Tangentially Elongated Gaussian Belief Propagation for Event-based Incremental Optical Flow Estimation

Anonymous CVPR submission

Paper ID 4712

Abstract

Optical flow estimation is a fundamental functionality in computer vision. An event-based camera, which asynchronously detects sparse intensity changes, is an ideal device for realizing low-latency estimation of the optical flow owing to its low-latency sensing mechanism. An existing method using local plane fitting of events could utilize the sparsity to realize incremental updates for low-latency estimation; however, its output is merely a normal component of the full optical flow. An alternative approach using a frame-based deep neural network could estimate the full flow; however, its intensive non-incremental dense operation prohibits the low-latency estimation. We propose tangentially elongated Gaussian (TEG) belief propagation (BP) that realizes incremental full-flow estimation. We model the probability of full flow as the joint distribution of TEGs from the normal flow measurements, such that the marginal of this distribution with correct prior equals the full flow. We formulate the marginalization using a message-passing based on the BP to realize efficient incremental updates using sparse measurements. In addition to the theoretical justification, we evaluate the effectiveness of the TEGBP in real-world datasets; it outperforms SOTA incremental quasi-full flow method by a large margin. The code will be open-sourced upon acceptance.

1. Introduction

Optical flow estimation, which computes the correspondence of pixels in different time measurements, is a fundamental building block of computer vision. One needs to estimate the flow at low latency in many practical applications, such as autonomous driving cars, unmanned aerial vehicles, and factory automation robots. Most of the existing optical flow algorithm utilizes dense video frames; it computes the flow by searching the similar intensity pattern [15, 29]. Recently, methods using deep neural network (DNN) [29] demonstrate impressive accuracy at the cost of higher-computational cost. Either model-based or DNN-based, the frame-based algorithm needs to compute the entire pixel for every frame, even when there are subtle changes or no changes at all. This dense operation makes it difficult to realize low-latency estimation, especially on the resource-constrained edge device.

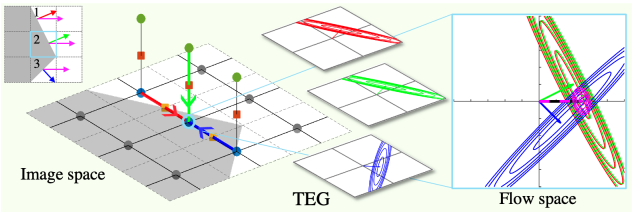

Figure 1. **Overview of the proposed TEGBP**. We model a belief about the full flow from a normal flow measurement using TEG (RGB ellipse). The mean of the marginal distribution of each TEG with an appropriate prior equals the full flow (magenta arrow). The marginal (magenta ellipse) is computed incrementally using local message passing (RGB arrows) based on BP.

The event-based camera is a bio-inspired vision sensor, which asynchronously detects intensity change on each pixel. Thanks to the novel sensing mechanism, the camera equips favorable characteristics for optical flow estimation, such as high dynamic range (HDR), blur-free measurement, and, most importantly, sparse low-latency data acquisition. Many researchers have explored the way to utilize sparsity to realize efficient low-latency estimation; one extends the well-known Lucas-Kanade algorithm [7], and the other exploits the local planer shape of the spatiotemporal event streams [6]. These methods could utilize the sparsity for efficient *incremental* processing; however, the flow computed in this way (e.g., by plane fitting) is the *normal flow* which is different from the *full flow* we want to obtain. The normal flow is the component of the optical flow perpendicular to the edge (i.e., parallel to the intensity gradient). Some work tried to recover the full flow from the normal flow [2]; however, it does not precisely equal the *full* optical flow (refer to

Sec. 2). There exist methods that could estimate full flow, such as a variational method [5], a multi-scale extension of contrast maximization [25], or a frame-based DNN [14]. However, they need to apply non-incremental dense operation for all the pixels of the event frame (dense representation constructed from sparse events) for every frame, which prohibits the low-latency estimation on the edge device.

Our research goal is to realize an *incremental full* flow algorithm from sparse normal flow measurements. To this end, we propose *Tangentially Elongated Gaussian (TEG) Belief Propagation (BP)*. We compute the full flow using the normal flow measurements, which can be observed directly from an event camera or computed cheaply using an existing algorithm[1]. Notice that given a single measurement of normal flow, there are infinite possibilities for the full flow along the tangential direction of the normal flow. We model this uncertainty using the TEG, Gaussian distribution, which has a large variance along the tangential direction of the normal flow. The probability density of full flow on each pixel is given as the marginals of the joint distribution of TEG data factor and some prior factor on a sparse graph (Fig. 1, Sec. 3.3.2). We leverage the sparse graph to formulate the *incremental full* flow estimation algorithm using message-passing based on BP [9]. We evaluate the effectiveness of the TEGBP on real-world data captured from aerial drones and automobiles. TEGBP outperforms SOTA *incremental* method [2] by a large margin.

2. Related work

This section reviews the related literature on optical flow estimation using sparse event signals.

2.1. Incremental Algorithm for Normal Flow

Lucas-Kanade based method The traditional frame-based Lucas-Kanade method finds the flow by aligning the small patch of intensity between consecutive frames using its spatial gradient and temporal change. The event-based counterpart [7] uses the integration of events to approximate them for each incoming event. This approach realized highly efficient estimation by utilizing the sparsity of input events. However, it can only estimate the normal flow, which is the component of the full optical flow perpendicular to the edge (i.e., parallel to the intensity gradient) [11].

Plane fitting based method When an edge undergoes liner motion, triggered events generate a time surface, which can be locally approximated by a plane in a spatiotemporal space. The local plane fitting method [3, 6, 17, 19] computes the optical flow by fitting the plane to the sparse event points. However, the flow obtained by plane fitting is also the normal component of the full flow.

Recently, a method for computing the quasi-full flow has been proposed [2]; it estimates the full flow by considering the average normal flow in multiple resolutions. It incrementally updates the average and selects the flow with the largest average norm from the multiple resolutions. It expects the flow to have the largest norm to correspond to the full flow. The average operation induces bias on the estimated flow; therefore, it equals the full flow on the very limited scene where the true motion in the window is orthogonal to the edge direction (refer to Sec.5 in their paper). We'll also experimentally compare this point in Sec. 4.3.

In summary, current sparse incremental algorithms are limited to normal flow.

2.2. Dense Algorithm for Full Flow

Variational method A variational optimization-based approach realizes the full flow estimation by incorporating intensity frame obtained either by the event/frame hybrid camera [22] or by simultaneously estimating image intensity [5]. It could estimate full flow; however, it requires additional intensity observation or intensive dense optimization to recover the intensity frame.

Contrast maximization method Contrast maximization (CMax) [12, 27] computes the underlying motion (optical flow) of events stream by finding their alignment. It computes the alignment by maximizing the focus score (e.g., variance) of an image generated by warping the events (IWE) using the optical flow parameter. The original algorithm can not be used to estimate pixel-wise flow because it tends to converge to the degenerated solution (e.g., warps all events to a single point) [24, 35]. Recently, a hierarchical CMax approach avoiding the degenerated solution for pixel-wise estimation has been proposed [25]. Although this approach works well even in the practical scenario, it is not incremental; it must recompute the IWE for each sliding time window by using all events in the patch for every hierarchical scale (this involves intensive iteration by itself), making it difficult to operate in real-time.

