Gyrevento: Event-based Omnidirectional Visual Gyroscope in a Manhattan World

Daniel Rodrigues Da Costa, Pascal Vasseur, Fabio Morbidi

***Abstract*—In this paper, we study the problem of estimating the orientation of an event omnidirectional camera mounted on a robot and observing 3D parallel lines in a man-made environment (Manhattan world). We present Gyrevento, the first event-based omnidirectional visual gyroscope. Gyrevento does not require any initialization, provides certifiably globally optimal solutions, and is scalable, since the size of the nonlinear least-squares cost function is independent of the number of lines. Thanks to the Cayley-Gibbs-Rodrigues parameterization of a 3D rotation, this cost function is a degree-four rational function in three variables, which can be efficiently minimized via off-the-shelf polynomial optimization software. Numerical simulations and real-world experiments with a robot manipulator show the effectiveness of our visual gyroscope and elucidate the impact of camera velocity on the attitude estimation error.**

I. Introduction

In the last decade, *event cameras* have been increasingly used in computer vision and robotics [1], and they have made serious inroads into real-world applications [2]. However, their potential with agile robots, has yet to be fully exploited. Event cameras asynchronously measure per-pixel brightness changes, and output a stream of events that encode the time, location and sign of these variations. They have several attractive properties, compared to standard frame-based cameras: high temporal resolution (in the order of microseconds), very high dynamic range (up to 140 dB), low power consumption, and high pixel bandwidth (in the order of kHz), which results in reduced motion blur. Finally, the spatial resolution of consumer-grade event cameras is comparable to that of traditional cameras (1280 $\times$ 720 pixels in Prophesee EVK4 - HD).

An area of active research in event-based vision, is *visual odometry*. This is a challenging task, since event cameras do not output images, and traditional computer vision algorithms cannot be applied directly. Numerous event-based visual odometry algorithms have appeared in the recent literature [3]–[10]. However, with the exception of [3], [4], [7], [9], which only take a stream of events from a monocular camera as input, the other methods rely on an Inertial Measurement Unit (IMU) [5], [6], a second extrinsically-calibrated depth camera [10], or combine events and grayscale frames from a DAVIS sensor [8].

Line features have a simple mathematical representation and they are ubiquitous in urban environments (e.g., doors, windows). Moving event cameras can easily detect edges in a 3D scene and several methods have been recently proposed for event-based line detection and tracking. In [11], a variant of Line Segment Detection (LSD) algorithm tailored to event data, called ELiSeD, is introduced, while in [12], the authors presented a method based on the detection of event planes in the (x, y, t) space and tracking of these planes over time. On the other hand, in [13], an algorithm for line and segment detection which relies on iterative event-based weighted least-squares fitting, is proposed. In [14], a geometric model of line-generated manifolds is introduced, and the linear velocity of an event camera is estimated from angular rates measured by an IMU. This work has been recently improved in [15] (see also [16], where a stereo event camera and an IMU are used). Finally, in [17], the authors describe a line-based PTAM (Parallel Tracking and Mapping) algorithm that estimates structure and motion, and produces accurate camera pose estimates in real-time, by using an error-state Kalman filter. The line extraction procedure is based on the Hough transform.

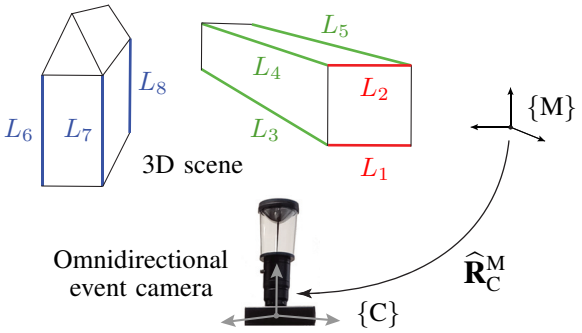


Figure 1. Three parallel and mutually orthogonal groups of 3D lines $\{L_1, L_2\}$ (red), $\{L_3, L_4, L_5\}$ (green), $\{L_6, L_7, L_8\}$ (blue), are observed by an omnidirectional event camera. *Gyrevento* estimates the orientation of the camera frame {C} with respect to the Manhattan (or world) frame {M}.

Unfortunately, event cameras, as conventional vision sensors, suffer from a *narrow field of view* (FoV), which is problematic when they are mounted on high-speed robots. Relatively few works in the literature have addressed this problem and dealt with omnidirectional event cameras. In [18], the authors developed a stereo system consisting of two event cameras mounted on a rotating head, which provides real-time 360° panoramic vision. In [19], a panoramic tracking and mapping algorithm that only relies on the pixel positions of the events in the continuous stream is proposed. Finally, in [20] a multi-view stereo method, called EOMVS, is designed to estimate a depth map and reconstruct a 3D scene using an omnidirectional (fisheye) event camera.

This paper presents *Gyrevento*, a new event-based omnidirectional visual gyroscope[1] (see Fig. 1). The combination of a 360° view and an event-based mechanism offers two

The authors are with the MIS laboratory, University of Picardie Jules Verne, 33 rue Saint-Leu, 80039 Amiens Cedex, France. Corresponding author: D. Rodrigues Da Costa, Email: daniel.rodrigues.da.costa@u-picardie.fr

This work was supported by AID (Agence de l'Innovation de Défense), through the research project EVENTO, "*Omnidirectional Event Cameras for High-Speed Robots*" (2021-2024) and by the ANR-FWF EVELOC "*Event-based Visual Localization*" project (ANR-23-CE33-0011).

[1]By *visual gyroscope*, we mean an algorithm which exploits the visual information to estimate the 3D orientation of a camera (but *not* its velocity).

advantages which have not been jointly explored so far, in the literature: the primitives (lines, in our case) always lie within the FoV of the camera (persistence of measurement) and the gyroscope is robust to strong accelerations (no motion blur). Unlike conventional IMUs, our visual gyroscope is drift-free, thus providing more reliable long-term attitude estimates. Our work builds upon a large body of research on vanishing point extraction and attitude estimation in a Manhattan world, for perspective and omnidirectional color cameras [21]–[24]. By using the unified central projection model [25], these results have been adapted to meet the special requirements of an omnidirectional event camera. Gyrevento processes an asynchronous stream of events, and detects and tracks great circles on the unit sphere (the projection of groups of 3D parallel lines in the Manhattan world) to estimate the orientation of the camera frame. A RANSAC-based classifier generates hypotheses for orthogonal vanishing points. Our nonlinear least-squares formulation relies on the Cayley-Gibbs-Rodrigues (CGR) parameterization [26], [27] of a 3D rotation matrix and on the minimization of a multivariate rational function, whose degree is independent of the number of observed 3D lines. Compared to the solution based on the eigendecomposition of a large multiplication matrix in [21], [28], our approach is simpler yet provably optimal (in fact, the minima are certified to be global [29]). In summary, the original contributions of this paper can be listed as follows:

1) We propose the first visual gyroscope for event omnidirectional cameras only based on the detection and tracking of line features. Gyrevento is initialization-free, provides globally optimal certified solutions and is scalable (it is independent of the number of 3D lines observed by the camera).
2) We take advantage of the CGR parameterization of a 3D rotation to define a nonlinear least-squares cost function. This multivariate rational cost function is efficiently minimized with GloptiPoly [30], for global attitude estimation.
3) In our real-world experiments with a manipulator, for a given trajectory, we study the effect of camera velocity on the attitude estimation error. The corresponding omnidirectional event datasets have been publicly released.

