공학박사학위논문

Robust Event-based Angular Velocity Estimation in Dynamic Environments

동적 환경에 강건한 이벤트 카메라 기반 각속도 추정 알고리즘 개발

2022년 **8**월

서울대학교 대학원

기계항공공학부

이 상 일

Robust Event-based Angular Velocity Estimation in Dynamic Environments

동적 환경에 강건한 이벤트 카메라 기반 각속도 추정 알고리즘 개발

지도교수 김 현 진

이 논문을 공학박사 학위논문으로 제출함

2022년 **5**월

서울대학교 대학원

기계항공공학부

이 상 일

이상일의 공학박사 학위논문을 인준함

2022년 **6**월

위 원 장 : 김 유 단

부위원장 : 김 현 진

위 원 : 박 찬 국

위 원 : 김 아 영

위 원 : 김 표 진

Robust Event-based Angular Velocity Estimation in Dynamic Environments

A Dissertation

by

Lee, Sangil

Presented to the Faculty of the Graduate School of

Seoul National University

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

August 2022

Robust Event-based Angular Velocity Estimation in Dynamic Environments

Lee, Sangil

Department of Mechanical and Aerospace Engineering

Seoul National University

APPROVED:

Youdan Kim

Youdan Kim, Chair, Ph.D.

H. Jin Kim

H. Jin Kim, Ph.D.

Chan Gook Park

Chan Gook Park, Ph.D.

Ayoung Kim

Ayoung Kim, Ph.D.

Pyojin Kim

Pyojin Kim, Ph.D.

This dissertation is dedicated to:
my family who has shown unwavering faith in
and devotedly supported me throughout the process,
my professor who has introduced event cameras
and guided me enthusiastically,
and all the past and the current members
of the Lab for Autonomous Robotics Research
who have built friendships with me
and been a source of inspiration.

Abstract

Robust Event-based Angular Velocity Estimation in Dynamic Environments

Lee, Sangil
Department of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

This dissertation addresses the problem of estimating the angular velocity of the event camera with robustness to a dynamic environment where moving objects exist. These vision-based navigation problems have been mainly dealt with in frame-based cameras. The traditional frame-based cameras such as monocular or RGB-D image sensors capture the whole frame of absolute intensity and/or depth, thus making them easy to recognize temporary environments. However, even under the common illumination conditions, these sensors require a certain amount of time to collect light during which data is not output, thus latency occurs. Also, a video with a high dynamic range, e.g., a 14-bit raw image, is not usually used for computer vision due to its extremely high data bandwidth, and the general 8-bit image sequences easily lose their intensity data under overexposure or underexposure environments. Contrary to the conventional cameras that produce frames, events cameras operate asynchronously by imitating the human eye. Event cameras respond to the intensity changes in the temporal domain and generate an event that is triggered at the pixel whose intensity has changed. Due to the nature of the event cameras, they output data stream with low latency and high time resolution in us units. Besides, the event cameras only perceive relative intensity, they can have a higher dynamic range of more than 120 dB, whereas the standard cameras have 60 dB approximately. I take these advantages of the event cameras to detect moving objects and estimate the ego-motion of the camera in dynamic environments.

The first part of the dissertation focuses on asynchronously estimating optical flow streams with low latency and robustness to various scenes. Due to the fundamental difference between

traditional and event cameras, most existing algorithms construct event frames by stacking the timestamp value of many events and exploit the legacy of traditional computer vision algorithms. However, this approach increases latency in proportion to the size of a time window, and the size has to be set heuristically. I estimate an optical flow stream with very low latency by enhancing the existing block matching algorithm. The locally estimated optical flow is more accurate than that of the method using a global event frame, in front of irregularly textured scenes. To validate the latency of optical flow, I present the result of angular velocity estimation by using the proposed optical flow stream. Then, the latency is computed by the optimization approach comparing the estimated and ground-truth angular velocity. The evaluation results suggest that the proposed optical flow has very low latency while showing comparable accuracy to event-frame-based algorithms. Besides, the performance of angular velocity estimation is superior to the other existing algorithms in terms of accuracy and robustness with low latency under 15 ms consistently.

The second part of the dissertation proposes an angular velocity estimation with motion segmentation. Unlike traditional cameras, since event cameras detect intensity changes, their event data can be dominated by a small but fast-moving object. To eliminate the influence from the movement of an undesirable object, I utilize the optical flow stream of the first work and intra-pixel-area method and separate an image frame into the static and dynamic regions. Moreover, since event cameras do not produce events at stationary, a classification model should be addressed in the temporal domain and be able to segment motion temporally. Thus, I employ the dual-mode motion model to update models that determine the region occupied by moving objects. Then, the angular velocity of ego-motion is estimated from a bunch of optical flows belonging to the static region. The evaluation results suggest that the proposed algorithm divides the image frame into static and dynamic parts successfully and estimates the angular velocity robustly in dynamic environments.

Keywords: Event camera, dynamic environment, motion segmentation, angular velocity, intra-pixel-area event, optical flow, robustness

Student Number: 2017-32808

Table of Contents

List of Tables

List of Figures

Abbreviations and Acronyms

2D	2-Dimensional
3D	3-Dimensional
ABMOF	Adaptive time-slice Block-Matching Optical Flow algorithm
ANN	Artificial Neural Network
APS	Active Pixel Sensor
ATIS	Asynchronous Time-based Image Sensor
CDM	Closest Distance Metric
CM	Contrast Maximization
CMOS	Complementary Metal–Oxide–Semiconductor
CNN	Convolutional Neural Network
DAVIS	Dynamic and Active-pixel VIsion Sensor
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DoF	Degree-of-Freedom
DSI	Disparity Space Image
DVS	Dynamic Vision Sensor
EMSMC	Event-based Motion Segmentation by Motion Compensation
eSNN	event-based SNN for angular velocity regression
EV-FN	EV-FlowNet
fps	frames per second
G-AER	Group Address-Event-Representation
IMU	Inertial Measurement Unit
IPA	Intra-Pixel-Area
IQR	InterQuartile Range
IWE	Image of Warped Events

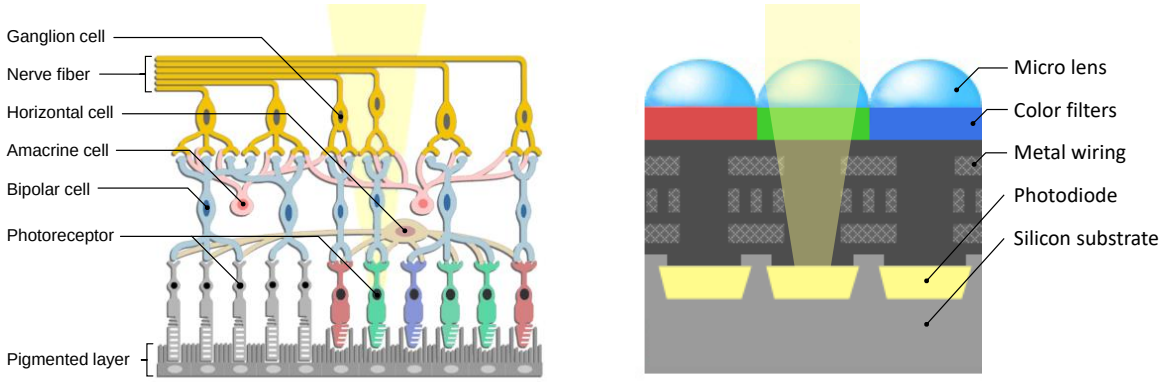
jAER	java tools for Address-Event Representation
LGN	Lateral Geniculate Nucleus
LiDAR	Light Detection And Ranging
Mbps	Megabits per second
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MVSEC	Multi Vehicle Stereo Event Camera dataset
NMOS	N-Channel MOSFET
PMOS	P-Channel MOSFET
RANSAC	RANdom SAmple Consensus
RGB-D	Red, Green, Blue, and Depth
RGC	Retinal Ganglion Cell
RMSE	Root Mean Square Error
SAE	Surface of Active Events
SLAM	Simultaneous Localization And Mapping
SNN	Spiking Neural Network
SOFAS	Simultaneous Optical Flow And Segmentation
SONAR	SOund Navigation And Ranging
Spike-FN	Spike-FlowNet
VO	Visual Odometry

1

Introduction

Recently, the role and market of robotic systems are expanded from the manufacturing environment to cover individual purposes. In order for a robot to interact with humans and perform assigned tasks in various and unknown environments, sensing systems for the robot are required. Existing sensors in robot systems are mainly inspired by human senses: vision (infrared, CMOS cameras), hearing (microphone, SONAR, gyroscope), touch (force sensor, touch screen), smell (gas sensor), and taste (water pollution sensors), etc. Among these sensors, a vision system is very important for a mobile robot system to recognize the relationship between itself and the environment. Most of their vision sensors construct 2D images (infrared, RGB/monocular) or 3D point cloud (LiDAR, depth sensor). Particularly, RGB/monocular cameras are widely used in robotic systems because of their versatility and similarity to the vision system of humans; Fig. 1.1 shows the structure of the biological vision system and the electrical image sensor.

In each figure, the light intensity is received in the photoreceptor and photodiode, and converted to the interpretable signal. In our eyes, the light reflected from an object passes through the cornea, pupil, lens, and vitreous body, then focuses an image onto the retina, which has three major cell components: photoreceptor, bipolar cells, and ganglion cells. The retina converts the



(a) (b)

Figure 1.1: Anatomy of vision systems: (a) biological retina of human and (b) the conventional Front-Side-Illuminated (FSI) image sensor. These figures are modified and used inspired by [1,2].

light intensity into electrical impulses and transmits the signal into the brain through nerve fibers. Photoreceptor cells are composed of rod cells and cone cells. The rod cells are more sensitive to light than cone cells, thus they work better in low illumination environments. On the other hand, the cone cells work better in bright environments, and they respond to the wavelength of light depending on the type of cone cells, S-cone (blue), M-cone (green), L-cone (red).

Like the three types of cone cells, the conventional RGB cameras have a color filter in front of photodiode sensors. Each pixels output the intensity of each color, and the processor generates an image of full resolution by using 2D interpolation. However, the limitation of the traditional cameras comes from their synchronous system. Their pixels are exposed to the light during the same exposure time and form a frame. The general frame-based vision sensor with 30 fps (frame per second) not only loses the detailed information of moving objects, so-called motion blur, but also transmits redundancy data for static scenes. When the frame rate of video is increased by 30 times to detect moving object with high temporal resolution in ms units, the bandwidth on a digital transmission is up to 345 Mbps (Megabits per second) from 240×180 pixels at 1000 fps, for reference, theoretical transmit rate of USB 2.0 is 480 Mbps.

To overcome the limitation of conventional cameras, respond to moving object, and reduce redundant data transmission, event cameras, also known as silicon retina, neuromorphic cam-

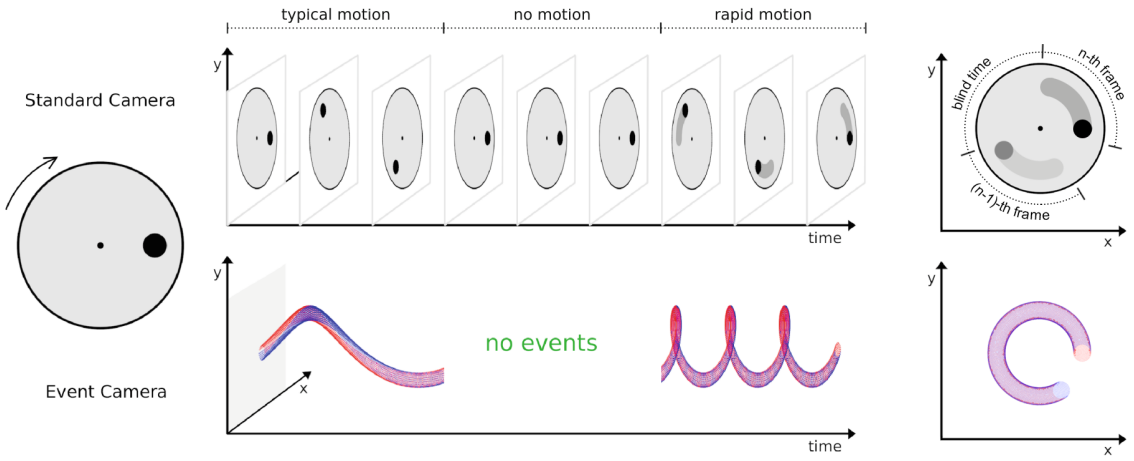


Figure 1.2: Comparison between the output of an event camera and a frame-based sensors. This figure is adopted from [3] (https://youtu.be/LauQ6LWTkxM?t=35s) and modified inspired by [4]. Frames with a fixed frame rate are represented in the upper, and event streams (positive in blue and negative in red) are shown in the lower. On the right, the upper image depicts two overlaid frames and the lower shows the stacked events during some interval in the xy-plane.

eras, or dynamic vision sensors, was first introduced in [20, 21] and practically proposed in [10–12, 15, 16]. These kinds of sensors are motivated by the asynchronism of human eyes. Similar to the photoreceptor of the human retina that responds to the light and converts the light into a nerve signal, the photodiode of the silicon retina senses the light change and transmits an event. The event cameras have the following features in comparison with conventional cameras: low-latency, high time resolution, and wide dynamic range. An asynchronous output of the event stream makes the vision sensor perceive a scene with low latency and high time resolution. Contrary to conventional cameras which have a fixed frame rate, the pixel of event cameras triggers a brightness change and transmits events individually and independently. Additionally, since pixels do not wait for a global exposure time and operate on a logarithmic scale of light intensity, event cameras have a wider dynamic range (> 120 dB) than frame-based cameras (~ 60 dB).

Fig. 1.2 describes the output of a frame-based camera and an event camera. The frame-based camera produces the synchronized dense image for each frame with a fixed frame rate and shows

motion blur and blind time because of the frame rate and global exposure time. On the contrary, the event-based camera produces the sparse stream of events asynchronously without typical motion blur and blind time. Due to the fundamental difference from frame-based cameras, the event stream cannot be directly exploited for the traditional computer vision techniques such as feature detection, tracking, motion segmentation, and estimation.

When it comes to implementing event-based computer vision algorithms, the methods can be categorized as an event-by-event basis or event-frame-based approaches depending on whether a single event or multiple events are processed at the same time. Some filter-based or network-based algorithms are implemented on an event-by-event basis, and event-frame-based algorithms construct event frames by stacking the timestamp of events according to the time window or the number of events. In addition, depending on how events are processed, there are two approaches: model-based and model-free [22]. The model-based approach uses intensity formulas or motion models to compute feature, optical flow or motion, etc. Meanwhile, model-free approach process events as an input of an artificial neural network.

In the dissertation, I exploit the event-by-event basis and model-based approach, and propose an optical flow estimation algorithm with low latency and robustness to various scenes. Compared with the existing event-frame-based algorithms which estimate the optical flow using event group, the proposed algorithm produces an asynchronous optical flow stream with very low latency while maintaining similar accuracy. In particular, the estimated angular velocity shows better performance than state-of-the-art algorithms in terms of accuracy, latency, and robustness to various scenes. Moreover, an algorithm that estimates the angular velocity component of the ego-motion is proposed by exploiting the event-based optical flow stream. The angular velocity estimation complemented by the proposed motion segmentation algorithm can estimate the ego-motion in a dynamic environment where moving objects such as vehicles exist.

1.1 Literature Survey

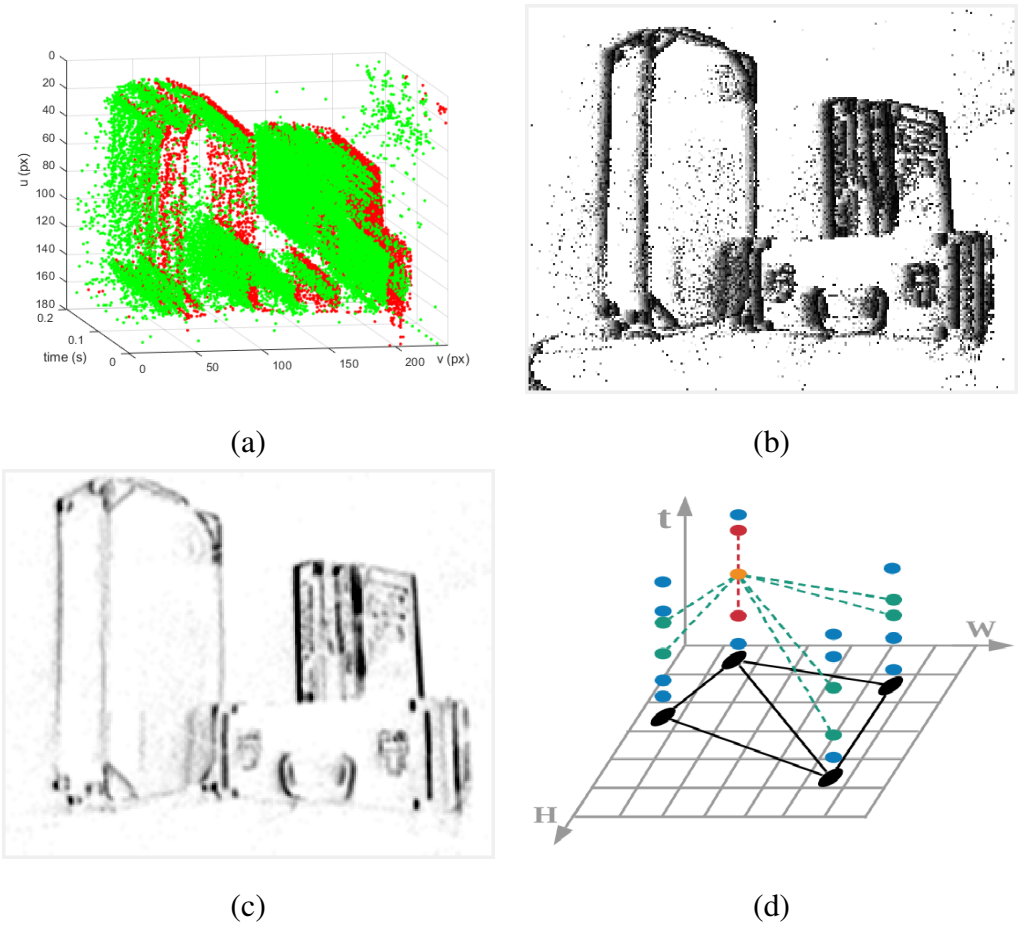
1.1.1 Event Representation

Event cameras produce asynchronous events which are frame-less information whereas conventional cameras such as Charge-Coupled Device (CCD), Complementary Metal–Oxide Semiconductor (CMOS), depth camera output image in the form of a frame. Since a single event has little information, it is difficult to exploit the traditional computer vision algorithms by using these streams. Thus, to adopt the asynchronous frame-less event, many approaches have tried to transform events into alternative representations and fetched the processed event data. In the following, I briefly review several event representations.

Event stream: Some research to maintain the high temporal resolution of event cameras update their algorithm for each incoming event, $e_k = (\mathbf{x}_k, p_k, t_k)$. Because a single event has little information and implies both spatial motion and temporal change, it is difficult to interpret a single event. In order to process event stream one by one, probabilistic filters [4,23,24] or Spiking Neural Network (SNN) [25, 26] are widely chosen.

Event packet: To infer a common temporal motion or spatial connectivity between events, event packet $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ are fetched and processed at once [27–30]. The size of the event packet N_e is set heuristically depending on the applications and scenes. A rich-textured scene demands a large number of events, whereas a small number of events are needed in a poor-textured scene. Also, it has to be determined small enough if the algorithm assumes that a motion is constant during time intervals.

Image of stacked events: By stacking the timestamp of event packet, Surface of Active Events (SAE), also known as time slice or time image, is constructed. SAE Σ_e is an image of the same size with the frame resolution of the sensor and is composed of the latest timestamps of



(a) (b)

(c) (d)

Figure 1.3: Examples of event representations in usage: (a) Event stream with colored polarity (positive in green and negative in red), (b) SAE with denoting the latest event as darker, (c) IWE with denoting the higher contrast as darker, (d) Spatio-temporal graph [5]. (a)-(c) are generated by the same event group on the DVSFlow16 sequence [6].

events in the corresponding pixel:

$$\Sigma_e : \mathbb{N}^2 \to \mathbb{R} \tag{1.1}$$
$$\mathbf{x}_k \mapsto \Sigma_e(\mathbf{x}_k) = t_k.$$

Depending on the usage, SAEs for positive or negative events are updated separately [31]. Since SAE includes timestamp and implies local visual flow, there are many literature exploiting SAE [19, 32–34].

Motion-compensated image of events: Another 2D representation of event packet is an Image of Warped Events (IWE) [28, 35–37] and event-count image [38]:

$$I(\mathbf{x};\theta) = \sum_{k=1}^{N_e} b_k \delta(\mathbf{x} - \mathbf{x}'_k(\theta)), \tag{1.2}$$

where b_k can be p_k or 1 whether the polarity is used or not, δ is Dirac delta function, and $\mathbf{x}'_k(\theta)$ is the pixel position of event warped by motion parameter θ. IWE was proposed to describe event trajectory along the time axis and find constant motion during time window. In practice, the Dirac delta function can be replaced by the normal probability density function with zero mean [28, 39], or unit sample function $\delta[\mathbf{x} - \mathbf{x}'_k(\theta)]$ with discretization grid size smaller than 1 for precise resolution [38].

Spatio-temporal points: To utilize the temporal relationship between event stream, 3D representation also presented in (x, y, t) space [5, 40, 41]. A. Mitrokhin et al. [40] is developed based on PointNet++ [42] for their graph convolutional network. In [5], they proposes a spatio-temporal graph to segment moving object with smooth boundaries. [43–45] use voxel grid to construct the Disparity Space Image (DSI) volume and estimate semi-dense depthmap.

1.1.2 Event Processing

To cope with an asynchronous and sparse event stream, there are mainly two categories. These categories are divided into 1) event-by-event basis and 2) event group, whether how many events are required and processed for an algorithm.

Event-by-event basis algorithms operate for each incoming event. These types of algorithms fetch a single event and produce outputs with a high temporal resolution, which is the one of advantages of event cameras. However, events are derived from both the spatial and temporal contrast, thus the system of the algorithm needs information of the past events. Therefore, to deal with a single event, most research implements statistical-filtering-based [4, 23, 24] or artificial-neural-network-based approach [25, 26, 46]. A statistical filter fetches an event and compares event occurrence frequency of events [24] or intensity changes [23, 47] with event generation

model. Then, they update the filter state by the innovation caused by a mismatch of the model and measurement.

Another plenty of research on an event-by-event basis utilizes the Artificial Neural Network (ANN). The structure of ANN is motivated by the nerve network of the human, each neuron of the network fetches an input value or the output of the previous neuron and produces a signal to the next neuron through their action mechanism. Among ANN, an SNN processes a single event and activates neurons only related to the input, thus reducing computational loads. In [26], they regress the angular velocity with comparable accuracy to ANN-based algorithm. Also, C. Lee et al. [25] introduce hybrid architecture composed of SNN and ANN to estimate dense flowmap while resolving the spike vanishing issue.

Event group forms various event representations above depending on the purpose of research. Because a single event carries little information, implies both spatial and temporal contrast, and is vulnerable to noise such as thermal noise, processing multiple events simultaneously can increase the signal-to-noise ratio and yield reliable results. Whether the group of events forms 2D or 3D representations, events are grouped according to their timestamp or number.

The former grouping method [26,48,49] is similar to the conventional cameras that have fixed time intervals. The algorithm forming event frame with frame rate operates periodically, but its computational loads vary depending on the motion since fast motion induces lots of events. Also, since slow motion generates not too enough events to describe the scene, the constructed event frame has few valid data and yields unreliable results. On the contrary, the event group of a large time interval describes the motion of events well [40], but the event trajectories of an edge can be overwritten by another edge in 2D representations [34] and constant motion assumption during time intervals is hard to be held.

On the other hand, the latter grouping method [5, 27, 36, 50, 51] is more suitable for the data-driven sensor. This type of algorithm operates frequently under a fast motion or rich-textured scene, whereas produces output rarely at stationary cameras, not yielding redundant results. However, in front of the over-rich-textured scene, it is difficult to estimate reliable motion with an insufficient number of events. Thus, the number of events in the group varied depending on the

texture level of the scenes.

Besides, the common limitation of the above grouping method is that a non-uniform texture of the scene is not considered. In that situation, some region may suffer from the redundancy input causing edge bleeding, whereas other region lacks events to describe the scene. Some research proposes a method that adjusts a time window [52] or the number of events [30, 50] not to set the parameter heuristically, but this adjustment does not resolve the non-uniformed generation of events in the image frame. Overall, the tuning parameter for grouping events is critical to the performance of the algorithm.

1.2 Contributions and Outline

The dissertation aims to estimate angular velocity robustly in dynamic environments. The proposed algorithm updates the motion parameter of the static environment while taking advantages of event camera such as low latency and high temporal resolution. To improve the robustness of the algorithm, I propose an intra-pixel-area event that is robust to pixel noise, and its applications such as visual flow, lifetime estimation, and edge detection are shown in Chapter 3. Next, Chapter 4 estimates a scene-robust optical flow stream while satisfying low-latency characteristic. The estimated asynchronous optical flow stream is validated on publicly available datasets in terms of accuracy and robustness to various scenes. Moreover, the proposed optical flow stream leads to an accurate angular velocity estimation with low latency and high temporal resolution. Finally, in Chapter 5, I segment different motions of static backgrounds and dynamic foregrounds by using the legacy of Chapters 3 and 4, which are used to construct a grid-based flowmap. Then, the proposed dual-mode motion models estimate the angular velocity of the ego-motion. The contributions of this dissertation are summarized as follows.

Chapter 3: Intra-pixel-area Events and Applications

- **Intra-pixel-area events**: I propose an intra-pixel-area event inside a pixel area on an image plane. In this chapter, the intra-pixel-area event is presented as a new threshold for the plane fitting methods.
- **Robustness to pixel noise**: A fitting plane method with intra-pixel-area event estimates a visual flow robustly and the lifetime of the event precisely.
- **Sharp edge map estimation**: The visual flow and lifetime of event are used to estimate the edge map. The proposed edge map detection performs better than the existing algorithm in terms of similarity metric.

Chapter 4: Low-Latency and Scene-Robust Asynchronous Optical Flow Stream Estimation

- **Optical flow estimation with low latency**: The proposed optical flow estimation algorithm computes local optical flow using local time slices, thus it produces optical flow stream with very low latency while showing comparable accuracy to the existing event-frame-based optical flow estimation algorithms.

- **Robustness to motion speed and texture level of scene**: I evaluate the algorithm in terms of latency and accuracy on scenes with various motion speed and texture levels. The evaluation suggests that the proposed algorithm shows low latency within 15 ms consistently and high accuracy in comparison with the existing angular velocity estimation algorithms.

Chapter 5: Robust Angular Velocity Estimation in Dynamic Environments

- **Dual-mode motion model**: The proposed dual-mode motion model for event camera segments static background and moving objects in an image frame without prior information about the number, shape of objects.

- **Robustness to moving objects**: The proposed algorithm accurately and robustly estimates the angular velocity of the camera itself in the collected datasets, which include various situations such as moving vehicles, stationary cameras.

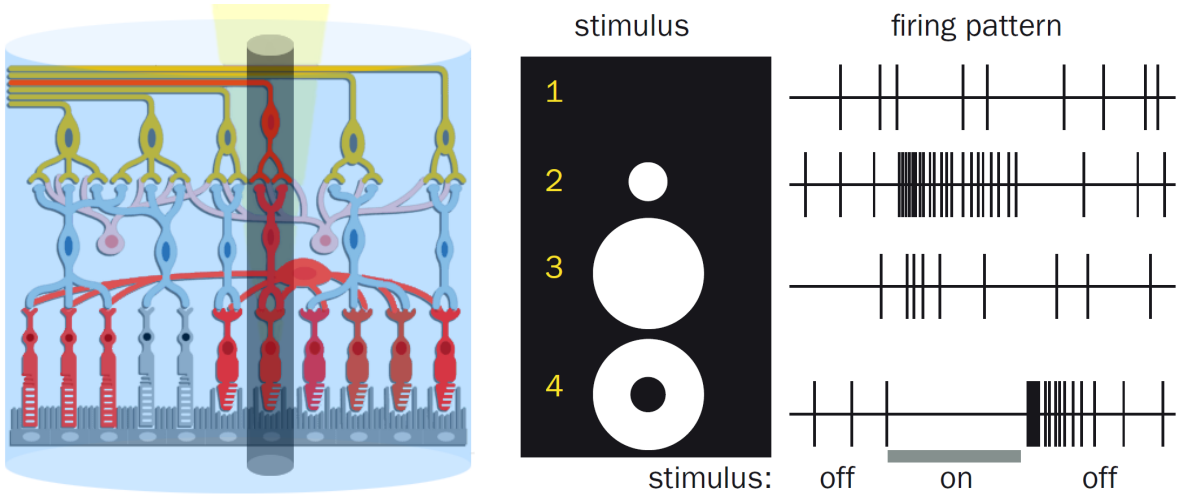
2

Preliminaries

2.1 Retina

The biological retina of mammals, including humans, consists of complex layered structures as shown in Fig. 1.1. The photoreceptor layer contains approximately 6 million cones and 120 million rods cell that respond to the light and convert photons into electrical signals. The rods are sensitive to light of low illumination and respond to even single photon [53], and distributed in the outer fovea thus using in peripheral vision. On the other hand, the cone cells serve as high acuity and color vision, and most of them are distributed in the nearby fovea. The sensitivity of photoreceptors is adapted to light, thus our visual system has a large dynamic range over 10^{11}-fold of ambient light level [7].

The bipolar cells stimulate or suppress the signal of photoreceptor depending on their types: ON-center or OFF-center. They have a receptive field to integrate light information of the center and surrounding photoreceptors as shown in Fig. 2.1. These photoreceptors group are connected by a horizontal cell and the number of connected photoreceptors varies as angles relative to fovea; the number ranges from one at nearby fovea to thousands in the peripheral retina. Consequently,



(a) (b)

Figure 2.1: Description of the center-surround receptive field. In (a), stimulated neurons and related neurons are marked in red. The center and surround are represented as black and blue cylinders, respectively. Refer to Fig. 1.1(a). In (b), the firing patterns of ON-center RGC were depicted as vertical bars in response to the corresponding stimulus on the left. The stimulus duration is indicated by the thick horizontal bar below. This figure is adopted and modified from [7, 8].

photoreceptors near fovea are responsible for high acuity. Then, the receptive fields of ON-center and OFF-center bipolar cells respond to the light in different ways. The ON-center bipolar cell is excited and increases its firing rate when the light of the center photoreceptors is more intense than the surroundings. Conversely, the OFF-center bipolar cell responds to the dark in the center. Concretely, bipolar cells of 10 or more types can be classified based on their dendritic connectivity with rods and/or cones, ON/OFF, and their physiological responses to light [54]. The integration of sustained or transient responses of bipolar cells can represent temporal information of light, and each synapse encodes luminance or temporal contrast [55]. In Fig. 2.1(b), the transient response to temporal contrast can be seen in the changes in stimulus frequency.

Similar to bipolar cells, Retinal Ganglion Cells (RGCs) have various center-surround receptive fields, and the amacrine cells work laterally connecting bipolar cells and RGCs just as the

horizontal cells mediate the signal transmission from the photoreceptors to the bipolar cells. RGCs are divided into at least two main classes based on their functions: midget cell (or P-type cell) and parasol cell (or M-type cell). Midget cells have a small and simple center-surround receptive field and the name "midget" comes from their small size of the dendritic tree and cell body. About 80-90% of RGCs are midget cells and they constitute the parvocellular pathway that responds to color and shape in a sustained response. On the other hand, parasol cells have a large receptive field with relatively many cones and rods. Also, the name "parasol" is derived from their relatively large size. About 5-10% of RGCs are parasol cells and they constitute the magnocellular pathway that responds to motion and changes in luminance in transient response with short latency.

Subsequently, the parvocellular pathway originated from midget cells engages the parvocellular layers of the Lateral Geniculate Nucleus (LGN), and its information arrives at primary visual cortex (V1) in the brain. Similarly, the magnocellular pathway originated from parasol cells engages the magnocellular layers of the LGN, and its information arrives at V1. Information leaving V1 is divided into mainly two streams again: the dorsal stream (a.k.a. "where" stream) is mainly derived from magnocellular pathway and the ventral stream (a.k.a. "what" stream) is from both parvocellular and magnocellular pathway. The dorsal stream is responsible for analyzing object location and motion, whereas the ventral stream is associated with shape, color, and object recognition.

Besides, our visual system perceives motion in a variety of ways, such as direction-selective RGCs responding to a specific direction in which light travels and simple/complex cells in V1 that respond to a movement of line segment by the convergent input of a series of LGN neurons aligned in a line [56]. On the other hand, traditional CMOS cameras focus on capturing static scenes and specialize in shape, color, and object recognition, just like the what stream in the brain. Since the conventional cameras lack of where stream, silicon retina sensors have been developed to deal with motion and temporal contrast information.

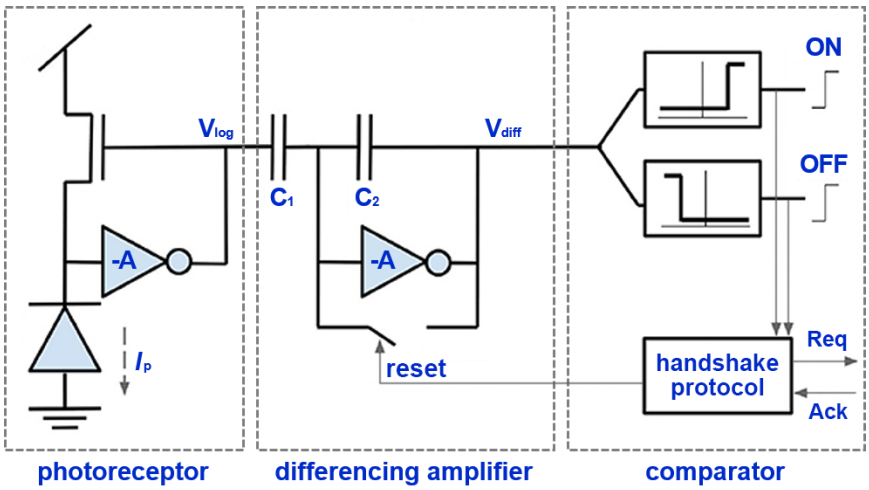


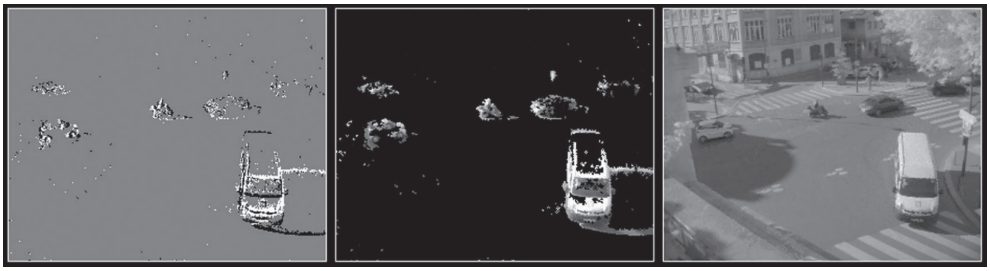
Figure 2.2: Abstract schematic of single pixel in a silicon retina. Each pixel consists of three main components: photoreceptor, differencing amplifier, and comparator [9].

