# Realtime Time Synchronized Event-based Stereo

Alex Zihao Zhu[0000−0002−2195−014X], Yibo Chen[0000−0001−9542−7741], and
Kostas Daniilidis[0000−0003−0498−0758]

University of Pennsylvania, Philadelphia PA 19104, USA

**Abstract.** In this work, we propose a novel event based stereo method which addresses the problem of motion blur for a moving event camera. Our method uses the velocity of the camera and a range of disparities to synchronize the positions of the events, as if they were captured at a single point in time. We represent these events using a pair of novel time synchronized event disparity volumes, which we show remove motion blur for pixels at the correct disparity in the volume, while further blurring pixels at the wrong disparity. We then apply a novel matching cost over these time synchronized event disparity volumes, which both rewards similarity between the volumes while penalizing blurriness. We show that our method outperforms more expensive, smoothing based event stereo methods, by evaluating on the Multi Vehicle Stereo Event Camera dataset.

**Keywords:** Event Cameras, Stereo Depth Estimation, 3D Computer Vision.

## 1 Introduction

Event cameras are neuromorphically inspired asynchronous visual sensors that register changes in the log intensity of the image. When such a change is detected, the camera immediately returns an event, $(x, y, t, p)$, to the host, consisting of the pixel position of the change, $x, y$, timestamp, $t$, accurate to microseconds, and polarity, $p \in \{-1, 1\}$, indicating whether the intensity decreased or increased. Over time, the output of the camera can be represented as a constant stream of events. The asynchronous nature of the cameras, combined with with extremely high temporal resolution, allow for high speed, low latency measurements, in situations where traditional cameras may fail. In addition, the cameras exhibit very high dynamic range (120dB vs 60dB for traditional cameras), allowing them to operate in a number of challenging lighting environments. Finally, the cameras also have much lower bandwidth and power consumption. One interesting use case for these cameras is stereo depth estimation, where they can provide high speed depth information for tasks such as obstacle avoidance and high speed 3D tracking.

Supplementary video: `https://youtu.be/4oa7e4hsrYo`.

However, a major problem facing general event-based methods is that of time synchronization. That is, events generated at different times may correspond to the same point in the image space, but will appear at different pixel positions due to the motion of the point. This problem manifests itself in two ways. Between cameras, this problem is analogous to having unsynchronized cameras for frame based stereo methods, where the epipolar constraint breaks down due to the motion between the images, and occurs when events are not generated at the same time between the two cameras. Within a single camera, this causes effects similar to the motion blur seen in frame based images. For the stereo matching problem, which is often solved using appearance based similarity metrics, this blurring is highly detrimental, as it often alters the appearance of each image differently. A number of event based stereo methods have approached these problem with asynchronous methods (e.g. [11,15,17]), which process each event independently. However, these methods must either forego the information provided by the spatial neighborhood around each event, or use fine tuned temporal windows to once again ensure time synchronization, as there are no guarantees that neighboring events were generated at a similar time.

In this work, we show that this problem can be resolved for stereo disparity matching if the velocity of the camera is known. In particular, we propose a novel event disparity volume for events from a stereo event camera pair, that uses the motion of the camera to temporally synchronize the events with temporal interpolation at each disparity. Our method takes inspiration from past works from Zhu et al. [19], Gallego et al. [5] and Rebecq et al. [14,13], which took advantage of the high temporal resolution of the events to remove motion blur from an event image using an estimate of the motion in the scene, such as optical flow or camera pose. To estimate optical flow, we use the motion field equation, given camera velocity and a set of disparities, and similarly interpolate the position of the events at each disparity to a single point in time, which we represent as a novel temporally synchronized event disparity volume. We show that, in addition to removing motion blur at the correct disparities (where the motion field equation is valid), this volume allows us to disambiguate otherwise challenging regions in the image by inducing additional motion blur.

We then define a novel matching cost over this event disparity volume, which rewards similarity between patches, while penalizing blurriness inside the patch. We show that this cost function is able to robustly distinguish the true disparity, while being extremely cheap to compute, using only bitwise comparison operations over a sliding window.

Our method, implemented in Tensorflow, runs in realtime at 40ms on a laptop grade GPU, with significant further optimizations available. We evaluate our results on the Multi Vehicle Stereo Event Camera dataset[1] [20], and show significant improvements in disparity error over state of the art event based stereo methods, which rely on additional, more computationally expensive, smoothness regularizations.

The main contributions of this paper are summarized as:

[1] Dataset website: `https://daniilidis-group.github.io/mvsec/`

– A novel method for using camera velocity to generate a time synchronized event disparity volume where regions at the correct disparity are in focus, while regions at the incorrect disparity are blurred.
– A novel block matching based cost function over an event disparity volume that jointly rewards similarity between left and right patches while penalizing blurriness in both patches.
– Evaluations on the Multi Vehicle Stereo Event Camera dataset, with comparisons against other state of the art methods, and evaluations of each component of the method.

