Continuous Event-Line Constraint for Closed-Form Velocity Initialization

Xin Peng*[123]
pengxin1@shanghaitech.edu.cn
Wanting Xu*[123]
xuwt@shanghaitech.edu.cn
Jiaqi Yang[123]
yangjq@shanghaitech.edu.cn
Laurent Kneip[1]
lkneip@shanghaitech.edu.cn

[1] Mobile Perception Lab
ShanghaiTech University
Shanghai, China

[2] Shanghai Institute of Microsystem and Information Technology
Chinese Academy of Sciences
Shanghai, China

[3] University of Chinese Academy of Sciences
Beijing, China

Abstract

Event cameras trigger events asynchronously and independently upon a sufficient change of the logarithmic brightness level. The neuromorphic sensor has several advantages over standard cameras including low latency, absence of motion blur, and high dynamic range. Event cameras are particularly well suited to sense motion dynamics in agile scenarios. We propose the continuous event-line constraint, which relies on a constant-velocity motion assumption as well as trifocal tensor geometry in order to express a relationship between line observations given by event clusters as well as first-order camera dynamics. Our core result is a closed-form solver for up-to-scale linear camera velocity with known angular velocity. Nonlinear optimization is adopted to improve the performance of the algorithm. The feasibility of the approach is demonstrated through a careful analysis on both simulated and real data.

1 INTRODUCTION

Event Cameras, such as the DVS [18], are bio-inspired visual sensors that differ substantially from traditional frame-based cameras. The pixels of an event camera operate asynchronously and trigger an event whenever there is sufficient change in the sensed logarithmic brightness level. More specifically, if the change of logarithmic brightness $L(\mathbf{x},t) \doteq \log I(\mathbf{x},t)$ at pixel $\mathbf{x} \doteq (x,y)^\mathsf{T}$ on the image plane surpasses a threshold C, the event camera will output a four-tuple signal $e = \{x,\ y,\ t,\ s\}$ where t is a timestamp and s is a binary polarity indicating whether the brightness has increased or decreased. Therefore, each pixel has its own sampling rate and outputs data proportionally to the amount of motion between camera and scene and in dependence of the gradient of the visual input. An event camera does not produce images at a constant rate, but rather a stream of asynchronous, sparse events in a space-time volume with approximately microsecond time resolution.

* indicates equal contribution.

Due to its exact nature, event cameras have several advantages over standard cameras including low latency ($\tilde{1}\mu$s), absence of motion blur, high dynamic range (140 dB vs 60 dB for traditional cameras [9]), and low power consumption. These beneficial properties enable an event camera to tackle vision tasks even in challenging conditions such as increased agility or low illumination conditions. One of the critical applications of an event camera is ego-motion estimation given that existing pipelines based on standard camera easily fail under high-speed motion or challenging illumination [15, 23]. However, the technical obstacle for event-based motion estimation is the fact that events are asynchronous and do not communicate absolute intensity information. Traditional motion estimation algorithms are therefore not appropriate and novel algorithms are needed.

Most of the state-of-the-art works in event-based motion estimation rely on learning-based approaches [16], filter-based methods [8, 14] and optimization methods [9, 20, 29]. Learning-based approaches need a huge amount of data to train the network, and filter-based methods are computationally complex and often need an initial guess. Since most problems are nonlinear, the results of optimization methods highly depend on a good initial guess. [19, 24] provide globally optimal solvers, which do not rely on good initial guess however they are computationally demanding and limited to homography environments. However, the methods are computationally demanding and limited to homography scenarios. Line features have already been used in event-based structure-from-motion frameworks. An example is given by Hough^{2}Map [26] which detects, tracks, and triangulates general lines. Other methods require highly artificial, black-and-white textures [21]. Brändli et al. [6] propose Event-Based Line Segment Detector (ELiSeD), which adopts the idea behind the LSD algorithm [28] to the event-based case.The method performs incremental event-based detection and tracking of lines in arbitrary scenes, but is not yet validated in the context of a full structure-from-motion framework. DVS sensors are often equipped with an Inertial Measurement Unit (IMU) (i.e. DAVIS 240 [4]), which is why researchers have also considered event-based visual-inertial odometry [17, 22, 27] In particular, Le Gentil et al. [17]introduce a line-based event-inertial odometry framework. In their event-based line tracking front-end, they draw concepts from [5] and [7], and detect line segments as locally spatio-temporal planar patches.