We also describe a generic calibration procedure for event central panoramic cameras, which is of independent interest for the robotic and computer vision communities. An extended version of this paper, with additional theoretical results and numerical experiments with a car in CARLA simulator, is available at: http://home.mis.u-picardie.fr/~fabio/Gyrevento.html

The remainder of this paper is organized as follows. In Sect. II, we briefly introduce some mathematical tools for later reference. In Sect. III, the problem studied in the article is formulated and the different functional blocks of Gyrevento are described. The theory is validated via extensive numerical and hardware experiments in Sect. IV. Finally, in Sect. V, the main contributions of the paper are summarized and possible avenues for future research are outlined.

Notation: Throughout this paper, $\mathbf{I}_n$ denotes the $n \times n$ identity matrix, $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ the canonical basis for $\mathbb{R}^3$ and "$\triangleq$" stands for equality by definition. Finally, $\mathrm{SO}(n) \triangleq \{\mathbf{R} \in \mathbb{R}^{n\times n} : \mathbf{R}^T\mathbf{R} = \mathbf{I}_n, \det(\mathbf{R}) = 1\}$ is the special orthogonal group in dimension n.

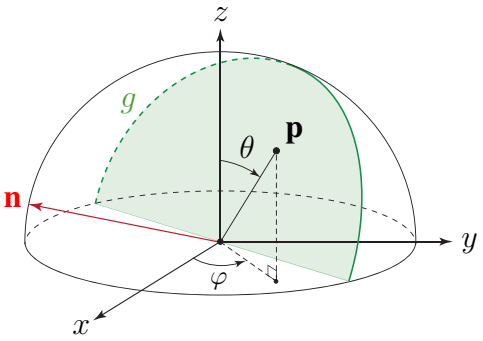


Figure 2. Great circle g (green) and its unit normal vector $\mathbf{n}$ (red). A generic point on the unit sphere is denoted by $\mathbf{p}$. For the sake of clarity, only the upper hemisphere is shown.

II. Preliminaries

A. CGR parameterization

Using the Cayley-Gibbs-Rodrigues (CGR) parameterization, a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ is expressed as

$$\mathbf{R}(\mathbf{s}) = \frac{\overline{\mathbf{R}}(\mathbf{s})}{1 + \mathbf{s}^T\mathbf{s}}, \quad \overline{\mathbf{R}}(\mathbf{s}) \triangleq (1 - \mathbf{s}^T\mathbf{s})\mathbf{I}_3 + 2[\mathbf{s}]_\times + 2\mathbf{s}\mathbf{s}^T, \quad (1)$$

where $\mathbf{s} = [s_1, s_2, s_3]^T$ is the vector of CGR parameters [31, p. 469], and

$$[\mathbf{s}]_\times = \begin{bmatrix} 0 & s_3 & -s_2 \\ -s_3 & 0 & s_1 \\ s_2 & -s_1 & 0 \end{bmatrix}, \quad (2)$$

is the corresponding skew-symmetric matrix. The rotation matrix $\mathbf{R}$ can be equivalently expressed as $\mathbf{R}(\mathbf{s}) = (\mathbf{I}_3 + [\mathbf{s}]_\times)(\mathbf{I}_3 - [\mathbf{s}]_\times)^{-1}$ which is generally known as Cayley's formula. The CGR parameterization is closely related to the other well-known attitude representations: unit quaternion and axis-angle [27].

B. Fitting great circles on the unit sphere

A great circle of a unit sphere (the intersection of the sphere and a plane that passes through the center point of the sphere), can be specified by two parameters α and β. These parameters correspond to the directional angles of the normal vector $\mathbf{n}$ to the plane containing the great circle in the Cartesian coordinate system whose origin is located at the center of the sphere [32, Sect. 4.1], see Fig. 2. A generic point $\mathbf{p}$ on the unit sphere can be expressed as $\mathbf{p} = [\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta]^T$ where $\theta \in [0, \pi]$ is the polar angle and $\varphi \in [0, 2\pi]$ is the azimuthal angle. Given a point $\mathbf{p}$ and a great circle g with unit normal vector $\mathbf{n} = [\sin\alpha\cos\beta, \sin\alpha\sin\beta, \cos\alpha]^T$, the distance from $\mathbf{p}$ to the plane containing g is given by $\delta = |\mathbf{p}^T\mathbf{n}|$. Let us now consider N points $\mathbf{p}_k$, $k \in \{1, \ldots, N\}$, on the unit sphere. The great circle fitting problem can be stated as the problem of finding a plane in order to minimize the sum of squares of distances between the N points and the plane. The objective function to minimize is then

$$V(\mathbf{n}) = \sum_{k=1}^{N} (\mathbf{p}_k^T\mathbf{n})^2,$$

where $\mathbf{n}$ is the normal vector to the plane containing the best-fit great circle. In other words, the optimal normal $\mathbf{n}$ (in the least-squares sense) is the one which minimizes the sum of squared

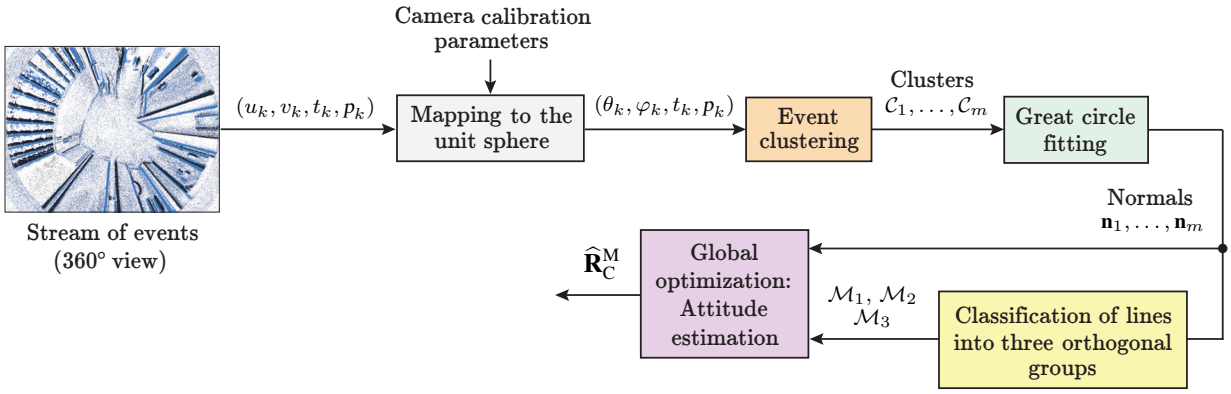


Figure 3. Flowchart of Gyrevento.

residuals $\delta_k^2 = (\mathbf{p}_k^T \mathbf{n})^2$. This problem can be converted into a standard eigenvalue problem. In fact, by introducing the matrix $\mathbf{A} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N]^T \in \mathbb{R}^{N\times 3}$, we can equivalently rewrite the objective function as

$$V(\mathbf{n}) = \mathbf{n}^T \mathbf{A}^T \mathbf{A}\, \mathbf{n}. \tag{3}$$

The solution for $\mathbf{n}$ is thus the eigenvector of matrix $\mathbf{B} \triangleq \mathbf{A}^T\mathbf{A} \in \mathbb{R}^{3\times 3}$ corresponding to its smallest eigenvalue. Having found the unit-norm vector $\mathbf{n}$, one may readily describe the great circle by its two angles α and β.

