Student:
Syma Afsha [symaafsha.eece@gmail.com]

Event-based Data and Motion Compensation

Questions and Answers

Clip a sequence of 0.2 s (10-10.2 s)

Convering an Event Image

A sequence of events, each described by a position (x, y), a timestamp, and a polarity, and converts them into a 2-channel image representation, where each channel corresponds to a different polarity of events.

Algorithm 1 Convert Events to Image

Require: $events$, an $N \times 4$ numpy array where each row represents an event in the form $(x, y, timestamp, polarity)$.
Require: $image\_size = (H, W)$, the dimensions of the output image, equal to the sensor size.
Ensure: $events\_image$, a 2-channel image of shape $(2, H, W)$ where each channel counts the number of positive or negative events at each pixel location.

Initialize $events\_image$ to zeros of shape $(2, H, W)$
for each event e in $events$ **do**
    Extract $x, y, \_, p$ from e
    $channel \leftarrow 0$ if $p > 0$ else 1
    Increment $events\_image[channel, y, x]$ by 1
end for
return $events\_image$ =0

Visualization of Event Data includes two visualization techniques:

- **Heatmap Visualization:** Visualizes the density of events in different areas for event-based camera, can be insightful for understanding the dynamics captured by the sensor.

Algorithm 2 Visualize Events Heatmap

Require: *events_image*, a 2-channel image of event counts.
Require: *polarity*, the polarity channel to visualize ('pos', 'neg', or 'both').
Ensure: Visualization of the event heatmap.

if *polarity* is 'pos' **then**
  $image \leftarrow events\_image[0, ...]$
else if *polarity* is 'neg' **then**
  $image \leftarrow events\_image[1, ...]$
else
  $image \leftarrow \text{sum}(events\_image, axis = 0)$
end if
Display *image* as a heatmap =0

- **RGB Visualization:** Displays positive and negative events in different colors on an RGB image.

Algorithm 3 Visualize Events Image RGB

Require: *events_image*, a 2-channel image of event counts.
Require: *polarity*, the polarity channel to visualize ('pos', 'neg', or 'both').
Ensure: *image*, an RGB image visualizing the events.

Initialize an all-white *image* of shape $(H, W, 3)$
if *polarity* is 'pos' or 'both' **then**
  Mark positions with positive events in red on *image*
end if
if *polarity* is 'neg' or 'both' **then**
  Mark positions with negative events in blue on *image*
end if
Display *image*
return *image* =0

Question 01

How clear is the obtained events image? Why? (Explain by showing your resulting images, and analyzing the motion of your chosen sequence).

Answer: For a short time sequence of 0.2 seconds, specifically from 10 to 10.2 seconds,the motion within the chosen sequence is analyzed by observing the change in patterns over time. Each pixel operates independently, capturing changes in light intensity which are translated into events.

The positive polarity heatmap Figure 1 shows areas where brightness has increased, highlighted by bright lines that trace the edges and movement within the scene. These lines suggest the motion of objects or changes in light intensity, and they contrast with the darker background, creating a distinct image of movement. The color intensity on the heatmap corresponds to the number of positive events, the brighter or more towards the yellow-end of the spectrum, the higher the count. The negative polarity heatmap image 2 shows where brightness or change in light intensity has decreased. Bright spots, particularly in yellow and red, indicate a higher density of negative events, which are areas where objects may have obstructed light, casting shadows, or where light sources have been blocked. The combined heatmap Figure 3 represents a comprehensive view of all events, though the overlap of positive and negative events result in a less sharp representation.

The red-colored RGB Figure 4 representation of positive events shows up brightly against the white background, making it easy to see where the increases in light intensity have occurred. The distribution and direction of the red pixels can be linked to the movement of objects or light sources in the image. In the same way, blue indicates Figure 5 places where intensity decreases in the RGB representation of negative events. Coherent blue pixel patterns or lines indicate the motion directions of objects or shadows in the scene. The combining red and blue pixels in the combined RGB image Figure 6, which represents both polarities, show the connection of light and shadow. The illustration is colorful, but how well it separates the two colors determines how easily one can differentiate between positive and negative events that happen. The individual movements could be difficult to see if there is a large overlap.

In conclusion, these images offer an in-depth representation of the positions and the time of light intensity changes, providing a clear representation of the scene's movement and light dynamics.

