Flow-Based Visual-Inertial Odometry for Neuromorphic Vision Sensors

Mahmoud Khairallah

université
PARIS-SACLAY

Flow-Based Visual-Inertial Odometry for Neuromorphic Vision Sensors

Odométrie Visio-Inertielle Basée sur Le Flot Optique pour Les Capteurs Neuromorphiques de Vision

Thèse de doctorat de l'université Paris-Saclay

École doctorale : n°580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Robotique
Graduate School : Science de l'ingénierie et des systèmes.
Référent Université d'Évry Val d'Essonne

Thèse préparée dans l'unité de recherche **IBISC (Université Paris-Saclay, Univ Evry)** , sous la direction de **Samia BOUCHAFA-BRUNEAU**, Professeure des université, la co-direction de **David ROUSSEL**, Maître de conférénce, le co-encadrement de **Fabien BONARDI**, Maîte de conférénce.

Thèse soutenue à Paris-Saclay, le 16/09/2022 par

Mahmoud Z. KHAIRALLAH

Composition du Jury

Pascal VASSEUR Professeur des université, Université de Picardie Jules Verne	Président
Catherine ACHARD Professeure des université, Sorbonne Université	Rapporteure
Samia AINOUZ Professeure des université, INSA Rouen	Rapporteure
Michèle GOUIFFES Maîtresse de conférences HDR, Université Paris-Saclay	Examinatrice
Omar TAHRI Professeur, Université de Bourgogne	Examinateur
Samia BOUCHAFA-BRUNEAU Professeure des université, Université d'Evry-Val-d'Essone - Université Paris-Saclay	Directrice de thèse

THESE DE DOCTORAT

NNT : 2022UPAST117

université PARIS-SACLAY | ÉCOLE DOCTORALE Sciences et technologies de l'information et de la communication (STIC)

Titre : Odométrie Visio-Inertielle Basée sur Le Flot Optique pour Les Capteurs Neuromorphiques de Vision

Mots clés : Caméra non-traditionnelle, Odométrie visiuelle, Flot optique, détection des lignes

Résumé : Plutôt que de générer des images de manière constante et synchrone, les capteurs neuromorphiques de vision -également connus sous le nom de caméras événementielles, permettent à chaque pixel de fournir des informations de manière indépendante et asynchrone chaque fois qu'un changement de luminosité est détecté. Par conséquent, les capteurs de vision neuromorphiques n'ont pas les problèmes des caméras conventionnelles telles que les artefacts d'image et le Flou cinétique. De plus, ils peuvent fournir une compression sans perte de donné avec une résolution temporelle et une plage dynamique plus élevée. Par conséquent, les caméras événmentielles remplacent commodément les caméras conventionelles dans les applications robotiques nécessitant une grande maniabilité et des conditions environnementales variables. Dans cette thèse, nous abordons le problème de l'odométrie visio-inertielle à l'aide de caméras événementielles et d'une centrale inertielle.

En exploitant la cohérence des caméras événementielles avec les conditions de constance de la luminosité, nous discutons de la possibilité de construire un système d'odométrie visuelle basé sur l'estimation du flot optique. Nous développons notre approche basée sur l'hypothèse que ces caméras fournissent des informations des contours des objets de la scène et appliquons un algorithme de détection de ligne pour la réduction des données. Le suivi de ligne nous permet de gagner plus de temps pour les calculs et fournit une meilleure représentation de l'environnement que les points d'intérêt. Dans cette thèse, nous ne montrons pas seulement une approche pour l'odométrie visio-inertielle basée sur les événements, mais également des algorithmes qui peuvent être utilisés comme algorithmes des caméras événementielles autonomes ou intégrés dans d'autres approches si nécessaire.

Title : Flow-Based Visual-Inertial Odometry for Neuromorphic Vision Sensors.

Keywords : Non-traditional cameras, Visual odometry, Optical flow estimation, Line detection.

Abstract : Rather than generating images constantly and synchronously, neuromorphic vision sensors - also known as event-based cameras- permit each pixel to provide information independently and asynchronously whenever brightness change is detected. Consequently, neuromorphic vision sensors do not encounter the problems of conventional frame-based cameras like image artifacts and motion blur. Furthermore, they can provide lossless data compression, higher temporal resolution and higher dynamic range. Hence, event-based cameras conveniently replace frame-based cameras in robotic applications requiring high maneuverability and varying environmental conditions. In this thesis, we address the problem of visual-inertial odometry using event-based cameras and an inertial measurement unit. Exploiting the consistency of event-based cameras with the brightness constancy conditions, we discuss the availability of building a visual odometry system based on optical flow estimation. We develop our approach based on the assumption that event-based cameras provide edge-like information about the objects in the scene and apply a line detection algorithm for data reduction. Line tracking allows us to gain more time for computations and provides a better representation of the environment than feature points. In this thesis, we do not only show an approach for event-based visual-inertial odometry but also event-based algorithms th at can be used as stand-alone algorithms or integrated into other approaches if needed.

To the person I kept telling I wanted to become a doctor and engineer and always laughed at me and eventually I, somehow, did,

to my mom ...

Acknowledgements

Before delving into this thesis's mathematical and scientific contributions, I would like the reader to spare me a moment to appreciate people without whom this work would have never been completed. I am deeply in debt to Prof. **Samia BOUCHAFA-BRUNEAU**, thesis's directress, even a long time before I started this work. I would like to thank her for her constant support and devotion. Additionally, I could not have undertaken this adventure without the help of Assoc. Prof. **David ROUSSEL**, thesis co-director, and his continual pieces of advice. I have special thanks to Assoc. Prof. **Fabien Bonardi**, thesis co-supervisor. I will never forget how all of my supervisors helped me and gave me the liberty to do research and explore new paths. I also appreciate the exchanges and studies I made with my colleague and friend, Ph.D. student **Abanob SOLIMAN**.

Finally, a big part of me cannot neglect how my friends always stood up for me and helped me whenever I was down. Unfortunately, there's a long list of friends, that I consider my second family, which prevents me from being able to cite all their names. I will always bear in mind, for all of them, what they have done for me.

0.1 Synthèse En Français

L'assimilation des robots à l'industrie pour réaliser des missions nécessitant une intervention humaine persistante a toujours été un des enjeux majeurs qui a retenu l'attention des chercheurs. les propositions à cet enjeu varient selon de nombreux aspects, c'est-à-dire, la nature de la tâche et son contexte (intérieur, extérieur, recherche et sauvetage, conditions sévères, etc.), la configuration du robot, le niveau d'autonomie, la précision et les précautions de sécurité. Avoir un système robotique capable de répondre à ces exigences tout en respectant les aspects mentionnés est une tâche méticuleuse qui nécessite des considérations réfléchies. Le premier défi qui entrave l'assimilation du robot dans les environnements industriels, en particulier s'il s'agit d'un robot mobile, est sa capacité à percevoir l'environnement et à le reconnaître correctement. La perception robotique en tant que tâche implique la capacité du robot à naviguer tout en connaissant sa localisation, en évitant les obstacles et en planifiant un chemin optimal pour accomplir une mission particulière.

De nombreuses techniques de perception robotique ont été introduites dans l'état-de-l'art depuis l'essor des robots industriels pour la perception robotique. Certaines techniques utilisaient des informations acquises à partir des actionneurs du robot, c'est-à-dire l'odomètre et le compteur de vitesse ; ces techniques proprioceptives souffriraient d'une forte dérive. Après l'émergence de la technologie "Micro Electronic Mechanical Systems (MEMS)", d'autres techniques dépendaient des états proprioceptifs du robot acquis à l'aide d'une centrale inertielle (IMU) assistée d'un système de positionnement global (GPS). Ces techniques fourniraient une meilleure précision mais échouent toujours dans de nombreux scénarios. Récemment, les caméras sont exploitées en perception robotique pour leur légèreté et leur consommation d'énergie relativement faible. les caméras sont utilisées pour la perception robotique car elles fournissent des données moins bruyantes et ne souffrent pas de glissement, ce qui offre un haut niveau de précision. Dan le cadre de cet thèse, nous sommes motivés pour étudier la perception robotique basée sur la vision afin

d'améliorer la localisation et la navigation. Bien que les caméras conventionnelles offrent une précision beaucoup plus élevée que les autres capteurs, elles présentent néanmoins certains inconvénients qui limitent les capacités du robot. De plus, ils peuvent échouer dans certains cas extrêmes en raison de leur mode de fonctionnement restrictif. Nous investigueons la substitution des caméras conventionnelles par des capteurs de vision neuromorphiques qui ressemblent à la fonctionnalité de l'œil biologique. Nous étudions les avantages et les contraintes des capteurs de vision neuromorphiques et montrons comment ils peuvent être exploités en perception robotique.

Plutôt que de générer des images de manière constante et synchrone, les capteurs neuromorphiques de vision -également connus sous le nom de caméras événementielles, permettent à chaque pixel de fournir des informations de manière indépendante et asynchrone chaque fois qu'un changement de luminosité est détecté. Par conséquent, les capteurs de vision neuromorphiques n'ont pas les problèmes des caméras conventionnelles telles que les artefacts d'image et le Flou cinétique. De plus, ils peuvent fournir une compression sans perte de donné avec une résolution temporelle et une plage dynamique plus élevée. Par conséquent, les caméras événmentielles remplacent commodément les caméras conventionelles dans les applications robotiques nécessitant une grande maniabilité et des conditions environnementales variables. Dans cette thèse, nous abordons le problème de l'odométrie visio-inertielle à l'aide de caméras événementielles et d'une centrale inertielle. En exploitant la cohérence des caméras événementielles avec les conditions de constance de la luminosité, nous discutons de la possibilité de construire un système d'odométrie visuelle basé sur l'estimation du flot optique. Nous développons notre approche basée sur l'hypothèse que ces caméras fournissent des informations des contours des objets de la scène et appliquons un algorithme de détection de ligne pour la réduction des données. Le suivi de ligne nous permet de gagner plus de temps pour les calculs et fournit une meilleure représentation de l'environnement que les points d'intérêt. Dans cette thèse, nous ne montrons pas seulement une approche pour l'odométrie visio-inertielle basée sur les événements, mais également des algorithmes qui peuvent être utilisés comme

algorithmes des caméras événementielles autonomes ou intégrés dans d'autres approches si nécessaire.

nous commençons par une étude sur l'état de l'art des caméras événementielles en montrant les capacités et les contraintes de ces caméras. Puis nous introduisons les compatibilités de leur mode de fonctionnement avec le modèle mathématique qui régit la création du flux optique. Par conséquent, nous proposons le premier schéma d'odométrie visuo-inertielle basé sur le flux optique. Nous fournissons un algorithme de flot optique capable de fournir une précision compétente par rapport à la famille d'algorithmes de flot optique avec un temps du calcul comarable. De plus, nous continuons à améliorer la qualité du flot optique estimé chaque fois que possible après avoir obtenu l'estimation initiale du flux optique. Néanmoins, en gardant la même notion de réduction de la complexité des solutions que nous proposons, notre algorithme de détection et de segmentation de lignes offre des résultats très précis basés sur des conditions simples. Notre algorithme est libéré de la triangulation ou de la sélection d'images clés, ce qui permet de gagner plus de temps pour le processus d'optimisation. En outre, il peut être réglé pour optimiser différentes tailles de fenêtres glissantes pour une optimisation basée sur la dynamique de l'application et peut supprimer les événements inutiles pour maintenir l'applicabilité en temps réel.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The assimilation of robots into the industry to achieve missions that require persistent human intervention has always been an open question that captured researchers' attention. Answers to this question vary depending on many aspects, i.e., task nature and its context (indoor, outdoor, search and rescue, severe conditions, etc), robot's configuration, level of autonomy, accuracy and safety precautions. Having a robotic system capable of meeting the requirements while respecting the mentioned aspects is a meticulous task that necessitates thoughtful considerations. The first challenge that hinders robot assimilation in industrial environments, particularly if it is a mobile robot, is its capability to perceive the environment and recognize the surroundings correctly. Robotic perception as a task involves the robot's ability to navigate while knowing its localization, avoiding obstacles and planning an optimal path to achieve a particular mission.

Many robotic perception techniques have been introduced in the state-of-the-art since the rise of industrial robots for robotic perception. Some techniques used information acquired from the robot's actuators, i.e., wheel odometer and speedometer; these proprioceptive techniques would suffer from high drift. After the emergence of Micro Electronic Mechanical Systems (MEMS) technology, other techniques depended on the proprioceptive

states of the robot acquired using an Inertial Measurement Unit (IMU) aided by a Global Positioning System (GPS). These techniques would provide better accuracy but still fails in many scenarios. Recently, CMOS[1] vision sensors are exploited in robotic perception for their light weight and relatively low power consumption. CMOS cameras are used for robotics perception because they provide less noisy data and do not suffer from slipping which offers high level of accuracy. Using our level of expertise in computer vision, we are motivated to investigate vision-based robotic perception in order to improve localization and navigation.

Although conventional CMOS cameras provide much higher accuracy than other sensors, still, they have some drawbacks which constrain the robot's motion. Moreover, they may fail in some edge cases due to their restraining mode of operation. In the context of this thesis, we investigate the substitution of conventional frame-based cameras with neuromorphic vision sensors that resemble the functionality of the biological eye. We study the advantages and restrictions of neuromorphic vision sensors and show how they can be exploited in robotics perception.

1.2 Philosophy

We believe that the outcomes of any scientific research would vary profoundly based on the philosophy adopted to approach research, even if the same procedure has been followed. For which reason, we prefer to point out to the reader the philosophy we endorse in this thesis for a comprehensive understanding.

Throughout the work carried out in this thesis, our mindset is always oriented towards finding a solution using minimal resources possible (information, sensors, power consumption and computational power) to achieve the best possible outcome and make sure they

[1] complementary metal-oxide semiconductor

have been fully exploited. We kept this philosophy in mind while constantly asking if the found solution provides satisfying accuracy, runs fast enough and, most importantly, if this solution is the simplest it can be or not. Furthermore, we iteratively asked these questions to improve the outcomes attained. Also, we visit scrutinizingly all the details needed to provide a complete solution while keeping equilibrium without disrupting the big picture. Understanding this philosophy would help knowing why each chapter in this thesis is put in its place the way it is.

1.3 Thesis Outline

In the second chapter, we start by introducing a brief history of robotics and the advancements achieved in the robotics field. We then discuss the meaning of robotic perception and the different types of sensors used in it. Hence, we show why conventional cameras are restraining and what is the proposed frame-based alternatives. We argue why a paradigm shift is needed and present the neuromorphic vision sensors and the current challenges of employing them in robotics. Finally, we present the state-of-the-art of vision-based perception and introduce our proposed solution.

The third chapter discusses the first challenge we approached: How to benchmark new algorithms for novel sensors. We present the different trials to benchmark different types of neuromorphic algorithms. Henceforth, we show how we approached the problem to generate ground-truth that fits for our application.