## 2 Related Work

Early works in stereo depth estimation for event cameras, such as the works by Kogler et al. [7] and Rogister et al. [15], attempted to perform matching between individual events in a fully asynchronous fashion, using a combination of temporal and spatial constraints, which Kogler et al. showed to perform better than basic block matching between pairs of event images. However, these methods suffer from ambiguities in matching when single events are considered.

To address these ambiguities, Camuas-Mesa et al. [2] use local spatial information in the form of local Gabor filters as features, while in [3], they track clusters of events to aid in tracking with occlusion. Zou et al. [22] use a novel local event context descriptor based on the distances between events in a window, which they extend in [23] to produce a dense disparity estimate. Similarly, Schraml et al. [16] use a cost based on the distance between events to generate panoramic depth maps.

In addition, several works have applied smoothing based regularizations to constrain ambiguous regions, which have seen great success in frame based stereo. Piatkowska et al. [11,12], have applied cooperative stereo methods [8] in an asynchronous fashion, while Xie et al. [17,18] have adapted belief propagation [1] and semiglobal matching [4], respectively, to similar effect. These regularizations have shown significant improvements over the prior state of the art.

These prior works have all shown promising results for event-based stereo matching, but do not explicitly handle the time synchronization problem without abandoning the rich spatial information around each pixel.

The problem of time synchronization has been approached in other problems, where it has been used to remove motion blur from event images. Zhu et al. [19] and Gallego et al. [5] use this synchronization as a cost function to estimate optical flow and angular velocity, respectively. Rebecq et al. [13] use the pose of a single camera from multiple views to generate a disparity space volume, in which the correct depth is similarly deblurred. Rebecq et al. [14] use a state estimator with pose and sparse depths to generate ‘motion compensated’ event images, on which they perform feature tracking. More recently, Mitrokhin et al. [9] use the synchronized images to perform object detection and tracking.

## 3 Method

The underlying problem of stereo disparity matching can be thought of as a data association problem. That is, to find correspondences between the points in the left and right images, at a given point in time. In this work, we assume that the cameras are calibrated and rectified, so that every epipolar line in both images is parallel to the x axis, and the correspondence problem is reduced to a 1D search along the x dimension. While some prior works such as [15] in the event based literature have tried to perform matching on an event by event basis, we use the spatial neighborhood around each pixel for a more detailed and robust matching, by making a locally constant depth assumption.

It is possible to perform matching on event images generated directly from the event positions. However, such an image generated from the raw events is very prone to motion blur, unless the time window is carefully selected, with a method such as the lifetime estimation in [10].

Motion blur is generated when events are captured at different points in time, such that events corresponding to the same point in the image may occur at different pixels due to the motion of that point. However, the works in [19], [5] and [14] show that motion blur can be removed from an event image if the optical flow for each pixel is known. In Sec. 3.1, we leverage this technique to both remove motion blur at the correct disparities, while further blurring the events at incorrect disparities. We then describe a novel event disparity volume representation of these time shifted events in Sec. 3.2, on which we apply a novel cost function that leverages this focus-defocus effect to allow us to discriminate the true disparity at each pixel, as described in Sec. 3.3. Finally, Sec. 3.4 discusses methods to then use the cost function to estimate the true disparity at each pixel.

An overview of the method can be found in Fig. 1

### 3.1 Time Synchronization through Interpolation

For a given disparity, $d$, we can approximate optical flow using the motion field equation, with an assumption of known camera velocity. The motion field equation describes the relationship between the linear ($\mathbf{v}$) and angular ($\boldsymbol{\omega}$) velocity of a camera, depth $Z$ of a point $(x, y)$, which we treat here as a function of disparity, $d$, and the motion of the point in the image, which we approximate to be the optical flow ($\dot{x}_i$, $\dot{y}_i$):

$$\begin{pmatrix}\dot{x}_i(d)\\ \dot{y}_i(d)\end{pmatrix} = \frac{1}{Z(d)}\begin{bmatrix}-1 & 0 & x_i\\ 0 & -1 & y_i\end{bmatrix}\mathbf{v} + \begin{bmatrix}x_i y_i & -(1+x_i^2) & y_i\\ 1+y_i^2 & -x_i y_i & -x_i\end{bmatrix}\boldsymbol{\omega} \tag{1}$$

$$Z(d) = \frac{fb}{d} \tag{2}$$

where $f$ is the focal length of the camera and b is the baseline between the two cameras.

