2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Fast Localization and Tracking using Event Sensors

Wenzhen Yuan [1] and Srikumar Ramalingam[2]

***Abstract*— The success of many robotics applications hinges on the speed at which the underlying sensing and inference tasks are carried out. Many high-speed applications such as autonomous driving and evasive maneuvering of quadrotors require high run time performance, which traditional cameras can seldom provide. In this paper we develop a fast localization and tracking algorithm using an event sensor, which produces on the order of million asynchronous events per second at pixels where luminance changes. The events are usually fired at the high gradient pixels (edges), where luminance changes occur as the sensor moves. We develop a fast spatio-temporal binning scheme to detect lines from these events at the edges. We represent the 3D model of the world using vertical lines, and the sensor pose can be estimated using the correspondences from 2D event lines to 3D world lines. The inherent simplicity of our method enables us to achieve a run time performance of 1000 Hertz.**

I. INTRODUCTION

Conventional cameras see the world as a sequence of frames. This frame-based representation is a very inefficient and redundant encoding of the world, which results in wastage of power and time. In addition, each pixel has the same exposure leading to difficulty in imaging scenes that contain high dynamic range, i.e., scenes containing very dark and very bright regions. Dynamic vision sensors (DVS) [16], [17], [6], [3] on the other hand, are inspired by human retina [7]. These sensors do not send entire image frames at fixed rates. Instead, they only send pixel-level changes as ON and OFF events (similar to ON and OFF retinal ganglion cells) at exactly the time they occur. As a result, the events are transmitted at microsecond time resolution.

Event-based sensing in robotics and vision : Several applications, which were commonly solved using frame-based cameras, are now being developed using event-based sensors. The Lucas-Kanade tracking algorithm was extended for DVS sensor for computing the optical flow [1]. Certain registration methods such as iterative-closest-point (ICP) have been extended for event sensors to control micro-grippers [21]. DVS has been shown to be useful for evasive maneuvering of quadrotors [19]. In their work, a pair of DVS sensors are used as a stereo camera for reconstructing objects thrown at a quadrotor for predicting and avoiding the collision.

In the case of asynchronous event-based cameras, the concept of epipolar geometry can not be directly used. Epipolar geometry provides a relationship between corresponding points in a pair of images captured from different viewpoints. Recently, a general version of epipolar geometry applicable for asynchronous event-based stereo-configuration has been developed [2]. The stereo-matching is also computed using event histograms and the estimated depth has been used for the task of gesture recognition [15]. Beyond stereo, there have been methods that use as many as six synchronized event sensors for 3D reconstruction [4].

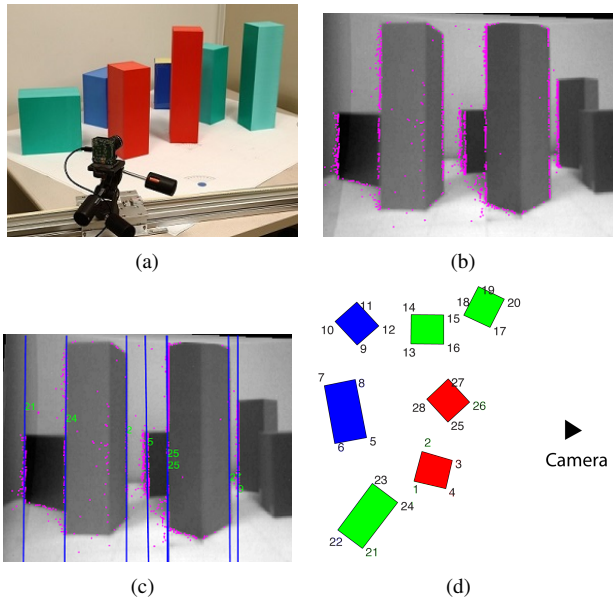


Fig. 1. We solve the classic problem of pose estimation using an event sensor (a) A scene with cuboids and DAVIS240B event sensor. (b) The intensity image (used only for visualization) and events data are denoted by magenta dots. (c) Detected vertical lines from the events data. (d) Top view of the blocks and the sensor pose. We match 2D vertical lines from the events data to the 3D vertical lines in the world. Using these matches, we compute three degrees of freedom (DOF) planar pose.

Censi and Scaramuzza built a visual odometry algorithm using a DVS sensor and a CMOS camera [5]. They developed a novel calibration algorithm to spatio-temporally calibrate the DVS sensor and the CMOS camera. Their approach estimates the relative motion of the DVS events with respect to the previous CMOS frame. They show high accuracy on the rotation estimates, whereas translation measurements are shown to be noisy. In this work, we focus on using only the event sensor to achieve very high speed. We believe that for certain applications the use of CMOS camera would provide complementary benefits, but the algorithm has to be designed in such a manner that the low frame rate of CMOS camera would not be a bottleneck. DVS solutions have been proposed for particle-filter based localization and

[1] Wenzhen Yuan is with Department of Mechanical Engineering, and Computer Science and Artificial Intelligence Laboratory(CSAIL), MIT, Cambridge, MA 02139, USA yuan_wz@csail.mit.edu
[2] Srikumar Ramalingam is with Mitsubishi Electric Research Labs (MERL) ramalingam@merl.com

mapping [23], [24], [11]. It is possible to build a mosaic of a scene solely using a DVS sensor without any additional sensor. This is achieved by tracking the camera motion and using the estimated motion for registering and integrating the events data [13]. In a closely related work [20], a four-sided square is detected and used for estimating the pose of a camera at micro-second resolution. In our work, we use only vertical lines and we can obtain poses from more general scenes.