DNN based method Frame-based DNN is often utilized to compute the full flow [14, 31, 32, 34]. Recently, a frame-based high-accuracy model called RAFT [29] has been imported into the event-based vision literature showing the SOTA performance for the dense flow estimation [14]. These methods can not consume sparse events directly; the sparse input must be converted into dense frame representation. The dense convolution is computationally expensive because it needs to operate for all pixels, even when there are subtle changes or no changes at all, making it difficult to operate at a higher rate on the edge device.

[1]Here in after, we consider the normal flow as *measurement*. We focus on an algorithm to process the sparse *measurement* of normal flow to estimate the full flow.

In summary, some dense algorithms could estimate full flow; however, it compromises the event's sparse asynchronous nature, and their non-incremental dense operation prohibits them from realizing low-latency processing.

2.3. Sparse Asynchronous Computing Architecture

Events are asynchronously triggered only from limited pixels which detect intensity changes. Several techniques have been developed for efficient processing of the sparse signals on CPUs by taking advantage of the highly optimized cache mechanism [16, 28]. Besides, sparse data-flow architectures [8, 9], which differ from prevalent Neuman-type computing architectures, are gaining attention for sparse signal processing. For example, bundle adjustment has been solved on the novel architecture using an algorithm based on Gaussian BP [21]. Our goal is to establish the *incremental full flow* estimation algorithm by taking advantage of the sparsity of events, which is expected to run efficiently on CPUs or emerging data-flow processors.

3. Proposed method

3.1. Preliminary

3.1.1 Normal flow

The normal flow $\mathbf{v}_i^\perp \in \mathbb{R}^2$ on pixel i is a normal component of full flow $\mathbf{v}_i \in \mathbb{R}^2$, which we want to estimate. One can not decide the full flow uniquely from the normal flow, and there are infinitely many possibilities along the line perpendicular to the normal flow (Fig. 3 left). There are several options for obtaining the normal flow; some event-based cameras, such as CeleX-V [26] directly observe it, or it can be cheaply computed from the sparse events (Sec. 2.1).

The moving edge in 3D space triggers events; we can compute the normal flow by finding the local plane on the triggered events. In this study, we adopt a naive least-squares fitting of the plane using events in a small spatiotemporal window. An erroneous normal flow may significantly deteriorate the full flow accuracy, especially on highly textured regions such as leaves on trees. One can improve the results by utilizing an advanced noise removal algorithm [4] or by adopting more sophisticated local flow estimation methods, such as small neural networks.

3.1.2 Belief propagation

Factor graphs geometrically represent the structure of probabilistic problems. A factor graph $G = (V, F, E)$ is composed of a set of variable nodes $V = \{\mathbf{v}_i\}_{i=1:N_v}$, a set of factor nodes $F = \{f_s\}_{s=1:N_f}$ and a set of edges E. Each factor node f_s represents a probabilistic constraint $f_s(V_s)$ between a subset of variables $V_s \subset V$. The factorization of the variables is explicitly represented in the graph

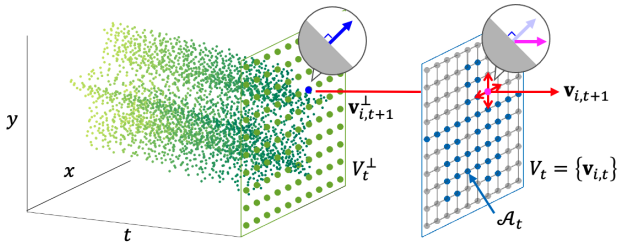

Figure 2. **Processing pipeline**. Observation set at t is represented as $V_t^\perp$ where the normal flow is sparsely assigned. The TEGBP incrementally update its estimation about full flow V_{t+1} using previous estimation V_t and new observation $\mathbf{v}_{i,t+1}^\perp$.

by connecting factor nodes with the variable nodes they depend on. Probabilistically, these factors are the independent terms that make up the joint distribution:

$$p(V) = \prod_{s=1}^{N_f} f_s(V_s). \tag{1}$$

Belief propagation (BP) is a generic distributed algorithm for computing the marginal distribution of a set of variables from their joint distribution. The marginal for a single variable $\mathbf{v}_i$ is computed by the integral over all the other variables:

$$p(\mathbf{v}_i) = \int p(V) d\mathbf{v}_1 \cdots d\mathbf{v}_{i-1} d\mathbf{v}_{i+1} \cdots d\mathbf{v}_{N_v}. \tag{2}$$

BP computed the marginals asynchronously using *message-passing* between connected nodes on the graph. It is substantially more efficient by leveraging the topology of the sparse graph than naive integration (which involves giant matrix inversion). Refer to the seminal work of [9] for the derivation of BP, condition for convergence guarantee, and discussion about the stability.

3.2. Problem formulation

Given measurements of normal flow $V_t^\perp$ on the the pixel grid where each normal flow is assigned to the corresponding pixels (within time interval $[t - \tau, t]$), we want to estimate the posterior distribution of the corresponding full flow $p(V_t \mid V_t^\perp)$. Once the posterior distribution of full flow is obtained, the actual full flow output is given by the maximum a posteriori (MAP) estimate as:

$$\hat{V}_t = \arg\max_{V_t} p(V_t \mid V_t^\perp). \tag{3}$$

Our goal is to formulate an *incremental full* flow estimation algorithm using sparse observations of local normal flows (Fig. 2). Solving the $\arg\max$ of the joint distribution of V_t for every incoming normal flow measurement is

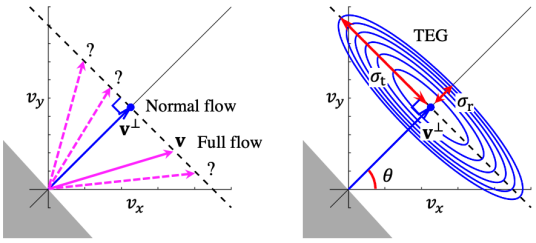

Figure 3. **The relation of normal flow and full flow, and TEG**. There are infinitely many possible candidates for full flows given a normal flow (left). The TEG models this uncertainty using Gaussian having large variance along the tangential direction of the normal flow (right).

infeasible both in terms of memory and computational perspective; therefore, we want to formulate f as an incremental sparse update function, i.e., new observation interacts sparsely with the previous results $p(V_t \mid V_t^\perp)$ as follows:

$$f : [p(V_t \mid V_t^\perp), p(\mathbf{v}_{i,t+1}^\perp)] \mapsto p(V_{t+1} \mid V_{t+1}^\perp). \quad (4)$$

In the following section, we'll derive TEGBP, an algorithm based on BP for incremental full flow estimation.

