Visual-Inertial Odometry using Square-Root UKF on Lie groups to estimate orientation of monocular camera

Gonçalo Dinis Ramos Costa Pereira
goncalo.c.pereira@tecnico.ulisboa.pt
Instituto Superior Técnico, Lisboa, Portugal
November 2021

Abstract—The oculomotor system is responsible for directing and controlling eye movements in humans but also in other animals. Eye saccades are extremely fast and accurate movements and the 6 extra-ocular muscles are able to control the orientation of the eye in a two-dimensional manifold. This makes modelling and design of a bio-inspired robotic eye a very challenging problem to solve. One of the goals of the ORIENT project is to develop a prototype robotic eye that can mimic the human system. Estimating the 3D orientation of the mechanical prototype is crucial to develop control strategies that could help us further improve the design. In previous works, this estimation was done using an Inertial Measurement Unit (IMU) and a camera separately. However, the camera can only be used for low frequency estimation due to motion blur, and the IMU suffers from poor signal-to-noise ratio in slow movements. The main focus of this work is to develop an algorithm that fuses the information of these two sensors to estimate the orientation of the camera, using an approach very common in Visual-Inertial Odometry (VIO). We use the recently introduced Unscented Kalman Filter on Lie Groups methodology and compare to the ground-truth. We assess the performance of the filter in simulation and with a real world dataset, obtained using the Kinova Gen3 robotic arm. Finally, we discuss how this approach can be used in the robotic eye and suggest possible improvements for future work.

Keywords: Eye saccades, orientation estimation, Visual-Inertial Odometry, Unscented Kalman Filter

1. Introduction
The human eye has 6 extra-ocular muscles, 3 agonist-antagonist pairs and these muscles provide 3 degrees of freedom for rotation. Yet, to point the eye in any given direction (gaze direction), only two degrees of freedom are required. The eye is also capable of extreme fast and accurate movements, called saccades. This poses a very challenging and interesting problem for neuroscientists but also in the robotics field, because not only is the oculomotor system non-linear, but also, trying to develop a biomimetic model of the human eye with extra degrees of freedom brings additional constraints to the mathematical formulation of the problem [10]. That is why there is currently research being held in Lisbon in the context of the ORIENT project in order to design and test a humanoid eye-head robotic system that follows the same principles as human biology. There has already been quite alot of previous work done in this project, namely in the development of a (recent) biomimetic eye prototype that consists of six independent motors controlling the 6 cables that approximate the extra-ocular muscle geometry of the human eye. The team has also developed models for these prototypes [10, 8] and subsequent control techniques [1, 7] as well as work in the computer vision area [13].

1.1. Problem statement
The eye prototype uses a standard camera and an Inertial Measurement Unit (IMU) to estimate the eye’s orientation. In previous works [13], the orientation was estimated using these two sensors separately and from here the following conclusions were drawn: The camera can be used for low frequency motion estimation (10-20 Hz) but if the movements are too fast, there is motion blur problems. Eye saccades have peak velocities of up to 700 deg/s [10] and so, only when the eye slows down are we able to capture images with decent quality. On the other hand, the IMU is faster (100-200 Hz) but has the problem of drifting over time due to accumulation of integration errors and so can only be used for
short periods of time when estimating poses. These two types of sensors complement each other and have long been used to solve various problems in the field of robotics and navigation. By taking advantage of the complementary nature of the camera and IMU, it is possible to get a more accurate and solid estimate of the orientation of the eye system.

1.2. Objectives

The main objective of this work is to develop an algorithm that estimates the orientation of camera in the specific context of saccadic eye movements. To tackle this, the approach will involve fusing information of both camera and inertial sensors. With this in mind, the following steps aim to help solve this problem: analyse and study the different approaches/algorithms that have been used to perform sensor fusion; perform real world benchmarking that allows the comparison of different algorithms; test the algorithms with simulated data and real data and, finally, assess the performance in terms of accuracy and error.

2. State of the art

When it comes to existing literature, there are a lot of different ways to estimate pose by fusing information from visual and inertial sensors. In the field of robotics, the most common approaches involve SLAM (Simultaneous Localization and Mapping), SfM (Structure from Motion) and VIO (Visual-Inertial Odometry) \[2\] \[3\] \[17\] \[9\]. In the SLAM estimation process, besides computing the pose of the visual-inertial sensor, there is also estimation of feature positions (landmarks in 3D space) and map creation (through loop closure) for better accuracy \[4\]. SfM is very similar to SLAM in the sense that it usually involves a camera and other sensors (such as IMU or LIDAR) to create a map of the environment \[19\]. However, the focus of this problem is to create an accurate 3D structure of the environment and is solved offline, whereas SLAM focuses mostly on real-time estimation (online). Finally, VIO also estimates the position and velocity of an agent but it does not include the feature positions in the state. This results in a less accurate estimation when comparing to the two previous methods, but it is faster and more efficient.