2.2 Silicon Retina

Silicon retina, also known as neuromorphic cameras, dynamic vision sensors, or event cameras, was introduced in [21]. They imitated the structure and operation in the vertebrate retina and developed an artificial CMOS VLSI visual system whose subsystems are inspired by: photoreceptors, bipolar cells, and RGCs. In their system, phototransducing element induces photocurrent proportional to the light intensity, a resistive network integrates the spatial and temporal information by modeling the relationship between the bipolar and horizontal cells, and thus events are generated like action potential in neurons.

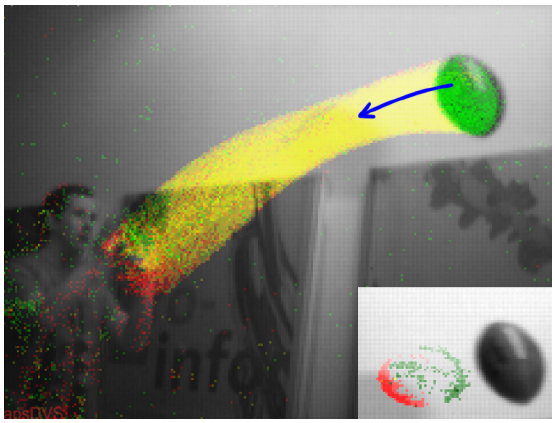
Fig. 2.2 shows an abstract schematic of a single pixel in the recent silicon retina consisting of mainly three components: photoreceptor, differencing amplifier, and comparator circuits. In photoreceptor, light collected in a photodiode leads to the photocurrent I_p. When the gate-source voltage, V_{GS}, less than threshold voltage, V_{TH}, is applied to the Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET), the MOSFET operates in the subthreshold region or week-inversion region, while causing a leakage current

$$I_p = \mu C_d \frac{W}{L} V_T^2 \left(\exp \frac{V_{GS} - V_{TH}}{\eta V_T} \right) \left(1 - \exp \frac{-V_{DS}}{V_T} \right) \tag{2.1}$$



(a) ATIS [57]

(b) DVS [10]



(c) DAVIS [15]

Figure 2.3: Sampled snapshot of silicon retina sensors. In (a) and (b), positive and negative events are represented by white and black dots, and in (c), green and red dots. In (a), both images on the left are the output of ATIS and the image on the right is the reference for indicating absolute intensity.

in a unit of $\mathrm{pA} \sim \mu\mathrm{A}$. For a photodiode generating photocurrent in a similar units to leakage current, the photocurrent induces MOSFET with common source amplifier between drain and gate nodes to generate logarithmic voltage similar to the above equation. In general, since thermal voltage, V_T, is about 26 mV at room temperature and V_{DS} is larger than V_T, thus, the term $\exp(-V_{DS}/V_T)$ goes to zero and the light-to-logarithmic circuit induces the voltage V_{log} below:

$$V_{GS} = V_{log} - V_p, \tag{2.2}$$

$$V_{log} = \eta V_T \log(I_p/I_0) + K, \tag{2.3}$$

where V_p is the cathode voltage of the photodiode, I_0 is the leakage current at $V_{GS} = V_{TH}$, and η is a coefficient relating to the physical characteristics of MOSFET. Then, the differencing amplifier cuts off DC voltage by capacitor C_1 and amplifies the AC component of input voltage by passing through the capacitor C_2. The gain of differencing amplifier in a small-signal model is calculated from

$$v_{diff} = -Av_b \text{ and } \frac{v_{log} - v_b}{1/sC_1} = \frac{v_b - v_{diff}}{1/sC_2}, \tag{2.4}$$

where v_b is the voltage variance of the node between capacitors and v_{diff} is determined by the ratio of capacitors, $-C_1/C_2$, i.e.,

$$v_{diff} = -\frac{C_1}{C_2 + (C_1 + C_2)/A} v_{log} \approx -\frac{C_1}{C_2} v_{log}, \tag{2.5}$$

for a large voltage gain, A. Consequently, v_{diff} denotes the amplified changes in voltage from the reference voltage. Then, the comparator triggers a positive or a negative event when the voltage variance exceeds the threshold value. Immediately after transmitting the event, V_{diff} is reset to the reset level by the handshake protocol which is required to communicate information between the pixel array and the host receiver to handle asynchronous event readout. The following section describes well-known event-based sensors. Fig. 2.3 shows the output of the sensors to be mentioned below, and the specifications of the sensors are summarized in Table 2.1.

2.2.1 DVS

Dynamic Vision Sensor (DVS) [10,11] is practically the first proposed dynamic vision sensor as a product. The transistor-level schematic of DVS is shown in Fig. 2.4. As the same with the abstract schematic above, the photodiode generates photocurrent and M_{fb} operating in the subthreshold region induces a voltage in logarithmic scale. The cascode common-source amplifier with M_{pr} as a current source consists of M_{cas} and M_n, and functions the same as an amplifier in Fig. 2.2. Then, the voltage passes through the source follower composed of M_{b2} and M_{sf}. After that, the voltage V_{diff} is compared to the the ON/OFF threshold voltage that is determined by the gate voltage of MOSFET M_{ONn} and M_{OFFn}, thus triggering the transmission of an event with the corresponding polarity.

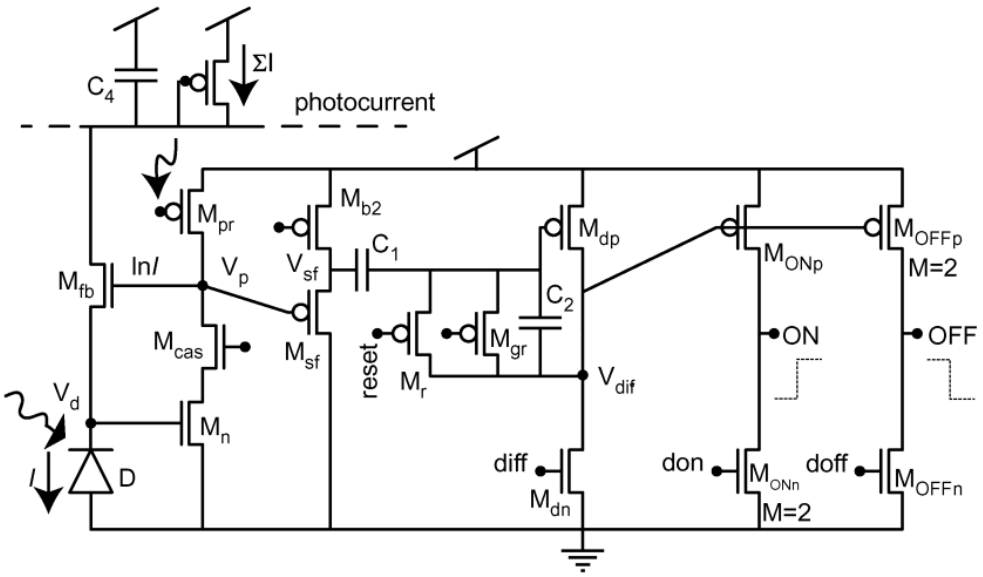


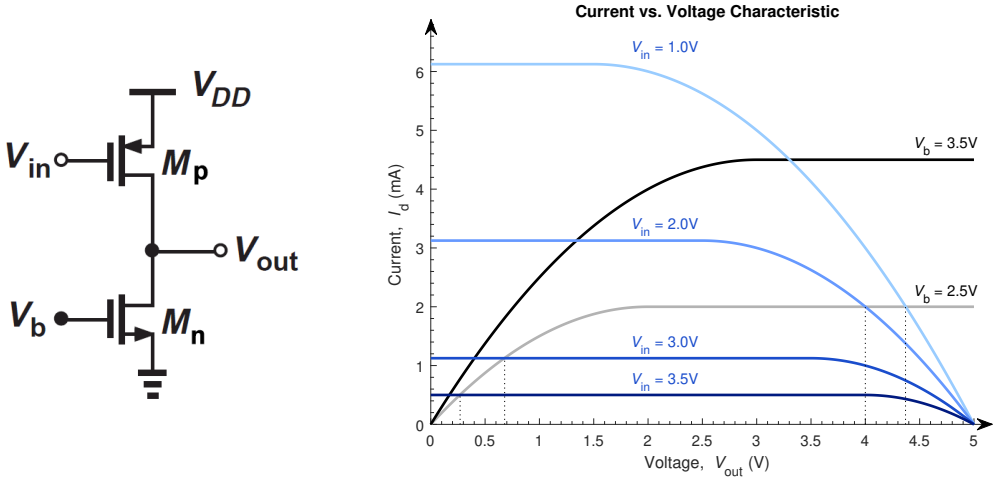
Figure 2.4: Transistor-level schematic corresponding to the abstract schematic in Fig. 2.2 and DVS128 [10, 11].

The single comparator circuit is depicted in Fig. 2.5(a). The circuit consists of P-channel MOSFET (PMOS) and N-channel MOSFET (NMOS), and the threshold of input voltage is determined by V_b. With $V_{SS} = 5\text{V}, V_{th,p} = V_{th,n} = 0.5\text{V}, k_p = k_n = 200\mu\text{A} \cdot \text{V}^{-2}$ and $W_p/L_p = W_n/L_n = 5/1$, Fig. 2.5(b) shows the current-voltage characteristic of both PMOS and NMOS without considering channel length modulation, and they are represented by blue and gray curves, respectively [58]. Then, the output voltage is at the intersection of two curves, since the drain currents of PMOS and NMOS are equal. The threshold voltage of comparator circuit $V_{th,comp}$ is determined by computing input voltage at which the saturated currents of the two MOSFETs are the same:

$$\frac{1}{2}k_p\frac{W_p}{L_p}(V_{DD} - V_{th,comp} - V_{th,p})^2 = \frac{1}{2}k_n\frac{W_n}{L_n}(V_b - V_{th,n})^2, \tag{2.6}$$

$$V_{th,comp} = V_{DD} - V_{th,p} - \sqrt{\frac{k_n \cdot W_n/L_n}{k_p \cdot W_p/L_p}}(V_b - V_{th,n}). \tag{2.7}$$

Therefore, when the input voltage V_{in} decreases below $V_{th,comp}$, the change in voltage is triggered, and the threshold voltage is adjusted by V_b as shown the black lines in the figure and (2.7).



(a) Comparator unit (b) Current-voltage characteristic

Figure 2.5: Single comparator circuit in Fig. 2.2. In (b), the current-voltage characteristic of PMOS, M_p, and NMOS, M_n, of (a) are represented by blue and gray curves, respectively.

2.2.2 ATIS

Asynchronous Time-based Image Sensor (ATIS) [12] is developed to measure the absolute intensity of photodiode together with temporal contrast of DVS circuit. Similar to the magnocellular and parvocellular cells in the biological retina, ATIS pixel consists of a change detector that is identical to DVS and an exposure measurement block that is sensitive to luminance of the scene as shown in Fig. 2.6, respectively. To measure the exposure and gray-scale value of an image, ATIS uses another photodiode, PD2, to collect the light and uses Pulse Width Modulation (PWM) time-based encoding to generate a gray-level event. Thus, contrary to the conventional camera, ATIS generates events and gray-level events asynchronously. The exposure measurement is triggered only if the corresponding change detector transmits an event, and then M_{Rst} is reset making V_{int} be near V_{SS} and the capacitor is charged [13]. Following that, the photodiode in the exposure measurement block induces the photo-current proportional to the light intensity, and the voltage V_{int} is dropped and triggers the time interval t_{int} by comparing V_{int} with V_{refH} or V_{refL}. Immediately after the reset of exposure measurement block, $V_{int} > V_{refH} > V_{refL}$ is satisfied, and by

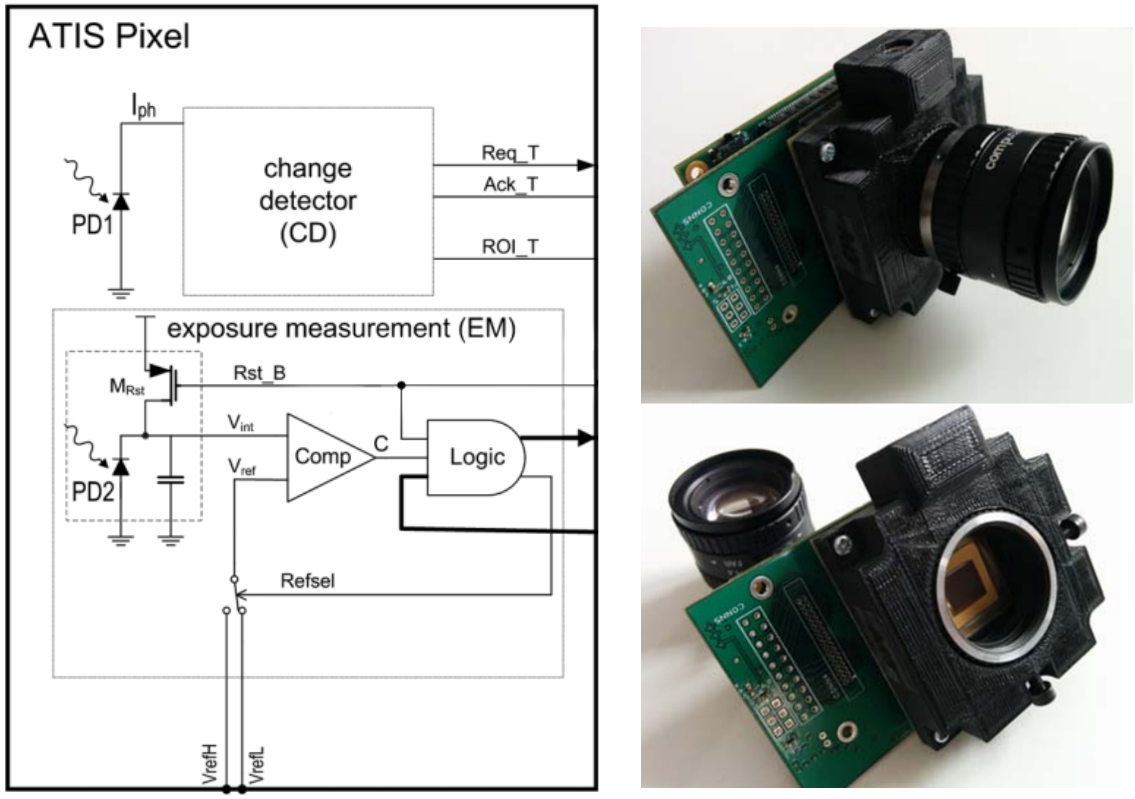


Figure 2.6: Schematic of single pixel in ATIS [12, 13] and the product. The figures are modified from [13, 14].

logic properly switching Ref_{sel}, t_{int} refers to the time it takes for the voltage V_{int} to drop from V_{refH} to V_{refL}. Therefore, the gray-level value of the event is approximately calculated by $1/t_{int}$. Although ATIS fulfills high temporal resolution for not only temporal contrast (event) but also absolute intensity value, its fill factor is compromised due to its separate two photodiodes.

2.2.3 DAVIS

Another product to combine temporal contrast and absolute intensity in a single pixel is a Dynamic and Active-pixel VIsion Sensor (DAVIS) [15] and its schematic is shown in Fig. 2.7. Contrary to ATIS above, Active Pixel Sensor (APS) and DVS share a single photodiode to detect a temporal change and capture an absolute intensity. By setting TX $=$ CR $=$ V_{DD} and after

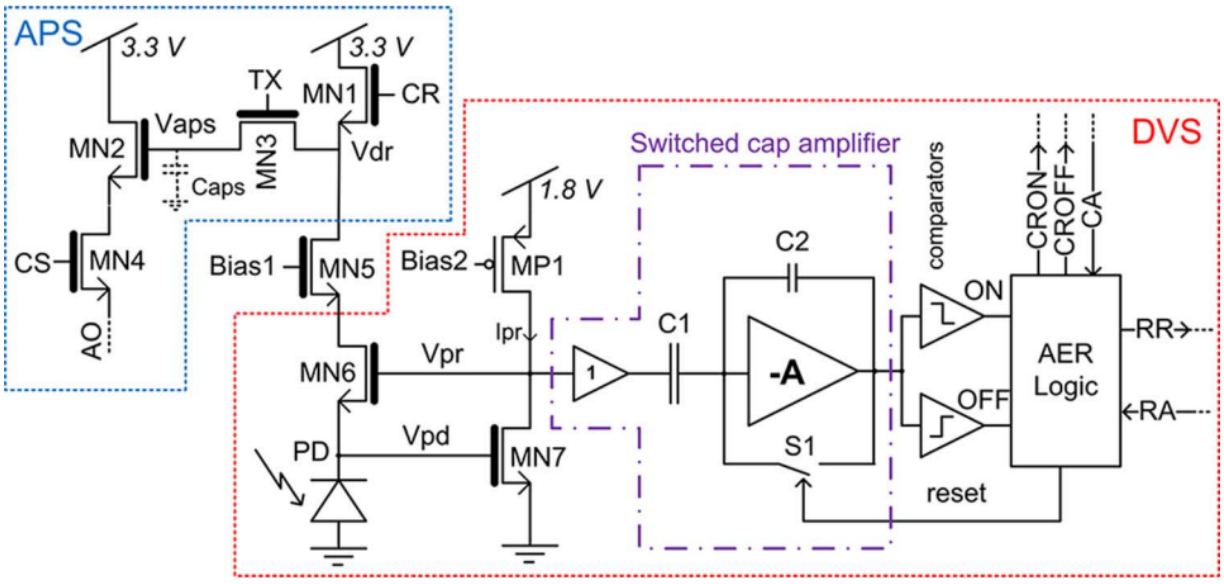


Figure 2.7: Schematic of single pixel in DAVIS [15].

disconnecting them, the voltage of C_{aps}, V_{aps} is charged to reset voltage. Then, by disconnecting CR and connecting TX, the photo-current induced by PD is charged in the capacitor and V_{aps} is read out. Since DAVIS produces an image sequence with a fixed frame rate as the same as the conventional cameras, it can exploit the legacy of traditional computer vision algorithms. However, it also means that APS of DAVIS has limitations of conventional cameras such as motion blur, low dynamic range, etc.

2.2.4 Samsung DVS

Samsung DVS has increased frame resolution to recognize human interactions and shapes for commercial applications. Samsung DVS Gen1 to Gen3 [16] are built with VGA resolution (640×480 pixels), and Samsung DVS Gen4 [59] have achieved 1280×960 pixel resolution. Since the pixel size is reduced due to high resolution, M_{LOG1} and M_{N2} (see Fig. 2.8) are designed to enhance the gain of photoreceptor. Also, their image sensor is fabricated by Back-Side-Illuminated (BSI) process to increase fill factor. Besides, to handle massive events, Group Address-Event-Representation (G-AER) is implemented by binding adjacent 8 pixels along the column into a

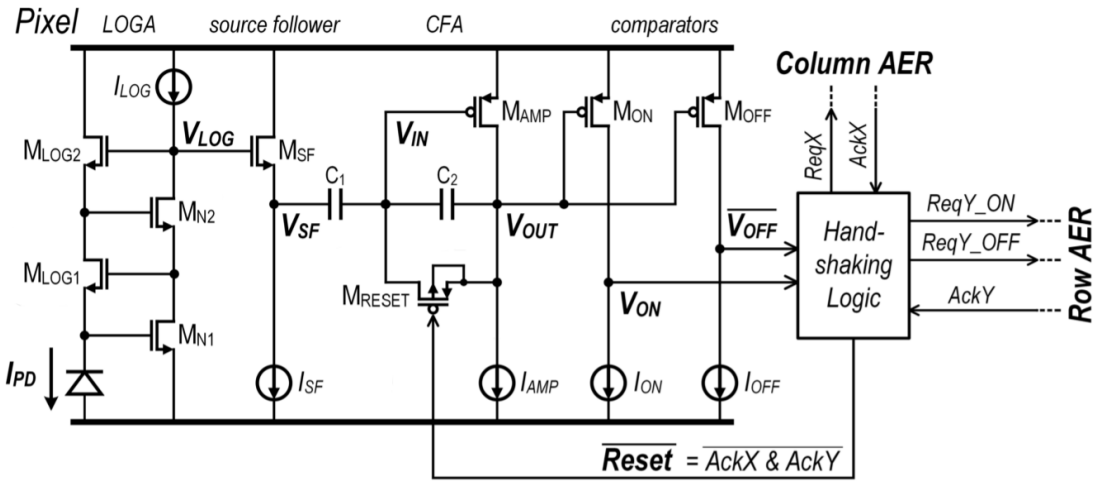


Figure 2.8: Schematic of single pixel in Samsung DVS Gen2 [16]

group. Each group of G-AER is accessed by a column address [$0 \sim 639$] and a group address [$0 \sim 59$], and transmits a packet consisting of ON/OFF 8-bit data. However, it captures event data by scanning columns across the sensor like rolling shutter cameras. In addition, because events belonging to the same group have the same timestamp thus compromising temporal resolution, this sensor is more suitable for batch processing rather than event-by-event operation.

2.3 Event Camera Datasets

In the dissertation, I have used publicly available datasets to validate the performance of the proposed algorithm and compare it with the existing algorithms. The following sections describe datasets used in this dissertation and the specifications of well-known datasets for event cameras are summarized in Tables 2.2 and 2.3.

Chapter 3 employs DVSFlow16 [6] dataset to evaluate the accuracy of local plane fitting method and DAVIS240C [60] dataset to evaluate the similarity score between two edgemaps detected from gray image and events. Some of DVSFlow16 sequences are captured under the purely translating motion in front of simple pattern images, thus it is easy to validate lifetime estimation method. Also, few sequences of DAVIS240C provide high contrast scene whose edge

is easily detected so that edge detection methods can be compared to each other.

In Chapter 4, Multi Vehicle Stereo Event Camera (MVSEC) [61] dataset is used to estimate the accuracy of optical flow vectors and DAVIS240C dataset is utilized to assess the performance of angular velocity in terms of accuracy and latency. Since MVSEC dataset provides the ground-truth flowmap, the accuracy of the proposed optical flow estimation algorithm is validated by average end-point error. On the other hand, the accuracy and latency of angular velocity are evaluated by comparing with the gyro of the Inertial Measurement Unit (IMU) under the 3 Degree-of-Freedom (DoF) rotational motion of the DAVIS240C dataset.

Chapter 5 estimates the angular velocity of the ego motion in dynamic environments that moving object exists and the qualitative and quantitative performances are validated on the author-collected dataset, LARR-DE. Already, there exist datasets recorded in dynamic environments, e.g., EVIMO2 [49], EED [38], but they are not captured under the rotational motion, and their camera moves continuously during the sequences in order to persistently trigger events belonging to the background. Meanwhile, datasets captured under the rotational motion, e.g., DVSFlow16, DAVIS240C, does not include a rigid moving object. Therefore, I have collected datasets that include rotational motion, rigid moving object, and stationary moment in order to clearly evaluate the strength of the proposed algorithm.

Table 2.1: Specifications of silicon retina sensors.

	DVS [11]	ATIS [13]	DAVIS [15]	Samsung DVS Gen2 [16]
Functionality	asynchronous temporal contrast event	asynchronous temporal contrast event + asynchronous absolute intensity	asynchronous temporal contrast event + synchronous image frame	asynchronous temporal contrast event
Chip size	6×6.3 mm^2	9.9×8.2 mm^2	5×5 mm^2	8.0×5.8 mm^2
Array size	128×128	304×240	240×180	640×480
Pixel size	40×40 μm^2	30×30 μm^2	18.5×18.5 μm^2	9×9 μm^2
Fill factor	8.1%	20% EM, 10% CD	22%	22%*
Power consumption	24mW	175mW (high), 50mW (low)	14mW (high), 5mW (low)	50mW (high), 27mW (low)
Dynamic range	120dB	125dB	130dB DVS (0.01 lux), 51dB APS	90dB
Min. latency	15μs @ 1klux	4μs @ 1klux	3μs @ 1klux	N/A

*The number is inferred as a ratio of the pixel area and the photodiode area while ignoring the effect of the microlens.

Table 2.2: Specifications of event camera datasets.

Dataset	Sensors	Data	Condition	
			Motion	Environments
DVSFlow16 [6]	DAVIS 240C (240 × 180)	event, gray image, imu	rotation, translation, slow	indoor, static (artificial pattern)
DAVIS240C [60]	DAVIS 240C (240 × 180) OptiTrack System	event, gray image, imu, pose	6-DoF, rotation, translation, slow/fast	indoor/outdoor, static/dynamic
MVSEC [61]	2×DAVIS 346B (346 × 260) VI Sensor (752 × 480) 16-Ch Velodyne, GPS	event, gray image pair, depthmap, flowmap, imu, pose	6-DoF, slow	indoor/outdoor, static/dynamic
EVIMO2 [49]	2×Prophesee Gen3 (640 × 480) Samsung DVS Gen3 (640 × 480) Flea3 RGB Camera (1280 × 1024)	event, RGB image, depthmap, pixel-wise object mask, trajectory	6-DoF, slow	indoor, dynamic
EED [38]	DAVIS 240C (240 × 180)	event, gray image, imu, bounding box	6-DoF, slow	indoor, dynamic
LARR-DE	DAVIS 240C (240 × 180)	event, gray image, imu	rotation	indoor/outdoor, dynamic

continued on next page.

Table 2.3: Specifications of event camera datasets (continued).

CED [62]	DAVIS 346 (346 × 260)	Bayer-encoded event, RGB image	6-DoF, driving	indoor/outdoor, static/dynamic
DDD20 [63]	DAVIS 346B (346 × 260) OpenXC (vehicle data)	event, gray image, imu, lat/long, vehicle data (steering angle, throttle, rpm, pedal status, etc.)	driving	outdoor, static/dynamic, various weather, day/night
UZH-FPV [64]	miniDAVIS346 (346 × 260) Stereo camera (640 × 480) Leica Nova MS60 TotalStation	event, gray image, stereo image, imu, odometry, pose	flight	indoor/outdoor, static
ViViD++ [65] (handheld)	DAVIS 240C (240 × 180) FLIR A65 (650 × 512) Asus Xtion Pro Live (640 × 480) 16-Ch LiDAR	event, gray image, RGB image, thermal image, depthmap, imu	6-DoF, slow/fast	indoor/outdoor, day/night
ViViD++ [65] (driving)	DVXplorer (640 × 480) Flea3 RGB Camera (1280 × 1024) FLIR A65 (650 × 512) 64-Ch LiDAR RTK-GPS	event, RGB image, thermal image, point cloud, imu, position	driving	outdoor, day/night

3

Intra-pixel-area Events and Applications

The event camera, unlike traditional CMOS sensor, generates sparse event data at a pixel whose log-intensity changes. Due to this characteristic, theoretically, there is only one or no event at the specific time, which makes it difficult to grasp the world captured by the camera at a particular moment. An image reconstruction is the one of the method to recognize the current scene. To construct image frame from events solely, there are two methods; one is to gather events according to their timestamp and another is to stack a specified number of events. However, these stacking events require tuning parameter, time window and the number of events, which are set heuristically and critical to the quality of the event frame. In this work, I propose an robust visual flow against pixel noise so that edgemap is reconstructed by estimating a lifetime of an event from visual flow. A lifetime of event keeps the event alive until the event is generated in a nearby pixel, thus the shape of an edge is preserved. To estimate visual flow robustly, I consider a pixel area to fit a plane on SAE and call the point inside the pixel area closest to the plane as an Intra-Pixel-Area (IPA) event. These IPA events help the fitting plane algorithm to estimate lifetime robustly and precisely. Therefore, the proposed algorithm estimates visual flow precisely, and detects edgemap better in terms of sharpness and similarity metric than the accumulation of events

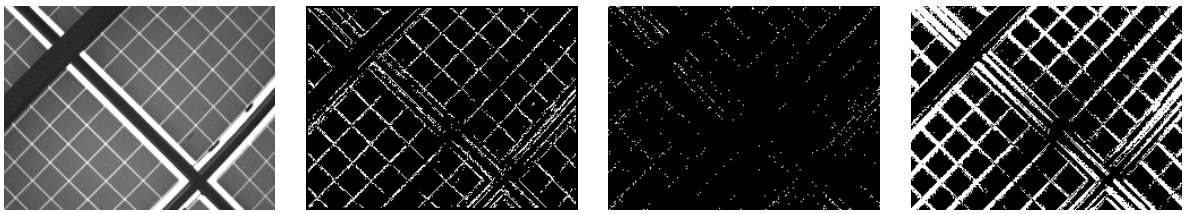
over fixed counts or time intervals, when compared with the existing edge detection algorithms, both qualitatively and quantitatively.

3.1 Introduction

Event camera is a new type of vision sensor, which is motivated by the human eye, unlike standard CMOS sensors [11, 15]. Event camera is composed of an independent circuit for each pixel and these circuits generate events asynchronously when the log-intensity of light applied to the pixel changes. These timestamped event streams have small bit-rate of tens or more kilobytes, whereas the absolute intensity measurement of the standard camera has large bit-rate of tens or more megabytes. By these virtues, the event cameras have several advantages such as low latency, high temporal resolution, whereas the traditional camera captures image frame at a few tens of frames per second, usually.

A wide range of fundamental computer vision applications, essential in autonomous navigation, object detection for avoidance, and Augmented/Virtual Reality (AR/VR), may significantly benefit from the extremely low latency and fast response time of the event camera. However, most algorithms for computer vision cannot be applied directly to the event cameras due the new features mentioned above. So, there has been an increasing interest for developing algorithms suitable for event cameras in applications such as optical flow [19, 35], Visual Odometry (VO) [3, 52, 66], and Simultaneous Localization And Mapping (SLAM) [4, 23]. In this chapter, I focus on the visual flow estimation for time-continuous edge detection algorithm of the event camera, which could be utilized for edge-based VO [67–69] or SLAM [70].

Since the event camera produce event depending on the apparent motion, a visual flow can be estimated from the motion history of events. Its simple implementation includes gradient of the image of stacked events [19]. Also, the simplest way to detect an edge for the event camera is to accumulate events in a specified number or fixed time interval. However, its result is too sensitive to the speed of the camera or image because the value of accumulation volume is heuristically determined as shown in Fig. 3.1(c) and (d). If the number of events generated per unit of time is



(a) Gray image (b) Proposed (c) 1ms (d) 30ms

Figure 3.1: Extracted sharp edgemap of the proposed algorithm. The edgemaps are detected during rotational motion.

less than the expected accumulation, the edge becomes sparse, otherwise, edge bleeding occurs. Also, such accumulation dilutes the advantages of an event camera, low latency.

In the chapter, I aim to fit local plane and estimate visual flow robustly against noise using Intra-Pixel-Area (IPA) pixel approach, then to detect thin edge pixels regardless of the speed of the camera. Also, the proposed algorithm is designed based on an event-by-event basis, thus updates the edge pixels with low latency and high temporal resolution.

3.1.1 Related Work

Since event cameras detect the apparent motion and produce the event at the edge of a scene, visual flow is can be estimated from the history of events. Benosman et al. [18, 19] and E. Mueggler et al. [34] compute visual flow using spatio-temporal surface of events, also known as SAE. They use Lucas-Kanade approach on an SAE [18], compute the gradient of the tangent plane to an SAE using interative minimization [19] and RANSAC scheme [34]. Direction Selective (DS) method [17] divides the orientation types into four, and filters the incoming event out with an OrientationFilter using the correlation between the current and the past events. Due to the predefined filter types, the direction of visual flow has only eight types.

J. Conradt et al. [71] estimate local optic flow as the inverse of the time difference between adjacent events and find consistent global flow consisting of pan, tilt, and yaw by solving the least square problem. J. Lee et al. [72] firstly extract Features from Accelerated Segment Test (FAST)

keypoint in the grayscale image. They update expected polarization map for each incoming event and track keypoints while estimating optical flow vectors.

Also, there are few methods developed for edge detection from the event camera data. Naïve methods are to accumulate events over a fixed time interval or numbers of events. However, in the case of accumulating events over time, the size of the interval depends on the speed of the camera because more events are generated when the camera moves faster. Therefore, the edge bleeding occurs or enough events are not triggered. On the other hand, if an edge is constructed by a fixed number of events accumulation, the quality of edge detection depends on the environment. A sequence captured in a simple environment with little gradient generates fewer events than in a complex environment. Consequently, it is difficult to determine a proper value of fixed time interval or number of events manually.

In order to detect edge robustly against various environmental properties, some algorithms have focused on detecting line edge in a structured environment. An Event-based Line Segment Detector (ELiSeD) [73] computes gradient direction of SAE with Sobel filters, and clusters similar orientations within neighbour events. However, due to the above accumulation approach, edge bleeding occurs, thus lines fitted on the clustered segments may become incorrect. In [74], their event-by-event basis algorithm is inspired by the Hough transform and spiking neuron model. For each event, Hough transform converts their position to the so-called spikes in the (r, θ) space. The detected spikes stimulate the corresponding neuron in the (r, θ) space. This procedure is repeated while the potential value of the neuron exceeds the threshold, and the algorithm generates the line by using the triggered neuron. However, the performance depends on the resolution of (r, θ) parameter space, and is significantly degraded in areas where multiple lines meet.

Some research aim to detect edges, not just line segments that are frequently found in artifacts. F. Barranco et al. [75] detects the contour of foreground objects. They extract features from the accumulated events such as orientation, timestamp, motion, and time texture. Then the boundary is predicted from the learned Structured Random Forest (SRF) given DVS features. However, since this algorithm is developed for object segmentation, it is prioritized to detect the boundary of the foreground, and the performance of the overall edge extraction may be degraded. E. Mueggler

et al. [34] estimates the lifetime of event from local plane fitting on the SAE based on event-based visual flow [19]. However, naïve RANdom SAmple Consensus (RANSAC) method could not be successfully adapted for the event camera, thus causing imprecise estimation. Therefore, I propose a IPA approach for RANSAC in order to estimate a local plane robustly and precisely. Also, I quantitatively evaluate algorithms in terms of similarity metric, which have not been done in most of the previous works.

3.1.2 Contributions and Outline

The main contributions can be summarized as follows:

1. I propose a IPA event approach for loss function of RANSAC, thus achieving robustness against noise and enhancing the performance of visual flow and edge detection for event cameras.

2. I evaluate the edge detection algorithms for the event camera by quantitatively measuring similarity metric.

The rest of this chapter is organized as follows: In Section 3.2, I first characterize the event camera and describe the details of the algorithms. Next, I validate the performance of the proposed algorithm qualitatively in Section 3.4. Finally, Section 3.6 summarizes the extension of this chapter.

3.2 Event Pre-processing

An event camera has independent pixels that detect light changes [11]. When the log-intensity of light applied to a pixel increases or decreases above/below the factory threshold, $\Delta \log(I) = \pm C$, the corresponding pixel (x_i, y_i) generates an event $\mathbf{e}_i = (x_i, y_i, t_i, p_i)$ asynchronously. The event consists of time-stamp, t_i, uv-coordinated position of the pixel (x_i, y_i), and polarity, p_i, where 1 or positive means log-intensity increments and 0 or negative means decrements.

In many existing algorithms for DVS [19,34,76], SAE, Σ_e is used to obtain information about when the event occurs in the corresponding pixel. When an event $\mathbf{e}_i$ appears, the timestamp of the event, t_i, is assigned to $\Sigma_e(y_i, x_i)$.

3.2.1 Event Buffer

Even in the fixed event camera, the camera generates many events which are regarded as noise. These noise data can be suppressed by lowering the sensitivity of the event camera, but it also reduces the number of meaningful events that occur where the image intensity changes. To deal with noise events, I design an algorithm which precedes local plane fitting. The idea is that an event occurs along the gradient edge of the image, and the edge with adjacent pixels will move simultaneously on the image so that events will occur in a short time at adjacent pixels. In other words, events of the same polarity are generated during a short period of time within milliseconds along the same edge segment.