Question 01 for 1s sequence

Clip a sequence from 4.38-5.38s

How clear is the obtained events image? Why? (Explain by showing your resulting images, and analyzing the motion of your chosen sequence).

Answer: The duration of the time interval greatly affects how clear the images are when using an event-based camera to record movement.

Images with a higher event density are usually obtained from a 0.2-second sequence, which simplifies the observation and analysis of precise, well-defined motions. When analyzing fast, accurate movements, the directions of moving objects are easier to see and discern.

Images from a 1-second sequence, on the other hand, display events across an extended period of time, which may cause overlapping events to obscure specific motion routes. Because several

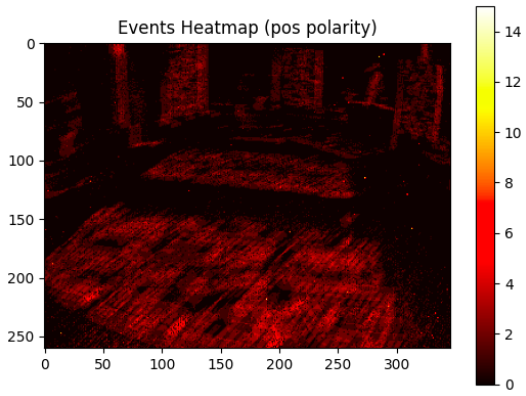

Figure 1: Positive polarity heatmap visualization

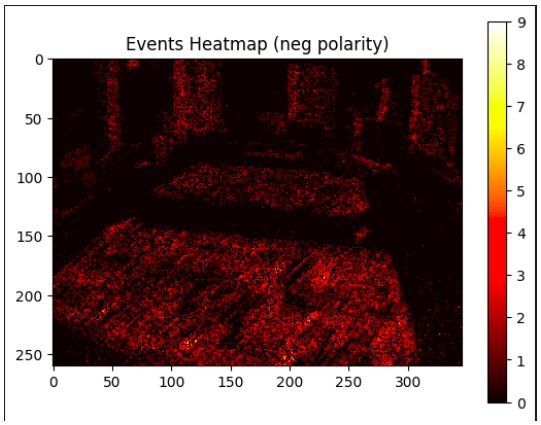

Figure 2: Negative polarity heatmap visualization

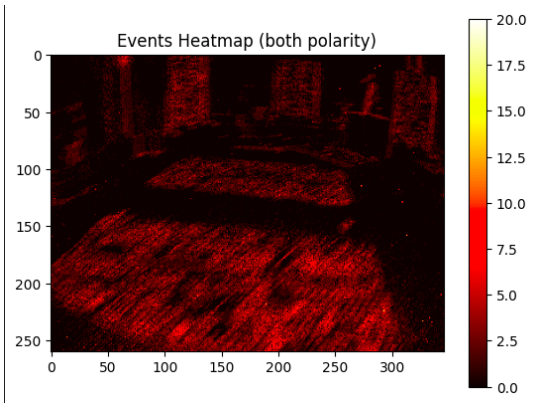

Figure 3: Combined polarity heatmap visualization

activities combine into a broader overview of motion, it could become more difficult to identify precise movements as a result. A lengthier sequence can give an improved understanding of the dynamics of the scene, but it also runs with the risk of including extra noise, which can make analysis more difficult.

Events tend to spread throughout time in the 1-second RGB images shown in Figure 10, 11, and 12, therefore they may appear less "clear" in terms of event density than the 0.2-second interval. Still, they capture the whole experience of both increases and decreases in light, and over a longer duration, they offer useful information about the scene's dynamics. Though it may also confuse several movements, making it more difficult to differentiate between simultaneous or quickly sequential activities,

The reduced density of bright spots shown in heatmap Figure 7, 8, and 9, could make it more challenging to track fast or brief changes, as events are less concentrated and more dispersed

