What’s in the Flow? Exploiting Temporal Motion Cues for Unsupervised Generic Event Boundary Detection

Sourabh Vasant Gothe, Vibhav Agarwal, Sourav Ghosh, Jayesh Rajkumar Vachhani, Pranay Kashyap, Barath Raj Kandur Raja

Samsung R&D Institute Bangalore, India
{sourab.gothe, vibhav.a, sourav.ghosh, jay.vachhani, kashyap.p, barathraj.kr}@samsung.com

Abstract

Generic Event Boundary Detection (GEBD) task aims to recognize generic, taxonomy-free boundaries that segment a video into meaningful events. Current methods typically involve a neural model trained on a large volume of data, demanding substantial computational power and storage space. We explore two pivotal questions pertaining to GEBD: Can non-parametric algorithms outperform unsupervised neural methods? Does motion information alone suffice for high performance? This inquiry drives us to algorithmically harness motion cues for identifying generic event boundaries in videos. In this work, we propose FlowGEBD, a non-parametric, unsupervised technique for GEBD. Our approach entails two algorithms utilizing optical flow: (i) Pixel Tracking and (ii) Flow Normalization. By conducting thorough experimentation on the challenging Kinetics-GEBD and TAPOS datasets, our results establish FlowGEBD as the new state-of-the-art (SOTA) among unsupervised methods. FlowGEBD exceeds the neural models on the Kinetics-GEBD dataset by obtaining an F1@0.05 score of 0.713 with an absolute gain of 31.7% compared to the unsupervised baseline and achieves an average F1 score of 0.623 on the TAPOS validation dataset.

1. Introduction

In 2023, video has accounted for 82.5% of all web traffic, making it the most popular form of content on the internet. Increased video consumption has made video understanding a critical task in computer vision, comprising video classification [8, 23, 35], object segmentation [39, 40], action localization [3, 37, 41], and captioning [38, 42], among others. However, the memory requirements of the models, the feasibility of real-time inference, and domain generalization are typical constraints on these solutions.

Current state-of-the-art video models [9, 17, 21, 31, 33] have been mainly focused on building upon a limited set of predefined action classes and usually process short clips to generate global video-level predictions. With the growth of video content, the number of classes is expanding, and the predefined target classes cannot encompass them all. Recently, the GEBD [30] task was introduced with the objective of studying the long-form video understanding problem through the lens of human perception [34]. GEBD aims to locate class-agnostic event boundaries in a video, regardless of its category. It considers the following high-level causes of event boundaries: changes in subject, action, shot, environment, and object of interaction. The outcome of GEBD has many potential applications: video summarization, video editing, short video segment sharing (Fig. 2), and enhancing video classification and other downstream tasks [38].

There are three main approaches to the GEBD problem: supervised, unsupervised, and self-supervised. Several supervised methods [10, 12, 14, 19, 20, 32] propose to decode the self-similarity representation produced by the frame features for boundary detection. The literature investigates a wide variety of approaches, including efficient represen-
2. Related Work

2.1. Generic Event Boundary Detection

Generic event boundary detection (GEBD) [30] aims to localize the moments where humans naturally perceive taxonomy-free event boundaries that break a longer video into shorter temporal segments. Previous methods [12, 30, 32] formulate the GEBD task as binary classification, which predicts the boundary label of each frame by considering the temporal context information. However, it could be more efficient because the redundant computation is conducted while generating the representations of consecutive frames. Kang et al. [14] proposed to use the temporal self-similarity matrix (TSM) as intermediate representation and used contrastive learning as an auxiliary to learn better from the TSM results. Li et al. [20] proposed solving GEBD using the compressed video features and achieved 4.5× faster-running speed than the baseline method [30] on GPU. Recently, Gothe et al. [10] developed the most miniature model to solve the GEBD task with the lowest inference time on GPU. SC-Transformer [19] introduced a structured partition of sequences (SPoS) mechanism to learn structured context using a transformer-based architecture for GEBD. To enrich motion information, optical flow is introduced as a new modality in [13]. TeG [26] proposed a generic self-supervised model for learning persistent and more fine-grained features and uses a 3D-ResNet-50 encoder as its backbone. However, all these methods require substantial memory and computational resources along with the labeled data.