3.3. Tangentially Elongated Gaussian BP

3.3.1 Tangentially Elongated Gaussian

The true full flow $\mathbf{v}_i$ exists somewhere along the line perpendicular to the normal flow $\mathbf{v}_i^\perp$ (Fig. 3 left). In other words, we are uncertain about the full flow from the single normal flow measurement. We model this uncertainty using tangentially elongated Gaussian (TEG), A Gaussian distribution having infinitely large variance along the tangential direction of the observed normal flow (Fig. 3 right). We defined the precision matrix of TEG as follows:

$$\Lambda^{\text{teg}} = \mathtt{R}(\theta)\Lambda^{\text{m}}\mathtt{R}(\theta)^\top, \quad (5)$$

where $\mathtt{R}(\theta)$ is 2D rotation matrix parameterized by the rotation angle θ of the normal flow where $\theta = \arctan(v_y^\perp / v_x^\perp)$, and $\Lambda^{\text{m}} = \text{diag}(1/\sigma_{\text{r}}^2, 1/\sigma_{\text{t}}^2)$.

In principle, $\sigma_{\text{t}} = \infty$ and $\sigma_{\text{r}} = 0$ corresponds to the constraint of the full flow from a single noiseless normal flow measurement. In practice, we use finite σ_{t} considering the maximum deviation from the normal flow and nonzero σ_{r} considering the noise in the normal flow. The appropriate value of σ_{t} and σ_{r} may depend on the given data set; however, we used the same value for all experiments.

3.3.2 Marginalization of TE-Gaussian with Valid Prior

As we saw earlier, the full flow estimation from the normal flow is inherently an ill-pose without some prior knowledge (when $\sigma_{\text{t}} = \infty$ and $\sigma_{\text{r}} = 0$). Without a prior, there are infinitely many possibilities along the line perpendicular to the normal flow; any point on the line has the same likelihood. To obtain a correct solution, we need to use a correct prior. Our algorithms could incorporate any prior, e.g., simple smoothness prior (e.g., total variation), an application-specific prior such as flow converges to the vanishing-point in an automotive scenario, or learned prior using neural networks. In this study, we adopt smoothness prior for simplicity. The important observation here is that the mean of the marginal distribution of TEGs with the correct prior equals the correct full flow.

Example. *To see this, let's consider the case of Fig. 1. We assume the corner having an edge angle of $\theta_1 + \frac{\pi}{2}$, and $\theta_3 + \frac{\pi}{2}$ moves linearly with uniform velocity (all pixels have the same velocity) toward the direction of θ_0, i.e., $\hat{\mathbf{v}} = [\cos\theta_0, \sin\theta_0]^\top$. The correct prior in this situation is uniform (infinitely strong smoothness prior, $\sigma_p = 0.0$). In this scenario, we'll observed $\mathbf{v}_1^\perp = \mathbf{v}_2^\perp = [\cos(\theta_1 - \theta_0)\cos\theta_1, \cos(\theta_1 - \theta_0)\sin\theta_1]^\top$, $\mathbf{v}_3^\perp = [\cos(\theta_3 - \theta_0)\cos\theta_3, \cos(\theta_3 - \theta_0)\sin\theta_3]^\top$.*

The joint distribution of full flow $V := \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ from normal flow measurements and the prior is given as follows:

$$\begin{aligned} p(V) =& \mathcal{N}(\mathbf{v}_1; \mathbf{v}_1^\perp, \Sigma_{\theta_1})\mathcal{N}(\mathbf{v}_2; \mathbf{v}_2^\perp, \Sigma_{\theta_2})\mathcal{N}(\mathbf{v}_3; \mathbf{v}_3^\perp, \Sigma_{\theta_3}) \\ & \mathcal{N}([\mathbf{v}_1; \mathbf{v}_2]; \sigma_p^2\mathtt{I})\mathcal{N}([\mathbf{v}_2; \mathbf{v}_3]; \sigma_p^2\mathtt{I}). \end{aligned} \quad (6)$$

Marginalizing for $\mathbf{v}_2$, $p(\mathbf{v}_2) = \int p(V) d\mathbf{v}_1 d\mathbf{v}_3$ and putting $\sigma_t = \infty$ and $\sigma_r = 0$, we'll get $p(\mathbf{v}_2) = \mathcal{N}\left(\mathbf{v}_2; [\cos\theta_0, \sin\theta_0]^\top, \text{diag}(0, 0)\right)$; the MAP of this marginal equals to the true flow $\hat{\mathbf{v}}$. □

One can also visually verify this result by considering the intersection of the lines of the major axis of the TEG. Notice that the marginal is equal to the full flow even when an average of the normal flow, such as employed in [2], differs from the full flow. In the above case, the average normal flow ($\mathbf{v} = [(2\cos(\theta_1 - \theta_0)\cos\theta_1 + \cos(\theta_3 - \theta_0)\cos\theta_3)/3, (2\cos(\theta_1 - \theta_0)\sin\theta_1 + \cos(\theta_3 - \theta_0)\sin\theta_3)/3]^\top$) is different from the true flow ($[\cos\theta_0, \sin\theta_0]^\top$). This is a steak difference from the existing incremental sparse method using an average of normal flow and is a key to realizing a significant accuracy boost over the technique.

3.3.3 Factor Graph of Full Flow

Now we construct the sparse factor graph for full flow using the TEG. The joint of Eq. (3) can be factorized using Bayes theorem as follows:

$$\hat{V}_t = \arg\max_{V_t} p(V_t^\perp \mid V_t) p(V_t). \quad (7)$$

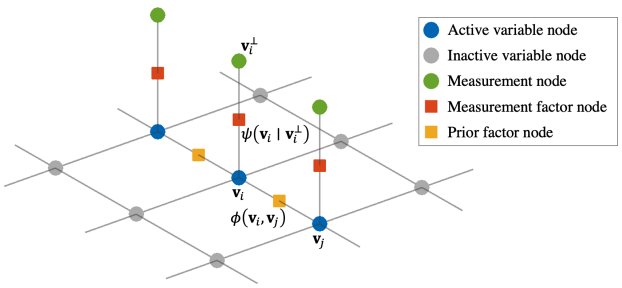

Figure 4. **2D factor graph of TEGBP**. The topology changes dynamically depending on the active set $\mathcal{A}_t$. The measurement node and the active variable node on the same pixel are connected by measurement factor node, and each neighbor active variable node is connected via prior factor node.

