Visual Inertial Odometry with Event Cameras

José Pedro Ribeiro Gomes
Instituto Superior Técnico / UTL, Lisbon, Portugal
josepgomes@tecnico.ulisboa.pt

I. Introduction

Vision plays a very important role in the animal kingdom, and virtually every higher order animal has developed some sort of visual system to improve their chance of survival. As such, it is only natural that sensors that can imbue artificial systems with the sense of sight have been created, in particular cameras (what we call throughout this work as "conventional cameras", to distinguish from event cameras, explained briefly). However, conventional cameras do not exactly mimic animal's visual system. They are much slower (typically produce around 20-30 fps), produce redundant information, and are much more energy-costly. Furthermore, they are not very good with scenes with high contrast (as detail is lost in bright and dark areas), or with high movement (as images produced suffer from motion blur).

Neuromorphic hardware appears as a bio-inspired approach to hardware development that tries to replicate the advantages of animal systems, either in speed, energy efficiency, or any other positive or desirable attribute. The work described within this report focuses on Dynamic Vision Sensors type of neuromorphic cameras (DVS cameras [1]), which are a type of event cameras that report changes in the brightness captured by each pixel (precisely, the log-intensity of the brightness captured by each pixel). Unlike conventional cameras (that record a sequence of the intensity of all pixels in the scene, and therefore produce redundant information, and are not energy efficient), event cameras produce "events", which contain the timestamp, pixel location, and polarity of the change in the pixel.

This approach has multiple advantages, such as 1) lower latency (because there is no need for video compression at the camera level), 2) higher energy efficiency, 3) no redundant information, 4) higher temporal resolution (in the order of microseconds (as opposed to milliseconds of conventional cameras)), and 5) higher dynamic range, to name a few.

With the advantages of event cameras in mind, we set out to develop a system that is able to estimate the pose (position and orientation) of a system based on event cameras. This work appears integrated in the ORIENT project, that focuses "[$\cdots$] on Neuroscience, with a goal to better understand how the brain coordinates movements in the eyes and head, in order for humans to orient themselves in the world and in relation to any object that might be around" [1]. In particular, this work appears as a study on the possibility of using an event camera to estimate the orientation of the eye.

[1] https://welcome.isr.tecnico.ulisboa.pt/orient-project-collaboration/

A. Overview of Related Works

Event cameras have demonstrated to be useful in multiple tasks where speed is paramount, of which quadrotor control comes immediately to mind ([2]). They have also been adopted in areas such as flow estimation ([3]) , and image and video reconstruction ([4]) , to name a few areas.

The problem of pose estimation shares some goals and similarities with SLAM (Simultaneous Localization and Mapping), as one of the problems is that of localization, which relies on a correct estimation of the pose of the system. According to [5], the first work on camera tracking with an event camera was presented in [6], and proposed an implementation based on particle filters, but was limited to a planar motion.

[7] proposed an implementations which, event though limited to rotation, and therefore without the need of translation or depth, paved the way for more complex implementations, such as [8], which we consider to be the state of the art in terms of localization using event cameras.

Multiple authors have opted for approaches that try to rely on bridging the "classic" approaches with event cameras. For example, [9] relies on features tracked by [10], and combines them with IMU information by means of an Extended Kalman Filter (EKF).Our proposed approach borrows from this idea.

On the "conventional" side of pose estimation, multiple SLAM approaches based on vision (Visual SLAM, or vSLAM) are worth mentioning, however we highlight ORB-SLAM ([11]).

B. Problem Formulation

This work appears in the sequence of the previous works in the Orient group (in particular [12]) where the orientation of an eye is estimated by means of visual odometry. The eye produces very fast (under 200 ms) and short (usually under 25 deg) movements, called saccades, on the order of 700 deg/s ([13]). This poses some challenges for conventional cameras, that are susceptible to motion blur.

It is the goal of this work to design and suggest methods that may be used to solve this problem of eye orientation, by focusing on the larger problem of localization using event cameras (for instance, though not relevant in the context of an eye, we decided to estimate translation, not only to compare our approaches with others, but also to try and contribute to the state of the art).

We propose an approach that leverages visual and inertial information (by means of events cameras with frames, events and Inertial Measurement Unit (IMU)) in order to estimate the pose of the system using an Unscented Kalman Filter (UKF) based on Lie groups. We then propose further approaches that

rely less and less on frames, trying to push the information obtainable from events to its limit.

Our approach is inspired by SLAM approaches, that simultaneously create a map while localizing the system in it, of which ORB-SLAM and EKFSLAM are common examples. Though our implementation is not that of SLAM, since no map is being generated (only a local map relevant for localization), most concepts are still valid, in particular we borrowed the idea of a state that contains the pose of the system (and some other useful variables) that is constantly being estimated.

C. Contributions

We propose a Lie group-based UKF approach to solve the pose estimation problem which, to the best of our knowledge, is a novel approach in the context of event cameras. This approach attempts to combine visual information in the form of events and frames, with inertial information obtained from an IMU, by means of an Unscented Kalman Filter. Also, we propose the use of the Event-based Kanade-Lucas-Tomasi tracker (EKLT, [14]) in the context of pose estimation, which, to the best of our knowledge, has not yet been done before.

Furthermore, we list, implement and compare possible ways to improve said tracker by taking into account the current estimation of the pose, in effect improving sensor output by combining it locally (at the sensor level) with measurements from outside the sensor, which is an unusual approach to improving sensor reading.

Through this work, multiple tools have been developed, such as trajectory generators for simulations, motion capture scripts, simulations, datasets, and the UKF implementation itself, which we hope can be of use for future works.

II. Background and State of the Art

A. Camera Projection Model and Imaging Features

1) Camera Model: An important concept to take into account is that of camera model, which models the correspondence between points in the world and their position in image space. A common model used is the projective model (others can be used, such as perspective, affine, and orthographic models), in particular the pinhole model (Fig. 1). In this model, the mapping from world (3D) to camera (2D) is performed by tracing a ray of light from the world, through an infinitesimally small aperture (pinhole), and into the image plane. Important parameters in this model are the focal distance (distance from aperture to image plane), and image (or optical) centre (centre of the image plane), which are both intrinsic parameters.