A critical concern in inertial odometry frameworks is given by bootstrapping. Our method aims at linear velocity initialization, which can neither be obtained directly from IMU, nor an event camera. Most existing event-inertial odometry algorithms pay little attention to the initialization question. Though using the assumption of known angular velocities, our work is the first to focus on linear velocity initialization and proposes a novel closed-form solver. It may be used to bootstrap other event-based inertial odometry frameworks, and in particular supports fusion at the level of velocities rather than positions. This has the advantage of requiring only single integration of inertial signals. The main contributions are listed as follows:

- We make use of trifocal tensor geometry [11] to formulate the relationship between events, lines and the ego-motion of the event camera, the so-called Continuous Event-Line Constraint (CELC).
- To the best of our knowledge, this leads to the first closed-form translational velocity solver. It relies on the assumptions of known angular velocity and constant speed resulting in a linear constraint, and furthermore enables nonlinear optimization to improve performance.
- Important steps towards a DVS pendant of epipolar geometry and relative motion es-

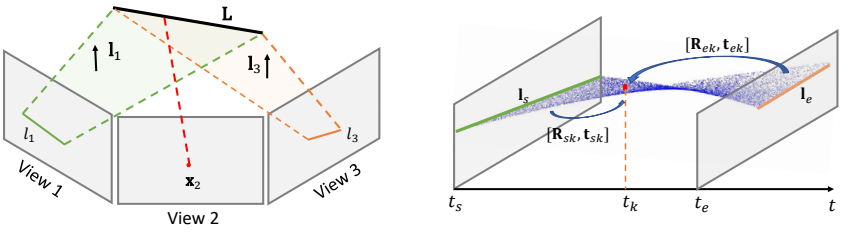


Figure 1: Geometry of trifocal geometry.

timation for normal cameras.

The paper is organized as follows. Section 2 reviews the general idea of trifocal tensor geometry with lines and introduces the continuous event-line constraint. In Section 3, we employ CELC for linear velocity estimation with a closed-form solution, and provide all implementation details. In Section 4, we analyze and evaluate the proposed algorithm via both simulated and real experiments. Section 5 gives final remarks.

2 Continuous Event-Line Constraint

We give a review of the trifocal tensor with standard cameras. We then derive the continuous event-line constraint (CELC) for an event camera which indicates the relationship between events, lines and camera dynamics.

2.1 Review of Trifocal Tensor

The trifocal tensor plays an analogous role in three views to that played by the fundamental matrix in two [□]. It encapsulates all the (projective) geometric relations between three views including the incidence relationship of three corresponding points, three corresponding lines, and point with line incidence relations. The line-point-line incidence relation is the most relevant for our work.

Lets denote a 3D line $\mathbf{L}$, its two corresponding line projections l_1 and l_3 in views 1 and 3, respectively, and an image point $\mathbf{x}_2$ in view 2 which is the projection of a 3D point on $\mathbf{L}$. The geometry is illustrated in Fig. 1(a). Let's define the second view as the reference view. We furthermore define $[\mathbf{R}_{12}|\mathbf{t}_{12}]$ as the Euclidean transformation parameters from view 2 to view 1, and $[\mathbf{R}_{32}|\mathbf{t}_{32}]$ as the Euclidean transformation parameters from view 2 to view 3. In the calibrated case, the line-point–line incidence relation is given by

$$\mathbf{f}_2^{\mathsf{T}}(\mathbf{l}_1^{\mathsf{T}}[\mathbf{T}_1,\mathbf{T}_2,\mathbf{T}_3]\mathbf{l}_3) = 0, \tag{1}$$

where $\mathbf{f}_2$ is the bearing vector corresponding to pixel $\mathbf{x}_2$ in view 2, and $\mathbf{l}_i = \mathbf{K}^{\mathsf{T}} l_i$ with $i = 1,3$ are the normal vectors of the planes crossing $\mathbf{L}$ and the camera centers of views 1 and 3, respectively. $[\mathbf{T}_1,\mathbf{T}_2,\mathbf{T}_3]$ defines the trifocal tensor, and the formulation for the 3×3 matrices $\mathbf{T}_i$ is given by

$$\mathbf{T}_i = \mathbf{r}_i^{12}\mathbf{t}_{32}^{\mathsf{T}} - \mathbf{t}_{12}\mathbf{r}_i^{32\mathsf{T}},\ i = 1,2,3, \tag{2}$$

where $\mathbf{r}_i^{12}$ denotes the i-th column of $\mathbf{R}_{12}$, and $\mathbf{r}_i^{32}$ the i-th column of $\mathbf{R}_{32}$. For further details on the trifocal tensor including its derivation, the reader is kindly referred to Chapter 15 of [□].

2.2 Continuous Event-Line Constraint – CELC

Our task consists of event camera ego-motion estimation. Our assumption is that events are mostly triggered by the reprojection of sharp appearance and occlusion edges, which—for the sake of a simplified derivation—are furthermore assumed to be straight in 3D. Note that this assumption may not be limiting, as man-made environments often present themselves in a form where the majority of such edges are indeed straight.

