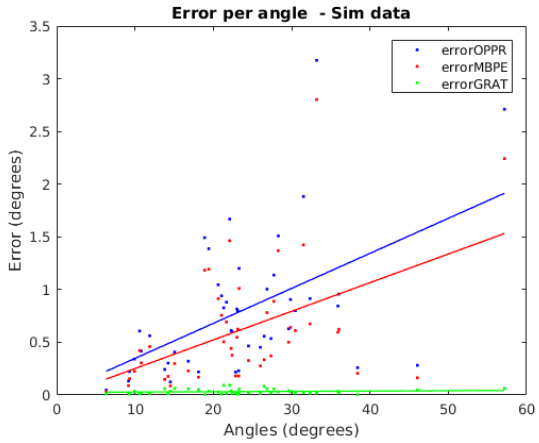

Figure 9: Experiment 4 - Error per saccade amplitude (in de- grees) under the simulation without robust estimation.

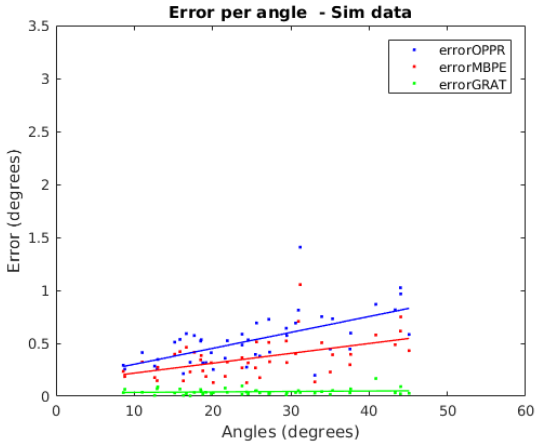

Figure 10: Experiment 5 - Error per saccade amplitude (in de- grees) under the simulation with robust estimation.

Method	Mean	Standard Deviation
OPPR	0.79 deg	0.63 deg
MBPE	0.61 deg	0.56 deg
GRAT	0.03 deg	0.02 deg

Table 2: Experiment 4 mean error and standard deviation.

running the algorithms without robust estimation. The amplitude of the saccades was set to a variance of 4 deg in the normal distribution.

5.1.5 Experiment 5 - With robust estimation

Figure 10 and Table 3 are the same as in Experiment 4 but with robust estimation.

5.2. Real system

To test the algorithms under the real system, 10 images of a scene, with the setup mentioned on 5, were took with and without a chessboard at the same time as the IMU collected the current orientation of the eye prototype, for each experiment. Joining the images/orientations with each other made up to 45 different saccades.

5.2.1 Experiment 6 - Rotation estimation error

Figure 11 presents the error per saccade amplitude on the real system in degrees for the camera, and Figure 12

Method	Mean	Standard Deviation
OPPR	0.53 deg	0.24 deg
MBPE	0.36 deg	0.18 deg
GRAT	0.04 deg	0.03 deg

Table 3: Experiment 5 mean error and standard deviation..

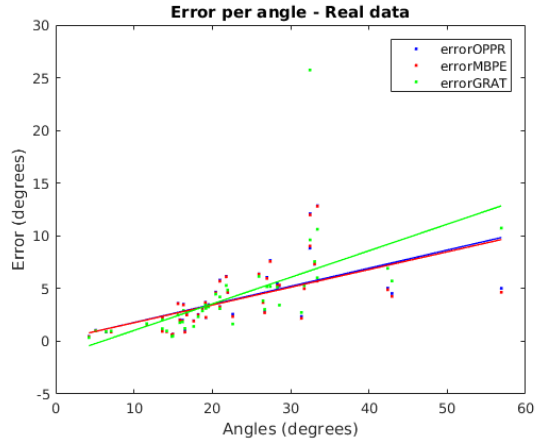

Figure 11: Experiment 6 - Error per saccade amplitude (in degrees) under the real system for the camera estimation.

shows the same for the IMU. Table 4 displays the corresponding mean error and standard deviation for each method and for the IMU. The lines on the graphs below represent the best fitting of the points.

5.2.2 Computational time

When using the created C++ library, the computational times seen on Table 5 were obtained for each of the processes that have to be executed to estimate the eye prototype's orientation and for the IMU estimation.