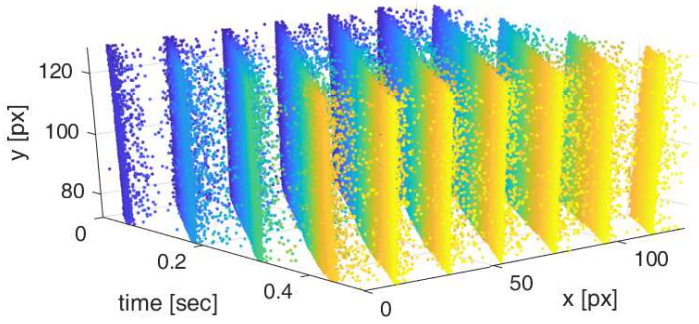
Fig. 3.2 illustrates the SAE in the situation where the several straight stripes move perpendicular to the principle axis of the camera from $t = 0$ to $t = 0.5$. In the following evaluation including Fig. 3.2, I set $\tau_{min} = 0.01$. As shown in Fig. 3.2(a), the raw event stream has a lot of noise data between planes. By pre-processing raw events with the event buffer procedure, I can reduce noise events successfully.

3.3 Event-based Visual Flow and Lifetime

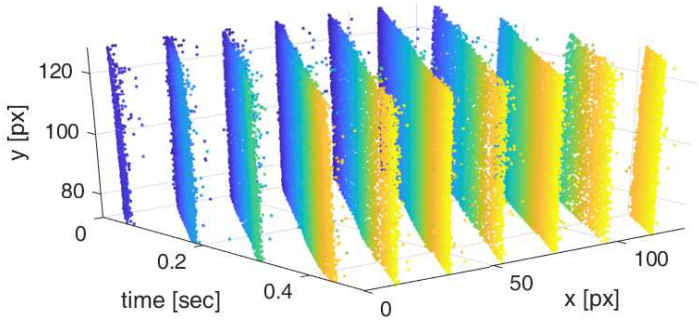
In order to estimate the lifetime of events, I utilize the existing algorithm [19, 34]. They suggest that the lifetime of an event means the time until the corresponding world point that has generated the current event before triggers a new event, and also this period indicates the maximum amount of time before a new event occurs nearby:

$$\tau(\mathbf{x}) = \max\ \Delta t\ \text{ s.t. }\ ||\mathbf{x}|| = 1, \tag{3.1}$$

where $\Delta t = \Sigma_e(\mathbf{x} + \Delta\mathbf{x}) - \Sigma_e(\mathbf{x})$ and $\mathbf{x} = (x, y)$ in pixel coordinates.



(a) Raw events



(b) The filtered events

Figure 3.2: The (x, y, t) pointcloud of events with or without the event buffer. For the stripes sequence which captures the several lines moving perpendicular to the camera principal axis. Between each plane, the noise appears as an isolated point.

To find the steepest slope in the SAE, they estimate the normal vector of the plane, $\mathbf{n} = (n_1, n_2, n_3)$, which is fitted on the SAE locally, and visual flow, $(v_x.v_y)$, and lifetime of the event, τ, are calculated below:

$$v_x = -n_3/n_1,\ v_y = -n_3/n_2, \tag{3.2}$$

$$\tau(\mathbf{x}) = 1/v = 1/\sqrt{v_x^2 + v_y^2} = \frac{1}{n_3}\sqrt{n_1^2 + n_2^2}. \tag{3.3}$$

3.3.1 Local Plane Fitting

The event camera generates a *digital* event when the average brightness of pixel changes, and the position of the generated event is fixed as the center point of the pixel area, thus the true position of an *analog* event is not captured in the event stream. Therefore, I use RANSAC to robustly find a local plane fitted on the SAE. For RANSAC, I use past events in an $N \times N$ window including the current event which is the center of the window. Next, I use k-mean clustering to separate non-triggered or oldest event pixels from the N^2 pixels around the current event by setting k as 2. The initial cluster centroid are set to the minimum and the current timestamp. The set including the current event is selected. Then, I choose the current event and two past events randomly in the selected set to compute the candidate plane.

3.3.2 Intra-pixel-area Event

By the way, the gradient change occurs at some sub-pixel point inside the pixel theoretically, but actually, the event is triggered by the change of the average brightness of a pixel, and the position of the event indicates the center of the pixel. Therefore, I introduce an *IPA event*, which exists somewhere inside a pixel area, $S_{\mathbf{x}}(\delta)$:

$$S_{\mathbf{x}}(\delta) = \{(x, y) | x - \delta < \mathbf{x}_1 < x + \delta, y - \delta < \mathbf{x}_2 < y + \delta\}, \tag{3.4}$$

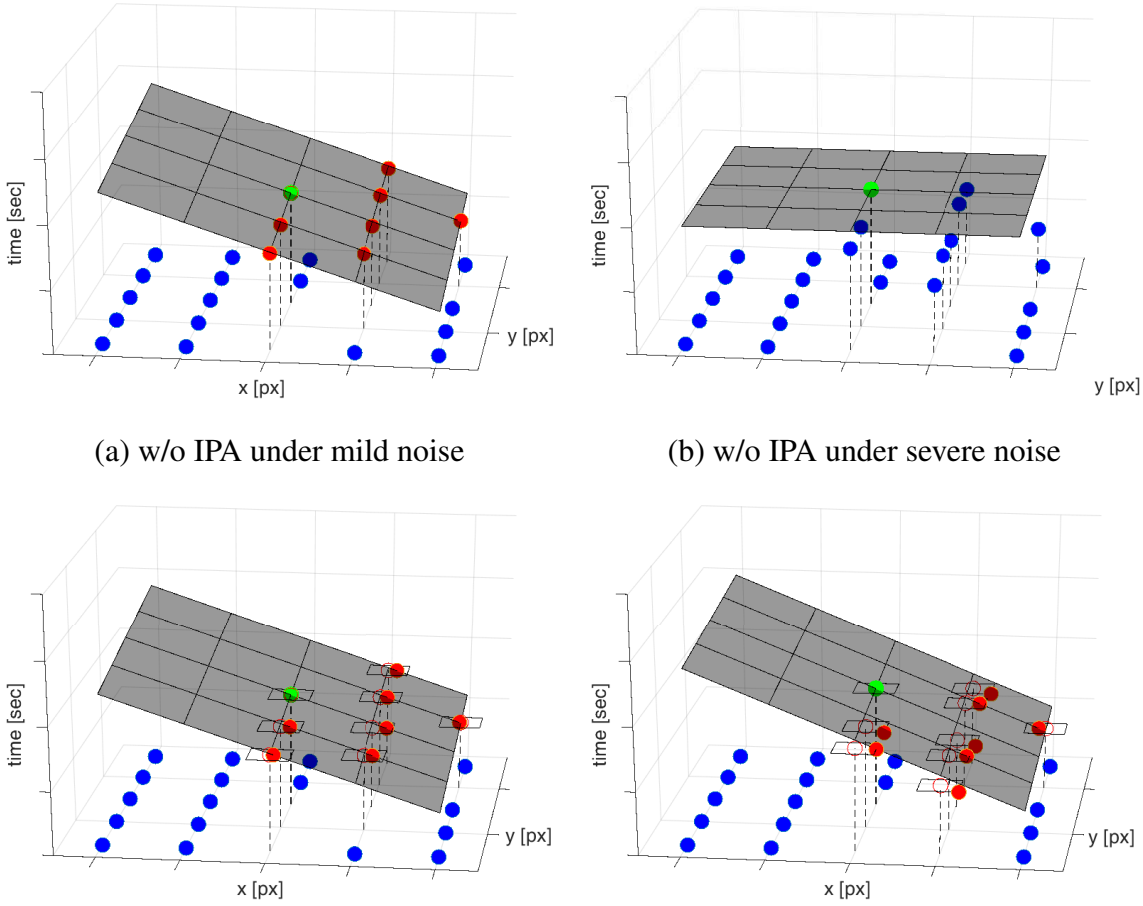
where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ is the pixel of the current event. Thus, I choose the current event and float event of two past events randomly in RANSAC. I compute loss function as the distance between candidate plane, $\mathbf{n}$ $s.t.$ $\mathbf{n}^{\mathrm{T}}\mathbf{x} = 1$, and IPA events, $\mathbf{z}$. Then, the distance function is defined as below:

$$d = \min \mathrm{dist}(\mathbf{n}, \mathbf{z}),\ \forall \mathbf{z} \in S_{\mathbf{x}}(\delta), \tag{3.5}$$

where

$$\mathrm{dist}(\mathbf{n}, \mathbf{z}) = \frac{|\mathbf{n}^{\mathrm{T}}\mathbf{z} - 1|}{||\mathbf{n}||_2^2} \tag{3.6}$$

which contains the current event.



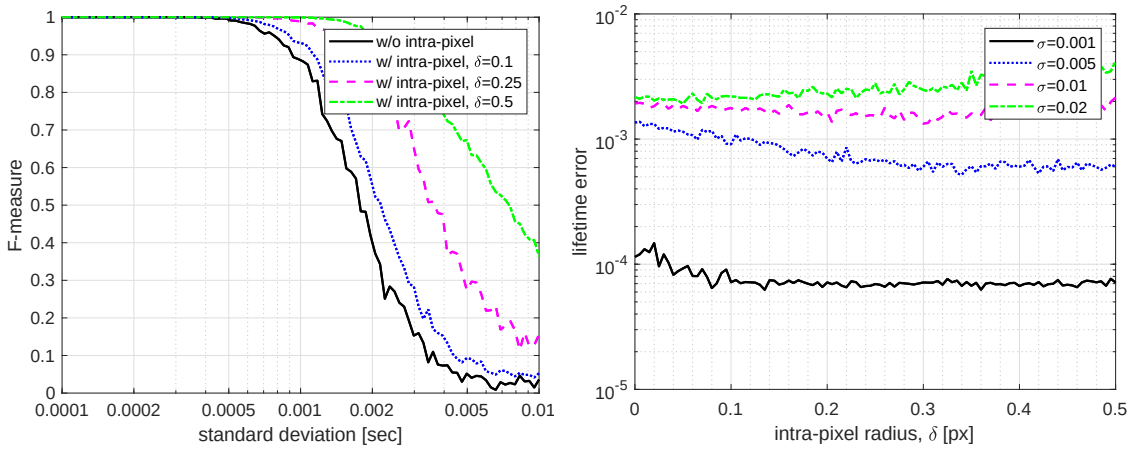
(a) w/o IPA under mild noise

(b) w/o IPA under severe noise

(c) w/ IPA under mild noise

(d) w/ IPA under severe noise

Figure 3.3: Description of the IPA event. After computing a local plane (gray) with or without IPA event by RANSAC, the outliers (blue), inliers (red), and the current event (green) are drawn. In (c), raw inlier events are represented as an *unfilled* red circle, whereas the closest points in each intra-pixel area of inlier event are depicted as a *filled* circle. Also, the intra-pixel areas of the inlier events are depicted as a black rectangle.



(a) w/ or w/o IPA approach

(b) various δ

Figure 3.4: F-measurement evaluation graph with a standard deviation of data noise. (a) F-measure versus standard deviation of data noise depending on intra-pixel radius, (b) lifetime error versus intra-pixel radius depending on data noise.

I show the result of IPA events in Fig. 3.3. It describes the estimation of local plane fitted on the SAE around the current event (green). RANSAC is executed from events in a 5×5 window, and the value of events are given manually for simulation. In particular, the timestamp value of events in Fig. 3.3(b) and (c) are contaminated by severe noise. Comparing the result of Fig. 3.3(a), IPA event approach makes the candidate plane more likely to be voted on by more events, taking into account the positional variation of past events, thus making more robust against the noise.

Furthermore, I analyze the influence of the IPA approach in terms of F-measure and lifetime error as shown in Fig. 3.4. The event data of a 5×5 window used in the analysis are manually made assuming that a single line passes, as shown in Fig. 3.3, which is a very common condition in dataset sequence. Moreover, I add Gaussian noise in whole pixels or several randomly-chosen pixels in Fig. 3.4(a) and (b), respectively. F-measure is defined below:

$$\text{F-measure} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}, \tag{3.7}$$

where recall is a fraction of determined true among a total of true, and precision is a fraction of

true among a total of determined true. Also, lifetime error is the difference between the truth and the estimated values. For each evaluation, I take the average value by repeating a total of 1,000 times.

In Fig. 3.4(a), a higher value of the intra-pixel radius, δ, makes the RANSAC obtain more true inliers robustly under globally-generated noise. However, as shown in Fig. 3.4(b), it also tends to cause lifetime estimation error under scattered noise, because RANSAC may choose undesired noise data due to large intra-pixel radius. From the above analyzes, δ is manually set to 0.25 for the evaluation.

3.3.3 Constant Velocity Assumption

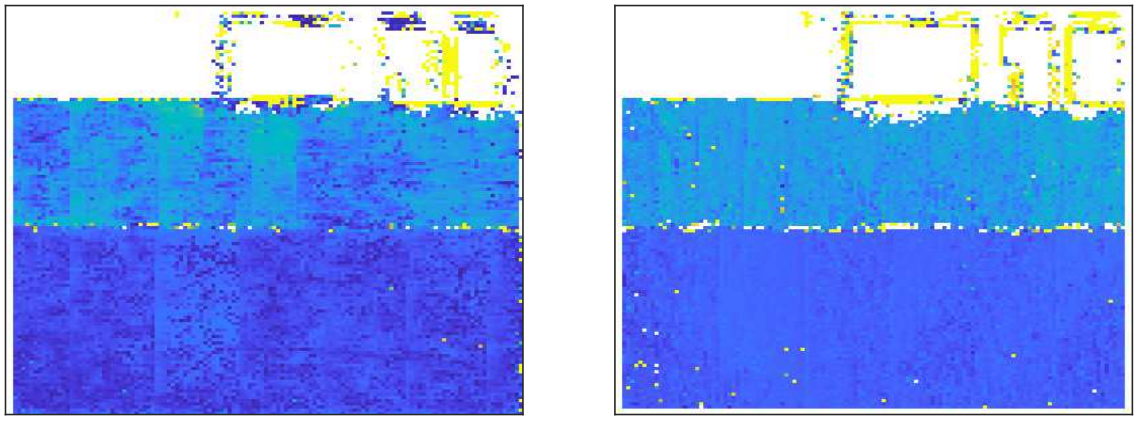
For the local plane fitting, it is assumed that an edge moves at a constant velocity locally. But in practice, edge segments do not move at the same velocity, and they experience acceleration and deceleration. If an edge undergoes acceleration, a lifetime of the edge is overestimated, thus edge bleeding appears. In view of the accelerating situation, when an event is generated earlier than expected before, edge bleeding can disappear by lowering the lifetime of the pixels present in the opposite direction of the local SAE gradient heuristically as the following, rather than deleting the only opposite pixel [34]:

$$\tau(\mathbf{z}) = \max(\tau(\mathbf{z})/\gamma, 0), \quad \text{if } \gamma > 1, \tag{3.8}$$

where

$$\gamma = -2 \cdot \langle \nabla \Sigma_e(\mathbf{x}), (\mathbf{z} - \mathbf{x})/||\mathbf{z} - \mathbf{x}|| \rangle, \tag{3.9}$$

with $\mathbf{z} \in S_{\mathbf{x}}(1)$ around the current event pixel, $\mathbf{x}$, as mentioned in (3.4) and $\langle \cdot, \cdot \rangle$ is inner product in $\mathbb{R}^2$. On the other hand, the lifetime is underestimated when an edge is decelerating, and this case could be a difficult problem to handle by the lifetime approaches. When the camera stops completely in the moment, it is impossible to predict the lifetime of the latest pixel.



(a) E. Mueggler et al. (b) Proposed

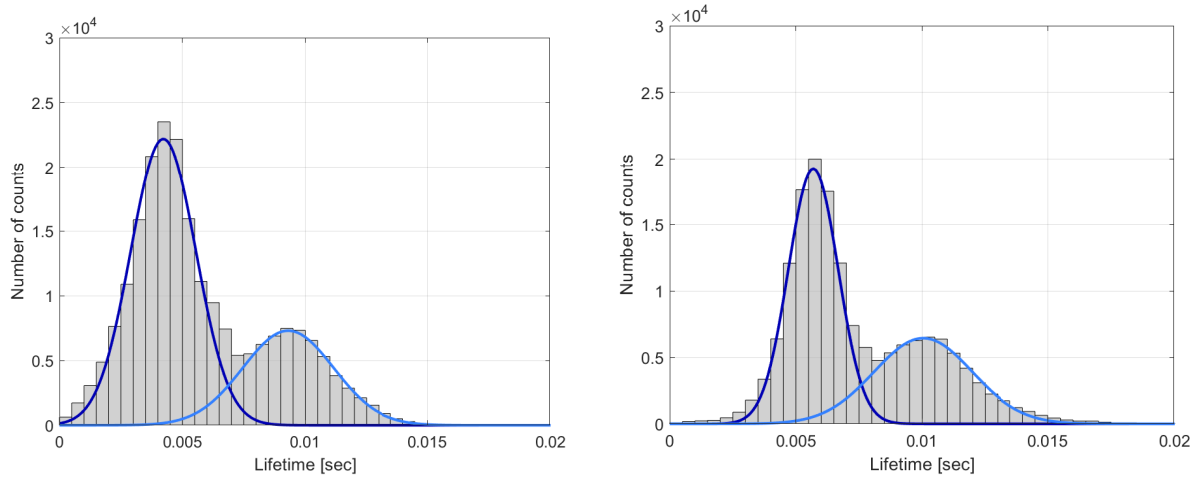
Figure 3.5: Accumulation of lifetime estimates. Yellow color means a large lifetime, whereas blue color means a small lifetime.

3.4 Evaluation and Results

I qualitatively evaluate the comparison algorithms using a sequence provided by DVS (128×128 pixels) in [34]. The other sequences are gray images and events captured using DVS240C (240×180 pixels) for quantitative comparison of edge detection performance.

3.4.1 Qualitative Evaluation

For qualitative evaluation, I first show the accumulation and histogram of lifetime estimates. These analyzes are provided for only a stripe sequence because the sequence is captured at a constant velocity making it easy to verify consistent lifetimes. Through the analysis of lifetime accumulation in Fig. 3.5, it is possible to see how uniform the lifetime is in areas where the stripes pass at a constant speed. Fig. 3.5 shows the accumulation of lifetime estimated during whole stripes sequence [34]. Comparing (a) and (b), E. Mueggler et al.'s algorithm shows small lifetime denoted as deep blue sometimes, while the proposed algorithm shows uniform values across each



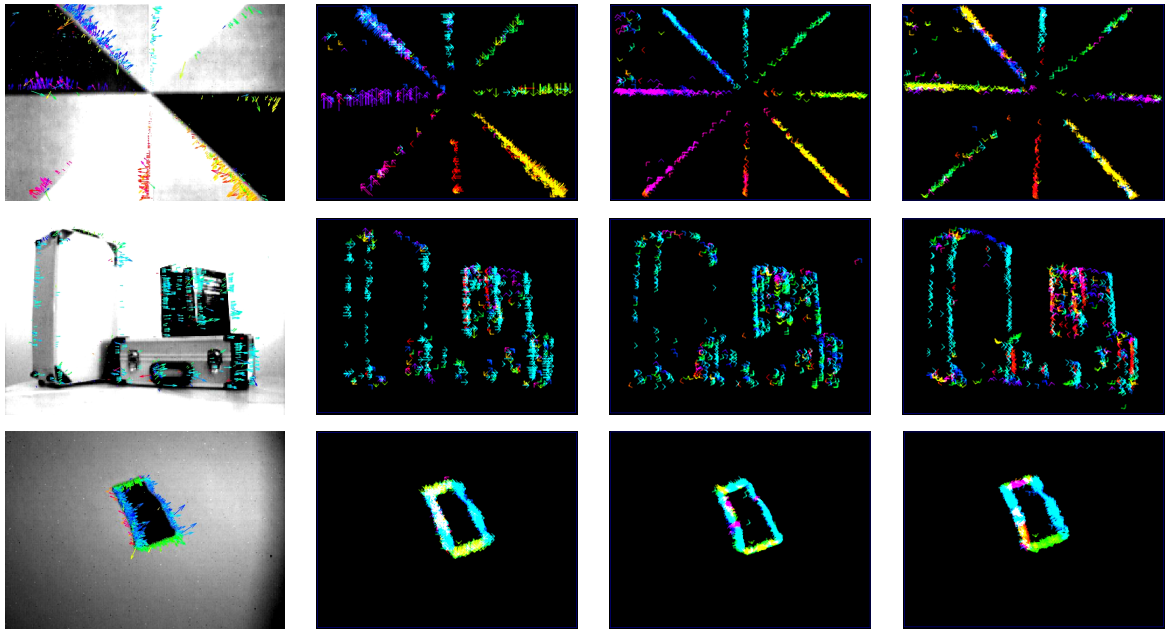
(a) E. Mueggler et al.

(b) Proposed

Figure 3.6: Histogram of lifetime estimates. Gaussian curves fitted to the histogram are drawn overlaid.

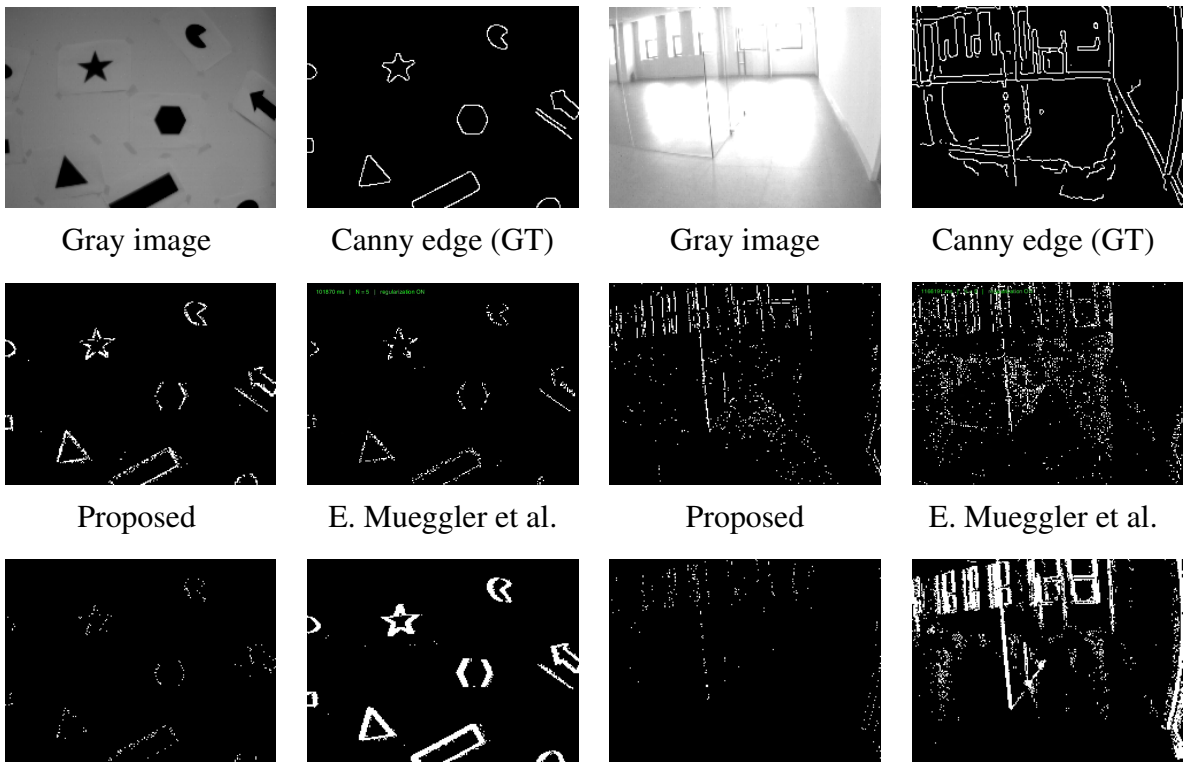
area, in top and bottom. Further, the analysis of lifetime histogram performed in [34] shows how precisely the lifetime is estimated through two main peaks as shown in Fig. 3.6. As in the left part of the lifetime histogram in each Fig. 3.6, the result of the proposed algorithm represents a sharper histogram, which means that the proposed algorithm estimates lifetime precisely. The mean and standard deviation of the Gaussian curve in each subfigure are (a) $\mu_1 = 0.0042, \sigma_1 = 0.0013, \mu_2 = 0.0093, \sigma_2 = 0.0019$ and (b) $\mu_1 = 0.0057, \sigma_1 = 0.0010, \mu_2 = 0.010, \sigma_2 = 0.0019$, respectively.

Moreover, I present qualitative results on DVSFlow16 dataset [6]. Fig. 3.7 shows the snapshot of the visual flow estimation on each sequence. As mentioned previously, DS produces visual flow with only eight directions. The proposed visual flow method and LocalPlane estimate flow vector using plane fitting on an SAE, but the proposed algorithm with IPA approach yields accurate results on IMU_APS_rotDisk and IMU_APS_translBoxes sequences. In third row, the ground-truth movement of black square is in the right direction, however, visual flow results including mine are dominated by the gradient vector due to aperture problem.



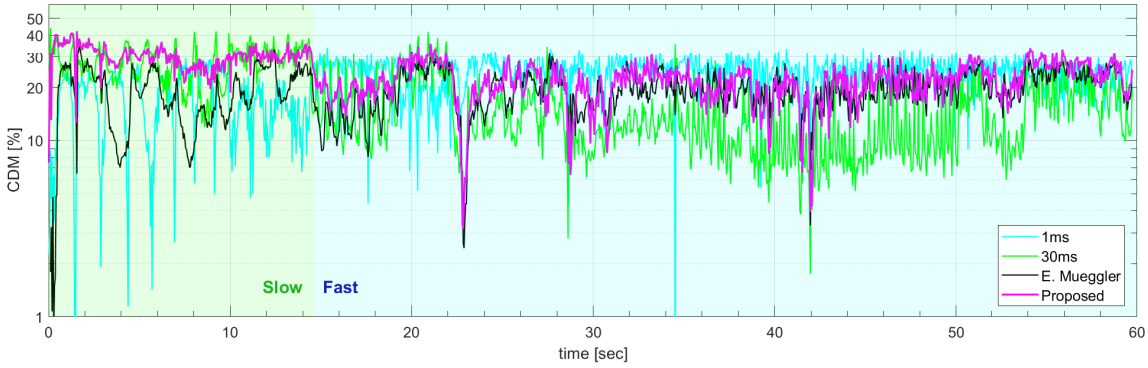
(a) Proposed (b) DS (c) LK (d) LP

Figure 3.7: Qualitative result of the proposed visual flow with gray image, DS [17], Lucas-Kanade (LK) [18], and Local Plane (LP) [19] algorithms from left to right. The sequences used in each row are IMU_APS_rotDisk, IMU_APS_translBoxes, and black_square_translating [6] from top to bottom, respectively.

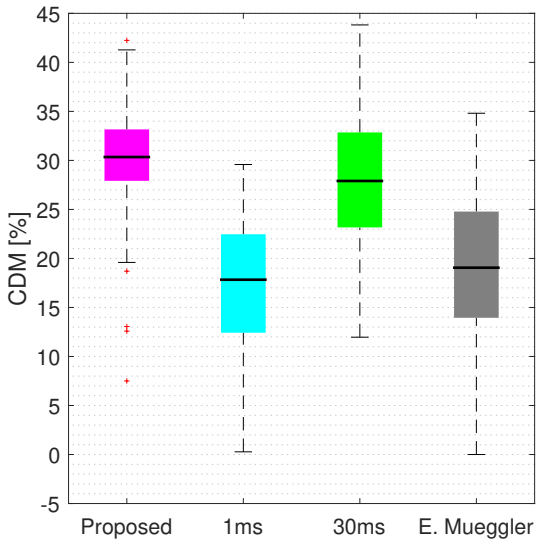


(a) shape_translation sequence (b) corridor sequence (self-collected)

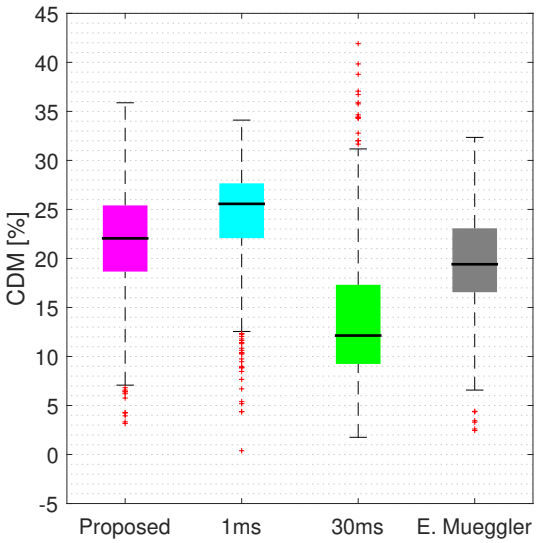
Figure 3.8: Qualitative result of: gray image, ground-truth computed by Canny edge detector, E. Mueggler et al.'s algorithm, 30ms, 1ms accumulation, and proposed algorithm for each sequence in clockwise from top-left.



(a) Similarity over time



(b) Boxplot on the slow section



(c) Boxplot on the fast section

Figure 3.9: Performance analysis of the whole sequence on the shapes_translation sequence. The camera moves slowly in the beginning and quickly in the latter part of the sequence. Thus, 1ms and 30ms accumulation of event shows different performance depending on the speed of camera.

Table 3.1: Evaluations for CDM value in % between Canny edgemap and the estimated edgemap: proposed, 1ms and 30ms accumulation, and E. Mueggler et al.'s algorithm. Best results in each sequence are in bold.

Section		Proposed	1ms	30ms	E. Mueggler
Slow	Median	**30.34**	17.83	27.90	19.06
	IQR	**5.25**	10.05	9.69	10.83
Fast	Median	22.05	**25.57**	12.13	19.41
	IQR	6.75	**5.59**	8.07	6.52

3.4.2 Quantitative Evaluation

In this section, I evaluate the algorithms by measuring similarity to the result of the traditional edge detector from a gray image. For the similarity metric with an edge image, I use the implementation of Closest Distance Metric (CDM) in the work of [77, 78]:

$$CDM_{\eta}(f,g) = 100\left(1 - \frac{\mathcal{C}(\mathcal{M}_{cd}(f,g))}{|f \cup g|}\right), \tag{3.10}$$

where η is the neighbourhood radius to find matching edge pixels between two images, f and g, $\mathcal{C}(\mathcal{M}_{cd}(f,g))$ is the cost of a pair matched by closest-distance criteria, and $|f \cup g|$ is the number of edge pixels belonging to f or g. In evaluation, η is set to 3, and I use 8-connected chessboard distance metric as mentioned in [78]. In addition, for comparison, the results of the Canny edge detector [79] are considered as groundtruth and the performance of algorithms is confirmed by various Canny edge detector parameters because Canny's algorithm can produce fine and well-connected edges with 1-pixel width.

Fig. 3.8 shows the result of: gray image, Canny edge, proposed algorithm, 1ms, 30ms accumulation, and E. Mueggler [34]. Although the lifetime estimation including mine and E. Mueggler's algorithm tend to leave tracks on the edge image, they show a thinner edge than the result of accumulation. In particular, the performance of edge detection over time for a whole sequence is shown in Fig. 3.9 and Table 3.1. The proposed algorithm shows relatively consistent perfor-

Table 3.2: Computational load between w/ or w/o IPA approach. The computation time is measured in seconds per 1k events. The most time-consuming procedures (fitting plane and RANSAC) are compared to each other, and the usage ratio is the ratio of events used for processing among total events.

	Proposed	E. Mueggler
fitting plane (sec)	0.411	0.389
RANSAC (sec)	0.344	0.335
usage ratio (%)	96.6	97.5

mance in terms of the CDM rather the other methods regardless of the camera motion as shown in Fig. 3.9(b) and (c). Sometimes, the proposed algorithm shows bad performance denoted as red-cross outliers when the camera motion is almost stationary. The reason is that a local SAE gradient cannot be estimated precisely when the camera moves slowly. Besides, when the camera stops, the lifetime estimation is theoretically impossible due to the lack of events. Note that the performances of the accumulation methods depend on the speed of the camera in the bottom figure. As can be seen from the InterQuartile Range (IQR) values in Table 3.1, that of 1ms drastically decreases while that of the proposed algorithm slightly increases. This is because the fast section does not include the stationary movement of the camera, whereas the slow section includes stationary movement between changes in direction. Thus, the performance of 1ms accumulation method is improved in the "fast and non-stop" section. At the same time, the CDM value of the proposed algorithm slightly degenerates when it detects edges in the vignetting regions in which the Canny edgemap is not detected.

3.5 Discussion

Table 3.2 summarizes the computational load of the proposed algorithm and E. Mueggler's algorithm, which are executed in MATLAB on i7 processor without GPU computing and the only difference between the two algorithms is whether the IPA approach is applied or not. Because

the distance function of a plane and a IPA event finds the minimum distance as mentioned in (3.5), the computational load increases slightly in fitting plane and RANSAC processes which consume most of runtime. The minimum distance between a plane $\mathbf{n} = (n_1, n_2, n_3)$ and the IPA of an event $\mathbf{z} = (x_e, y_e, t_e)$ is computed simply as

$$d = \begin{cases} 0, & \text{if } \text{sign}((\mathbf{n}^{\mathrm{T}}\mathbf{z} - 1) \cdot (\mathbf{n}^{\mathrm{T}}\hat{\mathbf{z}} - 1)) < 0, \\ \min \text{dist}(\mathbf{n}, \mathbf{z}) = \text{dist}(\mathbf{n}, \hat{\mathbf{z}}) = \dfrac{|\mathbf{n}^{\mathrm{T}}\hat{\mathbf{z}} - 1|}{||\mathbf{n}||_2^2}, & \text{otherwise}, \end{cases} \tag{3.11}$$

where

$$\hat{\mathbf{z}} = \begin{pmatrix} x \\ y \\ t \end{pmatrix} = \begin{pmatrix} x_e - \delta \cdot \text{sign}((\mathbf{n}^{\mathrm{T}}\mathbf{z} - 1)n_1) \\ y_e - \delta \cdot \text{sign}((\mathbf{n}^{\mathrm{T}}\mathbf{z} - 1)n_2) \\ t_e \end{pmatrix}. \tag{3.12}$$

Due to simple modification, the increase in computational load is not very significant.

3.6 Summary

In the work, I proposed a IPA approach to estimate visual flow and detect an edge from events of a dynamic vision sensor by virtue of the lifetime estimation with an event buffer. The designed event buffer regards isolated events as noise and reduces them effectively. Also, the proposed IPA approach makes the algorithm find local plane fitted on SAE robustly so that visual flow and the corresponding lifetime is estimated precisely. Moreover, I analyzed the effectiveness of the IPA approach by the F-measure versus the standard deviation of timestamp noise and the estimation error versus the intra-pixel radius. For evaluation, I qualitatively compare the proposed visual flow with the existing visual flow algorithms and the edgemap with lifetime estimation and naïve event accumulation approach. In addition, with the well-known edge similarity metric, I measure the performance of the edgemap reconstruction algorithm quantitatively. Then, I confirm that the proposed algorithm performs better in terms of sharpness and similarity to the Canny edge than the accumulation of events over fixed counts or time intervals, and the existing lifetime estimation algorithm. Further, by utilizing the detected edge, the proposed algorithm can be employed as a pre-processing part of edge-based VO or SLAM for autonomous robots.