III. Design of the Visual Gyroscope

A. Problem formulation

Consider an event-based omnidirectional camera rigidly attached to a robotic platform. The robot moves in a man-made environment (Manhattan world) and the event camera observes line features in the 3D space. Our goal is to design a visual gyroscope which takes the asynchronous stream of events from the camera as input, and outputs an estimate of the orientation of camera frame $\{\mathrm{C}\}$ with respect to the world or Manhattan frame $\{\mathrm{M}\}$ (see Fig. 1 and the flowchart in Fig. 3).

A stream of asynchronous events generated by a neuromorphic camera can be mathematically represented as $\mathbf{e}_k = [u_k,\, v_k,\, t_k,\, p_k]^T$ [1], [13]. This 4-tuple describes an event occurring at time t_k (a.k.a. timestamp) at pixel $[u_k,\, v_k]^T$. The polarity $p_k \in \{-1, +1\}$ is the sign of brightness change detected by the camera since the last event at pixel $[u_k,\, v_k]^T$.

B. Mapping to the unit sphere

Thanks to their 360° FoV on the horizontal plane, omnidirectional cameras are ideal for fast-moving robots since the observed features are always visible. To describe these sensors, we used the unified central projection model of [25], which encompasses a breadth of cameras (perspective, catadioptric and fisheye [33]). Event $\mathbf{e}_k = [u_k,\, v_k,\, t_k,\, p_k]^T$ is thus mapped to $\mathbf{e}_k^{\ominus} = [\theta_k,\, \varphi_k,\, t_k,\, p_k]^T$ on the surface of the unit sphere of the unified model (see Fig. 4). The design of our visual gyroscope takes full advantage of the spherical representation. In fact, 3D lines in the Manhattan world, become great circles on the unit sphere and all parallel lines in the 3D space intersect at two points on the sphere, called *vanishing points* (green squares in Fig. 4). These points can be obtained by computing the cross product of the normal vectors to two great circles. With reference to Fig. 4, let $\mathbf{d}_i$ be the 3D direction of the i-th 3D line L_i expressed in the Manhattan frame $\{\mathrm{M}\}$, and $\mathbf{n}_i$ the normal vector to the corresponding great circle g_i on the unit sphere, expressed in the camera frame $\{\mathrm{C}\}$. Both $\mathbf{d}_i$ and $\mathbf{n}_i$ are unit vectors.

The mapping to the unit sphere requires the camera to be intrinsically calibrated, i.e. the parameters $\{f_u,\, f_v,\, \mathrm{u}_0,\, \mathrm{v}_0,\, \xi\}$ need to be known, where $(f_u,\, f_v)$ are the sizes of unit length in horizontal and vertical pixels, respectively, $(\mathrm{u}_0,\, \mathrm{v}_0)$ are the coordinates of the projection of the optical center of the camera onto the image plane in pixels, and ξ is the distance between the unit sphere's first projection center and the perspective second projection center of the camera [34].

C. Event clustering and great circle fitting

Our visual gyroscope relies on 3D lines (which map to great circles on the unit sphere), as primitives. To detect the circles from the continuous data stream, we take all the events with the same polarity over a small time window T. In other words, we slice the event cloud into batches, whose size depends on the camera's event rate. We then apply DBSCAN [35] to find m clusters of events, $\mathcal{C}_1, \ldots, \mathcal{C}_m$, which represent candidate great circles on the unit sphere. DBSCAN only requires three parameters: `minPts`, the minimum number of nearest neighbors required to form a dense region; ρ, the radius of a neighborhood with respect to a point, and `distFcn`, the distance function between two points. If the ρ-neighborhood of a point contains at least `minPts` neighbors, then DBSCAN identifies the point as the *center point*. Note that the choice of the distance function is strongly related to that of ρ, the most common metric being the Euclidean distance. However, since the events $\mathbf{e}_k$ are mapped to the unit sphere (cf. Sect. III-B), the *great-circle distance* is the natural choice in our case. More precisely, we set $\texttt{distFcn} = 1 - \cos\gamma$, where γ is the angle subtended by two points along a great arc of the unit sphere.

Once the m clusters have been estimated by DBSCAN, the procedure described in Sect. II-B is used to fit m great circles to them. Let $\mathbf{p}_k = [\sin\theta_k \cos\varphi_k,\, \sin\theta_k \sin\varphi_k,\, \cos\theta_k]^T$ be the 3D coordinates of event $\mathbf{e}_k^{\ominus}$ and let $\mathcal{C}_i = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N_i}\}$, $i \in \{1, \ldots, m\}$, be the i-th cluster. Then, the normal vector $\mathbf{n}_i$ to g_i is computed via the eigendecomposition of matrix $\mathbf{B}_i = \mathbf{A}_i^T \mathbf{A}_i$ where $\mathbf{A}_i = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N_i}]^T \in \mathbb{R}^{N_i\times 3}$.

D. Classification of 3D lines

To estimate the orientation of the camera frame with respect to the Manhattan frame, the knowledge of the normal vectors

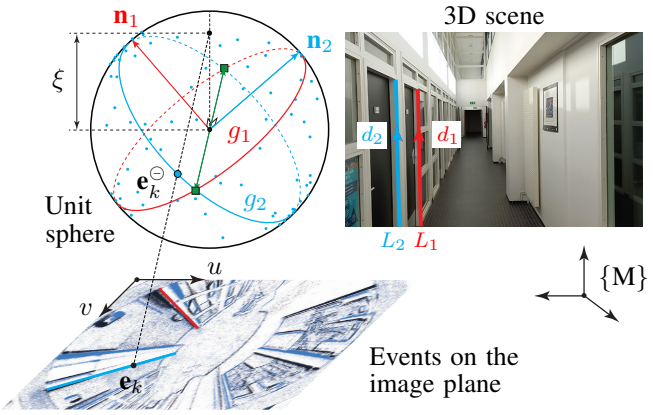


Figure 4. [Top right] Parallel lines L_1 (cyan) and L_2 (red) with directions $\mathbf{d}_1$ and $\mathbf{d}_2$ in the Manhattan frame {M}. [Left] Great circles g_1 and g_2 corresponding to L_1 and L_2, with normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ in the camera frame {C}. The two vanishing points are marked as green squares on the unit sphere.

to the great circles $\mathbf{n}_1, \ldots, \mathbf{n}_m$ alone, is not sufficient. In fact, a crucial step consists in partitioning the 3D lines in the Manhattan world, into three parallel and mutually-orthogonal groups $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$. As in prior research [21], [22], to classify the lines, we opted for a variant of RANSAC (generically known as adaptive RANSAC, such as 3-line RANSAC), which guarantees more accurate results.