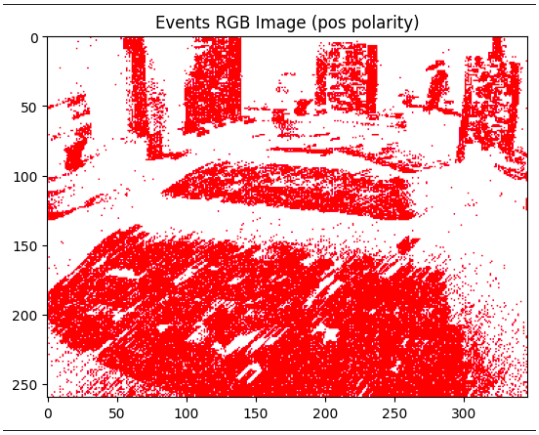

Figure 4: Positive polarity RGB visualization

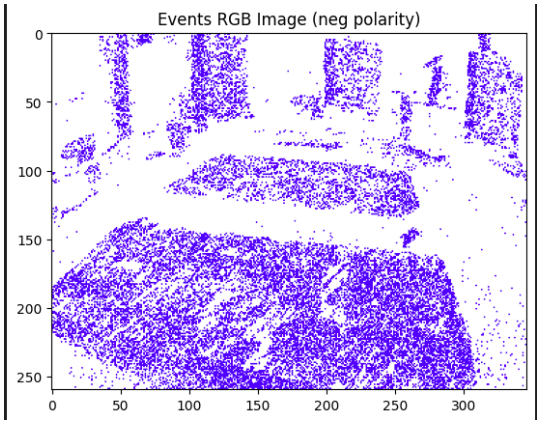

Figure 5: Negative polarity RGB visualization

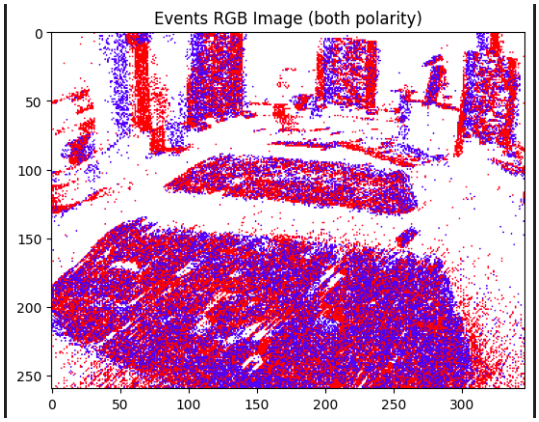

Figure 6: Combined polarity RGB visualization

across the image. Thus, for a clearer, more in-depth motion analysis, a shorter sequence, such as 0.2 seconds, is typically preferable.

Frame-based Approach

The objective of this approach is to create a sequence of events image frames before feeding them to the network.

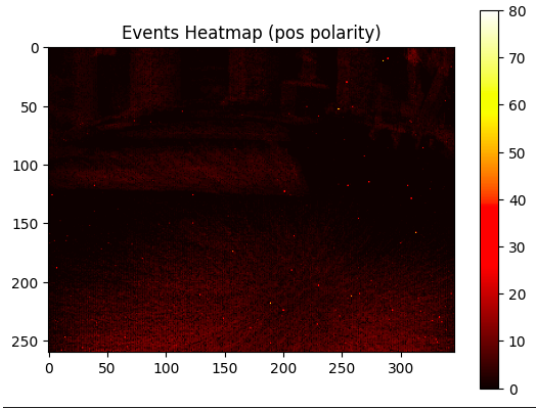

Figure 7: Visualization Heatmap (positive polarity)

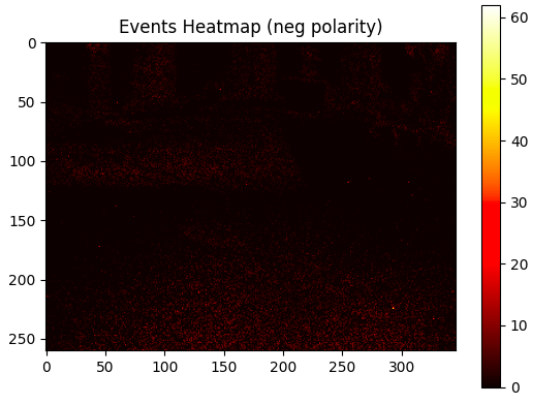

Figure 8: Visualization Heatmap (negative polarity)

Conversion of Events to Frames

Taking an input of events, each with a timestamp, position (x, y), and polarity p, and discretizing these events into a series of frames based on the timestamps.