Assuming that the optical flow for each pixel is constant within each time window, we can then estimate the position of a point generating an event

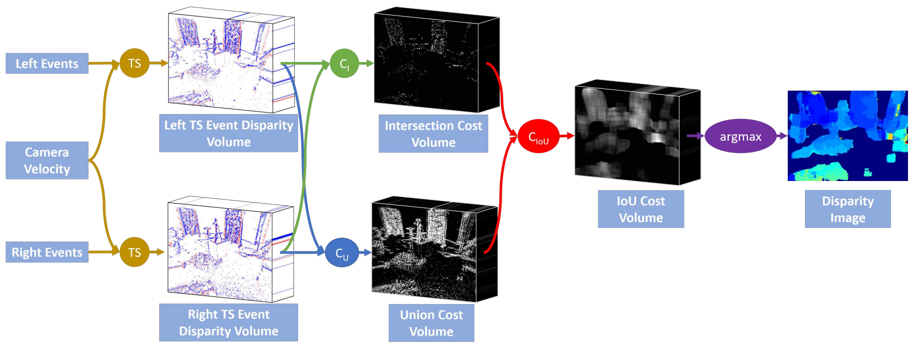

Fig. 1: Overview of our method. Given an input of left and right events and camera velocity, left and right time synchronized event disparity volumes are generated (Sec. 3.1 and 3.2). The intersection and union costs are calculated by combining the two disparity volumes, and the final IoU cost volume is computed (Sec. 3.3). Finally, the disparity is computed in a winner takes all scheme over the IoU cost volume (Sec. 3.4). Best viewed in color.

$(x_i, y_i, t_i, p_i)$ at a constant time $t'$ with linear interpolation:

$$\begin{pmatrix} x'_i(d) \\ y'_i(d) \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} \dot{x}_i(d) \\ \dot{y}_i(d) \end{pmatrix} (t' - t_i) \tag{3}$$

Assuming a static scene and accurate velocities and disparities, the set of time synchronized events, $\left\{ \left( x'_i(d) \; y'_i(d) \; t' \; p_i \right) \right\}$, is assumed to have no motion blur for all $x_i$ and $y_i$ with disparity $d$.

## 3.2 Time Synchronized Event Disparity Volume Generation

However, the true depth for each pixel is unknown for this problem. Instead, we select a range of disparities over which to search, and apply (3) to the set of events from the left camera for every disparity within the range.

At each disparity level, $d$, we generate an image based on the time shifted events, where a pixel with more positive events is set to 1, more negative events is set to -1, and no events is set to 0. Note that the time shifted event positions are rounded to the nearest integer to index into the image.

$$I_L(x, y, d) = \text{sign}\left( \sum_i p_i \right) \tag{4}$$

$$\begin{aligned} i \in & \{ i | (x'_i(d), y'_i(d)) = (x, y) \} \\ p_i \in & \{-1, 1\} \end{aligned}$$

This is similar to standard methods that generate images by summing events at each pixel, but the additional sign operator allows the image to be robust to the left or right camera generating more events at each pixel than the other.

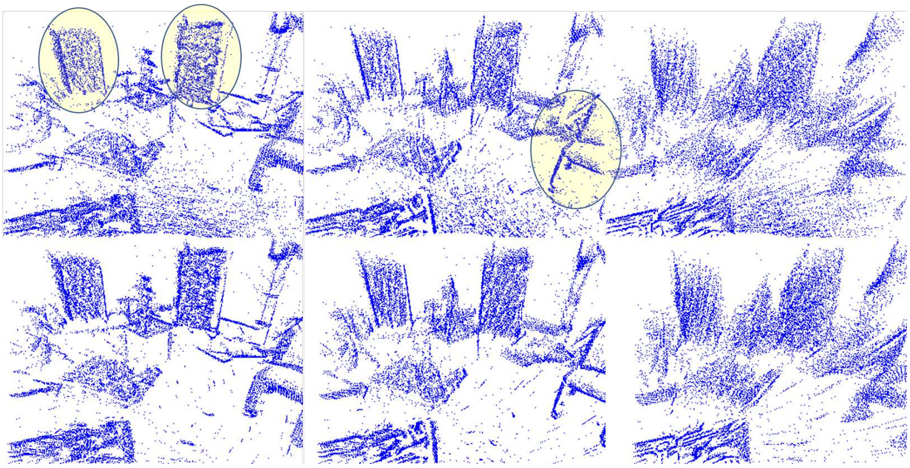

Fig. 2: Sample slices of the left (top) and right (bottom) time synchronized event disparity volumes at disparities 6 (left), 16 (middle) and 31 (right). Only positive pixels are shown for clarity. At disparity 6, the boards at the back are in focus, while at disparity 16, the chair in the front is in focus (both circled in yellow). The other features at the wrong depth are blurred. The right slices have been shifted horizontally by the disparity, as in (5), so that corresponding points should be at the same x position in both images. Best viewed in color.

The result is a 3D volume for the left camera, where each slice in the disparity dimension represents the images generated according to (4), using the disparity corresponding to that slice. That is, when the camera moves with some linear velocity, this flow would have a deblurring effect on points where the pixel position matches the disparity, and potentially apply further blurring on points where the disparity is incorrect. In the case when the camera's motion is pure rotation, the flow will produce unblurred images at each disparity slice.