Our variant of RANSAC serves the purpose of identifying the dominant directions. A dominant direction is characterized by the vanishing point which best describes one of the directions in the Manhattan frame. The first step consists in selecting all possible pairs of distinct great circles. For each pair, we find one of the vanishing points, by calculating the cross product of the normals to the two great circles (see the green squares in Fig. 4). Once all the vanishing points have been obtained, those that are sufficiently close to the directions estimated in the previous step, are grouped into three categories. The centroid of each group of vanishing points is computed, to make the algorithm less sensitive to outliers. Finally, to determine the three sets of normals associated with the three newly-established dominant directions, we choose those which are close enough to the great circles of these dominant directions (see Sect. IV for more details).

E. Global optimization: Attitude estimation

In the previous sections, we have seen how to estimate the normal vectors to the great circles and how to partition the 3D parallel lines into different orthogonal groups. The last module of our visual gyroscope takes these data as input, and provides an estimate $\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}}$ of $\mathbf{R}_{\mathrm{C}}^{\mathrm{M}} \in \mathrm{SO}(3)$, the rotation matrix representing the orientation of the camera in the Manhattan frame {M}, see Fig. 3. Following [21, Sect. 3], we can formulate this problem, in general, as

$$\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}} = \arg\min_{\mathbf{R}} \frac{1}{2}\sum_{i=1}^{m} \frac{1}{\sigma_i^2}\left(\mathbf{d}_i^T \mathbf{R}\, \mathbf{n}_i\right)^2, \quad \text{s.t. } \mathbf{R}^T\mathbf{R} = \mathbf{I}_3,\ \det(\mathbf{R}) = 1, \tag{4}$$

where σ_i is a positive weight which reflects the uncertainty in each line-normal observation, $\mathbf{d}_i$ is the 3D direction of line L_i in {M}, and $\mathbf{n}_i$ is the noisy estimate of the normal to the great circle g_i in {C} (cf. Sects. III-B and III-D). The nonlinear weighted constrained least-squares problem (4) (for $m \geq 3$) is known in the literature as the *orientation-from-line-correspondences problem*. In what follows, we will make the simplifying assumption that $\mathbf{d}_i \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ (i.e. we restrict ourselves to the three cardinal directions).

By using the CGR parameterization (cf. Sect. II-A), problem (4) can be equivalently rewritten as

$$\widehat{\mathbf{s}} = \arg\min_{\mathbf{s}} J, \quad J = \frac{1}{2}\sum_{i=1}^{m} \frac{1}{\sigma_i^2}\left(\mathbf{d}_i^T \mathbf{R}(\mathbf{s})\, \mathbf{n}_i\right)^2. \tag{5}$$

Note that compared to (4), the optimization problem (5) is *unconstrained* (the constraint is dropped since the CGR parameterization ensures that $\mathbf{R}(\mathbf{s})$ is a rotation matrix).

To algebraically find the global minimum of problem (5), the authors in [21], first determined all the critical points of J, by solving the optimality conditions and then chose the one(s) that minimize (5). The optimality conditions are three quintic polynomials in the variable $\mathbf{s}$, whose solutions comprise the critical points of (5). Directly solving these polynomial equations is challenging, and in [21], the authors computed the *multiplication matrix* (a generalization of the companion matrix to multivariate polynomial systems), whose eigenvalues are the roots of the associated (saturated) polynomial system. While this elegant method has the advantage of being not iterative and not requiring any initialization, the construction of this (40×40) matrix and the computation of its eigenvalues, are far from trivial.

To circumvent this difficulty, in this paper we propose to numerically solve problem (5) with *GloptiPoly*, a publicly-available optimization toolbox for Matlab [30]. GloptiPoly builds up a hierarchy of semidefinite programming (SDP) or linear matrix inequality (LMI) relaxations of the generalized problem of moments, whose associated monotone sequence of optimal values converges to the global optimum. Note that the cost function in (5) can be expressed as

$$J = \frac{g(\mathbf{s})}{h(\mathbf{s})} = \frac{\displaystyle\sum_{i=1}^{m} \frac{1}{\sigma_i^2}\left(\mathbf{d}_i^T \overline{\mathbf{R}}(\mathbf{s})\, \mathbf{n}_i\right)^2}{2(1 + \mathbf{s}^T\mathbf{s})^2},$$

where $g(\mathbf{s})$ and $h(\mathbf{s})$ are polynomials of degree 4 in $\mathbf{s}$. The minimization of a *rational function* as J, can also be formulated as a linear moment problem [36], [37], and GloptiPoly is able to extract a certifiably globally optimal solution (which corresponds to the best attitude estimation of the camera) from the solution of SDP relaxations. Once the optimal vector of CGR parameters $\widehat{\mathbf{s}}$ has been computed, the corresponding rotation matrix $\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}}$ can be determined via equation (1).

In contrast to [21], an explicit expression for the weights σ_i appearing in problem (5), is given in [28]. However, this leads to a computationally intractable optimization problem, since the degree of the rational cost function rapidly grows with each new summand. To get around this problem, the authors relaxed the formulation, by assuming that the variances σ_i^2 are approximately the same for all measurements. To conclude this section, two remarks are in order.

