Probabilistic Estimation of Event-Based Normal Optical Flow (SB2-16)

WEN, Ruilin
Supervisor: Bert Shi
Probabilistic Estimation of Event-Based Normal Optical Flow (SB2-16)

WEN, Ruilin

Supervisor: Bert Shi

Overview
The event-based monoscopic camera offers another way of coping with the traditional optical flow problems. This new track of technology has been developed to bypass the computational limitations of conventional hardware. The essential idea of the camera is to emulate how the human visual system works. Each pixel on the camera, operating independently and asynchronously, is a primary functional unit whose mechanism works similar to the retina stimulus - when it stimulates, in this case the amount of charge in light intensity, matches a threshold, it outputs a spike of electronic signal. By collecting the spikes generated by the entire set of pixels and learning their spatial patterns, many of the conventional computer vision tasks can be achieved with much higher temporal resolution and significant reduction in latency. This is especially true when applied to optical flow tasks.

Figure 1. A spike occurs asynchronously when the light intensity changes at a certain pixel location reaches the threshold.

In this case, our project proposes a new event-based optical flow method to estimate the normal flow of moving edges. In probabilistic manner together with the way it represents the flow vector (a term of time-gaps determines that it is robust compared to other existing algorithms. The algorithm is also fast because it updates the flow vectors at every incoming spike, allowing for the calculation of flow vectors at any instant during motion.

Objectives
• Flow estimation of proposed algorithm in Matlab to test its feasibility
• Fine-tune parameters to achieve maximum performance
• Evaluate the algorithm performance by comparing it with existing event-based algorithms

Algorithm Mechanism
1. Initialization phase
The estimator is initialized with a few essential parameters. The right choice of which strongly affects the performance of its estimates. Each pixel location holds a stream of data about the stimulus, and the right order of the flow vector is determined by the size of the vector. This process is repeated until it reaches the threshold. The estimator then uses the data file which contains a stream of events, or spikes. This leads to the belief update phase.

Figure 2. The belief array at each pixel location

2. Belief Update phase
a) During this phase, the estimator carries out updates recursively for every incoming spike, which are fed from an FPGA event buffer first. The information contained in a single spike is simply its location denoted by (x, y) coordinate as well as the polarity indicating an incremental or decremental change in intensity of the event.

To avoid the belief being accumulated too high, we decay the belief by an exponential function, speed of which determined by a parameter β:

β = 1 - e^(-time)

The belief is then propagated through a normally distributed probability density function in terms of the time gap. After a weighted least squares (WLS) calculation, we obtain two values, A and B, which represent the flow it takes from the current pixel to the neighboring pixel along the x and y axes, respectively (Figure 3).

Figure 3. Initial flow after different time-gaps

3. Flow Vector Rendering phase

Figure 4. Speed calculation illustration

\[ V_x = \sqrt{A^2 + B^2} \]

\[ V_y = B \cdot A \]

We then obtain the x and y component of the normal flow vector. Consequently, the flow points indexing groups a number of spikes periodically to generate the normal flow chart.

Evaluation & Results
The metric we use here is to compare our performance with others is called the Relative Average Endpoint Error (RAEE), which essentially calculates the average deviation of the estimated vector from the ground-truth vector.

In the formula shown below, \((x', y')\) stands for the k-th sample of the ground flow and \((x, y)\) stands for the corresponding ground-truth vector. The sum of all the relative deviations are averaged by the number of spikes in total. The lower the RAEE, the higher the accuracy of the algorithm.

\[ RAEE = \frac{1}{N} \sum_{k=1}^{N} \sqrt{(x_k - x_k')^2 + (y_k - y_k')^2} \]

We compare three sets of data samples (results show in Figure 5), two of which are synthetically generated and the third captured from real camera-based hardware. As for the flow plots shown in Figure 4A, the result of our algorithm is plotted as the top-left corner.

Conclusion
In this project we proposed a new event-based optical flow algorithm. Tests on evaluation datasets show that the performance surpass existing counterparts. However, since there are large amounts of uncertainties that have not been evaluated, further testing should be conducted. Moreover, in order to testing an authentic hardware with real-time live data, future work shall focus on the implementation of the algorithm in actual programming languages. Moving the algorithm onto the AIR platform, which is supported by a community of researchers on event-based methods utilizes the Java language to develop algorithms, can be a promising next step for this project.