Regarding unsupervised GEBD approaches, PySceneDetect [2] is a Python library that detects shot changes by considering pixel changes in the HSV colorspace. However, generic event boundaries consist of various boundary causes like the change of action, subject, and environment, implying that only a tiny portion of event boundaries can be detected with this approach. PredictAbility (PA) [30] computationally assesses the predictability score over time and then locates the event boundaries by detecting the local minima of the predictability sequence. CoSeg [36] devises a transformer-based frame feature reconstruction scheme and adopts ResNet-18 [11] as the backbone. UBoCo [15] proposes an unsupervised/supervised method using the TSM as the video representation. UBoCo’s unsupervised framework for GEBD combines Recursive TSM Parsing (RTP) and the Boundary Contrastive (BoCo) loss. However these models belong to a high memory regime.
2.2. Learning motion and visual correspondences

Motion plays a crucial role in video understanding, and many SOTA models [13, 18, 22, 27, 32] incorporate motion information by using optical flows. Lucas and Kanade’s image registration method [24], also known as gradient-based optical flow, enables motion estimation possible with high-speed computation. Pyramidal Lucas and Kanade [1], Gunnar Farneback [5–7] are other well-known methods for motion estimation.

DDM-Net [32] applies progressive attention to multilevel dense difference maps (DDM) to characterize motion patterns and jointly aggregate motion and appearance cues in a supervised setting. MotionSqueeze (MS) [18] introduces an end-to-end trainable, model-agnostic and lightweight module to extract motion features on the fly for video understanding. However, it requires training via backpropagation and integration with pre-existing video architectures. Rai et al. [27] presents a self-supervised model for GEBD by reformulating training objectives at frame-level and clip-level to learn effective video representations using the MS [18] module. However, these are parametric methods that require training on large datasets. To the best of our knowledge, there is no unsupervised and non-parametric (algorithmic) solution with high performance in generic event boundary detection.

3. Proposed Methodology

GEBD takes a video as input and returns a set of boundary timestamps. Mathematically, it maps an ordered sequence of L frames, \( f_1, f_2, \ldots, f_L \) (that may also be represented as \( F \in \mathbb{F} \)), to a set of timestamps \( \{ b_1, b_2, \ldots, b_M \} = B \), that denote the event boundaries. It then naturally follows that \( M \leq (L - 1) \). For all practical purposes, \( M \ll L \) and \( b_j \in B, j \), such that timestamp \( b_i \) corresponds to a unique frame \( f_j \). Thus, we formulate the GEBD task as:

\[
T : \mathbb{F} \rightarrow \mathbb{B}
\] (1)

Here, we describe our approach, FlowGEBD, that solves this task using pixel tracking, flow normalization, and their ensemble (with temporal refinement) as shown in Fig. 3.

3.1. FlowGEBD with Pixel Tracking (PT)

In this section, we present a method that leverages sparse optical flow to determine event boundaries by monitoring the flow of a subset of pixels.

3.1.1 Framewise mode

We process a video frame-by-frame, considering each frame as a unit. Each frame \( f \) of width \( w \) and height \( h \) comprises a 2-dimensional matrix of pixels, \( p_{u,v} \), where \( u,v \in \mathbb{Z}^+ \) (positive integers), \( u \in [1,w] \), and \( v \in [1,h] \). We only consider the luminance information of pixels. Hence, \( p_{u,v} \) can be represented as a real number \( (p_{u,v} \in \mathbb{R}) \), \( 0 \leq p_{u,v} \leq 1 \).
We define two categories of patches where patches of the same category do not overlap each other. The first among these are “base patches”, which distribute the pixels equally and independently along the width and height of the frame. We refer to the other category as “centroidal patches”, as each of their edges joins the centroids of adjacent base patches. Centroidal patches help capture events that span across the intersections of base patches. Fig. 3 (right) depicts the arrangement of patches, where the total number of patches (each of width \(w_g\) and height \(h_g\)) is given by:

\[
N_g = n_w \times n_h + (n_w - 1) \times (n_h - 1)
\]

Where \(n_w\) and \(n_h\) represent the cardinality of base patches along the frame width and height, respectively.

**Method.** In this mode, we independently process the entire frame sequence \(N_g\) times, once for each patch, considering only the corresponding patches. During this, we skip the temporal boundary refinement stage mentioned in Algorithm 1. We then take a union of the \(N_g\) predicted boundary sets and apply boundary refinement to derive \(B\). It may be noted that framewise mode is a specialized case of patchwise, where \(n_w = n_h = 1 \implies N_g = 1\).

