






SUPPLEMENTARY MATERIAL:

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

A. Precision, Recall and F1

Tables 2 and 3 show precision, recall and F1 scores at different Relative Distance (Rel.Dis.) thresholds for FlowGEBD (PT, FN and Ensembled). Following the benchmark [2], we mainly consider 0.05 threshold for our analysis since boundaries with high Rel.Dis. is less relevant for short videos.

Table 2 shows that the precision values for FN and Ensembled methods are 0.6507 and 0.6289, respectively. In Ensembled method, we comprehend contiguous boundaries to belong to one cluster. For example, if boundary timestamps obtained from PT and FN (without temporal refinement) are: {1.25, 1.5, 1.75} and {2.0, 2.25, 2.50, 2.75}, respectively, then output for Ensembled approach would be a single median boundary {2.0}. Therefore, a marginal drop in precision is an outcome of possible decrease in true positives. However, recall boosts significantly to 0.8237. Fig. 1 depicts that 8% of the boundary causes are characterized by multiple coupled scenarios (example: *change of subject + action*). Therefore, by ensembling PT and FN methods, we remove some unnecessary boundaries and thus convert the false negatives to true positives.

The TAPOS dataset [1] contains Olympics sports videos with 21 actions. Both PT and FN yield at par performance on all three metrics, as illustrated in Table 3. This demonstrates the effectiveness of our proposed algorithms, since both PT and FN are able to detect action changes accurately. Additionally, 43% videos in TAPOS validation set have more than one boundary. Hence, we are able to appropriately detect uncommon events and improve true positives. Our interpretation is validated by the higher F1, precision and recall scores for the Ensembled method.

B. Latency Analysis of FlowGEBD

We sample 100 videos from the Kinetics-GEBD [2] validation dataset to verify the complexity and measure the in-

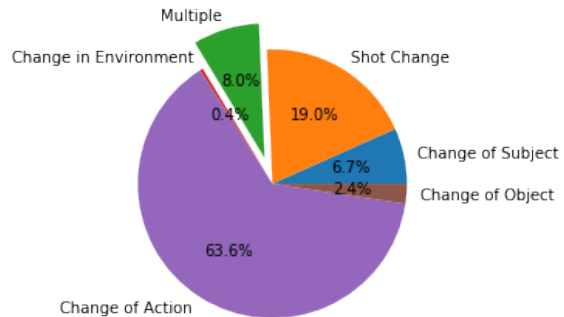


Figure 1. Boundary events distribution on Kinetics-GEBD validation set

ference time. In Table 1, the w (width), h (height) values are increased linearly by a factor of two. As anticipated, the inference time increases by $4\times$.

C. Qualitative Results

The qualitative results on Kinetics-GEBD are shown in Fig. 2. This illustrative representation provides a clear and comprehensive depiction of FlowGEBD’s accuracy in identifying event boundaries across different scenarios. The third row in the Fig. 2, vividly highlights the effectiveness of patchwise processing.

Frame Resolution	Average inference time per frame (ms)		
	Pixel Tracking	Flow Normalization	Ensembled
160 × 160	2.266	6.428	6.503
320 × 320	7.744	20.236	20.311
640 × 640	16.525	69.675	69.75

Table 1. Latency of FlowGEBD with respect to frame resolution (measured on Edge Device, CPU)

Metric	Rel.Dis. threshold	0.05	0.1	0.15	0.2	0.25	0.30	0.35	0.4	0.45	0.5	Avg
Precision	PT	0.6437	0.7593	0.7828	0.7929	0.7981	0.8006	0.8025	0.8041	0.8053	0.8069	0.7796
	FN	0.6507	0.7846	0.8178	0.8340	0.8411	0.8453	0.8479	0.8496	0.8514	0.8528	0.8175
	Ensembled	0.6289	0.7452	0.7684	0.7770	0.7811	0.7831	0.7843	0.7855	0.7863	0.7876	0.7627
Recall	PT	0.7724	0.8887	0.9145	0.9266	0.9326	0.9366	0.9393	0.9407	0.9426	0.9442	0.9138
	FN	0.7361	0.8729	0.9072	0.9242	0.9328	0.9376	0.9410	0.9433	0.9455	0.9469	0.9087
	Ensembled	0.8237	0.9315	0.9508	0.9586	0.9626	0.9646	0.9657	0.9666	0.9673	0.9683	0.9460
F1 Score	PT	0.7022	0.8189	0.8435	0.8546	0.8601	0.8633	0.8655	0.8671	0.8685	0.8702	0.8414
	FN	0.6908	0.8264	0.8602	0.8767	0.8846	0.8891	0.8920	0.8940	0.8960	0.8974	0.8607
	Ensembled	0.7133	0.8280	0.8499	0.8583	0.8624	0.8644	0.8656	0.8667	0.8675	0.8686	0.8445

Table 2. Precision, Recall and F1 results on Kinetics-GEVD validation set with different Rel.Dis. thresholds. (Detailed version of Table 1 from main paper)