Assuming independence of observations and dependence between neighboring-pixel variables, Eq. (7) is further factorized to the measurement factor ψ and the prior factor ϕ defined on a sparse graph on the image grid (Fig. 4) as:

$$p(V) = \prod_{n \in \mathcal{A}_t} \psi(\mathbf{v}_n) \prod_{(i,j) \in \mathcal{E}_t} \phi(\mathbf{v}_i, \mathbf{v}_j), \tag{8}$$

where $\mathcal{A}_t$ is the set of *active* nodes corresponding normal flow measurements $V_t^\perp$, and $\mathcal{E}_t$ is the set of edges between active neighbor nodes. The measurements factor ψ is defines as TEG (refer to Sec. 3.3.1):

$$\psi(\mathbf{v}_n) = \mathcal{N}^{-1}\left(\cdot; \boldsymbol{\eta}_n, \Lambda_n^{\text{teg}}\right), \tag{9}$$

where $\boldsymbol{\eta}_n = \Lambda_n^{\text{teg}} \mathbf{v}_n^\perp$. In this study, we assume the flow is smooth; therefore, use the following prior factor ϕ:

$$\phi\left(\begin{bmatrix}\mathbf{v}_i \\ \mathbf{v}_j\end{bmatrix}\right) = \mathcal{N}^{-1}\left(\cdot; \mathbf{0}, \Lambda^{\text{p}}\right), \tag{10}$$

where $\Lambda^{\text{p}} = \mathtt{J}_{\text{p}}^\top \operatorname{diag}(1/\sigma_{\text{p}}^2, 1/\sigma_{\text{p}}^2) \mathtt{J}_{\text{p}}$ and the jacobian $\mathtt{J}_{\text{p}} = \left[\begin{smallmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{smallmatrix}\right]$ for computing the difference between the neighbor nodes.

To model the outlier that can not be expressed as a Gaussian (quadratic cost), we employed the Huber cost both for measurement and prior factors. We adopted the technique of [1, 9] to maintain the Gaussian form.

3.3.4 TEGBP

Equipped with the sparse probabilistic graph, we now formalize the incremental message-passing algorithm base on BP (Sec. 3.1.2) for the marginalization of the graph to estimate the full flow (Fig. 5). As the graph of active nodes is loopy, GBP needs to store a belief at each variable node for iterative belief update.

Message-passing is asynchronously triggered upon a space observation of normal flow. The measurement factor sends a message to the connected variable node and updates its belief with messages that have been sent from the active node as follows:

$$\boldsymbol{\eta}_i^{\text{b}} = \sum_{j \in \mathcal{C}_i \cap \mathcal{A}_t} \boldsymbol{\eta}_{j \to i}, \quad \Lambda_i^{\text{b}} = \sum_{j \in \mathcal{C}_i \cap \mathcal{A}_t} \Lambda_{j \to i} \tag{11}$$

where $b(\mathbf{v}_i) = \mathcal{N}^{-1}(\cdot; \boldsymbol{\eta}_i^{\text{b}}, \Lambda_i^{\text{b}})$ is the belief on node i, $\mu_{j \to i} = \mathcal{N}^{-1}(\cdot; \boldsymbol{\eta}_{j \to i}, \Lambda_{j \to i})$ is the message from node j to node i, and $\mathcal{C}_i$ is the 4-neighbors connected node j. After that, the variable node sends the following message about its updated beliefs to its neighbors:

$$\boldsymbol{\eta}_{j \to i} = -\Lambda_{01}^{\text{p}} (\Lambda_{11}^{\text{p}} + \Lambda_j^{\text{b}} - \Lambda_{j \to i})^{-1} (\boldsymbol{\eta}_j^{\text{b}} - \boldsymbol{\eta}_{j \to i}) \tag{12}$$

$$\Lambda_{j \to i} = \Lambda_{00}^{\text{p}} - \Lambda_{01}^{\text{p}} (\Lambda_{11}^{\text{p}} + \Lambda_j^{\text{b}} - \Lambda_{j \to i})^{-1} \Lambda_{10}^{\text{p}}, \tag{13}$$

where, $\Lambda_{\alpha\beta}^{\text{p}}$ is a 2×2 sub-matrix of Λ^{p} indexed by α and β. For a single normal flow measurement, the belief is propagated for K hops, which are repeated for N_{itr} times (Fig. 5 correspond to $K = 2$).

Note that our BP algorithm is guaranteed to converge to the correct MAP solution with sufficient iteration because all the belief is realized as a Gaussian (there is no guarantee for a variance because the graph is loopy) [30].

Coarse-to-fine message-passing scheme [10] is adopted to speed up the convergence. The measurement factor of the coarser layer l is computed as the sum of active measurement factor messages in a 2×2 block of finer layer $l - 1$. The BP is executed coarse-to-fine; the coarse layer (message) estimation is copied to the finer layer as the initial guess at the start of the message passing on the finer layer. The short paths in the course graph improve efficiency for capturing long-range interactions and speeds up the convergence.

4. Experiment

We conducted experiments to compare the accuracy of TEGBP with the state-of-the-art incremental flow estimation algorithm called aperture-robust multi-scale flow estimation (ARMS) [2]. Both algorithms are incremental that could asynchronously estimate flow from sparse normal flow measurement. We used the same normal flow measurement for both algorithms for a fair comparison.

4.1. Experimental setting

4.1.1 Normal flow measurement

We utilize the simple plane fitting to compute the normal flow measurements that serve as inputs to TEGBP and ARMS. We first apply noise removal for the raw events

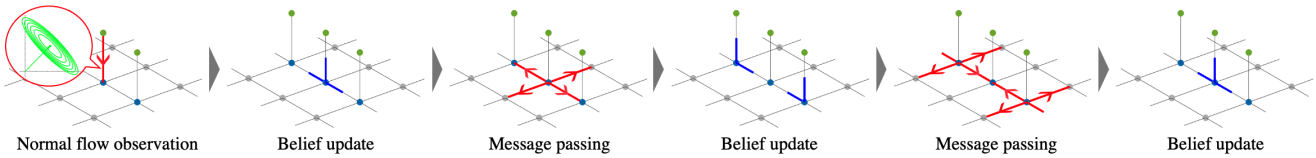

Figure 5. **A schematic showing a single iteration of TEGBP**. (a) When a single normal flow has been measured, the message from the measurement factor is sent to the variable node as TEG, (b) the node updates its belief using the message received from the neighbor active nodes. (c) The nodes broadcast the updated belief to the neighbor, and (d) the neighbor nodes update their beliefs. (a)-(d) is repeated until convergence. To prevent the blowup of the message communication, it is stopped after K hops.

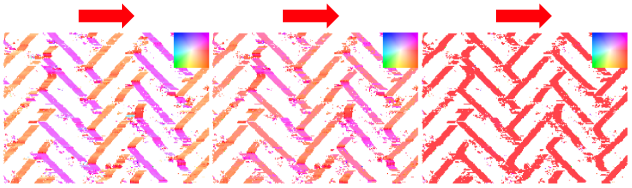

Figure 6. **The result on ESIM-bricks**. The estimated full flow from ARMS and TEGBP are compared with ground-truth flow indicated by the red arrow. They use the same normal flow measurement (left). TEGBP successfully estimates the full flow, while ARMS fails.

using a refractory filter [17] of 40 ms duration. Then using the filtered events, we performed plane fitting for each event within a small spatiotemporal window of the size $r \times r \times \Delta t_{\text{pf}}$. During the plane fitting, we removed the outlier using the criteria proposed in [2]; we applied this process three times. We summarize the normal flow computation parameters in Tab. 1 (top).

4.1.2 Full flow estimation

Both TEGBP and ARMS use the same sparse normal flow measurement, and could be run asynchronously at the rate of normal flow; however, we process N_b observation at once for computational efficiency on MATLAB [18] (interactive numerical computing environment) we used to compare the accuracy. The specific parameters for the (quasi) full flow computation are summarized in Tab. 1 (bottom).