### 3.2. FlowGEBD with Optical Flow Normalization

We explore another approach that leverages dense optical flow and determines event boundaries by observing the normalized optical flow.

**3.2.1 Framewise mode**

As we have observed in FlowGEBD with pixel tracking (section 3.1.2), the framewise mode is a specialized case of patchwise. Hence, we discuss the Optical Flow Normalization method in the more generalized patchwise mode, and the same can be adapted for framewise by using \(n_w = n_h = 1\).

**3.2.2 Patchwise mode**

**Method.** For every frame \(f_i\), we identify the set of \(N_g\) patches, denoted by \(G_{f_i}\), for a fixed \(n_w\) and \(n_h\). Then, for every consecutive frame \(f_{i-1}\) and \(f_i\), we compute \(N_g\) dense optical flows (one between each patch pair, \(g_{fi-1}\) and \(g_{fi}\)). For this, we use the Gunnar Farneback algorithm [7]. Then, we use the maximum flow displacement corresponding to all patch pixels as the “flow of the patch” or its “PatchFlow”. After processing \(L\) frames and their \(N_g\) patches, we accumulate the PatchFlow for each patch across temporal dimension and normalize the displacement values. We hypothesize that considerable displacement of PatchFlow in the temporal dimension constitutes a change in action or event. For any patch, if the normalized value for frame index \(i\) exceeds a constant threshold \(\theta_2\), we deem to have encountered an event boundary and add the corresponding frame index \(i\) to our

Can we improve further? The framewise approach monitors the key pixels in a frame. However, for a video of a moderately large field of view, it is often the case that an event boundary may be denoted by the change in actions of certain subjects in the video, even if the background remains static. Furthermore, the main subjects of the frame are typically positioned along the grid lines and at the intersections to make it more aesthetically pleasing following the Rule of thirds [4]. For such cases, the event boundary can be determined more accurately if we decompose a frame into a grid of patches instead of having the entire frame as a single unit. Patchwise processing provides advantages like capturing subject change in small areas and detecting action change from one patch to another. Thus, in the next section, we propose an approach that processes each frame as a composition of multiple patches.

### 3.1.2 Patchwise mode

A patch \(g_f\), derived from frame \(f\), consists of a subset of the frame pixels. More specifically, patch \(g_f(u, v, w_g, h_g)\) consists of all pixels, \(p_{i,j} \in f\), where \(i, j \in \mathbb{Z}^+, i \in [u, u + w_g)\) and \(j \in [v, v + h_g)\). We denote the set of all such patches in frame \(f\) as \(G_f\).
working set of event boundaries $B_{temp}$. Finally, we apply temporal refinement on $B_{temp}$ to compute the refined event boundary set $B$. This approach for patchwise processing of dense optical flow is detailed in Algorithm 2.

### 3.3. FlowGEBD with ensembling of Pixel Tracking and Flow Normalization

Pixel tracking helps determine event boundaries based on the sparse optical flow of a few key pixels. On the other hand, flow normalization aggregates the dense optical flow of all pixels and offers a lossless method to determine a set of event boundaries using “PatchFlow”. To obtain an ensemble of both approaches, we independently take the event boundaries from both without performing the temporal refinement stage. Instead, we take a union of the predicted sets from these two approaches and perform temporal refinement over the union.

**Temporal Refinement.** We analyze the elements of a set of predicted boundary timestamps along the corresponding temporal dimension to identify “rare boundaries” and “popular boundaries”. Rare or isolated boundaries are those where the event changes in a single patch and typically with no neighboring timestamps, i.e., for a rare boundary $b_i$, there exists no other boundary within the duration $b_i \pm \theta_3$. On the other hand, popular boundaries are dense clusters where multiple boundaries have been determined within the temporal vicinity.

We may interpret contiguous popular boundaries as belonging to one cluster and each rare boundary as a standalone single-element cluster. Then, we select one representative element for each cluster by identifying its median boundary element and consider only such elements for the final set of refined event boundaries. The generic Algorithm 3 identifies such clusters and determines an optimal boundary in each of them.

### 4. Experiments

In this section, we conduct multiple experiments and evaluate both algorithms, followed by the ensembled method.