4

Low-Latency and Scene-Robust Asynchronous Optical Flow Stream Estimation

Event cameras are bio-inspired sensors that capture intensity changes of pixels individually, and generate asynchronous and independent “events”. Due to the fundamental difference from the conventional cameras, most research on event cameras builds a global event frame by grouping events according to their timestamps or their number to employ traditional computer vision algorithms. However, in order to take advantage of event cameras, it makes sense to generate asynchronous output on an event-by-event basis. In this chapter, I propose optical flow estimation algorithm with low latency and robustness to various scenes to utilize the advantage of event camera by enhancing the existing optical flow algorithm. Furthermore, I estimate angular velocity with low latency using the proposed optical flow stream. For the validation of algorithms, I evaluate the accuracy and latency of optical flow with publicly available datasets. Moreover, I assess the performance of the proposed angular velocity estimation in comparison to the existing algo-

rithms. Both validations suggest that the proposed asynchronous optical flow shows comparable accuracy to the existing algorithms and the latency is reduced by half compared with the existing block matching algorithm on average. Also, the proposed angular velocity estimation is superior to the existing algorithms in terms of accuracy and robustness while showing low latency within 15 ms consistently.

4.1 Introduction

Event camera, also known as DVS, is a bio-inspired sensor which behaves differently from conventional cameras. The event camera, first designed in [21] and practically proposed in [10, 11], generates asynchronous events at a pixel whose log-intensity changes, unlike frame-based cameras which capture the whole absolute intensity image. The asynchronous characteristic of event cameras brings some advantages; it generates event steam with lower latency in microseconds and grasps the motion relative to it with higher time resolution than the conventional camera.

However, this new paradigm makes it difficult to apply the existing image-based traditional computer vision techniques, such as feature detection, tracking, optical flow, and motion estimation, to event cameras directly. The current research on event cameras can be categorized into two approaches. One builds an event frame by stacking events in a specified count or time window. The other utilizes event asynchronously, which is also known as event-by-event basis operation.

A simple way to adopt the asynchronous approach is to use a single incoming event. However, due to sensor noise, an event may occur where an apparent motion does not exist. Even if an algorithm is performed with valid events only, it takes quite a bit of time to build a local intensity map [23] and obtain the desired output for bootstrapping. This is because the individual event has little information and it implies both spatial and temporal information not just either of the two, which makes it difficult to interpret a single event.

In this chapter, I am interested in developing computer vision applications for event cameras without deteriorating the low latency characteristic, one of the important strengths of event cameras (see Fig. 4.1). The latency of the algorithm can be attributed to practical latency and/or

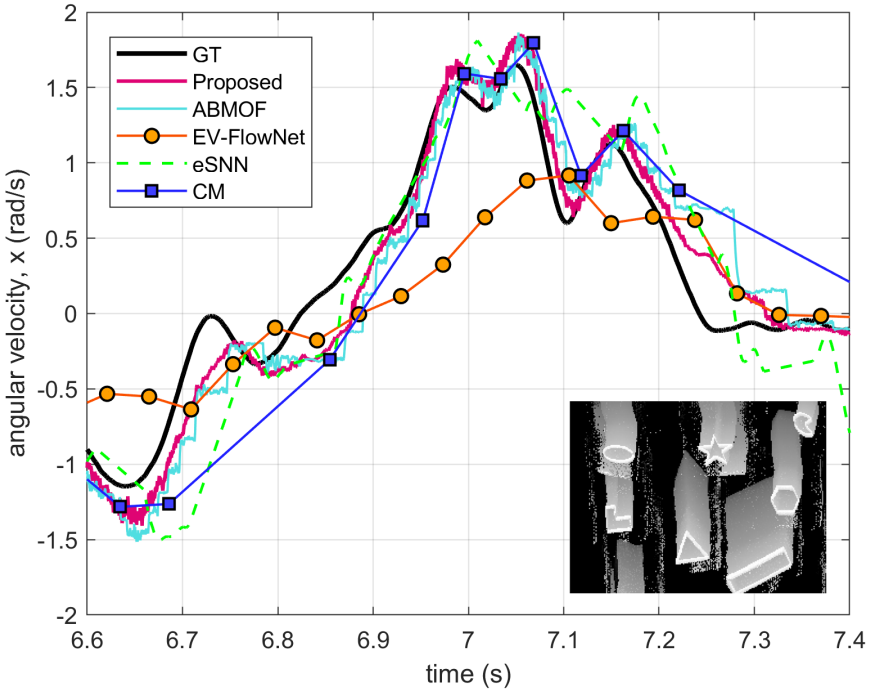


Figure 4.1: Qualitative result of angular velocity estimation on shapes_rotation sequence. The bottom-right grayscale image denotes an image of stacked events over the time interval of [6.6, 7.4] seconds showing the camera movement. Bright values are recent events. Best viewed in color.

theoretical latency. The practical latency includes actual computation time associated with programming setup or hardware performance. Meanwhile, theoretical latency denotes the time to gather events until estimation of optical flow or angular velocity is completed ignoring computation time. I aim to reduce theoretical latency that occurs independent of programming setup or hardware performance. Particularly, I focus on estimating asynchronous optical flow stream on an event-by-event basis. The reason why I choose optical flow is that it is one of the fundamental elements in many computer vision algorithms, which can be utilized in feature tracking, motion estimation, etc. To measure the latency of the optical flow stream, I estimate 3D angular velocity from a bunch of optical flows. The evaluation suggests that the proposed angular velocity estimation shows higher accuracy and lower latency than the other existing algorithms, which also

implies a low latency of the proposed optical flow.

4.1.1 Related Work

For dealing with asynchronous event streams, there are roughly two main approaches. The first type of research performs an algorithm on each incoming event without constructing event frames, and makes use of the strength of event cameras. Each event has a small amount of information, thus most research with an event-by-event basis implements filter-based [4, 23] or network-based algorithms [26]. However, since the time for a filter or a network to converge is not negligible, it is an obstacle to the high-speed capability of event cameras.

Another type of research builds an event frame by stacking events according to their timestamp [43, 80] or grouping them with a specific number [35]. The former stacking method is very similar to the frame of the conventional camera, which has a fixed frame rate. However, when the camera moves slowly, the number of stacked events might be insufficient to obtain valid output, because the event frame consists of a small number of events. Conversely, in a fast motion, bleeding edges can appear in event frames [34]. On the other hand, the quality of the latter grouping method depends on the degree of textures. In the scene with sparse textures, the number of events in the group should be reduced to prevent edge bleeding, and the inverse relationship is also established. Particularly, in the case of a scene with non-uniform textures, some areas of the frame suffer from bleeding edge while others suffer from lack of events. Although some research capture the adaptive event frame with different time windows [52] or the number of events [30, 50], their global event frame still loses details of the scene consisting of non-uniform textures because their event frames stack events across the whole frame as mentioned above.

In summary, algorithms on an event-by-event basis suffer from slow convergence of a filter or a network, which diminishes the advantage of event cameras: low-latency. On the contrary, building an event-stacked image requires heuristic adjustment of the time window or the number of events and also diminishes the advantage of event cameras: high time resolution or low latency. For these reasons, in order to maintain the capability of event cameras, I design an optical flow algorithm that is asynchronous not only temporally but also spatially.

Optical Flow

The existing optical flow algorithms for event cameras are roughly divided into two categories as above. Adaptive time-slice Block-Matching Optical Flow algorithm (ABMOF) [30,50] computes optical flow at where the event triggers for every incoming event using the two most recent of the previous time slices that are similar to SAE [81]. It produces asynchronous optical flow streams, but latency occurs because it uses two of the past time slices, not including the current event. Moreover, in the case of a scene where the texture is not uniform, events are rapidly accumulated in some areas with complex textures, so a new time slice may be generated even though sufficient events are not accumulated in other areas with sparse textures. LocalPlane [19] generates normal flow by using the gradient vectors, which is computed from the local plane fitting on SAE. Although it estimates flow vector with negligible latency, local plane fitting approaches are vulnerable to the aperture problem, thus generating optical flows that are different from true optical flows.

On the other hand, in [25, 31, 35, 43, 80], they aggregate events to estimate optical flow, thus the latency of their algorithms depend on heuristic parameters such as time window or event counts. Bardow et al. [43] use variational method for solving motion field map and brightness image simultaneously. They discretize time window into K intervals each of length δ_t ms that is set heuristically depending on the speed of motion in a scene. Contrast Maximization (CM) framework [35] solves various problems such as optical flow, depth, and motion estimation by maximizing the sharpness of stacked image of warped events according to the model parameter. It uses a set of several tens of thousand events, which is determined heuristically depending on the texture of a scene. Also, their simple framework can be adapted to use a set of events within a time window and both versions are validated in this chapter. Zhu et al. [31, 80] train Convolutional Neural Network (CNN) to learn to estimate optical flow in self-supervised and un-supervised manners, respectively. However, CNN-based algorithms take image-like data as input, and the input representation consists of tens of thousands of events. In order to handle asynchronous and discrete events over time, there is research on combining event camera and

SNN [25, 46]. Spike-FlowNet (Spike-FN) [25] present a deep hybrid architecture by integrating SNNs for encoder and analog neural networks for residual and decoder layers. However, their accumulators in between SNNs and analog neural network layers collect output from SNNs until all event images have passed, thus increasing the latency.

Angular Velocity

Research on angular velocity estimation for event cameras includes an algorithmic method that utilizes an events group [28, 35, 82] and a network-based method with an event-by-event basis [26]. The former algorithm, referred to as CM, estimates angular velocity using a warping model which transforms an event in (x, y, t) space into the same xy-plane with a rotation parameter. This optimization-based method produces precise angular velocity, but it requires heuristic parameter adjustment depending on the scene as mentioned before. In [26], they propose angular velocity regression with SNN. The input representation of their network is a set of events within a time window of 1 ms. The network successfully predicts the angular velocity, but it takes a settling time of 50 ms due to SNN's dynamics at the beginning of predictions and has not been validated for a drastic motion.

4.1.2 Contributions and Outline

I tackle an optical flow and angular velocity estimation problem without any assumptions about the environment, initialization, or additional sensors. Main contributions can be summarized as follows:

- I propose a low-latency algorithm that uses only events and robustly estimates optical flow in various environments.
- I present an accurate 3D angular velocity estimation algorithm, which fetches the proposed asynchronous optical flow stream.
- I compute latency between estimates and ground-truth in an optimization-based method and analyze algorithms thoroughly in terms of latency and accuracy.

The rest of the chapter is organized as follows: In Section 4.2, I explain the proposed asynchronous optical flow and angular velocity estimation algorithms. Next, I evaluate the proposed algorithm and the existing algorithms on publicly available datasets to analyze the accuracy and latency in Section 4.4. Then, Section 4.5 discusses additional validation of performance for various configurations and future work related to limitations, followed by the summary of the chapter in Section 4.6.

4.2 Asynchronous Optical Flow

4.2.1 Event Camera

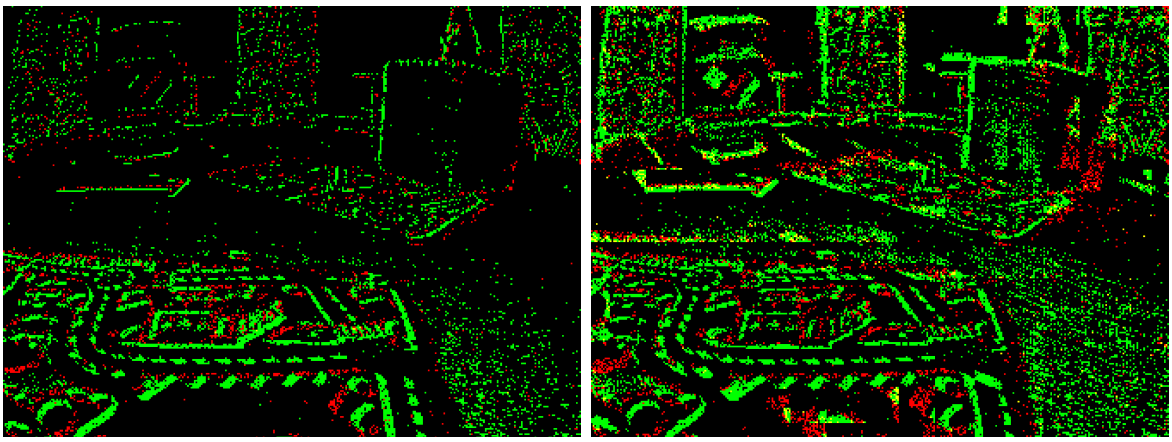
Unlike conventional cameras which output a global frame, pixels of event cameras capture brightness changes individually [11]. Each pixel emits an event when the pixel's log-intensity change exceeds the factory threshold. At that time, the i-th event $\mathbf{e}_i$ consists of the corresponding pixel location $\mathbf{p}_i = (u_i, v_i)$, timestamp t_i, and polarity p_i that means the sign of log-intensity change:

$$\mathbf{e}_i = (t_i, \mathbf{p}_i, p_i). \tag{4.1}$$

As many existing algorithms [19, 34, 35, 80] do, I also utilize SAE for extracting a high-level information from event stream. SAE, also referred to as time slice or event frame, has the same dimension as the camera frame, and the incoming event sets the value of SAE at the corresponding pixel location as the timestamp of the event.

4.2.2 Local Time-Slice Update

ABMOF [30] computes optical flow where the event triggers for every incoming event. It divides the image frame into grids of fixed size, counts the number of events accumulated in each grid, and uses the count as a criterion for constructing a time slice to prevent edge bleeding from the fast camera motion. However, each time slice does not consider the texture distribution, thus erroneous optical flow can occur in areas with low texture. As can be seen from Fig. 4.2(a), details



(a) ABMOF (b) Proposed

Figure 4.2: Snapshot of time slice of (a) ABMOF and (b) the proposed algorithm. Positive and negative events are drawn with green and red dots, respectively. Since the proposed algorithm does not construct a full-sized time slice, local time slices of pixels is drawn overlaid on a frame for visualization purposes only.

are lost on the upper side of the time slice. To overcome the above limitation, I construct local time slices for all pixels individually and compute optical flow between two slices, thus not only enhancing accuracy in a non-uniformly textured scene but also reducing latency.

The proposed optical flow algorithm differs from ABMOF in two ways. First of all, contrary to ABMOF that constructs globally shared time slices, the proposed approach maintains independent queues for all individual pixels to construct local time slice. The capacity of the queue is defined as twice the number of elements in the local time slice patch of size $(w \times w)$. The front half of the elements forms the previous local time slice and the rear half constitutes the current local time slice. Each queue fetches all incoming events within its local window and updates its local time slice individually so that the update rate of each time slice depends on the degree of local texture. In other words, the local time slices of a high-textured area in the frame is updated more than that of a low-textured area. Hence, as shown in Fig. 4.2, the proposed algorithm constructs local time slices with similar levels of event density for any region so that an optical flow can be estimated accurately. Secondly, the proposed algorithm compares the current

Algorithm 1 Asynchronous optical flow estimation

Input: $\mathbf{e}_i = (t_i, \mathbf{p}_i, p_i)$ ▷ *single event*

Output: $OF_i = (t_i, \mathbf{p}_i, \mathbf{v}_i, dt_i)$ ▷ *optical flow*

1: **for all** $\mathbf{p}_j \in \text{adj}(\mathbf{p}_i)$ **do** ▷ *adjacent pixels in* $(w \times w)$

2: push $\mathbf{e}_i$ into queue($\mathbf{p}_j$)

3: **end for**

4: **if** queue($\mathbf{p}_i$) is full **then**

5: construct $\text{SAE}_{i,curr}$ and $\text{SAE}_{i,prev}$ from queue($\mathbf{p}_i$)

6: compute $\mathbf{v}_i$ by matching $\text{SAE}_{i,curr}$ and $\text{SAE}_{i,prev}$

7: $dt_i \leftarrow \text{mean}(\text{SAE}_{i,curr}) - \text{mean}(\text{SAE}_{i,prev})$

8: **return** OF_i

9: **end if**

time slice including the latest event to the previous time slice consisting of the front half of the queue, whereas ABMOF finds the best matching block between two previous time slices. Because ABMOF does not use the current event when computing optical flow, its latency increases. Consequently, the proposed approach utilizing the current event immediately estimates optical flow with less latency.

Algorithm 1 demonstrates the pseudocode of the algorithm explained in Section 4.2.2, and Fig. 4.3 illustrates steps 1 through 5 of Algorithm 1. For every incoming event, I push the event into the corresponding queues which cover the pixel position of the event. Then, if the queue is full, I divide the elements of the queue in half and stack them to construct two time slices. In common with ABMOF, I use the diamond search method [83] for block matching algorithm based on the Sum of Absolute Difference (SAD) between two time slices. For efficient and accurate search, I find the best matching block quickly and iteratively in the current time slice with a large diamond search pattern and make an exhaustive search within a small diamond search pattern. After block matching, asynchronous optical flow stream contains the timestamp t_i (the latest timestamp of the queue) and pixel location of the event $\mathbf{p}_i$ (starting point of the vector),

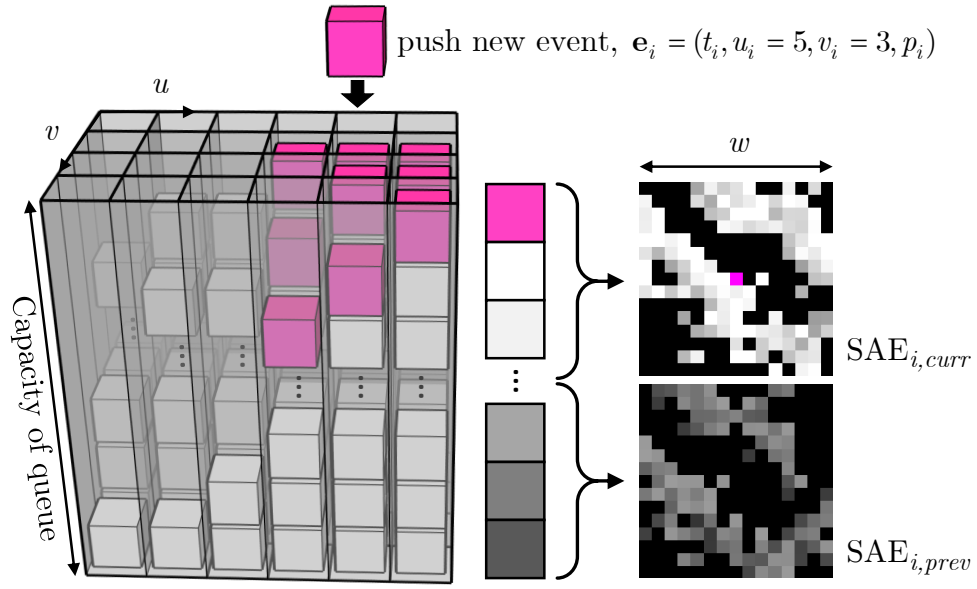


Figure 4.3: Description of the local time slice. Each bin denotes the queue of a pixel, and a new event (magenta cube) is pushed into adjacent bins. For visual simplicity, I illustrate the case for patch size $w = 3$ in the 3D image and its vertical direction representing the queue capacity. Elements in a full queue are divided in half, and construct the current and previous local time slices, SAE_{curr} and SAE_{prev}, respectively. Bright values are recent events.

optical flow vector in pixels $\mathbf{v}_i$, and the average time difference between two time slices dt_i so that the optical flow can be converted to $\mathbf{v}_i/dt_i$ having a unit of px/s:

$$OF_i = (t_i, \mathbf{p}_i, \mathbf{v}_i, dt_i). \tag{4.2}$$

ABMOF computes optical flows through the diamond search for efficiency, so the v_x, v_y values of the optical flow vector $\mathbf{v}$ are integers in pixels. Because of the nature of this search method, the direction of an optical flow vector is discretized depending on the magnitude of the vector. For example, if the best matching block in the current local time slice is one pixel away from the previous local time slice, the angle of the vector has eight directions: (-1,-1), (0,-1), (1,-1), (-1,0), (1,0), (-1,1), (0,1), (1,1). Nevertheless, the usage of local time slice can precisely derive the magnitude of optical flow in px/s units, thanks to the high temporal resolution, and

obtain accurate angular velocity from the bunch of optical flow vectors. However, I enhance the accuracy of the previous block matching algorithm by computing an optimum optical flow vector with a sub-pixel resolution using quadratic interpolation. The quadratic interpolation method estimates an optical flow vector using scores of the surrounding of the previous optical flow result. This additional procedure is able to compute optical flow in a precise direction even in a small movement and improve the angular velocity estimation.

In practice, I do not construct time slices from the beginning repeatedly, but update them gradually for each incoming event for efficiency. When a new event is fed into the queue, an event in the middle of the queue begins to construct the previous time slice. At that time, the previous time slice is only affected by the event in the middle and the oldest event which will be popped. Also, the current time slice is modified by only the event in the middle and the new event. In other words, the time slice stacks the timestamp of the event when fetching an event. Next, for an event that will be popped, the value at the corresponding pixel location in the time slice is reset as zero only if the existing value of the time slice is the same as the timestamp of the event.

4.3 Angular Velocity Estimation

To deal with a single piece of data from an asynchronous optical flow stream, the use of filters could be considered. Since the equation solving optical flow given an angular velocity and a pixel location is a linear transformation and a linear operator to a Gaussian distribution also results in the same distribution, the Kalman filter can be applied. However, the filter-based method increases latency due to its dynamic characteristics and requires many tuning parameters for the covariance matrices.

Because of the above reasons, I decide to compute angular velocity analytically from a bunch of optical flows. Given the vector $\boldsymbol{\omega}$ representing 3D angular velocity, optical flow under a pure rotation is computed by $\mathbf{v}_i = KR(\boldsymbol{\omega} dt_i)K^{-1}\mathbf{p}_i - \mathbf{p}_i$, where K is the intrinsic matrix of the camera, $R(\cdot)$ is the rotation matrix of the corresponding $\boldsymbol{\omega}$ and time passed dt_i, and $\mathbf{p}_i$ is image

point in the pixel coordinates. In a short time interval, the equation can be approximated as below:

$$\mathbf{v}_i = \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} x_i y_i & -1 - x_i^2 & y_i \\ 1 + y_i^2 & -x_i y_i & -x_i \end{pmatrix} dt_i \boldsymbol{\omega}, \tag{4.3}$$

where f_x, f_y are the focal length of the principle axes in pixels, x_i, y_i are image points in the normalized image coordinates, and dt_i is the time interval while the image point has moved. Let (4.3) be $\mathbf{v}_i = A_i dt_i \boldsymbol{\omega}$ for simplicity, and I find the angular velocity by solving the least-squares problem:

$$\boldsymbol{\omega} = (A_{1:n}^{\mathrm{T}} A_{1:n})^{-1} A_{1:n}^{\mathrm{T}} T_{1:n}^{-1} \mathbf{v}_{1:n}, \tag{4.4}$$

where $A_{1:n} = [A_1^{\mathrm{T}}, \cdots, A_n^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{2n \times 3}$, $\mathbf{v}_{1:n} = [\mathbf{v}_1^{\mathrm{T}}, \cdots, \mathbf{v}_n^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{2n \times 1}$, and $T_{1:n} \in \mathbb{R}^{2n \times 2n}$ is diagonal matrix composed of dt_i, i.e., $T_{2i-1,2i-1} = T_{2i,2i} = dt_i, \forall i = 1, ..., n$ for n measurements.

In addition, let me assume that each vector of optical flow stream has 2D Gaussian pixel noise, i.e., $\mathbf{v}_i \sim N(\bar{\mathbf{v}}_i, \Sigma_{\mathbf{v}})$. Since the proposed asynchronous optical flow stream has integer values $\mathbf{v}_i = (v_x, v_y)_i$ due to the diamond search method, I set $\Sigma_{\mathbf{v}} = 0.5^2 I_2 \in \mathbb{R}^{2 \times 2}$. Then, the covariance of $\boldsymbol{\omega}$ is

$$\begin{aligned} \mathrm{Var}(\boldsymbol{\omega}) &= (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} T^{-1} \mathrm{Var}(\mathbf{v}) T^{-1} A (A^{\mathrm{T}} A)^{-1} \\ &= \sigma^2 (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} T^{-2} A (A^{\mathrm{T}} A)^{-1}. \end{aligned} \tag{4.5}$$

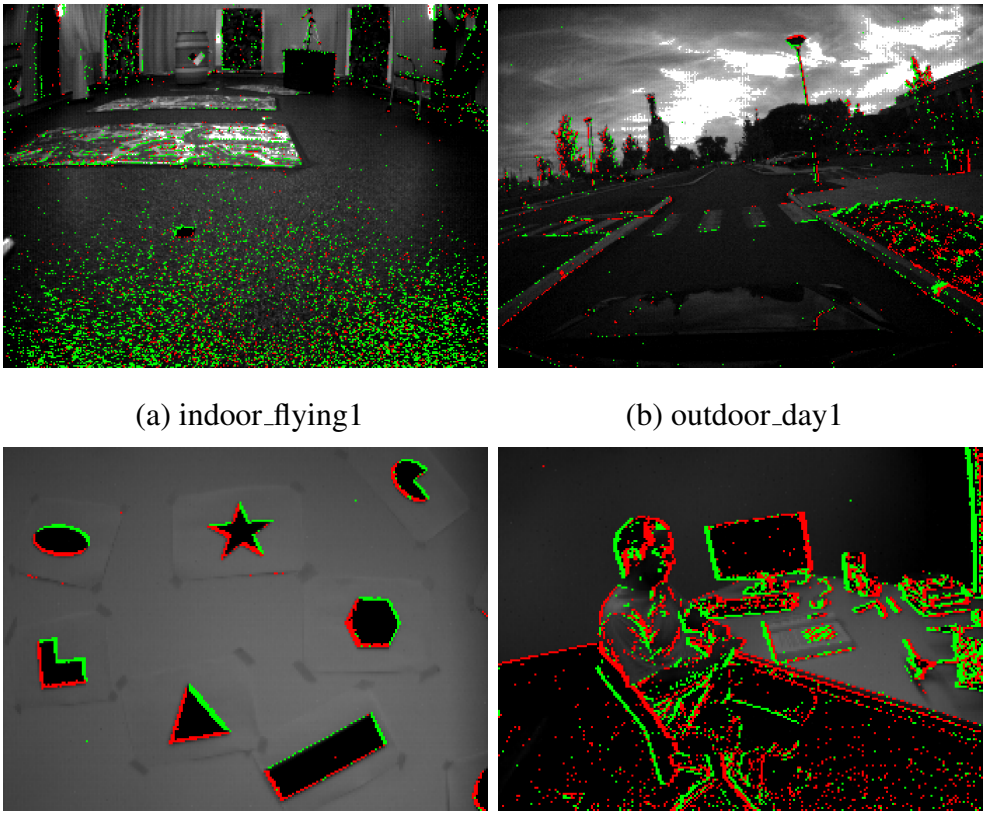
To handle asynchronous stream input, the proposed algorithm iteratively computes the above variation using all incoming optical flow, and estimates 3D angular velocity when $\det(\mathrm{Var}(\boldsymbol{\omega}))$ is less than the heuristic threshold value or the number of measurement n reaches the threshold, $n_{\max}$. The determinant of the covariance matrix is computed sequentially:

$$B_{1:n} = B_{1:n-1} + A_n^{\mathrm{T}} A_n / dt_n^2, \tag{4.6}$$

$$C_{1:n} = C_{1:n-1} + A_n^{\mathrm{T}} A_n, \tag{4.7}$$

$$\det(\mathrm{Var}(\boldsymbol{\omega}))_n = \sigma^2 \det(B_{1:n}) / \det(C_{1:n})^2, \tag{4.8}$$

where $B_{1:1} = A_1^{\mathrm{T}} A_1 / dt_1^2$ and $C_{1:1} = A_1^{\mathrm{T}} A_1$. To reduce the influence of optical flow belonging to dynamic object, I utilize RANSAC scheme to estimate accurate angular velocity.



(a) indoor_flying1 (b) outdoor_day1

(c) shapes_rotation (d) dynamic_rotation

Figure 4.4: Snapshot of tested sequences. Positive and negative events between two successive frames are overlaid with green and red dots, respectively, on the grayscale image.

4.4 Evaluation and Results

I evaluate optical flow and angular velocity estimation algorithms in terms of accuracy and latency. The accuracy of optical flow is validated on MVSEC [80] sequences while the accuracy of angular velocity is on DAVIS240C [60] sequences. Each snapshot of sequences is shown in Fig. 4.4. The latency of optical flow and angular velocity estimation algorithms, which is a main interest of this chapter, is computed by the optimization approach. Particularly, I compute the latency of optical flow algorithms indirectly via back-end angular velocity estimation, since it is difficult to compute the latency of optical flow directly. In evaluation, I set block width w as 25

px, the capacity of queue as 300 for optical flow, i.e., 150 events exist in a single local time slice. Also, the heuristic threshold for the determinant of the covariance of angular velocity is set to $0.001 \approx 0.3^6 \approx \sigma^2_{\omega_x}\sigma^2_{\omega_y}\sigma^2_{\omega_z}\text{rad}^6/\text{s}^6$, and the maximum number of optical flows n_{max} to 150 for angular velocity. These parameters are the same for all sequences with rich or poor texture of scene, fast or slow motion.

4.4.1 Latency Computation

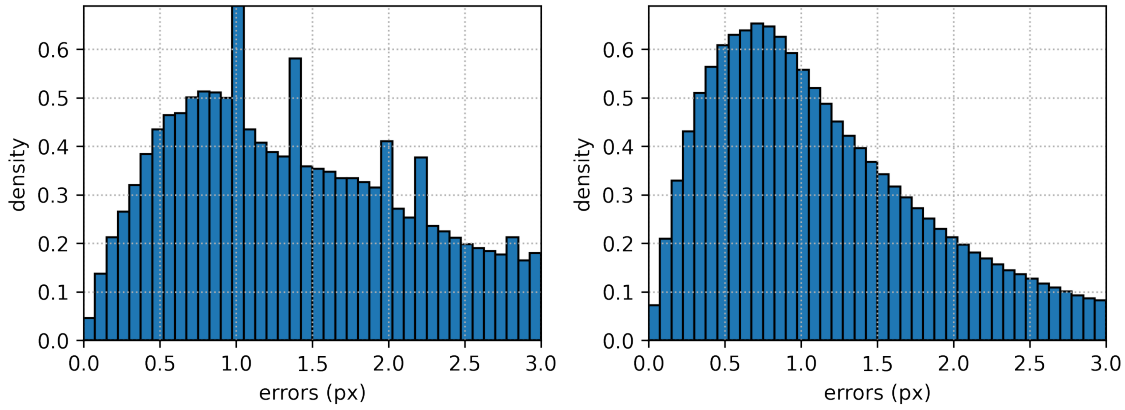
To estimate the latency of the proposed algorithms, I compare angular velocity estimates and ground-truth motions. Even though DAVIS240C sequences have mainly sinusoidal movements, the frequency spectrum of the angular velocity estimates is corrupted due to the estimation noise, and thus it is difficult to estimate the phase difference between estimates and ground-truth. Instead, I compute latency by minimizing the sum of squared errors between the estimates and ground-truth motions. Also, in order to focus on computing latency and consider overestimation or underestimation and bias, I add scale $A = aI_3 \in \mathbb{R}^{3\times3}$ and bias parameters $\mathbf{b} \in \mathbb{R}^{3\times1}$ for affine transformation and optimize below:

$$\min \sum_i w_i L_\delta \left(A\hat{\mathbf{x}}_{est}(\tau_i + \tau_d) + \mathbf{b} - \hat{\mathbf{x}}_{gt}(\tau_i)\right), \tag{4.9}$$

where τ_d is an important parameter meaning the latency which I pay attention to. τ_i is the i-th timestamp of estimates $\mathbf{x}_{est}$, and $\hat{\mathbf{x}}_{gt}(\tau_i)$ is an interpolated value of the ground-truth at τ_i. To reduce the sensitivity of outliers, I use the Huber loss function $L_\delta(\cdot)$. Also, to diminish the influence of high-frequency noise, the weight w_i is designed to be proportional to the magnitude of the slope of low-pass-filtered $\mathbf{x}_{gt}$. In evaluation, because the noise level and accuracy are different for each axis, I compute the latency of each angular velocity for each axis and report the maximum latency among them.

4.4.2 Optical Flow Estimation

I validate the performance of the low-latency optical flow algorithm qualitatively and quantitatively on MVSEC sequences. MVSEC provides multiple sensors' data including events and



(a) w/o compensation (b) w/ compensation

Figure 4.5: Histogram of end-point errors of optical flow vectors with or without vector compensation on indoor_flying1 sequence.

ground-truth flowmap of DVS as a form of a frame. Due to the frame-based ground-truth, I scale the magnitude of asynchronous optical flow vector of the proposed algorithm by the ratio between the time interval of flowmap and the time difference of optical flows. For example, the magnitude of optical flow with a small time difference is enlarged under the assumption that the optical flow is constant during the interval between successive flowmaps. Moreover, since the events are generated at the edge of a scene at which the true flowmap has discontinuities, the starting point of optical flow cannot be matched with the corresponding point of the flowmap exactly under a translational motion. Overall, to reduce faulty end-point errors caused by unsynchronized timestamps and discontinuity, I compensate the magnitude and the starting point of optical flows as below:

$$\hat{\mathbf{v}}_i = \mathbf{v}_i \times \frac{t_{gt,k} - t_{gt,k-1}}{dt_i}, \tag{4.10}$$

$$\hat{\mathbf{p}}_i = \mathbf{p}_i + \mathbf{v}_i \times \frac{t_{gt,k} - t_i}{dt_i}, \tag{4.11}$$

$$\text{s.t. } t_{gt,k-1} < t_i < t_{gt,k}, \ \forall i,$$

where $t_{gt,k}$ is the timestamp of the k-th true flowmap. Since the true flowmap represents optical flows based on the current scene, I validate asynchronous optical flows by projecting them onto

Table 4.1: Evaluations of optical flow on ground-truth flowmap.

Sequence	**Average end-point error (px)**				
	Proposed	ABMOF	Zhu et al. [31]	EV-FN	Spike-FN
indoor_flying1	1.09	1.28	0.58	1.03	0.84
indoor_flying2	1.76	1.98	1.02	1.72	1.28
indoor_flying3	1.54	1.74	0.87	1.53	1.11
outdoor_day1	2.43	2.88	0.32	0.49	0.49

the synchronized flowmap. The effect of asynchronous optical flow compensation is shown in Fig. 4.5. Thanks to the compensation using (4.10) and (4.11), the outliers in Fig. 4.5(a) caused by the faulty comparison disappear, thus making distribution smooth. The compensation not only banishes the outliers but also reduces errors overall.

For validation, I compare the proposed optical flow algorithm with ABMOF, EV-FlowNet (EV-FN) [80]. I implemented ABMOF in C++ by referring to java tools for Address-Event Representation (jAER) open-source project (http://jaerproject.org). However, ABMOF fails on MVSEC sequences because the sequences have a non-uniformly textured scene. Under such sequences, a slice duration adjustment module of ABMOF does not work properly, thus slowing down update rates and drastically reducing accuracy. Also, its outlier rejection discards a correct optical flow under fast motions in DAVIS240C sequences. Thus, I disable slice duration adjustment and outlier rejection. Instead, to validate the performance of ABMOF in a good condition, I test ABMOF with the same fixed number of events as the proposed algorithm, but other parameters are the same as the original version [30]. Additionally, I also tested LocalPlane proposed in [19], but its performance deteriorates in complicated realistic environments such as MVSEC sequences because its optical flow is computed from the slope of the local SAE plane and different from the true optical flow. Thus, I did not state the performance of LocalPlane in this chapter. The code of EV-FN is available in public, and I use them in evaluations.

I test optical flow algorithms on four sequences which are preferred for validation in ex-

Table 4.2: Evaluations for latency and accuracy of optical flow in static environments. Best results are in bold.

