UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÓ

Pla de recerca:

Event based SLAM

William Oswaldo Chamorro Hernández

Directors:

Juan Andrade Cetto

Joan Solà Ortega

Setembre 2019

Contents

1 Introduction and motivation

One of the main goal in mobile robotics research is to develop autonomous vehicles capable of navigating at several speed levels, that work indoors and outdoors, precisely localize in GPS-denied environments -such as cities with tall buildings or through underpasses- and, coming in and out of very different lighting conditions. To deal with most of these key elements, visual odometry comes as an inexpensive alternative to other odometry systems based on laser, inertial or wheel sensors, among others [1].

The technological growth of digital cameras has allowed to address the most complex problems in visual odometry and to face the most challenging conditions in the environments [2]. Those efforts resulted in the emergence of applications for instance visual SLAM [3], 3D mapping [4], or visual guidance systems [5], among others; and, whose development is closely linked to the upgrade of visual technologies. In spite of new technologies designed for visual applications and fast processing capabilities in computers nowadays, the most common issues with the visual odometry systems are the limited frame rate and bandwidth of conventional cameras (e.g.30 Hz for GoPro Hero 3 [6]) [7] [8]. These limitations compromise algorithms performance in uncontrolled environments with natural lighting, and difficult the information processing for fast movements in the scene where the images start to blur.

The aforementioned problems with classic cameras can be overcome by using Event-based cameras. These are bio-inspired vision sensors whose output is an asynchronous stream of pixel address-events that respond to brightness changes in the scene at a microsecond resolution [9]. The pixels of the event camera are completely independent of each other, and since no frame or image is produced, they are transmitted as soon as a logarithmic change of intensity -greater or equal to a predefined threshold- is detected. These changes on brightness produce *events*; usually with a very low latency (about 10μs [10]). The advantages of using an event camera are that it is extremely fast -events have a timestamp of a microsecond resolution-, if there are no consistent brightness changes, the events are not produced; as a result, this camera avoids producing redundant information, and because of its constructive characteristics it exhibits high dynamic ranges (e.g., 120dB for the Davis 240C model [11]). This means that the cameras become suitable for high-speed and/or under the most demanding light conditions without suffering motion blur.

In order to exploit the advantages of the event cameras and apply them to the mobile robotics field, our research will concentrate on solving the localization and mapping problem by using the asynchronous information provided by these cameras. To this end, we will accurate track the six degrees of freedom (6DoF) of the camera motion -for high-speed applications-, and recover the 3D scene information. The aim is to develop an integrated event based SLAM device that works with two parallel threads of Tracking and Mapping (PTAM [12]) in real time and for highly dynamic and challenging lightning conditions. This approach is unconventional and distinguishes itself from conventional camera algorithms due to its non-dependence of image intensity values and in the asynchronous nature of events [13]; hence, novel algorithms for depth estimation, spatiotemporal feature extraction, feature aggregation and tracking need to be studied. The fast responses of the event camera allow measurements in the range of the MHz; thus, depending of the solution implemented, the estimations, computations and overall results for tracking and mapping can be performed with no delay.

Related to the pose estimation problem, the scenario has been dominated by Bayesian filters [10] that estimate the 3D position and orientation, where the events can be used to perform a fast estimation update. Depending on the complexity or the techniques used, these algorithms may be computationally demanding and typically requires dedicated hardware [14] (a GPU). Thus, fast data association techniques for improving the response times of the estimation algorithm need to

be taken into account, where all the incoming data will be filtered as to rapidly remove outliers, rendering only the most relevant events. The aim is to come up with fast and inexpensive solutions for robotic applications that responses in real time in standard computers.

The event-based SLAM problem in its most general setting (6DoF and 3D scenes) is being currently addressed in scenarios with large motion demands and increased complexity, especially applied to unmanned aerial vehicles (UAVs) and humanoids for highly dynamic maneuvering. In this thesis we will explore the perception issues associated to an event based camera for tracking and mapping and its applications.

2 Objectives

The main goal of this doctoral thesis is the study and development of new techniques to accurate obtain the spatial trajectory of an event based camera, and its localization inside the scene; by using the asynchronous sequence of events produced when the camera moves. These techniques will be developed for real time operation and aimed to work on standard computers with ROS environment for future robotic applications. The accuracy and performance will be validated against ground truth trajectories recovered with an absolute positioning system (Optitrack), and also in natural scenes with different complexities and light conditions. These objectives are also within the scope of the EB-SLAM project of the Spanish State Research Agency (DPI2017-89564-P).

The tasks that we will address include camera pose estimation on 3D (6DoF) as an event-only approach; techniques for fast data association and outlier rejection to process all the incoming events to produce an estimation with a time stamp in the order of the microseconds, aimed for fast movements; smoothing methods to fit a continuous time trajectory (such as cubic splines [15]) given a raw estimation.

In order to provide a robust, accurate and smooth trajectory in natural scenes, we will also explore sensor fusion approaches for the event camera. As a matter of fact, the DAVIS240C model -which is available in our laboratory- incorporates a 1KHz inertial sensor that can be exploited for this task [16]. Furthermore, the difficulties of using an inertial sensor such as obtaining a proper initialization and accurately integrating angular velocities and linear accelerations will be addressed [17].

We will come up with monocular space discretization methods to perform a semi-dense 3D reconstruction of the scene based on the camera spatio-temporal information and events accumulation. These methods should provide a way of creating a map with the recovered 3D points and store the information in a spatial grid made of *Voxels*. Natural edges in the scene will be initially used to produce computationally efficient mapping algorithms -as it was reported in [18]-.

For the monocular depth estimation and feature extraction, we will study proper landmark parametrization forms for fast asynchronous data association. Our starting point on this task will be a wire-frame landmark representation (because of the natural edge highlighting by the events) and Random Consensus and Sampling (RANSAC) techniques for spatial feature extraction obtained from a projective sampling grid. Another important aspect to be considered is the map optimization. During this task all the extracted 3D features should be merged on a map located at a larger global reference and properly updated to have non-repetitive reliable landmarks. As the camera moves, the current view may contain occluded landmarks that should not be tracked at this time. By departing from the optimized global map, we will look for strategies for selecting landmarks that will be temporally seen in the screen to avoid estimation errors due to the back projection of occluded landmarks onto the current view. Since tracking and mapping are in a closed loop, we will consider bootstrapping methods to launch the tracker before having a built map, and they will evolve as the problem complexity increases.

The localization and mapping tasks to achieve the main objective of this work are summarized as follows:

- Event based tracking techniques for highly dynamic environments and challenging lighting conditions.
- Monocular depth estimation and feature detection based on a proper landmark parametrization.
- Map merging optimization for feature aggregation in a global reference.
- Temporal landmark selection from the optimized map for tracking according to the current camera view to handle occluded landmarks.

3 Expected contributions

This PhD theses seeks to contribute with techniques for localization and mapping that run on real time in standard computers; which will serve for future applications like humanoid or UAV self localization in environments without GPS readings, undergoing high dynamic movements and, under conditions of poor or severe illumination changes. This event based SLAM approach is expected to work with natural scenes indoor and outdoor as the natural light does not affect the camera performance and since pixel images will no be used.

We aim to propose an approach able to report an accurate pose estimation as soon as an event is recorded by the camera. This implies that we need to exploit the sparse structure, motion models and mathematical formulation of the estimation thread, by means of nonlinear optimization techniques to solve efficiently the set of kinematic constraints at a speed in the microsecond scale. We will contribute among other things, with Lie group formulations for the SLAM problem that allow simplified computations such as Jacobians calculations.

As the camera moves all the detected features are merged on a global map which increases over time. In this regard, we will develop optimal techniques to select the most relevant features from the global map and reproject over the event image. This will improve the tracking efficiency and help to identify occluded components in the scene that should not be consider for tracking. To date, we have proposed an approach for fast tracking and mapping of a wire-framed hand-created scene, whose results provide a pose estimation in less than one microsecond with an accuracy comparable with that of an optitrack measurement. Based on these tracking results, we performed a 3D recovery approach to create a scene point cloud from which the segment landmarks are extracted. It runs on a parallel thread and has a response time in the order of the microseconds. This firsts results will be submitted shortly publication and the approach is still in further development. These initial contributions are focused on monocular systems.

All the algorithms, techniques and methodologies will be implemented with open source platforms. The libraries to solve the event based SLAM problem will be written in C++ and, aimed to work with ROS in any robotic platform that requires a visual self localization system.