As mentioned previously, for this project, it is required that the estimation process is accurate but also fast enough to allow the development and design and controllers for future work. The VIO approach is therefore, from the ones described above, the best option. From the existing VIO methods, there are three major paradigms: filtering, fixed-lag smoothing and full-smoothing \[17\]. Filtering algorithms only allow estimation of the latest state of the system. Classic approaches include the Extended Kalman Filter (EKF) \[23\] and the Unscented Kalman Filter (UKF) \[22\], which use the covariance matrix to represent uncertainty. Fixed-lag smoothers estimate states included within a certain time window and rule out older states. This approach is usually more accurate than filtering since they are robust to outliers (by the use of robust cost functions for example). However, fixed-lag smoothers are similar to filters when it comes to inconsistency and linearization errors. Lastly, full-smoothing approaches, estimate the entire history of states by solving a large non-linear optimization problem. This guarantees high accuracy since it can update based on the complete history, but has the drawback of being too computationally expensive due to the complex nature of the optimization problem. For our sensor fusion problem, we opted for a filtering method, namely the UKF, since the smoothing approaches become easily unfeasible due to their complexity and high computational cost.

3. Background & methods

The present work revolves around a recent variation of the UKF that aims to solve VIO using differential geometry, more specifically, Lie group theory (UKF-LG) \[3\] \[9\] \[5\]. That is because the quantities that we wish to estimate live on smooth manifolds, namely the orientation (rotation matrices). For more background on Lie groups and differential manifolds consult \[2\] \[15\]. In this chapter we lay down the fundamental notions of Riemannian manifolds: the Special Orthogonal Group, \(SO(3)\), and the Special Euclidean Group, \(SE(3)_{z+p}\), described in sections 3.1 and 3.2 respectively. From section 3.3 until section 3.9, we formulate our VIO estimation problem by describing the kinematics of our sensor system, the state dynamics that include the noisy IMU observations as way to propagate the state and the correction/update done with the observation of a certain number of landmarks through the camera. Finally, in sections 3.10 through 3.13 we go into more detail with regards to the UKF on Lie groups methodology.

3.1. Special Orthogonal Group \(SO(3)\)

The Special Orthogonal Group is defined as the set of 3D rotation matrices \(SO(3) := \{R \in \mathbb{R}^{3 \times 3} : R^T R = R R^T = I_3, \det(R) = 1\}\). This set of matrices, together with matrix multiplication as the group operation and the transpose as the inverse, becomes a matrix Lie Group.

The tangent space to the group \(SO(3)\) at the identity is denoted as \(so(3)\) and is also called the Lie algebra. It is defined as the space of \(3 \times \) skew symmetric matrices. Given a vector \(\omega \in \mathbb{R}^3\),

\[
\omega^\wedge = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \in so(3) \tag{1}
\]
where \((\cdot)^\wedge\) is the hat operator as in [20] [2] [9]. Conversely, there is the vee operator, \((\cdot)^\vee\) that maps a skew symmetric matrix into a vector in \(\mathbb{R}^3\) as \(\Omega = \omega^\wedge \Rightarrow \Omega^\vee = \omega\). The exponential map transforms an element of the Lie algebra, \(\mathfrak{so}(3)\), into a rotation matrix in \(SO(3)\) and corresponds to the Rodriguez’s formula (Figure 1). For convenience, the definition is re-written below:

\[
\exp(\omega^\wedge) = I_3 + \sin(\|\omega\|)\omega^\wedge + \frac{1 - \cos(\|\omega\|)}{\|\omega\|^2}(\omega^\wedge)^2
\]

(2)

On the other hand, the logarithmic map converts a rotation matrix back into a skew symmetric matrix and is defined as \(\log(R) = \theta[R - R^T]/2\sin(\theta)\), where \(\theta = \cos^{-1}\left(\frac{\text{tr}(R) - 1}{2}\right)\) is the rotation angle around a rotation axis, \(\hat{n}\). Letting \(\omega = \theta\hat{n}\), then \(\omega = \log(R)^\wedge\). By convention, \(\theta\) is chosen such that \(\|\omega\| < \pi\).

### 3.2. Special Euclidean Group \(SE(3)_{2+p}\)

In recent works [5] [6] [5] it has been explored the Lie group structure that appears naturally in the SLAM problem. This “new” Lie group structure was named \(SE(3)_{2+p}\), with \(p\) referring to the visual landmark measurements. It includes the matrices written in the form

\[
\chi \in \mathbb{R}^{(5+p)\times(5+p)} : \chi = \begin{bmatrix} R & v & o & p_1 & \ldots & p_p \\
0_{(2+p)\times 3} & I_{(2+p)} \end{bmatrix}
\]

(3)

with \(R \in SO(3)\) and \(v, o, p_1, \ldots, p_p \in \mathbb{R}^3\). Even though this structure is commonly used to solve the SLAM problem, for this work, we are not interested in creating a map of the environment or get a sense of its 3D structure (hence VIO). The inclusion of the landmark positions in the state has the purpose of trying to minimize error and getting a more accurate estimation. To compute the inverse of an element in \(SE(3)_{2+p}\), we have

\[
\chi^{-1} = \begin{bmatrix} R^T - R^T v - R^T o & -R^T (p_1 & \ldots & p_p) \\
0_{(2+p)\times 3} & I_{(2+p)} \end{bmatrix}
\]

(4)

And since \(R^T \in SO(3)\) and \(-R^T v, -R^T o, -R^T (p_1 & \ldots & p_p) \in \mathbb{R}^3\) the inverse element \(\chi^{-1}\) is also a member of \(SE(3)_{2+p}\). The Lie algebra of \(SE(3)_{2+p}\) is defined as

\[
\xi^\wedge = \begin{bmatrix} \xi^\wedge_R & \xi^\wedge_v & \xi^\wedge_o & \xi^\wedge_{p_1} & \ldots & \xi^\wedge_{p_p} \\
0_{(2+p)\times(5+p)} \end{bmatrix} \in \mathfrak{se}(3)_{2+p}
\]