Absolute pose estimation methods: In this work, we show a fast and accurate absolute pose estimation method using an event sensor. The events are fired mainly at the high gradient pixels, which are connected set of edge pixels. In order to achieve very high speed, we restrict our focus to man-made scenes with line features. We show a method to detect vertical lines from images at a very high speed (more than 1000 detections per second) using a simple binning scheme. We also develop a novel absolute pose estimation algorithm that uses correspondences between 2D lines from events data and 3D world lines. We use the inertial measurement unit (IMU) to measure absolute roll and pitch angles and thus align the sensor's view to the world's vertical direction. In a related work [8], IMU was used to compute the rotation and stabilize event based sensors. Using point features, relative [10], [12] and absolute pose [14] estimation methods with known vertical direction have been been developed before in the context of regular cameras. Despite the simplicity of our approach, the vertical line-based pose estimation problem has not been solved before.

II. Dynamic and Active Pixel Vision Sensor

In this work, we use dynamic and active pixel vision sensor (DAVIS 240B). This sensor comes with both intensity image (APS) and events data (DVS) along with IMU on board. The resolution of the APS frame is 240×180 and the dimensions of the events data is 190×180. The APS frame rate is 23 Hz and the IMU operates at 2300Hz. The IMU provides acceleration and angular velocity in 6 dimensions. In this work, we don't rely on APS data for our algorithm and it is only used to illustrate the results. We rely on IMU data to only align the camera in the vertical direction.

The dynamic vision sensor (DVS) denotes the pixel-level brightness changes using only two states. Each pixel asynchronously emits an ON event if the log-compressed light intensity increases by a fixed amount and an OFF event when it decreases. An event E is noted as tuple $< x, y, t, s >$ where (x, y) is the pixel location, t is the time-stamp and s is the event polarity: 1 for ON signals and 0 for OFF. The sensor generates on the order of million events per second. We used jAER software [9] to record the original data and our algorithm is implemented in C++.

As with any sensor, the event sensor produces noisy data that requires careful understanding to obtain useful information. We briefly show illustrations of the events data for simple 3D scenes to better appreciate the practical issues of events data in Figure 2. When walking in the corridor, for a time-slice of 100 ms, we obtain 34792 events out of which 10318 events are the ones with positive polarity and the remaining 24474 events are negative ones. In general, we observed more noisy points to have negative polarity. The average time interval between two consecutive events is 2.85 microseconds and the shortest time between two events is 0.13 microseconds. In regions with strong contrast, we see multiple firings and we show the events with various time-intervals between firings in Figure 2(d).

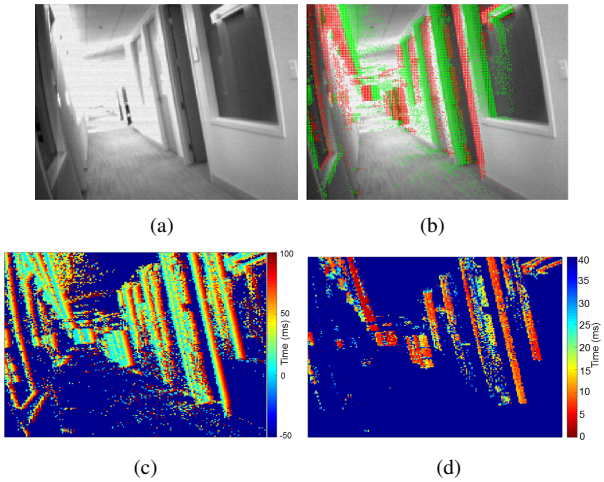


Fig. 2. (a) An image of a corridor scene. (b) The generated events while walking in the corridor. The red points show events with polarity $+1$ and green ones denote events with polarity -1. (c) The color-coded timestamps of the first events in different pixel locations. (d) The color-coded time difference between two events at individual pixels. When the intensity change is large, the same point will provoke multiple events, and the time difference between the two events on the same point indicates the strength of the contrast.

III. Algorithm

We use vertical lines as the feature to estimate the pose of an event sensor given a 3D model of the world. Considering the nature of the events data, it would be difficult to develop a keypoint detector like SIFT. By ignoring the height, the 6 DOF localization problem is simplified to a 3 DOF problem. This 3 DOF (planar location and yaw rotation) is more important in localization of on-road vehicles and mobile robots, while the remaining variables (height, roll, pitch angles) can be obtained using IMU and other ways. Detecting only the vertical lines also makes the feature extraction much faster than finding all lines with different orientations. Our algorithm comprises of the following steps:

A. Camera alignment using IMU

The IMU information is used to rotate the event data to align the vertical direction in the sensor's view to the world's. This ensures that the vertical lines the sensor 'sees' are also vertical lines in the world, regardless of the sensor's attitude. The gyroscope has large and time-dependent drift, and thus we applied AHRS algorithm introduced in [18] to get the sensor's attitude in the world coordinate system. Since the accelerometer only measures the gravity vector, the algorithm does not get precise yaw angle when the sensor is static or at

low speed. Thus we only use the roll and pitch angles from the IMU.

Based on the attitude measured by IMU, we obtain R to denote the rotation to align the sensor's view and the world's vertical direction. The relation between the pixel $p'(x', y')$ in the rotated image and its position $p(x, y)$ in the original image is

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = s\mathsf{KRK}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (1)$$

Here K is the camera matrix for the event sensor, and s is the scale factor that denotes that the relation is up to a scale.