From this model, we can derive the projective geometry (Fig. 2), where the image plane is modelled in front of the optical centre, and the optical axis is orthogonal to the image plane. In this model, lines are projected to lines, collinear features remain collinear, and tangents and intersections are preserved, but parallel lines (in the world) eventually meet at a vanishing point, since angles are not preserved.

This geometry makes it clear that projection between world and image are given by the triangular similarity $x = f\frac{X}{Z}, y = f\frac{Y}{Z}$, where X, Y, and Z represent 3D coordinates in the real

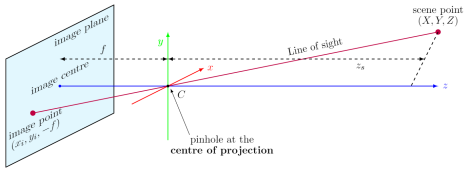


Fig. 1. Pinhole camera model and projectile geometry

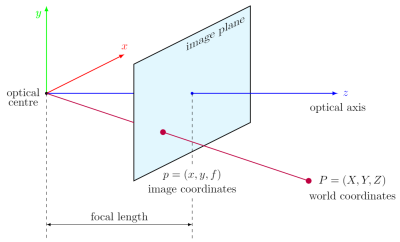


Fig. 2. Projective geometry

world, x and y their corresponding projected 2D points, and f the focus length of the camera.

The intrinsic parameters combine these properties, namely focal length (f), offset to the optical centre (c_x and c_y), and skew (s, from non-orthogonality between optical axis and image plane), and constitute the intrinsic parameters matrix K. This matrix is also present in the projective matrix P, which relates the points in world space to their corresponding image plane position, given by

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} K & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

which is critical for computer vision.

Assuming camera rotation and translation (extrinsic parameters, since image and world frames are not centred), the projective model can be expanded to include rotation R and translation t, and becoming completely generic, in what is called the camera matrix, given by

$$\mathbf{x} = K \begin{bmatrix} R & t \end{bmatrix} \mathbf{X} \quad . \tag{2}$$

2) Feature Detection: In the context of computer vision and image processing, image features are distinctive landmarks in the images, preferably insusceptible to point-of-view, scale, and the aperture problem. Features are important as they provide information on the image, which can be used for recognition, matching, reconstruction, and tracking, among many other applications. Many types of features can be considered, such as edges, corners, blobs, ridges, and shapes. The process of identifying features in an image is called feature detection, and multiple detectors have been described in the literature, dependent on the features of interest, such as Canny and Sobel detectors (for edges), Hough transform (for shapes), Laplacian operator (for blobs), and Harris detector (for corners).

a) Harris Corner Detector: The typical example for a classic corner detector is the Harris Corner Detector ([15]). It works using the following steps: 1) Compute the x-wise $I_x(x,y)$ and y-wise $I_y(x,y)$ partial image derivatives, 2) Compute the second-order derivatives $I_x^2(x,y)$ and $I_y^2(x,y)$, and cross-derivatives $I_xI_y(x,y)$, 3) Compute the second-moment matrix $M(x,y)$, 4) Compute the Harris score, and 5) Detect local extrema whose Harris score is greater than the set threshold.

The partial derivatives are computed by applying a Sobel derivative kernel (usually 3x3 or 5x5 kernels) to the whole image, producing the x-wise $I_x(x,y)$ and y-wise $I_y(x,y)$ partial image derivatives. From these derivatives, we can define the vector $\nabla I(x,y) = (I_x(x,y), I_y(x,y))^T$ and the second-moment matrix for each pixel ($M(x,y)$), defined as $M(x,y) = \sum_{(x,y)\in patch} g(x,y)\nabla I(x,y)\nabla I^T(x,y)$, where $g(x,y)$ is a Gaussian weighting function centred around (x,y), which controls the "sharpness" of the edge. Then, we can compute the Harris score as defined by

$$H(x,y) = \lambda_1\lambda_2 - k \times (\lambda_1 + \lambda_2)^2 = \det(M) - k \times \text{trace}(M)^2 \tag{3}$$

where k is an empirical value, $k \in [0.04; 0.06]$. Finally, if $H(x,y) \geq H_0$, we consider the pixel as a corner. This will produce corner blobs. In order to select a single pixel to represent the corner, we select the local extrema (the pixel with the highest Harris score).

B. Event Cameras

Event cameras, also called neuromorphic cameras, silicon retina or dynamic vision sensor (DVS) are image sensors that respond to changes in brightness in the scene. Unlike conventional cameras, which capture full image frames at a fixed frequency (commonly 30Hz or 60Hz), producing redundant information and requiring a high bandwidth for transmission, each pixel in an event-based camera operates independently and asynchronously, reacting to changes of brightness in the scene, eliminating the transmission of redundant information, allowing for much higher temporal resolution (in the order of microseconds, as opposed to the milliseconds of conventional cameras). They possess other interesting properties, such as high dynamic range (above 120dB), which allows for scenes with both bright and dark zones, and does not suffer from under/overexposure, nor motion blur.

Events are triggered when the brightness in a certain pixel surpasses a certain threshold. In particular, discrete brightness steps are pre-defined, and whenever brightness detected crosses the threshold, an event is generated. Positive crossings generate ON events, and negative crossings generate OFF events. In effect, each pixel is constantly working as a comparator (with corresponding electronic to support this mode of working).

Events are then defined as a four-component vector:

$$\mathbf{e} = \left((x,y)^T, t, pol\right)^T = (\mathbf{p}, t, pol)^T \tag{4}$$

The component $p = (x,y)^T$ refers to the spatial position of the event in the camera. The component t refers to the timestamp of the event, and is of extreme importance due to the microsecond temporal resolution of the camera. Lastly, the parameter pol refers to the polarity of the event (ON/OFF events). With this event structure, it is common to represent events in a three-dimensional (space-time), representation, as shown in Fig. 3.(d), which shows the space-time evolution of events generated from a rotating black bar on a white background, over a period of 1000 ms.

Conventional cameras and event cameras have fundamentally different modes of operation and output. As such, a comparison of the behaviour in the same scene, and an analysis of the output, is interesting. Fig. 3 shows the response of both a conventional camera and an event camera when presented with a disk rotating at a high speed, with a black dot. The fixed capture of the conventional camera is unable to keep up with the speed of the dot, and the images suffer from motion blur (not represented) and some discontinuity between frames. The event camera, however, due to its asynchronous event generation and temporal resolution, is able to continuously produce events relating to the movement of the dot.

