Event-Based Monocular Camera Mapping

Benjamin Kuo
Carnegie Mellon University
benjamik@andrew.cmu.edu

Kazuya Otani
Carnegie Mellon University
kotani@andrew.cmu.edu

Max Hu
Carnegie Mellon University
yuanh@andrew.cmu.edu

Cyrus Liu
Carnegie Mellon University
xiyuanl1@andrew.cmu.edu

Abstract

A recent paper on Event-Based Odometry (EVO) was presented in [1]. EVO is a visual odometry pipeline using data from an Event-Based camera, where the output is not an intensity image but a stream of asynchronous events at microsecond resolution. Such cameras are capable of tracking fast motions while avoiding motion blur, and at the same time recovering a semi-dense 3D map of the environment. We implemented the mapping module of the EVO pipeline in MATLAB and demonstrated semi-dense 3D reconstructions from two event-based camera datasets.

1. Motivation

Event-based cameras are relatively new, and promise a variety of benefits over standard frame-based cameras. Drawn by its novelty, we attempt to implement a visual odometry pipeline using data gathered from event-based cameras.

1.1. Visual Odometry and SLAM

Visual odometry is the process of determining the robots position and orientation based on camera data. Simultaneous Localization and Mapping, or SLAM, additionally uses the sensor data to build a map of the world around the robot. The field has advanced significantly in the past decade, allowing robots to leave controlled laboratory environments and operate in the real world. However, there are still limitations. In particular, current localization algorithms have latencies of 50-200ms. This makes it difficult to perform agile maneuvers with time constants on a similar scale. While impressive maneuvers have been performed in motion-capture environments, fully autonomous robots with on-board localization have been limited to slow, measured actions. Event-based cameras have the potential to fix this problem, by allowing localization and mapping at up to 1MHz.

Figure 1. The DAVIS240C event-camera has a resolution of 240x180 pixels, with a dynamic range of 130dB and a typical latency of $100\mu s$ and $1ms$.

1.2. Event Cameras

Event based cameras are a relatively new technology, with a bulk of the work done in the last decade. These cameras are loosely based off of biological eyes, and react to changes in intensity. Because of this, event cameras natural edge detectors. Unlike traditional cameras, this vision sensor does not have a constant frame rate. Instead each pixel can fire asynchronously whenever there is deemed to be a large enough intensity change. Along with specialized hardware, the camera can produce events at an extremely high rate, on the order of microseconds. Furthermore, the camera can be used in cases with non-constant lighting due to the fact that it only tracks changes in intensity. The camera is able to be incredibly bandwidth efficient because it only sends changes in intensity, and never the full image. Because of all these new paradigms in a visual sensor, there is a need for new algorithms and techniques to process the data that the camera produces.

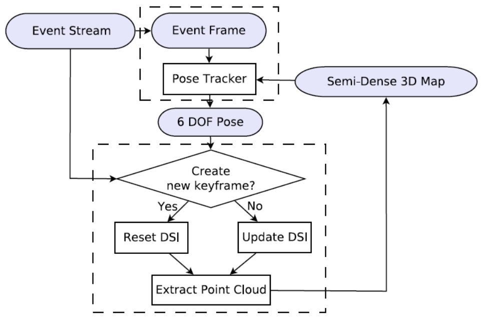

Figure 2. EVO algorithm pipeline

2. Related work

2.1. Event-based Visual Odometry (EVO) [1]

This paper is what grabbed our initial interest in using event-based cameras. The paper lays out a pipeline for doing pose tracking while simultaneously building a semi-dense map of the environment. The underlying paradigm of the methods used stem back to the work on PTAM. This paper does used a frame to frame tracker thread as well as a map building thread as well. Unlike PTAM however, the methods used here are not feature-based, but rather direct, using all of the information coming from the camera. Though our work does not cover the tracking portion of this paper, we will note that the underlying tracking algorithm revolves around the Lucas-Kanade Algorithm with projections of the semi-dense map, which can converge fast enough due to the rate at which data is received. This differs from PTAM which uses projections of features, and goes through a coarse-to-fine search to arrive at the pose estimate. EVOs mapping portion was built upon another piece of work, [2] (EMVS).