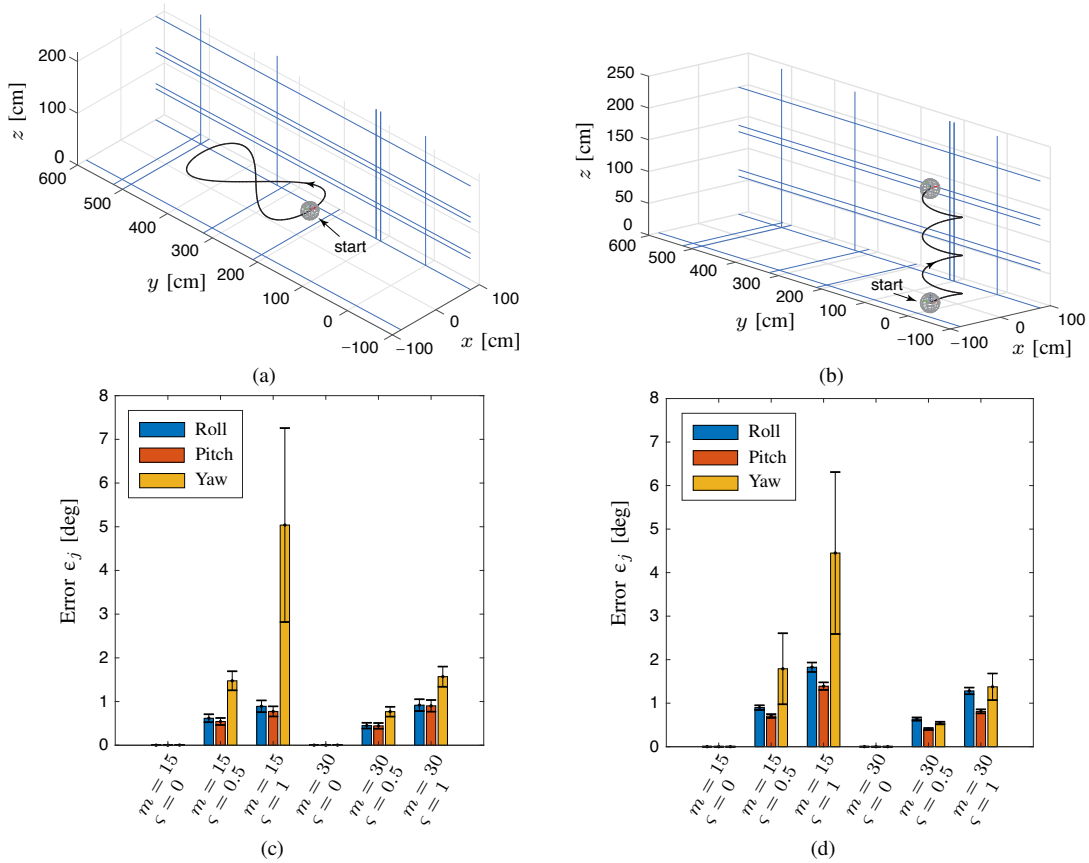


Figure 5. *Numerical simulations:* (a) Figure-of-eight trajectory; (b) Helicoidal trajectory; (c),(d) Mean and standard deviation of the estimation error $\epsilon_j = |\phi_j - \widehat{\phi}_j|$, $j \in \{1, 2, 3\}$, over 50 trials. The errors for different numbers of lines, $m \in \{15, 30\}$, and different levels of noise, $\varsigma^* = \kappa\varsigma$, $\kappa = \sin(3°)$, $\varsigma \in \{0, 0.5, 1\}$, are compared.

***Remark** 1: (Characterization of the solutions):* In [21, Sect. 3.5], the authors have studied the existence of solutions to problem (5), and determined the number of global minima. They have shown that regardless of the number of measurements, there exist at least four solutions for a camera's orientation from observation of lines with known directions in the Manhattan frame. Specifically, assuming noisy-free measurements in the camera frame, if $\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}\star}$ is one solution, then $\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}} = \mathbf{\Pi}\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}\star}$ with $\mathbf{\Pi} \in \{\mathbf{I}_3, \mathbf{R}_x(\pi), \mathbf{R}_y(\pi), \mathbf{R}_z(\pi)\}$, is a solution too, where $\mathbf{R}_x(\pi)$ denotes the 3×3 elementary rotation of an angle π about the x-axis. ◇

***Remark** 2: (Singular configurations):* When the rotation angle corresponding to $\mathbf{R} \in \mathrm{SO(3)}$ is equal to $\pm\pi$, the CGR parameterization is degenerate. In practice, even in the proximity of $\pm\pi$, we can have extremely large values for the CGR parameters. To overcome this problem, in [28], the authors simply proposed to rotate the measurements or 3D lines to an arbitrary (randomly generated) reference frame, find the global minimum (or minima), and then rotate the solution(s) back to the original frame. ◇

IV. Experimental validation

A. Numerical simulations: Global optimization

In our first test, we evaluated the accuracy and robustness of the global optimization module described in Sect. III-E (magenta block in Fig. 3), in a synthetic environment. To have a realistic setup, we manually extracted 30 lines (from windows, door frames, fixtures, etc.) in a $7 \times 2 \times 2.5$ m^3 hallway located in our laboratory building (5 lines are parallel to the x axis, 10 are parallel to the y axis, and 15 are parallel to the z axis). Two trajectories of the omnidirectional event camera have been considered in this environment. The first trajectory is a *figure-of-eight* (see Fig. 5(a)), with parametric representation

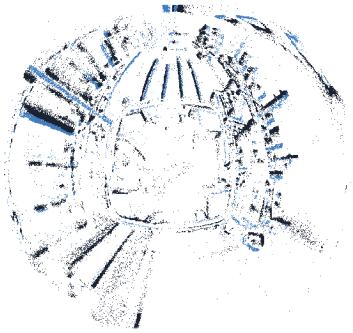
$$\begin{cases} x = 60\sin(2aj), \;\; y = 120\cos(aj) + 300, \;\; z = 60, \\ \phi_1 = \phi_2 = 0, \;\; \phi_3 = -\arctan\Big(\dfrac{\cos(2aj)}{\sin(aj)}\Big) - \dfrac{\pi}{2}, \\ j \in [-180°,\, 180°]. \end{cases} \tag{6}$$

where $(x,\, y,\, z)$ are the coordinates of the optical center of the camera in centimeters, $(\phi_1,\, \phi_2,\, \phi_3)$ are the roll, pitch and yaw angles of the camera in radians, and $a = \pi/180°$. The second trajectory is a *helix* (see Fig. 5(b)), expressed by

$$\begin{cases} x = 30\,(\sin(aj) - 1), \;\; y = 30\cos(aj), \;\; z = j/6, \\ \phi_1 = 50\,a(1 - e^{-j/180}), \;\; \phi_2 = -3\,a(1 - e^{-j/180}), \\ \phi_3 = aj, \quad j \in [0,\, 3 \times 360°]. \end{cases} \tag{7}$$

In Figs. 5(a),(b), a set of 15 lines, marked in blue, is shown. We used equation (6) to describe the planar motion of a camera mounted on a wheeled robot, and equation (7), the ascending motion of a quadrotor. We set the number of observed lines either to 15 or to 30. In the first case, to guarantee that the Manhattan world assumption is satisfied, 3 lines are imposed (one per coordinate axis), while the other 12 are drawn at

(a)



(b)

Figure 6. (a) Experimental setup: The catadioptric event camera is mounted on the end-effector of a Stäubli TX-60 robot and observes the parallel lines in the environment. (b) Event frame captured by the catadioptric event camera, 2.5 seconds after the beginning of Sequence 1a (a pixel is blue if the polarity $p_k = +1$, and black if $p_k = -1$).