The following chapters separately show how each required chunk to construct our scheme is studied and improved. The fourth chapter presents why optical flow is consistent with the mode of operation of neuromorphic vision sensors and why we believe optical flow makes an excellent candidate to be used for neuromorphic motion state estimation. Finally, we show how we improve the quality of optical flow for our application.

The fifth chapter presents how we can exploit geometric information in the environment to augment the data provided by neuromorphic vision sensors and exploit lines for better data association. We offer a more straightforward method for neuromorphic line detection and segmentation without disturbing the detection quality or the computational time.

The complete scheme of our neuromorphic visual-inertial odometry system is presented in the sixth chapter. We assemble all the blocks introduced in this thesis in an optimisation scheme, which we believe to be the first to exploit optical flow information for neuromorphic visual odometry. Finally, we conclude the outcomes of our thesis and discuss the possible future work for improvements in the seventh chapter.

Each chapter in this thesis represents a work published in different conferences. In our work published in VISSAP 2021, we proposed an evaluation dataset for event-based optical flow and tested different state-of-the-art optical flow algorithms. Consequently, we discussed the restrictions of each algorithm and how they can be improved. After evaluating and understanding the capacity of the tested algorithms, we introduced a faster and more accurate event-based optical flow algorithm that uses Principal Component Analysis (PCA) and applied different regularization techniques to improve the estimated optical flow, which was published in ICIP 2022 conference. In order to use optical flow for 6-DoF pose estimation, we need to increase the information we know about the environment such as depth and geometry. For this reason, we developed an event-based line detection and tracking algorithm capable of detecting lines in varying situations in real-time. We presented this work in ICPR 2022 conference. In ACCV 2022, we introduce an optimization scheme using optical flow. Based on the proposed optical flow tracking method, we introduce an optimization scheme set up with a twist graph instead of a pose graph. Upon validation on high-quality simulated and real-world sequences, we show that our algorithm does not require any triangulation or key-frame selection and can be fine-tuned to meet real- time requirements according to the events' frequency.

1.4 Contribution

In the context of this thesis, we published:

- Khairallah, M. Z., Bonardi, F., Roussel, D., Bouchafa, S. (2021). Data-set for Event-based Optical Flow Evaluation in Robotics Applications. VISSAP2021
- Khairallah, M. Z., Bonardi, F., Roussel, D., Bouchafa, S. (2022). Flow-Based Line Detection and Segmentation for Neuromorphic Vision Sensors. ICPR 2022.
- Khairallah, M. Z., Bonardi, F., Roussel, D., Bouchafa, S. (2022) PCA Event-Based Optical Flow: A Fast and Accurate 2D Motion Estimation. ICIP 2022

Under submission:

- Khairallah M. Z., Soliman A., Bonardi F., Roussel D., Bouchafa S., (2023) Flow-Based Visual-Inertial Odometry for Neuromorphic Vision Sensors Using Non-Linear Optimization with Online Calibration. VISSAP 2023

2 Vision-Based Perception

Chapter abstract

The ability to perceive for any robot is essential in order to achieve any assigned task properly. In this chapter, we present a brief history of robotics and how a robot is able to see. Then, we demonstrate the limitations of conventional vision sensors and their possible alternatives. Finally, as a solution, we present why neuromorphic vision sensors are suitable for robotic perception and what are the challenges we need to tackle to provide robust robotic self-motion perception solutions.

2.1 Introduction

"Emo, please don't interrupt me! I am trying to work". A sentence that may be said frequently by a person who owns the AI[1] desktop pet EMO produced by the Living.ai company. A cognitive robot that understands its surroundings and develops a personality accordingly. It acts as a tiny flatmate who has a taste for music and may seek your attention many times a day. Such a technology, which combines rigid logical thinking, artistic facial expressions and responsive emotions, is the fruit of a journey of hundreds of years of research. Although it is difficult to trace back the whole timeline of robotics,

[1] Artificial intelligence

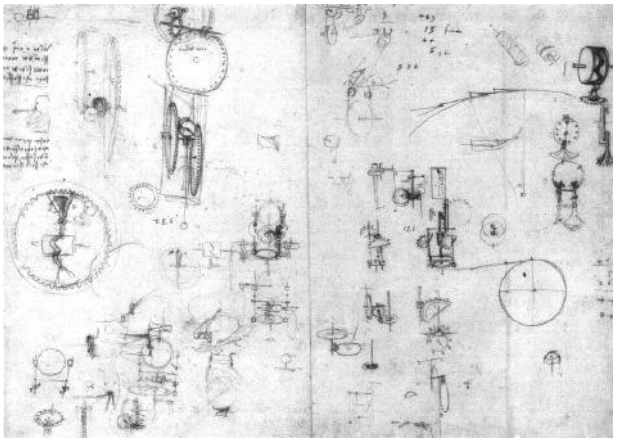
some events are significantly remarkable to play the role of game-changer in the history of robotics. More than eight hundred years ago, Ismail al-Jazari, an Islamic polymath, inventor and mechanical engineer who served as the chief engineer of Artuklu Palace (currently in Turkey), published his book [al Jazarī, 1973] presenting many of his robotic inventions (with illustration). Amongst these inventions, we can find the crank mechanism, connecting rod, reciprocating piston engine, suction pipe, suction pump, double-acting pump, cam, camshaft, segmental gear, the first mechanical clocks driven by water and weights. Most importantly, Al-Jazari demonstrates, in simple "Do It Yourself" language, many humanoid programmable robots that can provide water for ablution or serve wine to the king (see Figure 2.1a).

Roaming from the east to the west about three hundred years later, during the period of renaissance, Leonardo Da Vinci presented his humanoid knight robot amongst his notebooks and drawings in Codex Atlanticus [Rosheim, 2006]. Da Vinci drew accurately different human body joints and depicted them in his robot model (see Figure 2.1c). Although no final design of this robot was found, according to Da Vinci, this robot had sufficient degrees of freedom to imitate human motion. Scientists tried to replicate this robot based on the understanding of his sketches.

Amidst the past century, Joseph Engelberger - father of modern robotics - founded his company which provided the first real industrial robot: The Unimate [Engelberger, 2012]. His robot was electrically powered and used hydraulic arms to perform repetitively dangerous tasks without human interference, e.g., metalworking and welding in car factories (see Figure 2.1b). According to the frequent definitions of modern robots, this robot can be considered the first to be closely defined as a robot (see Definition 2.1).

(A) Illustration of one of Al-Jazari's Robots

(B) Engelberger's Unimate



(C) Da Vinci's sketches for knight robot

FIGURE 2.1: Left: illustration of Al-Jazari's robot that resembles a girl serving water and soap to wash hands [al Jazar2̄, 1973]. Middle: sketches of joints and mechanisms used in Da Vinci's knight [Rosheim, 2006]. Right: Engelberger with his partner G. Dovel next to their Unimate [Engelberger, 2012].

Definition 1 ***Industrial Robot***

A programmable device that can repetitively perceive, analyze and act to substitute humans and does not necessarily resemble human beings[a]

[a] many definitions vary in the state-of-the-art, here we try to provide a concise and informative definition.

According to the industrial robots definition (see Definition 2.1), in this thesis, we shed light on the navigation task of agile robots and how to provide new possible solutions in this domain. Agile robots are the kind of robots able to achieve high manoeuvrability, modify

their configuration and consequently take rapid decisions in order to accomplish relatively tricky tasks. Robotic navigation encompasses many tasks, including obstacle avoidance [Zohaib et al., 2013], environment segmentation [Minaee et al., 2021], and mission and path planning [Souissi et al., 2013], all of which depend on how the robot *perceives* the world.

2.2 Robotic perception

Robotic perception is the robot's ability to interpret changes, which happen either to the robot itself or its surrounding environment, using built-in sensors the same way human beings use their senses. For example, if a robot moves one step forward, it should consequently perceive that the whole environment is shifted one step closer (assuming a static environment). Also, if an object is moved from its initial position, the robot should recognise this modification. Sensors used in robotics vary in their nature, capabilities and applications. These sensors can be classified into two groups [Rubio et al., 2019]:

- **Proprioceptive/Exteroceptive:**

  Proprioceptive sensors are able to measure quantities belonging to the robot itself, e.g. battery voltage, joint angles, acceleration and velocities. Inertial measurement unit (IMU), wheel encoder and potentiometers belong to these sensors. Exteroceptive sensors capture information from the surroundings. This information can be light intensity, travelled distances, sound amplitude or electromagnetic waves. Some of these sensors are Charge-Coupled Device/Complementary Metal-Oxide-Semiconductor (CCD/CMOS) cameras, Ultrasonic sensors or a Global Positioning System (GPS).

- **Active/Passive:** Active sensors diffuse energy in the environment and evaluate its response, then concludes a representation about the environment. For instance, Laser rangefinders, reflectivity sensors and magnetic encoders. On the other hand,

sensors that only absorb energy emitted by the environment are called passive sensors like temperature sensors, compasses and CCD/CMOS cameras.

According to this classification, each group is considered effective in specific applications and environmental setup. Consequently, based on our area of expertise, we focus on vision-based perception in scenarios that require agility and manoeuvrability. Vision-based perception systems can use conventional stereo cameras, RGB-D cameras or monocular cameras aided with sensors fusion to improve estimation quality and tackle monocular-camera-based issues such as absolute scale ambiguity. In order to attain a power-efficient, lightweight and low-cost system, this thesis is focuses on monocular vision-based sensor fusion systems. Monocular conventional vision systems would provide acceptable solution to many robotic applications, however, they may undergo serious issues when used in challenging scenarios. In the following section we demonstrate the strengths and weaknesses of CMOS/CCD vision sensor.

2.3 Conventional Vision Sensors

To provide more comprehensive awareness to a machine of its surroundings, conventional cameras [Wanlass and Sah, 1991] were introduced in robotics [Myers, 1980] as a reasonable candidate for their relatively low power consumption and low data transmission latency. Since the eighties and until today, standard frame-based cameras have proven acceptable performance in many fields from surgery [Gumbs et al., 2021], food processing industry [Zhu et al., 2021], product assembly and quality control [Silva et al., 2018], agriculture [Paul et al., 2020] to self-driving cars [Yaqoob et al., 2019] and many other fields (see Figure 2.2). Standard cameras' mode of operation provides synchronously acquired images at relatively low frequency (from 20 to 120 Frames per second). Although these cameras function well in many applications, their mode of operation can be undermined in scenarios where abrupt changes or high manoeuvrability are present such as handheld systems and drones.

(A) Fanuc's vision-aided robotic arm in car's production line [Fanuc, 2022]

(B) First google self-driving car[waymo, 2022]

(C) Virtual Incision surgical vision-aided robot[Incisio, 2022]

FIGURE 2.2: Examples of well-known vision-aided robots in different fields of industry

2.3.1 Advantages

Being a subject of consistent development for the last fifty years, standard cameras' chips have reached a size where a high definition camera can be embedded in a less than one-centimetre thick mobile phone. These chips have a sufficiently large amount of photoreceptors where the signal provided by each one can help create crisp and sharp images. The voltage created by each pixel is quantized with sufficient resolution to maintain distinctive colours. Hence, sharp images with relatively accurate photometric information can be acquired. The richness of photometric information provided by standard cameras can play a vital role for a machine to distinguish the surrounding environment.

In modern digital cameras, each pixel is a whole electrical circuit containing the photoreceptor, signal amplifier, analog to digital converter, and other components. The ratio between the surface area of the photoreceptor to the whole pixel area is called the pixel's fill factor, where standard cameras are endowed with sufficiently large fill factor (can reach 70%). Consequently, the provided photometric information would have acceptable Signal to Noise Ratio (SNR) in good lighting conditions because larger photoreceptor area allows better acquisition of light intensity.

Finally, one advantage that is indispensable, especially for standard cameras, is that they have been integrated into the industry for more than fifty years. We find many fast-growing companies such as Aquifi which employs machine vision and deep learning to provide low-cost solutions for logistics in industrial environments and LMI Technologies which exploits machine vision to automate factory lines inspection. For the future, according to the market research conducted by Mordor Intelligence, the Compound Annual Growth Rate (CAGR) of robotic vision market is expected to gain 9.86% by 2027 in spite of the impact of COVID-19 pandemic. Based on the current state, the amount of investment put in machine vision and future expectations, vision-aided systems gained a level of maturity and reliability, and as a result, most of state-of-the-art vision algorithms are developed for these sensors.

2.3.2 Limitations

Although standard cameras are efficient enough in most cases, some scenarios would exceed the capabilities of these cameras to some edge cases where their performance may deteriorate. Standard cameras are designed to function in a synchronous manner with a predetermined frame rate where each frame should wait for information from all the pixels to be acquired. This intrinsic feature to standard cameras implies a build-up latency for data acquisition and latency to be able to transmit such an amount of data. Data acquisition and transmission latencies restrict standard cameras from providing a high frame rate. Consequently, in scenarios where camera motion or environment changes fast enough, the acquired frames may suffer from blurry patches (see Figure 2.3a). Blurry images impair the quality and performance of machine vision systems.

In most cases, capturing full-frame data does not add up to provide any additional knowledge other than the previous frames. Such operating mode results in a lot of data

(A) A camera moving fast to focus on the car causing the surrounding highly blurry[Photography, 2022].

(B) A captured sunset where spots of the image look black to tolerate the low dynamic range[Mob, 2022].

(C) A grainy noisy image due to low lighting conditions[Qin, 2022].

FIGURE 2.3: Cases where standard cameras may fail to deliver acceptable performance.

redundancy[2] which means that the efficiency of the lossless compression of the obtained data is not optimal. Nevertheless, the data redundancy -other than being useless- may not only require more power consumption but also cost more computational power to process all the acquired data. The power consumption and computational power hinder the reliability of standard cameras where lightweight is a critical requirement.

Abiding to standard frame-based nature, the collected signal should be quantized and normalized in a way that allows encoding most of the data to be visible according to the dynamic range. Consequently, based on the lighting conditions, some parts of the image may appear much brighter or darker according to the camera dynamic range. By definition, the dynamic range is the ratio between the maximum possible signal voltage and the noise floor under dark conditions, which is described by the equation [Posch et al., 2010a]:

$$DR = 10 \log \left(\frac{V_{sat}^2}{V_{dark}^2 + V_{reset}^2 + V_{out}^2} \right) \tag{2.1}$$

Where V_{sat} is the maximum allowed voltage at the integration node and V_{dark} is the dark-current voltage, V_{out} is the readout noise voltage and V_{reset} is the reset voltage. Since in most standard cameras the exposure time and integration capacitance are held constant, then most of standard cameras provide a saturation linear dynamic range limited to 60-70

[2] data redundancy varies depending on the dynamics of the scene

dB whilst the dynamic range of natural scenes would exceed 140 dB [Xiao et al., 2002] (see Figure 2.3b).