We apply a similar operation to the events from the right camera, except that the x position of each shifted event is further shifted by the disparity at each level:

$$I_R(x,y,d) = \text{sign}\left(\sum_i p_i\right) \tag{5}$$
$$i \in \{i | (x_i'(d) + d, y_i'(d)) = (x,y)\}$$
$$p_i \in \{-1, 1\}$$

This generates a set of disparity volumes similar to traditional plane sweep volumes, where the potential matching right pixel corresponding to $I_L(x,y,d)$ is $I_R(x,y,d)$. We show some example slices of this volume in Fig. 3, where the blurring and deblurring effects can be clearly seen.

## 3.3 Matching Cost

Finally, we apply a novel sliding window matching cost that leverages both the deblurring and blurring effects of Sec. 3.1. First, it penalizes windows with many events, as this would indicate areas with an incorrect disparity due to the blurring incurred by the temporal interpolation. Given a local spatial window $W(x, y, d)$ around a pixel $(x, y)$ at a given disparity $d$, we encode this using a union term, defined as:

$$C_U(x, y, d) = \sum_{x^*, y^* \in W(x,y,d)} I_L(x^*, y^*, d) \cup I_R(x^*, y^*, d) \tag{6}$$

$$a \cup b = \begin{cases} 1 & a \neq 0 \text{ or } b \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

We carefully choose the union operator instead of addition, in order to not double penalize pixels with events in both volumes.

Second, the cost rewards windows that are similar. That is, we would like pixels between the two images to have events with the same polarity. We encode this using an intersection term, defined as:

$$C_I(x, y, d) = \sum_{x^*, y^* \in W(x,y,d)} I_L(x^*, y^*, d) \cap I_R(x^*, y^*, d) \tag{7}$$

$$a \cap b = \begin{cases} 1 & a = b \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

This is similar to the tri-state logic error function presented in [7], except we explicitly do not reward pixels that are both 0 in the intersection term, as we only want to capture associations between events, and not between pixels without events.

The final cost, then, can be thought of as an analogy to the intersection over union cost:

$$C_{IoU}(x, y, d) = -\frac{C_I(x, y, d)}{C_U(x, y, d)} \tag{8}$$

Minimizing this final cost will implicitly maximize the similarity between the two windows, while minimizing the blurring in each. By computing this cost function at every pixel and disparity, we generate a cost volume, where each element $(x, y, d)$ in the volume contains the cost of pixel $(x, y)$ being at disparity $d$.

## 3.4 Disparity Estimation

Given the cost volume, the fastest way to obtain the estimate for the true disparity at each pixel is to compute the argmax across the disparity dimension of the cost volume, in a winner takes all fashion:

$$\hat{d}(x, y) = \arg\min_d \, C_{IoU}(x, y, d) \tag{9}$$

However, we can also apply any traditional optimization method for stereo disparity estimation over the cost volume, such as semi-global matching [4] or belief propagation [1].

### 3.5 Outlier Rejection

While the cost function in (8) was relatively robust in our experiments, there were still some regions of the image where it was unable to resolve the correct disparity, which we need to remove from the final output. In particular, we found that pixels with a low final IoU cost typically corresponded to pure noise in the image, where the number of intersection matches was low compared to the number of events in the window. Therefore, any disparities with $C_{IoU}$ less than a parameter $\epsilon_c$ are considered outliers. In addition, windows with a low number of events do not provide enough support to find a meaningful match, and so we consider outliers any disparities with $C_U$ less than $\epsilon_n \times \|W\|$, where $\epsilon_n$ is a parameter and $\|W\|$ is the number of pixels in the spatial neighborhood.

## 4 Implementation Details

In our experiments, unless otherwise stated, we use a disparity range ranging from 0 to 31 pixels, and a square window with side length of 24 pixels. For outlier rejection, $\epsilon_c$ and $\epsilon_n$ were both set to 0.1. At each time step, a constant number of events is passed to the algorithm. For our experiments, we used 15,000 events.

As every step of the algorithm is vectorizable with matrix notation, the algorithm was efficiently implemented on GPU in Tensorflow. In particular, (2) and (3) are implemented as a matrix operations, (4) and (5) are performed using scatter_nd, and the costs in (6) and (7) are computed by computing the costs for each pixel at each disparity, and applying two 1D depthwise convolutions with a kernel of ones of the same length as the window size (one along the rows, one along the columns).

With all operations fully vectorized, the algorithm takes 40ms to run on a laptop NVIDIA 960M GPU, including transfer time to the GPU. With further optimizations and an implementation in raw CUDA or OpenCL, we expect this time to reduce further. This corresponds to a runtime of around 2.7$\mu$s per event, compared to the 0.65-2ms reported in [17]. However, it should be noted that the competing methods were implemented in MATLAB on CPU, and would almost certainly see speed improvements if ported to other languages/devices. In addition, our method is relatively insensitive to the number of events, as a large proportion of the run time ($\sim$40%) is consumed in the sliding window cost. For example, processing a window of 30,000 events takes 46ms to run, corresponding to a runtime of 1.53$\mu$s per event.