### 4.1. Dataset

**Kinetics-GEBD.** Our approach is evaluated primarily on the challenging Kinetics-GEBD [30], a benchmark dataset for locating the boundaries of generic events in the video. It consists of 54,691 videos of 10 seconds each that span a broad spectrum of video domains in the wild and is open-vocabulary, taxonomy-free. The ratio of the train, validation, and test sets in Kinetics-GEBD is equal, with each set including roughly 18,000 videos chosen from Kinetics-400 [16]. FlowGEBD is an algorithmic unsupervised method, so we do not require train data. We evaluate our methods on the val-

---

**Algorithm 2:** FlowGEBD using Optical Flow Normalization (Patchwise mode)

- **Data:** Video of resolution $w \times h$ as a sequence of $L$ frames, $F = (f_1, f_2, \ldots, f_L)$; Parameters:
  - Patch width $w_g$ and height $h_g$

- **Result:** Event Boundaries, $B = \{b_1, b_2, \ldots, b_M\}$

```plaintext
G_{f_i} \leftarrow \text{patches}(f_i, w_g, h_g) \quad // \text{Gets} \ N_g \text{ patches (Fig. 3)}
for f_i \in (F - f_1) do
    G_{f_i} \leftarrow \text{patches}(f_i, w_g, h_g)
    \Phi^g_{i-1} \leftarrow \{} \quad // \text{Placeholder for all optical flows in frame } f_i
    \text{for } (g_{f_{i-1}} g_{f_i}) \in (G_{f_{i-1}}, G_{f_i}) \text{ do}
        \varphi \leftarrow \text{denseFlow}(g_{f_{i-1}}, g_{f_i}) \quad // O(w_g h_g)
        \Phi^g_{i-1} \leftarrow \Phi^g_{i-1} \cup \{\text{max}(\varphi)\}
    end
    end
    end
    end
    B_{temp} \leftarrow \{\}
    \text{for } \Phi^g_1, \Phi^g_2, \ldots, \Phi^g_{L-1} \in \text{PatchFlows} \quad // \text{corresponding patches}
    \Phi \leftarrow \left(\Phi^g_1, \Phi^g_2, \ldots, \Phi^g_{L-1}\right)
    ||\Phi||_2 \quad // \text{L2-norm}
    B_{temp} \leftarrow B_{temp} \cup \{\text{argmax}_{\theta_2}(\Phi)\}
end
\text{return } B_{temp} \quad // \text{Refer to Algorithm 3}
B \leftarrow \text{refine}(B_{temp}) \quad // \text{Flush } K
\text{return } B
```

denseFlow(): Dense optical flow using the Gunnar Farneback’s algorithm

$\theta_2$: A constant threshold

---

**Algorithm 3:** Temporal Refinement of Boundaries

- **Data:** Event Boundaries, $B = \{b_1, b_2, \ldots, b_M\}$
- **Result:** Refined Event Boundaries, $\hat{B} \subseteq B$

```plaintext
\hat{B}_{temp} \leftarrow \{\}; \hat{B} \leftarrow \text{sorted}(B) \quad // O(M \log M)
K \leftarrow () \quad // Placeholder for a cluster of boundary elements
for b_i \in B do
    \text{if } (K \neq () \land (\forall k \in K | (|b_i - k| \geq \theta_3)))
        \hat{B}_{temp} \leftarrow \hat{B}_{temp} \cup \{\text{median}(K)\}
        K \leftarrow ()
    end
    K \leftarrow K \cup \hat{b_i}
\hat{B}_{temp} \leftarrow \hat{B}_{temp} \cup \{\text{median}(K)\} \quad // \text{Flush } K
\text{return } \hat{B}_{temp} \text{ as } \hat{B}
```

$\theta_3$: A constant threshold
with 21 actions. The dataset authors manually defined how

Table 2. F1 results on TAPOS validation set with different Rel.Dis. thresholds. The ensembled method achieves the best F1 score compared

Table 1. F1 results on Kinetics-GEBD validation set with different Rel.Dis. thresholds. FlowGEBD achieves the best F1@0.05 scores for