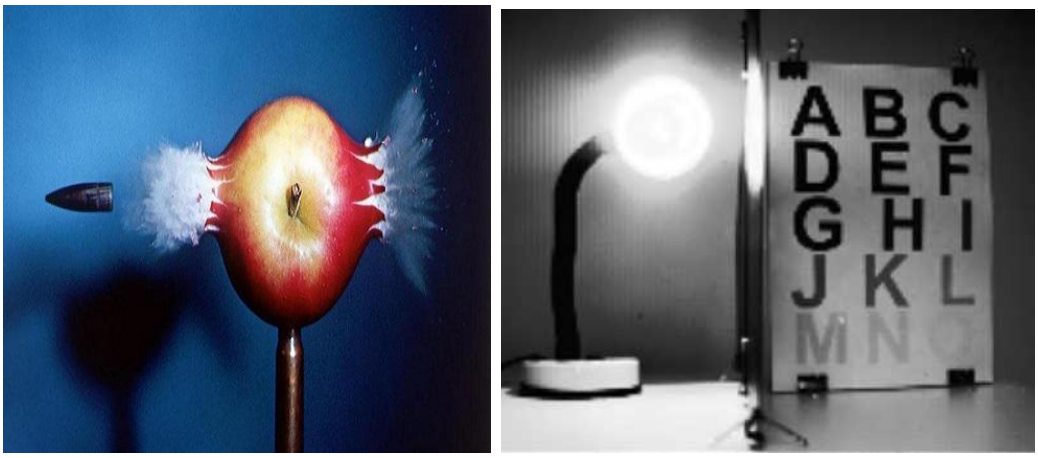
Finally, to ensure that the signal is transmitted without being corrupted due to noise can be guaranteed whenever the ratio between the signal and noise is sufficiently large and can be modeled as [Posch et al., 2010a]:

$$SNR = 10 \log \left(\frac{V_{sig}^2}{V_{dark}^2 + V_{photo}^2 + V_{reset}^2 + V_{out}^2} \right) \tag{2.2}$$

Where V_{photo} is the photo-current shot noise which is the dominant noise source. According to the definition and since the SNR is proportional to V_{sig} and the dominant term in the denominator is V_{photo}, it is seen that for standard cameras, SNR is strongly dependent on the scene illumination (see Figure 2.3c), which is not a stable measure for SNR. In order to be able to find a more stable measure of SNR, either modifying of the model itself or a whole paradigm shift is required so that the dependence of scene illumination would be eliminated.

2.3.3 Frame-Based Alternatives

As a trial to tackle some of the problems that arose from standard frame-based cameras, researchers developed high-speed cameras [Honour, 2009]. Unlike most standard frame-based cameras that are limited to $\backsim 30\ fps$, high-speed cameras can reach more than thousands of frames per second. As a result, these cameras can help in many applications requiring abilities that surpass the human eye (see Figure 2.4a), like transitional motion that happen within a fraction of a second. Although these cameras come up with a solution for the problem of high maneuvers and abrupt changes in the environment, they provide approximately the same dynamic range and redundant data as standard cameras. Also, other issues emerge if we adopt a frame-based mode of operation for high-speed cameras. Recording around thousand fps of high definition uncompressed



(A) Image of a bullet passing through an apple captured at high frame rate[School, 2017].

(B) A high dynamic range image where a light bulb is on while other details are sufficiently visible[Vargas-Sierra et al., 2014].

FIGURE 2.4: Alternatives provided in the state-of-the-art to encounter the limitations of standard frame-based cameras.

color images implies registering $(1000 \times 3 \times 1280 \times 720)$ bytes$\backsim 2.6Gb$ for only a single second. Generating a large amount of data in a small interval of time demands high data transmission capabilities in order to guarantee real-time communication. Even though lossless image compression would reduce the size of an image to a few megabytes, the capability to transfer the compressed data using the current technology is still questionable. Furthermore, processing such an amount of data in real-time is far beyond the affordable computational power already existing of currently available processing devices in the industry.

In environments where luminosity bandwidth is large enough, researchers proposed a high dynamic range CMOS camera [Vargas-Sierra et al., 2014] to expand the captured signal quality (see Figure 2.4b). The expanded dynamic range brings the black (or white) areas less indistinguishable. Consequently, the quality of algorithms using these sensors would provide better results. Still, the proposed solution suffers from low frame rate data and a very small array size (180×148) providing very pixelated images.

Therefore, with the proposed solutions, we are able to conquer the frame-based restriction in robotics applications once at a time if we are going to adopt the same mode of operation. However, adopting only a frame-based mode of operation seems unable to reach an equilibrium point to attain a comprehensive solution that does not suffer from low frame rate and dynamic range. From this point of view where solving an issue leads to another, a paradigm shift from the standard frame-based world is sought. Therefore, scientists proposed bio-inspired sensors mimicking biological processes to achieve optimal results in the past decade, including sensors that can replace standard frame-based cameras. These sensors imitate the human retina and are called neuromorphic vision sensors.

2.4 Neuromorphic Vision Sensors

Besides the issues present in frame-based chips (see Section 2.3.2), another issue in most of today's chips is that they depend on Von-Neumann architecture. Von-Neumann architecture isolates the memory and Central Processing Units (CPU) which entails the data to go back and forth between them. Although the advantages Von-Neumann architecture has, data transfer results in a significant waste of time and loss of optimality. A proposed non-Von-Neumann architecture is neuromorphic computing. The term Neuromorphic refers to processes that mimic the human brain and the functioning of the nervous system where, unlike computers, brains use massively parallel interconnected neural computing. The neuromorphic concept was first introduced in the 1980s as neuromorphic computing [Schuman et al., 2017]. In 1991, Mahowald and Mead published the first silicon neuromorphic chip called the silicon retina [Mahowald and Mead, 1991]. The electronic circuit of neuromorphic vision sensors (also called event-based cameras) simulates the maturely understood physiological parts of the human retina, namely, rods and cones cells, bipolar cells, and ganglion cells. Many versions of event-based cameras have been developed since 1989 that adopt different features. Mainly, each pixel in event-based cameras operates, unlike frame-based cameras, asynchronously and independently. Instead

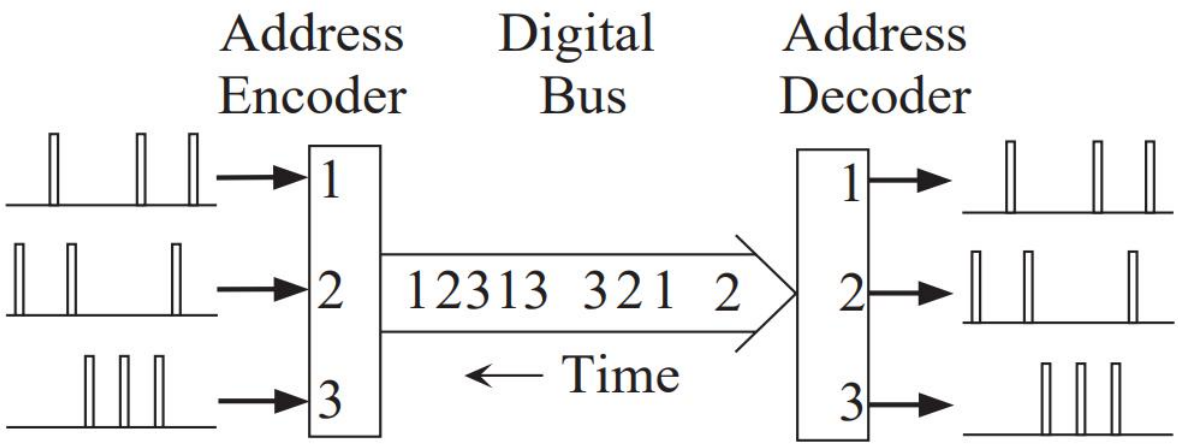
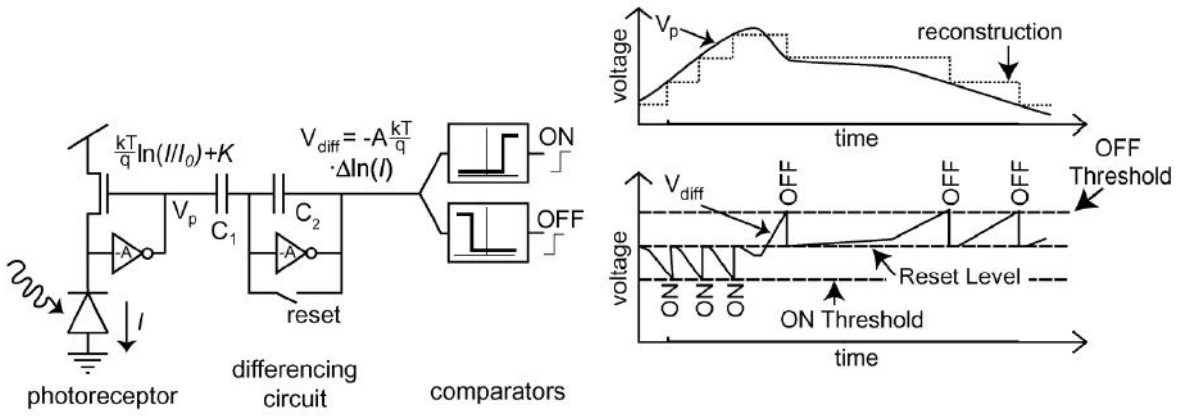


FIGURE 2.5: Address-event representation between a transmitter and a receiver chip where spikes are treated serially by the encoder and then decoded into their original form by the decoder for the receiver chip [Boahen, 2000].

of transmitting fixed frequency frames of photometric data, each pixel triggers a binary spike (an event) according to specific condition.

2.4.1 Address-Event Representation

In order to simulate the human brain nature where neurons communicate densely in parallel, neuromorphic systems use an address-event communication protocol. In address-event representation (AER), a cell creates a spike (event) independently whenever its internal state exceeds a certain threshold so that sparse spikes are communicated over a narrow channel [Boahen, 2000]. Figure 2.5 shows a schematic where an arbiter encodes each cell creating a spike (an event) with a unique address, then the decoder on the side of the receiver chip decodes the received signal into its appropriate original form. The transmitted data contains the spike address, and its polarity where the frequency of the received signal represents its intensity. Many neuromorphic vision systems have been developed based on the address-event representation [Cottini et al., 2013, Brandli et al., 2014, Serrano-Gotarredona and Linares-Barranco, 2013]. In the context of this thesis, we focus on two types of neuromorphic devices, namely, the Dynamic Vision Sensors (DVS) and Asynchronous Time-based Imaging Sensor (ATIS).



(A) A schematic of DVS circuit showing the photoreceptor, differencing circuit and comparator

(B) a graph showing the input voltage created by the light exposed to the photoreceptor and the output of the circuitry.

FIGURE 2.6: The DVS system proposed in [Lichtsteiner et al., 2008]

2.4.2 Dynamic Vision Sensors

One of the mature milestones counted in neuromorphic vision history was proposed by [Lichtsteiner et al., 2008] as dynamic vision sensors. Dynamic vision sensors (DVS) operate as change detection devices where each pixel fires, independently and asynchronously, an event whenever its exposed light intensity exceeds a certain threshold. The DVS retina simulates the human eye differential functionality capable of detecting change by incorporating a circuit containing a photoreceptor, a differencing circuit and two-resistor comparators. The photoreceptor transmits a logarithmic response according to the amount of received light. The differencing circuit receives the signal transmitted by the photoreceptor and amplifies it. Whenever the differencing voltage exceeds a certain threshold, its signal is reset. The two-resistor comparators send either a positive or a negative signal based on the signal received from the differencing circuit (see Figure 2.6). The output of DVS is a stream of events as a tuple $\langle x, y, p, t\rangle$ where x and y are the pixel position on the retina where the event is triggered, p is the polarity of the event (a positive event represent an increase in luminosity and a negative one represents the opposite). DVS

devices are mathematically described as:

$$\Delta L(x_i, y_i, t_i) = L(x_i, y_i, t_i) - L(x_i, y_i, t_i - \Delta t) = p_i \delta_l \tag{2.3}$$

Where $\Delta L(x_i, y_i, t_i)$ is the logarithmic luminosity change exposed to a specific pixel position (x_i, y_i) at time t_i responsible for firing an event with the polarity p_i if the luminosity exceeds the limit δ_l. The asynchronous and independent nature of each pixel where no redundant data is being transmitted allows a faster data transmission with microsecond latency.

2.4.3 Asynchronous Time-based Imaging Sensors

DVS devices can only provide binary events indicating if a change occurred in the environment or not. In order to tackle the lack of any photometric information in the environment, another milestone was proposed by [Posch et al., 2010a] as Asynchronous Time-based Imaging Sensors (ATIS). ATIS devices incorporate the same circuitry as DVS devices with an additional exposure measurement circuitry responsible for photometric information. The usage of the time-domain (or Pulse Width Modulation PWM) method, where light intensity is measured by the time it takes photocurrent to produce a certain voltage, is useful for the improvement of DR and SNR.

The exposure measurement circuit is realized as a time-based PWM circuit based on two global integration thresholds (V_{refH} / V_{refL}) (see Figure 2.7). With the increment in photo-voltage an event is fired and pulse is created, this pulse initiates an exposure measurement cycle. Photometric information is interpreted as an inversely proportional value to time needed for reference voltage integration.