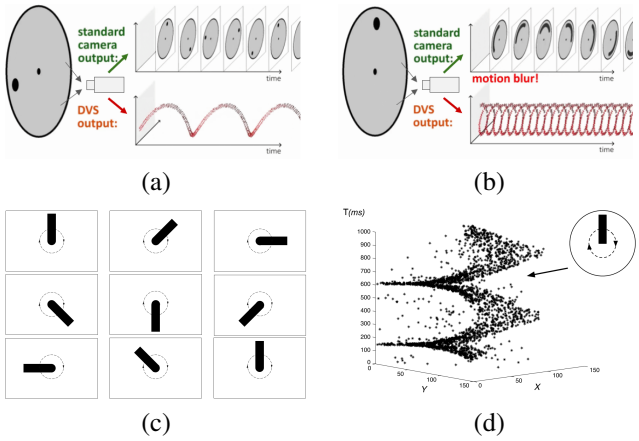


Fig. 3. Comparison of the output of a standard camera (above), and an event camera (below), when recording a rotating disk with a black dot, adapted from [16]. (a) shows a lower speed rotation; (b) shows a higher speed rotation, with motion blur from the conventional camera. Also, space-time representation of events (d) generated from a rotating black bar (c), from [17].

Advances in camera manufacturing have allowed for cameras that have both conventional camera pixel arrays, and event camera pixel arrays (DAVIS cameras). This enables hybrid algorithms, which take advantage of the benefits of event cameras, with the extensive research on conventional cameras.

1) Feature Detection and Tracking on Event Cameras: For event-based cameras, new types of features, as well of detectors, are being proposed, as classical techniques are not easily transferable in most cases, or result in a non-negligible performance decrease, due to conversion overhead from asynchronous events to frames. Due to the nature of events, gradient operators are not possible (at least directly applied to the event stream), since there is no image on which to apply them, and multiple techniques have been proposed. We choose to isolate the Event-Based Lucas-Kanade-Tomasi

Feature Tracker (EKLT), which was used for the proposed methods.

EKLT ([14]) is a hybrid feature tracking technique that is able to merge information from conventional cameras and events (and hence is more suitable for the new generation of DAVIS event cameras), that tracks corners across time. The method is based on the Lucas-Kanade tracker, hence the name EKLT (Event-based Lucas-Kanade tracker). This method tracks corners, as they are easy to recognize in both conventional cameras and correspond to areas with high event generation.

The idea behind this tracker is to detect features using a conventional frame, which are then tracked using events until a new frame arrives, at which point the estimation from events is compared to the corner detection in the new frame, in essence correcting this estimation. If the feature is not detected, it is still tracked in event space, as subsequent frames may re-detect missed features. This approach is particularly useful in high-speed movements, where motion blur becomes a problem for frames, but not for events.

This comparison between frames and events is crucial, and the key concept is "image variation in a frame patch". As previously discussed (Section II-B), event cameras respond to brightness changes in the environment. Therefore, it is not farfetched to compare events to image gradients, as zones with higher gradients in the world are precisely the ones that produce the most events. In fact, integration (accumulation) of events over a period of time produce results that are very similar to the gradient of the image, as shown in Fig. 4. This is the idea at the core of this approach, as the brightness change behaves as the descriptor for the features, and are used as patches for a Lucas-Kanade inspired patch comparison and matching, using both the patch and estimated velocity (estimated through events)

While a new frame is not received, the corner is tracked in event space and the local patch is being created for comparison with a frame patch created from image gradients, as shown in Fig. 4.

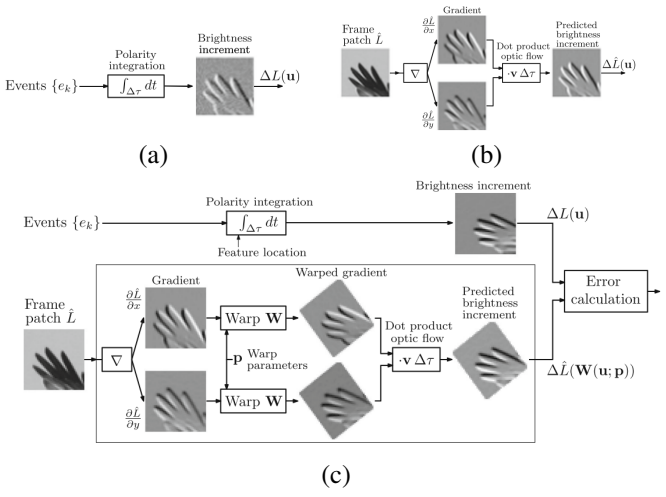


Fig. 4. Comparison of the brightness change from event integration (a), versus the brightness change from image gradient (b). (c) shows the block diagram of EKLT, illustrating the comparison between brightness changes from images and events, from [14].

It is worth noting, however, that the dependence on corners may present a problem for low-textured, or highly organic environments, where high quality corners are not always present. Also worth mentioning is the parameter v shown in Fig. 4, which accounts for the optical flow. Though inconspicuous at first glance, v is crucial for the generation of the Predicted Brightness Increment, as it estimates the flow angle (the direction objects in the image are moving), which is needed to predict the polarity of the events and generate the template based on frames to compare against the real Brightness Increment generated from events. In the extreme case that $v = 0$, the Predicted Brightness Increment is blank, and the matching of templates fail. This is a problem that can occur when changing direction of movement abruptly.

C. Inertial Measurement Unit (IMU)

The event cameras used for this work all have an Inertial Measurement Unit (IMU), with coordinate frame matching the camera frame. As such, explaining the type of information that is produced by the IMU is fundamental. The IMU is a sensor that reports the linear acceleration and the angular velocity of a body by means of accelerometers and gyroscopes (sometimes also the orientation by means of magnetometers, but the IMU used did not have this capacity and therefore it is not analysed).

Starting with the gyroscope, and considering a single axis, it provides a measurement $\tilde{\omega}$ that relates to the angular velocity of that axis, and is (according to [18]) given by

$$\begin{aligned} \tilde{\omega} &= \omega + \omega_b + \eta_\omega \\ \eta_\omega &\sim N\left(0, \sigma^2_{gyro}\right) \end{aligned} \tag{5}$$

which corresponds to the true angular velocity ω corrupted by the sensor bias ω_b and the sensor noise η_ω, which is modelled as additive, zero-mean Gaussian noise.