4 State of the art

Event cameras have become commercially available since 2008 although, they are just starting to come out from the prototype stage and become commercial products. Their prices are not yet competitive to other sensors. There is a big commercial interest in exploiting these novel vision sensors for mobile robotics, augmented and virtual reality (AR/VR), and video game applications [10]. On this regard, dedicated companies like iniVation, Samsung, or Prophesee, among others,

have released event-cameras based in the dynamic vision sensor (DVS) [11]; Nowadays, one of the most common models is the DAVIS camera (by iniVation) that combines the DVS with an active pixel sensor (APS) at the pixel level. It allows simultaneous output of asynchronous events and synchronous frames [9]. This camera model has a flexible structure compatible with Linux. In this sense, in order to motivate research on new algorithms for event-based visual applications, the Perception Lab of The ETH Zurich has released a C++ driver for the DAVIS and DVS sensors that uses a "*libcaer*" library to handle the internal firmware, and several datasets that include ground truth and inertial measurements of natural and synthetic scenes. Our thesis will make use of this driver, which allows to handle event messages, images and camera information in the ROS environment, in addition to deal with communication delays between the sensor and the computer. The package also provides a calibration tool for both intrinsic and stereo calibration. [19] [11] [20]

SLAM in its more general setting is a self localization problem addressed on scenarios with different complexities, methodologies and perception forms. The literature is dominated by methods that address the localization subproblem first as it has a lower complexity than the mapping [3]. A review of the state of the art on the tackled problems about event based SLAM is provided. This section is divided in two parts related to tracking and mapping respectively.

4.1 Event based tracking

The tracking problem lies on estimating data association between a series of events and known features in the scene, to finding an optimal pose of the camera -three dimensional translation and rotation- by minimizing the reprojection error between events and features. The first approach to this problem using event-cameras was focused on a 2D robot estimation, where the well-known "Condensation Particle Filter" was used. Originally this filter was designed to track curves [21], which in this case is used to associate the closest events to an artificial mark and to update the 2D robot position and orientation while the moving object is seen from the ceiling with a static camera. This tracker was tested on a scenario composed by black and white circular features, and then extended to a $\mathbf{SE}(2)$ SLAM where the map of the planar scene was reconstructed [22].

The gray scale image provided by the DAVIS sensor can also be used to perform event - pixel association. The authors of [23] refer to this method as an "instantaneous pixel-event map" for tracking the relative motion between two successive low-frequency frames and a temporal window of events. The core of this tracker is a Bayesian filter, which updates the camera pose by measuring the small displacements between the current event and the previous frame when the camera moves [23]. A similar approach was introduced in [24] to solve the SLAM problem where the DVS sensor was a support for a RGB camera in challenging light conditions. The main problem with these approaches, is the bottleneck caused by the image frame-rate and the low resolution of the DAVIS pixel image.

More related to our approach, is the geometrical data association between the event and a line-type landmark performed in [19]. This work tracked a planar wired-frame scene model based on a flat black and white squared panel. The pose estimation was achieved by minimizing; via least squares; the distances between the reprojection of each line and the events belonging to it. The result was a 6DoF pose tracker, capable of detecting fast quadrotor maneuvers and high-speed rotations. The authors reported accurate estimations at speed rates, unable to be detected with standard cameras due to image blurring. This approach was extended to a continuous trajectory tracking taking into account cubic B-splines to provide smooth estimations [25]. Our approach will be based on line-shaped 3D landmarks extracted from the scene instead of having previously hand created features only, then use them to build a wire-frame map that will be updated and improved as soon as new landmarks are detected.

The literature shows that Bayesian filters for 6DoF pose estimation are the dominant framework mainly for its capabilities on information fusion, which make possible system updating with

measurements that may come from different sources and on handling asynchronous data. For example, [26] proposes a system with three interleaved probabilistic filters that perform pose tracking, depth and log intensity estimation as part of a SLAM system. Its aim is to find the system state which best predicts a log intensity change consistent with the event just received. The system state for the filter is a mapped member of the Lie Group **SE**(3) to set a 3D rigid body transformation $T_i = (R_i|t_i)$ where $t_i \in \mathbb{R}^3$ and $R_i \in \mathbf{SO}(3)$. The previous event information, that lies in the same pixel of a reconstructed image-like log intensity keyframe with inverse depth, is used to update the camera pose. The results indicate that this method is computationally intensive, requiring a GPU for real-time operation, which is not the aim of this thesis, however the mathematical approach for the tracking method can run faster with a different update method.

Another probabilistic approach presented on [27] [28] implements a Bayesian filter to track the camera pose by updating the measurements with a likelihood function model that takes into account both, the event generation process and the presence of noise and outliers. It differs from the previously stated method in the consideration for updating the system and for the state vector, which is in this case $x_i = (T_i, T_j, T_k, C_{th}, \pi_m, \sigma_m^2)$, where T_i is the current pose, T_i and T_j are two poses along the trajectory to interpolate the pose of the last event, C_{th} is the threshold contrast and π_m, σ_m^2 are inlier parameters of the sensor model. In the same line [29] perform an edge-map alignment by minimizing the photometric error on a set of selected pixels whose 3D correspondences in the scene have already been established with an inverse compositional Lucas-Kanade method. The required map is estimated in parallel by using the event information. In a related approach, the visual tracker uses the output of an inertial sensor and event information to track the projections of sets of landmarks [30]. The objective is to improve the tracking robustness by combining both sensors, mainly via a Kalman Filter. The tracked features are extracted from the scene by analyzing their optical flow.

4.2 Event based mapping and 3D reconstruction

Depth estimation and 3D reconstruction from events have been tackled from monocular and stereo perspectives. In monocular mapping, a semi-dense 3D scene (i.e. a 3D edge map) can be recovered by integrating information from the events of a moving camera over a temporal window. The main issue is the estimation of the temporal correlation between events across multiple image planes required to estimate the depth from each pixel.

Similar to the tracking problem, the depth can be estimated with a Bayesian filter as is proposed in [26]. As it was stated before, this method is computationally expensive. In contrast, [18] proposes a space-sweep method that leverages the sparsity of the event stream to perform 3D reconstruction without any explicit data association or recovery of the intensity images. This approach maps a batch of events associated to a known pose onto a projective 3D grid made of n depth planes via planar homography. The method was initially stated on [31], and it provides a computationally efficient way to discretize the scene in projective voxels with a given size. Our mapping thread will made use of this approach as a starting point to create a point cloud that will be properly filtered to extract the line-type landmarks as the camera moves. The solution presented on [18] was tested in a PTAM implementation [29], where one thread performs space sweep discretization and the other camera tracking by aligning the 2D representation of the spatial grid named Disparity Space Image (DSI) with the event frame.

Other methods make use of depth sensors to reduce the complexity of the problem like in [32] where, a pulsed line laser and an event camera are used within an adaptive temporal filter for fast 3D reconstruction. In [33] a DAVIS sensor and a depth sensor are fused with a Bayesian filter to perform 3D SLAM, this is an extension to the 2D algorithm proposed in [22]. There are approaches focused on active 3D, whose working principle is emitting light onto the scene and measuring reflection with event cameras [34] [35].

In stereo mapping, the 3D reconstruction uses events on a very short time from two or more synchronized cameras that are rigidly attached. In contrast to monocular methods, stereo approaches usually solve the event correspondence problem across image planes (i.e. epipolar matching) and then triangulate the location of the 3D point. It requires a temporal window of accumulated events to find significant visual correlations between the left and right cameras [36] [37] [38]. This approach requires additional hardware to attach two independent cameras - currently there is no stereo models available in the market-, and software to keep both cameras synchronized.

5 Preliminary work

The report of the current state of our research in the two main topics addressed on this thesis is presented as follows.

5.1 Wired-frame event based tracking

We propose a fast event only tracking algorithm for line based landmarks. The aim is to estimate 6DoF camera pose by using an error state EKF, that works with a constant velocity model. As an initial approach we track an a priori stored map represented by straight line segments. We present a very fast algorithm to match events with reprojected line segments. In highly dynamic situations, events arrive in the order of millions per second asynchronously; thus, an optimal treatment of sparse matrices is required to deal with event updates in real time. Furthermore, the EKF is implemented as a composite Lie group to further simplify computations [39]. Our implementation uses the small header-only library for Lie theory "`manif`" [40]. To the best of our knowledge, this is the first time that real-time full 6 DoF event-based tracking is implemented using Lie group theory. The results of this section are about to be published.

5.1.1 Bootstrapping

With the aim of developing and testing the tracking efficiency, we create a wired framed map based on the line segments belonging to four squares lying on the surfaces of a cube and use the events that naturally respond to edges. This map becomes a starting point and a geometric reference for our 3D map. We are focused on extend the use of line features to a more natural scene.

To launch the tracker we need to know an initial guess of the camera position, which is estimated with is a simple approach that makes use of the pixel information of the DAVIS sensor. The line segments are parameterized using their endpoints and expressed in coordinates of the cube base frame, which has the origin on its vertex. FAST [41] endpoints are detected in the pixel camera and matched with these 3D coordinates, then the PnP algorithm [42] is used to estimate the initial camera pose. After that this image is not longer required. The aim of this research is also to propose a more efficient bootstrapping method that does not dependent on this static reference.