B. Fast line-detection using spatio-temporal binning

We use a simple voting algorithm based on event's locations and timestamps to detect vertical lines from the IMU-rotated image. After the rotation based on IMU data, the vertical lines in the world coordinate will also be vertical in the camera image, so that most of the events on the line will be located in a strip area on the image from $(x_i, 1)$ to (x_i+dx, y_{max}). We denote the area as $\mathcal{V}_{dx}(x_i)$. Additionally, points on one edge provokes multiple events, especially when the intensity contrast is large. The time interval Δt between two events firing on the same position measures the contrast intensity, as a smaller Δt indicates more event firings, thus a higher contrast on an edge. We use a bin $\mathcal{V}_{dx,dt}(x_i, \Delta t_j)$ to denote the pixels that is within $\mathcal{V}_{dx}(x_i)$ and has a firing interval between Δt_j and $\Delta t_j + dt$.

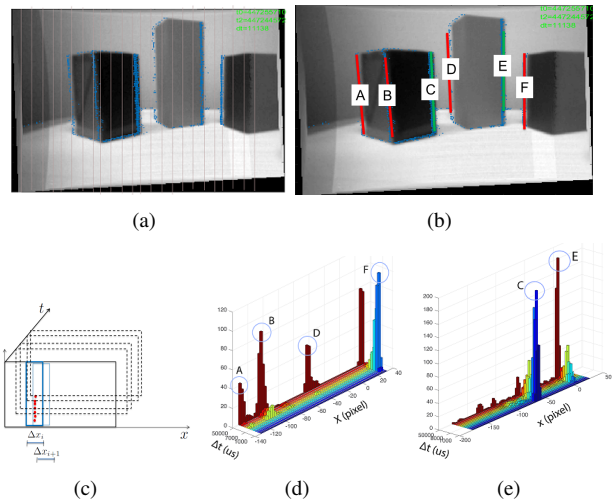
The events come as a sequence instead of a frame-based images like other cameras provide, so the 'frame' for the event sensor is manually decided. We choose a ΔT as the 'frame interval', which differs according to the sensor's approximate speed, and consider the events fired within the time interval to be of one frame. Then we count the events in each bin $\mathcal{V}_{dx,dt}(x_i, \Delta t_j)$. The bins are in an overlapping set of spatial and temporal intervals to ensure a better localization, as shown in Figure 3(c). We consider a bin $\mathcal{V}_{dx,dt}(x_i, \Delta t_j)$ to contain a vertical line if it satisfies the following conditions:

1) $\mathcal{V}_{dx,dt}(x_i, \Delta t_j) > \mathcal{T}_1$
2) The spatial neighbors have fewer votes as given by the following constraints:
   $\mathcal{V}_{dx,dt}(x_i, \Delta t_j) > \mathcal{V}_{dx,dt}(x_{i\pm1}, \Delta t_j)$
   $\mathcal{V}_{dx,dt}(x_i, \Delta t_j) > \mathcal{V}_{dx,dt}(x_{i\pm2}, \Delta t_j) + \mathcal{T}_2$
3) There is no other bin $\mathcal{V}_{dx,dt}(x_i, \Delta t_k)$ satisfying the above threshold conditions if $\Delta t_k < \Delta t_j$.

Here $\mathcal{T}_1$ and $\mathcal{T}_2$ are empirical parameters chosen to distinguish the lines. Condition 3 is to ensure the high contrast lines are of higher priority in the choice.

C. Pose Estimation using vertical lines

We make use of an Inertial Measurement Unit (IMU) to get the known vertical direction and thus solve a 3-point pose estimation problem for recovering three unknown motion parameters (two translation parameters and one rotational angle). Once we recover these 3 degrees of freedom, we already know the 5 unknown motion parameters out of the total of 6 unknowns. We can easily recover the remaining unknown using some non-vertical line. In this paper we mainly focus on extracting the 3 important motion parameters that will allow us to localize and track the event sensor in a scene with dominant vertical structures.



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

Fig. 3. The basic idea behind the vertical line detection is illustrated. (a) We show a scene with blocks, and the associated events are shown as blue dots. The image is divided into vertical strips, and events on the same vertical line are likely to fire in the same strip. (b) The detected lines using our spatio-temporal binning scheme. (c) A schematic diagram showing our spatio-temporal binning for detection of vertical lines. The positive and negative events are addressed individually. (d) Histogram of $\mathcal{V}(x_i, \Delta t_j)$ for positive events. We show the peaks while detecting lines as shown by A, B, D and F. (e) Histogram of $\mathcal{V}(x_i, \Delta t_j)$ for negative events. Lines are obtained from the peaks associated with lines C and E.

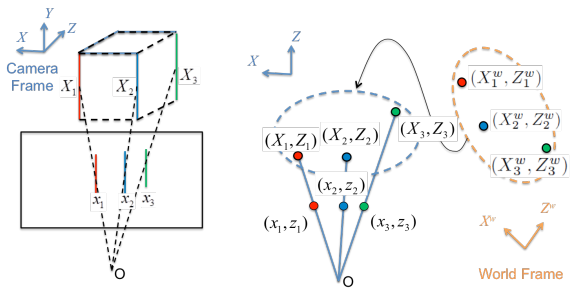


Fig. 4. On the left, we show the projection of lines from a cube to the image plane. On the right, we show the top view of the projection where vertical lines in the scene are denoted by 3D points $(X_1, Z_1), (X_2, Z_2), (X_3, Z_3)$ and vertical lines in the image are denoted by 2D points $(x_1, z_1), (x_2, z_2), (x_3, z_3)$. We show a schematic diagram of the projection of three 3D points to three 2D points on a plane. Let us denote the 3D points in the world coordinate frame to be $(X_1^w, Z_1^w), (X_2^w, Z_2^w), (X_3^w, Z_3^w)$ that can be transferred to camera coordinate frame using transformation (R, t).