(5)

with \(\xi^\wedge_R \in \mathfrak{so}(3)\) and \(\xi^\wedge_v, \xi^\wedge_o, \xi^\wedge_{p_1}, \ldots, \xi^\wedge_{p_p} \in \mathbb{R}^3\). Finally, the exponential map relates the elements of the Lie algebra to the matrix Lie group with the closed-form expression:

\[
\exp(\xi^\wedge) = I_{5+p} + \xi^\wedge + \left(1 - \cos\left(\frac{\|\xi^\wedge_R\|}{\|\xi^\wedge_R\|^2}\right)\right)(\xi^\wedge)^2
\]

\[
+ \left(\frac{\|\xi^\wedge_R\| - \sin\left(\frac{\|\xi^\wedge_R\|}{\|\xi^\wedge_R\|^3}\right)}{\|\xi^\wedge_R\|^3}\right)(\xi^\wedge)^3
\]

(6)

For the sake of simplicity of notation, the exponential and logarithmic maps are altered to operate in vectors instead of skew symmetric matrices. The example for the special orthogonal group is

\[
\begin{align*}
\text{Exp} : \mathbb{R}^3 & \rightarrow SO(3) ; \; \omega \mapsto \exp(\omega^\wedge) \\
\text{Exp}(\omega) & = \exp(\omega^\wedge) \\
\text{Log} : SO(3) & \rightarrow \mathbb{R}^3 ; \; R \mapsto \log(R)^\vee \\
\text{Log}(R) & = \log(R)^\vee
\end{align*}
\]

Figure 1: Representation of how the exponential mapping transforms points between the Lie group and the tangent space.

### 3.3. Gaussian distributions and uncertainty description

When working with Gaussian random variables living in a vector space, i.e., \(x \in \mathbb{R}^N\), they usually take the form

\[
x \sim \mathcal{N}(\bar{x}, \Sigma) \; \text{or} \; x = \bar{x} + \epsilon, \; \epsilon \sim \mathcal{N}(0, \Sigma)
\]

(9)

where \(\bar{x}\) is a noise-free component also known as the mean and \(\epsilon\) is a noise component with zero-mean and covariance, \(\Sigma\). This representation, however, does not work for matrix Lie groups because the closure property would not hold, i.e., \(R_1 + R_2 \notin SO(3)\). So, the way we define random variables for matrix Lie groups is using the exponential map [9] [2]. For example, in \(SO(3)\) the uncertainty definition is given by:

\[
R = \hat{R}\text{Exp}(\epsilon), \; \epsilon \sim \mathcal{N}(0, \Sigma)
\]

(10)

with \(\hat{R}\) being the noise-free rotation matrix (or mean) and \(\epsilon\) the perturbation with zero-mean and covariance, \(\Sigma\).
3.4. System model
Throughout this work, a selected number of coordinate frames are necessary in order to represent the measured quantities by the camera and IMU. The respective geometric transformations of the monocular visual-inertial system are shown in Figure 2.

The eye frame, \( \{E\} \), is centered at the center of rotation of the eye. The coordinate system is according to the neuroscience convention. Here, the torsional component, \( x \), is in the observation (gaze) direction, \( y \) is the horizontal component that points to the left and \( z \) is oriented upwards.

The IMU frame, \( \{I\} \), belongs to the moving IMU. The origin is located at the center of the accelerometer and all the measurement vectors are represented in this frame. The frame assignment of the IMU is done according to the specific sensor datasheet.

The camera frame, \( \{C\} \), is the coordinate frame of the camera, centered at the pinhole and its orientation is also according to the neuroscience convention.

The world frame, \( \{W\} \), is the frame in which the system will navigate. The pose of the IMU and the camera are determined with respect to this one.

Both the camera, \( \{C\} \), and IMU, \( \{I\} \), are rigidly attached to the eye that rotates around a fixed point in a spherical joint (in the Eye reference frame, mentioned above). So, their movements are constrained to move in a sphere, with coupled position and rotation [14].

![Coordinate frame assignment of the visual-inertial system with the corresponding geometric transformations.](image)

Figure 2: Coordinate frame assignment of the visual-inertial system with the corresponding geometric transformations.

3.5. Gyroscope observation model
The 3-axis gyroscope measures the angular velocity in rad/s of the body frame with respect to the world frame, expressed in the body frame. The measured rotation rate, \( I_\omega_{W,I}(t) \), is affected by additive Gaussian noise, \( w_g(t) \), and bias, \( b_g(t) \):

\[
I_\omega_{W,I}(t) = I_\omega_{W,I}(t) + b_g(t) + w_g(t)
\]

with \( w_g(t) \sim \mathcal{N}(0, \Sigma_g) \) and \( \Sigma_g = \sigma_g^2 I_3 \in \mathbb{R}^{3 \times 3} \).

3.6. Accelerometer observation model
The accelerometer provides measurements of the specific force (acceleration together with gravity) in m/s². Like the gyroscope, the accelerometer model assumes the measurements are corrupted by Gaussian noise, \( w_a(t) \), plus a bias, \( b_a(t) \):

\[
I\hat{a}_{W,I}(t) = I\hat{R}_{W,I}(t)\left( I\hat{a}_W(t) - Wg \right) + b_a(t) + w_a(t)
\]

with \( w_a(t) \sim \mathcal{N}(0, \Sigma_a) \) and \( \Sigma_a = \sigma_a^2 I_3 \in \mathbb{R}^{3 \times 3} \).