random out of the remaining 27 lines. In the second case, all 30 lines are taken. The measurements of the normal vectors $\mathbf{n}_1, \ldots, \mathbf{n}_m$ were corrupted with additive zero-mean white Gaussian noise of increasing magnitude. The noise acting on the three components of the normal vectors is independent and the standard deviation $\varsigma^* = \kappa\varsigma$, $\kappa = \sin(3°)$, $\varsigma \in \{0,\, 0.5,\, 1\}$. Figs. 5(c),(d) report the mean and standard deviation of the angular estimation error of the camera over 50 trials, for the two trajectories and for different values of m and ς^*. More precisely, given $\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}} = \mathbf{R}_z(\widehat{\phi}_3)\mathbf{R}_y(\widehat{\phi}_2)\mathbf{R}_x(\widehat{\phi}_1)$, we computed the statistics of the errors $\epsilon_j = \left|\phi_j - \widehat{\phi}_j\right|$, $j \in \{1, 2, 3\}$, in degrees. Problem (5), with weights $\sigma_1 = \ldots = \sigma_m = 1$, was numerically solved with GloptiPoly, using the conic programming solver SeDuMi [38]. To single out the "best solution" among the four possible minima of the cost function J in (5) during camera motion (cf. Remark 1), we used the *chordal distance* $d_{\mathrm{chor}}[\,\cdot\,,\cdot\,]$ in SO(3) between the current and previous rotation estimate. Thus, at time $\ell \in \{1, 2, \ldots\}$, we compute

$$\begin{aligned} &\arg\min_{\mathbf{\Pi}} \;\; d_{\mathrm{chor}}[\mathbf{\Pi}\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}\star}(\ell),\, \widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}\star}(\ell-1)] \;= \\ &\arg\min_{\mathbf{\Pi}} \;\; \left\|\mathbf{\Pi}\widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}\star}(\ell) - \widehat{\mathbf{R}}_{\mathrm{C}}^{\mathrm{M}\star}(\ell-1)\right\|_{\mathrm{F}}, \end{aligned}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix. While this method is easy to implement, it is rather sensitive to the accuracy of the initial camera orientation's estimate.

From an inspection of Figs. 5(c),(d), we can notice that the worst mean estimation error is 5.04° (4.45°) for the figure-of-eight (helicoidal) trajectory. In both cases, it corresponds to the yaw angle: this is due to the unequal repartition of 3D parallel lines in the specific simulation environment considered. Moreover, for the same magnitude of noise, the larger the number of observed lines, the more accurate the visual gyroscope, as expected.

B. Real-world experiments: Event camera on a robot arm

a) Camera specifications : In our real-world experiments, we used a Prophesee Evaluation Kit 3 - HD (EVK3). This event camera has a resolution of 1280 × 720 pixels, a latency of 220 μs, a dynamic range exceeding 110 dB and a maximum bandwidth of 1.6 Gbps. An event $\mathbf{e}_k = [u_k,\, v_k,\, t_k,\, p_k]^T$ is encoded on 64 bits: 28 bits for u_k and v_k, 32 bits for t_k, and 4 bits for p_k. A catadioptric event camera was obtained by screwing a VStone VS-C450MRTK objective with hyperbolic mirror, on top of EVK3.

b) Camera calibration : The calibration of an event camera is more challenging than that of a traditional camera, and it is a subject of ongoing research. Some authors have proposed to use blinking patterns or active illumination with static checkerboards, for the calibration of perspective event cameras. Recently, a dynamic and a learning-based approach have been described in [39] and [40], respectively, to simplify and make the calibration procedure more robust (see also [41] for fast calibration). However, with the exception of [20], where a fisheye event camera is considered, we are no aware of any other generic calibration algorithm for *omnidirectional event cameras*. To calibrate different typologies of panoramic event sensors (catadioptric, fisheye, etc.), we developed a protocol which consists in filming a flashing pattern in front of the static camera. A grayscale image is generated by accumulating the events over a fixed time window (that the user can select), which, in turn, is exploited to estimate the intrinsic parameters $\{f_u,\, f_v,\, \mathrm{u}_0,\, \mathrm{v}_0,\, \xi\}$ of the camera via conventional calibration algorithms (in our experiments, we used HySCaS [42]).

c) Experimental setup : To validate the overall pipeline reported in Fig. 3, experiments were carried out in a laboratory setting. Our catadioptric event camera was mounted on the end-effector of a 6-axis Stäubli TX-60 robot (see Fig. 6). The rotary encoders on the joints of the robot provide accurate measurements of angular positions and velocities at 250 Hz. The forward kinematics equations of the robot were used to compute the position and orientation of the end-effector (our ground truth), from specific values of joint parameters. The stream of events and the measurements from the encoders were synchronized offline, and the estimated orientation of the camera was updated at a fixed (arbitrary) frequency of 25 Hz. As in Sect. IV-A, problem (5), with weights $\sigma_1 = \ldots = \sigma_m = 1$, was numerically solved with GloptiPoly, using SeDuMi.

We evaluated the accuracy of Gyrevento, by considering two sequences of growing complexity. For a given trajectory, to study the effect of speed, three sets of camera velocities were tested. In what follows, we use $\boldsymbol{v}_C^{\max} = [v_x^{\max},\, v_y^{\max},\, v_z^{\max},\, \omega_x^{\max},\, \omega_y^{\max},\, \omega_z^{\max}]^T$ to denote the maximum velocity of the camera (expressed in the camera frame), where the linear velocities are in mm/s and the angular velocities in deg/s.

- **Sequence 1**: 1-DoF motion (pure rotation about the optical axis of the camera):
  a) $\boldsymbol{v}_C^{\max} = [0, 0, 0, 0, 0, 47.8]^T$, duration 18.20 s, T = 10 ms, r_g = 30° (see Fig. 6(b)),