In order to have 3DOF we use 3 gyroscopes, one for each orthogonal axis, and we assume no crosstalk between them. The bias is temperature dependent and can vary over time, but is generally modelled as a constant, and may be specified in the manufacturer's datasheet, alongside the sensor variance σ^2_{gyro}.

Continuing with the accelerometers, a similar reasoning can be applied. We have a measurement $\tilde{a}$ given by

$$\begin{aligned} \tilde{a} &= a + a_b + \eta_a \\ \eta_a &\sim N\left(0, \sigma^2_{acc}\right) \end{aligned} \tag{6}$$

dependent on the true value a, sensor bias a_b and noise η_a. The accelerometer deserves special attention, as it does not measure acceleration relative to a coordinate frame as one would expect. Rather, this acceleration is relative to a free fall state, meaning the reading is 0 when the sensor is free falling. When static, it measures the gravitational acceleration pointing upwards (since the accelerometer is accelerating upwards comparing to a free fall).

D. Pose Estimation

Pose estimation refers to the problem of trying to estimate the position and orientation (collectively called the pose) of a system. A common approach is to use information provided by a camera to detect and track a set of features, which are used to estimate the motion of the system (ego-motion), as there is a relation between the 3D point in the world and its corresponding 2D projection in camera space (Section II-A1), which is also influenced by the movement of the system (Section II-D1).

1) Camera Motion and Motion Field: There is much information to be extracted from a time-varying sequence of images. When a camera moves (or when objects move in front of a camera), there are changes in the captured image, that can be used to estimate the motion of the camera. Let us consider the perceived movement of scene points generated exclusively by the movement of the camera (ego-motion), with translational velocity T and angular velocity ω. In relation to the observer, a 3D point $P = (X, Y, Z)^T$ moves, according to [19], following

$$\dot{P} = -T - \omega \times P = - \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} - \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{7}$$

Recovering the projection equations presented in Section II-A1, in particular

$$\begin{cases} x = f\dfrac{X}{Z} \\ y = f\dfrac{Y}{Z} \end{cases} \Rightarrow \begin{cases} \dot{x} = f\dfrac{Z\dot{X} - X\dot{Z}}{Z^2} \\ \dot{y} = f\dfrac{Z\dot{Y} - Y\dot{Z}}{Z^2} \end{cases} \tag{8}$$

Combining (7) and (8), we arrive at

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} -T_x + \frac{x}{f}T_z \\ -T_y + \frac{y}{f}T_z \end{bmatrix} + \begin{bmatrix} \omega_x \frac{xy}{f} - \omega_y \left(f + \frac{x^2}{f} \right) + \omega_z y \\ \omega_x \left(f + \frac{y^2}{f} \right) - \omega_y \frac{xy}{f} - \omega_z x \end{bmatrix} . \tag{9}$$

Analysing this equation is interesting, and we can see the left part pertains to translation, and the right to rotation. Furthermore, we can also see that translational component depends inversely on depth, and there is scale ambiguity (meaning T and Z can only be recovered up to a scale factor). On the rotation side of things, we see that there is no dependence of depth, and therefore depth estimation is not possible without translation.

2) Visual Odometry: Visual odometry appears as the inverse problem to (9), as we want to estimate the translational and angular velocities of our camera, assuming we know how certain specific points in our camera plane (features), are moving along time. Odometry itself is the use of data from movement sensors to estimate the change of position and orientation of a system over time. It naturally follows that visual odometry is the use of visual information (in particular cameras) to estimate this displacement.

A typical pipeline for visual odometry is as follows: 1) Acquire image; 2) Undistort image, 3) Detect features (Match features between consecutive frames, and Obtain optical flow field), 4)Estimate camera motion from optical flow (a common approach is the use of Kalman Filter.

Visual odometry is particularly relevant in the context of SLAM, more specifically Visual SLAM (also called vSLAM), whose goal is to use visual information to construct a map of the system's surrounding environment, and to place the system in said map. In this case, the visual component is useful to estimate the motion of the system across time (and also to correct the predicted pose of the system when it passes through a previously mapped zone (loop closure)). Multiple SLAM approaches have been proposed, of which we highlight FastSLAM ([20]), ORB-SLAM ([11]), EKF-SLAM, and GraphSLAM ([21]). These approaches rely on conventional cameras to provide features that are used to construct a map and also be used as reference for the estimation of the pose of the system.

III. Visual Inertial Odometry based on Event Cameras

A. System Model

Our system consists of an event camera that contains an IMU sensor embedded. It is important to understand what each sensor is reading and what reference frame each one uses to make sense of the data being fed into, and received from, the system. By wanting to estimate the pose of our camera, what is implicit in this statement is that we have defined some sort of reference in the world (for example, the starting point of the camera trajectory), and we want to define its current position and rotation (pose) with regard to this initial reference.

Our IMU reports two types of information: angular velocity $\omega = [\omega_x\, \omega_y\, \omega_z]^T \in \mathbb{R}^3$ and linear acceleration $a = [a_x\, a_y\, a_z]^T \in \mathbb{R}^3$. However, this information also has a reference frame implied. As such, the accelerometer reads movement on the x-axis, for instance, this reading is placed in its own reference frame, and somehow needs to be related to the world frame so that in can be useful in the context of pose estimation (the same applies for gyroscope information, which also relates to the internal reference frame of the IMU).

Luckily, to relate two frames of reference, we only need to rotate their axis so they align, and translate their centres to match, by means of the rigid transformation $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$, where T denotes the transformation, R the rotation, and $\mathbf{t}$ the translation. The set of rigid body transformations constitutes the Special Euclidean Group (SE). Multiple transformations can be successively performed, so that nested frames of reference can be used (frames references that depend on other frames of references to be described, so that moving the former automatically moves the latter).