As shown in Figure 4, the transformation between a vertical line in the world and its corresponding image line is shown below.

$$\begin{pmatrix} x_i \\ z_i \end{pmatrix} = s_i \begin{pmatrix} cos\theta & -sin\theta & T_x \\ sin\theta & cos\theta & T_z \end{pmatrix} \begin{pmatrix} X_i^w \\ Z_i^w \\ 1 \end{pmatrix} \quad (2)$$

We rearrange the above equation, eliminate the scale variable s_i and stack the equations for three 3D to 2D correspondences to obtain the linear system $\mathsf{A}\mathbf{X} = \mathbf{0}$, where A and $\mathbf{X}$ are given below:

$$\mathsf{A} = \begin{pmatrix} z_1 & -x_1 & (-z_1 Z_1^w - x_1 X_1^w) & (z_1 X_1^w - x_1 Z_1^w) \\ z_2 & -x_2 & (-z_2 Z_2^w - x_2 X_2^w) & (z_2 X_2^w - x_2 Z_2^w) \\ & & \vdots & \end{pmatrix} \quad (3)$$

$$\mathbf{X} = \begin{pmatrix} T_x \\ T_z \\ sin\theta \\ cos\theta \end{pmatrix} \quad (4)$$

When there are 3 lines, the solution will typically be unique. Using the reduced row-echelon form we obtain the following linear system from the above equation:

$$\begin{pmatrix} 1 & 0 & 0 & \alpha \\ 0 & 1 & 0 & \beta \\ 0 & 0 & 1 & \gamma \end{pmatrix} \begin{pmatrix} T_x \\ T_z \\ sin\theta \\ cos\theta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (5)$$

where α, β and γ are constants obtained while computing the reduced row echelon form for matrix A. Using the above linear system we compute the three unknown motion parameters θ, T_x and T_z as follows:

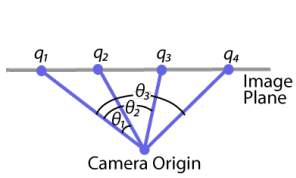
$$\begin{aligned} \theta &= tan^{-1}(-\gamma) \\ T_x &= -\alpha cos(tan^{-1}(-\gamma)) \\ T_z &= -\beta cos(tan^{-1}(-\gamma)) \end{aligned}$$

Given a set of n 2D to 3D line correspondences, where n is greater than 3, we can form a linear system $AX = 0$ using all correspondences and solve the motion variables using least squares.

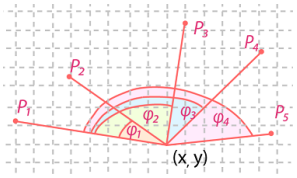
D. Correspondence Search using View Angle

The pose estimation algorithm requires 2D to 3D line correspondences in order to estimate the pose using the method in Section III-C. We use a spatial search method to find the correspondences, in order to reduce the computation complexity. We are given the 3D model of the world (in terms of vertical lines). We pre-process and synthesize all possible images of the lines from various viewpoints. We compare the synthesized images with real events data to identify the sensor location and orientation.

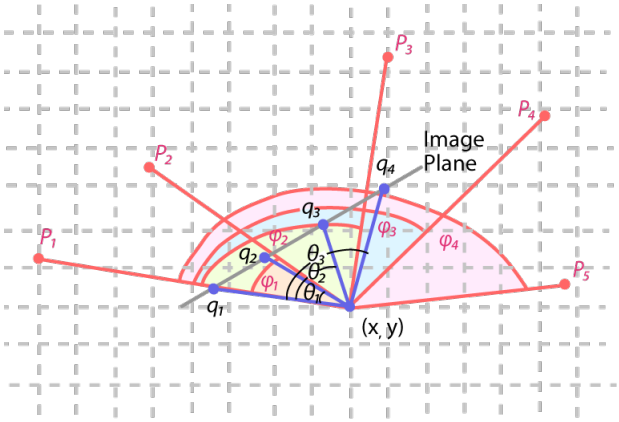
In this method, the vertical lines are represented by viewing angles in the x-y plane. In the camera image, as shown in Figure 5(a), we show a set of points $\{q_1, q_2, q_3, q_4\}$ denoting the vertical lines on the image, and $\{0, \theta_1, \theta_2, \theta_3\}$ denoting their viewing angles; in the world coordinate, as shown in Figure 5(b), 3D vertical lines are represented as $\{P_1, P_2, P_3, P_4, P_5\}$, and when viewing from location (x, y) the corresponding viewing angles are $\{0, \phi_1, \phi_2, \phi_3, \phi_4\}$. When the lines on the camera image match the lines in the world, their relative viewing angles shall be the same. Figure 5(c) shows a possible correspondence between $\{q_1, q_2, q_3, q_4\}$ and $\{P_1, P_2, P_3, P_4, P_5\}$, that supposing q_1 matches P_1. Here we have the correspondences:$\{(q_1, P_1), (q_2, P_2), (q_4, P_3)\}$ as their viewing angles are close. We calculate the matching cost as $abs(\theta_2 - \phi_2) + abs(\theta_4 - \phi_3)$. Searching the best correspondence means finding the largest set of correspondences with minimum matching cost. In the complete search area, we search for correspondences in each grid and pick the best one. Let m and n be the number of lines in the sensor plane and the world respectively, and let n_x and n_y be the two dimensions of the spacial grid for searching. In this case, the overall complexity of the searching is $O(n_x n_y nm)$, which is significantly lower than a brute-force search strategy.