The Pulses created by the change detection circuit closes the switch that connects the exposure measurement circuit where the logic control ensures that V_{refH} is connected as

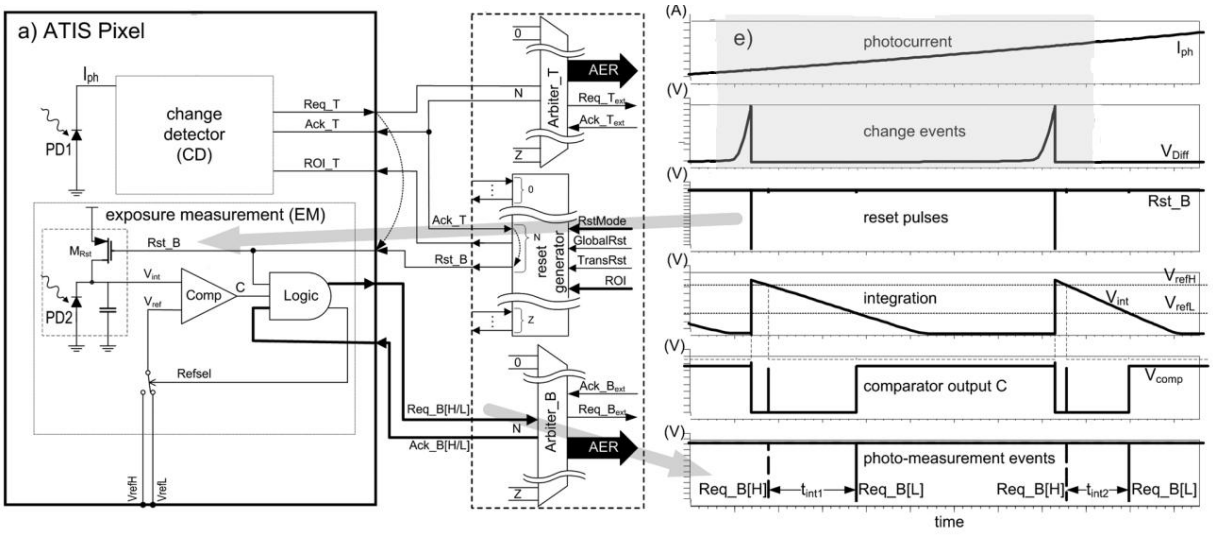


FIGURE 2.7: the scheme to the left shows the ATIS pixel two circuits and the graph to the right explains (from top to bottom) the photo-voltage, the events fired due to change detection, pulses that trigger the exposure measurement, the output of C comparator and the integration time measurements

reference voltage. Then integration starts, thus the voltage V_{int} decreases proportionally to the photo-voltage (illumination). At the point where the comparator C toggles (reaches V_{refH} voltage), it activates the $Req\_B[H]$ signal whilst the logic control changes the reference voltage to V_{refL}. The voltage will continue to decrease and when the voltage reaches the reference voltage V_{refL} it activates another request $Req\_B[L]$. At this point, the illumination exposed to the photo-diode is encoded as the time between the two requests $Req\_B[H]$ and $Req\_B[L]$ which is inversely proportional to the illumination intensity.

Whilst there's a possibility that the illumination in the scene is high enough that the change detection circuit would detect another event before the end of the exposure measurement cycle is terminated (which is interpreted by the reception of $Req\_B[H]$ signal). In this case, the first request is discarded and the illumination is incremented by the amount of voltage threshold predefined for event detection.

2.4.4 Challenges

The rise of event-based vision sensors at the beginning of the last decade captured the attention of many researchers to augment the performance of standard cameras. In [Gallego et al., 2022] a survey is provided to show the different applications where Event-Based cameras were introduced. From the primary elements needed like Feature detection and tracking [Drazen et al., 2011], [Gallego et al., 2022], optical flow estimation [Benosman et al., 2013], [Delbrück, 2008] and depth estimation [Rebecq et al., 2018], [Rebecq et al., 2017] to algorithms like tracking and mapping [Weikersdorfer et al., 2014], [Kim et al., 2016] and visual-inertial odometry algorithms [Mourikis and Roumeliotis, 2007], [Forster et al., 2016]. The paradigm shift proposed by Event-Based cameras in the nature of acquired data, latency and camera characteristics like the dynamic range and signal to noise ratio has introduced some challenging opportunities that can be summarized as:

- **Algorithms reinvention:** It is evident that adaptations or maybe total changes in computer vision algorithms are required due to the different natures of cameras. Frame-based algorithms take a sequence of synchronous images, and event-based cameras output a stream of independent asynchronous events that depend only on scene brightness and motion change. Such a difference demands total changes in the algorithmic level and the hardware used to process event-based data.
- **Noise handling:** All sensors provide the desired signal with added noise due to non-idealities in electronics. For imagery Sensors, transistors noise also participate broadly. Thus, noise modelling and suppression is vital as they would slow down the deployment of Event-based cameras in the industry and real-life scenarios.
- **Data association:** An event contains only information about the change of lighting of the scene. The question is how to provide concrete mathematical models able to exploit such a low amount of information to gain more information about the environment.

- **Benchmarking:** Event-based cameras are still considered an evolving research topic where improvements and changes are being made every day. Consequently, there is a lack of datasets able to cover all possible day-to-day scenarios where event-based cameras would be deployed. Moreover, event-based cameras, being under constant evolution, most of the algorithms present nowadays still did not unlock the full potential of event-based cameras to have an objective judgement of event-based camera capabilities.
- **Real-time applicability:** Event-based camera offers a great reduction of data to be transmitted or processed, however, they can provide up to several millions of events per second which would raise many questions about the ability of state-of-the-art algorithms to process this amount of data in real-time.

With these challenging opportunities, we focus, in this thesis, on how can they be approached to propose proper solution in the field of vision-based navigation based on neuromorphic vision.

2.5 Vision-based motion estimation and recognition

Robotic autonomous navigation is a sophisticated process that includes many tasks that vary in complexity and importance. These tasks can be summarized as robot's state estimation, path planning, trajectory generation, obstacle avoidance and mapping of the surroundings. Furthermore, being one of the sensors providing rich information about the surroundings, cameras (and vision sensors) are highly included in robotic autonomous navigation, especially in state estimation, obstacle avoidance and mapping.

Using vision sensors to estimate ego-motion was discussed in [Longuet-Higgins, 1981] where motion is estimated based on the correspondence of repeatable visual information to find the 3D back projection of 2D image points and hence recover a 6-DOF

motion using stereo image pairs or an up-to-scale 6-DOF estimation using monocular image pairs. Since then, this problem has been known as Structure From Motion (SFM), or less commonly, Structure and Motion (SaM) because it reconstructs the 3D points from the camera's motion and then the camera motion based on triangulation of the reconstructed 3D points. Many attempts have been proposed [Bolles et al., 1987], [Koenderink and Van Doorn, 1991], [Kanade and Morris, 1998], [Jacobs, 2001] to provide a higher estimation quality and faster calculations. A robust solution to SFM problem with real-time capabilities[3] was introduced in [Nister, 2003] where a faster factorization method is proposed side by side with RanSaC [Fischler and Bolles, 1981] for better outlier rejection. SFM is capable of reconstructing ordered or non-ordered sets of images to estimate motion and reconstruct the environment accurately. Another group of machine vision algorithms that shares most of the characteristics of SFM algorithms is visual odometry(see Figure 2.8), where only ordered images can be used to estimate consecutive relative poses with a higher level of optimization and less computational time. The term odometry first appeared in [Nistér et al., 2004] inspired from wheel odometry, which is responsible for incremental motion estimation using data encoded from a vehicle's wheels rotation. On the contrary to wheel odometry, visual odometry does not suffer from slipping leading to high drift over time. Visual odometry refines the estimated poses and structure using local optimization of pose graph [Mur-Artal et al., 2015a] or bundle adjustment [Qin et al., 2018]. Bundle adjustment optimizes the projection error of matched features between only existing images in the sliding window to reduce the computational time, which results in local optimization. Locally optimizing a certain number of recent features leads to incremental drift due to the accumulation of estimation errors. Correcting the accumulated drift may be done with the aid of GPS [Parra et al., 2011], IMU [Weiss and Siegwart, 2011] or LiDaR systems [Graeter et al., 2018]. Simultaneous Localization And mapping (SLAM) algorithms obtain global drift correction that would

[3]the time required to process the data is less than the time at which they are acquired

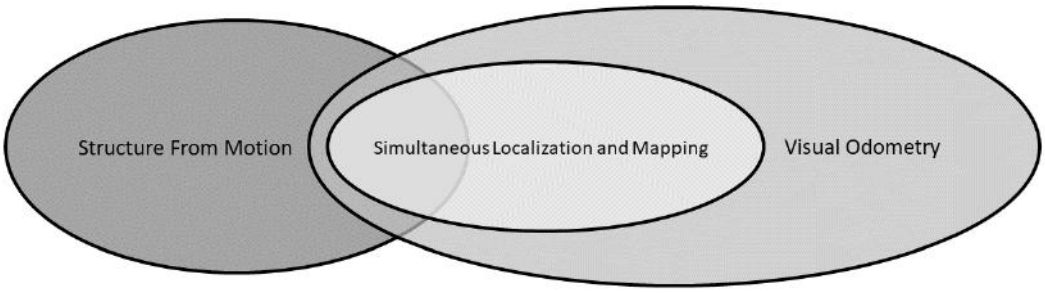


FIGURE 2.8: different machine vision motion estimation groups

require a global bundle adjustment which is implemented whenever loop closure is detected if a place is revisited using place recognition algorithms like the bag of visual words [Gálvez-López and Tardos, 2012]. SLAM can be considered a visual odometry that applies global bundle adjustment and place recognition, which leads to higher estimation accuracy and requires more computational time. In the context of this thesis, we are interested in understanding and improving event-based visual odometry.

In the realm of frame-based visual odometry, algorithms can be categorized as Sparse, semi-dense or dense methods. The sparse method optimizes the pose over distinctive matched points like corners or features in the scene. The semi-dense method uses only high gradient pixels for optimization. The dense method applies optimization directly to every pixel of the image. The inability to estimate accurate depth using only a single image with no prior information about the size of objects captured in the scene favored the usage of stereo cameras for visual odometry if no sensor fusion is used. Stereo Vision visual odometry first appeared to tackle the problem of planetary rover's navigation [Moravec, 1980] featuring a complete pipeline that still inspires many of today's works. In one of the first

attempts for sparse stereo motion estimation, [Matthies and Shafer, 1987] detected and matched corners to find the pose between successive stereo pairs, and the covariance matrix of this transformation is used in a Kalman filter scheme to estimate a local model. [Lacroix et al., 1999] improved the motion estimation by choosing correspondence points differently, and they are selected using the cross-correlation of disparity map between each stereo pair. Many other attempts have been carried out between 1980 and 2004 to improve vision-based motion estimation until the work of [Nistér et al., 2004] that explicitly coined the name visual odometry. In [Nistér et al., 2004], vision-based motion estimation for both stereo and monocular schemes was discussed on long-range real-time scenarios. Instead of using 3D to 2D projection, [Comport et al., 2007] used the quadrifocal tensor for direct 2D to 2D projection and no need for triangulation. [Kitt et al., 2010] used the trifocal tensor of related features of three images of the same scene augmented with an iterated sigma point Kalman filter to correct for non-linearity.

Parallel to research carried out to improve visual odometry based on sparse correspondence points, the dense method (also called direct methods) applies optimization on every pixel. [Horn and Negahdaripour, 1987] predicts image derivatives and finds the motion that minimizes the error between predicted and measured values on a brightness function (the image) of a monocular camera which was later improved with a better mathematical model in [Horn and Weldon, 1988]. [Stein and Shashua, 1996] extended Horn's work based on the work present in [Shashua and Hanna, 1995] to estimate a dense depth map of the scene by adding a third view to the optimization process. To boost the accuracy of visual odometry algorithms and with the rise of smartphones and augmented vision sensors, vision algorithms were aided mainly using either depth sensors or inertial measurement units. For example, [Tykkälä et al., 2011] used an RGB-D sensor to optimize a bi-objective function that combines the depth and photometric information but achieved a low frame rate due to the heavy optimization process. The same year, [Steinbrücker et al., 2011] proposed a faster RGB-D visual odometry algorithm that minimizes the back-projection error based

on lie algebra representation. In [Weiss et al., 2012], vision systems for aerial vehicles scenarios are augmented also using inertial measurement unit in a Kalman filter scheme to correct for more accurate pose and absolute scale factor.

A novel approach is introduced in 2013 to compromise between the trade-offs of either sparse or dense methods: The semi-dense approach. Semi-dense approach was first introduced by [Engel et al., 2013] that provides the accuracy of dense tracking while running in real-time since it does not use feature extraction or matching algorithms and depends solely on high gradient pixels. [Forster et al., 2014] proposed a probabilistic mapping to reject outliers and estimate better accuracy with a sufficiently high frame rate. With the advantages of semi-dense visual odometry, [Forster et al., 2016] augmented [Forster et al., 2014] with IMU measurements where the IMU measurements are presented on manifolds for better IMU error representation.

The change in the nature of vision sensors proposed by the event-based cameras required a paradigm shift on how the visual odometry problem is modeled and how it can be solved. During the past decade, many attempts were introduced where some have adapted the acquired data from event-based cameras to suit frame-based algorithms, while others reformulated the problem to fully exploit event-based capabilities. A novel method was presented in [Weikersdorfer and Conradt, 2012] using a particle filter for motion tracking to estimate the camera's rotation by creating mosaic images of the scene, while an extended Kalman filter is used to refine the gradient intensity results. In [Weikersdorfer et al., 2013], a particle filter is used to estimate the 2D motion of the used rig based on the work presented in [Weikersdorfer and Conradt, 2012] and a 2D map was simultaneously reconstructed. [Mueggler et al., 2014] developed a 6-DOF motion estimation for simple, uncluttered and structured environments that contain lines where the pose is estimated by minimizing the reprojection error of each detected line in the environment. [Rebecq et al., 2016b] proposed an event-based tracking and mapping

method to estimate the pose based on image alignment by warping event images using Lucas-Kanade method [Baker and Matthews, 2004] and constructed the map based on the event-based space-sweep present in [Rebecq et al., 2016a] to provide depth and 3D map. [Kim et al., 2016] pursued their work in [Kim et al., 2008] using also extended Kalman filter to estimate pose, gradient intensity and mapping implemented using a GPU. [Chamorro Hernández et al., 2020] Presented a Lie-Based Kalman filtering method fir tracking based on lines estimation for high speed applications.

Enhancing the robustness and accuracy of event-based cameras can be done, similar to standard cameras, by augmenting the camera with either other sensors, frame-based RGB-D cameras or another event-based camera for stereo-vision. [Censi and Scaramuzza, 2014] provided 6-DOF visual odometry by fusing the event-based camera with a CMOS camera where only rotation was accurately estimated and translation suffered deteriorated accuracy. [Kueng et al., 2016] tracked the feature detected in a CMOS image frame using the event-based camera and used a Bayesian depth filter to estimate the depth of 2D tracked features and obtain 3D points, which are used to minimize the reprojection error between 2D features and 3D points to estimate 6-DOF pose. [Weikersdorfer et al., 2014] used an extrinsically calibrated RGB-D sensor with an event-based camera to have an accurate transformation of each depth value in the events' frame and applied a Bayesian particle filter to estimate 6-DOF pose and a map. In a stereo setting, [Zuo et al., 2022] proposed a semi-dense depth map using An event camera assisted with a depth sensor.