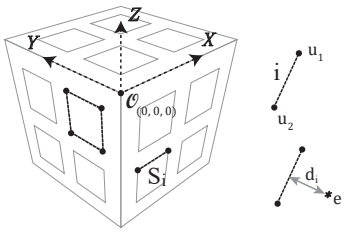

Figure 1: Spatial reference model and line parametrization for initial bootstraping

5.1.2 EKF: Lie Group Formulation

Motion Model of Constant Velocity: Our tracker is formulated as an error-state Kalman Filter operating on a Lie group. The system state $\mathbf{x}$ is represented as a composition of non-interactive Lie groups [39]. These are the camera position $\mathbf{r}$, and its orientation $\mathbf{R}$, linear velocity $\mathbf{v}$ and angular velocity $\boldsymbol{\omega}$. Orientation is expressed in $\mathbf{SO}(3)$ with angular velocity on its tangent space $so(3)$. Position and linear velocity are trivial vector additive groups inside Lie theory. The error state $\delta\mathbf{x}$ lies in the tangent space of the state. It is modeled as a Gaussian variable with mean $\bar{\delta\mathbf{x}}$ and covariance $\mathbf{P}$

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \in \mathbb{R}^3 \\ \mathbf{v} \in \mathbb{R}^3 \\ \mathbf{R} \in \mathbf{SO}(3) \\ \boldsymbol{\omega} \in \mathcal{T}\mathbf{SO}(3) \end{bmatrix} \quad \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{r} \\ \delta\mathbf{v} \\ \delta\boldsymbol{\theta} \\ \delta\boldsymbol{\omega} \end{bmatrix} \sim \mathcal{N}(\bar{\delta\mathbf{x}}, \mathbf{P}) \tag{1}$$

The camera motion is modeled as a constant velocity of the form

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \Delta t_k, \mathbf{n}_k) \,,$$

where Δt_k is the time difference between two consecutive events, which is time-variant due to the asynchronous nature of the events; and $\mathbf{n}_k = (\mathbf{v}_{\mathbf{n}k}, \boldsymbol{\omega}_{\mathbf{n}k})$ is the system Gaussian perturbation affecting the velocities. The state transition at time k is modeled as follows:

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \mathbf{v}_k \Delta t_k \tag{2}$$
$$\mathbf{v}_k = \mathbf{v}_{k-1} + \mathbf{v}_{\mathbf{n}k} \tag{3}$$
$$\mathbf{R}_k = \mathbf{R}_{k-1} \oplus \boldsymbol{\omega}_k \Delta t_k \tag{4}$$
$$\boldsymbol{\omega}_k = \boldsymbol{\omega}_{k-1} + \boldsymbol{\omega}_{\mathbf{n}k} \tag{5}$$
$$\tag{6}$$

The operator $\oplus$ represents the right plus operation for the Lie $\mathbf{SO}(3)$, given by

$$\mathbf{R} \oplus \boldsymbol{\theta} \triangleq \mathbf{R}\,\mathrm{Exp}(\boldsymbol{\theta}) \tag{7}$$

with $\mathrm{Exp}(\boldsymbol{\theta})$ the exponential map given by the Rodrigues formula [39].

Kalman Prediction: In the prediction stage the motion covariance propagation is

$$\mathbf{P}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \quad \in \mathbb{R}^{12\times 12} \,, \tag{8}$$

where $\mathbf{Q}$ is the perturbation matrix, given by

$$\mathbf{Q} = diag(\mathbf{0}, \sigma^2_{\mathbf{v_n}}\mathbf{I}, \mathbf{0}, \sigma^2_{\boldsymbol{\omega}_\mathrm{n}}\mathbf{I})\Delta t_k \quad \in \mathbb{R}^{12\times 12} \tag{9}$$

and $\mathbf{F}$ is the Jacobian of f with respect to $\mathbf{x}$,

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & \mathbf{I}\Delta t_k & 0 & 0 \\ 0 & \mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{J}_\mathbf{R} & \mathbf{J}_{\boldsymbol{\omega}} \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \quad \in \mathbb{R}^{12\times 12} \tag{10}$$

The derivation rules for the Lie Group $\mathbf{SO}(3)$, summarized in [39], indicate how to calculate the Jacobians for the rotation, using elementary Jacobian blocks and the chain rule. From the sparse matrix in (10) the Jacobians are

$$\mathbf{J}_\mathbf{R} \triangleq \mathbf{J}_\mathbf{R}^{\mathbf{R}\,\mathrm{Exp}(\boldsymbol{\omega}\Delta t_k)} = \mathbf{Ad}^{-1}_{\mathrm{Exp}(\boldsymbol{\omega}\Delta t_k)} = \mathrm{Exp}(\boldsymbol{\omega}\Delta t_k)^\top \tag{11}$$
$$\begin{aligned} \mathbf{J}_{\boldsymbol{\omega}} \triangleq \mathbf{J}_{\boldsymbol{\omega}}^{\mathbf{R}\,\mathrm{Exp}(\boldsymbol{\omega}\Delta t_k)} &= \mathbf{J}^{\mathbf{R}\,\mathrm{Exp}(\boldsymbol{\omega}\Delta t_k)}_{\mathrm{Exp}(\boldsymbol{\omega}\Delta t_k)} \mathbf{J}^{\mathrm{Exp}(\boldsymbol{\omega}\Delta t_k)}_{\boldsymbol{\omega}\Delta t_k} \mathbf{J}^{\boldsymbol{\omega}\Delta t_k}_{\boldsymbol{\omega}} \\ &= \mathbf{J}_r(\boldsymbol{\omega}\Delta t_k)\Delta t_k, \end{aligned} \tag{12}$$

where we use the notation shortcut $\mathbf{J}^\mathbf{y}_\mathbf{x} \triangleq \partial\mathbf{y}/\partial\mathbf{x}$ for the Jacobians, $\mathbf{Ad}^{-1}_\mathbf{R} = \mathbf{R}^\top$ is the inverse

adjoint matrix of $\mathbf{SO}(3)$, $\mathbf{J}_r(\boldsymbol{\omega}\Delta t_k)$ is the right-Jacobian of $\mathbf{SO}(3)$ given by

$$\mathbf{J}_r(\boldsymbol{\theta}) = \mathbf{I} - \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2 \tag{13}$$

$$\boldsymbol{\theta} = \|\theta\|\mathbf{u} = \boldsymbol{\omega}\Delta t_k \quad \in \mathbb{R}^3\,, \tag{14}$$

$[\cdot]_\times \in so(3)$ is a skew symmetric matrix, and $\boldsymbol{\theta}$ is a rotation vector with module the rotation angle θ and an unitary axis $\mathbf{u}$, calculated with the angular velocity per time. The computation of (8) is greatly accelerated by exploiting the sparsity of the Jacobian $\mathbf{F}$ and partitioning the covariance as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P_{rr}} & \mathbf{P_{rv}} & \mathbf{P_{rR}} & \mathbf{P_{r\omega}} \\ \mathbf{P_{vr}} & \mathbf{P_{vv}} & \mathbf{P_{vR}} & \mathbf{P_{v\omega}} \\ \mathbf{P_{Rr}} & \mathbf{P_{Rv}} & \mathbf{P_{RR}} & \mathbf{P_{R\omega}} \\ \mathbf{P_{\omega r}} & \mathbf{P_{\omega v}} & \mathbf{P_{\omega R}} & \mathbf{P_{\omega\omega}} \end{bmatrix} \quad \in \mathbb{R}^{12\times 12}\,,$$

The 3×3 blocks of $\mathbf{P}$ in (8) is updated as:

$$\begin{aligned}
\mathbf{P_{rr}} &\leftarrow \mathbf{P_{rr}} + (\mathbf{P_{vr}} + \mathbf{P_{rv}} + \mathbf{P_{vv}}\Delta t_k)\Delta t_k \\
\mathbf{P_{rv}} &\leftarrow \mathbf{P_{rv}} + \mathbf{P_{vv}}\Delta t_k \\
\mathbf{P_{rR}} &\leftarrow (\mathbf{P_{rR}} + \mathbf{P_{vR}}\Delta t_k)\mathbf{J}_\mathbf{R}^\top + (\mathbf{P_{r\omega}} + \mathbf{P_{v\omega}}\Delta t_k)\mathbf{J}_{\boldsymbol{\omega}}^\top \\
\mathbf{P_{r\omega}} &\leftarrow \mathbf{P_{r\omega}} + \mathbf{P_{v\omega}}\Delta t_k \\
\mathbf{P_{vv}} &\leftarrow \mathbf{P_{vv}} + \sigma_{\boldsymbol{v_n}}^2\mathbf{I}\Delta t_k \\
\mathbf{P_{vR}} &\leftarrow \mathbf{P_{vR}}\mathbf{J}_\mathbf{R}^\top + \mathbf{P_{v\omega}}\mathbf{J}_{\boldsymbol{\omega}}^\top \\
\mathbf{P_{RR}} &\leftarrow \mathbf{J_R}\mathbf{P_{RR}}\mathbf{J}_\mathbf{R}^\top + \mathbf{J}_{\boldsymbol{\omega}}\mathbf{P}_{\boldsymbol{\omega}\mathbf{R}}\mathbf{J}_\mathbf{R}^\top \\
&\quad + \mathbf{J_R}\mathbf{P_{R\omega}}\mathbf{J}_{\boldsymbol{\omega}}^\top + \mathbf{J}_{\boldsymbol{\omega}}\mathbf{P}_{\boldsymbol{\omega\omega}}\mathbf{J}_{\boldsymbol{\omega}}^\top \\
\mathbf{P_{R\omega}} &\leftarrow \mathbf{J_R}\mathbf{P_{R\omega}} + \mathbf{J}_{\boldsymbol{\omega}}\mathbf{P}_{\boldsymbol{\omega\omega}} \\
\mathbf{P}_{\boldsymbol{\omega\omega}} &\leftarrow \mathbf{P}_{\boldsymbol{\omega\omega}} + \sigma_{\boldsymbol{\omega_n}}^2\mathbf{I}\Delta t_k
\end{aligned}$$