Sequence		**Average latency (ms)**			**Average RMSE**[1] **(rad/s)**		
		Proposed	ABMOF	EV-FN	Proposed	ABMOF	EV-FN
shapes	low	**14.40**	31.86	26.95	**0.110** (0.108)	0.147 (0.137)	0.269 (0.264)
	mid	**5.15**	9.76	20.56	**0.228** (0.195)	0.350 (0.256)	1.545 (1.524)
	high	**2.93**	7.09	20.48	**0.350** (0.310)	0.691 (0.563)	3.082 (3.039)
	whole	**3.04**	7.31	20.45	**0.296** (0.261)	0.548 (0.442)	2.350 (2.311)
boxes	low	**7.51**	13.12	19.55	**0.138** (0.134)	0.142 (0.136)	0.156 (0.146)
	mid	**1.86**	3.97	20.91	**0.297** (0.293)	0.361 (0.352)	1.199 (1.162)
	high	**0.59**	2.35	22.57	**0.355** (0.354)	0.380 (0.361)	2.666 (2.596)
	whole	**0.58**	2.27	21.74	**0.328** (0.327)	0.362 (0.346)	1.955 (1.903)
poster	low	**8.21**	10.39	16.01	0.238 (0.238)	**0.186** (0.185)	0.202 (0.190)
	mid	**2.14**	3.24	20.74	**0.303** (0.302)	0.347 (0.341)	1.291 (1.267)
	high	**0.52**	1.93	19.81	**0.359** (0.357)	0.409 (0.389)	2.906 (2.835)
	whole	**0.49**	1.87	19.91	**0.328** (0.327)	0.371 (0.356)	2.009 (1.960)
dynamic	low	**8.15**	8.41	26.02	**0.231** (0.229)	0.232 (0.227)	0.405 (0.391)
	mid	**4.35**	6.46	22.40	**0.242** (0.239)	0.272 (0.267)	0.539 (0.521)
	high	**1.80**	3.23	21.83	**0.289** (0.286)	0.341 (0.331)	1.525 (1.493)
	whole	**1.71**	3.19	21.97	**0.264** (0.261)	0.299 (0.291)	1.004 (0.980)

[1]The number in parentheses indicates the RMSE of zero-latency angular velocity that is calculated from (4.9).

isting literature [25, 31, 80]: indoor_flying1, indoor_flying2, indoor_flying3, and outdoor_day1. (4.6) and (4.7) show qualitative results of optical flow on MVSEC and DAVIS240C sequences. Grayscale images are shown for visualization purposes only and I do not use them to estimate optical flow. In ground-truth flow images, flow vectors with magnitude are displayed as a color wheel with values, whereas the results of the algorithms are colored by direction only to verify wrong results from a distance. Among them, although EV-FN produces a dense flowmap, for Fig. 4.6, I mask its result with a binary image indicating whether there are any events in that pixel, in order to compare with the others clearly.

In the first and second columns, ABMOF produces erroneous optical flows in the upper right quadrant, where events are not triggered enough due to long distance and little apparent movement. This is because ABMOF computes optical flow with the full-size time slices. Likewise, EV-FN fails to estimate accurate optical flow sometimes as shown in the second column of Fig. 4.6 and in the fourth row of Fig. 4.7. Because EV-FN constructs event frame and produces dense flowmap, it estimates accurate optical flow even on the long straight lines. However, as mentioned in the paper [80] as a limitation, it is hard to estimate the accurate optical flow in the fast speed (first column) and highly dense scenes (fourth column). Especially, in the fourth column, events are over-accumulated, thus the optical flows are under-estimated.

Quantitative results evaluated with image pairs one frame apart are shown in Table 4.1. The performance of the proposed algorithm on indoor sequences is comparable with the other algorithms but deteriorates in the outdoor sequence. This is because long straight lines belonging to crosswalks and lane boundaries cause the aperture problem in a local time slice of both the proposed algorithm and ABMOF as shown in Fig. 4.4(b). Also, whereas Zhu et al. [31], EV-FN, and Spike-FN output flowmap as a form of frame and the true flowmap is also provided as a frame synchronized with gray images for easy comparison, the proposed method and ABMOF generate asynchronous frame-less optical flow. Hence, the compensated output of the proposed algorithm and ABMOF, which are linearly extrapolated to the timestamp of the true flowmap, may degrade accuracies in this comparison.

To compute the latency of asynchronous optical flow stream, I utilize the result of back-

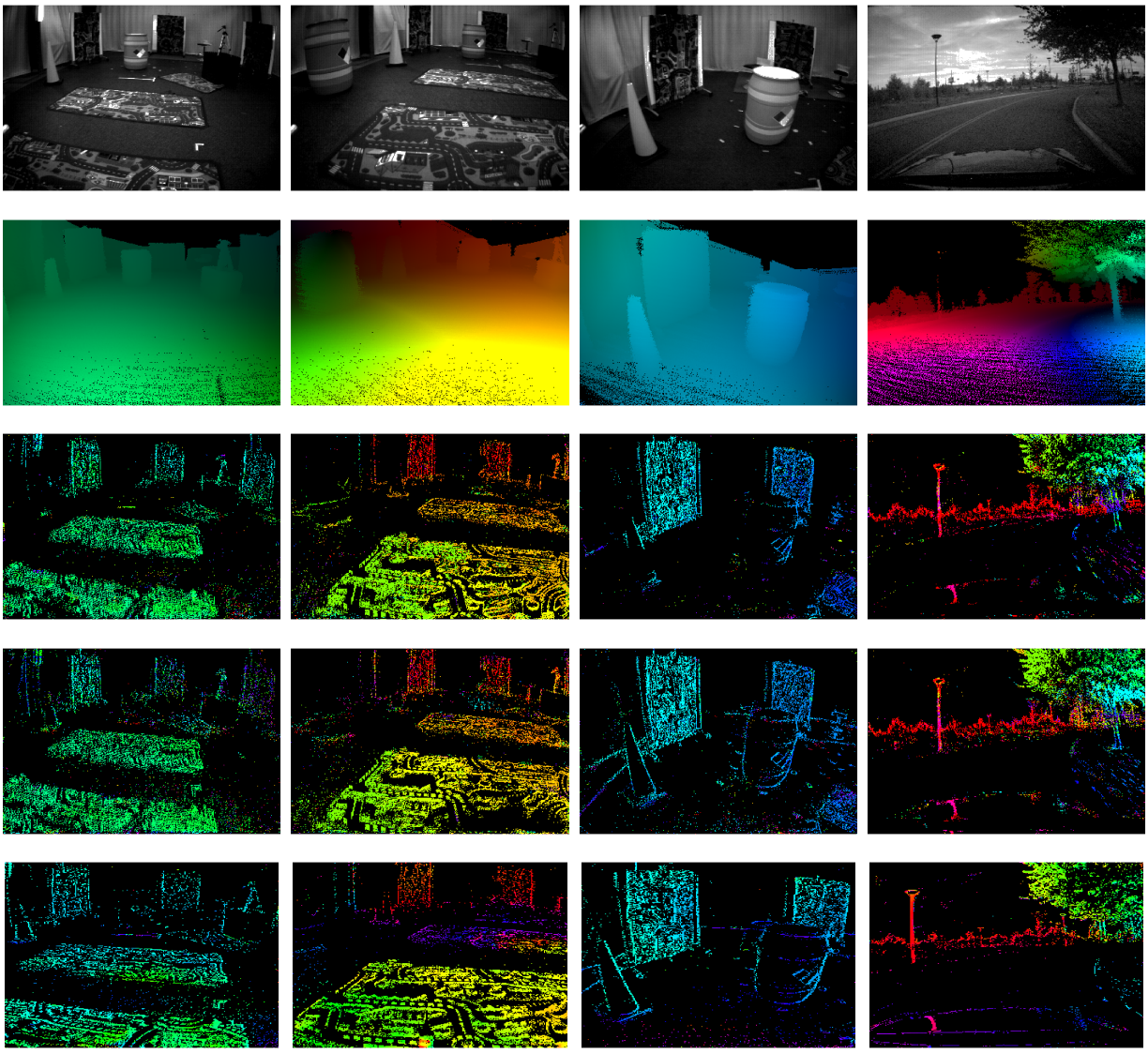


Figure 4.6: Qualitative results of optical flow on MVSEC sequences. Each column shows the flowmap results on the indoor_flying1, indoor_flying2, indoor_flying3, and outdoor_day1 sequences from left to right. Also, each row denotes gray image, ground-truth flowmap, the proposed algorithm, ABMOF, and EV-FN from top to bottom. Best viewed in color.

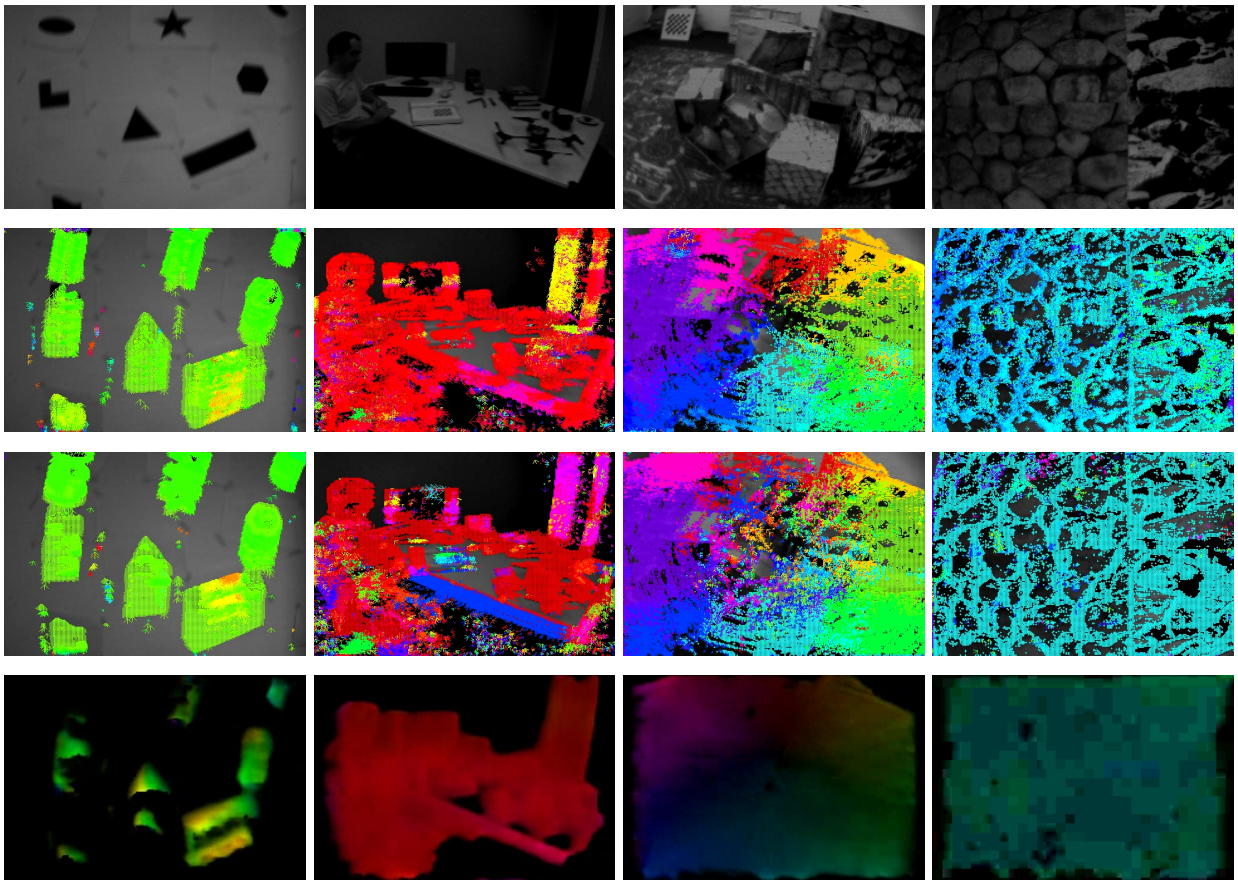
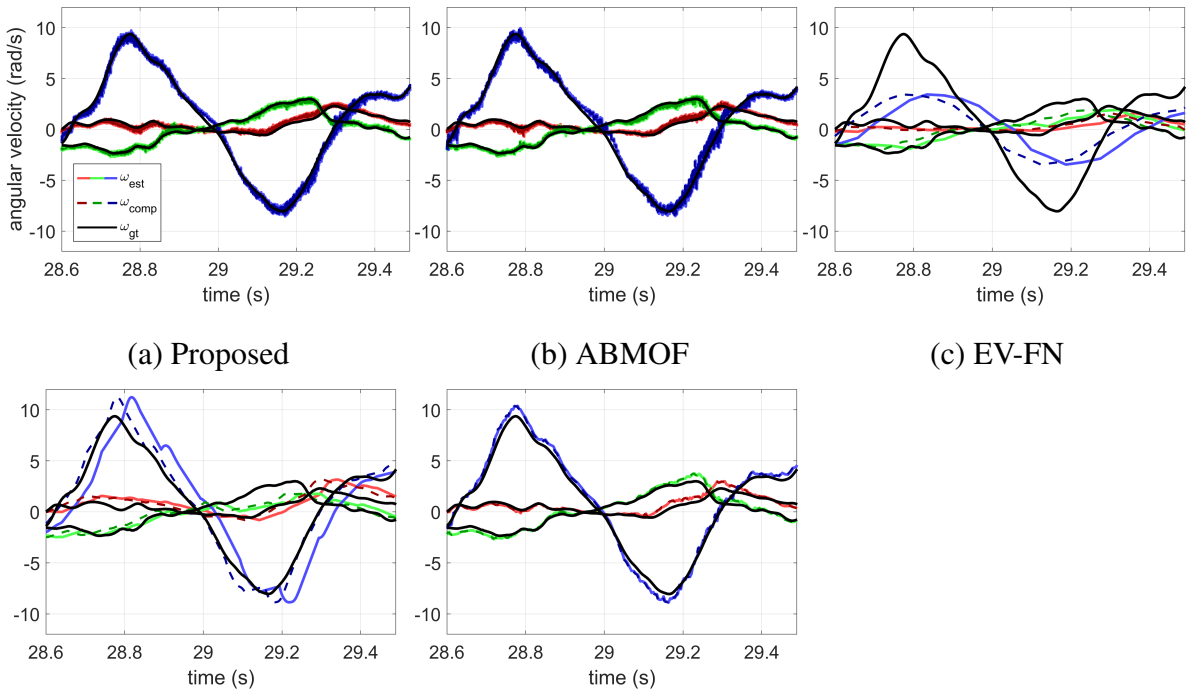


Figure 4.7: Qualitative results of optical flow on DAVIS240C sequences. Each column shows the flowmap results of the shapes_rotation, dynamic_rotation, boxes_rotation, and poster_rotation from left to right. Also, each row denotes gray image, the proposed algorithm, ABMOF, and EV-FN from top to bottom. In the poster_rotation sequence, the result of EV-FN is adjusted eight times brighter since it outputs a dark flowmap. Best viewed in color.



(a) Proposed (b) ABMOF (c) EV-FN

(d) eSNN (e) CM

Figure 4.8: Qualitative results of angular velocity on poster_rotation sequence of DAVIS240C. Figures show zoomed-in plots for $t \in [28.6, 29.5]$s. The ground-truth (black solid line), estimated (red, green, blue lines for ω_x, ω_y, ω_z), and zero-latency (dashed line) are compared. Best viewed in color.

end angular velocity estimation. The back-end process results in an additional latency of 0.02 (poster_rotation) or 0.2 (shapes_rotation) ms in average, but this value is added equally to the compared algorithms, so it does not affect the performance comparison and is not subtracted in Table 4.2. For the analysis of latency, I validate algorithms on DAVIS240C sequences, and Table 4.2 outlines the average latency and accuracy of the angular velocity estimation algorithm mentioned in Section 4.3, which fetches the output of the compared optical flow algorithms. Average RMSE is calculated between the estimated values and the angular velocity obtained from the gyroscope [28]. Since the latency of the event camera is affected by the speed of motion, I manually divide each sequence into three sections according to the degree of movement. The

fourth row of each sequence represents an execution over the whole sequence. The proposed algorithm is superior to the other algorithms in terms of latency and accuracy. ABMOF shows slightly worse accuracy than the proposed algorithm and has an additional latency of up to 15 ms. Meanwhile, as mentioned in their paper [80], EV-FN loses accuracy when the camera moves fast due to bleeding edge. Also, it shows a relatively large consistent latency of approximately 20 ms, half of the shutter speed, as it accumulates events and configures event frames for neural networks. Summarizing Tables 4.1 and 4.2, the accuracy of the proposed algorithm is comparable with the existing algorithms while significantly reducing the latency. Also, Table 4.3 shows the effectiveness of parabolic fitting method that aims to enhance the preciseness of optical flow vector. The parabolic fitting method significantly improves the accuracy of the proposed algorithm and ABMOF while maintaining a similar level of latency.

In comparing the proposed algorithm and EV-FN, since they estimate angular velocity from optical flow in the same way, the accuracy of angular velocity is affected by the quality of optical flow and latency. In order to eliminate the influence of latency and compare them in terms of the accuracy of optical flow, I calculate the zero-latency angular velocity $\hat{\mathbf{x}}_{est}(\tau + \tau_d)$ from (4.9) by time-shifting with the latency value, τ_d, and display its accuracy within parentheses in Table 4.2. Because the zero-latency estimates of EV-FN are still less accurate than the proposed algorithm, this analysis suggests that the proposed optical flow is more accurate than EV-FN on DAVIS240C sequences. As supported by the fact that the ground truth angular velocity with high temporal resolution evaluates optical flow well, the error of the proposed algorithm may have been lower than in Table 4.1 if the ground truth flowmap was provided with a high temporal resolution.

4.4.3 Angular Velocity Estimation

For performance validation of angular velocity estimation, I utilize DAVIS240C sequences that are captured under rotational motion, since MVSEC does not provide rotation sequences. Then, I analyze average latency and accuracy as explained in Section 4.4.2. I compare the proposed algorithm to the event-based SNN for angular velocity regression (eSNN) [26] and the implemented version of CM frameworks [28].

Table 4.3: Evaluations for latency and accuracy of optical flow w/ or w/o parabolic fitting in static environments. Best results in each algorithm (w/ or w/o) are in bold.

Sequence		**Average latency (ms)**				**Average RMSE (rad/s)**			
		Proposed		ABMOF		Proposed		ABMOF	
		w/	w/o	w/	w/o	w/	w/o	w/	w/o
shapes	low	16.79	**14.40**	32.16	**31.86**	**0.098**	0.110	**0.128**	0.147
	mid	6.14	**5.15**	9.99	**9.76**	**0.225**	0.228	**0.343**	0.350
	high	3.77	**2.93**	7.31	**7.09**	**0.349**	0.350	**0.662**	0.691
	whole	3.89	**3.04**	7.52	**7.31**	**0.294**	0.296	**0.526**	0.548
boxes	low	**6.63**	7.51	**11.61**	13.12	**0.122**	0.138	**0.114**	0.142
	mid	2.22	**1.86**	**3.84**	3.97	**0.178**	0.297	**0.189**	0.361
	high	0.75	**0.59**	**2.15**	2.35	**0.212**	0.355	**0.230**	0.380
	whole	0.72	**0.58**	**2.117**	2.27	**0.193**	0.328	**0.198**	0.362
poster	low	8.37	**8.21**	10.91	**10.39**	**0.201**	0.238	**0.167**	0.186
	mid	2.41	**2.14**	3.55	**3.24**	**0.246**	0.303	**0.223**	0.347
	high	0.56	**0.52**	**1.89**	1.93	**0.263**	0.359	**0.279**	0.409
	whole	0.53	**0.49**	**1.85**	1.87	**0.252**	0.328	**0.252**	0.371
dynamic	low	**8.14**	8.15	**8.23**	8.41	**0.180**	0.231	**0.161**	0.232
	mid	4.52	**4.35**	**6.31**	6.46	**0.191**	0.242	**0.208**	0.272
	high	1.97	**1.80**	4.21	**3.23**	**0.213**	0.289	**0.264**	0.341
	whole	1.90	**1.71**	4.17	**3.19**	**0.199**	0.264	**0.207**	0.299

Table 4.4: Evaluations for latency and accuracy of angular velocity in static environments. Best results are in bold.

Sequence		**Average latency (ms)**			**Average RMSE (rad/s)**		
		Proposed	eSNN	CM	Proposed	eSNN	CM
shapes	low	**14.40**	31.54	63.42	**0.110**	0.463	0.203
	mid	**5.15**	35.59	20.05	**0.228**	1.291	0.678
	high	**2.93**	35.93	14.57	**0.350**	2.578	1.154
	whole	**3.04**	35.99	14.42	**0.296**	1.972	0.987
boxes	low	7.51	45.49	**5.51**	**0.138**	0.447	0.145
	mid	1.86	39.90	**0.39**	**0.297**	1.333	0.380
	high	0.59	38.00	**0.26**	**0.355**	2.451	0.396
	whole	0.58	38.31	**0.18**	**0.328**	1.856	0.378
poster	low	8.21	39.74	**7.37**	**0.238**	0.458	0.250
	mid	2.14	40.31	**1.60**	**0.303**	1.163	0.463
	high	0.52	39.29	**0**	**0.359**	2.901	0.461
	whole	0.49	39.40	**0**	**0.328**	1.998	0.443
dynamic	low	**8.15**	46.36	9.68	0.231	0.594	**0.149**
	mid	**4.35**	43.67	5.63	0.242	0.628	**0.160**
	high	**1.80**	41.69	2.44	0.289	1.314	**0.183**
	whole	**1.71**	41.67	2.32	0.264	0.939	**0.169**

In [26], they train SNN with simulated event stream with an interval of 100 ms and do not penalize the error during the settling time of 50 ms. Because the regression time of 50 ms is too short to compute latency under the slow motion, I divide test sequences by 100 ms intervals and merge the angular velocity results of each interval. Then, I compute latency and accuracy using only the outside of settling time, that is, 50% of the whole sequence. Also, since they provide test code only, I use their open-source model pre-trained with a time step of one millisecond. On the other hand, [28] fetches 15000 events in a single step and this original version is denoted as CM here. However, their performance depends on the event grouping method or how many events are stacked. I will discuss the influence of parameters in Section 4.5.1.

Fig. 4.8 shows the angular velocity estimates of the compared algorithms. As can be seen from the time gap between dashed and solid lines, EV-FN and eSNN show large latency than the others. For reference, I omit estimates during the settling time in the plot of eSNN. Since poster_rotation sequence has huge amounts of textures, CM also estimates angular velocity with a negligible latency like the proposed algorithm and ABMOF. In Table 4.4, the proposed algorithm estimates angular velocity with consistently low latency and high accuracy for all sequences. Although CM shows better performance in terms of latency on boxes_rotation and poster_rotation sequences that are captured in front of rich textures, its latency is large on shapes_rotation and dynamic_rotation sequences: shapes_rotation and dynamic_rotation sequences were collected in front of texture-less simple shapes and natural office environment, respectively. For the same reason, because dynamic sequence includes long straight lines at the boundaries of walls or doors as shown in Fig. 4.7, the performance of the proposed algorithm is slightly degraded, whereas CM that uses batch processing shows the best results in that sequence. This is a fundamental limitation of the proposed algorithm, related to the aperture problem, and it can be resolved by increasing the block size but the computational efficiency is compromised. Overall, in the zoomed-in plot (Fig. 4.1), the proposed algorithm shows much lower latency and higher accuracy than the other algorithms in a low texture environment.

4.5 Discussion

4.5.1 Robustness

For validating robustness against texture and motion speed which are observed in DAVIS240C sequences, I test four more algorithms in Table 4.5: Kalman-Filter-based angular velocity estimation using the proposed asynchronous optical flow stream (KF), the proposed angular velocity estimation using ABMOF with the number of events, 100, that is fewer than ABMOF in Table 4.2 (ABMOF_*), and another two versions of CM framework ($\text{CM}_{5\text{k}}$, $\text{CM}_{50\text{ms}}$). $\text{CM}_{5\text{k}}$ fetches a less number of events, 5000, that is heuristically chosen to have similar latency to the proposed algorithm on shapes_rotation sequence. On the other hand, $\text{CM}_{50\text{ms}}$ uses time windows of 50 ms to collect events. In the table, latency and errors lower than the proposed algorithm are shown in bold, and (-) denotes failure case whose error increases monotonically.

For KF, I implement a robust random-walk-model-based Kalman filter with a 6-dimensional state including angular acceleration based on [84]. As mentioned in Section 4.3, latency is larger than the proposed least-squares-based algorithm, but showing lower accuracy due to the proposed accurate optical flow. A Kalman filter with a 3-dimensional state including rotation only has also similar results.

In the case of ABMOF_*, it constructs time slices with a fewer number of events than ABMOF, to decrease the latency of ABMOF tested previously. Consequently, its latency is reduced, but its error becomes large overall. Like between ABMOF and ABMOF_*, $\text{CM}_{5\text{k}}$ loses accuracy on dynamic_rotation sequence, even though its latency is decreased. It is noteworthy that $\text{CM}_{5\text{k}}$ fails on boxes_rotation and poster_rotation sequences that require more events due to the rich texture. Conversely, $\text{CM}_{50\text{ms}}$ shows consistent latency but its accuracy degrades on the fast section of each sequence. When I test $\text{CM}_{5\text{ms}}$ to further reduce latency, the accuracy becomes much worse.

Table 4.6 shows the evaluation results for more diverse CM frameworks. CM with more than 5k events has a greater latency, and its accuracy varies highly depending on the scene. Conversely, CM with less than 5k events has lower latency, but the accuracy is degraded and the algorithm

Table 4.5: Evaluations with different parameters. Results better than the proposed algorithm (in Table 4.4) are in bold.

Sequence		**Average latency (ms)**				**Average RMSE (rad/s)**			
		KF	ABMOF$_*$	CM$_{5k}$	CM$_{50ms}$	KF	ABMOF$_*$	CM$_{5k}$	CM$_{50ms}$
shapes	low	16.47	20.16	16.40	28.54	0.126	0.140	0.138	0.136
	mid	5.31	6.13	**5.14**	23.95	0.952	0.329	0.285	0.758
	high	-	3.79	**2.91**	26.94	-	0.454	0.397	1.960
	overall	-	3.84	**2.93**	26.56	-	0.401	0.353	1.450
boxes	low	12.16	11.27	-	22.92	**0.095**	0.333	-	**0.123**
	mid	3.28	2.60	-	26.66	**0.204**	0.511	-	0.742
	high	1.33	0.70	-	-	**0.263**	0.566	-	-
	overall	1.26	0.67	-	-	**0.239**	0.537	-	-
poster	low	14.25	**4.03**	-	23.52	**0.124**	0.484	-	**0.144**
	mid	3.31	**1.63**	-	24.90	**0.218**	0.498	-	0.654
	high	1.54	0.81	-	-	**0.283**	0.619	-	-
	overall	1.50	0.76	-	-	**0.250**	0.570	-	-
dynamic	low	15.28	**5.36**	-	27.85	**0.167**	0.502	-	**0.213**
	mid	10.36	**3.63**	**1.53**	25.04	**0.216**	0.608	1.280	0.263
	high	4.24	**1.71**	**0**	23.34	0.394	0.678	2.637	0.738
	overall	4.22	**1.56**	**0**	23.39	0.309	0.616	2.641	0.491

fails to estimate angular velocity successfully in a rich-textured environment. On the other hand, CM with longer than 50 ms fails to minimize contrast due to edge bleeding, and CM with shorter than 50 ms suffers from the lack of events in a rich-texture environment. On the contrary to the above algorithms that utilize the image of stacked events, the proposed approach shows reliable performance in various environments without fine-tuning parameters depending on the scene.

4.5.2 Future Work

In summary, to reduce the theoretical latency of optical flow estimation, I construct local time slices for all pixels while increasing memory requirement and computational cost. In comparison with ABMOF, the proposed approach increases the computational time by about 25 times, to 6.5kev/s (kilo events per second), on i7 laptop without GPU computing. However, since adjacent queues fetch the same event at the same time for each incoming event as shown in Fig. 4.3, parallel programming can enhance computation efficiency.

4.6 Summary

In the chapter, I aim to decrease the theoretical latency which is one of the important characteristics of event cameras, leading to short response time and high accuracy. In particular, I estimate asynchronous optical flow stream, and 3D angular velocity with low-latency to compute the latency of optical flow quantitatively. Contrary to the previous works, the proposed algorithm builds and maintains local time slices for every pixel in the form of a queue, thus generating optical flow that is independent of one another, like the event stream. Moreover, these highly informative optical flows can provide the exact analytic solution of angular velocity, thus satisfying low-latency. The overall evaluations suggest that the proposed algorithm shows higher accuracy than the previous works while reducing latency significantly. Besides, the accuracy and latency of the proposed algorithm are more consistent than other existing algorithms, regardless of the degree of texture and the speed of the camera. In particular, the latency has been reduced significantly in an environment with low texture. In summary, the proposed algorithm produces

asynchronous data stream like a DVS camera, but the significance of the chapter is that the container of the output is an optical flow having high-level information than an event. The proposed asynchronous optical flow stream can also perform the same role as optical flow in traditional computer vision problems such as object tracking, motion segmentation, and motion estimation. Thus, the proposed event-based optical flow stream can be utilized to handle visual perception problems for agile robotic systems and estimate the motion of the system with very low latency, especially even in texture-less environments such as indoor corridors.

Table 4.6: Evaluations for latency and accuracy of modifications of CM framework. The second row denotes the event grouping method of each CM framework. Results better than the proposed algorithm (in Table 4.4) are in bold.

Sequence		**Average latency (ms)**								**Average RMSE (rad/s)**							
		10ms	50ms	0.1s	0.2s	2.5k	5k	10k	15k	10ms	50ms	0.1s	0.2s	2.5k	5k	10k	15k
shapes	low	**7.93**	28.54	53.28	104.11	**5.62**	16.40	36.08	63.42	0.816	0.136	0.177	0.316	0.184	0.138	0.154	0.203
	mid	**2.56**	23.95	65.93	126.74	**1.46**	**5.14**	12.62	20.05	0.276	0.758	2.012	3.404	0.372	0.285	0.451	0.678
	high	**2.55**	26.94	73.89	-	**0.18**	**2.91**	8.54	14.57	0.351	1.960	4.909	-	0.491	0.397	0.710	1.154
	overall	**2.55**	26.56	72.52	-	**0.20**	**2.93**	8.56	14.42	0.450	1.450	3.649	-	0.441	0.353	0.615	0.987
boxes	low	8.16	22.92	49.58	-	-	-	**3.74**	**5.51**	0.966	**0.123**	0.197	-	-	-	1.345	0.145
	mid	2.76	26.66	-	-	-	-	**0.15**	**0.39**	0.301	0.742	-	-	-	-	0.994	0.380
	high	2.83	-	-	-	-	-	**0.51**	**0.26**	**0.234**	-	-	-	-	-	0.943	0.396
	overall	2.79	-	-	-	-	-	**0.40**	**0.18**	0.520	-	-	-	-	-	0.993	0.378
poster	low	-	23.52	46.65	92.32	-	-	**7.93**	**7.37**	-	**0.144**	**0.207**	**0.236**	-	-	1.464	0.250
	mid	-	24.90	-	-	-	-	**2.07**	**1.60**	-	0.654	-	-	-	-	1.523	0.463
	high	-	-	-	-	-	-	**0**	**0**	-	-	-	-	-	-	1.100	0.461
	overall	-	-	-	-	-	-	**0**	**0**	-	-	-	-	-	-	1.290	0.443
dynamic	low	**5.22**	27.85	52.73	-	-	-	**4.85**	9.68	0.348	**0.213**	0.353	-	-	-	**0.185**	**0.149**
	mid	**2.58**	25.04	46.84	-	-	**1.53**	**2.76**	5.63	**0.231**	0.263	0.468	-	-	1.280	**0.203**	**0.160**
	high	2.63	23.34	-	-	-	**0**	**0.81**	2.44	**0.166**	0.738	-	-	-	2.637	**0.253**	**0.183**
	overall	2.60	23.39	-	-	-	**0**	**0.83**	2.32	**0.266**	0.491	-	-	-	2.641	**0.225**	**0.169**

5

Robust Angular Velocity Estimation in Dynamic Environments

Event cameras are bio-inspired sensors that detect the brightness changes of pixels individually, which is originated from apparent scene motion. Due to the characteristic of event cameras, even fast moving objects are easily detected from a stationary event camera. However, at the same time, moving objects may dominate the output of event camera and thus deteriorate motion estimation performance of moving event cameras. In the paper, I propose a robust event-based angular velocity estimation in dynamic environments where moving objects exist. To distinguish between static and dynamic part in image frame, I exploit rigid-motion models for grid-based cells and update their models by asynchronous optical flow stream. I demonstrate angular velocity estimation with self-collected datasets published online. The evaluation results suggest that the proposed algorithm estimates angular velocity robustly, and its accuracy is superior to the other algorithm even in the existence of moving object.

5.1 Introduction

Event cameras, also known as silicon retina or dynamic vision sensors, respond to the pixel-wise brightness change. These sensors output the stream of a single "event" packet with position at which pixel's brightness increases or decrease. Thus events are mainly triggered by the apparent motion which may be caused by camera's motion or moving objects' motion. This characteristic makes easy to detect moving object in front of camera.

As compared with conventional cameras, event cameras have some advantages: low-latency, high temporal resolution, and high dynamic range. These features of event cameras are results from their asynchronous operation and suitable for agile robotic systems to achieve visual perception. Nevertheless, sparse and asynchronous event stream can not be directly utilized for the frame-based traditional robotics vision applications such as motion segmentation and estimation. Some research process a number of events simultaneously to construct an event frame and apply the frame-based algorithms. However, grouping events approach compromises the benefits of event cameras (low-latency and high temporal resolution) and the number of events must be set heuristically.

In this paper, I estimate angular velocity on an event-by-event basis, which takes advantage of event cameras, low-latency and high temporal resolution. Currently, existing event-based algorithms tried to address the fundamental computer vision problems such as motion segmentation [36, 38], angular velocity estimation [28, 35], optical flow [30, 31, 80], etc. However, most of them have assumed stationary environments where moving objects are not exist and have evaluated their performance on well-defined datasets, e.g., DAVIS240C, MVSEC. However, real environments include a various dynamic situation such as human interaction, other operating machines nearby. Particularly, data-driven cameras, like event cameras, generate large amounts of events on moving objects, so it is necessary to distinguish dynamic foregrounds from static backgrounds. Hence, I aim for robust angular velocity estimation in dynamic environments where moving object is exist.

To estimate angular velocity robustly and respond to apparent motions, I exploit the previ-

ously proposed research, low-latency and scene-robust optical flow estimation. By using asynchronous optical flow stream and the proposed dual-mode motion models, the proposed algorithm estimates angular velocity of ego-motion robustly in dynamic environments while separating dynamic foregrounds from an image without prior information, e.g., shape and the number of objects. To validate the performance of algorithms, I collected a dataset that includes moving objects at the office and road since there was no such dataset to the best of my knowledge. In the evaluation, the proposed algorithm shows higher accuracy in angular velocity estimation than other algorithms while successfully segmenting dynamic foreground in a static background.

5.1.1 Related Work

Similar to the most existing visual SLAM and visual odometry algorithms that assume that the world the camera is looking at is stationary, most existing event-based motion estimation algorithms [4, 23, 26, 28, 35, 82] assume stationary environments. However, since the event camera is a kind of data-driven sensors, even small but fast object can dominate a large portion of event stream, thus deteriorating the performance of motion estimation. Recently, some research address the detection of an moving object, and the successful detection of dynamic foreground is related to the performance of motion estimation which utilizes the remained region of static backgrounds.