In TEGBP, the message should be propagated to all connected nodes, and the process needs to be repeated until convergence; however, we found a small number of hops (K=2) and single iterations (N_{itr}=1) works well in practice.

4.2. ESIM-bricks

In this experiment, we compare the accuracy on a scene where the motion and edges are not perpendicular to the image edge. We synthesized the event stream by observing the image of bricks from a translating camera[2]. We used ESIM [23] library for this simulation, which could synthesize the realistic noisy event stream from the event camera moving around the predefined 3D environment. We compute the normal flow and use the same measurements for both methods.

Fig. 6 shows the result. We observe the drift in ARMS due to the average operation. TEGBP correctly estimated the full flow from the normal flow measurement directed at a different angle from the full flow.

Table 1. **Basic experimental setup**.

Parameter		ESIM	DVS	MVSEC	DSEC
Window size	r [pix]	5	5	3	5
	Δt_{pf} [ms]	40	40	40	40
Active duration	τ [ms]	50	50	100	100
Prior std	σ_{p} [pix]	0.1	1	0.5	1
Tangential std	σ_{t} [pix]	10	10	10	10
Radial std	σ_{r} [pix]	3	3	3	3
Num. of layers	L	5	5	5	5
Batch size	N_b	100	100	1000	1000

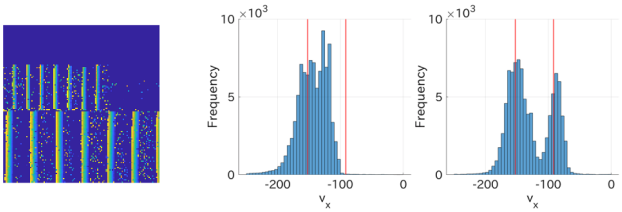

Figure 7. **The results on DVS-stripes**. The histograms of the flow (v_x) of ARMS and TEGBP are compared. The two red lines indicate the GT flow of the two stripes (the upper stripes have a smaller norm). The result from ARMS are shifted from the GT flow, especially on the area of upper stripes (smaller norm) On the contrary, our TEGBP successfully estimates individual flows.

4.3. DVS-stirpes

In this experiment, we compare the accuracy on a scene where some region includes different flows. We used a

[2]We'll publish this data along with the code.

Table 2. **The quantitative results on MVSEC**. The flow is evaluated on the ground truth intervals.

	indoor_flying1		indoor_flying2		indoor_flying3		outdoor_day1	
	AEE ↓	% Out ↓	AEE ↓	% Out ↓	AEE ↓	% Out ↓	AEE ↓	% Out ↓
Norm	2.30	24.5	3.61	42.9	3.13	36.3	3.44	43.0
ARMS [2]	1.71	12.7	2.67	26.6	2.28	21.6	2.64	25.7
Ours	**1.14**	**6.25**	**1.87**	**16.4**	**1.54**	**11.8**	**1.46**	**11.1**

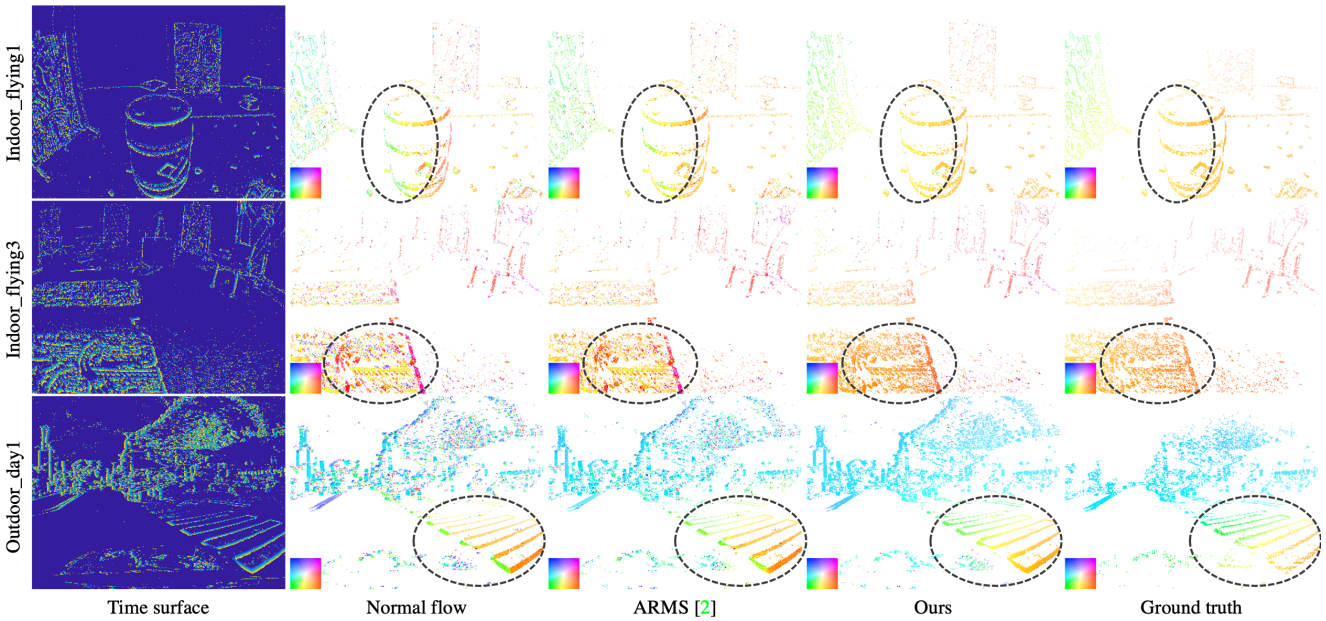

Figure 8. **The qualitative results on MVSEC**. See the color wheel on the bottom left for the color encoding.

scene from [19] where two stripes translating at different speeds are observed from a dynamic vision sensor (DVS).

Fig. 7 shows the result. We observed a significant shift in the ARMS's estimation when motions of different speeds are adjacent. That is because that ARMS determines the flow by selecting the scale having the maximum average norm in the window (it falsely picks the large norm in the slow region). In contrast, the TEGBP correctly estimates individual motion.

4.4. MVSEC

We quantitatively compare with ARMS in practical robotic scenarios. We used outdoor scenes captured by automobiles and indoor scenes captured by aerial drones from the Multi-vehicle Stereo Event Camera (MVSEC) dataset [33]. This dataset includes ground truth (GT) optical flow computed as the motion filed from the camera motion and scene depth. We evaluate the accuracy using the average endpoint error (AEE) and the percentage of pixels with endpoint error greater than 3 pixels (denoted by % Out). Both are measured on pixels where valid GT exists, and the plane fitting has been successful. We output flows asynchronously in batches with both methods and evaluate them at regular intervals (20Hz of GT rate).

Tab. 2 and Fig. 8 show the quantitative and qualitative results. ARMS improves the normal flow, while TEGBP improves more, resulting in significantly better accuracy. In contrast to the naive averaging used in ARMS, our TEGBP properly models the distribution of flows using TEG and prior.