We detect continuous line projections from event streams and figure out the relation between those events, the underlying 3D lines, and dynamic motion parameters by using the aforementioned trifocal tensor relations. The continuous set of events triggered by the projection of a straight 3D line $\mathbf{L}$ under motion forms a cluster of events $\mathcal{E}$ in a twisted manifold-like shape. As denoted by the green and red lines in Figure 1(b), let the two lines l_1 and l_3 represent the reprojection of the 3D line $\mathbf{L}$ at timestamps t_s and t_e, respectively. As introduced in Sec. 2.1, for an event $e_k \in \mathcal{E}$, l_1 and l_3 must then satisfy the trifocal relation (1). Here, let's define $[\mathbf{T}_1^k, \mathbf{T}_2^k, \mathbf{T}_3^k]$ as the trifocal tensor for the k-th event. The trifocal tensor is constructed by the transformation of the camera from t_k to t_s and the transformation from t_k to t_e. The trifocal tensor will be different for each individual event. Note that rather than introducing an individual rotation and translation for each event—which would introduce too many unknowns—, we make use of a locally constant velocity assumption and parameterize the relative translation and rotation as a continuous time function of the linear velocity $\mathbf{v}$ and angular velocity $\boldsymbol{\omega}$. Hence, the rotation $\mathbf{R}_{sk}$ from time t_k to time t_s can be represented by the continuous time function

$$\begin{aligned} \mathbf{R}_{sk} &= \exp(\hat{\boldsymbol{\omega}}(t_k - t_s)) \\ &= \cos(\theta)\mathbf{I} + (1-\cos(\theta))\mathbf{a}\mathbf{a}^\mathsf{T} + \sin(\theta)\hat{\mathbf{a}}, \end{aligned} \tag{3}$$

where $(\mathbf{a}, \theta)$ are the axis-angle parameters of the rotation $\mathbf{R}_{sk}$. The translation $\mathbf{t}_{sk}$ from time t_k to time t_s is given by

$$\begin{aligned} \mathbf{t}_{sk} &= \mathbf{J}_{sk}\mathbf{v}(t_k - t_s), \\ \text{where } \mathbf{J}_{sk} &= \frac{\sin(\theta)}{\theta}\mathbf{I} + (1 - \frac{\sin(\theta)}{\theta})\mathbf{a}\mathbf{a}^\mathsf{T} + \frac{1-\cos\theta}{\theta}\hat{\mathbf{a}}. \end{aligned} \tag{4}$$

By replacing t_s with t_e in equations (3) and (4), we can obtain $\mathbf{R}_{ek}$ and $\mathbf{t}_{ek}$. Based on (3) and (4), we finally obtain the continuous time formulation of the trifocal tensor

$$\begin{aligned} \mathbf{T}_i &= \mathbf{r}_i^{sk}\mathbf{t}_{ek}^\mathsf{T} - \mathbf{t}_{sk}\mathbf{r}_i^{ek\mathsf{T}},\ i = 1,2,3, \\ &= \mathbf{r}_i^{sk}(\mathbf{J}_{ek}\mathbf{v}(t_k - t_e))^\mathsf{T} - (t_k - t_s)\mathbf{J}_{sk}\mathbf{v}\mathbf{r}_i^{ek\mathsf{T}}. \end{aligned} \tag{5}$$

Using equation (5) and applying simple matrix multiplication, we obtain

$$\begin{aligned} \mathbf{l}_1^\mathsf{T}\mathbf{T}_i\mathbf{l}_3 &= \mathbf{l}_1^\mathsf{T}\left[(t_k - t_e)\mathbf{r}_i^{sk}\mathbf{v}^\mathsf{T}\mathbf{J}_{ek}^\mathsf{T} - (t_k - t_s)\mathbf{J}_{sk}\mathbf{v}\mathbf{r}_i^{ek\mathsf{T}}\right]\mathbf{l}_3 \\ &= (t_k - t_e)\mathbf{l}_1^\mathsf{T}\mathbf{r}_i^{sk}\mathbf{l}_3^\mathsf{T}\mathbf{J}_{ek}\mathbf{v} - (t_k - t_s)\mathbf{l}_3^\mathsf{T}\mathbf{r}_i^{ek}\mathbf{l}_1^\mathsf{T}\mathbf{J}_{sk}\mathbf{v}. \end{aligned} \tag{6}$$