Kalman correction: Each undistorted event $\mathbf{e}(u_e, v_e)$ is treated as an independent observation associated to a 3D segment stored a priori in a map in the form of two endpoints $(\mathbf{p}_1, \mathbf{p}_2) \in \mathbb{R}^3$. The error innovation z is a one-dimensional function that represents the Euclidean distance between the event and the matched projected segment S_i on the image plane, with a measurement noise $n_d \sim \mathcal{N}(0, N_d)$. The distance is computed by first projecting the endpoints $\mathbf{p}_i$ onto the camera $\mathbf{K}(\alpha_u, \alpha_v, u_0, v_0)$ at pose $\{\mathbf{R},\, \mathbf{r}\}$, then expressing the projective line $\mathbf{l}$, and finally computing the point-to-line distance $z = d(\mathbf{e}, \mathbf{l})$. The (scalar) innovation variance Z is given by

$$Z = \mathbf{J}_\mathbf{x}^z\mathbf{P}\mathbf{J}_\mathbf{x}^{z\top} + N_d \quad \in \mathbb{R} \tag{15}$$

where the Jacobian of the state with regard to the innovation is expressed as follows

$$\mathbf{J}_\mathbf{x}^z = [\mathbf{J}_\mathbf{r}^z, \mathbf{0}_{1\times 3}, \mathbf{J}_\mathbf{R}^z, \mathbf{0}_{1\times 3}] \quad \in \mathbb{R}^{1\times 12} \tag{16}$$

The Kalman correction is applied, based on the Lie group considerations as follows:

$$\text{Kalman gain :} \qquad \mathbf{k} = \mathbf{P}\mathbf{J}_\mathbf{x}^{z\top}Z^{-1} \in \mathbb{R}^{12\times 1} \tag{17}$$

$$\text{Observed error :} \qquad \delta\mathbf{x} = \mathbf{k}z \tag{18}$$

$$\text{State update :} \qquad \mathbf{x} \leftarrow \mathbf{x} \oplus \delta\mathbf{x} \tag{19}$$

$$\text{Covariance update :} \qquad \mathbf{P} \leftarrow \mathbf{P} - \mathbf{k}Z\mathbf{k}^\top \tag{20}$$

where we remark that in (17) we again exploit the sparsity of $\mathbf{J}_\mathbf{x}^z$ given on (16) where

$$\mathbf{J}_\mathbf{r}^z = \mathbf{J}_\mathbf{l}^z(\mathbf{J}_{\mathbf{u}_1}^\mathbf{l}\mathbf{J}_\mathbf{r}^{\mathbf{u}_1} + \mathbf{J}_{\mathbf{u}_2}^\mathbf{l}\mathbf{J}_\mathbf{r}^{\mathbf{u}_2}) \qquad \in \mathbb{R}^{1\times 3} \tag{21}$$

$$\mathbf{J}_\mathbf{R}^z = \mathbf{J}_\mathbf{l}^z(\mathbf{J}_{\mathbf{u}_1}^\mathbf{l}\mathbf{J}_\mathbf{R}^{\mathbf{u}_1} + \mathbf{J}_{\mathbf{u}_2}^\mathbf{l}\mathbf{J}_\mathbf{R}^{\mathbf{u}_2}) \qquad \in \mathbb{R}^{1\times 3} \tag{22}$$

are associated with the Jacobians $\mathbf{J}_\mathbf{r}^{\mathbf{u}_i}$, $\mathbf{J}_\mathbf{R}^{\mathbf{u}_i}$, $\mathbf{J}_{\mathbf{u}_i}^\mathbf{l}$, and $\mathbf{J}_\mathbf{l}^z$ that comes from the derivation of the error innovation z and, the expression of the projective lines $\mathbf{l}$, that are computed with two projective

endpoints $\underline{\mathbf{u}}_1$ and $\underline{\mathbf{u}}_2$. These last ones are the result of the back projection in the scene of the segments endpoints of the wire-frame map. The inverse of the innovation covariance Z^{-1} becomes a division by a scalar because of the matrix dimensions, and the update (19) is implemented by a regular sum for the composite blocks $\{\mathbf{r}, \mathbf{v}, \boldsymbol{\omega}\}$ and by the right-plus (7) for $\mathbf{R} \in \mathbf{SO}(3)$.

Motion Model of Constant Position: To further improve the computational burden, we relaxed the assumption from constant velocity to constant position. At a micro-second rate, motion is negligible; thus, this is a valid assumption. Since this new motion model does not take into account the linear and angular speed its size is reduced by half in the same way as the response time without a significant reduction in the tracking accuracy. The system state now is represented as member of the Lie Group **SE**(3), the set of 3D rigid body transformations, which is related to a reduced error estate that lies on the tangent space $\delta\mathbf{x} \in \mathcal{T}\mathbf{SE}(3)$.

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \in \mathbb{R}^3 \\ \mathbf{R} \in \mathbf{SO}(3) \end{bmatrix} \quad \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{r} \\ \delta\boldsymbol{\theta} \end{bmatrix} \sim \mathcal{N}(\bar{\delta\mathbf{x}}, \mathbf{P}) \tag{23}$$

The state transition model depends just on the state at the time $k-1$ and on the position and rotation perturbations ($\mathbf{r}_{\mathbf{n}k}$ and $\mathbf{R}_{\mathbf{n}k}$ respectively) as follows:

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \mathbf{r}_{\mathbf{n}k} \tag{24}$$

$$\mathbf{R}_k = \mathbf{R}_{k-1} + \mathbf{R}_{\mathbf{n}k} \tag{25}$$

This consideration leads to a significant simplification of (8), where the Jacobian $\mathbf{F}$ becomes an identity matrix $\mathbf{I} \in \mathbb{R}^{6\times6}$ and the perturbation matrix $\mathbf{Q} = diag(\sigma^2_{\mathbf{r_n}}\mathbf{I}, \sigma^2_{\mathbf{R_n}}\mathbf{I})\Delta t_k \in \mathbb{R}^{6\times6}$. The innovation covariance in (15) is recalculated taking into account its reduced sized elements. The Jacobian of the innovation in (16) does not change on form but, it does in size becoming $\mathbf{J}^z_{\mathbf{x}} = [\mathbf{J}^z_{\mathbf{r}}, \mathbf{J}^z_{\mathbf{R}}] \in \mathbb{R}^{1\times6}$. The update (19) is implemented by the right-plus, in addition it is important to remark that as the previous case, we also the exploit the sparsity of the innovation Jacobian and Z^{-1} is a division by a scalar.

5.1.3 Fast event-to-line matching

The EKF update algorithm above is enriched with outlier detection and rejection capabilities in what constitutes our event-to-line matching algorithm. The goal is to rapidly discard or validate events before proceeding with the update. For this, our algorithm uses epoch-based projection and image tessellation to accelerate the search for candidates.
The algorithm divides the event image into $n \times m$ (square) cells. Each landmark segment and each event can be easily located inside a cell. Arriving events are then compared against only the segments in its own cell instead of make comparisons with all the elements listed in the scene map. The event-line distances are computed using the event coordinates and the segment parameters and sorted in ascending order. To validate a match between an event and its closest segment the following three conditions (evaluated in this order) must be met:

- The distance d_1 to the closest segment is *below* a predefined threshold,

  $$d_1 < \alpha \tag{26}$$

  where we usually take $\alpha \sim 2\,\text{pixels}$.

- The distance d_2 to the second closest segment is *above* another predefined threshold,

  $$d_2 > \beta \tag{27}$$

  where we usually take $\beta \sim 2\alpha$.

- The orthogonal projection of the event onto the segment falls between the two endpoints,

$$0 < \frac{\mathbf{v}_1^\top \mathbf{v}_2}{\mathbf{v}_1^\top \mathbf{v}_1} < 1 \tag{28}$$

where $\mathbf{v}_1 = \mathbf{u}_2 - \mathbf{u}_1$ and $\mathbf{v}_2 = \mathbf{e} - \mathbf{u}_1$, and $\mathbf{u}_i$ are the endpoints in pixel coordinates, given by $\mathbf{u} = \underline{\mathbf{u}}_{xy}/\underline{\mathbf{u}}_w$.