4.5. DSEC

We compare the performance of TEGBP and ARMS on more diverse scenarios using more high-resolution input. To this end, we use the DSEC [13], a recently released dataset in automobile scenarios containing high-resolution event data of 480×640 captured from the vehicle that undergoes more diverse motion. It is designed for dense optical flow estimation, and the GT of the test sequence is hidden for leaderboard evaluation. Therefore we use their train sequence to compare sparse AEE error with ARMS.

Tab. 3 and Fig. 9 shows the quantitative and qualitative results. Similar to the MVSEC experiments, TEGBP significantly boosts accuracy over AMRS.

Table 3. **The quantitative results on DSEC**. The flow is evaluated on the ground truth intervals.

	thun_00_a		zurich_city_01_a		zurich_city_02_a		zurich_city_08_a		zurich_city_11_a	
	AEE ↓	% Out ↓	AEE ↓	% Out ↓	AEE ↓	% Out ↓	AEE ↓	% Out ↓	AEE ↓	% Out ↓
Norm	7.09	68.7	8.26	37.9	12.8	86.8	8.42	73.3	5.90	63.4
ARMS [2]	5.31	60.0	6.32	23.8	8.98	78.7	6.22	64.4	4.57	51.4
Ours	**4.26**	**45.5**	**5.61**	**23.2**	**8.70**	**72.6**	**5.08**	**49.5**	**3.27**	**32.2**

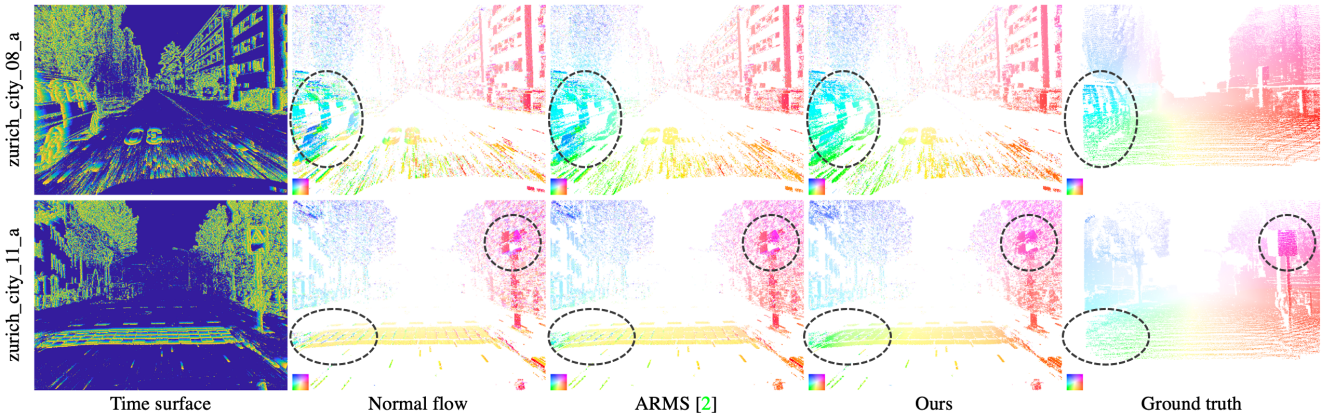

Figure 9. **The qualitative results on DSEC.**

5. Conclusion

We proposed TEGBP, a novel *incremental full* flow estimation algorithm for asynchronous event data. The proposed algorithm is backed up by the probabilistic model, which guarantees equality to the full flow when a valid prior is provided. It realizes efficient event-wise incremental updates based on the BP, where the MAP solution obtained by the local update is guaranteed to match the correct one. We demonstrate the effectiveness of the proposed method on a practical real-world dataset showing a significant boost in accuracy over the existing incremental algorithm.

There is still much room for improvement in accuracy compared to the recent computationally intensive machine learning-based method. We suspect the gap mainly comes from the noise in the normal flow and poor prior.

Toward the ultimate goal of realizing low-latency and high-accuracy full flow estimation in real-time on an actual edge device, we expect the gap to be closed by incorporating more robust normal flow estimation and more sophisticated prior.

5.1. Future work

Robust normal flow estimation As discussed in the experiment section, some scenes include strongly erroneous flow, e.g., normal flow from leaves on the tree. These measurement errors significantly degrade the accuracy. We expect the tiny CNN (e.g., input size of 7×7) trained to regress the normal flow from noisy input will improve the accuracy without sacrificing the computational efficiency.

Application specific smoothness prior Even if the perfect normal flow is obtained, we cannot expect good accuracy without good prior (Sec. 3.3.2). The uniform smoothness prior we utilized is not optimal in many practical scenarios. The discontinuous depth region does not conform to the prior. Even if the planer object undergoes linear motion, we won't observe uniform flow when it is not perpendicular to the optical axis. We can utilize more appropriate prior for a specific application to significantly boost the accuracy. For example, we may utilize the prior utilizing vanishing point for automotive applications (assuming the yaw rate is known from the vehicle sensor) similar to [20]. We left the exploration in this direction for future work.

Optimized implementation Our algorithm is very efficient in principle, requiring lower FLOPS, and the entire process can be parallelized. The normal flow could be cheaply obtained, e.g., directly from a camera or computed by the lightweight plane fitting (computation of normal flow is outside this study's scope). The core of our algorithm, message passing of (5), is also very cheap, requiring the communication of a tiny 6-dim vector with four neighbors. Therefore, we expect the optimized CPU implementation or adaptation of processors supporting the sparse computation could realize low-energy low-latency optical flow estimation in the edge device. We left the wall clock time and energy consumption evaluation as future work.

References

[1] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust Map Optimization using Dynamic Covariance Scaling. *IEEE International Conference on Robotics and Automation (ICRA)*, 2013. 5

[2] Himanshu Akolkar, Sio Hoi Ieng, and Ryad Benosman. Real-time high speed motion prediction using fast aperture-robust event-driven visual flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):1–12, 2020. 1, 2, 4, 5, 6, 7, 8, 11

[3] Myo Tun Aung, Rodney Teo, and Garrick Orchard. Event-based Plane-fitting Optical Flow for Dynamic Vision Sensors in FPGA. *IEEE International Symposium on Circuits and Systems*, 2018. 2

[4] R. Wes Baldwin, Mohammed Almatrafi, Vijayan Asari, and Keigo Hirakawa. Event Probability Mask (EPM) and Event Denoising Convolutional Neural Network (EDnCNN) for Neuromorphic Cameras. 2020. 3

[5] Patrick Bardow, Andrew J. Davison, and Stefan Leutenegger. Simultaneous Optical Flow and Intensity Estimation from an Event Camera. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 884–892, 2016. 2

[6] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-Based Visual Flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):407–417, feb 2014. 1, 2

[7] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Networks*, 27:32–37, mar 2012. 1, 2

[8] Andrew J. Davison. FutureMapping: The Computational Structure of Spatial AI Systems. *arXiv*, 2018. 3