6. Discussion

In Table 1 and Figure 6 (Experiment 1), the best two methods are OPPR and MBPE. MBPE seems to be the best method in this case when run under simulation. As the saccade amplitude increases, OPPR becomes worst This may be due to the fact that for a bigger saccade, the associated translation creates a larger effect that is not counter acted when using this algorithm. As for

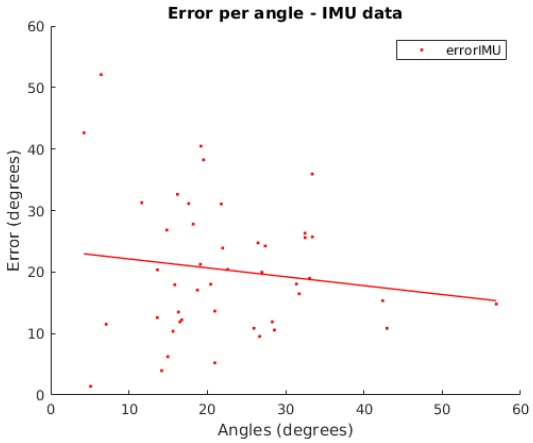

Figure 12: Experiment 6 - Error per saccade amplitude (in degrees) under the real system for the IMU estimation.

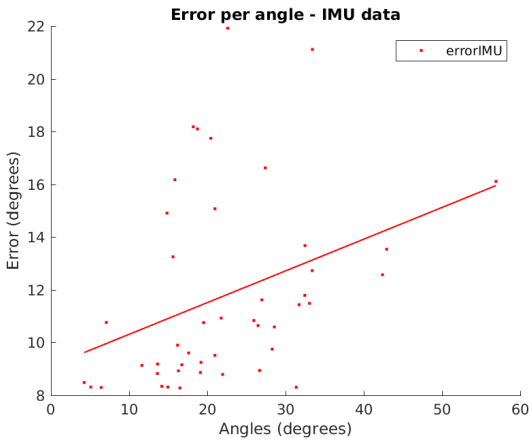

Figure 13: Experiment 6 but only visualizing the error on the horizontal and torsional axis.

Method	Mean	Standard Dev.
OPPR	3.90 deg	2.75 deg
MBPE	3.83 deg	2.75 deg
GRAT	4.13 deg	4.09 deg
IMU	20.34 deg	10.85 deg
IMU w/o vertical axis	11.79 deg	3.57 deg

Table 4: Experiment 6 mean error and standard deviation.

Process	Time
Image Capture	120 ms
Undistort + Find Keypoints	421 ms
Find Matches	16 ms
Robust Estimation	10 ms
OPPR	13e$-$2 ms
MBPE	212 ms
GRAT	51 ms
IMU Estimation	40e$-$4 ms

Table 5: Computational time for each of the necessary processes to estimate the orientation of the eye prototype in C++.

GRAT, it performs worst than the other algorithms, because the baseline length is small, the fundamental matrix expressed in (10), will have values very close to zero, that during intermediate calculus in the minimization can reduce the accuracy of the estimation. Other reason could be that, because the depth, λ, is not being estimated, opposite to MBPE, the leeway to estimate the rotation is reduced, ending up with a poorer guess.

In Experiment 2, it can be seen that GRAT is more sensitive to noise than the others.

The further away the points are, the more approximate to a pure rotation the movement becomes. By looking at Experiment 3, OPPR and MBPE, improve with distance, probably due to rejecting image sections. GRAT makes use of translation to get a good estimation, so it doesn't really gain anything from distance, except for a better initialization given by OPPR.

RANSAC+OPPR for robust estimation yield point matches that are more approximate to a pure rotation. By the results of Experiment 4, it's possible to see that it has slightly improved by using robust estimation for the methods OPPR and MBPE. For the former, this was very expected, as it is most accurate when dealing with pure rotations, as for MBPE by taking OPPR as an initialization parameter, it is normal to also have its results improved. GRAT, however, has a really small error now compared to the results in Experiment 1, which probably indicates that it is extremely sensitive to Gaussian noise in the images, that is not present here. This in fact makes sense, as this algorithm uses pixel coordinates directly to estimate the rotation.