Algorithm 4 Convert Events to Frames

Require: $events$, an $N \times 4$ numpy array where each row represents an event in the form $(x, y, timestamp, polarity)$.
Require: $image\_size = (width, height)$, the dimensions of the frame.
Require: num_frames, the number of frames to discretize the events into.
Ensure: $frames$, a 4D array of shape $(2, height, width, num\_frames)$ where the first dimension represents polarity (positive or negative), and each slice along the last dimension represents a frame.

$min\_timestamp \leftarrow \min(events[:, 2])$
$max\_timestamp \leftarrow \max(events[:, 2])$
$timestamps\_normalized \leftarrow \frac{events[:,2] - min\_timestamp}{max\_timestamp - min\_timestamp} \times num\_frames$
$frame\_indices \leftarrow \lfloor timestamps\_normalized \rfloor$
$frame\_indices \leftarrow \text{clip}(frame\_indices, 0, num\_frames - 1)$
Initialize $frames$ to zeros of shape $(2, height, width, num\_frames)$
for each event e in $events$ **do**
    $x, y, \_, p \leftarrow e$
    $frame\_index \leftarrow frame\_indices$[event index]
    $channel \leftarrow 0$ if $p > 0$ else 1
    $frames[channel, y, x, frame\_index] += 1$
end for
return $frames$ =0

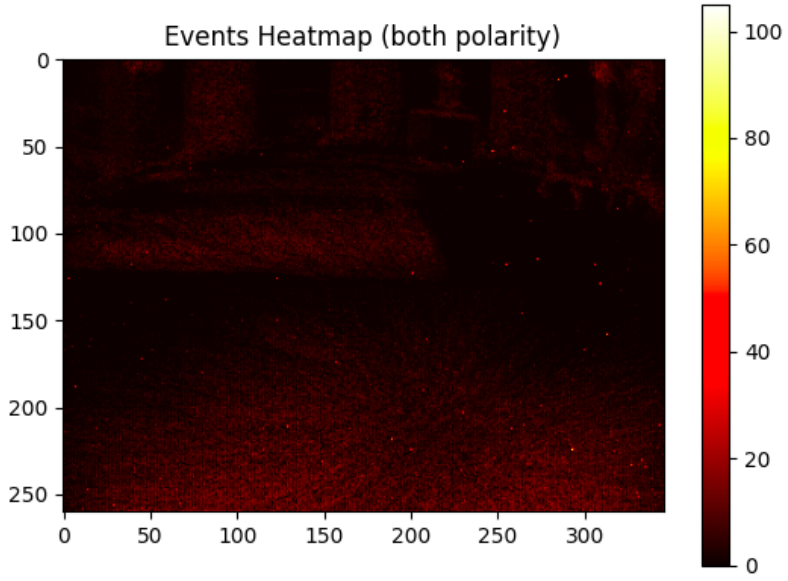

Figure 9: Visualization Heatmap (Combined Polarity)

Visualizing Combined Heatmaps

Creating heatmaps for each frame by summing up the events regardless of their polarity.

Algorithm 5 Visualize Combined Heatmaps

Require: $frames$, a 4D array of shape $(2, height, width, num\_frames)$.
Ensure: Visualization of combined heatmaps for all frames.
  for $i = 0$ to $num\_frames - 1$ **do**
    $combined\_frame \leftarrow \sum_{polarity} frames[:, :, :, i]$
    Display $combined\_frame$ as a heatmap with title "Frame $i + 1$"
  end for=0

Visualizing Combined RGB

Positive and negative events are visualized in different colors (red for positive, blue for negative) on a white background across all frames.

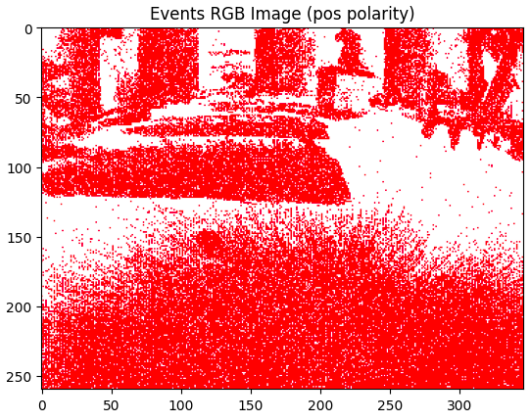

Figure 10: Visualization RGB (positive polarity)