4.2. Main Results

Kinetics-GEBD. Table 1 illustrates the results of our methods on the Kinetics-GEBD validation set along with unsupervised and supervised benchmarks. PT achieves a higher F1@0.05 of 0.702 than FN, with a strong recall of 0.91 (average) and a 0.77 (average) precision. Intuitively, PT is able to detect action change across patches along with subject change as the cardinality of the patch increases. FN obtains a high Avg. F1 value through a balanced precision and recall of 0.81 and 0.90, respectively. The FlowGEBD (Ensembled) outperforms all previous unsupervised methods with the highest F1@0.05 of 0.713 using a refinement approach (Algorithm 3) that combines PT and FN. Compared to unsupervised baseline PA [30], FlowGEBD obtains a significant gain of 31.7% in F1@0.05 and exceeds DNN-based unsupervised methods [15, 36], demonstrating the effectiveness of our proposed algorithms. Additionally, compared to the supervised baselines PC [30] and PC + Optical Flow [20], our method achieves 8.8% and 6.7% absolute improvement, respectively.

**TAPOS.** In addition to Kinetics-GEBD, we experiment on the TAPOS dataset [28] containing Olympics sports videos with 21 actions. The dataset authors manually defined how to break each action into sub-actions during annotation. Following [30], we re-purpose TAPOS for the GEBD task by performing boundaries localization between sub-actions in each action instance. TAPOS contains 1790 instances for validation, and we evaluate on the same.

**Implementation and Evaluation.** We run all our experiments on Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz equipped machine. We sample the video at 4 FPS and resize it to 160 × 160 as preprocessing. As described in [30], we use F1 at 0.05 Relative Distance (Rel.Dis.) as our primary evaluation metric. The predicted boundary is deemed accurate for a certain Rel.Dis. threshold if the difference between the predicted and ground truth timestamps is smaller than the threshold. We report F1 scores of different thresholds to range from 0.05 to 0.5 with a gap of 0.05.

<table>
<thead>
<tr>
<th>Supervision</th>
<th>Rel.Dis. threshold</th>
<th>0.05</th>
<th>0.1</th>
<th>0.15</th>
<th>0.2</th>
<th>0.25</th>
<th>0.3</th>
<th>0.35</th>
<th>0.4</th>
<th>0.45</th>
<th>0.5</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP [10] (Baseline)</td>
<td>0.062</td>
<td>0.735</td>
<td>0.804</td>
<td>0.829</td>
<td>0.844</td>
<td>0.853</td>
<td>0.859</td>
<td>0.864</td>
<td>0.874</td>
<td>0.877</td>
<td>0.879</td>
<td>0.830</td>
</tr>
<tr>
<td>PC + Optical Flow [20]</td>
<td>0.646</td>
<td>0.776</td>
<td>0.818</td>
<td>0.842</td>
<td>0.856</td>
<td>0.864</td>
<td>0.868</td>
<td>0.874</td>
<td>0.877</td>
<td>0.879</td>
<td>0.880</td>
<td></td>
</tr>
<tr>
<td>Gothe et al. [10]</td>
<td>0.712</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC-Transformer [19]</td>
<td>0.777</td>
<td>0.849</td>
<td>0.873</td>
<td>0.886</td>
<td>0.895</td>
<td>0.904</td>
<td>0.908</td>
<td>0.913</td>
<td>0.916</td>
<td>0.919</td>
<td>0.921</td>
<td>0.881</td>
</tr>
<tr>
<td>SBoCo-TSN [15]</td>
<td>0.787</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.892</td>
</tr>
</tbody>
</table>