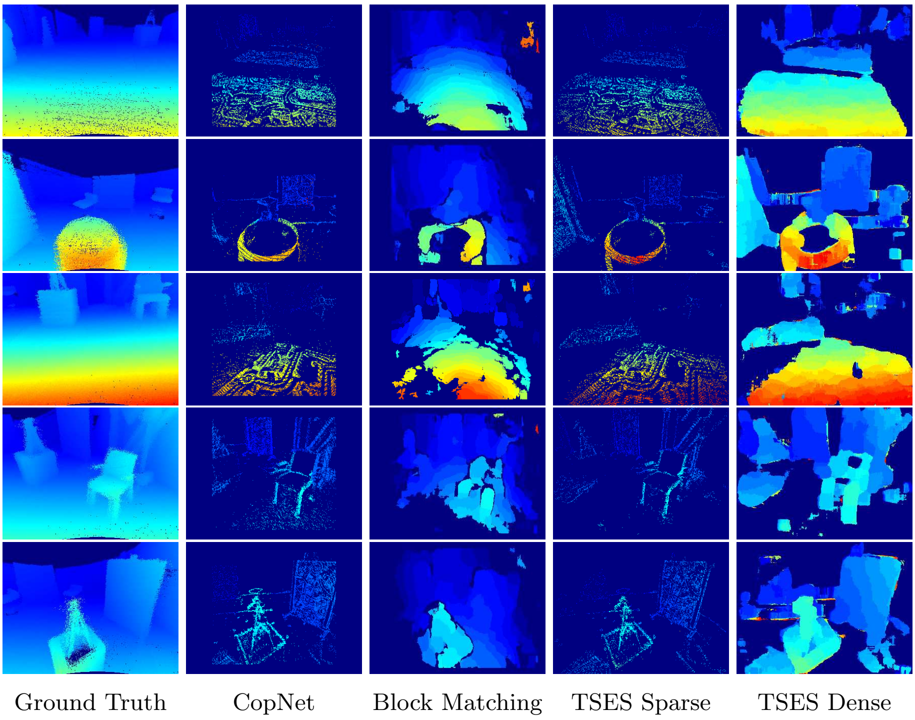

Fig. 3: Sample outputs from TSES (our method), compared against CopNet and block matching, with ground truth from MVSEC. Pixels without disparities are dark blue. Note that the border of the CopNet and block matching results are empty due to the window size. Quantitative results were only computed over points with disparities. Best viewed in color.

## 5 Experiments

### 5.1 Data

We evaluated our algorithm on the Multi Vehicle Stereo Event Camera (MVSEC) dataset [20]. MVSEC provides data captured from a stereo event camera pair, along with grayscale images and ground truth depth and pose of the cameras. We tested our method on the indoor_flying sequences, and evaluated against the provided ground truth depth maps. These sequences were generated from a stereo event camera pair mounted on a hexacopter, and flown in an indoor environment, with ground truth generated from lidar measurements. In particular, we used the following depth map frames (zero index) from these indoor_ flying sequences for evaluation: indoor_flying1: 140-1200, indoor_flying2: 120-1420, indoor_flying3: 73-1616. These frames were selected to exclude the takeoff and landing frames where the ground is closer than our selected maximum disparity.

| | Mean Disp. Error (pix) | | | Mean Depth Error (m) | | | % Disp. Err < 1 | | |
|---|---|---|---|---|---|---|---|---|---|
| | IF1 | IF2 | IF3 | IF1 | IF2 | IF3 | IF1 | IF2 | IF3 |
| TSES | 0.89 | 1.98 | 0.88 | 0.36 | 0.44 | 0.36 | **82.3** | **70.1** | **82.3** |
| CopNet | 1.03 | 1.54 | 1.01 | 0.61 | 1.00 | 0.64 | 70.4 | 52.8 | 70.6 |
| BM | **0.73** | **1.02** | **0.82** | **0.23** | **0.21** | **0.27** | 79.5 | 65.2 | 74.3 |
| SGBM | 1.96 | 3.06 | 1.86 | 0.38 | 0.38 | 0.41 | 69.9 | 56.8 | 66.7 |
| Algorithm Ablation | | | | | | | | | |
| T-S | 1.30 | 2.54 | 1.39 | 0.50 | 0.58 | 0.57 | 77.3 | 64.9 | 76.7 |
| I-S | 1.71 | 3.59 | 1.99 | 0.67 | 0.99 | 0.77 | 74.2 | 60.7 | 72.5 |
| IoU-NS | 1.43 | 2.29 | 1.42 | 0.52 | 0.47 | 0.53 | 67.8 | 59.0 | 68.3 |
| T-NS | 1.85 | 2.78 | 1.84 | 0.76 | 0.66 | 0.78 | 64.2 | 55.6 | 64.0 |
| I-NS | 2.21 | 3.20 | 2.12 | 0.80 | 0.80 | 0.78 | 61.6 | 53.5 | 62.7 |
| TSES w/ outliers | 1.87 | 2.83 | 1.73 | 1.28 | 1.18 | 1.15 | 74.3 | 64.3 | 75.2 |
| Velocity Noise Ablation | | | | | | | | | |
| 0% | **0.89** | 1.98 | **0.88** | **0.36** | **0.44** | **0.36** | **82.3** | 70.1 | 82.3 |
| 5% | 0.90 | **1.97** | **0.88** | **0.36** | 0.45 | **0.36** | 82.0 | **70.5** | **82.4** |
| 10% | 0.91 | 1.98 | **0.88** | 0.37 | 0.45 | **0.36** | 81.6 | 70.1 | 82.3 |
| 20% | 0.96 | 2.04 | 0.92 | 0.38 | 0.46 | 0.38 | 80.4 | 68.6 | 81.3 |
| 50% | 1.21 | 2.44 | 1.23 | 0.47 | 0.58 | 0.51 | 74.5 | 61.5 | 74.5 |
| 100% | 1.97 | 3.47 | 2.17 | 0.83 | 0.92 | 1.03 | 61.8 | 48.5 | 59.1 |
| Window Size Ablation | | | | | | | | | |
| 8 pix. | 2.40 | 3.83 | 2.52 | 0.78 | 0.87 | 0.86 | 65.4 | 55.3 | 63.8 |
| 16 pix. | 1.10 | 2.29 | 1.13 | 0.43 | 0.51 | 0.45 | 80.3 | 69.2 | 79.8 |
| 24 pix. | 0.89 | 1.98 | 0.88 | 0.36 | 0.44 | 0.36 | **82.3** | **70.1** | **82.3** |
| 32 pix. | **0.86** | **1.97** | **0.84** | **0.34** | **0.43** | 0.34 | 81.2 | 66.7 | 81.4 |
| 40 pix. | 0.89 | 2.05 | 0.91 | **0.34** | 0.44 | **0.33** | 78.6 | 61.9 | 77.9 |