(a) Vertical lines in sensor plane and their viewing angles, on vertical view.



(b) Vertical lines in the scene and their viewing angles based on a given sensor location (x, y), on vertical view.



(c) For the sensor location at (x, y) we match the viewing angles of sensor lines and the world lines.

Fig. 5. The view angles of the sensor lines and the world lines are used to identify the correspondences between them.

Note that the order of the vertical edges in the world may change when viewing from different sensor positions, if the edges are not on a single plane. Visibility of edges at different locations is also considered in the pre-processing.

We encountered a few challenges while using this search method: missing or spurious lines in the line detection; the modeling of the world is not accurate enough; repetitive scene geometry in the real world, especially while considering vertical lines. As a result, a good solution may have a higher matching cost than a wrong one. We resolve these issues by preferring solutions closer to previous ones, as further discussed in Section III-E. This strategy was necessary to get accurate pose estimates for handling some complicated scenes, like the outdoor building scene discussed in Section IV-C.

E. Pose Estimation: Initialization and Update

After extracting the vertical lines, if we know the previous pose, we update the current pose. If we don't have a reliable previous pose, we do a complete correspondence search for initialization. In the pose update case, we use the transformation matrix from the previous pose to project the world lines to the image plane, and compare them to the measured lines. We obtain the correspondences by simply looking at nearby lines.

In the pose initialization procedure, a complete correspondence search for the measured lines and world lines is done. We treat a pose unreliable in the following cases:

- There are too few line matches.
- The measured pose is too far from the previous reliable pose.
- A continuous sequence of previous poses have large reprojection errors.

On an average, the initialization takes about 20 ms whereas the pose update takes about 1 ms. If the pose estimation is stable, pose initialization is scarcely done, and the average estimation time is very short. When the scene or motion is complex, pose initialization is called many times leading to a gradual degradation in the run time performance.

IV. Experiments

We used three different experimental setups to test the accuracy and computational speed of the proposed methods:

- Boards - a pair of planar boards with black and white stripes,
- Blocks - a table top scene with toy blocks,
- Building - an outdoor scene of a building containing two perpendicular walls with many windows.

The blocks and boards correspond to indoor experiments, where we use a motorized linear stage and precision tripods for quantitative analysis of rotation and translation errors. In the outdoor case (building), we performed a qualitative analysis of the poses. In all the cases, we studied the computation time of the proposed algorithms on Intel Core i7-4790K, 4.00GHz processor and 32 GB RAM. In the supplementary materials, we show a video illustrating the setup and pose estimates.

A. Boards scene

As shown in Figure 6, we use two planar boards each containing black and white stripes of varying widths. The angle between the boards is 105°.

Fig. 6. The boards scene and the motorized rail to translate the sensor.

Translation: We use a motorized linear stage as shown in Figure 6 to generate a pure translation. We move the sensor at a uniform speed for 1 meter and move it back to the starting position. The estimated poses and trajectories are shown in Figure 7. On the left, we show the events and detected lines. On the right, we show the estimated poses and trajectory. It can be seen that the estimated trajectory is close to a straight line.

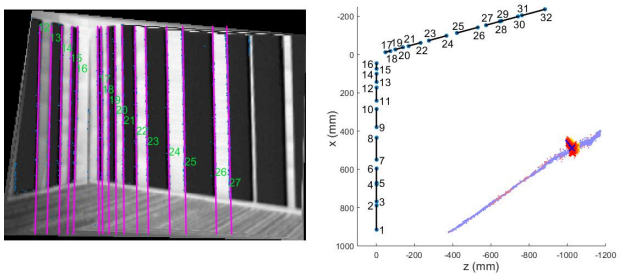


Fig. 7. The boards scene: We show the estimated poses for pure translation. On the left, we show the intensity image, the events (blue dots), the detected lines, and the associated line indices (in green). On the right, we show the top view of the geometry of the scene. Black line segments denote stripes and blue points denote the trusted sensor positions, whereas the red points denote unreliable ones that do not follow a smooth trajectory. The blue cross-mark show the current sensor position and orientation, and the red-to-yellow set of marks shows the very recent poses.

Figure 8 shows X and Z coordinates of the sensor position during the motion. As we expect, these coordinates increase and decrease linearly as we move forward and backward respectively. The orientation angle fluctuates within 5 degrees, indicating that the poses are accurate. We fit a line to the estimated camera positions and found that the RMS error is 7.6 mm. Note that the overall size of the scene spans more than 2 meters indicating that the translation error is less than 1% with respect to the scene size. Due to the difficulty in estimating the ground truth camera centers, this mean square entity is a reasonable estimate for the translation error.

Rotation: We fix the sensor's location, and rotate it along yaw axis using the tripod's pan-head. The rotation is controlled manually, 15° each time followed by a short pause. The calculated angle is shown in Figure 9 with respect to the time. During the pause, there is no event and the algorithm assumes that the sensor is static. The rotation angles at the pauses are shown in Figure 10. The RMS error for the angle is 3.37°.

Rotation and Translation: When the sensor moves linearly using the motorized stage, we generate arbitrary smooth rotations by manually rotating the tripod's pan-head. The estimated poses and the trajectory are shown in Figure 11. The rotation angles follow a smooth trajectory as we manually rotate the pan-head smoothly, as shown in Figure 12. Note that the trajectory is close to a straight line and the RMS error for the line-fit is 13.1 mm, which is slightly larger than the case of pure translation. This marginal increase in the translation error could be due to the offset between sensor's optical center and the tripod's pan-head's center.