3.7. State dynamics
In this model, we consider that the measurements of the gyroscope and accelerometer are responsible for the propagation of the orientation, \( W\hat{R}_I(t) \in SO(3) \), velocity, \( W\hat{v}_{W,I}(t) \in \mathbb{R}^3 \), and position, \( W\hat{o}_I(t) \in \mathbb{R}^3 \), of the IMU (from here on out called “body”) relative to the world frame. The IMU biases are modelled as random walks. Lastly, the 3D positions of \( p \) static landmarks, \( W\hat{P}_I(t) \in \mathbb{R}^3 \), are included in the state and will be tracked by the camera. Let us consider the following continuous time model of the system [6]:

\[
\begin{align*}
W\dot{\hat{R}}_I(t) &= W\hat{R}_I(t)I_\omega_{W,I}(t) \\
W\dot{\hat{v}}_{W,I}(t) &= W\hat{R}_I(t)I\hat{a}_{W,I}(t) - Wg \\
W\dot{\hat{o}}_I(t) &= W\hat{v}_{W,I}(t) \\
b_a(t) &= w_{bag}(t), \quad b_g(t) = w_{bga}(t) \\
W\hat{P}_I(t) &= 0, \quad i = 1, \ldots, p
\end{align*}
\]

where the vector \( I_\omega_{W,I}(t) \in \mathbb{R}^3 \) is the instantaneous angular velocity of the IMU relative to the world expressed in the IMU frame. \( I\hat{a}_{W,I}(t) \in \mathbb{R}^3 \) is the acceleration of the sensor relative to the world frame expressed in the sensor frame. The vectors \( b_a(t) \) and \( b_g(t) \in \mathbb{R}^3 \) are the IMU biases that are modelled as white noises. \( w_{bag}(t) \sim \mathcal{N}(0, \Sigma_{bag}) \) with \( \Sigma_{bag} = \sigma_{bg}^2 I_3 \in \mathbb{R}^{3 \times 3} \) for the gyroscope, and \( w_{bga}(t) \sim \mathcal{N}(0, \Sigma_{ba}) \) with \( \Sigma_{ba} = \sigma_{ba}^2 I_3 \in \mathbb{R}^{3 \times 3} \) for the accelerometer.

3.8. Time discretization
The dynamic model that describes the evolution of the state in [3, 7] is expressed in continuous time and, therefore, is now discretized using the first-order forward Euler method as in [9, 13]. The discrete time system can be expressed as

\[
\begin{align*}
R_k &= R_{k-1} \exp \left( (\hat{\omega}_k - b^g_{k-1} - w^gd_{k}) \Delta t \right) \\
v_k &= v_{k-1} + g\Delta t + R_k(\hat{a}_k - b^g_{k-1} - w^gd_{k}) \Delta t \\
o_k &= o_{k-1} + v_{k-1} \Delta t + \frac{1}{2}g\Delta t^2 \\
&+ \frac{1}{2}R_{k-1}(\hat{a}_k - b^g_{k-1} - w^gd_{k}) \Delta t^2 \\
p^i_k &= p^i_{k-1}, \quad i = 1, \ldots, p
\end{align*}
\]

where \( R_k \) is the rotation matrix, \( v_k \) is the linear velocity, \( o_k \) is the angular velocity, \( p^i_k \) is the position of landmark \( i \), and \( \Delta t \) is the discretization time step.
For convenience and readability, we dropped the coordinate frame superscripts and subscripts. The discrete time noise variables \( w_{k}^{gd} \) and \( w_{k}^{ad} \) have a covariance that is a function of the sampling time \( \Delta t: \Sigma_{k}^{gd} = \frac{1}{\Delta t} \Sigma_{a}^{gd} \) and \( \Sigma_{k}^{ad} = \frac{1}{\Delta t} \Sigma_{a}^{ad} \) for the gyroscope and accelerometer, respectively.

The bias is considered to remain constant between two consecutive image frames, \( k = i \) and \( k = j \):
\[
\begin{align*}
\mathbf{b}_{k}^{g} &= \mathbf{b}_{i+1}^{g} = \ldots = \mathbf{b}_{j-1}^{g} \\
\mathbf{b}_{k}^{a} &= \mathbf{b}_{i+1}^{a} = \ldots = \mathbf{b}_{j-1}^{a}
\end{align*}
\] (15)

3.9. Camera observation model

In combination with the measurements from the gyroscope and accelerometer, images from a camera are also collected. This information is used to detect landmarks using the standard pinhole model [13-19]. The measurement of landmark \( \mathbf{p}_{i} \) is given by,
\[
\mathbf{z}_{i} = \frac{1}{z_{w}} \left[ z_{v}^{x} \quad z_{v}^{y} \right] + \mathbf{n}_{i}, \quad i = 1, \ldots, p
\] (16)

with \( \mathbf{n}_{i} \sim \mathcal{N}(0, \mathbf{N}) \) and \( \mathbf{N} = \sigma_{z}^{2} \mathbf{I}_{2} \in \mathbb{R}^{2 \times 2} \). Each landmark observation, \( \mathbf{z}_{i} \), is obtained through the camera model in homogeneous coordinates:
\[
\begin{bmatrix}
z_{v}^{x} \\
z_{v}^{y} \quad z_{w}
\end{bmatrix} = \mathbf{K} \left[ \mathbf{R}_{j} \left( W \mathbf{R}_{j}^{T} \mathbf{p}_{i} + \mathbf{t}_{o_{w}}^{T} \right) + \mathbf{c}_{o_{j}} \right] \] (17)

where \( W \mathbf{p}_{i} \) is the position of landmark \( \mathbf{p}_{i} \) expressed in the world coordinate frame and \( \mathbf{K} \) is the intrinsic parameter matrix. In (17), the variables \( \mathbf{c}_{\mathbf{R}_{j}} \) and \( \mathbf{c}_{\mathbf{o}_{j}} \) refer to the constant relative pose between the IMU and the camera that is obtained from the extrinsic calibration of the two sensors.