If the match is validated, the segment is reprojected in the image and we proceed with the updating procedure. Otherwise, the event is discarded. The results of the tracking are displayed on Fig.2. The 3D camera position (red) is compared against the ground truth position given by an Optitrack system (blue). The visualization output for this tracking process in shown in Fig. 3, the cube is reprojected taking into account the estimated pose and plotted onto the image in green lines.

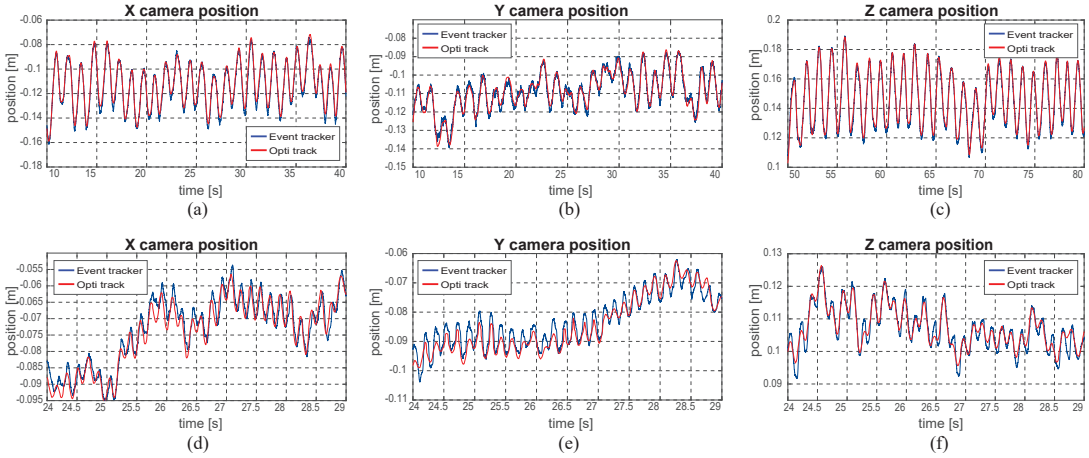

Figure 2: Tracking performance compared with an Optitrack ground truth: Camera position for a moderate speed circular motion sequence of 1 Hz (a-c); and for a high speed back and forth (d-f)

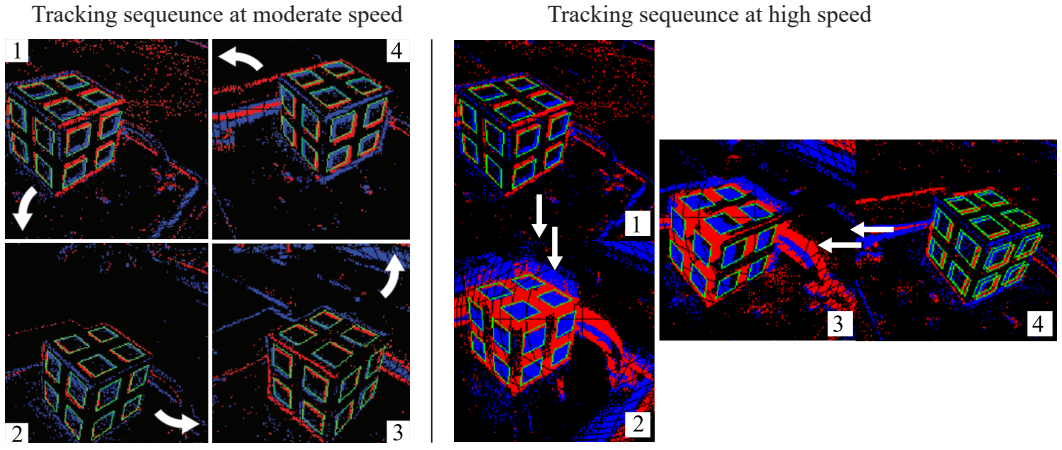

Figure 3: Tracking sequence visualization: Left.- Sequence for circular motion at moderate speed. Right.- Sequence for fast linear motion. Note the events are plotted in red and blue, and the reprojected tracked cube in green lines

5.2 Event-based 3D recovery

We address the problem of depth estimation and 3D reconstruction for mapping with a single event camera by taking inspiration in the space-sweep method initially described on [31] and tested

with events as in [18]. This method is based on a voting and maximization strategy to estimate semidense depth maps at selected viewpoints, these must be merged in a global reference frame to build larger 3D models. The 3D recovered features are filtered and displayed as a point cloud, we remove the outliers by applying a Ransac algorithm, which also allows to extract segment-like features after an iterative process. The depth estimation process is described below.

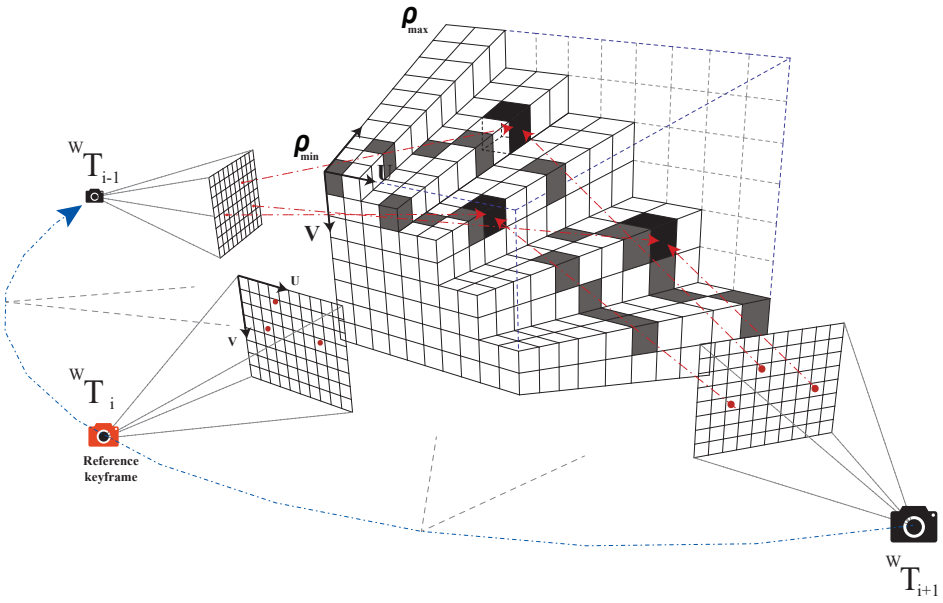

Figure 4: Projective grid with a ray tracing representation (red)from different camera poses

5.2.1 Monocular space sweep

Each event $\mathbf{e}(u_e, v_e, t_k, \rho_e)$ is back-projected from to the viewpoint of the camera at time t_i to a three dimensional grid that discretizes the volume containing the 3D scene, as is shown in Fig. 4. We choose a projective spatial grid located in a keyframe position; along the camera trajectory; with respect to a global reference frame. This grid is splitted into small volumes called voxels with a size of N_x, N_y; whose values are equal to the event camera resolution; and N_z depth planes that are equispaced in inverse depth. We note this keyframe pose as a rigid body transformation from the reference viewpoint i to the world w, where ${}^wT_i = (R_i|r_i) \in \mathbf{SE}(3)$, and the grid axes with U, V and depth ρ. To accelerate the events back-projection process, we exploit the particularity that the pose associated to two consecutive events does not have a significant change; thus we can assign a unique interpolated pose value to a batch of n_e events. All the poses are obtained from the tracker stated previously.

Lets justify the shape of the chosen grid. A projective ray in this space is given by a starting point $\mathbf{C}$ which is the optical camera center, a director vector $\mathbf{D}$ determined by a projected point at infinity and a depth factor λ related by the equations:

$$\mathbf{L} = \lambda \mathbf{D} + \mathbf{C} \quad \forall L_i = (U_i, V_i, \rho_i)^T \tag{29}$$

$$\mathbf{X_i} = \left(\frac{X_i}{Z_i}, \frac{Y_i}{Z_i}, \rho_i \right)^T \quad \forall \rho_i \propto \frac{1}{Z_i} \rightarrow \frac{k_{inv}}{Z_i}, \tag{30}$$

where $\mathbf{X_i}$ is the representation of an point of the ray $\mathbf{L}$ using Euclidean coordinates (X_i, Y_i, Z_i), whose depth value is proportional to the inverse of Z. Combining (29) and (30) to obtain a parametric equation

$\mathbf{X_i} = (\frac{\lambda \mathbf{D_x}+\mathbf{C_x}}{\lambda \mathbf{D_z}+\mathbf{C_z}}, \frac{\lambda \mathbf{D_y}+\mathbf{C_y}}{\lambda \mathbf{D_z}+\mathbf{C_z}}, \frac{k_{inv}}{\lambda \mathbf{D_z}+\mathbf{C_z}})$ and looking at the changes on the depth parameter λ results:

$$\frac{d\mathbf{X_i}}{d\lambda} = \left(\frac{\mathbf{D_x C_z} - \mathbf{D_z C_x}}{(\lambda \mathbf{D_z} + \mathbf{C_z})^2}, \frac{\mathbf{D_y C_z} - \mathbf{D_z C_y}}{(\lambda \mathbf{D_z} + \mathbf{C_z})^2}, \frac{-k_{inv} \mathbf{C_z}}{(\lambda \mathbf{D_z} + \mathbf{C_z})^2} \right) \tag{31}$$