**Unsupervised**

<table>
<thead>
<tr>
<th>Supervision</th>
<th>Rel.Dis. threshold</th>
<th>0.05</th>
<th>0.1</th>
<th>0.15</th>
<th>0.2</th>
<th>0.25</th>
<th>0.3</th>
<th>0.35</th>
<th>0.4</th>
<th>0.45</th>
<th>0.5</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA - Random [30]</td>
<td>0.336</td>
<td>0.435</td>
<td>0.484</td>
<td>0.512</td>
<td>0.529</td>
<td>0.541</td>
<td>0.548</td>
<td>0.554</td>
<td>0.558</td>
<td>0.561</td>
<td>0.564</td>
<td>0.506</td>
</tr>
<tr>
<td>PC [30]</td>
<td>0.396</td>
<td>0.488</td>
<td>0.520</td>
<td>0.534</td>
<td>0.544</td>
<td>0.550</td>
<td>0.555</td>
<td>0.558</td>
<td>0.561</td>
<td>0.564</td>
<td>0.527</td>
<td></td>
</tr>
<tr>
<td>PA [30]</td>
<td>0.656</td>
<td>0.758</td>
<td>0.783</td>
<td>0.794</td>
<td>0.799</td>
<td>0.803</td>
<td>0.806</td>
<td>0.808</td>
<td>0.810</td>
<td>0.810</td>
<td>0.782</td>
<td></td>
</tr>
<tr>
<td>SC-Transformer [19]</td>
<td>0.703</td>
<td>0.839</td>
<td>0.862</td>
<td>0.885</td>
<td>0.889</td>
<td>0.893</td>
<td>0.894</td>
<td>0.898</td>
<td>0.899</td>
<td>0.902</td>
<td>0.866</td>
<td></td>
</tr>
<tr>
<td>UBoCo-Reto [15]</td>
<td>0.702</td>
<td>0.846</td>
<td>0.862</td>
<td>0.879</td>
<td>0.888</td>
<td>0.889</td>
<td>0.895</td>
<td>0.897</td>
<td>0.904</td>
<td>0.905</td>
<td>0.866</td>
<td></td>
</tr>
<tr>
<td>SceneDetect [2]</td>
<td>0.275</td>
<td>0.300</td>
<td>0.312</td>
<td>0.319</td>
<td>0.324</td>
<td>0.327</td>
<td>0.330</td>
<td>0.332</td>
<td>0.334</td>
<td>0.335</td>
<td>0.318</td>
<td></td>
</tr>
<tr>
<td>Ours (PT 1)</td>
<td>0.702</td>
<td>0.819</td>
<td>0.844</td>
<td>0.855</td>
<td>0.860</td>
<td>0.863</td>
<td>0.866</td>
<td>0.867</td>
<td>0.869</td>
<td>0.870</td>
<td>0.841</td>
<td></td>
</tr>
<tr>
<td>Ours (FN 2)</td>
<td>0.691</td>
<td>0.826</td>
<td>0.860</td>
<td>0.877</td>
<td>0.885</td>
<td>0.889</td>
<td>0.892</td>
<td>0.894</td>
<td>0.896</td>
<td>0.897</td>
<td>0.861</td>
<td></td>
</tr>
<tr>
<td>Ours (Ensembled)*</td>
<td>0.713</td>
<td>0.828</td>
<td>0.850</td>
<td>0.858</td>
<td>0.862</td>
<td>0.864</td>
<td>0.866</td>
<td>0.867</td>
<td>0.868</td>
<td>0.869</td>
<td>0.845</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. F1 results on TAPOS validation set with different Rel.Dis. thresholds. The ensembled method achieves the best F1 score compared to other unsupervised methods. †: parametric (neural) Methods ×: Non-parametric Methods
applied directly to the TAPOS dataset. Compared to the unsupervised benchmark PA [30], our method obtains Avg. F1 score of 0.623, gaining 8% absolute improvement. We found no alternative SOTA unsupervised methods for GEBD on the TAPOS dataset to compare our results directly.

**Tuning Thresholds**  \( \theta_1 \) and \( \theta_2 \) serve as thresholds that govern the behavior of the PT and FN algorithms, respectively. Additionally, \( \theta_3 \) is a threshold regulating neighboring boundaries for clustering in the refinement process.

As per the hypothesis, in PT, a notable decrease in pixel count during temporal tracking indicates pivotal event changes, such as changes in subjects or environment. Our empirical findings indicate that marking an event boundary with a drop exceeding 60% (\( \theta_1 = 0.4 \)) yields better performance. In Fig. 4, it can be observed that the event boundaries are perfectly aligned with the trough. Likewise, in FN, our empirical observations indicate that marking a frame as an event boundary is effective when it contributes to over 25% (\( \theta_2 = 0.25 \)) of the overall normalized motion, signifying event changes in the video. Fig. 5 illustrates the alignment of peaks with event boundaries, visually validating our approach. The third threshold \( \theta_3 \) indicates the distance between two boundaries to consider them as belonging to the same cluster during refinement. Specifically, it is the Euclidean distance to split observations into clusters. We set \( \theta_3 = 0.50 \), i.e., twice the unit timestamp (time per frame) considered in our algorithm (4 FPS \( \Rightarrow \) 1 unit = 1/4 = 0.25).