[9] Andrew J. Davison and Joseph Ortiz. FutureMapping 2: Gaussian Belief Propagation for Spatial AI. *arXiv*, 2019. 2, 3, 5

[10] Pedro F Felzenszwalb and Daniel P Huttenlocher. Ef fi cient Belief Propagation for Early Vision. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004. 5, 13

[11] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(1), 2020. 2

[12] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3867–3876, 2018. 2

[13] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. DSEC: A Stereo Event Camera Dataset for Driving Scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954, 2021. 7

[14] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-RAFT: Dense Optical Flow from Event Cameras. *International Conference on 3D Vision (3DV)*, 2021. 2

[15] B. K. B. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17(1-3):185–203, 1981. 1

[16] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Bill Nell, Nir Shavit, and Dan Alistarh. Inducing and Exploiting Activation Sparsity for Fast Neural Network Inference. *Proceedings of Machine Learning Research*, 119:5533–5543, 2020. 3

[17] Weng Fei Low, Zhi Gao, Cheng Xiang, and Bharath Ramesh. SOFEA: A non-iterative and robust optical flow estimation algorithm for dynamic vision sensors. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020. 2, 6

[18] MATLAB. *version 9.13.0 (R2022b)*. The MathWorks Inc., Natick, Massachusetts, 2022. 6

[19] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime Estimation of Events from Dynamic Vision Sensors. *IEEE International Conference on Robotics and Automation (ICRA)*, 2015. 2, 7

[20] Jun Nagata, Yusuke Sekikawa, Kosuke Hara, and Yoshimitsu Aoki. FOE-based Regularization for Optical Flow Estimation from an In-vehicle Event Camera. *Proceedings of SPIE - The International Society for Optical Engineering*, 11049, 2019. 8

[21] Joseph Ortiz, Mark Pupilli, Stefan Leutenegger, and Andrew J. Davison. Bundle Adjustment on a Graph Processor. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2413–2422, 2020. 3

[22] Liyuan Pan, Miaomiao Liu, and Richard Hartley. Single Image Optical Flow Estimation with an Event Camera. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[23] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. ESIM: an Open Event Camera Simulator. *Conf. on Robotics Learning (CoRL)*, 87:969–982, 2018. 6

[24] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Event Collapse in Contrast Maximization Frameworks. *MDPI Sensors*, 22(14):1–20, 2022. 2

[25] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Secrets of Event-Based Optical Flow. *European Conference on Computer Vision (ECCV)*, pages 1–23, 2022. 2

[26] Chen Shoushun and Guo Menghan. Live Demonstration : CeleX-V : a 1M Pixel Multi-Mode Event-based Sensor Chen Shoushun. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–2, 2019. 3

[27] Timo Stoffregen and Lindsay Kleeman. Event Cameras , Contrast Maximization and Reward Functions : an Analysis. *Cvpr*, 2019. 2

[28] Daniel C. Stumpp, Himanshu Akolkar, Alan D. George, and Ryad B. Benosman. hARMS: A Hardware Acceleration Architecture for Real-Time Event-Based Optical Flow. *IEEE Access*, 10:58181–58198, 2022. 3

[29] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. *European Conference on Computer Vision (ECCV)*, 2020. 1, 2

[30] Yair Weiss and William T Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Information Processing Systems (NIPS)*, pages 673–679, 2000. 5

[31] Chengxi Ye, Anton Mitrokhin, Cornelia Fermüller, James A Yorke, and Yiannis Aloimonos. Unsupervised Learning of Dense Optical Flow, Depth and Egomotion from Sparse Event Data. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 2

[32] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras. *Proceedings of Robotics: Science and Systems*, jun 2018. 2

[33] Alex Zihao Zhu, Dinesh Thakur, Tolga Ozaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The Multi Vehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018. 7

[34] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based optical flow using motion compensation. *European Conference on Computer Vision (ECCV) Workshop*, pages 711–714, 2018. 2

[35] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 989–997, 2019. 2

Tangentially Elongated Gaussian Belief Propagation for Event-based Incremental Optical Flow Estimation

Paper ID 4712

A. Semi-dense estimation

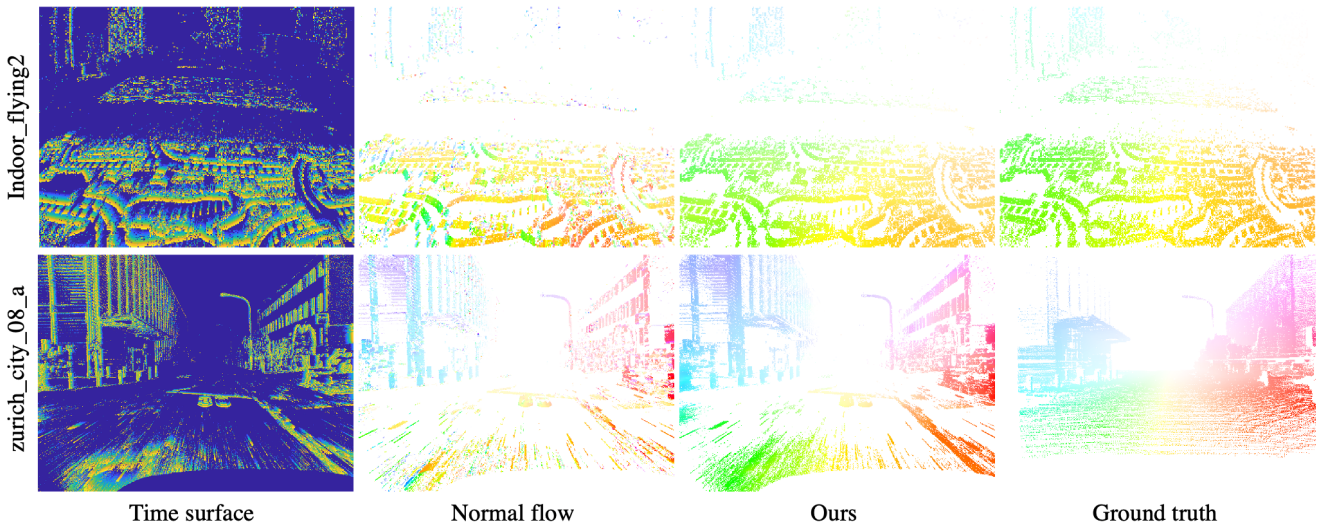

Figure A1. **Semi-dense estimation**. MVSEC (top) and DSEC (bottom). Refer to the `supp_MVSEC.mp4` for MVSEC and `supp_DSEC.mp4` for DSEC in the supplement.

In this supplemental section, we show the result of the TEGBP extension for *semi-dense* optical flow estimation. In the main paper, we described the method for estimating the flow at the sparse pixels, where the normal flow is observed (Sec. 3). We evaluate and report the results of the sparse estimation, mainly for the comparison with ARMS [2]. Normal flow estimation often fails, therefore, normal flow observation points are much more sparse than the raw event's points.

Our TEGBP can be easily extended to estimate the flow at any pixel point where we do not have an observation. In this section, we show the results on MVSEC and DSEC by estimating the full flow at events' points (semi-dense). By the semi-dense estimation, we can have an estimation on more pixels at the cost of a slight increase in the computational cost.