For smaller displacements of the camera center than the mean scene depth, we can consider $C_z \ll D_z$, and (31) becomes $\frac{d\mathbf{X_i}}{d\lambda} \approx (\frac{-\mathbf{C_x}}{\mathbf{(D_z \lambda^2}}, \frac{-\mathbf{C_y}}{\mathbf{(D_z \lambda^2}}, \frac{-k_{inv}\mathbf{C_z}}{\mathbf{(D_z \lambda^2}})$, that means for the given condition, the changes in depth vary in the same proportion in all the axes; as a result, all the rays projected inside the grid will be straight lines. The importance of having straight rays lies in the recovery of straight segments from the edges in the scene, that are required to build a wire-frame map. A simulation example to test this statement was carried out with a straight line in the space, and a batch of p camera poses previously recorded. The result displayed in Fig. 5 (left), shows how the sweep algorithm produces straight rays in inverse depth (blue lines) and it allows to recover a straight 3D line (red). On the other hand the sweep algorithm with the projective grid equispaced (Fig. 5 - right) in depth leads to curved rays and a curved line recovered. Is important to remark that in both cases the original 3D line proposed for this test is straight.

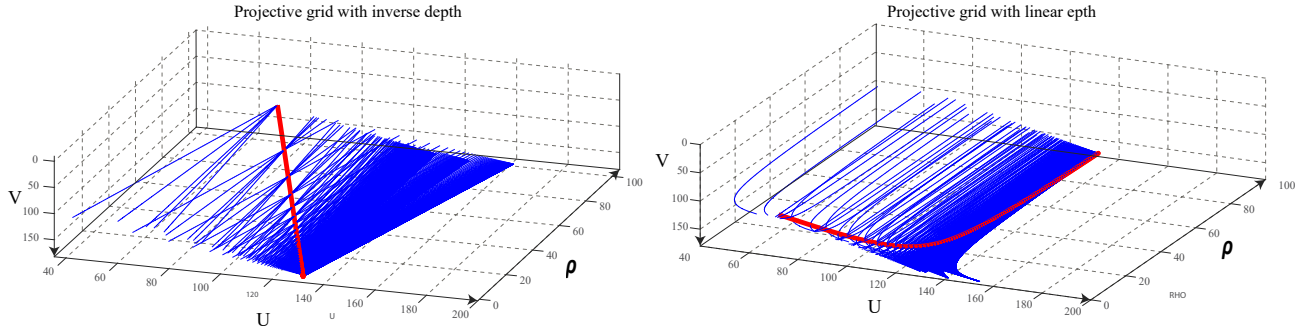

Figure 5: 3D straight line in projective grid. Left.- straight line in inverse depth (red). Right.- straight line (distorted) in linear depth(red)

For the same scene, the collection of rays in the projective grid is shown in Fig. 6, where we can notice the straight and curved rays for inverse and linear depth.

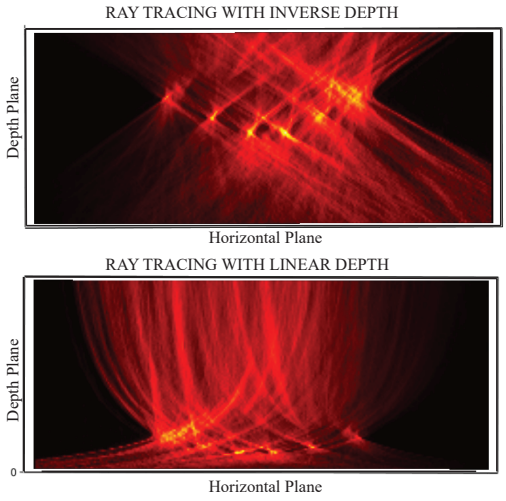

Figure 6: Ray tracing in the projective grid. Top.- rays in inverse depth. Bottom.- rays in linear depth

In order to maintain an accurate sampling we need to choose a different keyframe as soon as the camera displacement exceeds a certain percentage of the mean scene depth.

5.2.2 Event back-projection onto the reference frame

The planar homography matrix transfers the events onto the sweeping plane at some canonical position. In general terms, $\mathbf{H}_{\rho_\mathbf{i}}$ represents the homography mapping image points onto a depth plane $\rho = \rho_i$. The space sweep approach of [31] [18], computes the event projection onto the plane $\rho = \rho_0$ from the projective grid via planar homography $\mathbf{H}_{\rho_\mathbf{0}}$. Since homographies are invertible, we can write a homography that maps features between the plane $\rho = \rho_0$ and $\rho = \rho_i$ directly, by using the projection onto the plane ρ_0, then back projecting them to the ρ_i plane with:

$$\mathbf{H}_{\rho_\mathbf{i}}\mathbf{H}_{\rho_\mathbf{0}}^{-\mathbf{1}} \quad \sim (U(\rho_i), V(\rho_i), 1)^T \in \mathbb{R}^{3\times3}. \tag{32}$$

The homography to transfer the events $\mathbf{H}_{\rho_\mathbf{0}}(u_e, v_e, 1)^T \triangleq (U(\rho_0), V(\rho_0), 1)^T$ can be found using the inversion lemma as,

$$\mathbf{H}_{\rho_\mathbf{0}}{}^{-1} = \left({}^{\mathbf{i}}\mathbf{R}_{\mathbf{RV}} + \frac{1}{\rho_0}{}^{\mathbf{i}}\mathbf{r}_{\mathbf{RV}}\mathbf{n}^T\right) \quad \in \mathbb{R}^{3\times3}, \tag{33}$$

where ${}^{\mathbf{i}}\mathbf{R}_{\mathbf{RV}}$ and ${}^{\mathbf{i}}\mathbf{r}_{\mathbf{RV}}$ are the rotation matrix and translation vector from the reference keyframe to the instantaneous pose i of the event camera, and $\mathbf{n} = (0, 0, 1)^T$ is the unit normal to the plane. Lets call $\mathbf{C} = -{}^{\mathbf{i}}\mathbf{R}_{\mathbf{RV}}{}^T{}^{\mathbf{i}}\mathbf{r}_{\mathbf{RV}} \triangleq (C_U, C_V, C_\rho)$ we can express (32) as follows

$$\mathbf{H}_{\rho_\mathbf{i}}\mathbf{H}_{\rho_\mathbf{i}}^{-\mathbf{1}} = \left(\mathbf{I} + \frac{\rho_0 - \rho_i}{\rho_0(\rho_i - C_\rho)}\mathbf{C}\mathbf{n}^T\right) = \begin{bmatrix} 1 - \frac{C_\rho}{\rho_i} & 0 & (\frac{1}{\rho_i} - \frac{1}{\rho_0})C_U \\ 0 & 1 - \frac{C_\rho}{\rho_i} & (\frac{1}{\rho_i} - \frac{1}{\rho_0})C_V \\ 0 & 0 & 1 - \frac{C_\rho}{\rho_i} \end{bmatrix}. \tag{34}$$

Dividing the homogeneous matrix (34) by its last entry and writing (32) in expanded form gives:

$$\begin{matrix} U(\rho_i) = aU(\rho_0) + b_U \\ V(\rho_i) = aV(\rho_0) + b_V \end{matrix} \quad \text{where} \quad \begin{cases} a &= \frac{\rho_0(\rho_i - C_\rho)}{\rho_i(\rho_0 - C_\rho)} \\ b_U &= \frac{C_U(\rho_0 - \rho_i)}{\rho_0 - C_\rho} \\ b_V &= \frac{C_V(\rho_0 - \rho_i)}{\rho_0 - C_\rho} \end{cases} \tag{35}$$

This approach offers a fast way to compute the ray projection onto each depth plane where, if there is available information for the fist depth plane, the ray projection along the subsequent planes is easily computed by using (35). For each event an a priori calculation of the constants a, b_U and b_V improves the ray projection onto the projective grid. After this, we accumulate votes on each voxel with center $(U_{\rho_i}, V_{\rho_i}, \rho_i)$. To work with calibrated coordinates, a projective homography matrix $\mathbf{G}$ must be used instead of the Euclidean homography $\mathbf{H}$, which requires the intrinsic camera matrix $\mathbf{K}$ where $\mathbf{G} = \mathbf{K}\mathbf{H}\mathbf{K}^{-1}$ [43].

The voxels with the greatest probability of containing a 3D points will have a high vote counting, the rest, must be removed from the projective grid by means of vote thresholding, where we use the local maxima on each depth plane, then the recovered 3D points are stored in a point cloud for a further feature extraction. Fig. 7 indicates how the votes are organized on each voxel, the color intensity is proportional to the number of votes. We can notice the accumulation of votes in the edges highlighted by the events.