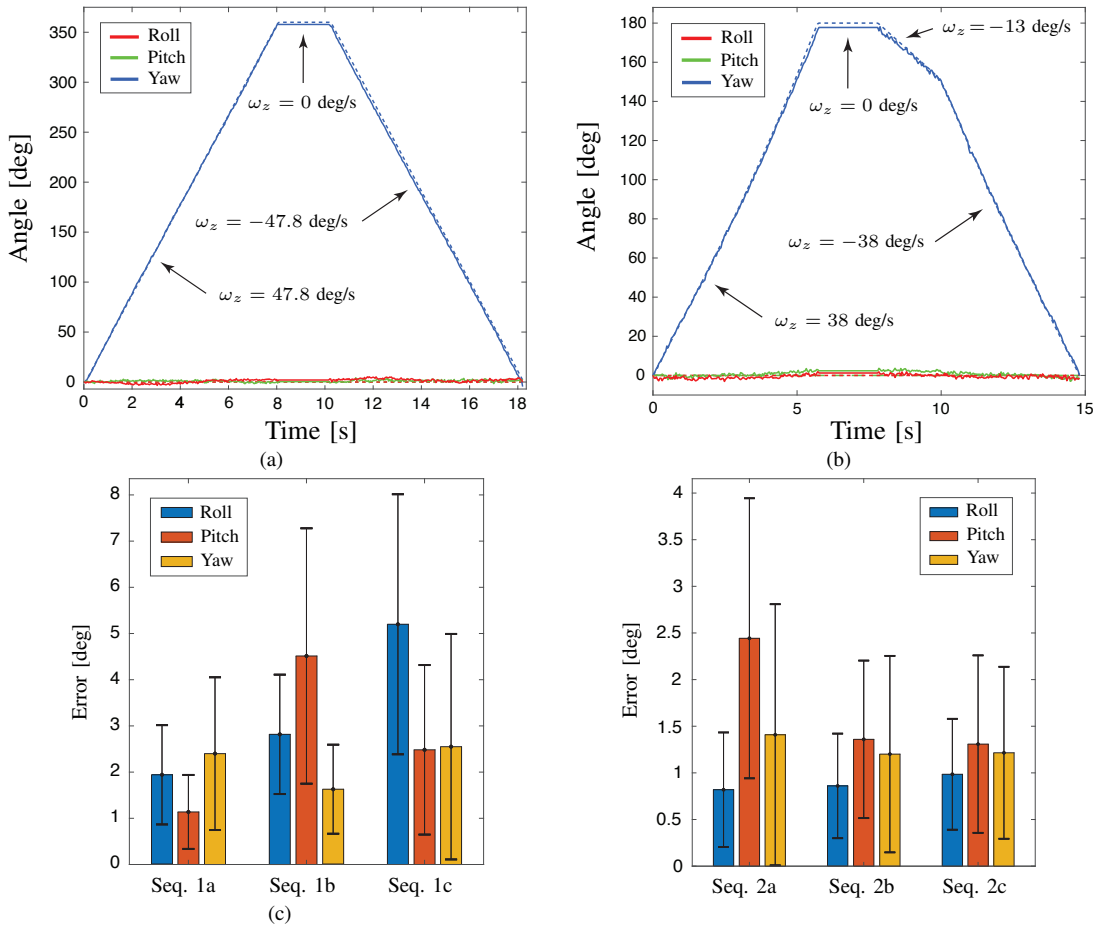


Figure 7. *Experimental results*: Actual orientation of the camera in degrees (dashed) and orientation estimated by Gyrevento (solid): (a) Sequence 1a, (b) Sequence 2c. Mean and standard deviation of the angular estimation error in degrees: (c) Sequences 1a-1c, (d) Sequences 2a-2c.

b) $\boldsymbol{v}_C^{\max} = [0, 0, 0, 0, 0, 91.8]^T$, duration 10.24 s, $T =$ 10 ms, r_g = 30°,
c) $\boldsymbol{v}_C^{\max} = [0, 0, 0, 0, 0, 136.6]^T$, duration 7.56 s, $T = 5$ ms, r_g = 45°.

- **Sequence 2**: 4-DoF motion (rotation about the optical axis of the camera and 3D translation in the $[-20, 290] \times [0, 230] \times [380, 670]$ mm^3 range):
  a) $\boldsymbol{v}_C^{\max} = [46.6, 55.6, 36.3, 0, 0, 19.2]^T$, duration 27.52 s, $T = 22.5$ ms, r_g = 18°,
  b) $\boldsymbol{v}_C^{\max} = [69.4, 80.8, 56.0, 0, 0, 28.8]^T$, duration 19.08 s, $T = 20$ ms, r_g = 20°,
  c) $\boldsymbol{v}_C^{\max} = [92.1, 108.0, 72.0, 0, 0, 38]^T$, duration 14.80 s, $T = 15$ ms, r_g = 22.5°.

The intrinsic parameters of the camera are, in pixels, $f_u = 310.2723$, $f_v = 308.8265$, $\mathrm{u}_0 = 601.7725$, $\mathrm{v}_0 = 372.3330$, and $\xi = 1.1099$. Moreover, $\rho = 0.75°$, $\texttt{minPts} = 3$, $r_g = 5°$, $L_{\text{arc}} = 7°$ and $L_d = 1°$. The last three parameters, r_g, L_{arc} and L_d, have been introduced for enhanced flexibility in line detection. The first parameter, r_g, is the angle of the spherical cone containing the vanishing points, with apex at the center of the sphere, used to identify the new dominant directions in RANSAC (cf. Sect III-D). L_{arc} denotes the minimum length of an arc on the unit sphere (corresponding to a straight line in the Manhattan frame) and L_d is the maximum thickness of an arc. We used a circular mask to discard the non-informative pixels in the outer rim of the omnidirectional image.

d) Discussion : Figs. 7(a),(b) show the time evolution of the actual orientation of the camera (dashed) and the orientation estimated by Gyrevento (solid) in Sequence 1a and Sequence 2c, respectively. The impact of camera velocity on the attitude estimation error is evaluated in Fig. 7(c) for Sequence 1 and in Fig. 7(d) for Sequence 2. For the sake of clarity, the outliers are not shown in the box charts in Figs. 7(c),(d): they are 6.7°, 9.2°, 12° and 11.5°, 8.2°, 4.0°, respectively (the inertia of the camera during the rapid accelerations/decelerations of the manipulator, is the main responsible for them). Figs. 7(c),(d) indicate that Gyrevento works best for angular velocities in the $[20, 50]$ deg/s range (the maximum mean error is less than 2.5°). However, it continues to deliver satisfactory performances for speeds up to 136 deg/s. Gyrevento can thus be potentially used onboard agile robots (e.g. quadrotors). The mean error in Sequence 2 is about twice as small as that in Sequence 1, since the 4-DoF camera motion offers richer visual information (and, ultimately, more exploitable parallel lines).

For additional results with 3-DoF and 6-DoF camera motions, the reader is referred to the Supplementary Material.

V. Conclusion and Future Work

In this paper, we have presented Gyrevento, the first omnidirectional visual gyroscope for event cameras only based on the observation of 3D lines in a structured environment.

The gyroscope has been successfully validated via simulation and real-world experiments with a robot manipulator. The experimental results have also shed light on the impact of camera velocity on the attitude estimation error.

In future research, we will work towards the design of an event-based omnidirectional visual odometer and we plan to relax the Manhattan world assumption about the existence of three mutually orthogonal directions (cf. Atlanta world assumption [24]). It would also be interesting to automatically incorporate the Manhattan constraints into the event-clustering step, which is, at present, the computational bottleneck of Gyrevento (its complexity is $O(n_e \log(n_e))$, where n_e is the number of detected events). On the other hand, the complexity of the great circle fitting and classification steps (cf. Fig. 3) is $o(n_e)$ and $O(m^2)$, respectively, with $n_e \gg m$. To robustify our multi-line tracker, we could exploit the polarity of the events and the spatio-temporal variant of DBSCAN, ST-DBSCAN [43]. Finally, in future work, we will weigh the pros and cons of other vectorial parameterizations of a 3D rotation [27], besides CGR.

SUPPLEMENTARY MATERIAL

The event datasets, calibration parameters, and videos relative to the experiments in Sect. IV and to additional tests performed with CARLA simulator (3-DoF motion) and with an IMU rigidly attached to the catadioptric event camera (6-DoF motion, Sequence 3), are available at:

http://home.mis.u-picardie.fr/˜fabio/Gyrevento.html

REFERENCES

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A.J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-Based Vision: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(1):154–180, 2022.

[2] D. Gehrig and D. Scaramuzza. Low-latency automotive vision with event cameras. *Nature*, 629(8014):1034–1040, 2024.

[3] H. Kim, S. Leutenegger, and A.J. Davison. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. In *Proc. Europ. Conf. Comput. Vis.*, pages 349–364, 2016.

[4] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza. EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Rob. Autom. Lett.*, 2(2):593–600, 2017.

[5] A.Z. Zhu, N. Atanasov, and K. Daniilidis. Event-based Visual Inertial Odometry. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 5391–5399, 2017.

[6] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza. Continuous-Time Visual-Inertial Odometry for Event Cameras. *IEEE Trans. Robot.*, 34(6):1425–1440, 2018.