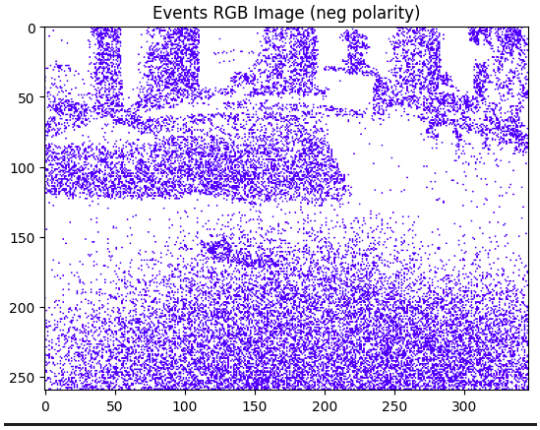

Figure 11: Visualization RGB (negative polarity)

Algorithm 6 Visualize Combined RGB

Require: $frames$, a 4D array of shape $(2, height, width, num\_frames)$.
Ensure: Visualization of combined "RGB" images for all frames.

1: **for** $i = 0$ to $num\_frames - 1$ **do**
2: $pos\_frame \leftarrow frames[0, :, :, i]$
3: $neg\_frame \leftarrow frames[1, :, :, i]$
4: Initialize rgb_frame to an all-white image of shape $(height, width, 3)$
5: Set pixels in rgb_frame where $pos\_frame > 0$ to red $(255, 0, 0)$
6: Set pixels in rgb_frame where $neg\_frame > 0$ to blue $(0, 0, 255)$
7: Display rgb_frame with title "Frame $i + 1$"
8: **end for**=0

Question 02

How do the images change after applying the frame-based approach? Why? (Explain by showing your resulting images, and analyzing the motion of your chosen sequence).

Answer: Event-based cameras capture changes in the scene's intensity for each pixel independently, which means they only provide data when the change in intensity is detected, leading to very sparse and noisy raw data.

When frame-based processing of event-based camera data is applied, which is shown in Figure 13 significant improvements in image quality and motion representation are observed. When events are combined into frames, it allows for a more familiar representation of the scene, which can be easier to interpret.

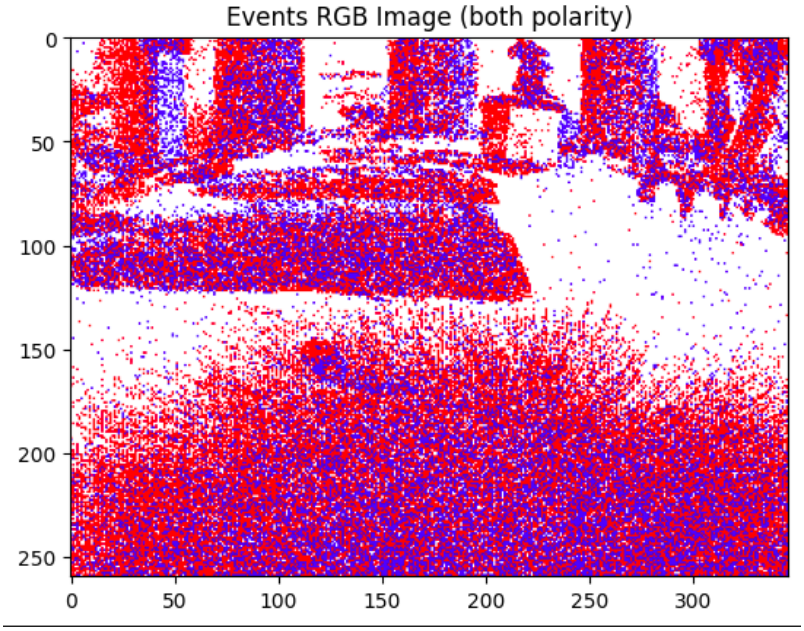

Figure 12: Visualization RGB (Combined Polarity)