The continuous event-line constraint (CELC) for the k-th event is obtained by combining (6) and (1), thus resulting in

$$\mathbf{f}_k^\mathsf{T}\mathbf{B}_k\mathbf{v} = 0, \tag{7}$$

where

$$\mathbf{B}_k = \begin{bmatrix} (t_k - t_e)\mathbf{l}_1^\mathsf{T}\mathbf{r}_1^{sk}\mathbf{l}_3^\mathsf{T}\mathbf{J}_{ek} - (t_k - t_s)\mathbf{l}_3^\mathsf{T}\mathbf{r}_1^{ek}\mathbf{l}_1^\mathsf{T}\mathbf{J}_{sk} \\ (t_k - t_e)\mathbf{l}_1^\mathsf{T}\mathbf{r}_2^{sk}\mathbf{l}_3^\mathsf{T}\mathbf{J}_{ek} - (t_k - t_s)\mathbf{l}_3^\mathsf{T}\mathbf{r}_2^{ek}\mathbf{l}_1^\mathsf{T}\mathbf{J}_{sk} \\ (t_k - t_e)\mathbf{l}_1^\mathsf{T}\mathbf{r}_3^{sk}\mathbf{l}_3^\mathsf{T}\mathbf{J}_{ek} - (t_k - t_s)\mathbf{l}_3^\mathsf{T}\mathbf{r}_3^{ek}\mathbf{l}_1^\mathsf{T}\mathbf{J}_{sk} \end{bmatrix}. \tag{8}$$

The incidence relation expresses the intrinsic relationship between events generated by a 3D line and first order camera dynamics. Considering the transformation of lines and trifocal tensor geometry, $\tilde{\mathbf{l}}_2 = \mathbf{B}_k\mathbf{v}$ represents the projected line in view k generated by reference lines $\mathbf{l}_1$ and $\mathbf{l}_3$ and motion dynamics. Any event $\mathbf{e}_k$ triggered by the same line should lie on $\tilde{\mathbf{l}}_2$, i.e. $\mathbf{f}_k^\top\tilde{\mathbf{l}}_2 = 0$. As scale is unobservable in the monocular setting, the unknown motion parameters $\boldsymbol{\omega}$ and $\mathbf{v}$ actually make up for only 5-DoF. However, equation (3) and (4) are nonlinear with respect to $\boldsymbol{\omega}$ and $\mathbf{v}$, which makes it hard to simultaneously figure out angular velocity and linear velocity. In the continuation, we therefore consider the case where angular velocities are given by an Inertial Measurement Unit (IMU).

3 Closed-form Velocity Initialization

Typically, a DVS sensor such as the DAVIS346, integrates an event camera and an IMU which provides angular velocity and acceleration. With the help of the prior known angular velocity, the nonlinear 5 DoF motion estimation problem is reduced to a 2 DoF problem: translational velocity estimation. The closed-form speed initialization algorithm proceeds in four steps. The first step consists of event clustering. Next, for each cluster we extract the lines $\mathbf{l}_1$ and $\mathbf{l}_3$ by using a small time interval of events at the beginning and the end of each cluster. Finally, using (7), we propose a linear closed-form speed solver for the linear velocity $\mathbf{v}$. The fourth and final step consists of nonlinear optimization improving the estimation result.

3.1 Line Clustering and Extraction

We adopt a strategy similar to the one leveraged in [7, 17], which considers events as a 3D point cloud in the space-time volume. The coordinates are given by the pixel position of the event and the timestamp, i.e. $e_i = [x_i, y_i, t_i/c]$. To balance the magnitude of the image coordinates and the timestamp of an event, the latter is normalized by a constant c whose value is chosen according to the average level of texture in the scene. The time span over which event clusters are formed is dynamically defined by considering a fixed number of N events. Events generated by the same line will approximately form a local plane in the 3D space-time volume of the event stream. Hence—ignoring the influence of rotational velocities, clustering events generated by the same line in 3D roughly amounts to plane clustering in a 3D point cloud. We employ the open-source C++ library *Cilantro* [31] to implement the clustering procedure, which operates in a region growing fashion inspired by connected component segmentation. For more details, please refer to [17].

For each event cluster $\mathcal{E}_j$ in which events are sorted with increasing timestamps, we utilize the first and last 0.005s intervals of events to extract the lines l_{1j} and l_{3j}. We use *cv::fitLine* from *OpenCV* [3] to extract the lines, and the algorithm is based on an M-estimator that iteratively fits the line using a weighted least-squares algorithm. We choose the Huber norm strategy ([13], page 43). Note that our algorithm uses variable timestamps around which l_{1j} and l_{3j} are fitted, which enables us to slide the 0.005s intervals towards the center of the entire cluster interval in case of insufficient events.

3.2 Linear Velocity Solver

With known angular velocity, the CELC (7) becomes linear in the translational velocity. It is furthermore easy to concatenate all linear constraints for all events of all clusters into one constraint. Given M event clusters $\mathcal{E}_j$ where $j = 1,2...,M$ and the corresponding extracted lines l_{1j} and l_{3j}, the constraints from each event cluster with N_j events can be stacked into the single linear problem

$$\left[\mathbf{B}_{11}^{\mathsf{T}}\mathbf{f}_{11} \ \dots \ \mathbf{B}_{kj}^{\mathsf{T}}\mathbf{f}_{kj} \ \dots \ \mathbf{B}_{N_M M}^{\mathsf{T}}\mathbf{f}_{N_M M}\right]^{\mathsf{T}}\mathbf{v} = \mathbf{A}\mathbf{v} = 0. \tag{9}$$