Table 1: Quantitative results from testing on the indoor_flying (IF) sequences of TSES (our method) and CopNet, along with ablation studies. Prefixes for the algorithm ablation are: IoU - Intersection over Union cost (8), I - Intersection cost (7), T - Time cost (10). Suffixes are with (S) and without (NS) time synchronization (3). Velocity noise was added to the linear and angular velocities separately, as zero mean Gaussian noise with variance equal to a percentage of the norm of each velocity.

The driving sequences were not included as the majority of the points in those sequences were beyond the depth resolved by a disparity of 1, and so a sub-pixel disparity estimator would be needed to achieve accurate results. In addition, we do not include results from indoor_flying4, as the majority of events are closer than the maximum disparity of 31, and are also generated by the low-texture floor, on which we could not generate reasonable results with any of the methods.

While our method generates disparity values whenever there are any events inside the spatial window, we report our results based on disparities on pixels where events appeared, in order to provide a fair comparison with other works.

We used the camera velocities provided in the dataset from [21], which were generated by linear interpolation of the lidar odometry poses provided from MVSEC, and are provided in addition to ground truth optical flow for the sequences in the dataset.

## 5.2 Comparisons

For comparison, we have implemented the CopNet method by Piatkowska et al. [12], and we include their results on the same dataset, using their provided parameters. For these experiments, we have used an $\alpha$ value of 1 (the scaling term in the matching cost, equation (3) in their paper), as the original paper stated a value of 0, which would result in a constant cost. In addition, we compare against block matching and semi-global block matching methods from OpenCV[2], applied to the grayscale frames from the DAVIS camera. Note that the grayscale frames are not time synchronized, and the time offset between the left and right frames is 4ms, 14ms and 14ms for indoor_flying 1, 2 and 3, respectively. However, we were still able to achieve reasonable performance. The quantitative results of these comparisons can be found in Tab. 1.

In addition, we attempted an implementation of the belief propagation based work by Xie et al. [17], but were unable to obtain reasonable results over this dataset, which is significantly more complex than those evaluated in the original work, consisting of a few objects moving in the scene. We believe that this is because their matching cost ($D(d_p)$) attempts to match individual events, without using the spatial neighborhood around the event. In our experiments, this matching cost failed to identify the correct disparity over the majority of the image, which we believe led the belief propagation to output incorrect results.

## 5.3 Ablation Studies

In addition to the comparisons, we performed a number of ablation studies over the parameters of the algorithm. All results can be found in Tab. 1.

**Algorithm Ablation** To test the effect of the time synchronized event disparity volumes, we performed additional experiments where the raw event positions were passed directly into (4) and (5) (i.e. by setting $(x'(d)_i, y'(d)_i) = (x_i, y_i)$). Experiments with and without time synchronization are denoted with the suffix -S and -NS, respectively.