3.10. Unscented Kalman Filter using Lie Group structure

Like with any Gaussian filter, the state distribution is defined by its mean, \( \hat{\mathbf{x}}_{k} \), and covariance, \( \mathbf{P}_{k} \). For our problem, the general mean state is given by
\[
\hat{\mathbf{x}}_{k} = \left( \hat{\mathbf{x}}_{k}, \mathbf{b}_{k} \right) \in \text{SE}(3)_{2+p} \times \mathbb{R}^{6} := \mathcal{S}
\]
\[
\hat{\mathbf{x}}_{k} = \begin{bmatrix}
\mathbf{R}_{k} & \mathbf{v}_{k} & \mathbf{0}_{k} & \mathbf{p}_{1} \cdots \mathbf{p}_{p} \\
\mathbf{0}_{(2+p) \times 3} & \mathbf{I}_{(2+p)}
\end{bmatrix} \in \text{SE}(3)_{2+p}
\] (18)
\[
\hat{\mathbf{b}}_{k} = \begin{bmatrix}
\mathbf{b}_{k}^{g} \\
\mathbf{b}_{k}^{a}
\end{bmatrix} \in \mathbb{R}^{6}
\]

where \( \mathcal{S} \) denotes the defined group for the state of the filter. The corresponding Lie algebra is denoted by \( \mathfrak{s} \). Under this topology, the orientation, velocity and position of the body (IMU) are combined with the landmark locations in a matrix of the form \( \mathbf{x}_{k} \in \text{SE}(3)_{2+p} \).

3.11. Uncertainty and state covariance

In order to define random variables for Lie groups we have to resort to another strategy since our state is not a vector. For the state \( \mathbf{x}_{k} \in \mathcal{S} \), its uncertainty is represented by
\[
\mathbf{x}_{k} \sim \mathcal{N}(\hat{\mathbf{x}}_{k}, \mathbf{P}_{k}) \rightarrow \left( \hat{\mathbf{x}}_{k}, \text{Exp}(\xi_{k}^{\mathcal{S}}), \mathbf{b}_{k} + \hat{\mathbf{b}}_{k} \right)
\] (19)

with \( \xi_{k} \sim \mathcal{N}(0, \mathbf{P}_{k}) \). The two Gaussian noise perturbations \( \xi_{k}^{\mathcal{S}} \), and \( \xi_{k}^{\mathcal{L}} \), for the matrix part \( \hat{\mathbf{x}}_{k} \), and for the vector part \( \mathbf{b}_{k} \) of the state, respectively. The two uncertainties in (20) are stacked together to give the total uncertainty of the state. For the bias vector, the Gaussian noise is additive, in the case of the matrix state we have to use the exponential map which is computed according to \( (5) \) and \( (6) \).

\[
\begin{align*}
\xi_{k}^{\mathcal{S}} &= \left[ \xi_{k}^{\mathcal{S}T} \xi_{k}^{\mathcal{L}T} \xi_{k}^{\mathcal{L}T} \xi_{p_{1}}^{\mathcal{L}T} \cdots \xi_{p_{p}}^{\mathcal{L}T} \right]^{T} \in \mathbb{R}^{9+3p} \\
\xi_{k}^{\mathcal{L}} &= \left[ \xi_{b_{a}}^{T} \xi_{b_{b}}^{T} \right]^{T} \in \mathbb{R}^{6}
\end{align*}
\] (20)

3.12. State propagation

The propagation of the state is described in Algorithm 7, and is done as follows. Letting \( f \) be the discrete time function that propagates the state variables described in (13)
\[
\hat{\mathbf{x}}_{k} = \left( \hat{\mathbf{x}}_{k}, \mathbf{b}_{k} \right) = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k}, \mathbf{w}_{k})
\] (21)

where \( \hat{\mathbf{x}}_{k} \) is the propagated mean state. The input vector \( \mathbf{u}_{k} \) is defined as 3-axis gyroscope and accelerometer readings, \( \mathbf{\omega}_{k} \) and \( \mathbf{\tilde{a}}_{k} \), respectively. Vector \( \mathbf{w}_{k} \) denotes the Gaussian noise of the process covariance matrix \( \mathbf{Q}_{k} \). The process starts by propagating the noiseless mean \( \hat{\mathbf{x}}_{k-1} \) and augment the covariance to include the process noise, \( \hat{\mathbf{S}}_{k-1} \). Then, sigma points for the augmented state uncertainty, \( \hat{\mathbf{X}}_{k-1} \), are generated and are passed through the dynamic model in (21). However, these sigma points are first mapped to the Lie group using (19). The propagated sigma points are finally transformed back to the tangent space where we then retrieve the covariance matrix \( \hat{\mathbf{S}}_{k} \). It is important to point out that both the propagation and the update steps follow the square-root implementation of the UKF, which propagates the covariance matrix through its Cholesky factor for being numerically more stable and computationally less expensive [21].