$\mathbf{A}$ can be computed from the known angular velocity, the extracted lines l_{1j} and l_{3j}, and all measured events. This linear problem could be solved using $\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{v} = 0$ via SVD. However, given that least square estimation methods lack robustness [13][2], we choose to use another robust M-estimator with a Huber norm [13], and employ iteratively re-weighted least-squares fitting for the nullspace extraction. As the number of events N is very large (about 100,000 in our real data experiments), and $\mathbf{A}$ is an $N \times 3$ matrix, we improve efficiency by randomly selecting 1000 samples out of the N to perform the M-estimation. Further details of our implementation can be found in the *Definition* part of *Chapter 1.3* in [25].

3.3 Degenerated Case

Note that our linear solver cannot always determine a unique solution. It is obvious that if the motion of the camera is a pure rotation—meaning that $\mathbf{v} = \mathbf{0}$—solving our linear equation 9 via SVD will not be possible. Another degenerate case exists if the camera moves along a straight line without rotation. It is obvious that in this case any translational velocity component along the direction of the 3D line $\mathbf{L} = \mathbf{l}_1 \times \mathbf{l}_3$ will not contribute to any appearance changes in the image, and therefore also no events. Hence, the 3D line direction needs to lie in the null space of the matrix $\mathbf{A}$, which means $\mathbf{A}(\mathbf{l}_1 \times \mathbf{l}_3) = 0$. Moreover, $\mathbf{A}\mathbf{v} = \mathbf{A}(\mathbf{v}_1 + \mathbf{l}_1 \times \mathbf{l}_3) = \mathbf{A}\mathbf{v}_1$. There exists an unobservable direction for the translational velocity given by the direction of the 3D line. In the case of linear motion, the unobservable direction also exists when there are multiple lines but all of them are parallel.

3.4 Nonlinear Optimization

In real cases, events are affected by both spatial and temporal noise as well as outliers in the form of salt and pepper noise. This may lead to errors in the event clustering and the line extraction. The linear solver introduced in the previous section therefore only offers an initial guess, which may be further refined by a maximum likelihood estimation to make the estimation more accurate and robust.

The objective is to minimize the geometric distance between the reprojected 3D line and the events. The cost function is given by

$$\min_{\mathbf{v}} \sum_{j=1}^{M} \sum_{k=1}^{N_j} d(\tilde{\mathbf{l}}_{2kj}, \mathbf{f}_{kj})^2, \tag{10}$$

where $d(\cdot)$ represents the distance function, and $\tilde{\mathbf{l}}_{2kj}$ is the reprojected line in the image plane at timestamp t_{kj}. One way to conveniently represent the 3D line during the nonlinear optimization is by its projections $\mathbf{l}_{1j}$ and $\mathbf{l}_{2j}$ in their relevant views. Given a candidate linear velocity, one can then again compute the trifocal tensor, and furthermore easily derive the

location of the projected 3D line at the time of the corresponding event using the line transfer equation $\tilde{\mathbf{l}}_{2kj} = \mathbf{B}_{kj}\mathbf{v}$.

The entire objective is minimized using a trust-region based method (e.g. Levenberg-Marquardt) and implemented with *Ceres* [1] using a robust cost function (Huber norm).

4 Experiments

We analyze the proposed closed-form algorithm both in simulation and on real data. We use the Euclidean distance ε and the cosine distance ϕ between the estimated results and ground truth as a metric to evaluate the accuracy of the estimated results. They are given as follows:

$$\varepsilon = \|\mathbf{v}_{\text{gt}} - \mathbf{v}_{\text{est}}\|_2, \ \phi = \arccos(\mathbf{v}_{\text{gt}}^{\mathsf{T}}\mathbf{v}_{\text{est}}), \tag{11}$$

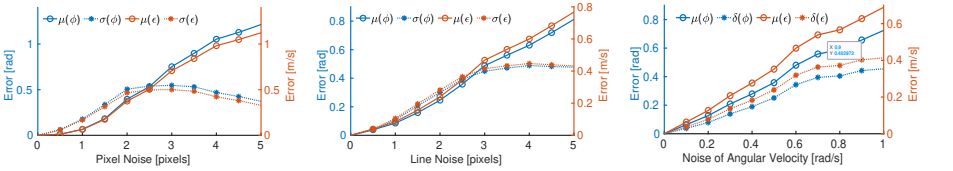
where $\mathbf{v}_{\text{gt}}$ and $\mathbf{v}_{\text{est}}$ are the ground truth and estimated linear velocities, respectively.