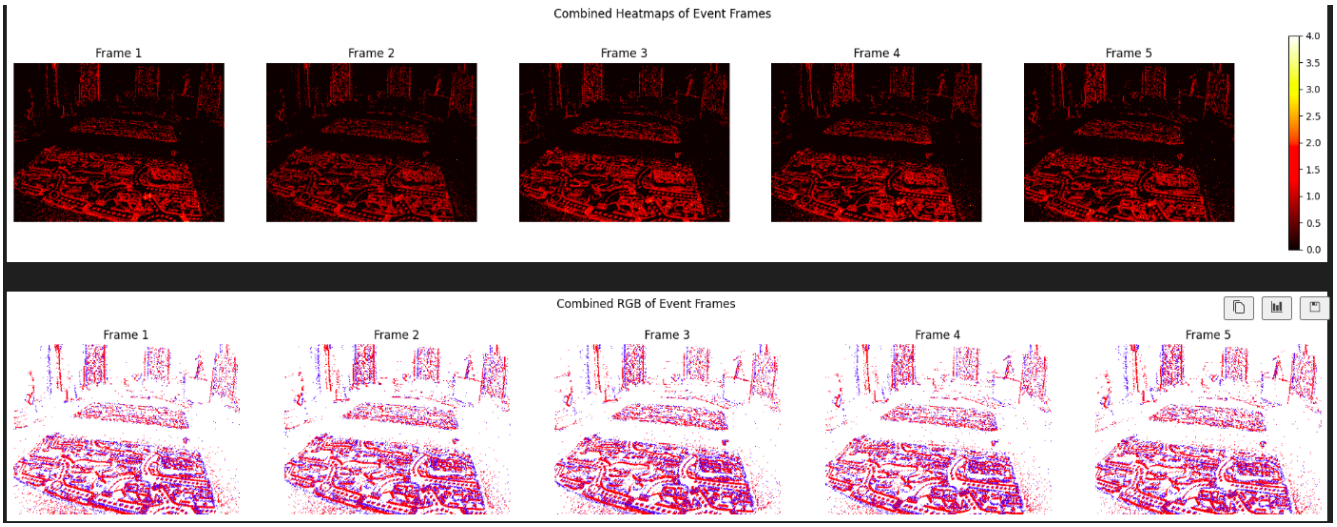

Figure 13: Combined Heatmap and RGB visualization of 5 frames

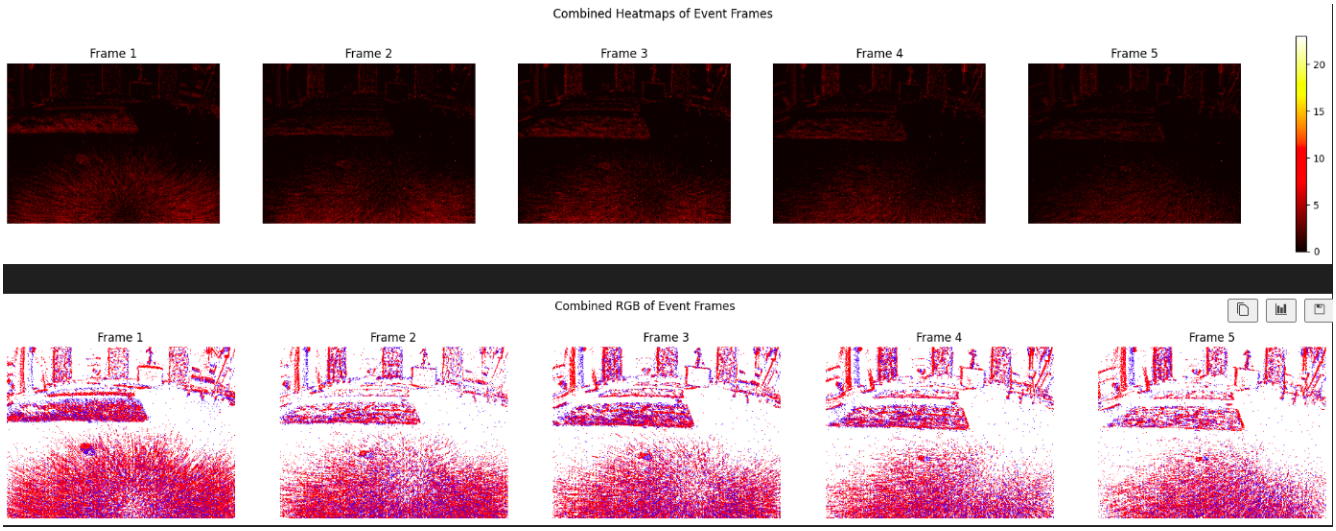

Figure 14: Combined Heatmap and RGB Visulization of 5 frames for 1s sequence

The frame-based method removes noise and random fluctuations from each individual event by combining multiple events over time. Due to their constantly shifting positions with respect to the camera, moving objects frequently provide a sequence of sequential events. When these events are combined over time, the movements of these objects become more visible, which simplifies motion analysis.