Using an Inertial Measurement Unit (IMU) helps improve estimates provided by monocular camera and obtain accurate absolute scale. [Zihao Zhu et al., 2017] tracks features using optical-flow-based expectation minimization and then uses the tracked features with IMU measurements in a structure-less Kalman filter scheme for pose estimation. [Mueggler et al., 2018] used splines on the manifold for better representation of IMU

Algorithm	Sensors	DOF	Method
[Kim et al., 2008]	DVS	3-DOF	Kalman filtering
[Weikersdorfer et al., 2013]	DVS	3-DOF	Particle filtering
[Weikersdorfer and Conradt, 2012]	DVS	3-DOF	Particle filtering
[Mueggler et al., 2014]	DVS	6-DOF	reprojection error minimization
[Rebecq et al., 2016b]	DVS	6-DOF	Keyframe optimisation
[Kim et al., 2016]	DVS	6-DOF	Kalman filtering
[Chamorro Hernández et al., 2020]	DVS	6-DOF	Lie-Based Kalman Filtering
[Censi and Scaramuzza, 2014]	DVS/CMOS	3-DOF	Bayesian filtering
[Kueng et al., 2016]	DVS/CMOS	6-DOF	reprojection error minimization
[Weikersdorfer et al., 2014]	DVS/RGB-D	6-DOF	Particle filtering
[Rebecq et al., 2017]	DVS/CMOS	6-DOF	Keyframe optimisation
[Zihao Zhu et al., 2017]	DVS/IMU	6-DOF	Kalman filtering
[Mueggler et al., 2018]	DVS/IMU	6-DOF	reprojection error minimization
[Vidal et al., 2018]	DVS/CMOS/IMU	6-DOF	Keyframe optimisation
[Zuo et al., 2022]	DVS/Depth	6-DOF	reprojection error minimization
[Le Gentil et al., 2020]	DVS/IMU	6-DOF	reprojection error minimization

TABLE 2.1: Visual Odometry algorithms and the sensors used for each one, the degrees of freedom of the estimated states and the their estimation method

readings and minimized the geometric reprojection and IMU error for 6-DOF pose estimation. [Vidal et al., 2018] proposed a SLAM system that combines an event-based camera, CMOS camera and an IMU to estimate an accurate scheme where they depend mainly on their work [Rebecq et al., 2017] based on feature tracking and non-linear keyframe optimization. Instead of depending on point features, [Le Gentil et al., 2020] exploited the geometric characteristics of the environments and used lines to estimate ego-motion of the camera (see Table 2.1).

The algorithms presented in the state-of-the-art of event-based cameras (see Table 2.1) vary in their approach, estimated states, used sensors and performance. Although they provided varying estimation accuracy, these algorithms prove that event-based cameras can afford reliability for motion estimation (with certain limits) and that they can tackle the problems faced using frame-based vision sensors such as varying lighting conditions [Rebecq et al., 2016b], [Vidal et al., 2018] and aggressive maneuvers [Mueggler et al., 2014], [Kim et al., 2016], [Rebecq et al., 2016b], [Rebecq et al., 2017]. Additionally, most of

these methods can run in real-time if events' frequency does not exceed a certain threshold[4]. Despite the fact that event-based cameras adopt an operation mode that differs from frame-based cameras, the introduced algorithms depend largely on concepts embraced for frame-based techniques such as feature extraction and keyframe optimization.

Event-based cameras trigger events whenever a change is detected. This operation mode is coherent with the brightness constancy condition (as will show in the following chapters), which explains how optical flow is estimated. Although many advancements have been introduced in event-based optical flow estimation [Benosman et al., 2013], [Low et al., 2020], [Almatrafi et al., 2020], we notice that no work introduced in the-state-of-the-art to exploit the consistency of even-based cameras operation mode and optical flow estimation for 6-DoF state estimation and only [Zihao Zhu et al., 2017] used optical flow as a method to feature tracking.

In the context of this thesis, we explore the reliability of event-based optical flow and how to fully exploit it for 6-DoF motion estimation. Additionally, we discuss the ability to liberate event-based cameras from keyframes or triangulation. Furthermore, since any perfect 6-DoF motion estimation algorithm will not be reliable unless it runs in real-time, we also discuss how depending on optical flow can provide the ability to control computational time based on events frequency.

2.6 Flow-Based Visual-Inertial Odometry

The problem of visual odometry, as explained, is the problem of trying to estimate motion states using a vision sensor. The output of a vision sensor (sequence of images for frame-based cameras $\{I_i\}_N$ or event packets for event-based cameras $\{E_i\}_N$) are used to estimate a 6-DOF pose. The camera pose can be represented by a 4×4 rigid body

[4]The usage of GPU may be needed

transformation matrix on $\mathbf{T}_{ij} \in SE(3)$ between each time step i and j where:

$$\mathbf{T}_{ij} = \begin{bmatrix} \mathbf{R}_{ij} & \mathbf{t}_{ij} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.4}$$

Where $\mathbf{R}_{ij} \in SO(3)$ is the rotation matrix and $\mathbf{t}_{ij} \in \mathbb{R}^3$ is the translation vector. The output of a visual odometry algorithm is a set of absolute transformation $\mathbf{C}_{0:N} = \{\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_N\}$ describing the pose of the camera at each time step where each absolute pose can be calculated by concatenating the previous absolute pose with the current relative pose:

$$\mathbf{C}_j = \mathbf{C}_i \mathbf{T}_{ij} \tag{2.5}$$

Many schemes were introduced in the state-of-the-art (see Section 2.5 "Page" 35) to provide a better way to estimate event-based visual odometry either as a stand-alone sensor or with the aid of other necessary sensors. Although an event is triggered according to change in the environment (see Equation 2.3 "Page" 32) which is mathematically and intuitively consistent with the brightness constancy constraint[5], we could not find an algorithm that basically depends on the optimization of the optical flow field for motion estimation. In the context of this thesis, we explore the opportunity to provide a visual-inertial odometry system that basically depends on the optical flow. We find the depth and pose by minimizing the errors between estimated and measured optical flow with the aid of an IMU described by the cost function:

$$F = \frac{1}{N}\sum_{i=1}^{N}\Delta_{iu} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jimu} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jba} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jbw} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{j\zeta} \tag{2.6}$$

where N is the number of estimated optical flow during optimisation span and M is the number of IMU measurements used. Δ_{iu}, Δ_{jimu} are the error $\Delta_{j\zeta}$ terms corresponding to optical flow estimation, the IMU measurements respectively. Δ_{jba} and Δ_{jbw} are the error

[5]Light does not change dynamically in our day-to-day experience and, if existing, it happens as an infinitesimal transition.

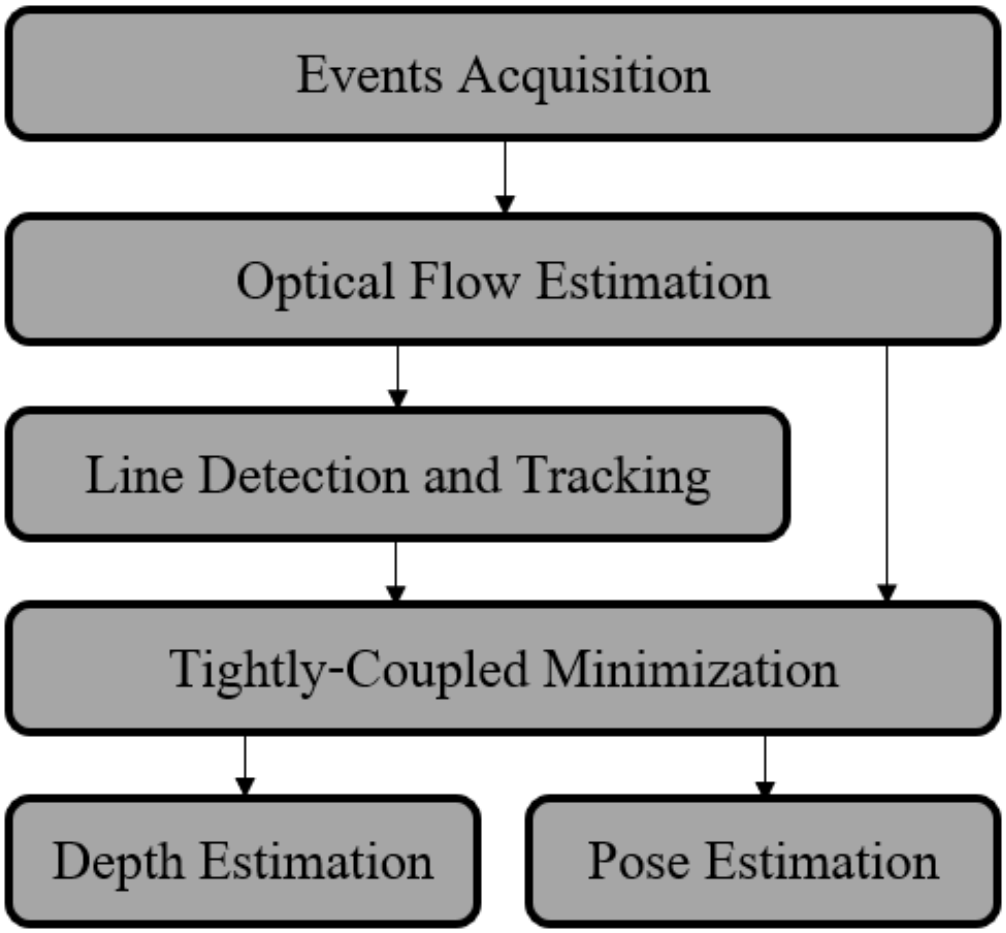


FIGURE 2.9: Blocks used in our proposed scheme

terms corresponding to the accelerometer and gyroscope bias. The error in the twist ζ is added to enhance the quality of the estimated states. Using optical flow for 6-DoF motion estimation at the absence of depth information is a computationally expensive process. For which reason, we exploit the geometric information available in indoor environments and use a Line detection and tracking algorithm to augment the known information of the environment and reduce the required computational time. Consequently, the obtained information are exploited in a tightly coupled optimisation scheme to estimate 6-DoF motion.

To make sure our scheme runs in real-time without loss of estimation quality, we improve the estimation of event-based optical flow using Principal Component analysis which improves event-based optical flow estimation and reduced the computational time. Our

scheme also provides a solution to alleviate the weight of heavy calculation needed for events optimization by implementing data reduction and applying line segments detection in structured environments. The used line detection and tracking scheme does not require search algorithms to cluster events into lines which does not affect the real time capabilities. The estimated optical flow, detected lines, and IMU measurements are used together to calculate the error terms of Equation 2.6 and estimate the 6-DoF pose and depth (see Figure 2.9).

Each of the following chapters demonstrates how we provide a solution to the assigned problem. For the necessity to have ground truth to validate the accuracy of our proposed algorithm, chapter 3 shows how we created our dataset. Chapter 4 presents the state-of-the-art for event-based optical flow and the novelty proposed in our method. Chapter 5 shows the improvement we made to provide an accurate line detection algorithm. Chapter 6 demonstrates how we exploit the IMU side-by-side with the optical flow to obtain 6-DOF poses. Finally, chapter 7 contains the conclusion to our work and perspectives for future work.

3 Benchmarking for Neuromorphic Vision Sensors

Chapter abstract

In the context of this thesis, at the moment when we started the development phase, there was a lack of convenient benchmarking datasets capable of meeting the requirement to test our developed algorithms. In this chapter we show the different types of benchmarking datasets for event-based algorithms that were introduced in the state of the art (before and after the creation of ours). We present the methodology we followed to create a benchmarking dataset capable of testing the quality of event-based optical flow as well as pose estimation algorithms. Finally, we show a validation step to make sure of the accuracy of our dataset.

3.1 Introduction

Many computer vision algorithms have been developed since the rise of vision sensors and computers capabilities. In each computer vision field, many variants of algorithms are provided to improve the quality of the expected outcomes. However, legitimate questions emerge concurrently with these technological leaps. For example, how can we trust the outcomes of each algorithm and what are the means we use to measure their quality. Also, is there an algorithm (or algorithms) that can serve as a baseline against which

we can compare our outcomes to evaluate the achieved improvements. A baseline in computer vision can be thought of as an algorithm that accomplishes a particular task and can attain acceptable performance, considered the zero line from which we measure the reached progress.

In some cases, a baseline may be a simple algorithm achieving the task with sufficiently acceptable performance in all aspects, or in other cases, it would be an algorithm providing the best-expected quality but suffering from some shortcomings hindering its usage. In order to measure the accuracy of an algorithm, researchers provide specific metrics to assess the algorithms using the "ground-truth". Ground-truth is predefined scenarios where highly accurate sensors are used to acquire the actual outcome to be estimated by an algorithm. Hence, the algorithm's quality can be deduced using the provided metrics and the ground truth.

Many benchmarking datasets are provided in computer vision. The deep neural network ImageNet [Russakovsky et al., 2015] uses millions of images of thousands of classes for image classification algorithms. Cityscapes dataset [Cordts et al., 2016] contains more than five thousand semantically segmented frames collected from about fifty different cities for the security of self-driving cars. KITTI project [Geiger et al., 2013] provides many sequences of car cruises to assess autonomous vehicles navigation algorithms. Middlebury dataset [Baker et al., 2011] covers many test cases for optical flow estimation like occlusion and non-normal flow.

The introduced perception benchmarking tools work fine for the traditional frame-based algorithms. However, with the condition that neuromorphic vision provides a new paradigm in robotic perception, the necessity of new benchmarking techniques arises. During the last ten years, many event-based benchmarking datasets have been developed to provide

the relevant tests for neuromorphic vision capabilities.

3.2 Neuromorphic Benchmarking

The process of creating benchmarking datasets capable of quantifying algorithms' performance is composed of three steps. The first step is a comprehensive understanding of the operation mode, limitations and capabilities of any sensor incorporated in the experiment. This step is required in order to conduct proper exploitation of each sensor and correct calibration. Secondly, the purpose for which the dataset is designed should be correctly described in order to choose the best conditions for the recorded sequence. Finally, the metrics against which performance will be tested need to be well defined in order to the quality of the tested algorithms to be accurately judged. Many datasets have been generated for neuromorphic vision sensors to evaluate many algorithms covering different scenarios. The following section presents some of the datasets created to serve as ground truth for different applications of event-based machine vision such as motion estimation, object classification and optical flow.

3.2.1 SLAM systems Benchmarking

There exists a number of neuromorphic datasets in the state-of-the-art to reinforce assessment of event-based SLAM related[1] algorithm. For this purpose, [Weikersdorfer et al., 2014] provided one of the first reliable datasets. A 128×128 event-based camera augmented with an RGB-D sensor was used to record this dataset (see Figure 3.1a). Five different indoor-only scenarios were recorded to obtain trusted depth information. The main challenge in these sequences was to correctly estimate the map and localisation simultaneously based on depth. However, no illumination variation or extreme velocities were tested. In addition,

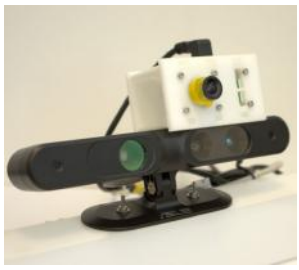
[1] SLAM, Visual/Visual-Inertial odometry, pose and depth estimation.

this dataset evaluates the quality of depth-augmented event-based cameras, which biases the judgement of event-based sensors because it depends mainly on depth information.