3.13. State update

For the update step of the filter, we will consider the camera measurements for each of the static landmarks. These measurements are the 2D pixel locations of the features tracked using the Kanade-Lucas-Tomasi (KLT) tracker [18] and take the form
of [10]. To facilitate reading, we compact all the camera observations in the following expression

\[ Z_k = \left[ \begin{array}{c} \hat{x}_k \\ \vdots \\ \hat{x}_k \end{array} \right] := H(\mathbf{\hat{x}}_k, \mathbf{n}_z) \]  

(22)

where \( \mathbf{n}_z \sim \mathcal{N}(\mathbf{0}, \mathbf{N}) \) is the Gaussian noise for one landmark and \( \mathbf{N} \) is the measurement covariance matrix.

Similarly to the propagation step, the update starts by computing the predicted measurement, \( \mathbf{Z}_0 \), using the propagated noiseless state, \( \mathbf{\hat{x}}_k \), and constructing the augmented state covariance by incorporating the measurement covariance, \( \mathbf{S}^A_k \). The \( 2L^A \) tangent space sigma points, \( \xi_{k,i}^b \), are then mapped to the respective Lie group and passed through the observation model \( H \) to give the sigma points of the predicted measurements, \( \mathbf{\hat{Z}}_k \). Next, the final predicted measurement mean, \( \mathbf{\hat{Z}}_k \), the cholesky factor of the measurement covariance, \( \mathbf{S}_{zz} \), and the cross covariance, \( \mathbf{P}_{xz} \), are computed so that we can get the Kalman gain, \( \mathbf{K} \), and the correction terms (innovation), \( \delta \hat{\mathbf{E}}^b \) and \( \delta \mathbf{\hat{E}}^b \). More details can be found in Algorithm 2.

4. Implementation
4.1. Full system architecture

The implemented system revolves around Figure 4 which shows a simple flow diagram with the main functions of the implementation. The measurements from the inertial sensor, i.e., angular velocity (rad/s) and acceleration (m/s²), are directly used to propagate the state of the filter and infer the motion of the body through simple Euler integration [9].

In the update step, the visual information provided by the camera is used to correct the trajectory and update the error covariances. It does so by detecting and tracking p static landmarks in the scene using the standard pinhole camera model in [3,9]. The filter uses the KLT tracker using minimum eigenvalue feature detection, both implemented in MATLAB [12].

In order to evaluate the performance of the filter, the orientation estimation is compared against a ground-truth. In the case of the simulator, as will be seen, this ground-truth corresponds to the generated trajectory. In the real case, the ground-truth is given by the joint angle feedback of the Kinova Gen3 robot that is used to infer the trajectory of the end-effector, where the camera and IMU are mounted (Figure 3).

4.2. Simulation (ESIM simulator)

In the previous work [14] done for this project, a MATLAB simulator that generates image points matches (with or without noise) with known rotations was developed. However, for this thesis, since the goal is to have a sense of the whole trajectory of the camera, this simulator becomes a bit short be-
cause we are not only interested in the initial and final orientation of the camera. Plus, this simulator skips the whole image processing task usually involved in visual odometry, such as feature detection and tracking. For these reasons, the option was taken to use the open source event camera simulator, ESIM [16], which was already being used in the context of the ORIENT project. The object of this work does not involve event cameras, however, this simulator is still able to provide regular camera frames, IMU readings and ground-truth (Figure 5).

4.3. Real system - Kinova Gen3

One of the main problems to solve in the previous implementation [14] was the aspect of the ground-truth when dealing with real world data. The ground-truth orientation could only be estimated up to an error of 4 degrees which compromised the comparison with the rotation estimation algorithms. For this work, the option was to use the Kinova Gen3 robotic arm with 7 degrees of freedom. Besides this very important feature, with the Kinova and its available API [2] (in C++, Python, MATLAB and ROS) it is possible to communicate with the robot via Ethernet connection and develop scripts to configure parameters, read sensor feedback, send joint commands, etc. The only downside of using the Kinova is its built in joint speed limitations of around 50 deg/s for the smaller joints and 57 deg/s for the bigger joints (see datasheet in [11]). Like with the experiment using the ESIM simulator, the purpose of working with the robotic arm was to send pre-computed, well defined trajectories that serve as ground-truth, and at the same time, collect the camera frames and inertial sensor readings.

The data collected contains the images, IMU measurements and the Kinova joint angles. The pre-computed trajectory is sent to the robot through its MATLAB API [2]. The measurements collected have the following characteristics: intensity frames at rate of 20 Hz (camera); inertial measurements (angular velocity in rad/s and linear acceleration in m/s²) at 100 Hz; joint angle feedback in radians at 10 Hz.

The ground-truth for this dataset was considered to be given by the joint angle feedback of the Kinova, due to the high precision of its actuators. The joint-wise trajectory is then transformed into the camera frame trajectory to give our ground-truth trajectory. Both the camera calibration and the extrinsic calibration between the camera and IMU were done recurring to the open-source Kalibr Toolbox from ETH Zurich [3].