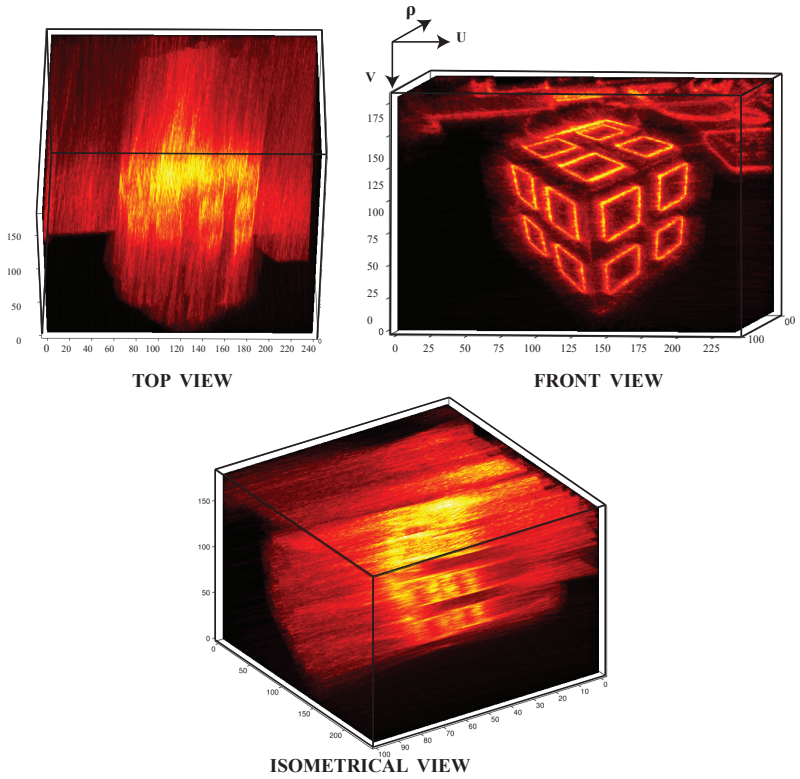

Figure 7: Projective grid with votes representation

5.2.3 Feature Extraction

We use a RANdom SAmple Consensus (RANSAC) algorithm to look for straight lines in the point cloud constructed above. This algorithm helps to estimate the parameters of a mathematical model from a set of observed data which contains outliers. The implementation of this algorithm for 2D line detection in a set of points with a big amount of outliers, has reported a high degree of accuracy compared with the least square method [44]. At this point, this feature extraction method is still in progress. The first tests have shown encouraging results, however all the segments must be properly parametrized and filtered to have accurate features. Our RANSAC strategy follow the next steps:

- **Random sample**: From the thresholded grid, select two random voxels to create an hypothetical line with a distance greater than a predefined value and with non empty voxels located on specific keypoints along this line.
- **Consensus**: vote for all the inliers to this hypothetical line across the grid.
- **Thresholding and fitting**: Remove the lines that do not have enough votes to be considered a valid line. For all potential winner candidates a 3D line will be fitted taken into account their associated inliers.

6 Work Plan

In order to reach the main objective of this research, we have divided the work in two main task related with tracking and mapping, then the aim is to create a closed loop between them to share information and locate the event camera in the scene. Currently we have proposed a wire-frame

tracker and a 3D recovery based on space sweep to recover straight lines. To close the loop is still required tasks of map optimization, occlusion removal, trajectory optimization among others. All the stages of the work will be formulated and validated with real data along the creation of the corresponding Matlab, C++ and ROS codes. The tasks we planned are listed below.

6.1 Task 1: Literature Review

The literature review is recursive throughout the thesis work, specially at the beginning of each task. It consists in a deep study of the literature related to tracking and mapping techniques with event cameras. This will allow to state a position in the field, and track the work of dedicated research groups. *Robotics and Perception* from ETH. Zurich is the pioneer in computer vision based on event cameras, and with which a research stay will be planned.

6.2 Task 2: Event based tracking

This task aims to study and develop tracking techniques based on events, for fast and robust information processing. Our goal is to produce a 3D pose estimation in a range up to MHz, based on the feature detected. This involves the following subtasks:

Task 2.1: Bootstrapping

Before launching the tracking algorithm we require the initial camera pose as well as some initial landmarks. As an initial approach we set a cubic pattern, parameterized on a global reference. Based on this cube we already can estimate an initial position and have an apriori map for testing. This bootstrap technique will evolve in the third year of the thesis as the complexity of the scene is increased.

Task 2.2: Tracking formulation

This task is related with the mathematical formulation for the Bayesian filter that estimates the pose with 6DoF (position and orientation). Our approach starts with an Error State Kalman Filter which has been studied in a Classic and Lie groups formulation with different motion models (Constant Position, Constant Velocity). The aim was to formulate a configuration that provides the fastest and most robust estimation. Line to point distance is used as update method, since we initially considered a segment landmark, although it might evolve to other measurement models

6.3 Task 3: Monocular depth estimation and feature aggregation

Once the tracking algorithm is in place, we need to create a method to involve a larger map. On this task we explore depth estimation and feature aggregation techniques that allow to extract new 3D features from the environment. This mapping task is dived in two subtasks:

Task 3.1: 3D Recovery

Our goal is to estimate depth for the main features of the scene. The space-sweep discretization is our starting point, from which we can recover a 3D point cloud. We will look for filtering techniques to clean the map and optimization methods to improve the current algorithm. At the moment it does not requires bigger computational resources and its implementation in C++ via threads offers a fast response.

Task 3.2: Feature aggregation and landmark parametrization

To solve this task, we plan to look for new landmarks in the 3D point cloud by using a RANSAC algorithm. It will fit the 3D points to a specific model -which will be initially formed by segments- and remove the outliers. Other landmark parametrizations will be taken into account to improve the accuracy of the detected features.

Task 3.3: Map optimization

We recover features from several keyframes at different locations along the camera trajectory, and all of them must be merged onto a global map. This task aims to develop a merging method for similar features, and a step to add new features to the map eliminating outliers.

Task 3.4: Occlusion removal

We plan to study techniques for selecting specific landmarks from the global map according to the current camera viewpoint. This task will help to avoid tracking occluded landmarks whose back projection should not be present at this time.

6.4 Task 4: Trajectory Refinement

This task seeks to obtain a smooth and robust estimation of the camera pose, based in the event-only approach. To this end, we plan to consider an external source of information to complement the events without affecting its fast performance and then, perform a smoothing approach (post estimation) to generate accurate and continuous trajectories.

Task 4.1: Sensor fusion

We will explore the use of external sources of information -being the camera IMU the preferred choice- to offer a more robust pose estimation on complex natural scenes. The sensor output and the events could also be fused via a Kalman Filter to improve the pose estimation.

Task 4.2: Trajectory smoothing

The estimated trajectories; based only in events; are described on discrete time, they contain noise caused by the high frequency of all incoming events. This task aims to generate smooth and continuous trajectories -such as cubic splines method- taking into account past predictions. We will study techniques able to produce a smooth prediction without affecting the tracking response time.

6.5 Task 5: Thesis writing

All the results will be integrated in the doctoral thesis. At least two journal papers are planned, one for the tracking method and one for the full system. Yearly conference papers to top-ranked conferences will also be proposed.

7 Resources

This research is partially supported by the EU H2020 project GAUSS (H2020-Galileo-2017-1-776293), by the Spanish State Research Agency through projects EB-SLAM (DPI2017-89564-P) and the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656. The work is being

developed at the Institut de Robotica i Informatica Industrial (IRI), UPC-CSIC, in Barcelona, within the Mobile Robotics research group.

In order to conduct the major part of the experiments and to test the presented approaches, we use the event camera DAVIS240C in conjunction with ROS environment. The camera drivers as well as the calibration software are available online as an open source application, these are up to date and support the newest firmware versions and camera models. This camera model contains an internal IMU; however, the IRI labs have in storage other inertial and laser sensors in case they are needed.

In addition in order to promote the research with event based cameras, The Perception Lab of the ETH Zurich has released several datasets of natural and synthetic scenes that contains events and the ground truth along the trajectory.

Our laboratory for this project is equipped with a standard desktop computer with Ubuntu operative system, ROS, Matlab among other applications. Our Mobile Robotics laboratory is equipped with an Optitrack system to provide ground truth trajectories.