4.1 Simulation

We start by evaluating the performance of the proposed approach over synthetic data. To generate synthetic data, we randomly generate line segments in 3D space within a volume of $x = [-2,2]$ m, $y = [-2,2]$ m, and $z = [3,6]$ m. With given angular and linear velocities, an event is generated by randomly choosing a 3D point on a line and projecting it into an image plane with the camera pose sampled by a random timestamp within a given time interval. In our experiments, we set the angular velocity of the camera as $\boldsymbol{\omega} = [0,\ 0,\ 2]$ rad/s, and the linear velocity as $\mathbf{v} = [1,\ 2,\ 0]$ m/s. We generate events from 5 lines within a time interval of 0.5 s. We disturb the pixel location of each generated event by zero-mean Gaussian noise with a standard deviation of 2 pixels. We also add Gaussian noise of $\mathcal{N}(0,2)$ pixels to the ground-truth endpoints of each starting and ending line pair l_{1j} and l_{3j}.

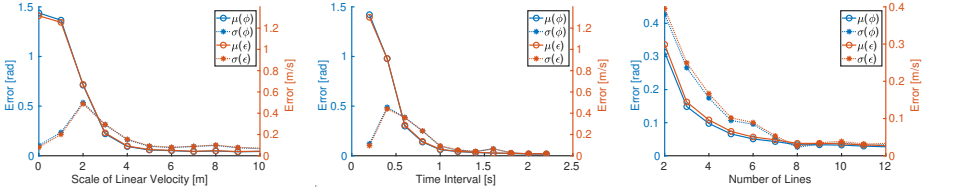
To evaluate the proposed solver, we adopt the single variable method to conduct various simulation experiments from two aspects. One is to evaluate the solver's robustness against noise including disturbance of events, errors of the extracted lines at the boundary of the interval and disturbances of angular velocities. The other one is to investigate the effect of certain factor such as the scale of the velocity, the length of the time interval, or the number of lines. Note that the errors for each level of each variable are averaged over 500 experiments. The detail configurations are as follows:

- **Robustness against event location noise:** The disturbance of each event is varied with a standard deviation reaching from 0 to 5 pixels with a step size of 0.5 pixels. The average μ and standard deviation σ of ε and ϕ is presented in Figure 2(a).
- **Robustness against noise in the end points of l_{1j} and l_{3j}:** We add pixel-level Gaussian noise to the two endpoints of the projected lines to test the robustness of the proposed linear solver. The noise is varied between 0 and 5 pixels with a step size of 0.5 pixels. Figure 2(b) indicates the simulated results.
- **Robustness against noise in angular velocity:** We add Gaussian noise to the known angular velocity to test the robustness of the proposed linear solver. The noise is varied between 0 and 1 rad/s with a step size of 0.1 rad/s. Figure 2(c) indicates the respective results.
- **Effect of speed:** We set the direction of the linear velocity as $[0.447,\ 0.894,\ 0]$ and the scale of the velocity is varied between 0 and 10 m/s. The simulation results are shown



(a) (b) (c)

Figure 2: Noise analysis. (a) shows the errors of the solver with increasing event location disturbance. (b) illustrates the error of our solver with different levels of noise added to the end-points of the fitted lines l_{1j} and l_{3j}. (c) displays the errors of the solver with increasing angular velocity disturbance. The error generally increases with noise.



(a) (b) (c)

Figure 3: Accuracy for other motion or solver parameters. (a) shows the error of the solution over an increasing scale of the velocity. (b) indicates the effect of the time interval length. (c) shows the effect of the number of observed lines.

in Figure 3(a). As can be observed, errors are decreasing with an increasing norm of the speed. In other words, the higher speed, the more accurate our solver is operating.

- **Effect of the time interval size:** We vary the time interval from 0.2 s to 2.2s with a step size of 0.2 s. Results (Figure 3(b)) indicate that the errors are decreasing as the time interval is increasing.
- **Effect of the number of lines:** The number of lines is varied from 2 to 10 with steps of 1. Figure 3(c) presents the results. The more lines are present in the scene, the higher the accuracy of the solver.

Without loss of generality, the errors increase as noise level are increasing (cf. Figure 2). Note that the solver is rather sensitive to noise, which is analogous to the trifocal tensor-based approaches for standard cameras [12, 30]. Furthermore, the more displacement the camera experiences during the chosen time interval, the higher the expected accuracy.