Experiment 6 for the camera estimation (Figure 11), done under the real system, performed similarly to Experiment 1 as OPPR and MBPE keep being better than GRAT. Experiment 6's best mean error for the camera is 3.83 deg from MBPE with a standard deviation of 2.75 deg. This error is probably due to noise in the image or even false matches that were not filtered out in the robust estimation. As it is real data, there is always noise associated and imposing stricter parameters for RANSAC will eventually not yield enough point matches to work with. As for the IMU estimation results (Figure 12), it manifests a huge error in relation to the camera. Besides that, this error doesn't seem to have a pattern like the camera's, where the error increases with the saccade amplitude. This outcome may be caused by unstable measurements during the experiment, as the IMU needs to stabilize its position for some time until it gives a correct orientation. It could also be justified by the drift specially when associated to rotations around the vertical axis (as it cannot use the force of gravity for

the measurements). This last motivation can be backed up by Figure 13, that shows the same as Figure 12 without the rotation estimation around the vertical axis, presenting half the error as previously and an increasing pattern.

Regarding the computational error, obviously, the IMU takes much less time estimating the orientation than the camera. For the quickest algorithm, OPPR, the total amount of time taken is $567\ ms$.

Having said that, it's possible to make the following remarks summarizing each estimation method.

- OPPR is the fastest and the most light-weight of the camera estimation methods. Because it ignores the translation associated to the eye prototype rotations, it performs better when the movement is closer to pure rotations. If the scenery is very far way, the algorithm improves. Using RANSAC+OPPR to filter out point matches also refines the estimation.

- MBPE seemed to be the best camera algorithm out of all the Experiments. However, because it has to estimate both the three Euler angles and depth, λ, for each point match, a total of $(2 + n)$ parameters, it is the one that takes the longest. It improves when OPPR improves, as it highly depends on the initialization.

- GRAT did the worst in the Experiments and proved to be very sensitive to pixel noise in the image. It does better when associated to bigger saccade amplitudes, as they have more translation associated. It has a big downside, which is not working for pure rotations, as mentioned on section 3.3.

- IMU estimation is the fastest method but the error is much higher than in the camera algorithms, most probably due to rotation around the vertical axis.

The most accurate method is the Minimization of the back projection error (MBPE).

7. Conclusions

The contributions to science laid by this work were the following.

- Empirical study on the best method to estimate orientation with the current prototype's particular constraints.

- Derivation of the translation in function of the rotation and the baseline (section 3.1).

- An orientation estimation method with better precision than an IMU.

- An open-source C++ library for similar contexts within the community.

- A Matlab simulator for generating virtual images to experiment with.

At last, because there is always room for improvement, the subsequent future work is presented.

In order to determine eye's orientation in real-time more quickly, one could eventually study how using a Kalman filter[10] on the IMU and the camera together could improve the estimation in accuracy and time. The Kalman filter would keep track of the estimated state of the system, that would be gathered through the IMU, and the variance or uncertainty of the estimate. At some point, the estimate would be updated, by the camera using MBPE, to reduce the uncertainty.

References

[1] 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. (October), 2017.

[2] D. G. Gower, J. Procrustes Problems. *Technometrics*, 47(3):376–376, 2005. ISSN 0040-1706.

[3] R. Hartley and A. Zisserman. Multiview Geometry in Computer Vision. *Cambridge University Press New York*, 2003.

[4] Z. Z. Determining the Epipolar Geometry and its Uncertainty: A Review. *International Journal of Computer Vision*, 27(2):161–195, 1998. ISSN 09205691.

[5] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6), 1981.

[6] D. Burschka and E. Mair. Direct Pose Estimation with a Monocular Camera. 2017.

[7] J. Craig. Introduction to ROBOTICS mechanics and control, 2004. ISSN 0018-9286.

[8] F. Vasconcelos, J. P. Barreto, and E. Boyer. Automatic Camera Calibration Using Multiple Sets of Pairwise Correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):791–803, 2018. ISSN 01628828.

[9] J. K. Y. Ma, S. Soatto and S. Sastr. An invitation to 3-d vision: from images to geometric models. *Springer*, 2004.

[10] 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*, pages 1059–1063, 2017.

[11] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP International Conference on Computer Vision Theory and Applications*, 2009.

[12] C. Lagarias, Jeffrey, A. Reeds, James, H. Wright, Margaret, and E. Wright, Paul. Convergence properties of the neldermead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.

[10] As a starting point: Zarchan P. and Musoff H. Fundamentals of Kalman Filtering: A Practical Approach. American Institute of Aeronautics and Astronautics, Incorporated. 2000.