Computational Time: We have two modules in our algo-

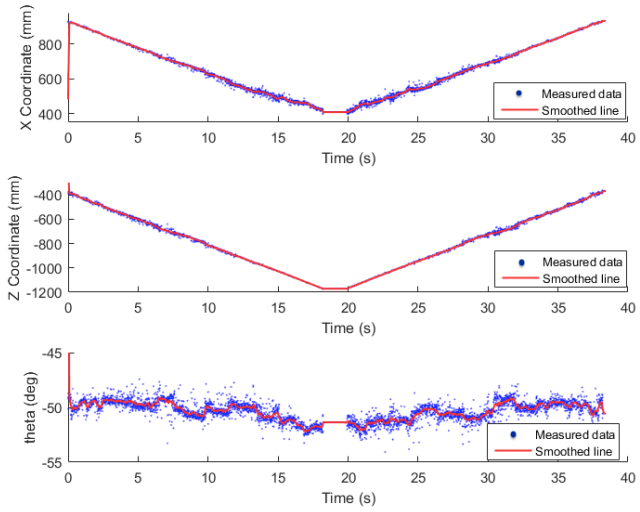


Fig. 8. The calculated X and Z positions and the yaw angle of the sensor when it translates uniformly on the motor stage. The blue dots are original data, and the red lines are smoothed estimation over a local average.

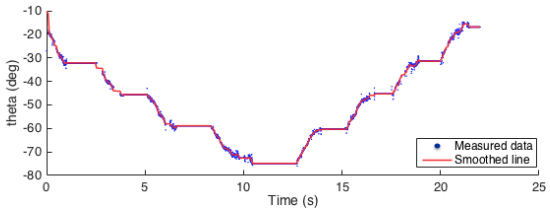


Fig. 9. Calculated position angle as time changes in the rotation experiment.

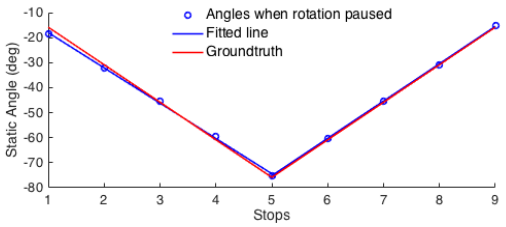


Fig. 10. Calculated sensor angles during pauses, in the rotation experiment. The red line is the approximate ground truth, blue plots are real data.

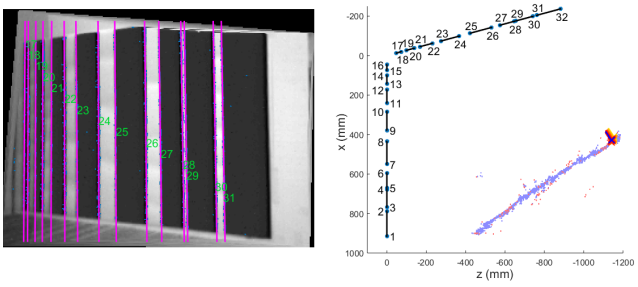


Fig. 11. Pose estimation results and the trajectory when both translation and rotation exist.

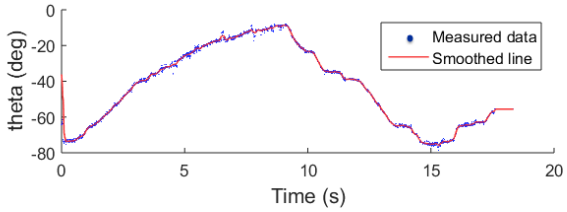


Fig. 12. The estimated yaw angles when the sensor is rotated manually while translating.

rithm. When we don't have a reliable previous pose, we localize the sensor using a complete search for 2D to 3D line correspondences. When we have a previous pose, we only need to update the position for small changes. The complete search for localization takes more time than the update. We show the computation time in Table I.

Reprojection error: Using the estimated poses we reproject the 3D lines to the image and measure the reprojection error. The reprojection error for the pose estimation is shown in Table II. As we observe, the error is generally less than 2 pixels, indicating that the pose estimates are accurate.

TABLE I

COMPUTATION TIME FOR BOARD SCENE

	Average Time	Initialization Time	Initialization Portion	Update Time
Translation	0.91ms	26.5ms	0.059%	$\sim$ 1ms
Rotation	2.26ms	28.2ms	6.6%	$\sim$ 1ms
Joint 1	1.02ms	22.9ms	0.8%	$\sim$ 1ms
Joint 2	1.50ms	23.6ms	2.4%	$\sim$ 1ms

TABLE II

REPROJECTION ERROR IN PIXELS FOR THE BOARD SCENE

	Translation	Rotation	Joint 1	Joint 2
Mean Error	2.176	2.275	2.144	2.180

B. Blocks Scene

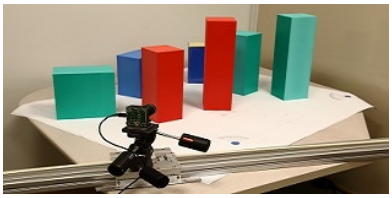


Fig. 13. The blocks scene: An experimental setup of a table top with cuboid blocks.

The experimental setup is shown in Figure 13. The blocks scene is more challenging than the boards scene because of the difficulty in establishing correspondences when the lines lie on separate planes. At times, two or more 3D lines might coincide to a single line in the view. These issues result in incorrect correspondences, which will lead to inaccurate poses. A careful check for all the visible lines for different viewpoints before the calculation could help to reduce error.