The camera also has a reference frame for the image frame. Luckily, this reference frame is aligned with the IMU reference frame. As such, the critical frames of reference are the world frame (on which we are estimating the pose of the camera), and the IMU/camera reference frame. This setup is shown in Fig. 5, showing the world frame $\mathcal{W}$, with regards to which we want to estimate the position and orientation of our system,

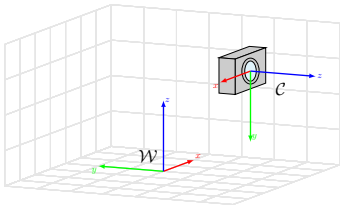


Fig. 5. Representation of our system model, with relevant coordinate frames of reference described, in particular, world frame $\mathcal{W}$ and camera frame $\mathcal{C}$.

and the camera frame $\mathcal{C}$, with regards to which the sensor readings are produced.

To reiterate, the odometry methods presented in this chapter intend to estimate the position of the camera, at all times, with regards to the world frame $\mathcal{W}$.

B. Visual Inertial Odometry

Our filtering and sensor fusion approach is based on [22], which proposes a filter that integrates IMU and visual information in the form of a UKF (Unscented Kalman Filter). This filter introduces several suggestions worth mentioning, such as 1) a Lie group structure for the state space (resulting in a matrix state space, rather than a vector state space), 2) integration of the landmark position in the Lie group, and 3) representation and computation of the uncertainty directly in the Lie group, rather than outside of it, followed by a conversion to the Lie group.

This filter that estimates the body pose (position and velocity) and 3D landmark positions, as well as accelerometer's and gyroscope's biases. The first two parameters (pose and landmark positions) are integrated in the Lie group structure, and the latter two (biases) are appended to the state estimation. Though this is not a SLAM implementation, as the whole area is not being mapped (only the local, current region in the vicinity of the body is estimated), the formulation itself is very similar as if it were a SLAM problem (and could be extended if desired).

Our approach, based on the integration of the visual information (processed by EKLT) with inertial information by means of the filter presented, aims to leverage a synergy and complementarity between the two types of information. The global overview of this approach is shown in Fig. 6.

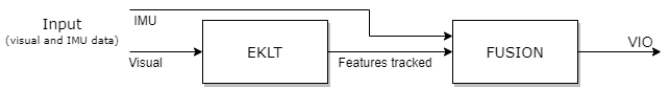


Fig. 6. First proposed approach, where the feature tracker EKLT is fed into the pose estimator, FUSION, a Lie groups-based Unscented Kalman Filter.

1) Filter Structure: This filter expands on the works of [23], which introduced a Lie group based EKF, following the suggestions of [24], that showed the similarities between the SLAM problem and the group $SE_{1+p}(3)$. [25] proposed a UKF implementation on $SE(3)$. Lastly, [22] proposed the inclusion of the landmarks into the group itself, creating a group $SE_{2+p}(3)$, coupled with a UKF implementation. Furthermore, two techniques of propagating the uncertainty were suggested, of which we kept the right uncertainty from their proposed Right-UKF-LG.

a) State Space: The state being estimated by the filter is given by the tuple (χ, b) where χ is defined as

$$\chi = \begin{bmatrix} \mathbf{R} & \mathbf{v} & \mathbf{x} & \mathbf{p}_1 \cdots \mathbf{p}_p \\ 0_{(p+2)\times 3} & & I_{(p+2)\times(p+2)} & \end{bmatrix} \tag{10}$$

and corresponds to a Lie group $SE_{2+p}(3)$ that incorporates the orientation $\mathbf{R} \in SO(3)$, velocity $\mathbf{v} \in \mathbb{R}^3$ and position $\mathbf{x} \in \mathbb{R}^3$, as well as the 3D positions of the landmarks $\mathbf{p}_1, \ldots, \mathbf{p}_p \in \mathbb{R}^3$.

$b \in \mathbb{R}^6$ is the bias vector, defined as

$$b = \begin{bmatrix} b_\omega^T b_a^T \end{bmatrix} \tag{11}$$

containing the gyroscope and accelerometer biases b_ω and b_a.

b) Dynamics Model: The system can be modeled by

$$\text{body state} \begin{cases} \dot{\mathbf{R}} = \mathbf{R}\left(\omega - b_\omega + n_\omega\right)_\times \\ \dot{\mathbf{v}} = \mathbf{R}\left(a - b_a + n_a\right) - g \\ \dot{\mathbf{x}} = \mathbf{v} \end{cases} \tag{12}$$

$$\text{IMU biases} \begin{cases} \dot{b_\omega} = n_{b_\omega} \\ \dot{b_a} = n_{b_a} \end{cases} \tag{13}$$

$$\text{landmarks } \dot{\mathbf{p_i}} = 0,\ i = 1, \ldots, p \tag{14}$$

where we have access to angular velocity ω and linear acceleration a through the IMU mounted on the system.

The notation $(\omega)_\times$ represents the skew symmetric matrix associated with the cross product with vector $\omega \in \mathbb{R}^3$

$$n = \begin{bmatrix} n_\omega^T & n_a^T & n_{b_\omega}^T & n_{b_a}^T \end{bmatrix}^T \sim \mathcal{N}(0, Q) \tag{15}$$

2) Measurement Model: Visual information is also fed into the system by means of a calibrated monocular event camera, in order to correct the predicted state of the system. The camera observes and tracks p landmarks through the standard pinhole model and corresponding projection model (Section II-A1):

$$y_i = \begin{bmatrix} y_u^i \\ y_v^i \end{bmatrix} + n_y^i \tag{16}$$

where y_i is the normalized pixel location of the landmark in the camera frames, and $n_y \sim \mathcal{N}(0, N)$ represents the pixel image noise.

This location is then compared with the expected location of the feature in camera space, obtained by projecting the estimated 3D position of the landmark into camera space through

$$\lambda \begin{bmatrix} x_u^i \\ y_u^i \\ 1 \end{bmatrix} = \Pi \left[\mathbf{R}_C^T \left(\mathbf{R}^T \left(\mathbf{p}_i - x \right) - x_c \right) \right] \tag{17}$$

where Π denotes our camera matrix, $\mathbf{R}_C^T$ our initial rotation of the system, $\mathbf{R}^T$ the current estimated rotation, $\mathbf{p}_i$ the $i-th$ landmark 3D estimated position, x the estimated position of the system, and x_c the initial position of the system.