2.2. Event-based Multi-View Stereo (EMVS) [2]

The EMVS algorithm provides a simple method for creating semi-dense depth maps using the data from a moving event-based camera. It exploits two properties on an event camera: (i) its ability to respond to scene edges, (ii) its rapid (almost continuous) responses as the camera moves. The method that EMVS uses is based on the Space-Sweep approach for Multi-View Stereo [3]. The Space-Sweep approach has the advantage that it does not require features in the traditional computer vision sense, and does not need explicit data association. The main idea of Space-Sweep is that prominent features in the environment (edges, corners) will be seen from multiple camera poses. If we project a ray out into the world from every camera pose in the directions where it detected features, we can assume that the points with the highest concentration of ray intersections is a feature in the world. The way this is implemented is with a Disparity Space Image (DSI), which discretizes the space in the environment into planes. The authors of EMVS chose to use a keyframe as the origin for their DSI. The algorithm consists of three steps:

- Warping (back-projecting) image features onto the discretized planes.
- Detecting local maxima in the DSI to identify features.
- Merging the features into the depth map.

With these simple steps, EMVS allows for the creation of an accurate semi-dense depth map. Note that this method does not need complex 3D point fusion methods or regularization to produce compelling large-scale 3D reconstruction results. These impressive results convinced us to focus on the mapping module of EVO.

2.3. Most Recent Work

The event-based camera dataset contains data from the inertial measurement unit on the DAVIS240C. Although EVO and EMVS do not take advantage of this data, recent work from the Robotics and Perception Group from the University of Zurich has proposed a continuous time formulation to provide a principled method of incorporating high frequency inertial measurements into pose estimation [6]. In this work, the pose trajectory is approximated by a smooth curve using cubic splines, as opposed to previous works which directly represented pose trajectories with discrete poses. A cubic spline representation significantly reduces the number of variables needed for trajectory estimation, and allows for natural interpolation. Using this method, the authors were able to estimate trajectories with a mean position error of less than 1% of the average scene depth, and mean orientation error of less than 1 degree.

3. Methods

For this project, we were not working with a physical camera. Within the past year, the Robotics and Perception Group at the University of Zurich released several data sets of event cameras, both real and synthetic/simulated [7]. Given the price and availability of a real camera, we opted to use two of the real life data sets to explore the use of this data for semi-dense mapping. The dataset contains data collected with a DAVIS240C from iniLabs. It includes events, images, IMU measurements and camera calibration from the DAVIS as well as ground truth camera poses from a motion-capture system.

We focused on the mapping module of the EVO pipeline. We selected datasets that provided ground truth data for the position and orientation of the camera so that we can immediately test the mapping capabilities without having to first implement the tracking portion of the problem as well. Figure 2 shows a representation of the original EVO pipeline;

we simply removed the pose tracker portion and replaced it with ground truth poses. A more detailed flowchart of our implementation is shown in Figure 3. The resulting algorithm is presented in the steps below.

3.1. Step 0: Keyframe Generation

The algorithm works by incrementally updating the map as new keyframes are created. The keyframe acts as a reference point for nearby camera poses. As in the EVO paper, we determine the need for a new keyframe based off of the distance from the last keyframe to the current pose. As an implementation detail, since we have the ground-truth data for our pose, we do not need to bootstrap the system as the original paper suggests. Instead we simply treat our first event image as a keyframe.

3.1.1 Step 0.1: Generate a Disparity Space Image

The addition of points to our global map starts with creating a Disparity Space Image (DSI). A DSI is a discrete 3D representation of the space in front of a camera. The nature of the space can be visualized as a conical voxel grid that projects out from the camera's frame. The best representation of this is shown in Figure 4. This discrete voxel grid is used essentially as a 3D histogram in the following steps. Our function for generating a new DSI took as inputs: minimum depth, maximum depth, and number of planes. We used 50 planes, following the implementation in the EVO paper. We manually chose minimum and maximum depths by inspecting the camera footage and approximating the depths (as far as we could tell, this was the method used in the original EVO implementation as well). The planes were generated to be uniformly spaced in inverse depth. This meant that planes closer to the minimum depth were closer together in depth, and the planes got further apart as the depth increased. This spacing makes the implicit assumption that as we get closer to the camera, more features will be seen, and thus greater resolution will be needed. In addition to the initialization of the DSI, a few other pieces of information are generated: the depth of each of the planes in the DSI is saved, as well as the scaling of each pixel in each layer.