Translation: We conducted two different linear motions, and the RMS errors are 6.1 mm and 3.9 mm respectively. The trajectory of one linear motion is shown in Figure 14.

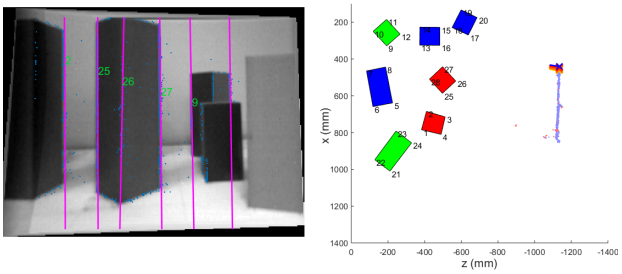
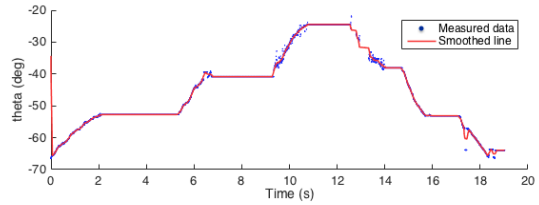
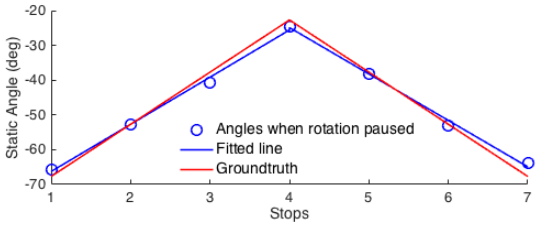


Fig. 14. Estimated pose and trajectory in a linear translation motion in the blocks scene.

Rotation: We generate pure rotation by rotating the tripod's pan-head by hand in the same manner as in the boards case. The estimated angles are shown in Figure 15(a), and the angles during pauses are shown in Figure 15(b). The RMS error for the angle is 5.6°.



(a)



(b)

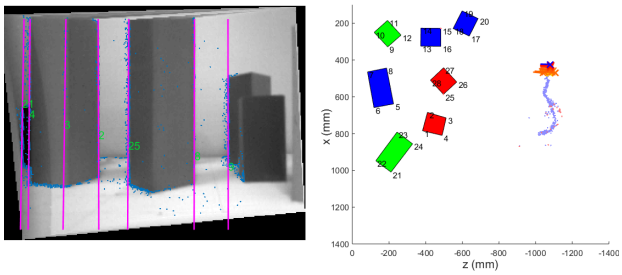
Fig. 15. The yaw angle estimation in rotation experiment in the blocks scene, when rotating the pan-head by hand. (a) shows sensor's angle with respect to time and (b) shows the angles at pauses, with the ground truth in red line.

Arbitrary motion: We perform a free-hand motion of the sensor. The estimated positions at different times are shown in Figure 16. We see smooth blue trajectories, despite some occasional red points indicating incorrect poses. These incorrect poses are due to mistakes in the 2D to 3D line correspondences.

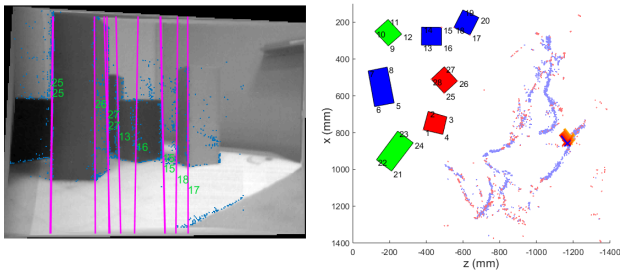
Computation time and error: The blocks scene contains fewer lines and thus takes less computation time compared to boards scene. The reprojection error is low, as shown in the Table IV, indicating that the estimated pose is likely to be accurate.

C. Outdoor scene

We tested our methods in an outdoor scenario where we used a free-hand motion to capture events data from the walls



(a)



(b)

Fig. 16. The sensor view and estimated sensor poses when the sensor is held by human and moves randomly around the scene.

TABLE III

THE COMPUTATION TIME FOR THE BLOCKS SCENE.

	Average Time	Initialization Time	Initialization Portion	Update Time
Translation 1	0.48ms	10ms	4.8%	∼0.2ms
Translation 2	0.23ms	10ms	2.3%	∼0.2ms
Rotation	0.23ms	10ms	2.3%	∼0.2ms
Free Motion	0.61ms	10ms	6.1%	∼0.2ms

of a building as shown in Figure 17. We show the pose estimation results in Figure 18. The calculation time is shown in Table V. The mean reprojection error for the building scene is given by 2.143.

The building contains a large number of windows providing many vertical lines necessary for our algorithm. We encountered several challenges due to the repeated nature of the vertical lines, large number of scene lines, illumination changes in outdoor setting and noisy events from horizontal lines. We used the line indices in the previous time-slice to decide on the number of lines to match in the correspondence search. This smoothness prior allowed us to detect lines from challenging outdoor scenes. We also carefully calibrated the line locations to reduce pose estimation error.

Fig. 17. We show a pair of images showing the L-shaped corner of an outdoor building, which is used in our experiment.