Metric	Rel.Dis. threshold	0.05	0.1	0.15	0.2	0.25	0.30	0.35	0.4	0.45	0.5	Avg
Precision	PT	0.3141	0.4327	0.4974	0.5478	0.5793	0.5991	0.6132	0.6221	0.6318	0.6373	0.5475
	FN	0.3053	0.4299	0.4958	0.5468	0.5808	0.5983	0.6136	0.6232	0.6310	0.6378	0.5463
	Ensembled	0.3195	0.4279	0.4852	0.5323	0.5614	0.5774	0.5924	0.5999	0.6062	0.6116	0.5314
Recall	PT	0.4084	0.5627	0.6467	0.7123	0.7533	0.7790	0.7973	0.8089	0.8214	0.8287	0.7119
	FN	0.3986	0.5612	0.6472	0.7137	0.7581	0.7810	0.8010	0.8135	0.8236	0.8325	0.7130
	Ensembled	0.4525	0.6060	0.6872	0.7540	0.7952	0.8178	0.8390	0.8496	0.8586	0.8663	0.7526
F1 Score	PT	0.3551	0.4892	0.5623	0.6193	0.6549	0.6774	0.6933	0.7033	0.7142	0.7205	0.6190
	FN	0.3458	0.4869	0.5615	0.6192	0.6577	0.6775	0.6949	0.7058	0.7145	0.7223	0.6186
	Ensembled	0.3746	0.5016	0.5688	0.6241	0.6582	0.6769	0.6945	0.7032	0.7106	0.7170	0.6229

Table 3. Precision, Recall and F1 results on TAPOS validation set with different Rel.Dis. thresholds. (Detailed version of Table 2 from main paper)



Figure 2. Visualization of detected boundaries on the validation set of Kinetics-GEVD compared with ground truth [2]. The first and second rows show our predictions for change in action and change in environment. The third row demonstrates the effectiveness of patchwise processing, where only a tiny fraction of the frame changes (shown with blue color patches) while the rest remains static.

D. Pseudo code

For better reproducibility, we provide the pseudo code of the proposed algorithms 1 and 2 in the main paper.

Pseudo code: FlowGEBD

```
1 def FlowGEBD(videoFile):
2     input_video = ParseVideo(videoFile)
3     total_frames = len(input_video)
4
5     #Initialize required containers & variables
6     patch_flow = []
7     PT_boundaries = []
8     FN_boundaries = []
9     idx = 1
10
11    #Preprocessing
12    frame_prev = convertRGBToGrayscale(...)
13    #Using API COLOR_BGR2GRAY(...)
14
15    patches_prev = getPatches(frame_prev)
16
17    #Initial pixels for PT method
18    p0 = sampleRandomPixels(patches_prev)
19    initial_len = len(p0)
20
21    #Process the video frame-by-frame
22    while idx < total_frames:
23
24        frame_curr = convertRGBToGrayscale(...)
25        #Using API COLOR_BGR2GRAY(...)
26
27        patches_curr = getPatches(frame_curr)
28
29        #Both algorithms are processed patchwise
30        for i in range(num_patches):
31            #Get Boundary Status from PT
32            p1, status = getPixelTrackingStatus(
33                args)
34            if status is True:
35                #Mark the boundary
36                PT_boundaries[i].append(idx)
37                p0[i] = sampleRandomPixels(
38                    patches_curr)
39                initial_len[i] = len(p0[i])
40            else:
41                p0 = p1
42
43            #Record PatchFlow from
44            max_flow = computeDenseMaxFlow(args)
45            patch_flow[i].append(max_flow)
46
47        patches_prev = patches_curr
48        idx += 1
49
50    FN_boundaries = getFNBoundaries(patch_flow)
51    #Ensembling of both boundary sets (Algo. 3)
52    boundaries = refine(PT_boundaries,
53                        FN_boundaries)
54
55    def getPixelTrackingStatus(args):
56        prev_frame, curr_frame, p0, initial_len = args
57
58        p1, st, err = getSparseOpticalFlow(...)
```

```
58    #Using API calcOpticalFlowPyrLK(...)
59
60    fraction_of_pixels = len(p1) / initial_len
61
62    return p1, (fraction_of_pixels <  $\theta_1$ )
63
64    def computeDenseMaxFlow(args):
65        prev_frame, curr_frame, p0 = args
66
67        denseFlow = getDenseOpticalFlow(...)
68        #Using API calcOpticalFlowFarneback(...)
69
70        reducedFlow = max(denseFlow)
71
72        return reducedFlow
73
74    def getFNBoundaries(patchFlow):
75        boundaries = []
76        for i in range(num_patches):
77            flow = normalize(patchFlow[i])
78            indices = argWhere flow >  $\theta_2$ 
79            boundaries.append(indices)
80
81        return boundaries
```

References

- [1] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Intra-and inter-action understanding via temporal action parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 730–739, 2020. 1
- [2] Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. Generic event boundary detection: A benchmark for event segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8075–8084, 2021. 1, 2