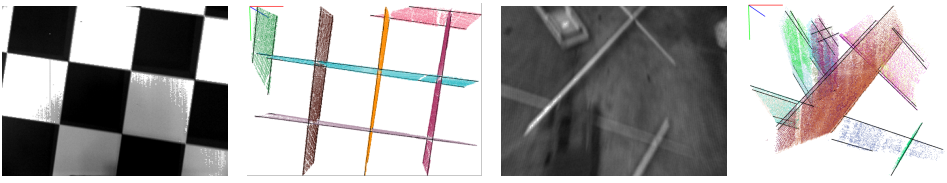
4.2 Real Data

To the best of our knowledge, we are the first to propose the CELC-based linear velocity solver for event cameras, and as such it is hard to compare against an existing SOTA algorithm. We therefore design our own baseline algorithm to evaluate the proposed method, which is based on the line-line-line incidence relationship [11].

$$\mathbf{l}_2 \times (\mathbf{l}_1^\top [\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3] \mathbf{l}_3) = 0. \tag{12}$$

Using equation (7), the line-line-line constraint under continuous motion is given by

$$\mathbf{l}_2^\wedge \mathbf{B}_k \mathbf{v} = 0, \tag{13}$$



(a) (b) (c) (d)

Figure 4: (a)-(b) Example data captured by an AGV with a downward-facing event camera. (c)-(d) Example data captured by an UAV with a 45° downward-facing event camera. (a) and (c) denote grayscale images, whereas (b) and (d) the identified event clusters corresponding to real-world line segments in a spatio-temporal view. The extracted lines at the beginning and at the end of each interval are shown in black. The coordinate system in the upper left corner of (b) and (d) means uses the red and green axes to denote the x and y coordinates of each event, and the blue axis to indicate the temporal axis.

where $\mathbf{l}_2^\wedge$ is the 3×3 screw symmetric matrix form of $\mathbf{l}_2$. The additional line l_2 is fitted at the center of the interval and by using the same strategy as introduced in Sec 3.1. With line features $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$ in three views as well as known angular velocity, we can again stack the constraints for all clusters and figure out the linear velocity through a similar robust nullspace calculation method as before. We denote this baseline implementation the continuous event-based line-line-line constraint (CE3LC).

We verify feasibility and practicality of our approach on two real-world datasets collected by an automated guided vehicle (AGV) and an unmanned aerial vehicle (UAV), respectively. The two datasets are collected by a DAVIS346, which has a resolution of 346x260 pixels.

(1) AGV with a downward-facing event camera

The AGV dataset is collected with a camera mounted on the front of an XQ-4 Pro robot and faces downwards (see Figure 4(a)-4(b)). We recorded a uniform circular motion sequence on a chessboard. Ground truth is obtained via an Optitrack motion capture system. Our algorithm is working in normalized coordinates, which is why normalization and undistortion are computed in advance.

Figure 4(b) shows the collected data and line clusterings. To alleviate the influence of noise on the events and resulting inaccuracies in the line clusters and extraction, we utilize a spatio-temporal window with 1,000,000 events (about 0.7s) to estimate the linear velocity.

(2) UAV with a 45° downward-facing event camera

We further evaluate our method on a sequence (Indoor45 9) from [6] which is captured by an UAV equipped with a 45° downward-facing event camera with a resolution of 346×260 pixels. The maximum velocity ($|\vec{v}|_{max}$) of the UAV is about 11.23 m/s, which means the intensity images are rather blurry. Therefore, it is difficult to use the intensity images to extract lines for pose estimation. The event camera however works well in such a challenging scenario. As can be observed in Figure 4(d), our strategy maintains successfully extracted event clusters and starting and ending lines from the raw stream of events. Note that in order to better distinguish lines that are very close, we separate events into positive and negative sets before running the actual clustering algorithm.

Table 1: AGV Errors

Method		CELC	CELC+opt	CE3LC
Seq1	ε [m/s]	0.2058	**0.2035**	0.6345
	ϕ [rad]	0.2063	**0.2038**	0.6457
Seq2	ε [m/s]	**0.1125**	0.1204	0.3180
	ϕ [rad]	**0.1123**	0.1201	0.3192
Seq3	ε [m/s]	0.2042	**0.1476**	0.6783
	ϕ [rad]	0.2043	**0.1471**	0.6921
Seq4	ε [m/s]	0.1590	**0.1455**	0.1951
	ϕ [rad]	0.1589	**0.1450**	0.1952
Seq5	ε [m/s]	0.2149	**0.1439**	1.0122
	ϕ [rad]	0.2154	**0.1441**	1.0615

Table 2: UAV Errors

Method		CELC	CELC+opt	CE3LC
Seq1	ε [m/s]	**0.2145**	0.2187	0.5192
	ϕ [rad]	**0.2150**	0.2193	0.5326
Seq2	ε [m/s]	0.2062	**0.1936**	0.6263
	ϕ [rad]	0.2067	**0.1940**	0.6536
Seq3	ε [m/s]	0.3619	**0.2499**	0.4756
	ϕ [rad]	0.3661	**0.2507**	0.4922
Seq4	ε [m/s]	0.2340	**0.2108**	0.5297
	ϕ [rad]	0.2347	**0.2110**	0.5379
Seq5	ε [m/s]	0.2126	**0.1118**	0.4828
	ϕ [rad]	0.2138	**0.1119**	0.4924

4.2.1 Analysis of the Results

We select 5 sequences from each dataset and the results are listed in Table 1 and Table 2. CELC indicates the proposed solver without optimization, CELC+opt the proposed solver with nonlinear optimization, and CE3LC the proposed baseline algorithm without nonlinear optimization. As can be observed, CELC+opt typically outputs better results than CELC, indicating the positive impact of nonlinear optimization. Furthermore, CELC outperforms CE3LC. The reason is given by the fact that CE3LC relies more heavily on the performance of 2D line fitting, while CELC utilizes all events measurements to constrain the problem.