3.2. Step 1: Generate Event Image

As mentioned in the EVO paper, the approach to generating event images is to accumulate events into a blank image. This accumulation continues until a certain threshold of a number of events are accumulated. The original implementation suggests that on average, they receive about 2000 events per image, and their threshold is based off of the current number of points in the global map, or whether or not a pixel has fired twice. Unfortunately, without their tracking techniques, we were unable to generate event images in this fashion. We simply accumulate events between each of the ground-truth readings. Though this somewhat reduces the advantage of the event based camera, the rate at which we were receiving images was about 200Hz. This is still much faster than typical cameras, and results in a much sparser event image.

3.3. Step 2: Back-project Event Image into DSI

The authors of EMVS refer to Collins original Space-Sweep method [3] in their paper. However, we found that the homography equations in [4] were better suited for this application. We calculated homography matrices to map points in the event image to points on each plane in the DSI, which allowed us to efficiently back-project event images.

The family of depth planes is defined as

$$\Pi_m = [n_m^T - d_m] \, for \; 1, \ldots, M \tag{1}$$

where n_m is the unit length normal of the plane and d_m is the depth of the plane to the origin of the key frame. The homography from the depth planes to image plane of camera is described as

$$H_{\Pi_m, P_k} = K(R_k^T + \frac{R_k^T C_k n_m^T}{d_m})K^{-1} \tag{2}$$

where K is the intrinsic matrix, R_k and C_k is the rotation and translation matrix of camera P_k with respect of the key frame camera. The use of the intrinsic matrix account for the fact that the principal point does not coincide with the image plane's origin.

To proceed with the back-projections, we use the inverse of H_{Π_m,P_k} which describes the homography from the image plane camera P_k to the depth planes Π_m.

3.4. Step 3: Find local maxima in the DSI

In the third step of the of the mapping pipeline, we obtain a semi-dense depth-map in the virtual camera frame by determining whether or not a 3D point is present in each DSI voxel. Scene points are most likely to occur in regions where viewing rays from multiple images almost intersect. The DSI represents a 3D-histogram that stores a ray-density function. Hence the scene points lie at the local maxima of the ray-density function.

The local maxima of the DSI $f(X)$ is found via a two-step procedure: First, we collapse the 3D DSI into a 2D depth-map $Z * (x, y)$, and an associated confidence map $x(x, y)$ by recording the locations and magnitudes of the local maxima. Then we filter out less confident pixels in the depth map using Adaptive Gaussian Thresholding: A pixel is selected if $c(x, y) > T(x, y)$, where $T(x, y) = c(x, y) \times G_\sigma(x, y) - C$. In practice, figuring out parameters for $G(x, y)$ and C was a bit of a task since the values mentioned in literature did not seem to work well. Our implementation uses a 5×5pixel Gaussian with $\sigma = 0.5$ and $C = -2$.