Since events do not naturally follow the organized and expectable pattern of producing visual information at a constant interval (which can be both advantageous and disadvantageous depending on the situation), changes are needed to provide a batch of features to the measurement model. The proposed approach is that of accumulation of the asynchronous features over a period of time, in order to simulate frames being received. We call these accumulation of event features over time *pseudo-frames*, and are usually of 20 ms or less, to take advantage of the speed of events.

Three strategies are proposed for the creation of the *pseudo-frames*, in particular 1) fixed time interval integration, 2) fixed number of features update, and 3) hybrid approach, combining both ideas. Though the hybrid approach should perform better, we found that a fixed interval integration was much more easily manageable (as it translates quite naturally to a conventional camera producing features, albeit at a much faster rate), we used the fixed interval integration when validating the approach.

IV. Closed Loop Integration of Sensor and Pose Filter on Event Cameras

Two problems identified when testing the previous approach were as follows: the number of features is limited; and sometimes features are lost, only to be found a few moments later, but with a different ID (which is not necessarily bad, but then the filter treats this feature as a new one, and all previous sightings are discarded). The first problem is of difficult resolution without major changes in the approach, as it is based on corner detection, which are common in images, but still limited. The second problem, however, implies improving the tracking of features so that they are kept alive for longer. As such, we set out to improve EKLT tracking performance.

A. Motivation for Improved Approach

Revisiting EKLT, at first glance it may seem like a simple implementation of KLT, where the matching patches are obtained from frames and events (as opposed to the normal strategy of both patches coming from frames), and, to a certain extent, this is true. But there is more to be said about the generation of these templates.

Looking back at Fig. 4, we can see that the x-wise and y-wise image gradients are generated, and (after being subject to a warp) are merged by means of a dot product with the flow angle. This parameter of the flow angle v may look innocent, but it is critical in the generation of the template, and has proven to be one of the main reasons for tracking loss. Another parameter to be optimized is the initial location of the feature, which corresponds to the expected position of the feature, and is fed into the optimizer as a starting value. This parameter is also important, but since features are updated very frequently (sometimes around 1 ms, in very fast paced scenes) it is not as critical as the flow angle (though it also plays an important role, in particular when a feature is lost (Section IV-D)).

With this in mind, we propose an approach where the current estimated pose is fed back into EKLT to help with the tracking of features. This approach creates a sort of ”closed loop”, where position from FUSION is fed into EKLT, which then provides information for FUSION, as shown in Fig. 7. To the best of our knowledge, though not novel, this is an uncommon and innovative approach where there is some sort of additional processing at the sensor level, with information external to said sensor.

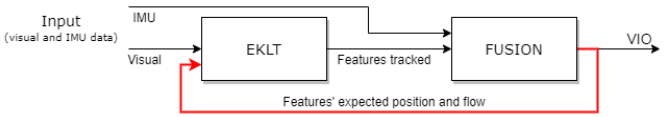


Fig. 7. Global view of our proposed closed loop integration suggestion, where the pose estimation is fed back into EKLT to help with feature detection.

B. Ego Motion and Optical Flow

In Section II-D1, it is already derived how the movement of the features being captured by the camera are influenced by its motion, in particular (according to [26]), following (8).

This way, we can predict the evolution of the position of the feature over time, meaning it is possible to estimate the flow angle. However, it is very important to take into account that these predictions work for the immediate proximity of the feature, i.e., this assumption only works for small timestep. If the timestep is bigger, the predicted movement will not match the real position of the feature.

C. Features Tracking complemented by the Pose Filter State

Our objective is to improve the feature tracking from EKLT by feeding it information from the current pose estimation, which, in turn, will benefit from a greater number of features. As already explained, the most critical (or, at least, the component that contributes most from loss of features) is the generation of the template to match from frames, which depends on two main components: the initial position location and the flow angle. Their importance (and relative importance) have already been mentioned (Section IV-A). We believe the pose estimator can (either directly or indirectly) help the tracker with regards to these two components.

Starting with the flow angle, we propose the use of (9) to determine it by means of

$$v = \text{atan2}\left(\frac{\dot{y}}{\dot{x}}\right) \tag{18}$$

where $\dot{x}$ and $\dot{y}$ are given by (9). We reiterate the importance of a small timestep, and synchronisation between the current estimate and EKLT, as disparities become more detrimental than beneficial. To tackle this problem, all timesteps are kept to a minimum, and are usually of about 1 ms. This ensures the assumptions for (9) are valid (according to our testing).

In terms of the estimations being used for motion, the angular velocity comes directly from the last measurement of the gyroscope (or some sort of mean or median of the last measurements, if readings are too contaminated by noise). The linear velocity, on the other hand, comes from the state, that estimates the filter velocity (along with system rotation and

position, landmark position, and sensor bias), converted to the camera frame.

Moving on to the initial feature position, our proposed filter structure keeps the estimated 3D position of landmarks in the state, which can be projected into camera space to obtain the estimated position of the features by means of the projection equation (17). This way, the tracker benefits from having an additional information of the features being tracked by adding the depth factor. In effect, by "helping" the tracker with the starting values, what is being done is placing the initial estimate inside the region of convergence, and closer to the global minimum, as the rest of the matching is still performed by the optimizer, that tries to match both templates, and estimate the current position of the feature (and its flow) in the process.

From our experiments, the importance of the flow angle is much greater than the initial feature location. This is because the neighbourhood of the feature is really small (the displacement between initial location and final location is typically around 2-3 pixels diagonally), whereas flow angle could have deviated significantly from the last optimization (imagine a rotation in the z axis of the camera, where features on the borders of the camera move faster than those on the inside), and influences the next matching negatively.

D. Set of Backup Features

In our case, the pose estimator is capable of storing features and landmarks over time (in effect, creating a sort of map, as per a SLAM formulation) and keeping these features (when they are lost and discarded by EKLT) in a sort of *zombie* or *dormant* state. Since we can project their predicted location onto camera space by means of (17), it is possible to awaken these features when they enter the FOV again, for example. This means that these features (that would eventually be detected again, but would be given a new ID, which would not benefit from using past sightings of these features), are able to be reidentified as used in the filter with the same ID.