Note that the accuracy of the algorithm highly depends on the accuracy of the line detection and fitting, the resolution of the camera, the number of lines in the environment, and other factors analyzed in the simulation experiments. For example, the resolution we use is only 346×260 pixels. As demonstrated by the KITTI dataset [10], a common resolution for normal cameras would be in the order of 1392×512, which is much higher. We furthermore believe that the number of studies on line detection and fitting in event streams is still rather limited, and better approaches would certainly benefit the method proposed in this paper.

5 CONCLUSION

Different from existing event-based motion estimation approaches, we are the first to exploit trifocal tensor geometry in order to constrain the dynamics of an event camera from an event stream generated by the continuous observation of arbitrary 3D lines. The closed-form velocity solver employs a novel constraint which we denote the Continuous Event Line Constraint (CELC). We believe that our algorithm is an important first step into the direction of velocity bootstrapping for DVS sensors, and our future work considers the embedding of this solver and the related constraints into a more complete, event-based visual-inertial framework for direct velocity estimation.

References

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] David F Andrews. A robust method for multiple linear regression. *Technometrics*, 16 (4):523–531, 1974.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[5] Christian Brändli, Jonas Strubel, Susanne Keller, Davide Scaramuzza, and Tobi Delbruck. Elised—an event-based line segment detector. In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–7. IEEE, 2016.

[6] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[7] Lukas Everding and Jörg Conradt. Low-latency line tracking using event-based dynamic vision sensors. *Frontiers in neurorobotics*, 12:4, 2018.

[8] Guillermo Gallego, Jon EA Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-based, 6-dof camera tracking from photometric depth maps. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2402–2412, 2017.

[9] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus is all you need: loss functions for event-based vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12280–12289, 2019.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] Richard I Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.

[13] Peter J Huber. *Robust statistics*, volume 523. John Wiley & Sons, 2004.

[14] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[15] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[16] Raphaela Kreiser, Alpha Renner, Vanessa RC Leite, Baris Serhan, Chiara Bartolozzi, Arren Glover, and Yulia Sandamirskaya. An on-chip spiking neural network for estimation of the head pose of the icub robot. *Frontiers in Neuroscience*, 14, 2020.

[17] Cedric Le Gentil, Florian Tschopp, Ignacio Alzugaray, Teresa Vidal-Calleja, Roland Siegwart, and Juan Nieto. Idol: A framework for imu-dvs odometry using lines. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5863–5870. IEEE.

[18] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 *times* 128 120 db 15 *mu*s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.

[19] Daqi Liu, Alvaro Parra, and Tat-Jun Chin. Globally optimal contrast maximisation for event-based motion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6349–6358, 2020.

[20] Daqi Liu, Alvaro Parra, and Tat-Jun Chin. Spatiotemporal registration for event-based visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4937–4946, 2021.

[21] Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based, 6-dof pose tracking for high-speed maneuvers. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2761–2768. IEEE, 2014.

[22] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440, 2018.

[23] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[24] Xin Peng, Ling Gao, Yifu Wang, and Laurent Kneip. Globally-optimal contrast maximisation for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[25] Andreas Ruckstuhl. Robust fitting of parametric models based on m-estimation. *Lecture notes*, page 40, 2014.

[26] Florian Tschopp, Cornelius von Einem, Andrei Cramariuc, David Hug, Andrew William Palmer, Roland Siegwart, Margarita Chli, and Juan Nieto. Hough^{2}map – iterative event-based hough transform for high-speed railway mapping. *IEEE Robotics and Automation Letters*, 6(2):2745–2752, 2021. doi: 10.1109/LRA.2021.3061404.

[27] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018.

[28] Rafael Grompone Von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: a line segment detector. *Image Processing On Line*, 2:35–55, 2012.

[29] David Weikersdorfer, Raoul Hoffmann, and Jörg Conradt. Simultaneous localization and mapping for event-based vision systems. In *International Conference on Computer Vision Systems*, pages 133–142. Springer, 2013.

[30] Juyang Weng, Thomas S. Huang, and Narendra Ahuja. Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization. *IEEE Computer Architecture Letters*, 14(03):318–336, 1992.

[31] Konstantinos Zampogiannis, Cornelia Fermuller, and Yiannis Aloimonos. Cilantro: A lean, versatile, and efficient library for point cloud data processing. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1364–1367, 2018.