This dataset has a few limitations that are worth mentioning. The vibrations of the actuator joints may throw-off and cause poor readings, specially in the IMU. It is also assumed that the trajectory planning done in MATLAB matches the movement.

https://www.kinovarobotics.com/en/resources/gen3-technical-resources
https://github.com/Kinovarobotics/matlab_kortex
https://github.com/ethz-asl/kalibr
performed by the Kinova, specially because we define a location of the principal point that might not correspond to the reality. There is error in the estimation of the relative pose between the IMU and the camera given by Kalibr. The timestamps may not be accurate and are not synchronised, which means that, whenever we get a camera measurement, the update does not refer to the actual movement but rather to the latest IMU measurement that was used in the propagation.

5. Error metrics

To evaluate the performance of the filter, for both the simulation and real experiment, we consider the geodesic distance between two points in \( SO(3) \), at time instant \( k \)

\[
e_k = \| \log(\hat{R}_k^T R_{GT}^k) \| \tag{23}
\]

where \( R_{GT}^k \) is the ground-truth orientation.

6. Experiments and results

6.1. Simulation (ESIM simulator)

The experiments with the ESIM were based on simple rotations around each one of the axes. The camera trajectories sent to the simulator had amplitudes of around 18 degrees (Figure 6).

The returned data (in the format of a .txt file) from the simulator consisted of the inertial measurements and ground-truth sampled at 1kHz, camera frames at around 15 Hz with timestamps, and the intrinsic and distortion parameters of the camera. The initial state of the filter was considered to be the first element of the ground-truth. The measurement noise for one landmark was set to 2 pixels standard deviation. Example plots for a rotation (in Euler angles) around the \( Y \) axis are found in Figures 7 and 8 respectively.

The Root Mean Squared Error (RMSE) for the camera rotation was calculated for each trajectory and is represented in Table 1.

<table>
<thead>
<tr>
<th>Axis</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE (°)</td>
<td>0.54</td>
<td>0.56</td>
<td>1.03</td>
</tr>
</tbody>
</table>

Table 1: Root Mean Squared Error in degrees for the rotation estimated by the filter.

From the plots and the table above, we can conclude that the filter is able to keep track of trajectory and have a low error.

One of the main issues found with this simulator was that, for our specific application, the image resolutions were quite low (which affects the tracking of features across the image) and we could not test faster movements (that resemble saccades for instance).

6.2. Dataset - Kinova Gen3

For the real world dataset, the procedure was very similar to the one used in the simulation. We generated pure rotation trajectories for each axis and sent
them to the Kinova. The obvious difference was the rate at which we sampled the data. The IMU captured at 100 Hz, the camera at 20 Hz and the joint angles for the ground-truth were sampled at 10 Hz. The reason for this value to be so slow resides in software constraints in the Kinova API, which restricts the feedback communication depending on the complexity of the commands that it has to execute. As with the ESIM, the filter was initialized to the first state of the ground-truth trajectory. It was early noticed that the real raw IMU measurements were extremely noisy and contained some outliers that were damaging the estimation. With this in mind, we decided to test the algorithm without, and with a prefiltering of the inertial measurements. We used the built-in MATLAB function that implements a median filter with window sizes of 10 or 15 samples.

We performed experiments for three different amplitudes of rotation (10, 15 and 20 degrees), for each individual axis. For each amplitude, we also tested the estimation against three different joint velocities (given by the trajectory planning). The procedure was repeated to include the cases where there is no pre-filtering of the inertial measurements and pre-filtering with a median-filter of window sizes of 10 or 15 samples. Tables 2, 3 and 4 summarize the results of the estimation in terms of the root mean squared error (difference between the estimated trajectory and the ground-truth provided by the Kinova). The initials MF10 and MF15 stand for "Median Filter size 10" and "Median Filter size 15", respectively. We also plotted the estimated trajectories against the ground-truth for three different experiments, as well as the evolution of the estimation error. They can be found in Figures 9, 10 and 11.

Table 2: Root Mean Squared Error in degrees for the rotation around the X-axis estimated by the filter.

<table>
<thead>
<tr>
<th>Amplitude</th>
<th>Max. speed</th>
<th>No pre-filter</th>
<th>MF10</th>
<th>MF15</th>
</tr>
</thead>
<tbody>
<tr>
<td>10°</td>
<td>9°/s</td>
<td>5.0°</td>
<td>1.5°</td>
<td>0.7°</td>
</tr>
<tr>
<td></td>
<td>11°/s</td>
<td>5.6°</td>
<td>2.1°</td>
<td>4.5°</td>
</tr>
<tr>
<td></td>
<td>13°/s</td>
<td>5.6°</td>
<td>2.1°</td>
<td>4.5°</td>
</tr>
<tr>
<td>15°</td>
<td>11°/s</td>
<td>10.1°</td>
<td>2.3°</td>
<td>4.7°</td>
</tr>
<tr>
<td></td>
<td>13°/s</td>
<td>11.0°</td>
<td>3.0°</td>
<td>4.6°</td>
</tr>
<tr>
<td></td>
<td>16°/s</td>
<td>7.3°</td>
<td>3.0°</td>
<td>2.4°</td>
</tr>
<tr>
<td>20°</td>
<td>13°/s</td>
<td>7.6°</td>
<td>3.6°</td>
<td>2.3°</td>
</tr>
<tr>
<td></td>
<td>15°/s</td>
<td>18.8°</td>
<td>3.0°</td>
<td>3.5°</td>
</tr>
<tr>
<td></td>
<td>18°/s</td>
<td>10.0°</td>
<td>2.2°</td>
<td>3.1°</td>
</tr>
</tbody>
</table>

Table 3: Root Mean Squared Error in degrees for the rotation around the Y-axis estimated by the filter.