To test the IoU cost, we tested with only the intersection cost (prefix I), as well as using the cost function from [12] (prefix T), which is defined as:

$$C_T(x,y,d) = \sum_{x^*,y^* \in W(x,y,d)} \frac{1}{(\alpha \cdot |I_L^t(x^*,y^*,d) - I_R^t(x^*,y^*,d)| + 1) \cdot C_U(x^*,y^*,d)} \tag{10}$$

[2] `https://docs.opencv.org/3.4/d2/d6e/classcv_1_1StereoMatcher.html`

where $\alpha$ is set to 1, the event images $I^t$ now represent the timestamp of the last event to arrive at each pixel and disparity:

$$I_L^t(x, y, d) = \max_{t_i} \{t_i | (x'_i(d), y'_i(d)) = (x, y)\} \tag{11}$$

$$I_R^t(x, y, d) = \max_{t_i} \{t_i | (x'_i(d) + d, y'_i(d)) = (x, y)\} \tag{12}$$

and we use our union cost in place of the number of events in the left window. We also tested with the inverse of the union cost, but this did not produce any reasonable results.

We also provide results of our full method, without the outlier rejection step.

**Velocity Noise Ablation** In practice, it is difficult to estimate the cameras' velocity with the same accuracy as the ground truth. To test the effect of noise on the velocity estimate, we perform additional experiments where we add zero mean Gaussian noise to the linear and angular velocities. The variance of the noise is set to a given percentage of the norm of the linear and angular velocities, separately.

**Window Size Ablation** We also tested the effects of the window size on the performance of our method over a range of window sizes.

## 5.4 Results and Discussion

**Comparisons** We present some qualitative results in Fig. 3, comparing our method to CopNet, block matching and ground truth. While both sets of results look visually reasonable, we can see that our method suffers less from foreground fattening [6] (e.g. the chair in the fourth row). Our method does, however, tend to produce erroneous results on the edges of images in the dense disparity image, but these correspond to pixels without any events, and are thresholded away in the sparse disparity image.

In addition, we provide quantitative results in Tab. 1, where we can see that our full method outperforms CopNet across almost every measure, as well as the other methods in the ablation study. In particular, while the disparity errors are similar, CopNet performs significantly worse in depth overall. Upon examining the results. We found that this error from CopNet was largely due to the fact that the method had over-smoothed the disparity output. This was mostly due to the window size used, which is relatively large (39x39). This oversmoothing tends to pull far away points closer (overestimates disparities), which leads to higher depth errors, as they are higher at lower disparity levels. However, we observed that reducing the window sizes resulted in a further reduction in the overall matching accuracy due to increased ambiguity in the matching, as noted by the authors in the original paper [12], so there was no immediate solution for this problem.

The block matching method performed better in terms of mean errors across all three sequences, although the mean disparity errors for sequences 1 and 3 are both less than 1 pixel, which is within the range of the discretization error. In addition, our method has a higher percentage of points with disparity error <1 across all sequences.

We were unable to achieve comparable performance from semi-global block matching, which tended to over smooth incorrect regions in the image.

**Algorithm Ablations** From the ablation study, we can see that removing each component of the method tends to result in a corresponding decrease in accuracy, with the time synchronization always resulting in better results. In addition, the addition of the union cost to the overall cost provides a significant improvement in accuracy over the intersection cost, which is a pure similarity measure.

Furthermore, we can see that our IoU cost outperforms the timestamp based cost in both situations, suggesting that it may be a better alternative for more complex methods, even without the time synchronization. When the proposed time synchronization is applied at the correct disparity, older timestamps are mapped to later timestamps from the same point in the image. As the time cost operates on an image that only keeps the latest timestamp, this results in images with timestamps that are very similar (all later events), which do not provide much discriminative power. Future work could consider all of the events that map to a pixel, but this requires a new cost function.

Finally, the results without outlier rejection have significantly higher mean disparity errors, suggesting that a large number of outliers were rejected by our method, while from the % disparity error $< 1$ results, we can see that only <10% of the points were rejected.

**Velocity Noise Ablation** The velocity noise ablation results show stable errors up to noise with variance up to around 20% of the velocity norm. We believe that a conventional state estimation pipeline for event cameras should be able to reliably estimate the camera velocity within these error bounds.

**Window Size Ablation** We found that window sizes between 24 and 40 pixels achieved the best results. However, larger window sizes increase the amount of foreground fattening, as well as the runtime of the algorithm. Therefore, we recommend a window size of 24 pixels for these test cases. In terms of run time, the algorithm took 33ms, 40ms and 50ms to run for window sizes of 16, 24 and 32 pixels, respectively. Similarly, the runtime was 25ms and 60ms for disparity ranges of 16 and 48, with a window size of 24 pixels.

## 6 Conclusions

We have proposed a novel method for stereo event disparity matching which uses the motion of the camera to synchronize the event streams in time. We show that our method, consisting of a simple temporal interpolation of the events, along with a lightweight matching cost, is able to outperform state of the art methods which perform expensive regularizations on top of the disparity map. In addition, as our disparity results are at a single time, analogous to an image frame, they can be directly passed into any frame based architecture such as a state estimator, as compared to an asynchronous disparity stream. We envision that this method will be coupled with a method for estimating camera velocity, such as a visual odometry algorithm, for real time performance.