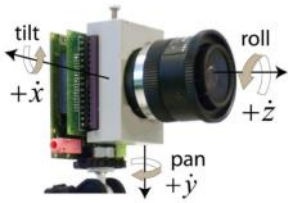
Therefore, to obtain an objective evaluation of event-based sensors, [Mueggler et al., 2017] focused on providing sequences where the main aspects for which these sensors were invented in the first place is challenged. A dynamic and Active-pixel Vision sensor (DAVIS) (see Figure 3.1b), capable of providing events and grayscale intensity frames at high framerate is incorporated with OptiTrack motion capture system. Indoor and outdoor scenarios are registered where different sequences of one, three or six degrees of freedom motion were dominant. Furthermore, varying illumination conditions are stressed to emphasize dynamic range capabilities. Although many conditions and scenarios are covered in this dataset, the dependence on motion capture system for ground-truth creation hindered obtaining outdoor ground truth. Nevertheless, only monocular hand-held motion datasets are provided.

Zhu et al. [Zhu et al., 2018] addressed the problem of stereo event-based SLAM where fine precision sensors are required in order to provide accurate measurements. A stereo rig composed of two 346×260 DAVIS sensors is used with a high precision Velodyne LiDar and Skybotix VI depth stereo camera (see Figure 3.1c). This dataset covers mainly scenarios for autonomous vehicles (drones, motorcycles or cars) but handheld motion scenarios are also covered. Indoor and outdoor sequences with varying lighting conditions are provided for drones and handheld scenarios.

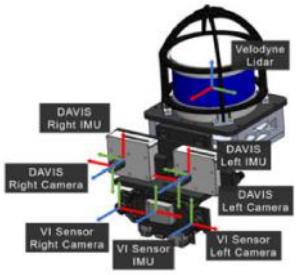
In one of the latest datasets, the issue of providing low spatial resolution is tackled by [Klenk et al., 2021]. A megapixel spatial resolution Prophesee camera is used. This dataset addresses scenarios suitable for virtual reality where all the sensors used are prepared to be either handhled or head mounted. High resolution stereo datasets are



(A) The system used in [Weikersdorfer et al., 2014] where an eDVS camera is attached to an Asus Xtion RGB-D sensor.



(B) The DVS camera used in [Mueggler et al., 2017] where the passive motion capture markers should be attached but not visible.



(C) CAD drawing of setting used in [Zhu et al., 2018]; a Velodyne LiDar, stereo Davis camera and stereo VI sensor with IMU

(D) The helmet used in [Klenk et al., 2021] with the recording kit attached; stereo ATIS camera, IDS stereo camera, an IMU and the motion capture markers.

FIGURE 3.1: different systems used to record datasets for SLAM related algorithms benchmarking.

recorded for sports activities with different motion conditions (biking, running, sliding and skateboarding). Ground truth is obtained using OptiTrack motion capture system (see Figure 3.1). The existing datasets in the state-of-the-art cover a wide range of event-based SLAM related scenarios and conditions, however there are still too many scenarios to be covered.

3.2.2 Object classification Benchmarking

The maturity gained in the field of neuromorphic vision and artificial intelligence pushed forward the exploitation of event-based cameras for object classification and semantic segmentation. However, being one of the most complex vision techniques, object classification and semantic segmentation require great attention to estimate correctly yet greater attention to create high quality benchmarking datasets. For event-based classification algorithm, [Li et al., 2017] created an event-based classification dataset based on CIFAR10 frame-based classification dataset [Krizhevsky et al., 2009]. CIFAR10 dataset consists in tens of thousands of labelled groups of images. CIFAR10 is adapted to event-based nature by recording the labelled images moving in order to capture events using an event-based camera. Many other datasets were created similarly by moving

images in front of an event-based camera. [Orchard et al., 2015] adapted MNIST and Caltech-101 datasets using a pan-tilt rig to move the camera in front of static images, [Hu et al., 2016] adapted Caltech-256 and other sequences by varying the light intensity of dataset images, [Lin et al., 2021] adapted the well-known ILSVRC2012 dataset. Although these datasets are relatively easy to record and provide a wide range of classification labels, the nature of the recorded datasets is limited to planer display or screen refresh rate, resulting in unnatural data.

Another important category of classification is pedestrians and human pose detection. Consequently, to circumvent the unnatural artifacts that may occur due to the adaptation of frame-based sequences, [Bi et al., 2019] recorded a real motion dataset. They present a large dataset of hand gestures of American sign language for the alphabet. The ASL-DVS dataset has low environmental noise present with constant illumination. [Amir et al., 2017] created a DvsGesture dataset including thousands of instances of hand and arm gestures. DvsGesture was recorded under different lighting conditions and collected from 29 different subjects. [Calabrese et al., 2019] presented the DHP19 human pose dataset to help evaluate event-based human pose algorithms. DHP19 was recorded using four cameras and a Vicon motion capture system providing the 3D position of human joints. Moreover, to improve the quality of self-driving cars and autonomous navigation, [Miao et al., 2019] designed a pedestrian dataset for human pose detection in different environments including corridors, streets and squares also an indoor dataset for fall detection.

Furthermore, to alleviate the challenge of employing event-based cameras in self-driving cars, [Binas et al., 2017] introduced the Davis Driving Dataset DDD17. This dataset does not only provide annotated recordings in different cities at different day times but also records the vehicle control and diagnostic data. An ATIS sensor was used in [Sironi et al., 2018] to record large real-world datasets. The grayscale images acquired from the sensor are used to extract bounding boxes around cars and street objects using

state-of-the-art object detector. This dataset provided a cars and non-cars classes. The same sensor was used in [de Tournemire et al., 2020] to provide more labeled objects in varying lighting and weather conditions.

3.2.3 Optical Flow Benchmarking

Optical flow estimation is one of the essential well-established vision algorithms. It acquired due attention for neuromorphic vision in the last decade. Similarly to SLAM related benchmarking datasets, optical flow estimation datasets require high precision sensors and accurate calibration. One of the earliest benchmarking optical flow datasets is provided by Rueckauer and Delbruck [Rueckauer and Delbruck, 2016]. They provided a synthetic sequence of moving lines and squares. Additionally, real sequences were recorded to evaluate rotational, translational and sinusoidally varying motion. The motion provided in this data set was constrained to only in-plane motion and used an IMU to obtain the ground truth optical flow. Accordingly, to counteract the effect of favoring only normal flow in [Rueckauer and Delbruck, 2016], [Almatrafi and Hirakawa, 2019] proposed another dataset where texture was not perpendicular to motion direction.

Using only an IMU to acquire the ground truth limits the motion type and is unable to capture moving objects' optical flow. [Barranco et al., 2016] used an IMU to acquire ground truth by using a robot mounted pan-tilt system where neuromorphic events are fused with depth maps from an RGB-D sensor to obtain high precision ground truth. The provided dataset does not only provide optical flow but also image frames, depth frames and 3D camera motion. [Mitrokhin et al., 2019] used a Vicon system to capture the camera and objects' motion. As a result, a relatively long dataset of fast-moving objects is recorded where, besides optical flow ground truth, depth and 3D motion is alo provided.

In the context of this PhD thesis, our area of interest is visual-inertial odometry, based on which we were motivated to develop our own dataset that can provide us with 6-DOF pose and optical flow ground truth. In the following section, we present our benchmarking dataset creation procedure.

3.3 Dataset Creation

As mentioned in the previous section, the main purpose of the dataset we present here is to provide an approach to assess event-based visual-odometry algorithms based on optical flow. Therefore, we used an ATIS sensor to record the data assisted with a VICON motion capture system for high precision ground truth acquisition. Our dataset presents different recorded sequences of scenarios featuring handheld camera in-plane rotation, translation and free motion with varying velocities (see Figure 3.2). The camera tracks a checkerboard for optical flow estimation of the scene (see Figure 3.4). The VICON system provides a 3D pose of the camera and the board with 200 Hz frequency. In order to maintain high-quality events, we start by conditioning the events recorded, and then we calibrate the used sensors. Hence, The ground truth is created using the VICON system. Finally, we validate the accuracy of the created ground truth. In the following sections, we show how each step is done.

3.3.1 Events Conditioning

Due to the noisy nature and sensitivity of DVS cameras, it is essential to get rid of uncorrelated events created by background activity or any other source like transistor switch leakage [Lichtsteiner et al., 2008]. Delbruck [Delbruck, 2008] employs an activity filter to reject incorrectly created events. The activity filter takes only one parameter (T the "support time"), which is the maximum time difference allowed between the current event and events created previously in the same neighbourhood. The activity filter uses

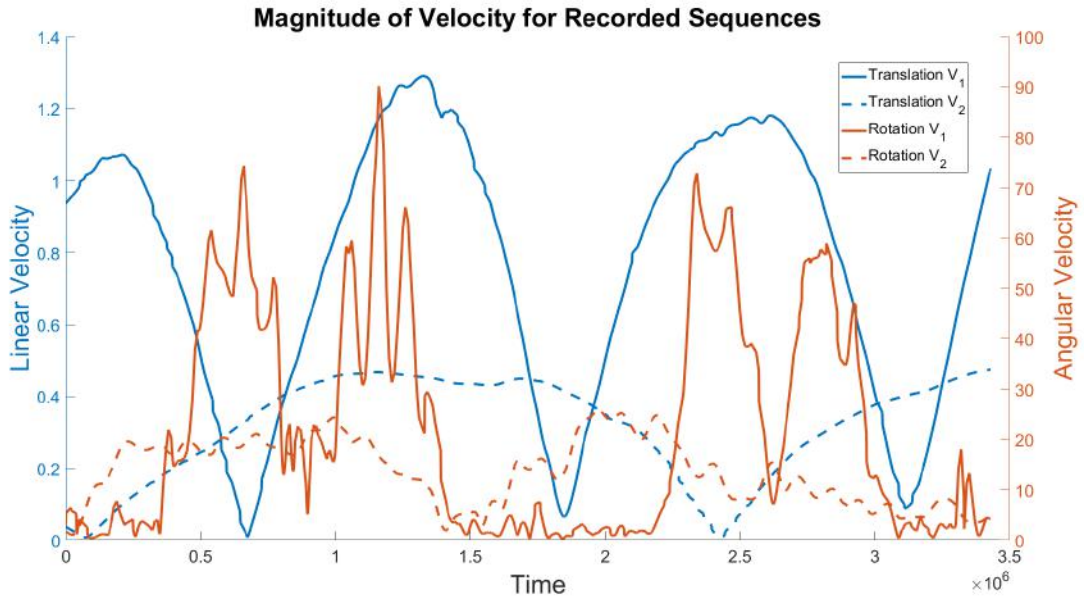


FIGURE 3.2: The magnitude of the velocities of the recorded sequences. The linear velocity is measured in [m/s] and the angular velocity in [deg/s]

the active events surface, which is a buffer saving the timestamp of the last existing event at a specific pixel position. The support time T decides whether an event can be passed as a true event or as noise. First, the event timestamp is stored for the pixel 8-neighbourhood. Second, a check between the timestamp stored in the event's location and the event's timestamp is performed: if an event occurred nearby the current event (within the support time T), the new event is passed or discarded otherwise.

Constant support time T implies the rejection or inclusion of events within a specific interval of environmental motion dynamics. We modified this filter to make it more robust to noise and adaptive to the dynamics of the environments (see Figure 3.3). Instead of having a constant time support T, we introduced an adaptive parameter T_f that depends on the frequency of created events since they are related to the dynamics of the environment. T_f is estimated using linear interpolation between the minimum and maximum possible support time T_{min} and T_{max}. The interpolation is done to find T_f based on the inverse of events' frequency. Event-based cameras can provide a wide range of frequencies depending on changes in the environment. Hence we use the inverse-log

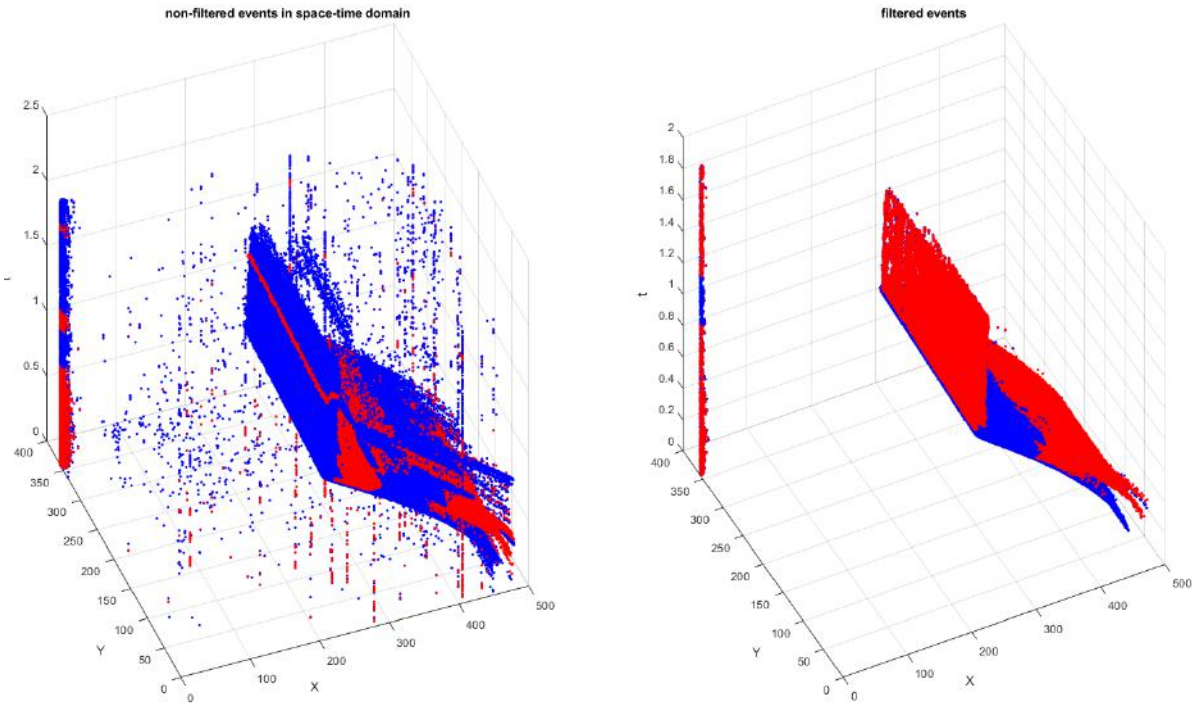


FIGURE 3.3: Left: a 3D neighborhood of positive and negative events created by a rod moving at varying velocity. Right: the created events filtered using the adaptive activity filter.

function for its decaying property, and we calculate T_f based on the following equations:

$$\alpha = \frac{k}{\log f_e} \tag{3.1}$$

$$T_f = \frac{T_{max} - T_{min}}{\alpha_{max} - \alpha_{min}}(\alpha - \alpha_{min}) + T_{min} \tag{3.2}$$

where we tune k to get a better logarithmic curve that would give the best value of T_f for different frequencies. α_{min} and α_{max} are the values of α which correspond to the lowest and highest values of events frequency f_e Where the frequency is calculated using the amount of acquired events previously in a sliding window. The recorded events are passed through this filter to maintain a higher quality of events acquisition (see Figure 3.3).