[7] H. Kim and H.J. Kim. Real-Time Rotational Motion Estimation With Contrast Maximization Over Globally Aligned Events. *IEEE Rob. Autom. Lett.*, 6(3):6016–6023, 2021.

[8] J. Hidalgo-Carrió, G. Gallego, and D. Scaramuzza. Event-aided Direct Sparse Odometry. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 5781–5790, 2022.

[9] D. Liu, A. Parra, Y. Latif, B. Chen, T.-J. Chin, and I. Reid. Asynchronous Optimisation for Event-based Visual Odometry. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 9432–9438, 2022.

[10] Y.F. Zuo, J. Yang, J. Chen, X. Wang, Y. Wang, and L. Kneip. DEVO: Depth-Event Camera Visual Odometry in Challenging Conditions. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2179–2185, 2022.

[11] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck. ELiSeD – An Event-Based Line Segment Detector. In *Proc. 2nd Int. Conf. Event-based Contr. Comm. and Sign. Proc.*, 2016.

[12] L. Everding and J. Conradt. Low-Latency Line Tracking Using Event-Based Dynamic Vision Sensors. *Front. Neurorobot.*, 12:4, 2018.

[13] D.R. Valeiras, X. Clady, S.-H. Ieng, and R. Benosman. Event-Based Line Fitting and Segment Detection Using a Neuromorphic Visual Sensor. *IEEE Trans. Neural Networks Learn. Syst.*, 30(4):1218–1230, 2019.

[14] L. Gao, H. Su, D. Gehrig, M. Cannici, D. Scaramuzza, and L. Kneip. A 5-Point Minimal Solver for Event Camera Relative Motion Estimation. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 8049–8059, 2023.

[15] L. Gao, D. Gehrig, H. Su, D. Scaramuzza, and L. Kneip. An N-Point Linear Solver for Line and Motion Estimation with Event Cameras. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 14596–14605, 2024.

[16] X. Lu, Y. Zhou, J. Niu, S. Zhong, and S. Shen. Event-based Visual Inertial Velometer. In *Proc. Robotics: Science and Systems*, 2024. Paper n. 88.

[17] W. Chamorro, J. Solà, and J. Andrade-Cetto. Event-based Line SLAM in Real-time. *IEEE Rob. Autom. Lett.*, 7(3):8146–8153, 2022.

[18] S. Schraml, A.N. Belbachir, and H. Bischof. An Event-Driven Stereo System for Real-Time 3-D 360° Panoramic Vision. *IEEE Trans. Ind. Electron.*, 63(1):418–428, 2016.

[19] C. Reinbacher, G. Munda, and T. Pock. Real-Time Panoramic Tracking for Event Cameras. In *Proc. IEEE Int. Conf. Comput. Photog.*, 2017.

[20] H. Cho, J. Jeong, and K.-J. Yoon. EOMVS: Event-Based Omnidirectional Multi-View Stereo. *IEEE Rob. Autom. Lett.*, 6:6709–6716, 2021.

[21] F.M. Mirzaei and S.I. Roumeliotis. Optimal Estimation of Vanishing Points in a Manhattan World. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2454–2461, 2011.

[22] J.-C. Bazin, C. Demonceaux, P. Vasseur, and I. Kweon. Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. *Int. J. Robot. Res.*, 31(1):63–81, 2012.

[23] P. Kim, H. Li, and K. Joo. Quasi-Globally Optimal and Real-Time Visual Compass in Manhattan Structured Environments. *IEEE Rob. Autom. Lett.*, 7(2):2613–2620, 2022.

[24] D. Yan, H. Jiang, T. Li, and C. Shi. Efficient Vanishing Point Estimation for Accurate Camera Rotation Estimation in Indoor Environments. *IEEE Rob. Autom. Lett.*, 8(11):6899–6906, 2023.

[25] C. Geyer and K. Daniilidis. A Unifying Theory for Central Panoramic Systems and Practical Implications. In *Proc. Europ. Conf. Comput. Vis.*, pages 445–461, 2000.

[26] J.A. Hesch and S.I. Roumeliotis. A Direct Least-Squares (DLS) Method for PnP. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 383–390, 2011.

[27] O.A. Bauchau and L. Trainelli. The Vectorial Parameterization of Rotation. *Nonlinear Dyn.*, 32(1):71–92, 2003.

[28] F.M. Mirzaei and S.I. Roumeliotis. Globally Optimal Pose Estimation from Line Correspondences. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 5581–5588, 2011.

[29] F. Bugarin, D. Henrion, and J.B. Lasserre. Minimizing the sum of many rational functions. *Math. Prog. Comp.*, 8(1):83–111, 2016.

[30] D. Henrion, J.-B. Lasserre, and J. Löfberg. GloptiPoly 3: moments, optimization and semidefinite programming. *Optim. Methods Software*, 24(4-5):761–779, 2009.

[31] M.D. Shuster. A Survey of Attitude Representations. *J. Astronaut. Sci.*, 41(4):439–517, 1993.

[32] X. Ying, Z. Hu, and H. Zha. Fisheye Lenses Calibration Using Straight-Line Spherical Perspective Projection Constraint. In *Proc. Asian Conf. Comput. Vis.*, pages 61–70, 2006.

[33] J. Courbon, Y. Mezouar, and P. Martinet. Evaluation of the Unified Model of the Sphere for Fisheye Cameras in Robotic Applications. *Adv. Robotics*, 26(8–9):947–967, 2012.

[34] J.P. Barreto. A unifying geometric representation for central projection systems. *Comput. Vis. Image Underst.*, 103(3):208–217, 2006.

[35] E. Schubert, J. Sander, M. Ester, H.P. Kriegel, and X. Xu. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.*, 42(3), 2017. Article n. 19.

[36] D. Jibetean and E. de Klerk. Global optimization of rational functions: a semidefinite programming approach. *Math. Program.*, 106(1), 2006.

[37] F. Kahl and D. Henrion. Globally Optimal Estimates for Geometric Reconstruction Problems. *Int. J. Comput. Vision*, 74(1):3–15, 2007.

[38] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Software*, 11:625–653, 1999.

[39] K. Huang, Y. Wang, and L. Kneip. Dynamic Event Camera Calibration. In *Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst.*, pages 7021–7028, 2021.

[40] M. Muglikar, M. Gehrig, D. Gehrig, and D. Scaramuzza. How to Calibrate Your Event Camera. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop*, pages 1403–1409, 2021.

[41] M. Salah, A. Ayyad, M. Humais, D. Gehrig, A. Abusafieh, L. Seneviratne, D. Scaramuzza, and Y. Zweiri. E-Calib: A Fast, Robust and Accurate Calibration Toolbox for Event Cameras. *IEEE Trans. Image Process.*, 33:3977–3990, 2024.

[42] G. Caron and D. Eynard. HySCaS: Hybrid Stereoscopic Calibration Software. [Web] https://mis.u-picardie.fr/~g-caron/software, 2011.

[43] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data Knowl. Eng.*, 60(1):208–221, 2007.