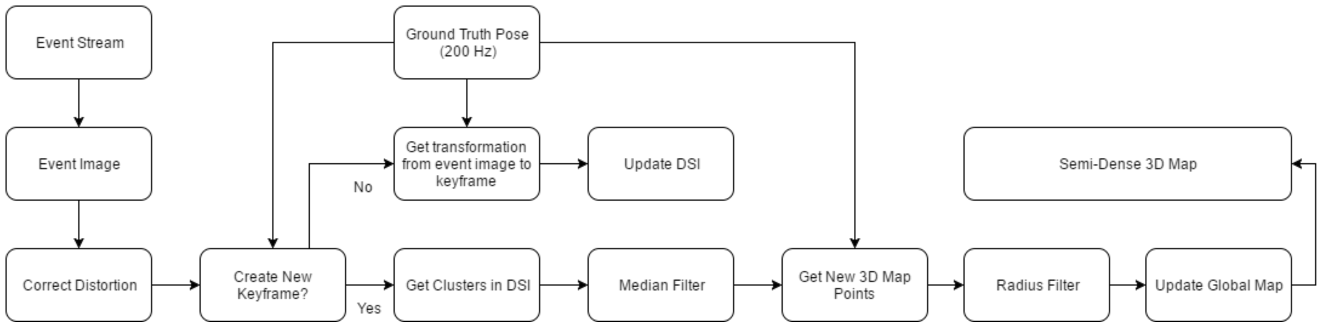

Figure 3. This figure shows an overview of our implementation of mapping with event-based data. The inputs to the system are the event stream and ground truth poses of the camera, and it outputs a semi-dense 3D map, and optionally, depth maps for each keyframe can be retrieved as well.

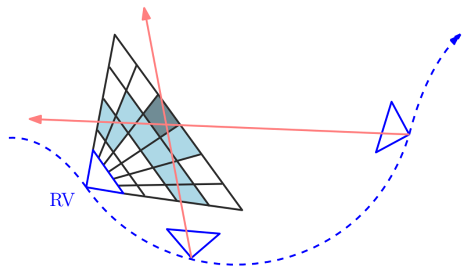

Figure 4. The DSI acts as a counter for ray intersections.

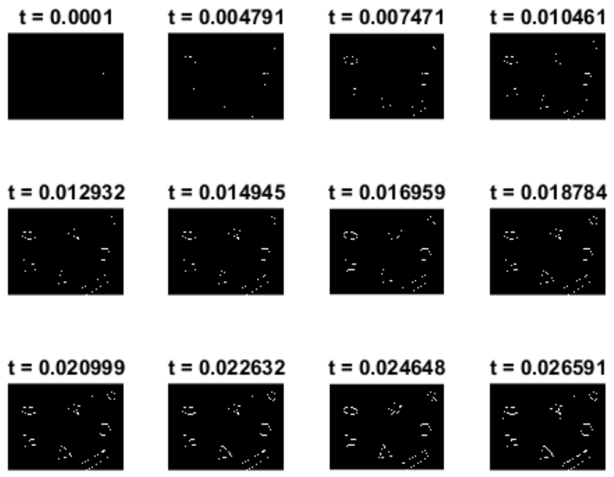

Figure 5. The process of building an event image over time. One can see the events being accumulated.

3.4.1 Step 3.1: Filtering

To remove outliers, we performed a two-stage filtering process on the features extracted from the DSI. First, we run a median filter over the depth map, only considering the non-zero values. This depth map is converted into a 3D

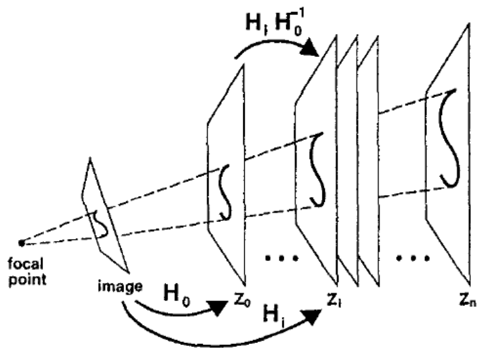

Figure 6. The projection of an event image onto the planes of the DSI [3].

point cloud using the depths and homographies between the planes, which are saved from step 0.1. We then run a radius filter on the resulting point cloud. The filter works for each point by counting the number of points within a specified radius. If a point has fewer neighbors than a certain threshold, it is considered an outlier and eliminated. We found that the two filter parameters (neighbor radius, neighbor count threshold) required some hand-tuning for each dataset.

3.5. Step 4: Recover 3D points

After step 3, the resulting data representation is essentially a depth map. The depth map however encodes the information only in terms of pixel coordinates, and the layer of the DSI the maxima appear on. In the creation of the DSI, the layers are defined by their distance from the keyframe pose. To recover 3D points, we project the pixels from the depth map onto their respective planes.