I take 5 number of frames instaed of 2 frames as a smaller number of frames with shorter temporal windows provides higher temporal resolution because they capture more detailed temporal changes, but they may contain less motion information per frame. Conversely, longer windows capture more extended motion over time, resulting in smoother representations of motion.

Question 02 for 1s sequence

How do the images change after applying the frame-based approach? Why? (Explain by showingyour resulting images, and analyzing the motion of your chosen sequence).

Answer: It is simpler to observe movements over time and identify trends because to the frame-based visualization's minimized noise capabilities. Consequently, this method provides an explanation of motion that is appropriate for analyzing the motion and movements of objects in the camera's range of vision.

Figure 14, the heatmaps show broader, less concentrated activity, which could be due to the accumulation of events over time. With more red and blue events overlapping, the RGB frames seem more intense, which could indicate more complex or long actions.

With a 0.2-second interval, the images are likely to be sharper with less overlap of events, enabling clearer tracking of motion paths. On the other hand, because events overlap over a longer length, the 1-second sequence has the potential to make it difficult to distinguish between individual movements, even though it may capture longer movement.

Question 03

For the same time duration, how would the clarity of the events image change if we increase the camera velocity for the same scene? Explain.

Answer: Increasing the camera's velocity for the same scene over a given time duration would affect the clarity of the event-based images for both 0.2-second and 1-second intervals. The higher camera velocity in both situations results in increased motion blur, a phenomenon that causes blurring in the image during exposure due to moving objects or the camera itself. The reason this occurs is that the camera records the same object in more than one location in a single frame or series of frames.

For a duration of 0.2 seconds, as the camera moves around the scene more quickly, more events triggered, which could result in more blurry images. While the events may get more densely packed, the individual object paths may become less distinct. There may be streaks of occurrences rather than concentrated areas, making it difficult to determine the specific direction of movement.

The increase in camera speed may produce an image that is even less clear and blurry during a one-second period, with the events spread over a larger area of the image. A high camera velocity would compound the blurring effect, making it difficult to discern between the movements of various moving objects because the events have been recorded over a longer period of time. It's possible for the motion trails to heavily overlap, producing a more scattered image where it's challenging to identify individual movements.

Thus, in both time durations, a higher camera velocity would increase the difficulty of precisely analyzing and interpreting movements within the scene. Objects or the scene change position too quickly, causing overlapping events that are hard to distinguish individually.Slower camera speeds generally provide clearer images by minimizing blur and keeping the event paths more distinct and easier to follow.

Question 04

In this lab, we only used event counts as encoding. Though this might seem a good idea, some information remains lost. Pinpoint which information is lost with such an encoding scheme and suggest alternatives.

Answer: In the lab, we investigated encoding strategies with event counts spanning periods of 0.2 seconds and 1 second, and discovered that, while this strategy reduces the complexity of event-based data, it also loses important features. More specifically, they only record the occurrence of an event without taking into account its intensity level, which is crucial for understanding the dynamics of the scene. As a result, the fine temporal resolution that captures precisely when each event happens is lost as the magnitude of intensity changes.

When using a shorter time duration (0.2s) for encoding event counts, quick movements within that timeframe might be missed, as events occurring very close together could get combined. On the other hand, with a longer time duration (1s), significant changes in motion may become blurred or smoothed out, making it challenging to distinguish complex motion patterns.

Using alternative techniques provides a deeper and more complex understanding of the scene, which is essential for applications requiring for accurate motion analysis or complex scene reconstruction.

Here some Alternatives to Event Count Encoding

Time Surface Encoding

- **Lab Implementation**: The lab focused on processing event data using event counts, where the number of events within a fixed time interval (0.2 seconds or 1 second) was recorded for each pixel location. Time surface encoding, which involves assigning timestamps to pixel locations, was not explicitly implemented in the lab.
- **Description**: Time surface encoding captures the precise timing of each event by assigning timestamps to pixel locations in an image. While it may not have been implemented in the lab, it is a common technique used in event-based vision research for preserving temporal information.