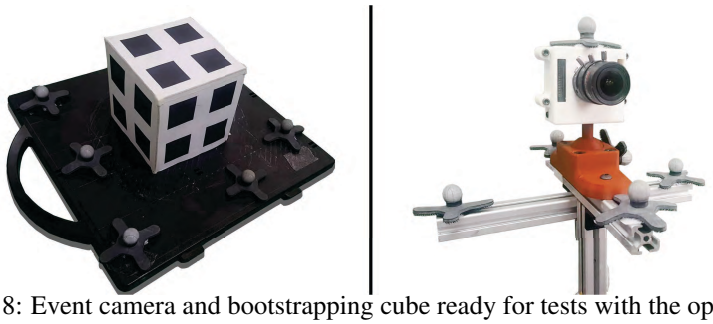

Figure 8: Event camera and bootstrapping cube ready for tests with the optitrack

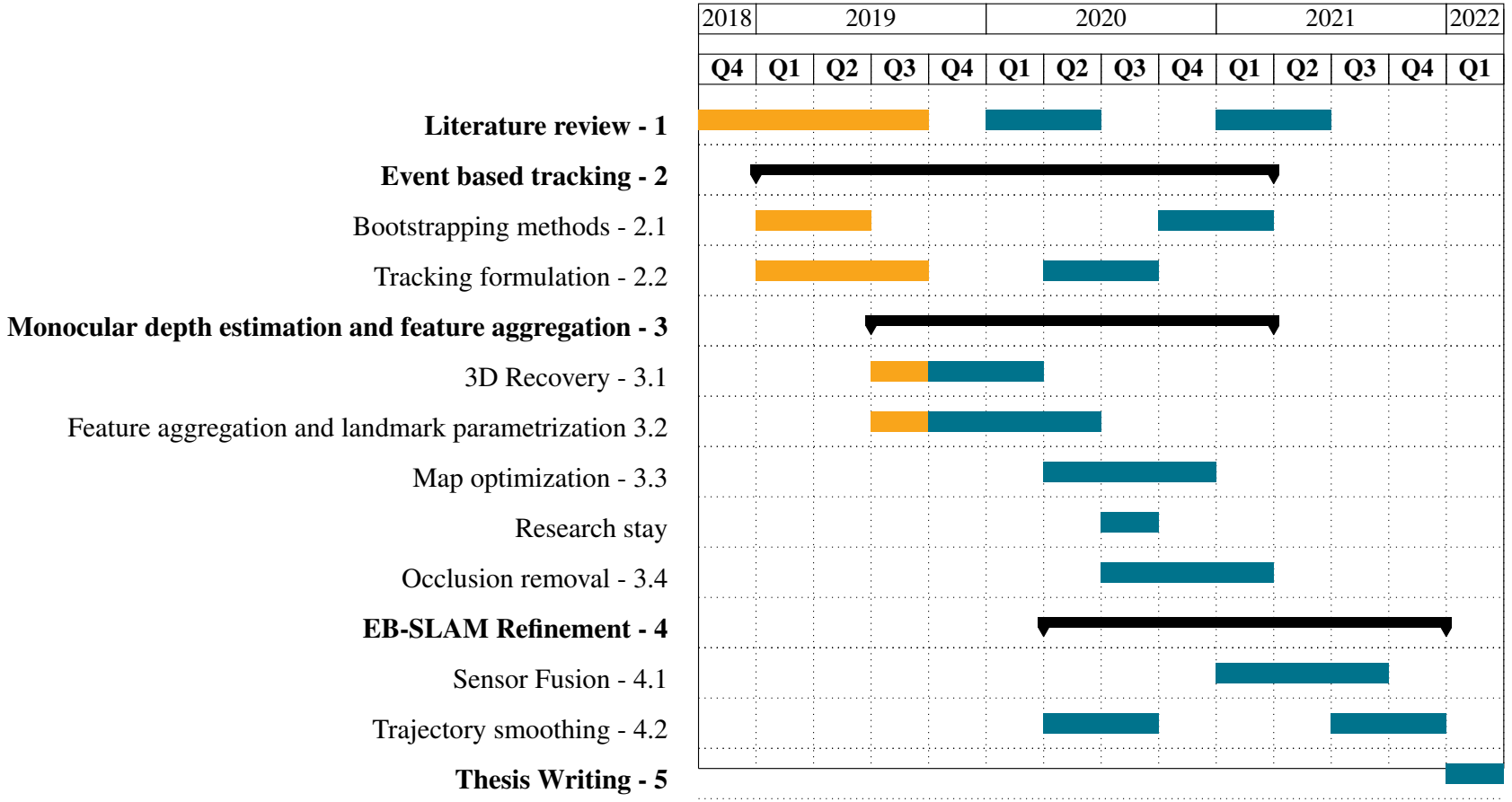

Figure 9: Intended work plan of the proposed thesis

References

[1] M. O. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *Springer Plus*, vol. 5, no. 1, p. 1897, 2016.

[2] T. Huang, "Computer vision: Evolution and promise," 1996.

[3] J. Sola, *Towards visual localization, mapping and moving objects tracking by a mobile robot: a geometric and probabilistic approach*. PhD thesis, 2007.

[4] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3d line-based map using stereo slam," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1364–1377, 2015.

[5] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[6] F. Fiorillo, M. Limongiello, and B. J. Fernández-Palacios, "Testing gopro for 3D model reconstruction in narrow spaces," *Acta Imeko*, vol. 5, pp. 64–70, 2016.

[7] A. Kreimer, E. Rivlin, and I. Shimshoni, *Algorithms for Visual Odometry*. PhD thesis, Computer Science Department, Technion, 2019.

[8] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," 2007.

[9] C. Brandli, R. Berner, Minhao Yang, Shih-Chii Liu, and T. Delbruck, "A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[10] 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.

[11] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, 2007.

[13] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2706–2719, 2013.

[14] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. Davison, "Simultaneous mosaicing and tracking with an event camera," in *British Machine Vision Conference*, 2014.

[15] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras.," in *British Machine Vision Conference*, vol. 2, p. 8, 2013.

[16] T. Delbruck, V. Villanueva, and L. Longinotti, "Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision," in *IEEE International Symposium on Circuits and Systems*, pp. 2636–2639, 2014.

[17] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, 2011.

[18] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, "EMVS: Event-based multi-view stereo—3d reconstruction with an event camera in real-time," *International Journal of Computer Vision*, vol. 126, no. 12, pp. 1394–1414, 2018.

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

[20] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

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

[22] D. Weikersdorfer, R. Hoffmann, and J. Conradt, "Simultaneous localization and mapping for event-based vision systems," in *International Conference on Computer Vision Systems*, pp. 133–142, 2013.

[23] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," in *IEEE International Conference on Robotics and Automation*, pp. 703–710, 2014.

[24] M. Milford, H. Kim, S. Leutenegger, and A. Davison, "Towards visual slam with event-based cameras," in *The Problem of Mobile Sensors Workshop in Conjunction with RSS*, 2015.

[25] E. Mueggler, G. Gallego, and D. Scaramuzza, "Continuous-time trajectory estimation for event-based vision sensors," in *Robotics: Science and Systems XI*, 2015.

[26] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *European Conference on Computer Vision*, pp. 349–364, 2016.

[27] G. Gallego, J. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-dof camera tracking for high-speed applications," *arXiv preprint arXiv:1607.03468*, vol. 2, 2016.

[28] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-DOF camera tracking from photometric depth maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 10, pp. 2402–2412, 2018.

[29] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, "Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2016.

[30] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5816–5824, 2017.

[31] R. T. Collins, "A space-sweep approach to true multi-image matching," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 358–363, 1996.

[32] C. Brandli, T. Mantel, M. Hutter, M. Höpflinger, R. Berner, R. Siegwart, and T. Delbruck, "Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, p. 275, 2014.

[33] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," in *IEEE International Conference on Robotics and Automation*, pp. 359–364, 2014.

[34] N. Matsuda, O. Cossairt, and M. Gupta, "MC3D: Motion contrast 3D scanning," in *IEEE International Conference on Computational Photography*, pp. 1–10, 2015.

[35] J. N. Martel, J. Müller, J. Conradt, and Y. Sandamirskaya, "An active approach to solving the stereo matching problem using event-based sensors," in *IEEE International Symposium on Circuits and Systems*, pp. 1–5, 2018.

[36] J. Kogler, C. Sulzbachner, M. Humenberger, and F. Eibensteiner, "Address-event based stereo vision with bio-inspired silicon retina imagers," in *Advances in Theory and Applications of Stereo Vision*, IntechOpen, 2011.

[37] S. Schraml, A. N. Belbachir, N. Milosevic, and P. Schön, "Dynamic stereo vision system for real-time tracking," in *IEEE International Symposium on Circuits and Systems*, pp. 1409–1412, 2010.

[38] J. Carneiro, S.-H. Ieng, C. Posch, and R. Benosman, "Event-based 3D reconstruction from neuromorphic retinas," *Neural Networks*, vol. 45, pp. 27–38, 2013.

[39] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," Tech. Rep. IRI-TR-18-01, Institut de Robòtica i Informàtica Industrial, Barcelona, 2018.

[40] J. Deray and J. Solà, "`manif`: a small C++ header-only library for Lie theory." `https://github.com/artivis/manif`, January 2019.

[41] E. Rosten, R. Porter, and T. Drummond, "FASTER and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010.

[42] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[43] E. Malis and M. Vargas, "Deeper understanding of the homography decomposition for vision-based control," 2007.

[44] Z. Lu, S. Baek, and S. Lee, "Robust 3D line extraction from stereo point clouds," in *IEEE Conference on Robotics, Automation and Mechatronics*, pp. 1–5, 2008.