Event-based Motion Segmentation by Motion Compensation (EMSMC) [36] segments moving objects by Expectation–Maximization (EM) algorithm. They update soft association matrix between the K pre-initialized motions and event packet. Then, E-step refines the association probability using a closed-form partitioning law and their M-step update motion parameter of clusters by maximizing the weighted sum of contrast of IWE. For initialization, they exploit their previous work, i.e., Simultaneous Optical Flow And Segmentation (SOFAS) [85], and estimate the motion parameter of a total of N_l clusters. In the paper, I have implemented EMSMC algorithm to compare with the proposed angular velocity estimation algorithm.

Similar to EMSMC, Y. Zhou et al. [5] utilizes the original version of contrast maximization framework (CM) [28] and EM algorithm whose E-step updates labels minimizing the designed energy terms and M-step refines the motion parameter of clusters given motion model. Because

they construct spatio-temporal graph of events and estimate the labels based on the graph connectivity of events, spatio-temporally smooth labels are obtained.

A. Mitrokhin et al. [38] proposes moving object detection and tracking. They revised contrast maximization framework [28] by introducing event-count image and time-image to compensate four dimensional motion: x-shift, y-shift, zoom, roll rotation. Through their global motion segmentation procedures, the four-parametrized motion of static backgrounds is estimated iteratively. Then, moving objects are detected as an area having large value of time-image as similar to [86]. Their second work [49] presents learning approach for motion segmentation. A depth network is trained in supervised approach with ground-truth depth, and pose network is trained with the mask information of objects. They predict the motion of moving objects and the camera with a mixture model. Another work [40] segments moving objects by exploiting PointNet++ [42] and training a Graph Convolutional neural network (GConv). They fetch a 3D graph of event stream in (x, y, t) space and infer the event cloud belonging to moving objects. But Gconv operates on large time intervals about several hundreds of milliseconds.

All of the above research construct event frame as an input of a motion compensation algorithm based on contrast maximization or a network. They stack events according to the number [5, 36] of timestamp of events [38, 40, 49], and these event-frame-based algorithms need to be tuned heuristically depending on the scene.

5.1.2 Contributions

The main contributions can be summarized as follows:

1. I propose a robust event-based angular velocity estimation algorithm in dynamic environments. It segments an image into dynamic foregrounds and static backgrounds on a grid-wise and estimates the motion of both parts.

2. I design a dual-mode motion model for asynchronous optical flow stream. It separates the dynamic foregrounds from the static backgrounds and tracks them with specified IDs without prior information such as shape, number of them.

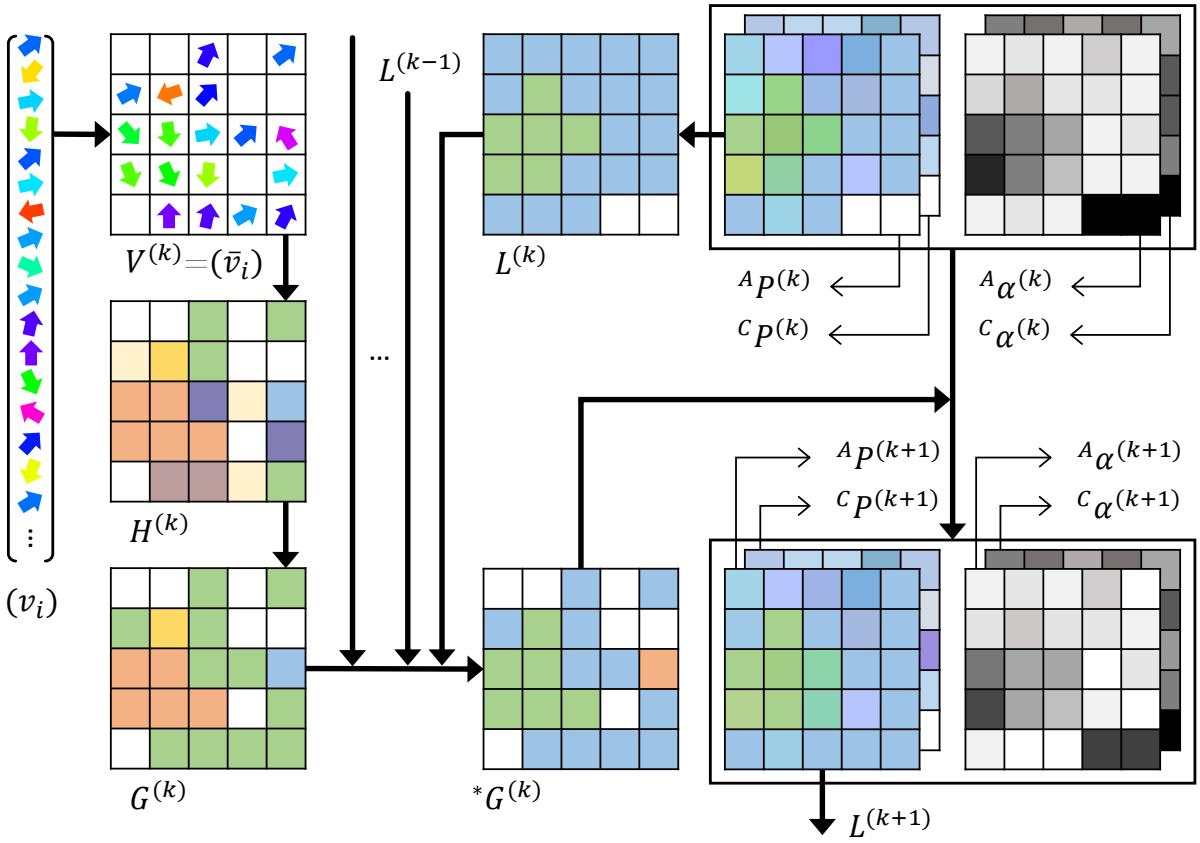


Figure 5.1: Pipeline description of the algorithm. Flowmap of grid-based optical flow $V^{(k)}$ is depicted based on a color wheel. For visual simplicity, optical flow stream are denoted as unit vectors and $c_{grid} = r_{grid} = 5$. $H^{(k)}$ and $G^{(k)}$ are the result of motion hypothesis search and refinement. Motion temporal matching yields ${}^{*}G^{(k)}$ from the label $L^{(k)}$ and the past. An IDs in $H^{(k)}, G^{(k)}, {}^{*}G^{(k)}, L^{(k)}$ are indicated as eigen colors, whereas the probability P are represented by the mixture of eigen colors. In age map α, the brighter denotes the older.

5.2 Motion Segmentation

The proposed algorithm consists of two stages: motion segmentation and dual-mode motion model management. Motion segmentation is further divided into spatial segmentation and temporal matching. Motion spatial segmentation aims to segment the image frame depending on the rotation model. This stage does not consider temporal transition, but only spatial segmentation, thus each segment is not identified. In other words, it is unknown which segment in the current result corresponds to which segment in the previous spatial segment result. In this regard, the purpose of motion temporal matching is to identify the results of successive spatial segmentation. Further, in order to reduce the influence of erroneous segments and consider a situation in which a stationary object suddenly moves, dual-mode motion models update the probability of which model is dominated in that cell.

In the motion spatial segmentation procedure, I first fetch a total of n_{flow} asynchronous optical flow vectors and construct grid-based flowmap by averaging fetched optical flow vectors in each cells. Then, I randomly select m flow vectors nearby among valid cells as samples for RANSAC to generate motion hypothesis. Next, all flow vectors are validated by the motion hypothesis, iteratively, n_{hyp} hypotheses are found. Finally, these hypotheses is refined by inlier subset and is merged with similar parameters though Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Fig. 5.1 describes the details of the motion spatial segmentation process and brief dual-mode motion model management.

5.2.1 Event-based Optical Flow

In order to segment an image frame depending on the rotation model, I exploit low-latency and scene-robust optical flow stream, which is previously proposed. It is because a rotation model can be explicitly estimated using at least three optical flow vectors, whereas three or more events are required to estimate the model using the event just as it is. However, since the proposed optical flow stream is generated asynchronously like event stream, when I classify each incoming optical flow vector as static or dynamic by using filter-based approach, the motion of static part is easily

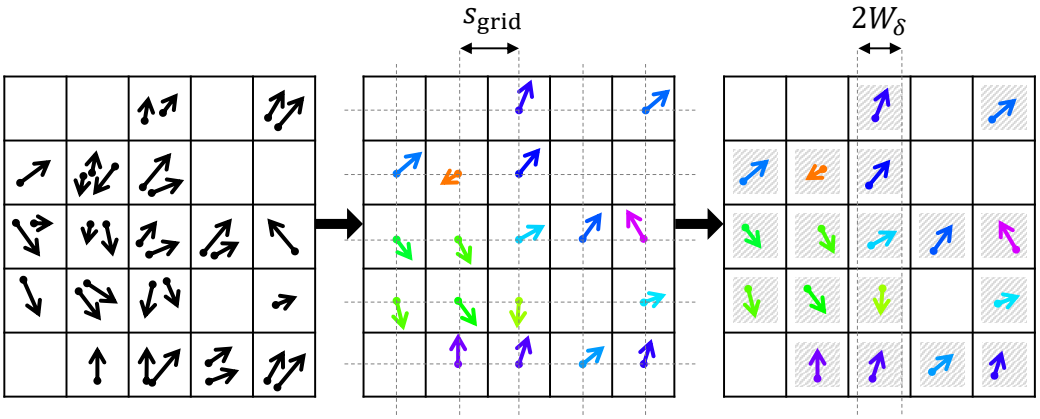


Figure 5.2: Construction of a grid-based optical flowmap. (left) raw optical flow stream, (middle) center-aligned flowmap, and (right) flowmap with intra-pixel-area approach.

corrupted by dynamic object and it is hard to recognize which event cluster belongs to the static backgrounds. Therefore, to segment incoming spatial information instantly, I divide an image frame of size $(w \times h)$ into a grid with a fixed spacing size s_{grid} in pixels, then collect incoming optical flow vectors into the corresponding cell including the pixel position of the vectors. After collecting a certain number of vectors, each cell takes an average vector on the collected optical flow vectors, $\bar{v}_i \in \mathbb{R}^2$. Therefore, the input of motion spatial segmentation is an flowmap of grid-based optical flow V and its elements are $\mathbf{v} = (\bar{v}_i)$, where n is the number of valid cells having an average optical flow vector and satisfies $0 \leq n \leq n_{cell} = c_{grid} \times r_{grid} = w/s_{grid} \times h/s_{grid}$ with the number of cells, n_{cell}.

As mentioned above, the optical flow of a cell v_i is obtained as the average of the optical flow observed inside the cell area. Additionally, the starting point of v_i should be compensated. If the optical flow of the cells are placed in the center of the cells, as shown in the middle of Fig. 5.2, the cell in which a single optical flow is observed shows wrong starting point. Accordingly, I adopt the intra-pixel-area (IPA) approach proposed in Chapter 3 to compute the starting point of v_i well described by a given rotation motion. The right figure in Fig. 5.2 describes the optical flow result

through IPA approach. The starting point of v_i of i-th cell, $\mathbf{x}_i^*$, are computed by

$$\mathbf{u}_i^* = \underset{\mathbf{u}}{\text{argmin}} \frac{1}{2} ||f(\boldsymbol{\omega}, \mathbf{x}_i + \mathbf{u}) - v_i||_2^2, \text{ s.t. } ||\mathbf{u}||_\infty \leq W_\delta, \tag{5.1}$$

$$\mathbf{x}_i^* = \mathbf{x}_i + \mathbf{u}_i^*, \tag{5.2}$$

where $\boldsymbol{\omega}$ is previously estimated rotation vector, $\mathbf{x}_i$ is the center position of the i-th cell, v_i is optical flow vector of the cell, and $f(\boldsymbol{\omega}, \mathbf{x})$ is a function computing optical flow with a given motion and position in the same manner as (4.3). Then, similar to (4.4), rotation vector is re-computed by

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}}{\text{argmin}} \sum_i \frac{1}{2} ||f(\boldsymbol{\omega}, \mathbf{x}_i^*) - v_i||_2^2. \tag{5.3}$$

At the implementation stage, $\boldsymbol{\omega}$ is estimated with center-aligned optical flow vectors firstly, and then I compute $\mathbf{u}^*$ of (5.1) by Newton's method instead of optimization method. by letting $f(\boldsymbol{\omega}, \mathbf{x}_i + \mathbf{u}_i) - v_i$ be zero, the equation is expressed as the below by 1^{st} order Taylor expansion:

$$\mathbf{e} = f(\boldsymbol{\omega}, \mathbf{x}_i + \mathbf{u}_i) - v_i \approx f(\boldsymbol{\omega}, \mathbf{x}_i) + \frac{\partial f}{\partial \mathbf{x}} \mathbf{u}_i - v_i = \mathbf{0}. \tag{5.4}$$

Thus,

$$\mathbf{u}_i = \left(\frac{\partial f}{\partial \mathbf{x}}^{\mathrm{T}} \frac{\partial f}{\partial \mathbf{x}} \right)^{-1} \frac{\partial f}{\partial \mathbf{x}}^{\mathrm{T}} (v_i - f(\boldsymbol{\omega}, \mathbf{x}_i)). \tag{5.5}$$

In addition, to limit the infinite norm of $\mathbf{u}$ below δ in (5.1), $\mathbf{u}_i$ is scaled by

$$\mathbf{u}_i^* = \mathbf{u}_i \min \left(1, \frac{W_\delta}{||\mathbf{u}_i||_\infty} \right). \tag{5.6}$$

5.2.2 Motion Hypothesis Search

In the search process, a point is randomly selected by weighting with min-error vector, $\mathbf{s} = (s_i) \in \mathbb{R}^n$, whose i-th value is defined as follows:

$$s_i = \min \left(\min_h e_{h,i}, 1 \right), \quad i = 1 \ldots n, \tag{5.7}$$

where

$$E = (e_{h,i}) = [\mathbf{e}_1, \ldots, \mathbf{e}_n] = \left[\text{dist}(\boldsymbol{\omega}_1, \mathbf{v}), \ldots, \text{dist}(\boldsymbol{\omega}_{n_{hyp}}, \mathbf{v})\right]^{\mathrm{T}} \tag{5.8}$$
$$= \begin{pmatrix} \text{dist}(\boldsymbol{\omega}_1, \bar{v}_1) & \cdots & \text{dist}(\boldsymbol{\omega}_1, \bar{v}_n) \\ \vdots & \ddots & \vdots \\ \text{dist}(\boldsymbol{\omega}_{n_{hyp}}, \bar{v}_1) & \cdots & \text{dist}(\boldsymbol{\omega}_{n_{hyp}}, \bar{v}_n) \end{pmatrix}$$

is an matrix representing distance error $e_{h,i}$ between the h-th hypothesis and flow vector of the i-th cell.

The min-error value measures how well the hypothesis is generated in the corresponding cell. A cell with the high min-error tends to belong to an inappropriate motion hypothesis, which results in a high probability to be chosen, consequently accelerating the convergence rate. The outer min operator prevents an cell with erroneous flow vector from being selected repeatedly with excessive probability. Thus, a one cell with high min-error value is selected with a high probability at the first. Then, I select $(m-1)$ vectors randomly near the chosen vector. Finally, the motion hypothesis $\boldsymbol{\omega}$ can be estimated from the selected m vectors using (5.3) and be inserted into hypothesis set Ω as a new hypothesis, iteratively. The result of motion segmentation using hypothesis search is denoted as H in Fig. 5.1

5.2.3 Motion Hypothesis Refinement and Clustering

Motion hypothesis refinement aims to calculate precise rotate vector and find multiple cells belonging to the same rotation motion. I first calculate distance error using (5.8) for optical flow vectors among all valid n cells. This process is designed to consider the situation where a part of a rigid object could appear in multiple regions on an image at the same time, in which case, I regard the multiple regions as belonging to the same object. Then, I re-estimate a motion hypothesis refined using the enlarged inlier set. For more details, please refer to [87].

In order to find distinct motions from the motion hypothesis set $\Omega = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{n_{hyp}}\}$, I utilize the DBSCAN, since DBSCAN does not require the number of clusters and is robust to noise. In the process, hypotheses are refined as the algorithm re-organizes n_{hyp} hypotheses into n_g distinct

motions, which is represented by integer matrix $G \in \mathbb{N}^{c_{grid} \times r_{grid}}$ whose elements are distributed from 1 to n_g in an natural number and $c_{grid} = w/s_{grid}, r_{grid} = h/s_{grid}$ is the dimension of grid cell. Then, $G^{(l)}$ denotes the motion spatial segmentation results at the l-th timestamp and its value means the index of the motion to which the corresponding pixel belongs.

5.2.4 Motion Temporal Matching

Motion spatial segmentation successfully classifies optical flow vectors depending on their rotate motions, however, it does not identify vectors whether they are belong to static backgrounds or dynamic foregrounds. In other words, the element value of $G^{(k-1)}$ does not correspond one-on-one to the same element value of $G^{(k)}$, and does not denote the same motion. Therefore, motion temporal matching method finds matching pairs set between the result of the previous motion spatial segmentation $G^{(k-1)}$ and the current result $G^{(k)}$.

The segment matching method calculates a kind of matching coefficient between two segment groups and finds corresponding segment pairs set $\hat{\mathcal{P}}$ between different timestamp. The matching coefficient $\mathcal{C}^{(k,l)}$ between spatial segmentation results at the previous k-th and the current l-th are computed from the constant position assumption. The constant position assumption means that distributions of the corresponding cell are not changed unexpectedly in a short time, thus I can consider computing a matching coefficient as intersection over union (IoU) method. However, IoU focuses on the overlap and coincidence of areas. Thus, IoU may yield a low value when the same object appears in different sizes as the result of spatial segmentation. This is possible because I fetch an asynchronous optical flow stream and the data depends on the velocity of an object. Therefore, I compute matching coefficient by

$$\frac{N_{i,j}^{(k,l)}}{\sqrt{N_i^{(k)} N_j^{(l)}}} \tag{5.9}$$

while reducing the influence of large segments by square root [87]. However, the data input of [87] and this algorithm are different in RGB-D-based dense flowmap and event-based sparse

flowmap. Consequently, I have revised (5.9) as

$$C_{ij}^{(k,l)} = \frac{\sum \left(K_\sigma * \delta(G^{(k)}, i)\right) \odot \delta(G^{(l)}, j)}{\sqrt{\sum \delta(G^{(k)}, i) \sum \delta(G^{(l)}, j)}}, \tag{5.10}$$

where the operator $*$ denotes 2-D convolution and $K_\sigma \in \mathbb{R}^{3\times 3}$ is a kernel matrix with all elements 1, which is designed to assign the same weight to adjacent cells. Also, the element of $B = \delta(A, x)$ with a matrix A and a scalar x is

$$B = (b_{ij}), \tag{5.11}$$
$$b_{ij} = \begin{cases} 1, & \text{if } a_{ij} = x, \\ 0, & \text{otherwise.} \end{cases}$$

and the operator $\odot$ denotes element-wise multiplication. Also, I consider the motion of segments as a kind of cost between two segments:

$$C_{ij}^{(k,l)} = \frac{1}{1 + ||\omega_i^{(k)} - \omega_j^{(l)}||_2^2}, \tag{5.12}$$

Thus, $\sum \delta(G^{(k)}, i)$ is the number of cells belonging to the i-th motion in the segmentation result at the k-th timestamp. Then, a matching pairs set is

$$\begin{aligned} \mathcal{P}^{(k,l)} &= \left\{ \mathcal{P}_i^{(k,l)} \right\} = \left\{ \left(\mathcal{P}_{i,prev}^{(k,l)}, \mathcal{P}_{i,curr}^{(k,l)} \right) \right\} \\ &= \left\{ \left(i,\ \underset{j}{\text{argmax}} C_{ij}^{(k,l)} \right) \right\}. \end{aligned} \tag{5.13}$$

When $\max_j C_{ij}^{(k,l)}$ is less than heuristic threshold th_{corr}, the i-th motion of the previous segmentation result is missed and/or the j-th motion of the current segmentation result is newly appeared. This j-th motion can be considered as the motion of new object or erroneous result of segmentation. To find the best matching pairs set, I compute correlation score $score^{(k,l)}$ between the previous k-th and the current l-th segmentation results as the average for $\max_j C_{ij}^{(k,l)}$ greater than th_{corr} only. Then, the best matching pairs set ${}^*\mathcal{P}^{(l)}$ at the current l-th segmentation is

$${}^*\mathcal{P}^{(l)} = \mathcal{P}^{(k_m,l)}, \tag{5.14}$$

where k_m is

$$k_m = \underset{l-t_m \leq k < l}{\operatorname{argmax}} score^{(k,l)}, \tag{5.15}$$

Finally, I rearrange index matrix $G^{(l)}$ to ${}^*G^{(l)}$ through ${}^*\mathcal{P}^{(l)}$ by substituting the value of $G^{(l)}$ with the second element of each corresponding pair. From now, I will call the index matrix ${}^*G$ as a label in the following paragraphs, while I have called the index matrix G as a segmentation result. Through this process, I control over-segmentation problem by merging over-segmented segments and track the object that appeared previously. Also, a new object with difference motion can be detected from the index of small correlation less than th_{corr}. Moreover, in order to distinguish between sparsely distributed noise or densely distributed object, I calculate a kind of spatial density for all segments:

$$V_i^{(l)} = \frac{\sum \left(K_\rho * \delta(G^{(l)}, i)\right) \odot \delta(G^{(l)}, i)}{\sum \delta(G^{(k)}, i)} \tag{5.16}$$

is the spatial density of the i-th segment at the l-th timestamp, and the kernel K_ρ is

$$K_\rho = \frac{1}{8}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.17}$$

to focus on measuring for density of mass.

5.3 Dual-mode Motion Model Management

Since the event camera output sparse and asynchronous event stream, the proposed optical flow estimation algorithm produces sparse and asynchronous optical flow stream. Although I collect several hundreds of optical flow vectors to classify them according on rotate motion hypotheses, the flowmap composed of collected vectors shows high sparsity. Because of the sparsity of the input data, simple kernel is applied in (5.10) and the spatial density in (5.16) is calculated to reject erroneous segment. In the paper, I apply the modified version of dual-mode simple Gaussian model [87, 88] to adapt to event-based asynchronous input data.

Dual-mode motion model consists of *apparent* and *candidate* models with a size of $c_{grid} \times r_{grid} \times n_{hyp}$ as the same with the $x, y-$size of ${}^{*}G^{(l)}$. The size of depth dimension of both models is the same with the maximum number of motion hypothesis, n_{hyp}. The depth dimensional vector $P_i^{(k)} \in \mathbb{R}^{n_{hyp} \times 1}$ denotes the probability vector of the i-th grid at the k-th timestamp. The h-th element of the probability vector $P_i^{(k)}$ represents the likelihood of the i-th cell belonging to the k-th motion. Both models have another parameter $\alpha_i^{(k)} \in \mathbb{R}$ representing the age of the i-th cell. The purpose of the candidate model and age parameter is to detect a new motion while distinguishing and rejecting erroneous segmentation result, whereas the apparent model aims to track the static backgrounds and the dynamic foregrounds robustly by help of the candidate model. In the following, I explain how both models are updated and detect dynamic foregrounds. Also, the probability vectors of both models are compensated by sparse optical flow vectors to update the corresponding cell denoting the same object regardless of local motion. This compensation method for sparse optical flow is described in the next place.

5.3.1 Motion Model Update

The basic principle is that when a label identical to the label of the current model is fetched, the corresponding model is updated. The label of the model of the i-th cell $L_i \in \mathbb{N}$ denotes the index of maximum value in the probability vector $P_i \in \mathbb{R}^{n_{hyp} \times 1}$:

$$L_i = \underset{1 \leq h \leq n_{hyp}}{\operatorname{argmax}} P_i(h) \tag{5.18}$$

Each model of the cell consists of a probability vector and an age, which are updated for certain conditions as mentioned below:

$$P_i^{(k+1)} = \frac{\tilde{\alpha}_i^{(k)}}{\tilde{\alpha}_i^{(k)} + 1} \tilde{P}_i^{(k)} + \frac{1}{\tilde{\alpha}_i^{(k)} + 1} vec({}^{*}G_i^{(k)}) \tag{5.19}$$

$$\alpha_i^{(k+1)} = \tilde{\alpha}_i^{(k)} + 1, \tag{5.20}$$

where $vec({}^{*}G_i^{(k)}) \in \mathbb{R}^{n_{hyp} \times 1}$ is a binary-valued vector whose ${}^{*}G_i^{(k)}$-th element only is one and the others are zeros, and ${}^{*}G_i^{(k)}$ is the result of segment matching algorithm in the k-th timestamp

as mentioned in Section 5.2.4. For example, $vec(2) = (0, 1, 0, 0, 0)$ for a brief case of $n_{hyp} = 5$. Also, $\tilde{P}_i^{(k)}$, $\tilde{\alpha}_i^{(k)}$ are the values of motion-compensated model, which will be discussed in Section 5.3.3.

However, under-segmentation and/or over-segmentation can occur and update wrong models. Similar motions of different objects can lead to an under-segmentation, and over-segmentation is mainly caused by the inaccurate motion of grid or non-rigid motion. Although asynchronous optical flow stream is validated in terms of accuracy in Chapter 4, averaging them in a grid may degrade the accuracy (see Section 5.2.1). Therefore, the candidate model complements the apparent model while preventing the erroneous segmentation from updating the apparent model directly. The probability vector, ${}^{A}P$, and age, ${}^{A}\alpha$, of the apparent model are updated as follows:

$$ {}^{A}P_i^{(k+1)} = \begin{cases} \frac{{}^{A}\tilde{\alpha}_i^{(k)}}{{}^{A}\tilde{\alpha}_i^{(k)} + 1} {}^{A}\tilde{P}_i^{(k)} + \frac{vec(G_i^{(k)})}{{}^{A}\tilde{\alpha}_i^{(k)} + 1}, & \text{if } G_i^{(k)} = {}^{A}L_i^{(k)}, \\ {}^{A}\tilde{P}_i^{(k)}, & \text{otherwise,} \end{cases} \tag{5.21} $$

$$ {}^{A}\alpha_i^{(k+1)} = \begin{cases} \min\left({}^{A}\tilde{\alpha}_i^{(k)} + 1, \alpha_{max}\right), & \text{if } G_i^{(k)} = {}^{A}L_i^{(k)}, \\ {}^{A}\tilde{\alpha}_i^{(k)}, & \text{otherwise.} \end{cases} \tag{5.22} $$

On the other hand, the probability vector, ${}^{C}P$, and age, ${}^{C}\alpha$, of the candidate model are updated as follows:

$$ {}^{C}P_i^{(k+1)} = \begin{cases} \frac{{}^{C}\tilde{\alpha}_i^{(k)}}{{}^{C}\tilde{\alpha}_i^{(k)} + 1} {}^{C}\tilde{P}_i^{(k)} + \frac{vec(G_i^{(k)})}{{}^{C}\tilde{\alpha}_i^{(k)} + 1}, & \text{if } G_i^{(k)} \neq {}^{A}L_i^{(k)}, \\ \mathbf{0}, & \text{else if } {}^{C}\alpha_i^{(k+1)} = 0 \text{ and } {}^{C}\alpha_i^{(k)} > 0, \\ {}^{C}\tilde{P}_i^{(k)}, & \text{otherwise,} \end{cases} \tag{5.23} $$

$$ {}^{C}\alpha_i^{(k+1)} = \begin{cases} \min\left({}^{C}\tilde{\alpha}_i^{(k)} + 1, \alpha_{max}\right), & \text{if } G_i^{(k)} = {}^{C}L_i^{(k)}, \\ \max\left({}^{C}\tilde{\alpha}_i^{(k)} - \tau \cdot (t^{(k+1)} - t^{(k)}), 0\right), & \text{otherwise,} \end{cases} \tag{5.24} $$

where $t^{(k)}$ denotes the k-th timestamp value, and ${}^{A}L^{(k)}, {}^{C}L^{(k)} \in \mathbb{N}^{c_{grid} \times r_{grid}}$ are the label of the apparent and candidate models at the k-th timestamp, respectively. τ is a parameter for decaying the age of the candidate model. The apparent model has a more simple update procedure than the

candidate model and its label ${}^{A}L^{(k)}$ updated at that time produces the result of motion segmentation. Contrary to the apparent model, the probability vector of the candidate model is updated for the input different from the label of the apparent model. By doing so, the candidate model detects different motions of a new object and, at some point, this new movement appeared in the apparent model by a swapping procedure as will be described. Erroneous segments generally do not reappear in the same location as the previous, the age of the candidate model whose label is not matched with the segmentation result is decayed and finally, the probability of that candidate model is initialized.

5.3.2 Motion Model Swapping

The two conditions that the apparent and candidate models are swapped with each other is when the age of the candidate model reaches the maximum age α_{max} or is larger than that of the corresponding apparent model. This model swapping is designed for preventing the apparent model from updating erroneous segments or missing new movements. In relation to the second case, let me consider the situation where a moving object just stops. The object is classified into dynamic foregrounds before and immediately after they stop. At that time, the label of the corresponding apparent model belongs to the dynamic foregrounds. After that, since the segmentation result implies that the object is static backgrounds, the corresponding apparent model does not updated whereas the candidate model is updated. Consequently, the age of the candidate model increases and then both models will be swapped with each other when conditions above are satisfied. The probability vector and age of both models are swapped as follows:

$$\begin{aligned} {}^{A}P_i^{(k+1)} &= {}^{C}\tilde{P}_i^{(k+1)}, \\ {}^{A}\alpha_i^{(k+1)} &= 0, \\ {}^{C}P_i^{(k+1)} &= \mathbf{0} \in \mathbb{R}^{n_{hyp}}, \\ {}^{C}\alpha_i^{(k+1)} &= 0. \end{aligned} \tag{5.25}$$

The output of the dual-mode motion model is derived from the probability vector of the apparent models with some criteria. I denote this output matrix as motion label, $M_i^{(k)} \in \mathbb{R}^{c_{grid} \times r_{grid}}$.

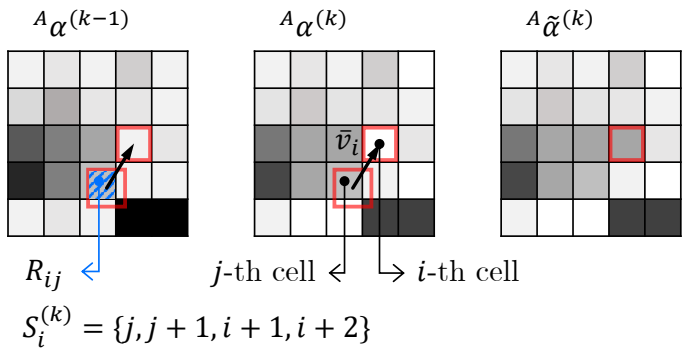


Figure 5.3: Description of how the parameters of cell are compensated. The average optical flow of i-th cell is $\bar{v}_i$, and then the parameters of i-th cell are compensated based on the parameters of cells that are elements of $S_i^{(k)}$.

The i-th element of motion label is updated as the label of the corresponding apparent label if the apparent model is updated or swapped. This partial update of motion label prevents the algorithm from prejudging the label of an unobserved or unmeasured motion of cell while preserving the previous motion label. In particular, since the event camera generates events with a static background only by its ego-motion, the motion label is maintained while there is no event. Under the assumption that the portion of the static background of the motion label is maintained continuously, the proposed algorithm estimates the angular velocity robustly.

5.3.3 Motion Model Compensation

In Section 5.3.1, the probability vector and age of both models are updated depending on the coincidence of the segmentation result and their label. These update criteria for motion model is based on the assumption that each cell represents a fixed point in the world coordinate consistently without considering the ego-motion of the camera. Consequently, the change in camera viewpoints causes the discrepancy between the segmentation result and the label of both models. Therefore, the parameters of both models are compensated through warping by sparse optical flow vectors.

To reflect the local optical flow on the model consisting of an probability vector and age value,

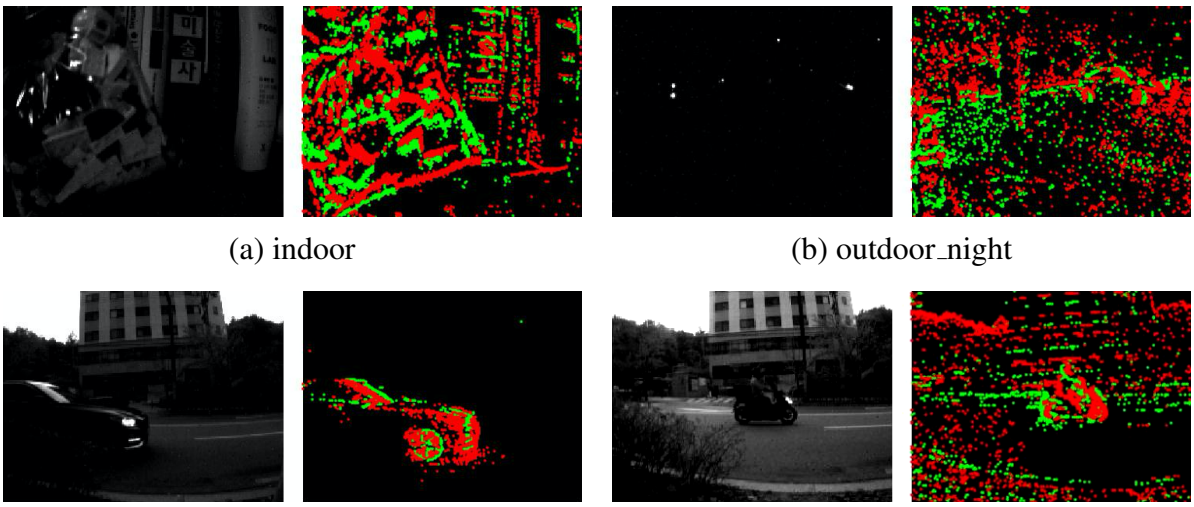
I use area-weighted interpolation. For cell that has valid optical flow vectors (see Section 5.2.1), the corresponding motion model are compensated individually on the two-dimensional. I consider a cell which has an valid optical flow vector, and this optical flow vector is calculated by the average of fetched optical flow vectors. As shown in Fig. 5.3, let this cell denote the i-th cell of apparent and candidate model in the current k-th timestamp. Then, the end-point of the reverse of its optical flow vector might indicate a sub-grid point in the previous $(k-1)$-th timestamp. So, the model parameter of the i-th cell is set to the interpolated parameter of the adjacency cells of that end-point. I denote the set of these adjacent cells overlapping with the i-th cell as $\mathcal{S}_i^{(k)}$ and an index j is an element of the set, i.e., $j \in \mathcal{S}_i^{(k)}$, and the maximum size of the set $|\mathcal{S}_i^{(k)}|$ is four since I deal with two dimensions. The overlapping area between the i-th cell and the j-th cell as R_{ij}. Also, I denote a weight for interpolation as ω_{ij} which are set to be proportional to R_{ij} and normalized to $\sum_j \omega_{ij} = 1$. Then, the compensated probability vector and age are calculated as follows:

$$\tilde{P}_i^{(k)} = \sum_{j \in \mathcal{S}_i^{(k)}} \omega_{ij} P_j^{(k-1)}, \tag{5.26}$$

$$\tilde{\alpha}_i^{(k)} = \sum_{j \in \mathcal{S}_i^{(k)}} \omega_{ij} \alpha_j^{(k-1)}. \tag{5.27}$$

5.3.4 Angular Velocity Estimation

To take advantage of event cameras, e.g., high temporal resolution, the proposed motion segmentation module and angular velocity estimation module of Chapter 4 are executed in parallel. As mentioned before, to segment image frame into static or dynamic, enough optical flow vectors should be collected into each cell. For better performance, the motion segmentation module requires a larger number of input data than the angular velocity estimation module. After the motion label is estimated, the angular velocity estimation module begins to collect optical flow vectors belonging to the static backgrounds, and computes the angular velocity of ego-motion with high temporal resolution through the criteria explained in Section 4.3.