This method also allows for bigger jumps in feature tracking, as the initial feature position can be set to a place that is far away from the previous estimated position (imagine a situation where this landmark becomes occluded, and therefore disappears, but is kept in memory by the pose estimator; when the landmark is no longer occluded, since the pose estimation kept running, the expected position based on current pose can be used). An interesting side effect of such an approach is that it is possible to rank features based on how distinct and/or observable they are, as features that are detected more, have more entries in the table, and therefore are probably the ones we want to use, as they are more robust.

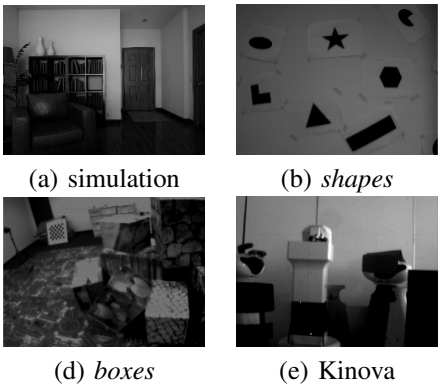
V. Experiments and Results

In order to validate the proposed approaches, simulation and real data was used. Simulated data was generated using ESIM ([27]), an event camera simulator, and real data used datasets available online that use real event cameras [2], as well as recordings we performed using the Kinova robot arm to generate trajectories. Fig. 8 shows samples from each recording. Given the group's interest in the visual and vestibular system, the focus of the work (and therefore, of the results presented), was directed towards rotational movements (though the trajectories themselves may (and do sometimes) have translational components).

[2]http://rpg.ifi.uzh.ch/davis_data.html

Movement	Mean error [deg]	Max error [deg]	RMSE [deg]
Rotation x axis	-30.49	-58.04	34.48
Rotation y axis	-8.14	-45.70	17.35
Rotation z axis	-7.03	-32.46	9.66

TABLE I
Open loop on *shapes* scene



(a) simulation (b) *shapes*

(d) *boxes* (e) Kinova

Fig. 8. Samples from scenes.

A. Experiment 1, Integrated Experiment with a DAVIS Camera Dataset

To test the proposed approaches, we tested the performance on the datasets available online, which consist of a series of movements of the camera, by hand, on multiple scenes. In particular, we tested the approach on the *shapes* and *boxes* datasets, as the former has clear contrast between background and shapes, and the latter has a much more textured environment.

Open Loop Approach: We start with the first proposed approach. The results running this approach are presented in Table I. We have decided to isolate each axis estimation for the sake of a less cluttered analysis, as well as to interpret the evolution of each axis independently. It is possible to observe that the obtained results are not exactly satisfactory. First, there is an obvious drift in the x axis that was not able to be compensated. We believe this drift is because of an uncompensated bias in the gyroscope, as this axis more easily loses features by moving out of the FOV, which means that the local map may itself drift overtime, and not correct sensor bias.

After, we tested the proposed approaches on the *boxes* scene. The results running this approach are presented in Table II. This experiment still produces some clear deviations on the true values. However, we believe these results are slightly better than the ones presented previously, as not only are the errors smaller (with the exception of the x axis rotation), the overall profile of the estimation more closely follows the true values, which is positive.

Movement	Mean error [deg]	Max error [deg]	RMSE [deg]
Rotation x axis	-20.40	-50.10	30.88
Rotation y axis	3.63	-10.25	6.13
Rotation z axis	5.96	-12.27	8.80

TABLE II
OPEN LOOP ON *boxes* SCENE

Movement	Mean error [deg]	Max error [deg]	RMSE [deg]
Rotation x axis	4.35	23.51	6.66
Rotation y axis	-1.53	-10.77	5.11
Rotation z axis	1.05	14.90	5.39

TABLE III
CLOSED LOOP ON *boxes* SCENE

Closed Loop Approach: We tested the closed loop approach on the *boxes* scenario. The results running this approach are presented in Table III. It is possible to see that the proposed approach does help with tracking, as the results show an improvement over the previous approach.

B. Experiment 2, Integrated Experiment on Simulation

To validate the approaches proposed, we generated a synthetic dataset based on ESIM, simulating a DAVIS240 event camera, with access to both frames and events simultaneously. The images produced have some distortion parameters, and some noise, but have no motion blur when simulating conventional frames, which would (in principle) give the upper hand to event cameras.

Open Loop Approach: In this case, we also performed a comparison with a conventional camera approach. The results are presented in Table IV. We observe that the approach with events outperforms the approach using frames, both in terms of the mean error, and the max error. We believe this has to do with two factors: the more frequent updates that events allow mean that the filter can be updated with visual information more frequently; the features, being tracked with events, are tracked better (which can be debatable as it has their own problems, as lack of good descriptors, and being limited to corner features). We repeated this same setup, in the same scene, but with other motions, in particular rotations in the other axes. The results are summarised in Table V.

Overall, the results are very uplifting, and validate the proposed approach, even if with a simulator, and outperform the conventional camera approach in all cases (even though sometimes a change in the settings for the filter or the tracker is needed).

Setup	Mean error [deg]	Max error [deg]
Image + IMU	1.05	4.41
Image + Events + IMU	0.85	3.05

TABLE IV
CONVENTIONAL VS EVENT-BASED APPROACHES

Movement	Mean error [deg]	Max error [deg]
Rotation x axis 18 deg	1.66	5.04
Rotation y axis 18 deg	0.65	2.43
Rotation z axis 18 deg	0.85	3.05

TABLE V
OPEN LOOP ON SIMULATION

Movement	Mean error [deg]	Max error [deg]
Rotation x axis 18 deg	1.73	5.89
Rotation y axis 18 deg	0.049	0.12
Rotation z axis 18 deg	0.80	4.23

TABLE VI
CLOSED LOOP ON SIMULATION

Closed Loop Approach: After testing the first proposed approach, we present the result for the second proposed approach (closed loop) using the same scene and setup. The results are presented in Table VI. These results seem to corroborate that the closed loop approach does, in fact, improve feature tracking, which, in turn, improves the estimation quality, as these results seem promising (even though simulated data was used as input, which does not necessarily transfer to the real system effortlessly).

Nevertheless, some limitations of this approach also presented themselves, the most relevant of which is the lower robustness of the filter. The closed loop presents itself as a double edged-sword, in the sense that good pose estimations lead to better tracking, which leads to better estimation, but a bad starting estimation is very detrimental to the tracker, which loses features very quickly. This can be shown for the case of x and y axis rotations. The former had some biases that hurt the performance of the filter, resulting in a rotation that was worse than the first proposed approach. However, for the former, some parameters must have been optimal, as the error is very low.