### 4.3. Ablation Studies

**Effect of Sampling.** Table 3 illustrates the result of two sampling techniques. In random sampling, we uniformly sample the fixed fraction of pixels from each patch. The corner detection [29] looks for a significant change in pixel intensity in all directions. This sometimes results in the sampling of fewer corner pixels. Thus, we observe that random sampling of pixels gives better F1 scores for PT. It may be noted that the sampling method does not apply to the FN since it computes dense optical flow.

**Effect of Spatial Granularity.** As detailed in Section 3.1.1 and 3.1.2, we conclude from Table 3 that computing patchwise offers higher performance than processing the entire frame as a single unit. Further, by introducing Centroidal patches, we can capture event change at the intersection of the base patches, leading to a noticeable increase in F1.

**Effect of Patch Size.** Besides the spatial granularity, patch size is essential to predict the accurate event boundaries, as illustrated in Fig. 3. We capture the effect of varying \( n_w \) in Table 4. A higher \( \mathcal{N}_g \) (indicative of a smaller patch size) results in more candidate boundary sets, reducing the likelihood of missing uncommon boundaries. Moreover, it helps effectively trace events in tiny regions. We have determined the ideal value for our approach to be \( n_w = n_h = 5 \). Processing beyond a specific patch size can introduce noisy boundaries to the candidate pool, lowering the F1.
spectrally. So, the overall time complexity is given by $O_Lg$.

4.5. Time Complexity of FlowGEBD

We theoretically assess the time complexity of FlowGEBD. From Algorithms 1, 2, 3, we observe that the time complexities (in patchwise mode) of pixel tracking, flow normalization, and temporal refinement are $O(LN_gw_gh_g|P_{\text{base}}|)$, $O(LN_gw_gh_g)$, and $O(M \log M) (\equiv O(L \log L))$, respectively. So, the overall time complexity is given by $O(LN_gw_gh_g|P_{\text{base}}| + LN_gw_gh_g + L \log L)$. We can further simplify this to $O(LN_gw_gh_g|P_{\text{base}}|)$. Since in pixel tracking, we use a sparse set of key pixels (i.e. $|P_{\text{base}}| \ll w_gh_g$), $|P_{\text{base}}|$ is a fraction of $w$ and $h$. Additionally, $N_gw_gh_g \propto wh$. So, we infer that the latency of FlowGEBD is directly proportional to $wh$ and $L$. Please consult the supplementary materials for the analysis of the inference time on sample videos.

Comparison of latency. Table 5 presents the latency per frame across different methods. Most of these methods employ ResNet-50 as their backbone, resulting in an average inference time of at least 46.4 ms per frame at a resolution of 160x160 on a GPU [10]. In contrast, PT and FN exhibit considerably lower inference time, consuming 2.26 ms and 6.42 ms, respectively. The ensemble approach takes 6.5 ms on average without compromising the F1 score. The reported inference time is measured on a Samsung Galaxy S21 Ultra device with 12 GB RAM. Furthermore, the estimation of optical flow can be accelerated on GPU by utilizing NVIDIA Optical Flow SDK [25].

5. Conclusion and Discussion

We introduce FlowGEBD, a non-parametric, unsupervised approach for generic event boundary detection. FlowGEBD comprises two independent algorithms, (i) Pixel Tracking and (ii) Flow Normalization, which can be deployed framewise or patchwise. FlowGEBD achieves the state-of-the-art results (Tables 1 and 2) on the Kinetics-GEBD and TAPOS at a strict relative distance (F1@0.05). This demonstrates that the motion information acquired from an optical flow alone is sufficient and obviates the need for complex neural models to achieve high performance. We performed an extensive ablation study and threshold sensitivity analysis to demonstrate the robustness of the proposed method. However, since FlowGEBD does not incorporate spatial semantics (high-level DNN features), it is more suitable for GEBD rather than specific action/event localization. The same effect is observed in the evaluation of TAPOS. In future work, we will explore the bi-directional processing of each frame to improve the performance.
References


[26] Rui Qian, Yeqing Li, Liangzhe Yuan, Boqing Gong, Ting Liu, Matthew Brown, Serge Belongie, Ming-Hsuan Yang, Hartwig Adam, and Yin Cui. Exploring temporal granular-