3.6. Step 5: Global Map Fusion

In this step, we transform the 3D points from the keyframe to the world frame, then add each point to the global map. As mentioned above, we (like the authors of

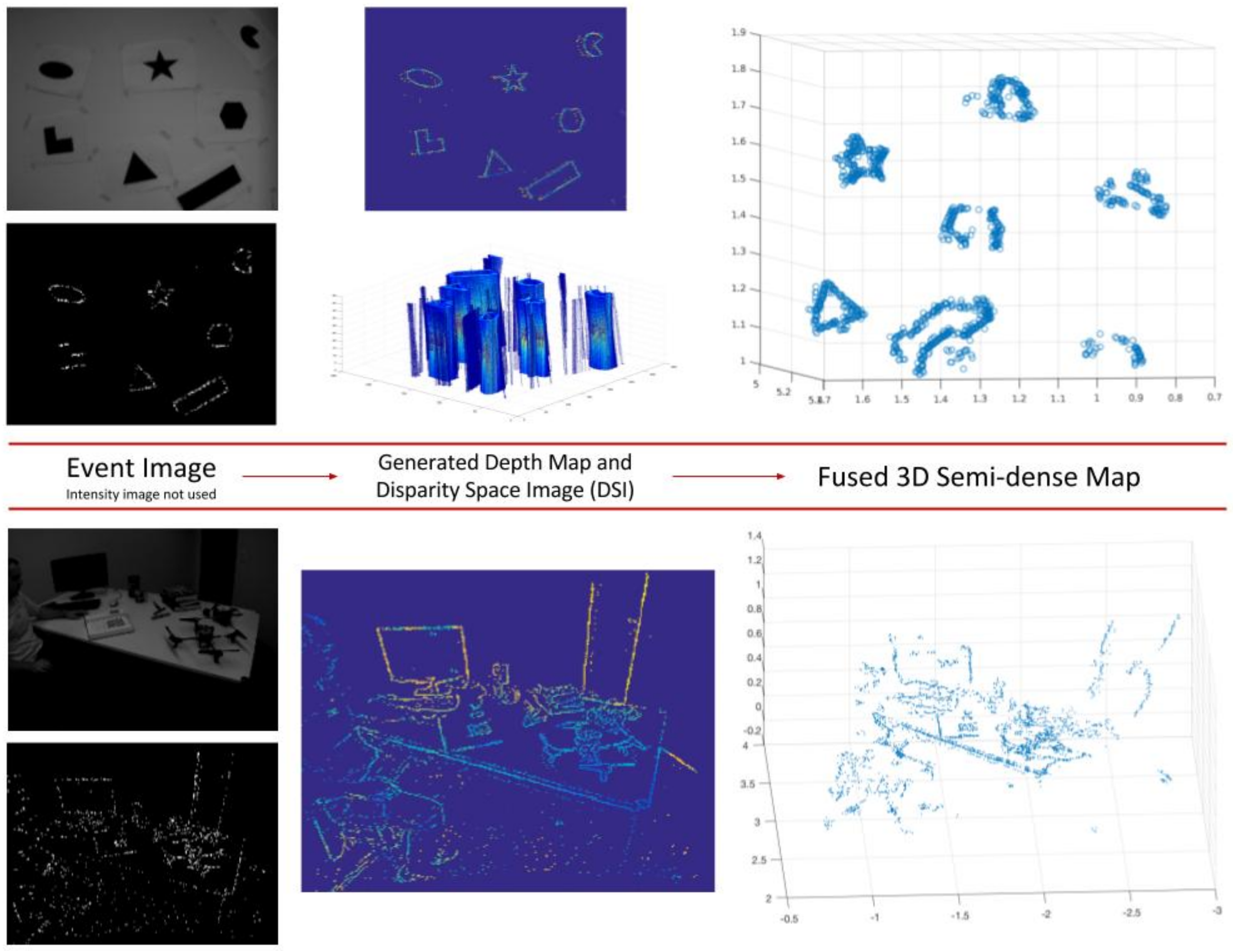

Figure 7. This figure shows part of the pipeline of the process of building the semi-dense maps. Note we do not use the intensity images at all, are we are just showing them to provide context for the maps.

EMVS) did not employ any complex fusion or regularization methods in this step. The benefit of event cameras is that the dense, continuous stream of events provides ideal data for the Space-Sweep algorithm, and there are not much errors in the resulting map.