C. Experiment 3, using the DVS Camera Mounted on the Kinova Arm

In this experiment, multiple rotation-focused movements were performed by means of a Kinova robot arm, with the hope of mimicking the eye saccadic movement, and being able to track it along time. This mimicking was mostly in terms of the velocity and acceleration profiles, not necessarily what is humanly possible (we consider torsional movements, which do not occur in the eyes, for example). Since the DVS camera is the camera considered for this experiment, either frames or events may be recorded at each time (exclusive or). The data recording encompasses two parts (i) image frames when the camera is still, and (ii) events when the camera is moving. The commutation from frames to events is not automatic, is placed in the script of the data acquisition. IMU is always recording.

After feeding this recording into EKLT it was verified that frames based features are effectively tracked, however the event based features are lost between tens to hundreds of milliseconds after detection, resulting in a drift in estimation.

In a second experiment, we have calibrated the IMU and initialized the pose estimation method with estimated biases of the IMU. Under these conditions, though not perfect, the estimated rotation much closely follows the real value,

In a third experiment, we took an hybrid approach leveraging the start of the recording, where the camera stays static until around 10 s, and therefore IMU output is mostly

noise (and gravity). As such, we start by running the filter considering frames (as if we were using a conventional setup), in order to estimate bias, obtaining a RMSE of 0.3635 deg on the z axis, which is actually quite interesting, though results mostly from a good estimation of the biases from the initial estimation from frames.

VI. Conclusion and Future Work

A. Conclusion

In this work we set out to develop a system for pose estimation that was based around event cameras, a novel type of visual sensor that is yet to be fully explored. In the end, two approaches were developed to tackle this problem. A first, which combined an Unscented Kalman Filter developed around a Lie group structure, with a feature detector and tracker based around events, which performed well under simulated environments, but ultimately under-performed in the real system. The second approach seemed even more promising based on simulations results, but its usefulness in real environments is debatable, as the initialisation of the filter is much more critical in this method (as poor estimations lead to poor tracking, in turn leading to poorer estimations). Nevertheless, when said initialisation was done more carefully, the filter performed acceptably.

Though far from perfect (in fact, both fell short of the current state of the art), both introduced new concepts that can be further improved, and not only produced acceptable results, but served as a basis to understand the current status of event cameras, their limitations and advantages. In a way, it served as a learning experience not only for myself, but for the group as well.

B. Future Work

We believe the ideas presented in this thesis are sound, and multiple possible paths can be taken to follow-up on this work. We believe the main shortcoming of the suggested approaches lies on the features being tracked, and the way they are fed into the filter. As such, a possible path is to try and "robustify" said features, such as using a RANSAC approach, or some other way to select good features. However, this is easier said than done, as we identified problem pertaining to the low number of features. Reducing this number even more may be dangerous.

Other solutions based on the methods proposed may include running parallel estimation of both approaches proposed, to see when it possible to reliable switch to the closed loop method, in effect taking advantage of the first moments to initialise the system correctly, and switch only when estimation is good and ensuring the closed loop will run correctly. Or even to try and predict the movement in real time, and switch between models, i.e., have translation-only, rotation-only, and combined models that can be selected based on the current predicted trajectory.

Lastly, as machine learning approaches seem to be ubiquitous, considering such an approach might be interesting. However, depending on the type of data being used (for example, if direct events are to be used), work on Spiking Neural Networks (SNN) may be needed, as the asynchronous nature of events is best captured by the asynchronous nature of SNN.

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128× 128 120 db 15μ s latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[2] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-dof pose tracking for high-speed maneuvers," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2761–2768, IEEE, 2014.

[3] H. Akolkar, S. Ieng, and R. Benosman, "Real-time high speed motion prediction using fast aperture-robust event-driven visual flow," *arXiv preprint arXiv:1811.11135*, 2018.

[4] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[5] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019.

[6] D. Weikersdorfer and J. Conradt, "Event-based particle filtering for robot self-localization," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 866–870, IEEE, 2012.

[7] C. Reinbacher, G. Munda, and T. Pock, "Real-time panoramic tracking for event cameras," in *2017 IEEE International Conference on Computational Photography (ICCP)*, pp. 1–9, IEEE, 2017.

[8] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[9] A. Zihao Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5391–5399, 2017.

[10] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4465–4470, IEEE, 2017.

[11] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[12] M. Martins, "Determining the orientation of a rgb camera embedded on an artificial eye," 2019.

[13] L. Luo, *Principles of neurobiology*. Garland Science, 2015.

[14] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Eklt: Asynchronous photometric feature tracking using events and frames," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 601–618, 2020.

[15] C. G. Harris, M. Stephens, *et al.*, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.

[16] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," 2017.

[17] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, pp. 91–106, 2015.

[18] B. Siciliano and O. Khatib, *Springer handbook of robotics*. springer, 2016.

[19] B. Horn, B. Klaus, and P. Horn, *Robot vision*. MIT press, 1986.

[20] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[21] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[22] M. Brossard, S. Bonnabel, and A. Barrau, "Unscented kalman filtering on lie groups for fusion of imu and monocular vision," in *Proc. Int. Conf. Robot. Automat.(ICRA)*, pp. 1–9, 2017.

[23] A. Barrau and S. Bonnabel, "An ekf-slam algorithm with consistency properties," *arXiv preprint arXiv:1510.06263*, 2015.

[24] S. Bonnabel, "Symmetries in observer design: Review of some recent results and applications to ekf-based slam," *Robot Motion and Control 2011*, pp. 3–15, 2012.

[25] G. Loianno, M. Watterson, and V. Kumar, "Visual inertial odometry for quadrotors on se (3)," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1544–1551, IEEE, 2016.

[26] D. J. Heeger and A. D. Jepson, "Subspace methods for recovering rigid motion i: Algorithm and implementation," *International Journal of Computer Vision*, vol. 7, no. 2, pp. 95–117, 1992.

[27] H. Rebecq, D. Gehrig, and D. Scaramuzza, "Esim: an open event camera simulator," in *Conference on Robot Learning*, pp. 969–982, 2018.