## Acknowledgements

Thanks to Tobi Delbruck and the team at iniLabs for providing and supporting the DAVIS-346b cameras. We also gratefully appreciate support through the following grants: NSF-IIS-1703319, NSF-IIP-1439681 (I/UCRC), ARL RCTA W911NF-10-2-0016, and the DARPA FLA program.

## References

1. Besse, F., Rother, C., Fitzgibbon, A., Kautz, J.: PMBP: Patchmatch belief propagation for correspondence field estimation. International Journal of Computer Vision **110**(1), 2–13 (2014)
2. Camuñas-Mesa, L.A., Serrano-Gotarredona, T., Ieng, S.H., Benosman, R.B., Linares-Barranco, B.: On the use of orientation filters for 3D reconstruction in event-driven stereo vision. Frontiers in Neuroscience **8** (2014)
3. Camuñas-Mesa, L.A., Serrano-Gotarredona, T., Ieng, S.H., Benosman, R., Linares-Barranco, B.: Event-driven stereo visual tracking algorithm to solve object occlusion. IEEE Transactions on Neural Networks and Learning Systems (2017)
4. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. International journal of computer vision **70**(1), 41–54 (2006)
5. Gallego, G., Scaramuzza, D.: Accurate angular velocity estimation with an event camera. IEEE Robotics and Automation Letters **2**(2), 632–639 (2017)
6. Geiger, A., Roser, M., Urtasun, R.: Efficient large-scale stereo matching. In: Asian conference on computer vision. pp. 25–38. Springer (2010)
7. Kogler, J., Humenberger, M., Sulzbachner, C.: Event-based stereo matching approaches for frameless address event stereo data. Advances in Visual Computing pp. 674–685 (2011)
8. Marr, D., Poggio, T., et al.: Cooperative computation of stereo disparity. From the Retina to the Neocortex pp. 239–243 (1976)
9. Mitrokhin, A., Fermuller, C., Parameshwara, C., Aloimonos, Y.: Event-based Moving Object Detection and Tracking. arXiv preprint arXiv:1803.04523 (2018)
10. Mueggler, E., Forster, C., Baumli, N., Gallego, G., Scaramuzza, D.: Lifetime estimation of events from Dynamic Vision Sensors. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on. pp. 4874–4881. IEEE (2015)
11. Piatkowska, E., Belbachir, A.N., Gelautz, M.: Cooperative and asynchronous stereo vision for dynamic vision sensors. Measurement Science and Technology **25**(5), 055108 (2014)
12. Piatkowska, E., Kogler, J., Belbachir, N., Gelautz, M.: Improved cooperative stereo matching for dynamic vision sensors with ground truth evaluation. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on. pp. 370–377. IEEE (2017)

13. Rebecq, H., Gallego, G., Mueggler, E., Scaramuzza, D.: Emvs: Event-based multi-view stereo3d reconstruction with an event camera in real-time. International Journal of Computer Vision pp. 1–21 (2017)
14. Rebecq, H., Horstschaefer, T., Scaramuzza, D.: Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In: British Machine Vis. Conf.(BMVC). vol. 3 (2017)
15. Rogister, P., Benosman, R., Ieng, S.H., Lichtsteiner, P., Delbruck, T.: Asynchronous event-based binocular stereo matching. IEEE Transactions on Neural Networks and Learning Systems **23**(2), 347–353 (2012)
16. Schraml, S., Nabil Belbachir, A., Bischof, H.: Event-driven stereo matching for real-time 3D panoramic vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 466–474 (2015)
17. Xie, Z., Chen, S., Orchard, G.: Event-based stereo depth estimation using belief propagation. Frontiers in Neuroscience **11**, 535 (2017)
18. Xie, Z., Zhang, J., Wang, P.: Event-based stereo matching using semiglobal matching. International Journal of Advanced Robotic Systems **15**(1), 1729881417752759 (2018)
19. Zhu, A.Z., Atanasov, N., Daniilidis, K.: Event-based feature tracking with probabilistic data association. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on. pp. 4465–4470. IEEE (2017)
20. Zhu, A.Z., Thakur, D., Ozaslan, T., Pfrommer, B., Kumar, V., Daniilidis, K.: The multi vehicle stereo event camera dataset: An event camera dataset for 3D perception. IEEE Robotics and Automation Letters (2018)
21. Zhu, A.Z., Yuan, L., Chaney, K., Daniilidis, K.: EV-Flownet: Self-supervised optical flow estimation for event-based cameras. arXiv preprint arXiv:1802.06898 (2018)
22. Zou, D., Guo, P., Wang, Q., Wang, X., Shao, G., Shi, F., Li, J., Park, P.K.: Context-aware event-driven stereo matching. In: Image Processing (ICIP), 2016 IEEE International Conference on. pp. 1076–1080. IEEE (2016)
23. Zou, D., Shi, F., Liu, W., Li, J., Wang, Q., Park, P.K., Shi, C.W., Roh, Y.J., Ryu, H.E.: Robust dense depth map estimation from sparse DVS stereos. In: British Machine Vis. Conf.(BMVC). vol. 3 (2017)