4. Results

Using the method described above, we demonstrated semi-dense 3D reconstructions from two datasets.The qualitative results are shown in Figure 7. The 3D representation of the scene can clearly be recovered with this method, with just the event stream and ground truth poses as inputs.

5. Challenges and Lessons Learned

Our primary challenge was the lack of source code or documentation available for working with event-based data, due to the novelty of the event-based cameras and datasets. As is typical with academic papers, implementation details were not readily apparent or available in literature, and the few that were mentioned were simple references to prior publications. We found that there were ambiguities within and between papers on several major implementation details. We dealt with this issue by working to understand the algorithm deeply, and seeking additional references on the topic that were not mentioned in the original EMVS paper.

Another issue we had during testing was the performance limitations of MATLAB. In particular, we found that the built-in `imwarp` and `pointCloud` functions were taking about 90% of our compute time. Ideally, these algorithms would be written in C++ for real-time performance.

Finally, the reliance on the ground truth data provided by the data set was at a lower frequency than expected, and this lead to some unforeseen issues. The first problem was the rate the data came in. Even though 200Hz is relatively fast, during very rapid movements of the camera, there was a slight ”motion-blur” effect on our event images. This is a result of many more events coming into the camera in between ground truth readings. Not only did this result in a ”thicker” representation of the edges, it also means that we were disregarding many data that were coming in. In addition to the ground truth data rate, we also had some issues with the axis representations as well. We had to understand the reference axes of the global frame of the camera and manipulate our data to conform with that. Had we implemented the EVO tracking based approach, we would have only had to do it relative to our start pose.

6. Conclusions

In this project, we demonstrated our implementation of an algorithm for semi-dense 3D reconstruction using an event camera. Based on the results we achieved with this relatively simple algorithm, we believe that this is a very promising direction for research in mobile robotics. Visual odometry and SLAM with event cameras will allow robots to localize better and move at higher speeds in potentially unknown environments. We hope to see wider adoption of event cameras in the mobile robotics community.

On a broader level, this project served as a difficult hands-on introduction to geometric methods in computer vision. It is interesting to see that the combination of a few simple geometric equations can produce a full SLAM algorithm. While machine learning, particularly neural networks, has produced record-breaking results in recent years, geometric methods seem to provide faster and more analytically provable algorithms for specific applications. In addition, there have been recent efforts to incorporate geometric knowledge of the world into learning-based algorithms. We look forward to seeing joint development of learning-based and geometry-based approaches in computer vision in the future.

7. Work Division

- Ben - Generation of event images, recovery of 3D points from depth map
- Kazu - Keyframe initialization, back-projection of event images into DSI
- Max - Global map fusion, DSI filtering
- Cyrus - Filtering/local maxima detection in DSI, back-projection of event images into DSI

References

[1] Rebecq, Henri, et al. ”EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time.” IEEE Robotics and Automation Letters 2.2 (2017): 593-600.

[2] Rebecq, Henri, Guillermo Gallego, and Davide Scaramuzza. ”EMVS: Event-based Multi-View Stereo.” British Machine Vision Conference (BMVC). No. EPFL-CONF-221504. 2016.

[3] Collins, Robert T. ”A space-sweep approach to true multi-image matching.” Computer Vision and Pattern Recognition, 1996. Proceedings CVPR’96, 1996 IEEE Computer Society Conference on. IEEE, 1996.

[4] Gallup, David, et al. ”Real-time plane-sweeping stereo with multiple sweeping directions.” Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on. IEEE, 2007.

[5] Brandli, Berner, Yang, Liu, Delbruck, ”A 240 180 130 dB 3 s Latency Global Shutter Spatiotemporal Vision Sensor.” IEEE Journal of Solid-State Circuits, 2014.

[6] Mueggler, Elias, et al. ”Continuous-Time Visual-Inertial Trajectory Estimation with Event Cameras.” arXiv preprint arXiv:1702.07389 (2017).

[7] Mueggler, Rebecq, Gallego, Delbruck, Scaramuzza, ”The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM.” International Journal of Robotics Research, Vol. 36, Issue 2, pages 142-149, Feb. 2017.