3.3.2 System Calibration

On one hand, the VICON system can be intrinsically calibrated using the provided software, and it provides 3D poses of the point of origin of the board along with the camera pose with 200 Hz frequency. On the other hand, it can provide the camera pose in its own coordinates system. Accordingly, we need to find the transformation between the camera and the VICON system. We do not use any SLAM or pose estimation algorithms to get the camera/VICON transformation. We exploit the gyroscope readings of the camera embedded IMU of the camera as a medium to find the transformation. This process requires accurate intrinsic and extrinsic calibration. The camera/VICON and camera/board transformations are found by the following equation:

$$T_{vic}^{c} = T_{IMU}^{c} \times T_{vic}^{IMU} \tag{3.3}$$

$$T_{B}^{c} = T_{IMU}^{c} \times T_{vic}^{IMU} \times T_{B}^{vic} \tag{3.4}$$

where T_a^b represents the rigid transformation from frame of reference a to b. The super/subscripts c, vic, IMU and B represent the camera, VICON system, IMU and board frames of reference respectively. The rigid transformation T_B^{vic} can be calculated easily using the data acquired from the VICON system. The following subsections show how the other rigid transformations are obtained.

Intrinsic Calibration

The intrinsic calibration is needed for the used camera and its embedded IMU. Intrinsic calibration means finding the intrinsic parameters of each sensors that satisfy the mathematical model of each sensor.

- **IMU Intrinsic Calibration:**

  The IMU parameters to be estimated are divided into two categories, deterministic and random. Deterministic parameters are scale factor, misalignment error and bias

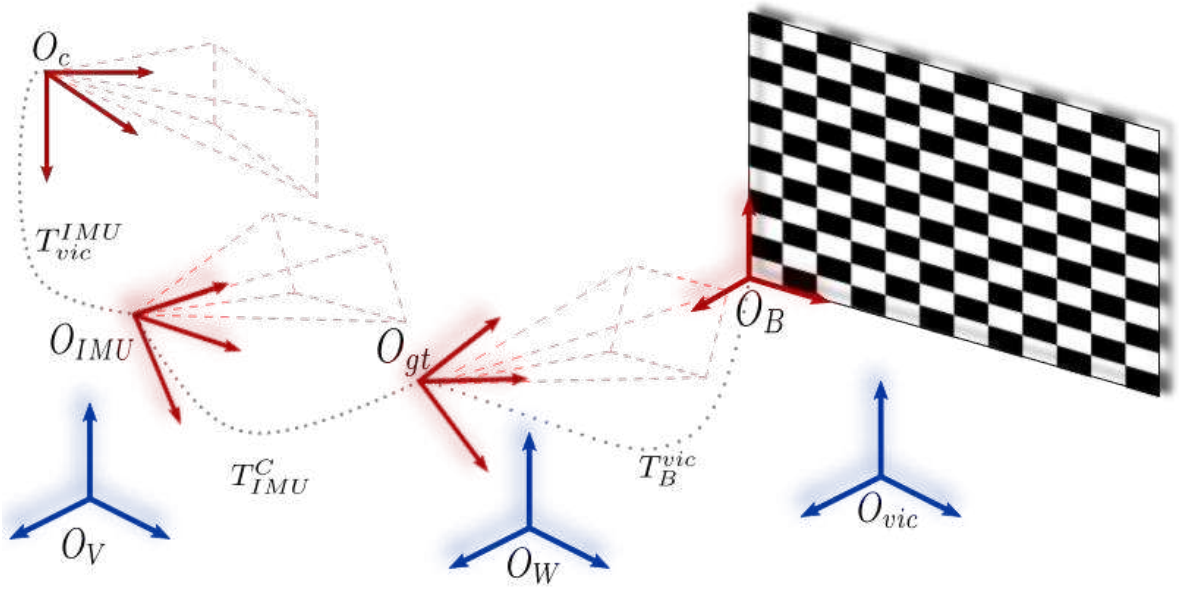


FIGURE 3.4: The coordinates system used in our setting

offsets. Random errors are the bias residuals and white noise added to the signal so that the IMU can be modeled as follows:

$$\omega_{IMU} = [I + M_g]\omega + b_g + \delta b_g + \epsilon_g \tag{3.5}$$

$$a_{IMU} = [I + M_a]a + b_a + \delta b_a + \epsilon_a \tag{3.6}$$

where ω_{IMU} and a_{IMU} are 3D vectors of the rotational velocity and linear acceleration obtained by the IMU, M_g and M_a are the matrices containing misalignment errors of the gyroscope and accelerometer respectively. b_g and b_a are the offset biases of the gyroscope and accelerometer. δb_g and δb_a are the bias residual of the gyroscope and accelerometer that changes with very low frequency, and ϵ_g and ϵ_a are white noises of the gyroscope and accelerometer.

Using the six-position calibration [El-Diasty and Pagiatakis, 2010], we obtain the deterministic parameters of the IMU except for the matrix M_g because of the inability to have an accurate known excitation source for the gyroscope, which still can be suppressed in a fusion scheme. We used Allan variance modelling

[El-Sheimy et al., 2007] in order to estimate the parameters controlling the random IMU parameters.

- **Camera Intrinsic Calibration:**

  The event-based camera adopts a pinhole model; consequently, intrinsic calibration is nothing different from frame-based cameras except for how data is acquired to obtain intrinsic parameters. Flashing patterns are used to adjust the sharpness and focus of the lens. Consequently, a varying lighting intensity flashing checkerboard on a screen is used to record events. Events are stacked and integrated to create frames to be used for calibration. The resulting images are used as input for the embedded MATLAB camera calibrator toolbox to estimate the camera's intrinsic parameters. The estimated parameters are the camera focal lengths (f_u, f_v), the camera principal point (c_u, c_v) and the radial and tangential distortion coefficients $(k_1, k_2, k_3, p_1, p_2)$.

Extrinsic Calibration

No extrinsic calibration is required between the camera and the checkerboard since the VICON system measures their 3D pose expressed in its own coordinates system. Therefore, only calibration between the camera 3D pose and the motion capture system is needed. The extrinsic calibration process is divided into temporal synchronization and spatial alignment between the signals to be calibrated to ensure the outputs from different systems are perfectly adjusted.

- **VICON/IMU Calibration:**

  The problem of two different frames of reference correspondence that observe the same states are widely known as Wahba's problem [Wahba, 1965], where it is defined, for given two 3D sets of measurements $\{X_i\}_{i=0}^{N-1}$ and $\{Y_i\}_{i=0}^{N-1}$, as the least square

minimization problem and described in the following equation:

$$\underset{\{s,R,t\}}{\operatorname{argmin}} \sum_{i=0}^{N-1} |Y_i - s(\mathbf{R}X_i + \mathbf{t})|^2 \tag{3.7}$$

where s, $\mathbf{R}$ and $\mathbf{t}$ are scale, rotation and translation between the two frames respectively. Solving this minimization problem is based on two assumptions: noise is suppressed in both measurements, and they are temporally synchronized. Since the IMU measurements suffer from high noise due to double integration, we chose to use the angles to be injected in the minimization problem. The choice is made for two reasons; only single integration is required for angles estimation. Also, the VICON system markers are placed near the camera's body frame, which suggests that the difference in $\mathbf{t}$ can be neglected compared to the distance between the camera and the board. We used an error state Kalman filter [Sola, 2017] to obtain accurate IMU angular estimation. The VICON system and the IMU are neither spatially aligned nor temporally synchronized. Therefore, we adopt an iterative scheme to find both the spatial transformation and temporal shift. We solve for an initial transformation between the two frames using Horn's reformulation of Wahba's problem [Horn, 1987]:

$$\underset{q}{\operatorname{argmin}} \sum_{i=0}^{N-1} (q \times \boldsymbol{\Theta}_{\boldsymbol{IMU}} \times q^*)\boldsymbol{\Theta}_{\boldsymbol{vic}} \tag{3.8}$$

where $\boldsymbol{\Theta}_{\boldsymbol{IMU}}$ and $\boldsymbol{\Theta}_{\boldsymbol{vic}}$ are the angles measured in IMU and VICON frames, and

sequence	$\phi[^\circ]$	$\theta[^\circ]$	$\psi[^\circ]$
`rotate_low`	0.7893	0.5988	1.5579
`rotate_high`	1.2885	0.9427	0.4945
`translate_low`	1.2347	1.5135	0.4813
`translate_high`	1.4031	1.6943	0.4838
`free_motion`	0.3923	0.5544	0.3923

TABLE 3.1: VICON / IMU mean angles differnce

Algorithm 1 IMU-VICON Calibration

Input: $\{\mathbf{\Theta}_{IMU}\}_{i=0}^{N-1}$, $\{\mathbf{\Theta}_{vic}\}_{i=0}^{N-1}$
Output: q , Δt
Initialize: $check = true$, $itr = 1$, $\epsilon = small\ value$
while check = true **do**
  $\mu_{IMU} = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{\Theta}_{IMU_i}$
  $\mu_{vic} = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{\Theta}_{vic_i}$
  $\mathbf{\Theta}_{IMU} = \mathbf{\Theta}_{IMU} - \mu_{IMU}$
  $\mathbf{\Theta}_{vic} = \mathbf{\Theta}_{vic} - \mu_{vic}$
  Calculate: $S_{mn} = \sum_{i=0}^{N-1}\mathbf{\Theta}_{IMU_m}\mathbf{\Theta}_{vic_n}$ ▷ 9 values
  Construct: $N_{4\times4}$ matrix ▷ see [Horn, 1987]
  Solve: $\{\lambda, V\} = eig(N)$
  $q_{itr} = V_{\lambda_1}$ ▷ q is the vector corresponds to maximum λ
  project: $\mathbf{\Theta}_{MU} = R_q\mathbf{\Theta}_{IMU}$
  Solve: $\Delta t_{itr} = xcorr(\mathbf{\Theta}_{IMU}, \mathbf{\Theta}_{vic})$ ▷ $xcorr$ is cross correlation matlab function
  Shift: $\mathbf{\Theta}_{IMU} = \mathbf{\Theta}_{IMU}(\Delta t : end)$
  $itr++$
  if $\frac{1}{N}\sum_{0}^{N-1}|\mathbf{\Theta}_{IMU_i} - \mathbf{\Theta}_{vic_i}| < \epsilon$ **then**
    check = false
  end if
end while
$q = \prod_{n=1}^{itr} q_n$
$\Delta t = \sum_{n=1}^{itr}\Delta t_n$

q is a unit quaternion that represents a rigid body transformation. We project the IMU readings in the VICON frame to apply cross-correlation between the two signals to find Δt that represents the temporal shift between the two signals. This process is repeated iteratively until the difference between the estimated calibration parameters reaches convergence (see Algorithm 1). It is noted that after two iterations, convergence is fulfilled. To make sure that every sequence is correctly correlated, we calculated the mean absolute difference between the VICON angles and IMU angles after being transformed in the VICON frame, results are shown in Figure 3.5. The paths of the different recorded sequences are shown in Figure 3.6 where there are two different sequences with varying dominant translational and rotational motion and another sequence of random motion. Table 3.1 shows that

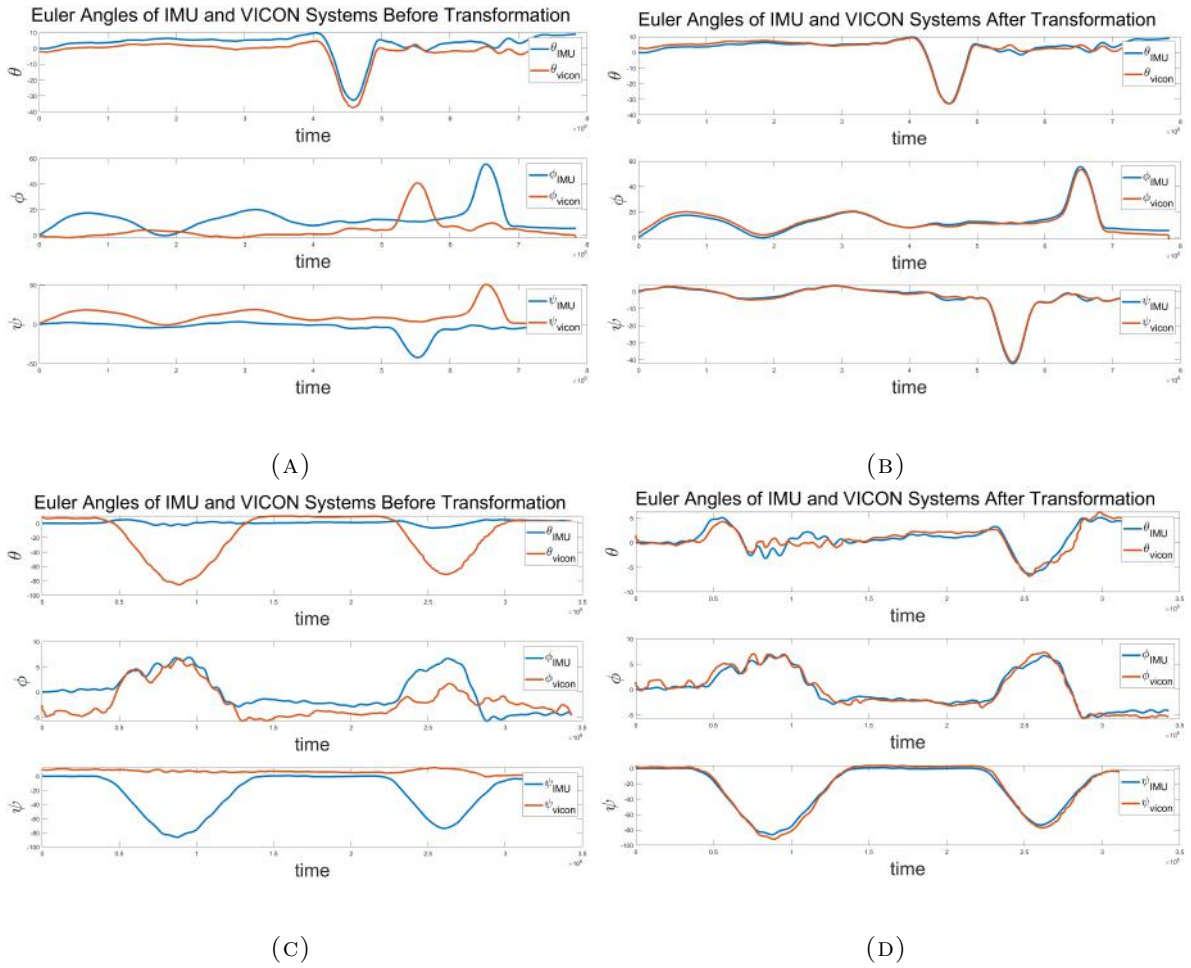


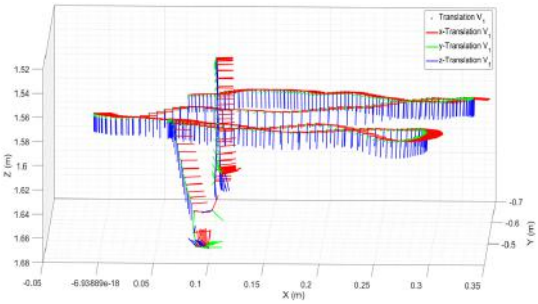
FIGURE 3.5: Results of the calibration between the IMU frame and the camera frame. Figures (A) and (C) represent the orientation of the camera and the IMU before the transformation and figures (B) and (D) show the alignment between them after transformation of two sequences of our recorded dataset.

the mean error between the angles of the IMU and the VICON system is acceptable and that the two signals are aligned.