(a) indoor (b) outdoor_night

(c) outdoor_day1 (d) outdoor_day2

Figure 5.4: Snapshot of the collected sequences. In each figure, a gray image and colorized event image are represented. Positive and negative events are drawn with green and red dots, respectively.

5.4 Evaluation and Results

I evaluate angular velocity estimation algorithm in terms of accuracy and latency. To the best of my knowledge, since there is no dataset which includes rigid moving object under rotational motion, I have collected dataset using DAVIS240C camera. For all tested sequences, the parameters of the proposed algorithm are the same: The number of sampling points and the searching radius in the motion spatial segmentation are set to $m = 3$ and $r_{search} = 2$ pixels. The parameters of DBSCAN are set to $p_{min} = 1$, $\epsilon = 0.5$. The number of fetched optical flow n_{flow} is 1000 and the spacing size of grid s_{grid} is 5 pixels. Also, the number of motion hypothesis n_{hyp} at initialization step is 15 and its minimum number of supported vectors is set to 15. For motion model management, I set α_{max} and τ to 3 steps and 50 s^{-1}, respectively. In the following, I describe the collected dataset.

Table 5.1: Descriptions of the dataset collected in dynamic environments.

Name	Duration	Description
indoor	17.6 seconds	Manual motion of planar object
outdoor_day1	5.4 seconds	Day, One vehicle, Camera stops at the end
outdoor_day2	2.4 seconds	Day, One motorcycle
outdoor_night	6.9 seconds	Night, Three vehicles

5.4.1 Dataset

In order to evaluate the proposed algorithm properly, I collected a dataset using DAVIS240C camera and its gyroscope for ground-truth. Each sequence is composed of event stream, 8-bit grayscale images with a size of 240×180, and IMU data. The dataset includes two types of moving objects: human's interaction or vehicles. The first type of sequence depicts a situation where a planar object moves in front of a camera in an office environment, and the second type is recorded in road environments. The snapshot of each sequence is displayed in Fig. 5.4. Also, in order to take advantage of event cameras, I included a sequence captured in night environment. Especially, to the best of my knowledge, the unique point about the collected dataset is that it contains a sequence in which the camera stops, that is outdoor_day1. Table 5.1 describes the details of the collected sequences and the download link of the dataset is available online at:

http://sangillee.com/_pages/larr-dvs-de-dataset

5.4.2 Angular Velocity Estimation

I evaluate robust angular velocity estimation, the previous work in Chapter 4, and the existing EMSMC algorithm in terms of latency and accuracy. The previous and the current algorithm have a common method to compute angular velocity, which are based on analytic solution. But, as the current algorithm is newly proposed in this chapter, the proposed algorithm computes

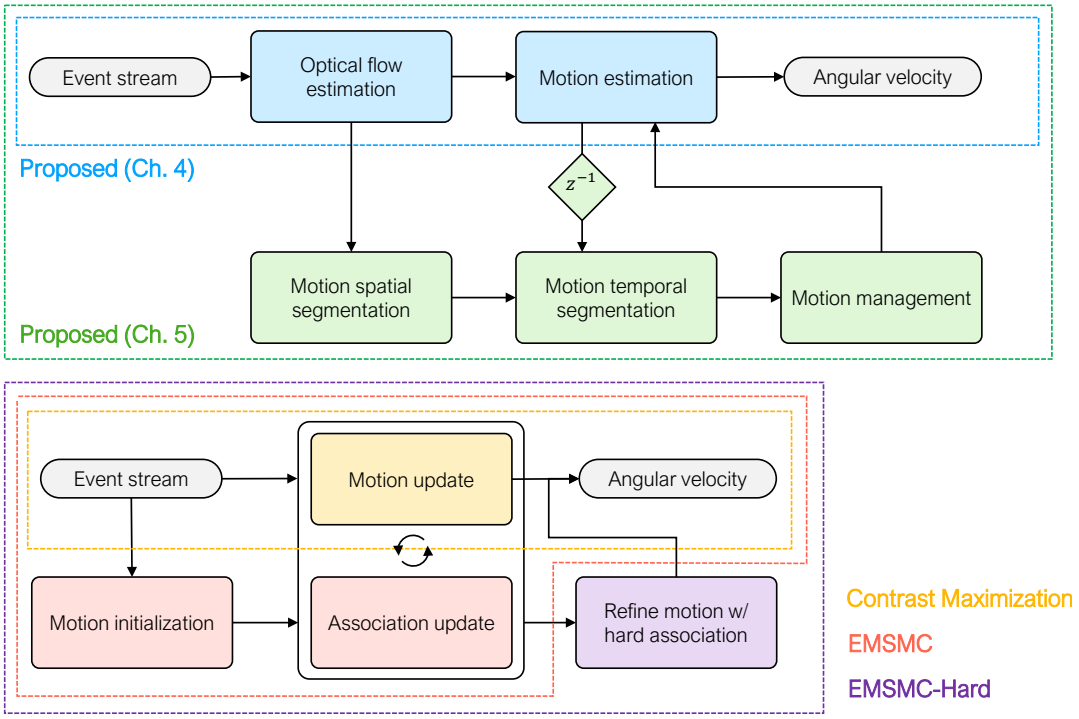


Figure 5.5: Simplified flowchart of the compared algorithms. The modules implemented in each algorithm is depicted inside dashed box.

angular velocity with a bunch of selected optical flows belonging to a static background, whereas the previous algorithm uses all fetched optical flows.

As mentioned before, EMSMC segments the motion of events in (x, y, t) space with motion compensation that is similar to contrast maximization framework. Since the algorithm uses K-means method, it requires the number of clusters in the initialization stage. Also, although a cluster can be eliminated when the cluster does not supported from any event, a cluster can not be added when a new object appears. Moreover, it only segments motions, thus it can not identify which segment belongs to the static background. Therefore, I have modified EMSMC algorithm to be compared properly to the proposed algorithm: (i) I re-initialize K clusters when any cluster is eliminated. (ii) I manually select a specific cluster to be compared to the motion of static background. Related to the second item, I record the motion of all clusters and divide the segments according to the re-initialization branch point. Then, I evaluate each segment compared

to the ground-truth motion and select a segment showing lower RMSE.

Moreover, because EMSMC updates the probability of an event belonging to a cluster and computes rotate vector with the probability as a weight for optimization, the computed motion of clusters might not be refined and precise due to the non-zero weight of event belonging to the moving object. Thus, I have attached an additional motion update module with hard association to make the weight of event have a value of 0 or 1. I denote this modified version of EMSMC as EMSMC-Hard from this place.

In validation, I also evaluate the algorithm previously proposed in Chapter 4 and a representative version of contrast maximization framework to refer to the angular velocity estimation without considering moving objects. The work in Chapter 4 and Chapter 5 have a common angular velocity estimation module. Thus, I can verify the improvement through the added modules in this chapter. Also, I evaluate the contrast maximization that is a core module of EMSMC, and the number of event group is set equal to that of EMSMC, i.e., 10000. Fig. 5.5 shows the simplified flowchart of the compared algorithms in the paper. Contrast maximization (CM) and EMSMC yield angular velocity value from the parameter of motion model directly, while EMSMC-Hard re-estimates motion parameter using hard association.

Fig. 5.6 shows the examples of internal parameter of compared algorithms on tested sequences. The different models and clusters in the proposed algorithm and EMSMC, respectively, are described by different colors to each other. Particularly, in the proposed algorithm, since the motion of green colored models are detected for the first time, static background are denoted as green models. Then, the motion of moving objects are detected and identified as different colors. In the case of EMSMC, a new ID is assigned to a newly generated cluster each time initialization is performed again. This is why the color of the clusters in the second row of Fig. 5.6 is different from that of the other rows. For the first, third, and forth rows where multiple motions exist, EMSMC divides two clusters successfully whereas IWE of CM is corrupted. However, for the second row where moving vehicle dominates most of event stream, EMSMC is not able to identify static background. On the contrary, the proposed algorithm not only distinguishes different motions but also identifies movements in a static background.

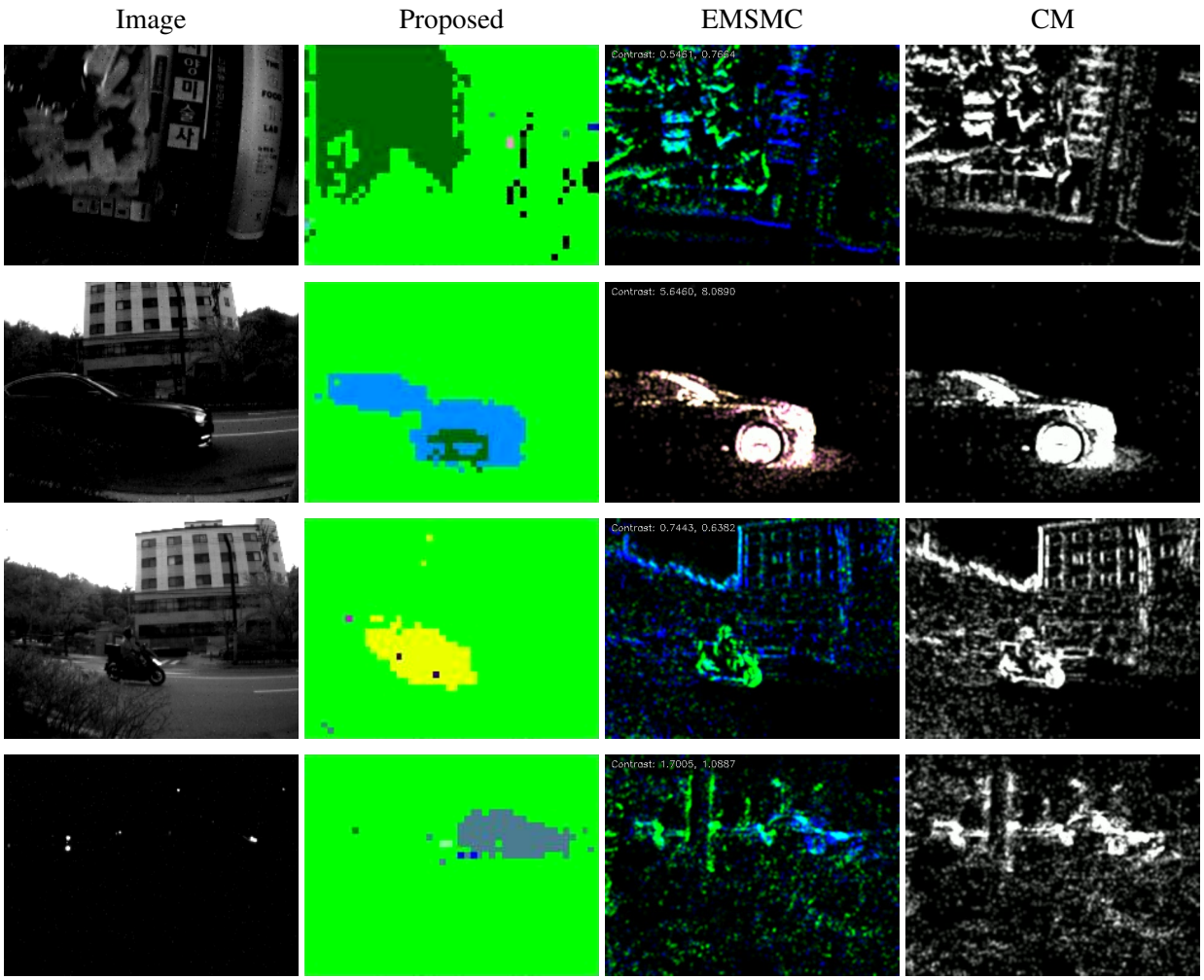
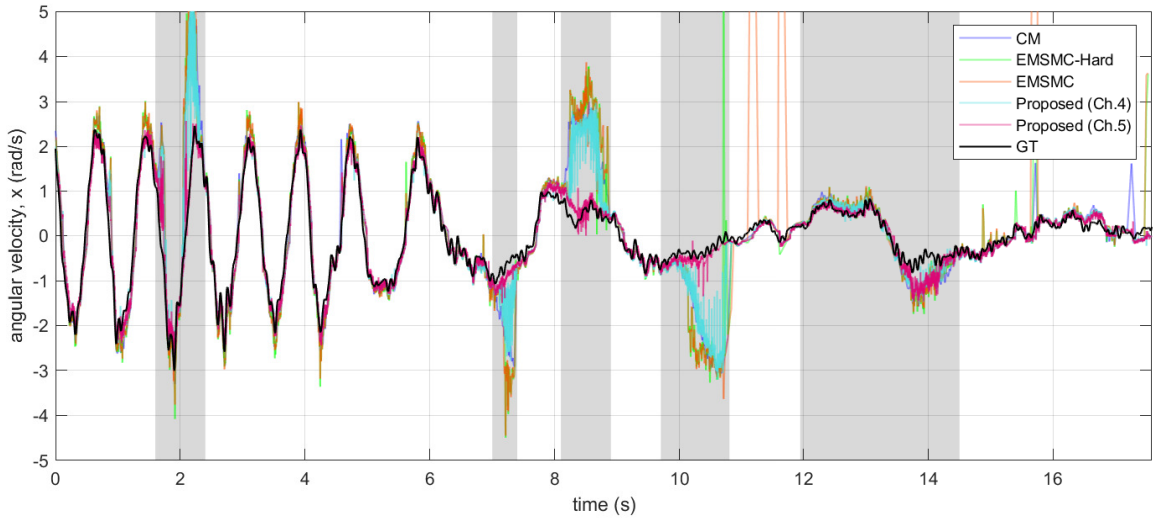


Figure 5.6: Snapshot of the internal parameter of compared algorithms on tested sequences. Each row indicates the indoor, outdoor_day1, outdoor_day2, and outdoor_night sequences from top to bottom. Also, each column denotes gray image, the proposed algorithm, EMSMC, and CM from left to right. Best viewed in color.

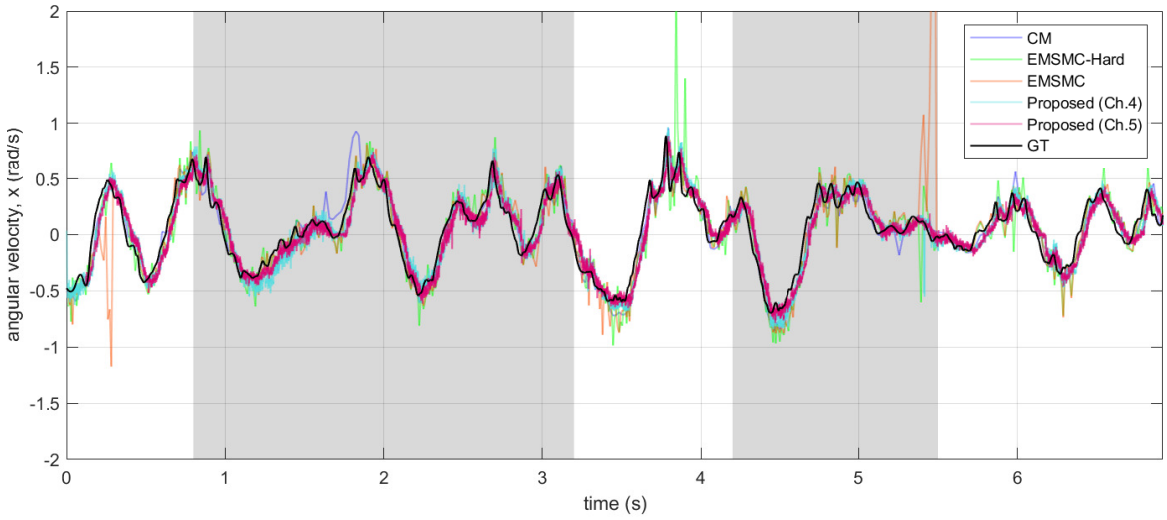
Table 5.2: Evaluations for latency and accuracies of angular velocity in dynamic environments. Best results are in bold.

Sequence			Proposed (Ch.5)	Proposed (Ch.4)	EMSMC	EMSMC-Hard	CM
Avg. latency		indoor	15.52	8.72	5.66	**5.54**	5.88
		outdoor_day1	14.32	12.59	10.22	11.71	**8.91**
		outdoor_day2	14.40	11.26	11.23	**8.02**	9.35
		outdoor_night	28.01	24.22	20.28	19.63	**18.78**
Avg. RMSE		indoor	**0.263**	0.537	0.937	0.808	0.874
		outdoor_day1	**0.132**	0.409	0.779	0.449	0.539
		outdoor_day2	**0.145**	0.153	0.391	0.390	0.542
		outdoor_night	0.158	**0.155**	1.242	0.347	0.294
Accuracy (higher is better)	$\epsilon < 0.5^3$	indoor	**0.392**	0.317	0.188	0.185	0.211
		outdoor_day1	**0.708**	0.588	0.515	0.433	0.667
		outdoor_day2	0.624	**0.691**	0.654	0.532	0.559
		outdoor_night	0.609	**0.622**	0.493	0.463	0.430
	$\epsilon < 0.5^2$	indoor	**0.850**	0.695	0.503	0.478	0.526
		outdoor_day1	**0.983**	0.775	0.761	0.726	0.770
		outdoor_day2	**0.976**	0.963	0.852	0.814	0.661
		outdoor_night	0.965	**0.968**	0.871	0.835	0.856
	$\epsilon < 0.5$	indoor	**0.961**	0.809	0.721	0.722	0.738
		outdoor_day1	**0.999**	0.816	0.841	0.821	0.822
		outdoor_day2	**1.000**	0.996	0.895	0.886	0.754
		outdoor_night	1.000	**1.000***	0.966	0.979	0.938

*This superscript indicates the best result, when compared by expanding the number of decimal places.

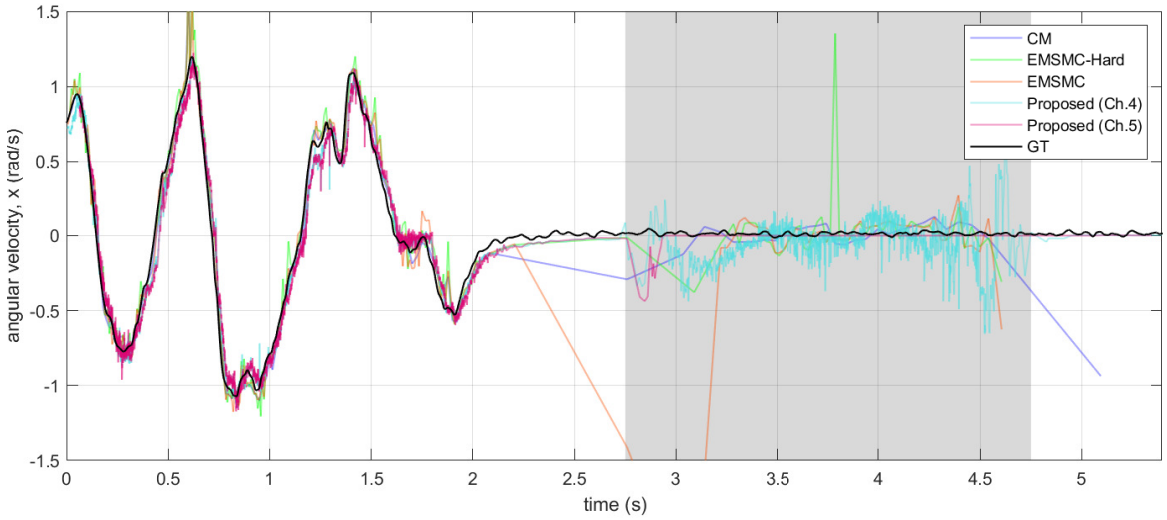


(a) indoor

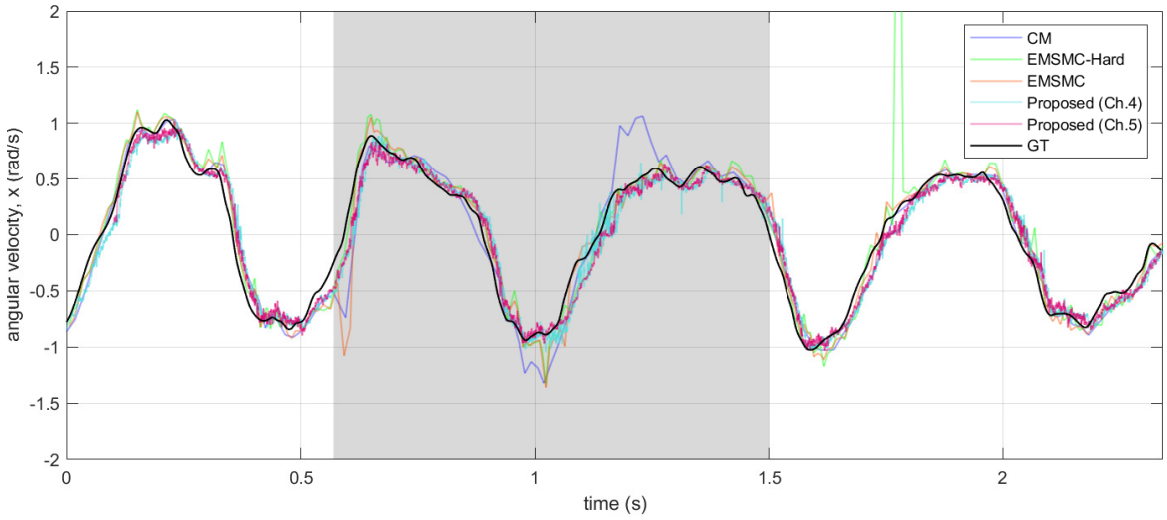


(b) outdoor_night

Figure 5.7: Angular velocity estimates of compared algorithms on (a) indoor and (b) outdoor_night sequences. The dark shaded region represents the time interval in which a moving object appears. For visual simplicity, only ω_x is illustrated. Best viewed in color.



(a) outdoor_day1



(b) outdoor_day2

Figure 5.8: Angular velocity estimates of compared algorithms on (a) outdoor_day1 and (b) outdoor_day2 sequences. The dark shaded region represents the time interval in which a moving object appears. For visual simplicity, only ω_x is illustrated. Best viewed in color.

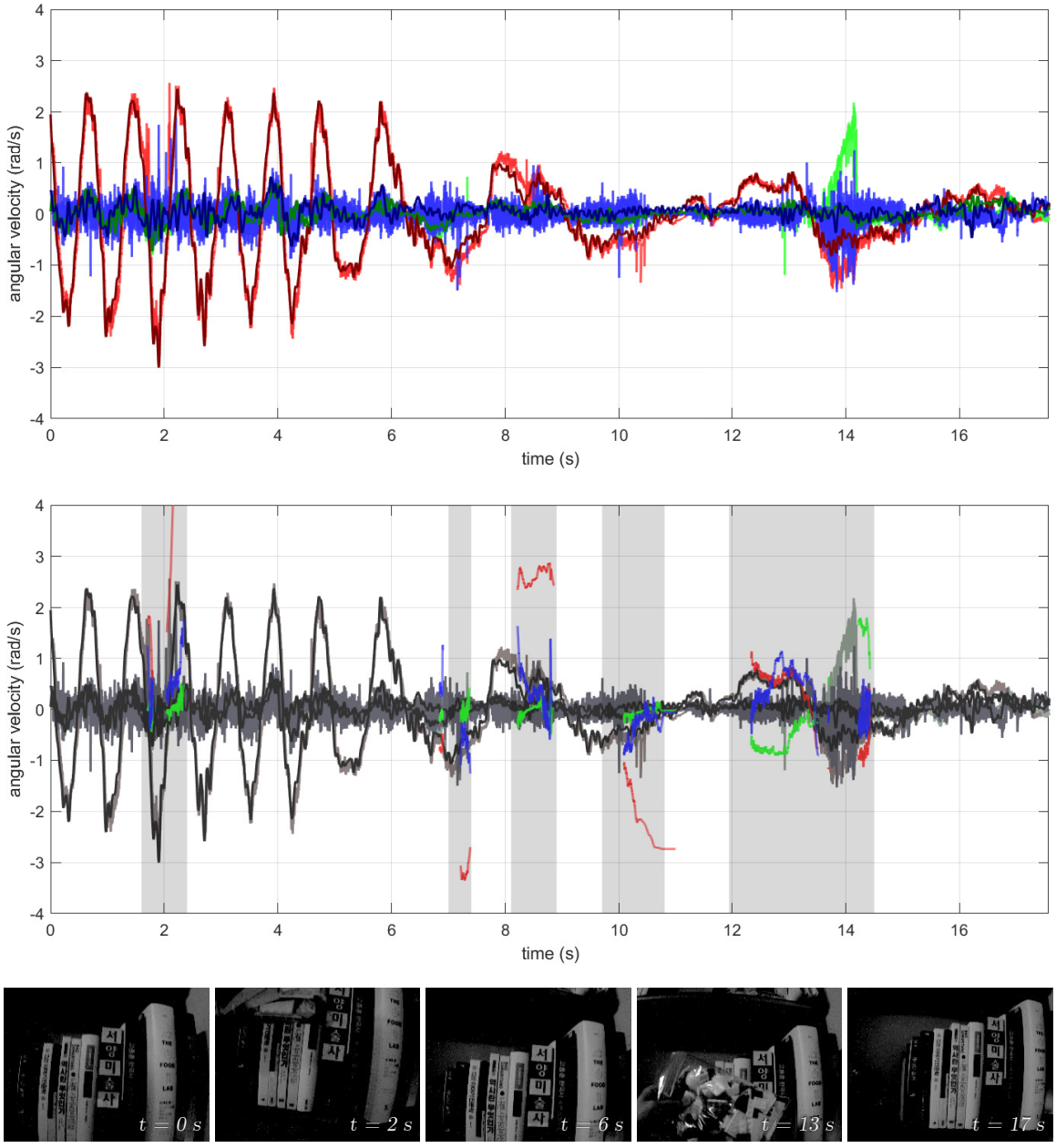


Figure 5.9: Qualitative results of the proposed angular velocity on indoor sequences. The ground-truth (deep red, green, blue), estimated (red, green, blue lines for ω_x, ω_y, ω_z) are compared in the upper plot. The lower plot shows the estimated angular velocity of moving object for each axis. The shaded region implies the existence of a moving object. Best viewed in color.

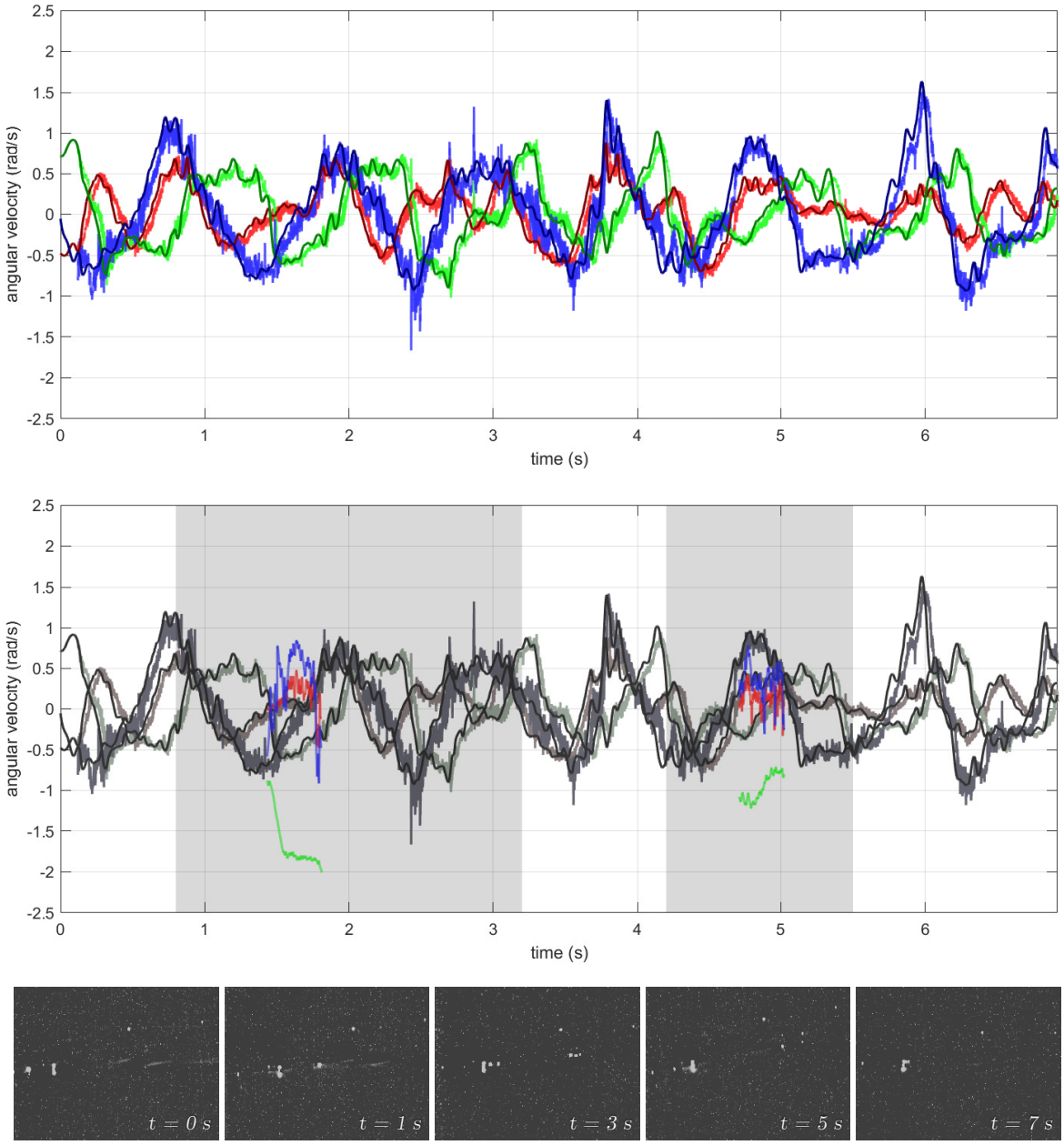


Figure 5.10: Qualitative results of the proposed angular velocity on outdoor_night sequences. The ground-truth (deep red, green, blue), estimated (red, green, blue lines for $\omega_x, \omega_y, \omega_z$) are compared in the upper plot. The lower plot shows the estimated angular velocity of moving object for each axis. The shaded region implies the existence of a moving object. Best viewed in color.

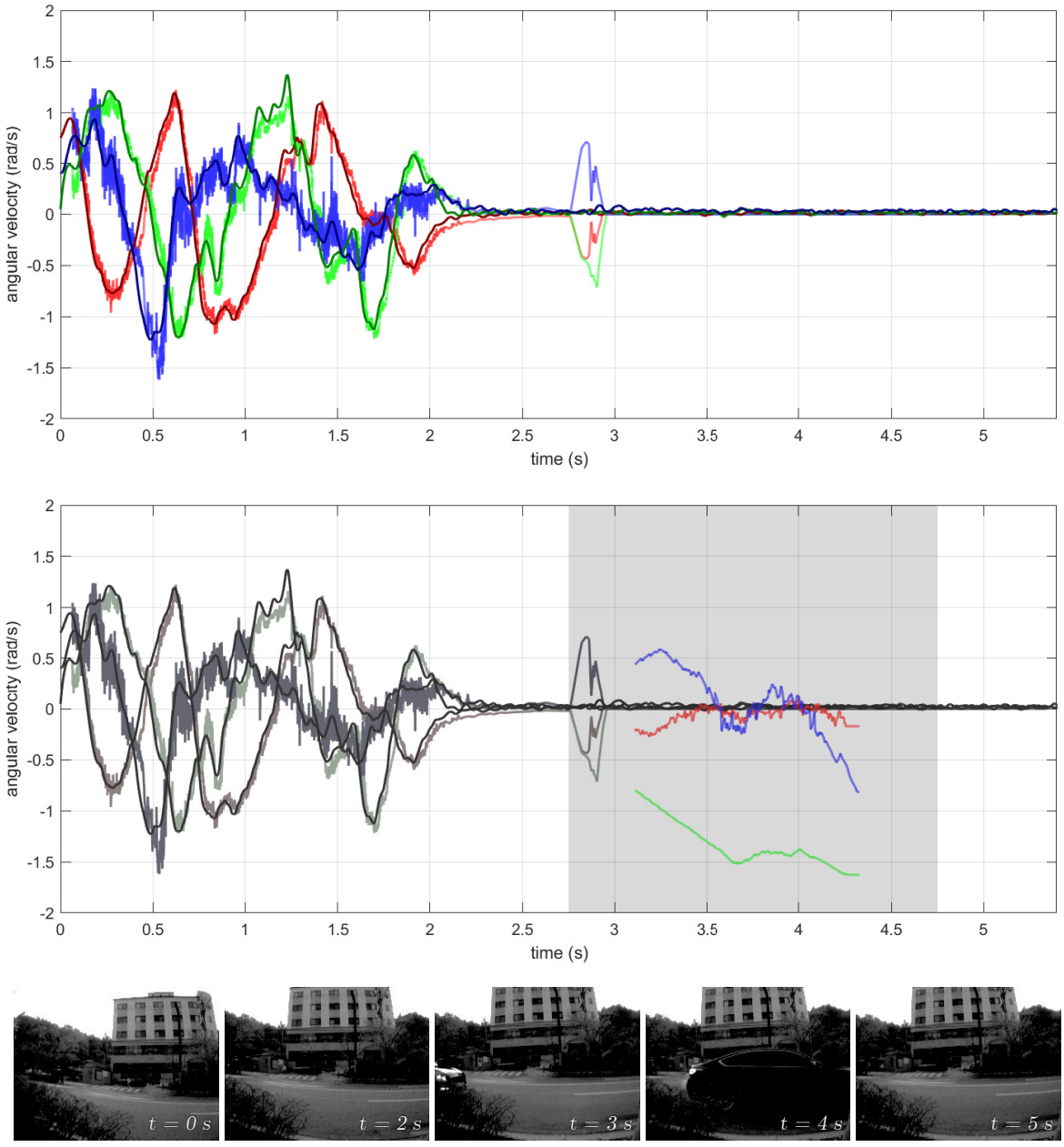


Figure 5.11: Qualitative results of the proposed angular velocity on outdoor_day1 sequences. The ground-truth (deep red, green, blue), estimated (red, green, blue lines for $\omega_x, \omega_y, \omega_z$) are compared in the upper plot. The lower plot shows the estimated angular velocity of moving object for each axis. The shaded region implies the existence of a moving object. Best viewed in color.