Event Timing Encoding

- **Lab Implementation**: Similar to event counts, the lab recorded the occurrence of events within fixed time intervals (0.2 seconds or 1 second) without capturing the exact timing of

each event occurrence at each pixel location. Event timing encoding, where the exact timing of events is recorded, was not explicitly implemented in the lab.

- **Description**: Event timing encoding preserves both spatial and temporal information by capturing the exact timing of each event occurrence at each pixel location. It allows for the reconstruction of event sequences with high temporal precision, which can be valuable for tasks requiring accurate timing analysis.

Polarity-Weighted Counts

- **Lab Implementation**: In the lab, polarity-weighted counts were not specifically implemented. Instead, event counts were recorded without distinguishing between positive and negative events.
- **Description**: Polarity-weighted counts involve separately accounting for positive and negative events when creating event-based images. While not implemented in the lab, this technique preserves polarity information, allowing for the detection of motion direction and intensity changes.

Spatial-Temporal Features

- **Lab Implementation**: The lab did not explicitly implement techniques for extracting spatial-temporal features from event data using methods such as spatiotemporal convolutions or recurrent neural networks.
- **Description**: Spatial-temporal features extraction involves capturing both spatial and temporal information simultaneously from event sequences. While not part of the lab implementation, this technique enables the learning of rich representations for event-driven perception tasks and facilitates the extraction of hierarchical features capturing spatial structures and temporal dynamics.

To provide a deeper understanding, Figure 15 illustrates how to make a difference using Polarity-weighted Counts, for instance.

In contrast to Figure 1, Figure 2, and Figure 3 heatmaps, this Polarity-weighted counts figure 15 takes into consideration the direction of changes in brightness, whereby positive changes increase the value of a pixel and negative changes decrease it. The image's color scale depicts the overall impact of the changes; the red areas denote minimal or balanced changes, while the brighter yellow parts indicate an increase of positive events over negative ones.

The polarity-weighted image encodes the direction of change, while the event counts heatmap does not. The polarity-weighted image makes it possible to identify the regions where brightness is

Algorithm 7 Convert events to a polarity-weighted image

events is an $N \times 4$ numpy array of events where each event is (x, y, t, p).

image_size is a tuple $(width, height)$ representing the size of the output image.

Returns a $width \times height$ numpy array representing the polarity-weighted image.

function ConvertEventsToPolarityWeightedImage($events, image\_size$)

  $polarity\_weighted\_image \leftarrow$ numpy.zeros($image\_size$, dtype=int32)

  for all $event \in events$ **do**

    $x, y, \_, p \leftarrow event.astype(int)$

    $polarity\_weighted\_image[y, x] \leftarrow polarity\_weighted\_image[y, x] + p$

  return $polarity\_weighted\_image$

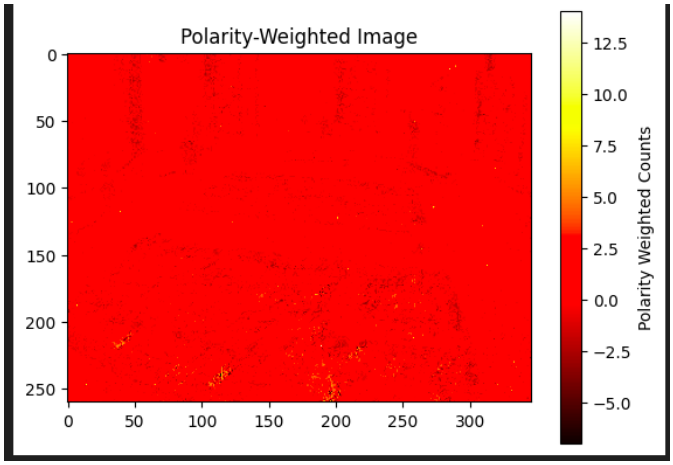

Figure 15: Polarity-Weighted Image

primarily increasing or decreasing. Moreover, it can offer greater visual contrast in regions where there is a significant imbalance in the direction of brightness change.

In summary, the polarity-weighted image reveals the major changes in a scene, allowing for a more detailed understanding of the dynamics. This is especially valuable in applications such as motion detection, where the direction of change is critical.