The extension to semi-dense (or dense) is simple. It is realized by sending the message to the pixel locations where we need an estimation. In the case of a semi-dense scenario, it is realized by setting the node having the observation of the raw event as *active* (irrespective of the plane fitting results).

Fig. A1 shows a snapshot from the semi-dense estimation on MVSEC and DSEC. Refer to the `supp_MVSEC.mp4` for MVSEC and `supp_DSEC.mp4` for DSEC in the supplement. We've selected, `indoor_flying2` from MVSEC and `zurich_city_08_a` from DSEC.

Note that, for this video demonstration, we estimate the semi-dense optical flow by exchanging messages at regular time intervals (e.g. a frame rate).

B. Ablation on the TEG parameters

In this section, we ablate the TEG parameter to demonstrate the key ingredient of the proposed TEGBP. As discussed in Sec. 3, our core idea for realizing the robust full-flow estimation is modeling the distribution of full-flow as TEG, Gaussian distribution having large variance along the tangential direction of the normal flow. We show the result by changing the variance along the tangential direction σ_t.

B.1. Toy example of Fig. 1

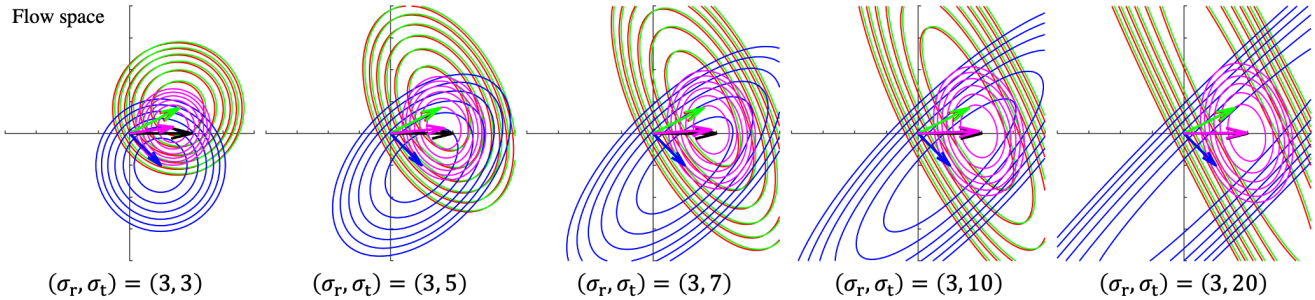

Figure A2. **TEG and its marginal for different** σ_t. Normal flow measurements modeled as TEG (RGB ellipse, having different σ_t) and its marginal with a uniform prior (magenta ellipse) are shown. The black arrow represents the true full flow.

Fig. A2 visualize the marginal distribution of the full flow in the case of toy example of Fig. 1 by changing σ_t. When σ_t is small and the TEG does not model the distribution of full flow from the normal flow measurement well, the full flow estimation, i.e, MAP solution of the marginal, is largely different from the true flow. As the σ_t becomes larger, and the TEG well represents the distribution of full flow, the MAP solution gets closer to the correct full flow. And the limit of $\sigma_t = \infty, \sigma_r = 0$, it equals to the true full flow as we showed in Sec. 3.3.2.

The MATLAB script for this visualization is attached to the supplement to help understand our core algorithm. It does not utilize the multi-scale scheme (Sec. C.1) for simplicity. Refer to the comments in the code for the implementation detail of the TEG construction from normal flow measurements and the proposed message-passing scheme. (We've also added a pointer to the important equations in the comments).

B.2. ESIM_bricks

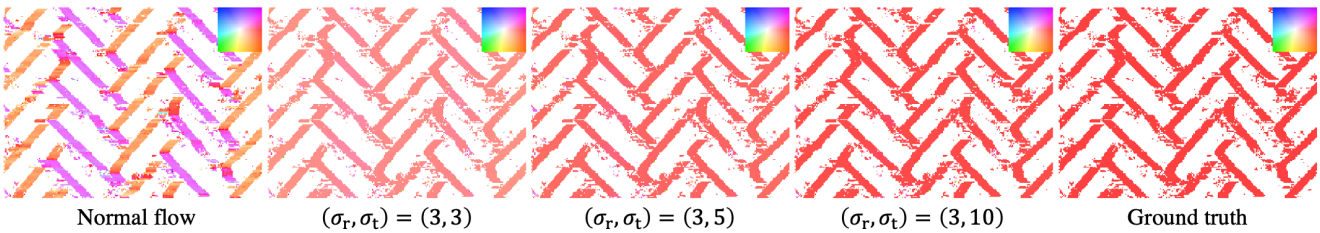

Figure A3. **Full flow estimation results using different** σ_t.

Fig. A3 shows the result by using different value of σ_t (refer to the detailed setup for Sec. 4.2). We'll get accurate flow as the TEG well represents the distribution of full flow by increasing σ_t.

C. Implementation Detail

Upon acceptance, we'll open source all the code to reproduce the results in this paper, e.g., the code for MVSEC and DSEC experiments along with the dataset for ESIM_bricks.

C.1. Coarse-to-fine message-passing

We adopted the coarse-to-fine message-passing scheme of [10] to speed up the convergence. The measurement factor of the coarser layer l is computed as the sum of active measurement factor messages in a 2×2 block of finer layer $l-1$ as follows:

$$\boldsymbol{\eta}_i^l = \sum_{j \in \mathcal{B}_i \cap \mathcal{A}_t} \boldsymbol{\eta}_j^{l-1}, \quad \Lambda_i^l = \sum_{j \in \mathcal{B}_i \cap \mathcal{A}_t} \Lambda_j^{l-1}, \tag{A1}$$

where $\mathcal{B}_i$ is the block corresponding the node i.

C.2. Algorithm

The algorithm of the proposed TEGBP is listed in Algorithm 1. The message passing part which corresponds to Fig. 5 is listed separately in Algorithm 2.

Algorithm 1 TEGBP

for all incoming normal flow $e_i = \{\mathbf{x}_i, t_i, \mathbf{v}_i^{\perp}\}$ **do**
  Calculate the message from the data factor by Eq. 5 and Eq. 9 and set to node at pixel $\mathbf{x}_i$.
  Update hierarchical data factor by Eq. A1.
  for all l in $[L, \cdots, 1]$ **do**
    Update belief by Eq. 11
    Message passing (Algorithm 2)
    if $l > 1$ **then**
      Copy the messages from a coarser layer l to a finer layer $l-1$.
    end if
  end for
  Determine the estimated value $\mathbf{v}_i$ as the mean of Gaussian belief.
end for

Algorithm 2 Message passing at node i

Add node i to queue Q.
for all k in K hops **do**
  Pop all nodes in Q and add to the temporary queue Q'.
  for all j popped from Q' **do**
    Message passing by Eq. 12.
    for all $k \in \mathcal{A}_t \cap \mathcal{C}_j$ **do**
      Update belief by Eq. 11.
      Add node k to queue Q.
    end for
  end for
end for