TABLE IV

REPROJECTION ERROR IN PIXELS FOR THE BLOCKS SCENE

	Translation1	Translation2	Rotation	Free Motion
Mean Error	0.978	0.816	1.477	1.586

TABLE V

POSE ESTIMATION TIME IN OUTDOOR SCENE

Avg Time	Init. time	Init. Portion	Update Time
1.49ms	22.8ms	3.0%	$\sim$ 1ms

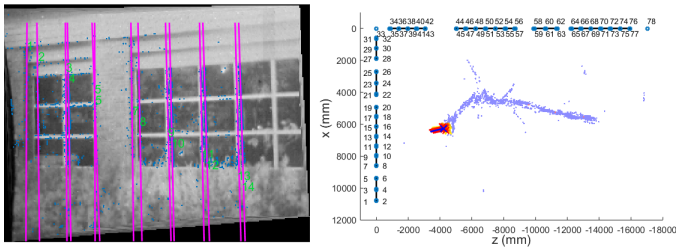


Fig. 18. The sensor's view and estimated poses and trajectory in the outdoor experiment.

In this work, we manually built the line-based model through measurements. In future, we plan to automate the process of line-based modeling using approaches similar to [22]. In general, we observed two challenging scenarios that degrade the performance. First, when the vertical lines have error in modeling the world, we have a large pose estimation error. Second, two lines can project to the same location in the image for some challenging viewpoints.

V. DISCUSSION

We showed a novel and a very efficient localization and tracking algorithm using an events sensor and an IMU. Our experiments showed that the translation and rotation errors are small and the computation time is very less. The DAVIS 240B events sensor enabled fast localization by providing high-speed events data, without which a 1000 Hertz inference algorithm can never be demonstrated. We have some interesting future avenues to explore. When the sensor moves towards the scene, the translation error is much larger than the case of parallel motion, because there are fewer events. In general, events are not fired when the edges are parallel to the motion direction. With more analysis of the sensor's motion model and better use of the previous poses, the localization could be improved. Since we are focusing on a line-based algorithm, dynamic humans or other objects should not cause too much problems. In future, we would also like to explore how robust the algorithm is with respect to dynamic objects.

Acknowledgments: We thank the anonymous reviewers, Jay Thornton, John Barnwell, William Vetterling, Jeroen Van Baar, Edward Adelson, Yuichi Taguchi, Tobi Delbruck, and Andrea Censi for useful discussions.

REFERENCES

[1] R. Benosman, C. Clercq, X. Largorce, S.H. Ieng, and C. Bartolozzi. Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, 2014.

[2] R. Benosman, S.H. Ieng, P. Rogister, and C. Posch. Asynchronous event-based hebbian epipolar geometry. *IEEE Trans. Neural Network*, 2011.

[3] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck. A 240 180 130 db 3 s latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 2014.

[4] J. Carneiro, S.H. Ieng, C. Posch, and R. Benosman. Event-based 3d reconstruction from neuromorphic retinas. *Neural Networks*, 2013.

[5] A. Censi and D. Scaramuzza. Low-latency event-based visual odometry. In *ICRA*, 2014.

[6] T. Delbruck, B Linares-Barranco, E. Culurciello, and C Posch. Activitydriven event-based vision sensors. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010.

[7] T. Delbruck and S.C. Liu. A silicon early visual system as a model animal. *Vision Research*, 2004.

[8] T. Delbruck, V. Villeneuva, and L. Longinotti. Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision. In *Proc. 2014 Intl. Symp. Circuits and Systems (ISCAS)*, 2014.

[9] Tobi Delbruck. Frame-free dynamic digital vision. In *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pages 21–26, 2008.

[10] F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. In *ECCV*, 2010.

[11] R. Hoffmann, D. Weikersdorfer, and J. Conradt. Autonomous indoor exploration with an event-based visual slam system. In *Europ. Conf. on Mobile Robots*, 2013.

[12] M. Kalantari, A. Hashemi, F. Jung, and J.P. Guedon. A new solution to the relative orientation problem using only 3 points and the vertical direction. *Journal of Mathematical Imaging and Vision*, 2011.

[13] H. Kim, A. Handa, R. Benosman, S.H. Ieng, and A.J. Davison. Simultaneous mosaicing and tracking with an event camera. In *BMVC*, 2014.

[14] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-form solutions to minimal absolute pose problems with known vertical direction. In *ACCV*, 2010.

[15] J. Lee, T. Delbruck, P. Park, M. Pfeiffer, C. Shin, H. Ryu, and B.C. Kang. Gesture-based remote control using stereo pair of dynamic vision sensors. In *International Conference on Circuits and Systems (ISCAS)*, 2012.

[16] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128x128 120 db 15 s latency asynchronous temporal contrast vision sensor. In *IEEE Journal of Solid-State Circuits*, 2008.

[17] S.C. Liu and T. Delbruck. Neuromorphic sensory systems. In *Current Opinion in Neurobiology*, 2010.

[18] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *Automatic Control, IEEE Transactions on*, 53(5):1203–1218, 2008.

[19] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza. Towards evasive maneuvers with quadrotors using dynamic vision sensors. In *ECMR*, 2015.

[20] E. Mueggler, B. Huber, and D. Scaramuzza. Event-based, 6-dof pose tracking for high-speed maneuvers. In *IROS*, 2014.

[21] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier. Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics. *IEEE Trans. Robotics*, 2012.

[22] S. Ramalingam and M. Brand. Lifting 3d manhattan lines from a single image. In *ICCV*, 2013.

[23] D. Weikersdorfer and J. Conradt. Event-based particle filtering for robot self-localization. In *Int. Conf. on Robotics and Biomimetics*, 2012.

[24] D. Weikersdorfer, R. Hoffmann, and J. Conradt. Simultaneous localization and mapping for event-based vision systems. *In Computer Vision Systems*, 2013.