<table>
<thead>
<tr>
<th>Amplitude</th>
<th>Max. speed</th>
<th>No pre-filter</th>
<th>MF10</th>
<th>MF15</th>
</tr>
</thead>
<tbody>
<tr>
<td>10°</td>
<td>9°/s</td>
<td>7.0°</td>
<td>3.1°</td>
<td>4.3°</td>
</tr>
<tr>
<td></td>
<td>11°/s</td>
<td>7.7°</td>
<td>2.1°</td>
<td>4.5°</td>
</tr>
<tr>
<td></td>
<td>13°/s</td>
<td>7.0°</td>
<td>2.5°</td>
<td>11.6°</td>
</tr>
<tr>
<td>15°</td>
<td>11°/s</td>
<td>12.9°</td>
<td>2.6°</td>
<td>2.7°</td>
</tr>
<tr>
<td></td>
<td>13°/s</td>
<td>8.6°</td>
<td>3.3°</td>
<td>1.7°</td>
</tr>
<tr>
<td></td>
<td>16°/s</td>
<td>10.4°</td>
<td>2.6°</td>
<td>2.6°</td>
</tr>
<tr>
<td>20°</td>
<td>13°/s</td>
<td>8.7°</td>
<td>3.6°</td>
<td>4.7°</td>
</tr>
<tr>
<td></td>
<td>15°/s</td>
<td>11.6°</td>
<td>2.2°</td>
<td>6.0°</td>
</tr>
<tr>
<td></td>
<td>18°/s</td>
<td>24.3°</td>
<td>3.6°</td>
<td>4.4°</td>
</tr>
</tbody>
</table>

Table 4: Root Mean Squared Error in degrees for the rotation around the Z-axis estimated by the filter.

The results from the rotation around the X-axis, represented in Table 2, show that the best performances were achieved for the case of greater amplitude of movement (20 degrees). For amplitudes of 15 degrees the filter revealed to be quite inconsistent and poor in terms of estimation error. It’s noticeable that the pre-filtering overall improves the error with the exception of a few cases. It would be expected, since the rotation around the torsional axis is the simplest one, that the filter would better estimate the orientation, however that is not always the case. With visible landmarks in almost every frame, it is possible that the source of these errors might come from fine tuning in parameters of the filter or bad initial guess of trajectory and biases.

For rotations around the Y and Z axis the results seem to improve slightly, especially in terms of consistency. In these experiments, the effect of pre-filtering the raw gyroscope and accelerometer measurements, has, overall, significant impact on the RMSE. If we increase the size of the window to 15 samples, the estimation starts to worsen, probably because we are filtering out reliable measurements.

Finally, from the inspection of the plots (Figures 9 and 10), it is possible to see that, in general, the filter does a good job of tracking the rotation around the “main” axis. A large amount of error seems to come from the drift around the axis where there is no rotational movement. This drift shows that filter is not able to keep track of the bias co-
directly, leading to bad estimations.

Figure 9: Example experiment: Rotation of 20 degrees around the Y-axis with maximum joint speed of 18 degrees per second for 3 different scenarios.

Figure 10: Example experiment: Rotation of 20 degrees around the Y-axis with maximum joint speed of 18 degrees per second for 3 different scenarios (zoomed in on the Y-axis).

7. Conclusions
The objective of this thesis was to develop an algorithm that could fuse sensor information from an IMU and a camera to accurately estimate the orientation of a camera that would be embedded in the current eye prototype. The proposed approach was the Unscented Kalman Filter on Lie groups, which can be leveraged in visual-inertial odometry. We found that the filter can indeed be used to track rotational trajectories, both in simulation and in real world scenarios, with reasonably good accuracy. However, there are still quite a few problems, namely in terms of consistency (the filter is sensitive to the tuning of noise parameters of the IMU) and computational efficiency (as of now, the filter is still slow and needs to be optimized and adapted to fit a real-time application). The fact that we were only able to perform slow movements for the benchmarking process, most likely also compromised the consistency of the filter because of the poor signal-to-ratio of the IMU at low frequencies. Faster movements would improve the readings, but would at the same time affect the camera frames. The depth ambiguity can also become a problem since it may cause the filter to diverge.

Overall, this method can definitely be used to estimate orientations for our existent eye prototype.

7.1. Achievements/contributions
The main contributions added to the project include an overview on how to deal with visual-inertial odometry problems; full characterization of the sensors used in the prototype for benchmarking purposes; design and development of a real world experiment with reliable ground-truth; implementation in MATLAB of a sensor fusion algorithm that combines inertial measurements with camera frames (UKF-LG).

7.2. Future Work
To finish this work, it is possible to identify several ways to improve or new paths to explore. Unfortunately, this work was not able to include minimization of the back projection error (MBPE) approach studied in [14]. So, a possibility is to try to incorporate the latter in the UKF-LG. To the best of our knowledge there is not really a dataset for this application of rotation only movements (especially eye saccades). For future work, it would be interesting to develop both a simulator and a real world setup specifically designed to help the problem of eye saccades. A prototype gimbal could be used in this new benchmarking process. As for the simulator, it could also be interesting to develop an
ESIM type simulator but for eye movements. The user would define a saccade trajectory and the result would be images with resolution close to what would be expected from the human eye, as well as, IMU measurements and ground-truth.

References