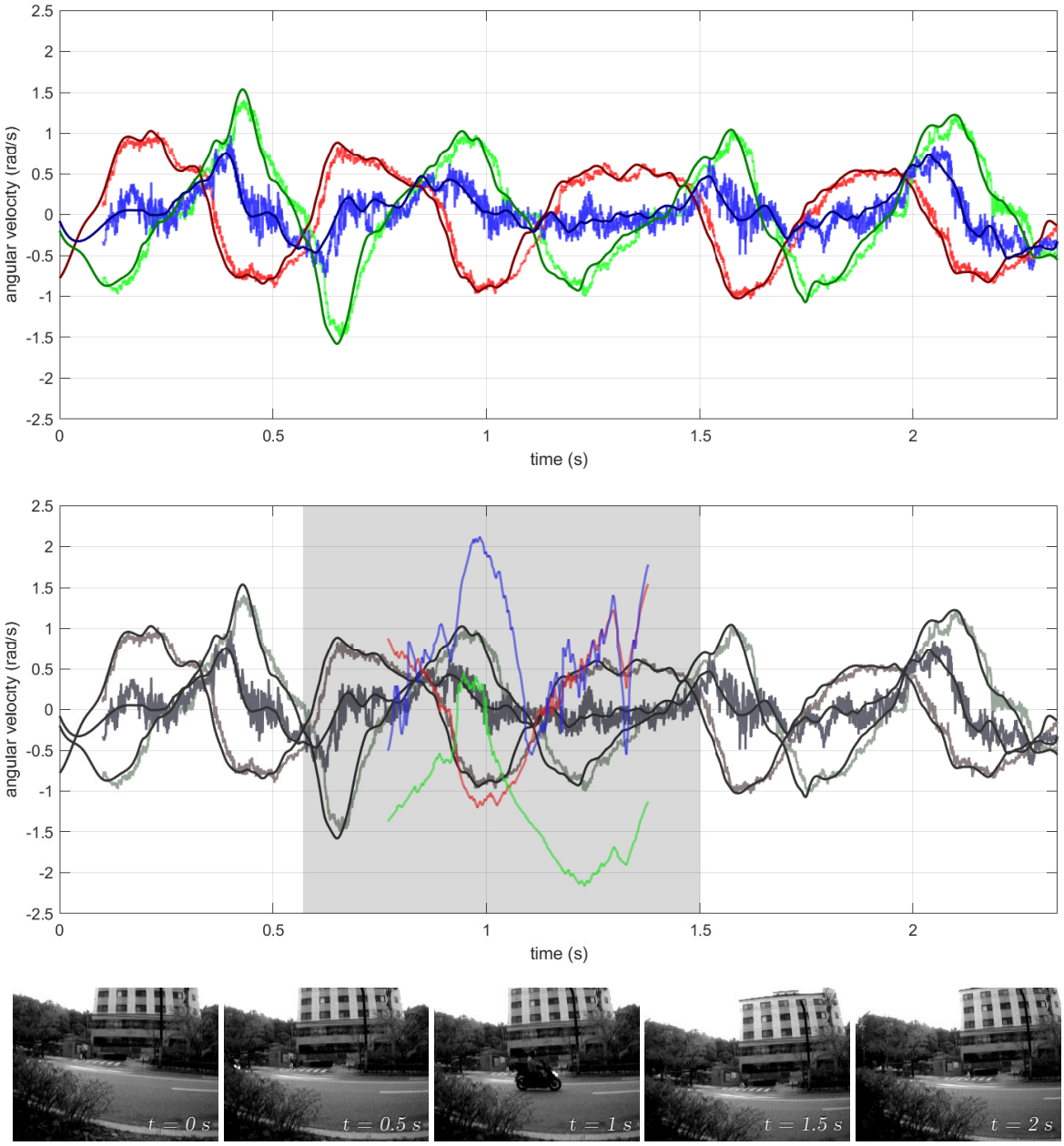


Figure 5.12: Qualitative results of the proposed angular velocity on outdoor_day2 sequences. The ground-truth (deep red, green, blue), estimated (red, green, blue lines for $\omega_x, \omega_y, \omega_z$) are compared in the upper plot. The lower plot shows the estimated angular velocity of moving object for each axis. The shaded region implies the existence of a moving object. Best viewed in color.

In Table 5.2, I compute the latency, error, and accuracy of the proposed and the existing algorithms by comparing with IMU. The latency is computed by (4.9) in an optimization-based method. The average RMSE is calculated by averaging each RMSE for three axes. Furthermore, I evaluate each algorithm using threshold metric below:

$$\text{Threshold: \% of } \boldsymbol{\omega}_i \text{ s.t. mean}(|\boldsymbol{\omega}_i - \boldsymbol{\omega}_{i,gt}|) = \epsilon < thr, \tag{5.28}$$

where $\boldsymbol{\omega_i}$ the i-th estimate of angular velocity and $\boldsymbol{\omega}_{i,gt}$ denotes the corresponding ground-truth value. This metric is inspired from [89], that introduces the threshold metric for evaluating the accuracy of predicted depthmap. Since angular velocity can have three-dimensional negative values unlike depth values having one-dimensional positive values, max and fraction in their paper are substitute with mean and absolute error, respectively. Note that the performance of EMSMC and EMSMC-Hard is evaluated by comparing estimates close to the ground-truth value among motion parameters of its multiple clusters with the ground-truth, because they can not identify background or objects, but only segment them spatially. I divide their motion parameters of multiple clusters by 10 timestamp intervals, then find the cluster that best matches the ground-truth value. Subsequently, the estimate of EMSMC and EMSMC-Hard to be compared to the ground-truth value is formed by putting estimates close to the true value in each interval together.

The RMSE of the proposed algorithm is superior to the other algorithms, while the latency is increased by few milliseconds compared with the previous work. The dynamic characteristic of dual-mode motion models may affect an increase in latency. Note that, for outdoor_day2 and outdoor_night sequences, accuracies are relatively good in each algorithm because moving objects appear in a small part of the image. For the same reason, the algorithm proposed in Chapter 4 shows the best result on outdoor_night sequence. From the threshold analysis, as the threshold value increases from 0.5^3 to 0.5^1, accuracy becomes high. Threshold accuracy is able to verify the performance of algorithm while being less contaminated by abnormally high errors. For all sequences, EMSMC shows higher RMSE than EMSMC-Hard. However, EMSMC shows better performance than EMSMC-Hard in terms of threshold accuracy. This implies that the performance of EMSMC is degraded by several large errors as shown in Fig. 5.7 and Fig. 5.8. Never-

theless, threshold accuracy suggests that EMSMC is better than EMSMC-Hard, which is related to the reason why they [36] have decided to use soft association. Overall, the algorithm proposed in Chapter 5 shows lower RMSE and higher accuracy than the other algorithms including the algorithm in Chapter 4.

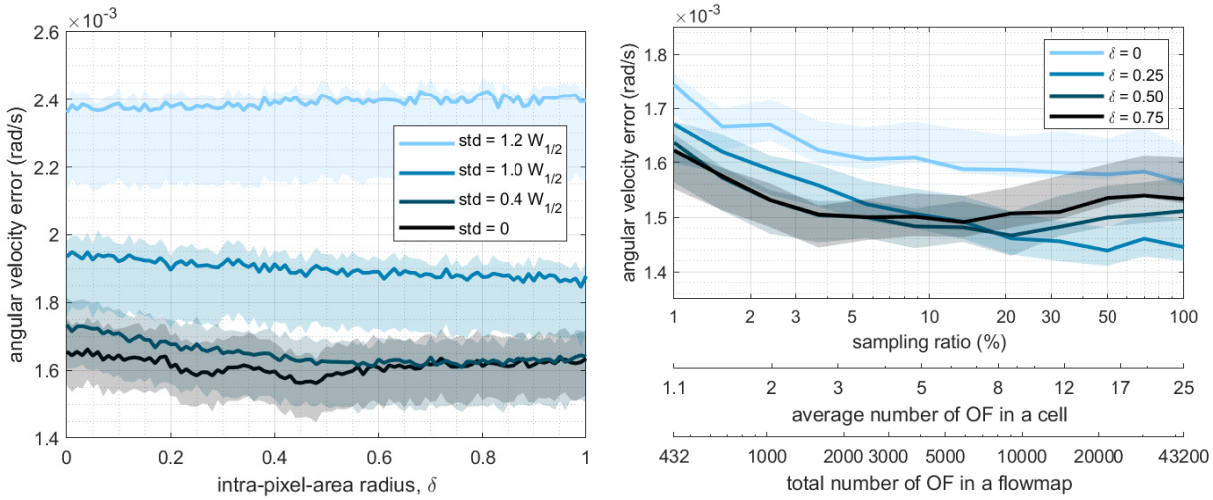
Furthermore, Fig. 5.7 and Fig. 5.8 show the angular velocity estimates of the comparison algorithms on the x-axis only for visual simplicity. The dark shaded region represents the time interval in which a moving object appears. When a moving object appears, the performance of other algorithms is degraded, while the proposed algorithm robustly and accurately estimates angular velocity. In particular, the proposed algorithm can identify the stationary motion of static background as shown in Fig. 5.8(a). Also, Figs. 5.9 to 5.12 show the angular velocity estimation result of the proposed algorithm and ground-truth values for the three axes. The upper plot shows the estimated angular velocity of the camera, and the lower plot shows the estimated angular velocity of moving object for each axis. The angular velocity of moving object denotes the relative motion to the camera, which will be discussed in detail in Section 5.5.2. The images below describe the scene of the main moments.

5.5 Discussion

5.5.1 Intra-pixel-area Event

Previously, the IPA approach proposed in Chapter 3 helps plane fitting method to compute more precise visual flow and detect edgemap. Similarly, the IPA approach makes a grid-based flowmap more accurate. The benefit of IPA approach is verified by simulated analysis in Fig. 5.13 and evaluation of real environments in Table 5.3. In simulated verification, each data point is generated by 10000 simulation tests with varying rotation motion, position of optical flow, and Gaussian noise. In both figures, angular velocity error is measured by L2 norm. Fig. 5.13(a) shows the accuracy of angular velocity versus IPA radius, δ, which is related to W_δ discussed in (5.1):

$$W_\delta = \delta \times \frac{s_{grid}}{f}, \tag{5.29}$$



(a) versus IPA radius

(b) versus optical flow sampling ratio

Figure 5.13: Accuracy of angular velocity versus (a) IPA radius, δ, and (b) optical flow sampling ratio. The shaded area represents the range between (a) the 45^{th} and 55^{th} percentiles or (b) the 48^{th} and 52^{nd} percentiles.

where s_{grid} is the spacing size of grid-based flowmap and f is the focal length in pixels. As a result, "$\delta = 0.5$" means that IPA includes the entire pixel area, "$\delta = 0$" indicates that the IPA approach is not used, and $W_{1/2}$ is the value of W_δ when $\delta = 0.5$. For an extremely severe noise level, $std = 1.2W_{1/2}$, the RMSE error increases as the δ grows. This is because IPA approach minimizes the error in (5.1) based on the rotation vector which is misestimated due to severe noise, thus resulting erroneous $\mathbf{u}^*$. In case of moderate noise levels, the accuracy is improved for higher value of δ. Meanwhile, the accuracy is minimized at about $\delta = 0.5$ for zero noise levels.

In Fig. 5.13(b), the control parameter is sampling ratio that means what percentage of the total number of optical flow vectors is an input. For example, a grid-based flowmap is constructed by 432 optical flow vectors in average, when sampling ratio equals to 0.01. Also, in this case, each cell among non-empty cells fetches 1.1 optical flow approximately. The plot can be interpreted by dividing it into left and right parts based on a total of about 5000 optical flow vectors. For the small number of optical flow in a flowmap less than about 5000, the larger IPA radius is, the

Table 5.3: Evaluations for latency and accuracies of angular velocity with or without the IPA approach and mean method. Best results are in bold.

Sequence			w/ IPA	w/o IPA	mean
Avg. latency		indoor	**15.52**	16.42	16.14
		outdoor_day1	14.32	14.36	**14.00**
		outdoor_day2	**14.40**	15.16	14.49
		outdoor_night	28.01	29.27	**27.77**
Avg. RMSE		indoor	**0.263**	0.264	0.292
		outdoor_day1	**0.132**	0.141	0.139
		outdoor_day2	0.145	0.145	**0.144**
		outdoor_night	**0.158**	0.160	0.161
Accuracy (higher is better)	$\epsilon < 0.5^3$	indoor	0.392	**0.402**	0.364
		outdoor_day1	**0.708**	0.690	0.682
		outdoor_day2	**0.624**	0.620	0.621
		outdoor_night	**0.609**	0.586	0.592
	$\epsilon < 0.5^2$	indoor	0.850	**0.854**	0.829
		outdoor_day1	**0.983**	0.980	0.982
		outdoor_day2	0.976	0.978	**0.980**
		outdoor_night	**0.965***	0.965	0.965
	$\epsilon < 0.5$	indoor	**0.961***	0.961	0.948
		outdoor_day1	**0.999**	0.997	0.998
		outdoor_day2	**1.000**	**1.000**	**1.000**
		outdoor_night	1.000	1.000	**1.000***

*This superscript indicates the best result, when compared by expanding the number of decimal places.

higher accuracy is. On the other side, there is some minimum point at about $\delta = 0.25$. Since the proposed algorithm fetches around 1000 optical flow vectors to construct a grid-based flowmap, I have decided to set δ to 0.5 and have evaluated the algorithm.

Evaluation results for latency, error, and accuracy of angular velocity depending on the method of constructing grid-based flowmap is shown in Table 5.3. I have also tested another "mean" method that takes the position of optical flow of a cell as the average of the position of optical flow observed. This simple method enhances the performance in terms of latency and error when compared with "w/o IPA" method. However, it is generally less accurate than "w/ IPA" because it does not consider rotational motion, whereas IPA updates starting point based on the pre-estimated motion. The threshold accuracy also suggests that the IPA approach is more effective than the mean method. The different performance of these methods may arise from a cell that exists at the boundaries of static and dynamic object. Exceptionally, for outdoor_day2 sequence that contains a small-sized moving object, mean method shows the best result in terms of RMSE. Overall, IPA approach enhances the latency and accuracy of angular velocity in situations where a dominant object exists.

5.5.2 Motion Segmentation and Estimation

The proposed algorithm not only estimates the angular velocity of ego-motion of the camera but also computes the rotational motion of moving objects. Since the dual-mode motion model detects and tracks the position of objects depending on their area and motion, the moving objects are tracked and identified as distinct IDs. The lower plots in Figs. 5.9 to 5.12 show the rotational motion estimation results of a moving object. The relative motion of moving object to the camera are detected and estimated in the shaded regions where the moving object appears. Especially in Fig. 5.11, because the vehicle moves from the left to the right in front of the camera, the estimated motion of the vehicle is dominated by y-axis angular velocity. Currently, the proposed motion segmentation and ego-motion estimation procedures are operated under the rotational motion assumption, thus the motion of moving objects is also restricted to rotational motion. Therefore, future work should include the full-DoF motion estimation so that the motion model

of the moving object can be exactly described.

Moreover, since the motion segmentation and the motion estimation modules of the proposed algorithm are not strongly coupled with each other, but they operate in parallel, thus, the proposed motion segmentation is able to be integrated with the existing algorithms in order to improve their robustness in dynamic environments. Meanwhile, to highly boost the practical performance, it is encouraged to be reduced computational load and be developed a pixel-wise segmentation with high resolution, although the motion estimation module of the proposed algorithm can exclude small moving objects by virtue of the RANSAC scheme.

5.6 Summary

In this paper, I developed a robust algorithm that identifies different motions and estimates angular velocity belonging to the static background. The motion segmentation module fetches a relatively large number of asynchronous optical flow vectors to identify the region of static background in the image frame, while the motion estimation module computes angular velocity with a fewer number of optical flow vectors thus satisfying high temporal resolution. The proposed dual-mode motion model for asynchronous and sparse optical flow stream data detects abnormally moving object and identifies the apparent motion of static background. To validate the performance of the proposed algorithm, I have collected datasets including moving objects and stationary camera, and have published online. In evaluation, the proposed algorithm is superior to the other algorithm in terms of accuracy. Particularly, the proposed algorithm can robustly estimate the stationary motion of static background when the camera stops and moving object appears, whereas other algorithms fail to estimate the angular velocity of ego-motion. In addition, since the proposed algorithm fetches asynchronous optical flow stream and explicitly calculates angular velocity from a bunch of optical flow vectors, motion estimation module (algorithm in Ch.4) without motion segmentation alone can estimate angular velocity more accurately than the existing algorithms. Further, the proposed motion segmentation which is loosely coupled with the estimated angular velocity can collaborate with other motion estimation algorithms and be

exploited in complex environments such as urban areas.

6

Conclusion

In the dissertation, I tackle the problem of dealing with dynamic environments that include extreme illumination conditions and moving objects such as human interaction, vehicles, etc. To overcome a harsh light environment, I take advantage of event cameras that have a wider dynamic range (120 dB) than conventional CMOS cameras (60 dB). Thus, I have focused on the enhancement of the event-based algorithm in terms of the robustness to various types of dynamic environments: (i) I have proposed an intra-pixel-area event and developed the algorithm that estimates visual flow and detects sharp edge map *robustly under the pixel noise* of the event. (ii) I have proposed an asynchronous optical flow stream with low-latency and *scene-robust* and have validated at various texture levels of a scene and the speed of the camera. (iii) By exploiting (i-ii), I have proposed the algorithm that estimates angular velocity *robustly in dynamic environments* including moving objects. In the following, I present a concise summary of each chapter addressed in this dissertation.

Chapter 3: Intra-pixel-area Events and Applications. I proposed an intra-pixel-area event that is a virtual floating event on the image plane in order to enhance the performance of RANSAC of a fitting plane algorithm. Intra-pixel-area events approach makes the plane fitting algorithm

robust to the pixel noise of event. Then, the lifetime of event is calculated from the inverse of the norm of the normal vector of the plane, and an edge map is constructed by alive events. As the plane fitting on an SAE is found robustly and accurately, the lifetime and by-product edge map are better estimated than the existing algorithm in terms of accuracy and CDM similarity metrics.

Chapter 4: Low-Latency and Scene-Robust Asynchronous Optical Flow Stream Estimation. I improved the performance of the existing block-matching algorithm for optical flow estimation in terms of latency and robustness to the non-uniform texture of a scene. Moreover, I developed an algorithm that estimates angular velocity with low latency by virtue of the proposed asynchronous optical flow stream, and I evaluate the latency of algorithms using the result of angular velocity estimates. In evaluation, the estimated optical flow shows much lower latency than other frame-based optical flow algorithms and the estimated angular velocity shows higher accuracy and robustness than the existing algorithms.

Chapter 5: Robust Angular Velocity Estimation in Dynamic Environments. I developed an algorithm that segments dynamic foreground from static background using dual-mode motion model management. The dual-mode motion model composed of apparent and candidate models can detect the motion of objects that is different from the apparent motion of the camera, and identify foreground and background. In evaluation, the proposed motion segmentation successfully identifies the motion of object even though the camera is stationary. Therefore, the angular velocity of the camera is more robustly and accurately estimated from the region of the static background than other algorithms.

The possible future work of this dissertation will be expanding the dimension of the motion of event cameras. Since I developed an algorithm that estimates only three of the 6-DoF motions, i.e., rotation, I restricted the situation of the collected dataset to rotational motion. If the dual-mode motion model estimates full-DoF motions, the motion of moving object can be described accurately. Besides, the 6-DoF pose estimation using only motion integration causes a drift error, thus, in addition to optical flow, other clues (e.g., feature point in conventional computer vision) that indicate the specified point in the world coordinate are also required to estimate the pose of event cameras while reducing the drift problem.

References

[1] "Retina – anatomy and physiology," https://www.getbodysmart.com/nervous-system/retina/, accessed: 2021-11-10.

[2] "Sony develops back-illuminated cmos image sensor, realizing high picture quality, nearly twofold sensitivity and low noise," https://www.sony.com/en/SonyInfo/News/Press/200806/08-069E/, accessed: 2021-10-27.

[3] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-dof pose tracking for high-speed maneuvers," *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pp. 2761–2768, 2014.

[4] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," *European Conference on Computer Vision*, pp. 349–364, 2016.

[5] Y. Zhou, G. Gallego, X. Lu, S. Liu, and S. Shen, "Event-based motion segmentation with spatio-temporal graph cuts," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[6] B. Rueckauer and T. Delbruck, "Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor," *Frontiers in neuroscience*, vol. 10, p. 176, 2016.

[7] L. Luo, *Principles of neurobiology*. Garland Science, 2020.

[8] D. H. Hubel, *Eye, brain, and vision.* Scientific American Library/Scientific American Books, 1995.

[9] P. Lichtsteiner, "An aer temporal contrast vision sensor," Ph.D. dissertation, ETH Zurich, 2006.

[10] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120db 30mw asynchronous vision sensor that responds to relative intensity change," *IEEE International Solid State Circuits Conference-Digest of Technical Papers*, pp. 2060–2069, 2006.

[11] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 db 15μ s latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[12] C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," *IEEE International Symposium on Circuits and Systems*, pp. 2130–2133, 2008.

[13] C. Posch, D. Matolin, and R. Wohlgenannt, "A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2010.

[14] D. Matolin, G. Orchard, J.-B. Floderer, and C. Posch, "Live demonstration: Event-driven high-speed imaging," *IEEE Biomedical Circuits and Systems Conference Proceedings*, pp. 174–174, 2014.

[15] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[16] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu, "A 640×480 dynamic vision sensor with a 9µm pixel and 300meps address-event representation," *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 66–67, 2017.

[17] T. Delbruck, "Frame-free dynamic digital vision," *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pp. 21–26, 2008.

[18] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural Networks*, vol. 27, pp. 32–37, 2012.

[19] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow." *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 2, pp. 407–417, 2014.

[20] C. A. Mead and M. A. Mahowald, "A silicon model of early visual processing," *Neural Networks*, vol. 1, no. 1, pp. 91–97, 1988.

[21] M. Mahowald, "Vlsi analogs of neuronal visual processing: a synthesis of form and function," Ph.D. dissertation, California Institute of Technology Pasadena, 1992.

[22] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[23] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison, "Simultaneous mosaicing and tracking with an event camera," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 566–576, 2008.

[24] D. Weikersdorfer and J. Conradt, "Event-based particle filtering for robot self-localization," *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 866–870, 2012.

[25] C. Lee, A. K. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks," *European Conference on Computer Vision*, pp. 366–382, 2020.

[26] M. Gehrig, S. B. Shrestha, D. Mouritzen, and D. Scaramuzza, "Event-based angular velocity regression with spiking networks," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[27] C. Reinbacher, G. Munda, and T. Pock, "Real-time panoramic tracking for event cameras," *IEEE International Conference on Computational Photography (ICCP)*, pp. 1–9, 2017.

[28] G. Gallego and D. Scaramuzza, "Accurate angular velocity estimation with an event camera," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 632–639, 2017.

[29] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, "Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time," *International Journal of Computer Vision*, vol. 126, no. 12, pp. 1394–1414, 2018.

[30] M. Liu and T. Delbruck, "Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors," *British Machine Vision Conference (BMVC)*, 2018.

[31] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 989–997, 2019.

[32] R. Li, D. Shi, Y. Zhang, K. Li, and R. Li, "Fa-harris: A fast and asynchronous corner detector for event cameras," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6223–6229, 2019.

[33] I. Alzugaray and M. Chli, "Ace: An efficient asynchronous corner tracker for event cameras," *International Conference on 3D Vision (3DV)*, pp. 653–661, 2018.

[34] E. Mueggler, C. Forster, N. Baumli, G. Gallego, and D. Scaramuzza, "Lifetime estimation of events from dynamic vision sensors," *IEEE international conference on Robotics and Automation (ICRA)*, pp. 4874–4881, 2015.

[35] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2018.

[36] T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, and D. Scaramuzza, "Event-based motion segmentation by motion compensation," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7244–7253, 2019.

[37] X. Peng, Y. Wang, L. Gao, and L. Kneip, "Globally-optimal event camera motion estimation," *European Conference on Computer Vision*, pp. 51–67, 2020.

[38] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, 2018.

[39] G. Gallego, M. Gehrig, and D. Scaramuzza, "Focus is all you need: Loss functions for event-based vision," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 280–12 289, 2019.

[40] A. Mitrokhin, Z. Hua, C. Fermuller, and Y. Aloimonos, "Learning visual motion segmentation using event surfaces," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14 414–14 423, 2020.

[41] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu, "Space-time event clouds for gesture recognition: From rgb cameras to event cameras," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1826–1835, 2019.

[42] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++ deep hierarchical feature learning on point sets in a metric space," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5105–5114, 2017.

[43] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 884–892, 2016.

[44] H. Rebecq, G. Gallego, and D. Scaramuzza, "Emvs: Event-based multi-view stereo," *British Machine Vision Conference (BMVC)*, no. CONF, 2016.

[45] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, "Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2016.

[46] F. Paredes-Vallés, K. Y. Scheper, and G. C. de Croon, "Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion

perception," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 8, pp. 2051–2064, 2019.

[47] G. Gallego, C. Forster, E. Mueggler, and D. Scaramuzza, "Event-based camera pose tracking using a generative event model," *arXiv preprint arXiv:1510.01972*, 2015.

[48] D. Liu, A. Parra, and T.-J. Chin, "Globally optimal contrast maximisation for event-based motion estimation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6349–6358, 2020.

[49] A. Mitrokhin, C. Ye, C. Fermüller, Y. Aloimonos, and T. Delbruck, "Ev-imo: Motion segmentation dataset and learning pipeline for event cameras," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6105–6112, 2019.

[50] J. Li, F. Shi, W.-H. Liu, D. Zou, Q. Wang, P. K. Park, and H. Ryu, "Adaptive temporal pooling for object detection using dynamic vision sensor." *British Machine Vision Conference (BMVC)*, 2017.

[51] J. Xu, M. Jiang, L. Yu, W. Yang, and W. Wang, "Robust motion compensation for event cameras with smooth constraint," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 604–614, 2020.

[52] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5816–5824, 2017.

[53] S. Hecht, S. Shlaer, and M. H. Pirenne, "Energy, quanta, and vision," *Journal of General Physiology*, vol. 25, no. 6, pp. 819–840, 1942.

[54] T. Euler, S. Haverkamp, T. Schubert, and T. Baden, "Retinal bipolar cells: elementary building blocks of vision," *Nature Reviews Neuroscience*, vol. 15, no. 8, pp. 507–519, 2014.

[55] N. W. Oesch and J. S. Diamond, "Ribbon synapses compute temporal contrast and encode luminance in retinal rod bipolar cells," *Nature neuroscience*, vol. 14, no. 12, pp. 1555–1561, 2011.

[56] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[57] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: bioinspired cameras with spiking output," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014.

[58] B. Razavi, *Design of analog CMOS integrated circuits*. Tata McGraw-Hill Education, 2002.

[59] Y. Suh, S. Choi, M. Ito, J. Kim, Y. Lee, J. Seo, H. Jung, D.-H. Yeo, S. Namgung, J. Bong *et al.*, "A 1280$\times$ 960 dynamic vision sensor with a 4.95-μm pixel pitch and motion artifact minimization," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2020.

[60] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[61] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3d perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.

[62] C. Scheerlinck, H. Rebecq, T. Stoffregen, N. Barnes, R. Mahony, and D. Scaramuzza, "Ced: color event camera dataset," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[63] Y. Hu, J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "Ddd20 end-to-end event camera driving dataset: Fusing frames and events with deep learning for improved steering prediction," *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, 2020.

[64] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset," *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[65] A. J. Lee, Y. Cho, S. Yoon, Y. Shin, and A. Kim, "Vivid: Vision for visibility dataset," *ICRA Workshop on Dataset Generation and Benchmarking of SLAM Algorithms for Robotics and VR/AR*, 2019.

[66] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 703–710, 2014.

[67] J. Jose Tarrio and S. Pedre, "Realtime edge-based visual odometry for a monocular camera," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 702–710, 2015.

[68] C. Kim, P. Kim, S. Lee, and H. J. Kim, "Edge-based robust rgb-d visual odometry using 2-d edge divergence minimization," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, 2018.

[69] Y. Zhou, H. Li, and L. Kneip, "Canny-vo: Visual odometry with rgb-d cameras based on geometric 3d-2d edge alignment," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 184–199, 2019.

[70] S. Maity, A. Saha, and B. Bhowmick, "Edge slam: Edge points based monocular visual slam," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2408–2417, 2017.

[71] J. Conradt, "On-board real-time optic-flow for miniature event-based vision sensors," *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1858–1863, 2015.

[72] A. J. Lee and A. Kim, "Event-based real-time optical flow estimation," *International Conference on Control, Automation and Systems (ICCAS)*, pp. 787–791, 2017.

[73] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck, "Elised—an event-based line segment detector," *Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pp. 1–7, 2016.

[74] S. Seifozzakerini, W.-Y. Yau, B. Zhao, and K. Mao, "Event-based hough transform in a spiking neural network for multiple line detection and tracking using a dynamic vision sensor." *British Machine Vision Conference (BMVC)*, 2016.

[75] F. Barranco, C. L. Teo, C. Fermuller, and Y. Aloimonos, "Contour detection and characterization for asynchronous event sensors," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 486–494, 2015.

[76] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," *British Machine Vision Conference (BMVC)*, 2017.

[77] K. Bowyer, C. Kranenburg, and S. Dougherty, "Edge detector evaluation using empirical roc curves," *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1, pp. 354–359, 1999.

[78] M. S. Prieto and A. R. Allen, "A similarity metric for edge images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1265–1273, 2003.

[79] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[80] A. Z. Zhu and L. Yuan, "Ev-flownet: Self-supervised optical flow estimation for event-based cameras," *Robotics: Science and Systems*, 2018.

[81] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *Journal of the Optical Society of America A*, vol. 2, no. 2, pp. 284–299, 1985.

[82] H. Kim and H. J. Kim, "Real-time rotational motion estimation with contrast maximization over globally aligned events," *IEEE Robotics and Automation Letters*, 2021.

[83] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

[84] G. Agamennoni, J. I. Nieto, and E. M. Nebot, "An outlier-robust kalman filter," *IEEE International Conference on Robotics and Automation*, pp. 1551–1558, 2011.

[85] T. Stoffregen and L. Kleeman, "Simultaneous optical flow and segmentation (sofas) using dynamic vision sensor," *Australasian Conference on Robotics and Automation*, pp. 52–61, 2017.

[86] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, 2020.

[87] S. Lee and H. J. Kim, "Real-time rigid motion segmentation using grid-based optical flow," *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1552–1557, 2017.

[88] K. Moo Yi, K. Yun, S. Wan Kim, H. Jin Chang, and J. Young Choi, "Detection of moving objects with non-stationary cameras in 5.8 ms: Bringing motion detection to your mobile device," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 27–34, 2013.

[89] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Neural Information Processing Systems*, pp. 2366–2374, 2014.

국문초록

본 논문은 움직이는 물체가 존재하는 동적 환경에서 카메라 자신의 각속도를 강인하게 추정하는 문제를 다룬다. 이를 위해 기존 카메라의 한계점을 보완하는 새로운 패러다임을 제시한 이벤트 카메라를 활용한다. 기존 컴퓨터 영상 처리에서 자주 활용되는 RGB-D 및 단안 카메라는 이미지 및 깊이맵을 프레임의 형태로 출력한다. 이러한 종류의 센서들은 적절한 광량 조건에서 빛을 모으기 위해 일정량의 노출 시간이 필요한데, 이 시간 동안에는 데이터가 출력되지 않아 지연 시간이 발생하며, 출력된 프레임에서는 모션 블러가 발생할 수 있다. 또한 정지된 화면의 밝기값을 계속 출력하는 등 불필요한 데이터가 많아서, 비트 깊이를 늘리거나 지연 시간을 줄이기 위해 초당 프레임수를 늘릴 경우, 이에 비례하여 처리해야할 데이터의 양이 증가한다. 또한, 고조도 및 저조도에서의 노출 과다, 노출 부족 현상도 기존 카메라의 활용 영역을 제한할 수 있다. 이와 달리, 이벤트 카메라는 각 픽셀이 독립적으로 빛의 로그 밝기값의 변화를 감지하며, 일정 비율만큼 밝아지거나 어두워지는 경우 해당 픽셀에서 이벤트를 발생시킨다. 이러한 특징 덕분에, 이벤트 카메라는 낮은 지연속도와 높은 시간 해상도로 데이터를 출력한다. 또한 60dB 정도의 다이내믹 레인지를 갖는 카메라보다 2배 가량 높은 130dB의 높은 다이내믹 레인지를 가져 활용 범위가 더욱 넓다. 따라서 본 논문에서는 동적 환경에서의 활용 가치가 높은 이벤트 카메라를 사용하여, 움직이는 물체가 존재하는 동적 환경에서 물체를 검출하고 카메라 자신의 움직임을 강인하게 추정하고자 한다.

본 논문의 첫번째 항목은 다양한 장면에 강인하면서 낮은 지연 속도로 광학 흐름을 추정하는 연구를 다룬다. 기존 카메라와 다른 이벤트 카메라의 동작 원리로 인해, 다수의 알고리즘은 기존 카메라의 프레임과 같은 가상의 이벤트 프레임을 생성하고, 이를 이용해 기존 알고리즘의 프레임워크를 답습한다. 그러나 가상의 이벤트 프레임을 생성하기 위한 파라미터의 값은 장면의 특성에 따라 크게 좌우되며, 이러한 방식은 이벤트 카메라의 특성 중 하나인 낮은 지연 속도를 증가시킨다. 따라서 본 항목은 광학 흐름을 비동기식으로 추정하여 낮은 지연 속도와 높은 정확도를 달성하는 것을 목표로 하며, 이를 위해 기존 블럭 매칭 알고리즘의 성능을 향상시킨다. 제안한 알고리즘은 국소적으로 블럭 매칭을 수행함으로써 불균일한 텍스쳐 때문에

이벤트가 고르지 못하게 출력되는 상황에서도 정확한 광학 흐름을 추정할 수 있다. 광학 흐름 추정 알고리즘의 정확도와 지연 속도를 측정하기 위해, 대중적으로 많이 사용되는 데이터셋을 이용하였다. 정성적, 정량적 평가를 통해 제안한 알고리즘은 광학 흐름 추정의 정확도 측면에서 타 이벤트 프레임 기반 알고리즘과 비슷한 수준을 유지하면서, 지연 속도 측면에서는 높은 향상을 보여주었다. 또한 제안한 비동기식 광학 흐름을 이용한 각속도 추정 결과는 타 알고리즘 대비 높은 정확도를 보여주며, 특히 다양한 장면 환경에서 15ms 이하의 일관된 지연 속도로 강인하게 각속도를 추정한 것을 확인하였다.

두번째 항목은 모션 분할 기법을 이용하여 움직이는 물체가 존재하는 동적 환경에서 카메라 자신의 각속도를 강인하게 추정하는 연구를 수행하였다. 밝기 변화가 검출되는 픽셀에 대해서만 데이터를 출력하는 이벤트 카메라의 특성상, 작지만 빠른 물체가 출력 데이터의 대부분을 점유할 수 있다. 배경이 아닌 외부 요인에 의해 나타나는 이벤트는 카메라의 움직임 추정 성능을 저하시킬 수 있으므로, 첫번째 항목에서 제안한 비동기식 광학 흐름을 이용하여 모션 분할을 수행하고 정적 배경의 움직임을 추정하고자 한다. 이를 위해, 이중 모드 움직임 모델을 설계하고, 정적 배경과 이동 물체의 위치를 그리드 레벨에서 추정한다. 이중 모드 움직임 모델은 이동 물체의 갯수를 필요로 하지 않으며, 이동 물체가 화면 밖으로 사라지거나 다시 나타나는 상황에도 대응할 수 있다. 제안한 알고리즘은 분할된 정적 배경 영역에 속하는 비동기식 광학 흐름을 이용하여 높은 시간 해상도로 카메라의 각속도를 추정할 수 있으며, 성능 검증을 통해 다양한 실내 및 실외, 낮과 밤의 동적 환경에서 타 알고리즘 대비 강인하고 정확하게 카메라 자신의 움직임을 추정할 수 있음을 확인하였다.

주요어: 이벤트 카메라, 동적 환경, 모션 분할, 각속도 추정, 픽셀영역 내 이벤트, 광학 흐름 추정, 강건성

학 번: 2017-32808