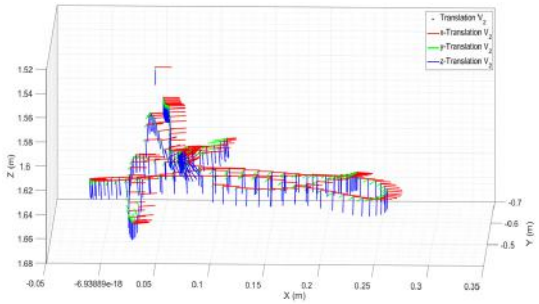
- **IMU/Camera calibration:**

  After getting the spatio-temporal calibration between the VICON and IMU we need to do the same between the IMU and the Camera. We used Kalibr toolbox [Furgale et al., 2014] to get the spatial transformation between the camera and IMU. Since Kalibr provides a temporal difference only of the provided data-set used for calibration (which was totally different that the data-set used for our comparison), we use the concept of cross correlation between the IMU absolute

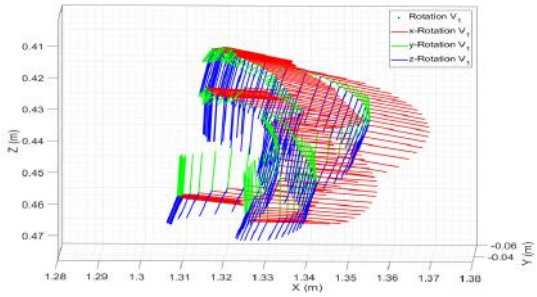
rotational velocity and the events frequency in order to find the best time shift. The choice of events frequency to be correlated with absolute rotational velocity is similar to the temporal synchronization used in [Censi and Scaramuzza, 2014] since with a higher velocity more events would be triggered per second, so the best synchronization will correspond to matching these two signals together.



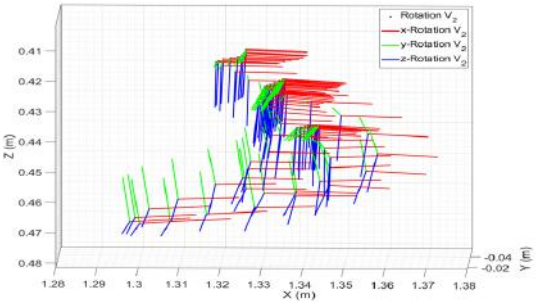
(A) Low velocity translation scenario.



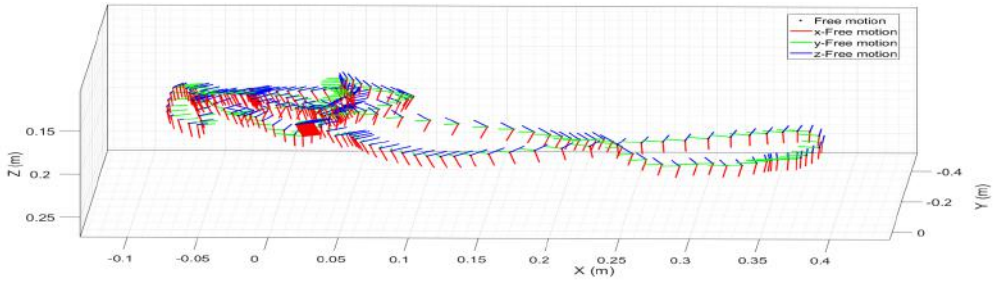
(B) High velocity translation scenario.



(C) Low velocity translation scenario.



(D) high velocity translation scenario.



(E) Free motion scenario

FIGURE 3.6: The different recorded paths in our dataset with different velocities

3.3.3 Ground Truth Creation

Based on the scenario explained in Section 3.3, the ground truth of the camera and checkerboard 3D poses is obtained using output of the VICON system and expressed in the camera frame of reference using Equations 3.3 and 3.4. Optical flow ground truth is obtained using the relative pose between the checkerboard and the camera and the 3D/2D projection model. Using the initial pose of the checkerboard (being planar), we create points at the edges of the checkerboard to replicate the initial position of the checkerboard. The created points are projected to the camera frame to validate initially if they coincide with the events triggered by the camera. The optical flow will be approximated as 2D motion projected from the checkerboard pose expressed in the camera as proposed in [Heeger and Jepson, 1992]. Having $\mathbf{V}$ and $\mathbf{W}$ as the relative rotational and translational velocity between the checkerboard and the camera, the optical flow can be obtained from the equation:

$$\mathbf{U}(x,y) = \frac{1}{Z}\mathbf{A}(x,y)\mathbf{V} + \mathbf{B}(x,y)\mathbf{W} \tag{3.9}$$

where x and y are the pixel coordinates and $\mathbf{A}$ and $\mathbf{B}$ are

$$\mathbf{A}(x,y) = \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \tag{3.10}$$

$$\mathbf{B}(x,y) = \begin{bmatrix} (xy)/f & -(f+x^2/f) & y \\ (f+x^2/f) & -(xy)/f & -x \end{bmatrix} \tag{3.11}$$

The created optical flow ground truth is generated at the frequency of the VICON system (200 Hz) and is used to find the optical flow of all the events. Interpolation between each two consecutive optical flow frames is used to find each event's optical flow according to the equation:

$$\mathbf{OF}(x,y,t) = \frac{(t-t_{i-1})\mathbf{U}(x_{i-1},y_{i-1}) + (t_i - t)\mathbf{U}(x_i,y_i)}{t_i - t_{i-1}} \tag{3.12}$$

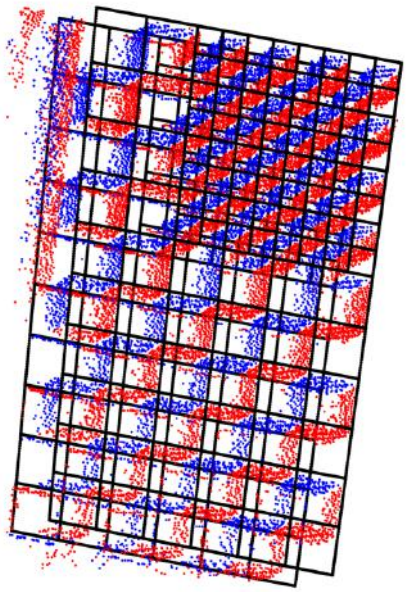


FIGURE 3.7: A sample of the events created between two consecutive synthetically created checkerboards (grey) with positive events in red and negative events in blue.

where $\mathbf{OF}(x, y, t)$ is the ground truth optical flow of an event created at (x, y, t) between frames $i-1$ and i, $\mathbf{U}(x_{i-1}, y_{i-1})$ and $\mathbf{U}(x_i, y_i)$ are the ground truth optical flow of the nearest pixels to the event position at time t_{i-1} and t_i respectively. Only events contained in the Region Of Interest ROI of the checkerboard are cropped and assigned an optical flow to ensure correct optical flow estimation (see Figure 3.7).

3.3.4 Ground Truth Validation

The process of calibration used to align the VICON system with the camera depended only on the gyroscope readings to estimate the transformation based on the assumption that distances between the camera, the IMU and the VICON system markers are negligible compared to the distance of objects in the scene. This assumption requires a validation of the resulting ground truth to make sure that no biases or deviations affect it. We randomly selected samples of two consecutive frames f_1 and f_2 and used the obtained optical flow to warp the events created between these frames. The events' final position

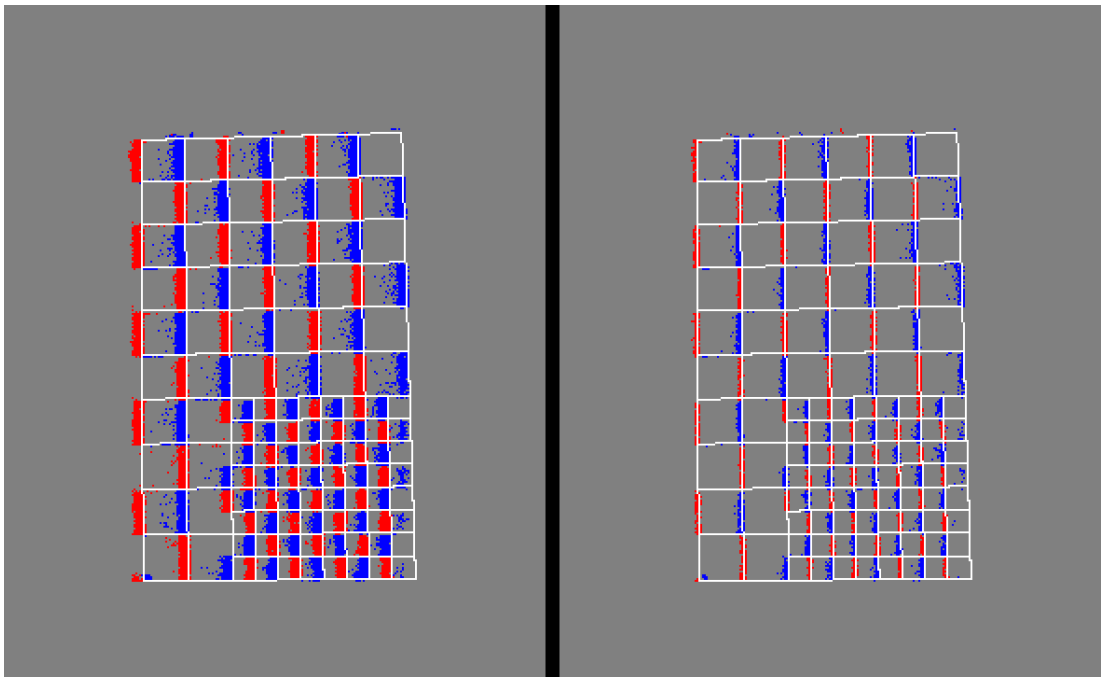


FIGURE 3.8: (left) the events and the synthetic created checkerboard before projecting the events using the optical flow ground truth, (right) the projected on and off events in the camera frame are sufficiently aligned with the created checkerboard.

warped in the frame f_2 will be:

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + (t_f - t_i)\mathbf{OF}(x_i, y_i, t_i) \tag{3.13}$$

where (x_i, y_i) and (x_f, y_f) are the event's position before and after warping, t_f is the f_2 frame timestamp and t_i is the event's timestamp. After warping, we check the error between the new event's position and the position of the nearest pixel of the synthetically created checkerboard. The warped events were aligned with the checkerboard and the error did not exceed 2 pixels. (see Figure 3.8).

3.4 Conclusion

Developing an infallible algorithm would never be acknowledged without being validated against credible dataset with ground truth. We presented a clear and straightforward

method to obtain optical flow ground truth for event-based cameras using a VICON system. This benchmarking dataset represents the first step in the development of our system to tackle the problem of visual-inertial odometry and allows for comparison with other stat-of-the-art algorithms. This dataset will be used in the following chapters side by side with some other datasets created afterwards to fulfil a wider and more diverse set of scenarios to assess our proposed algorithms.

4 Neuromorphic Optical Flow

Chapter abstract

Based on the coherence between the neuromorphic vision sensors operation mode and the brightness constancy condition of optical flow, we chose optical flow as the core algorithm for our visual-inertial pdometry algorithm. In this chapter we show the advancement achieved concerning event-based optical flow and present a novel method to estimate optical flow in a more straightforward and faster way. The introduced method uses Principal Component Analysis (PCA) to estimate the optical flow for its accuracy and low computational time. A comprehensive evaluation of our algorithm is conducted to show the improvements achieved with respect to the state-of-the art.

4.1 Introduction

The photoreceptor cells in the human eye, namely rods and cones, are responsible for converting light photons into an electrical signal to be processed by the brain whenever a change is detected. Rods detect the black and white gradients while cones distinguish colors and fine details with a rods/cones ratio up to 60:1, for which reason we can identify motion and blobs of moving figures in low light conditions [Gibson, 1950]. In computer vision, optical flow estimation follows the same methodology based on

detecting change, where camera frames are fed to algorithms governed by the brightness constancy change [Horn and Schunck, 1981]. Whenever a change in brightness is detected between frames, the optical flow can be estimated. Optical flow estimation generally requires pixel-wise operations for an accurate estimate, which causes heavy computations and resources. Although, flow plays a vital role in estimating ego-motion [Longuet-Higgins and Prazdny, 1980], depth [Lepisk, 2005] and foreground-background segmentation [Chen et al., 2014]. For faster computation, [Lucas and Kanade, 1981] proposed an iterative sparse method that works on image portions where motion is of high interest.

As discussed earlier, standard frame-based cameras do not adopt the best model to imitate the biological vision performance and are not optimal for machine vision algorithms involving motion. Standard cameras provide full-frame images at a fixed frequency resulting in data redundancy, low dynamic range, transmission latency and motion blur. Neuromorphic vision sensors [Lichtsteiner et al., 2008, Brandli et al., 2014, Posch et al., 2010a], as the name suggests, mimic the biological processes between the retina and the brain. The nature of neuromorphic sensors allows temporal resolution increase to microseconds scale and expands the dynamic range. Consequently, motion blur in standard frame-based cameras is omitted while consuming low power. In addition, the way event-based cameras operate, being consistent with optical flow, encouraged many researchers to find new solutions adapted to their neuromorphic nature [Rueckauer and Delbruck, 2016, Low et al., 2020, Gallego et al., 2018].

In this regard, neuromorphic vision sensors offer a better alternative to standard cameras in computer vision [Gallego et al., 2022]. Instead of transmitting a sequence of full-frame images at a fixed frequency, the pixels of these neuromorphic chips operate independently and asynchronously. The events generated by neuromorphic vision sensors carry, intrinsically, optical flow information because they respond only to luminosity changes.

Since the time intervals between events at the same location can reach microsecond scale, neuromorphic sensors provide a sufficiently large amount of data to be processed in short time intervals. Consequently, the large amount of sparse data provided requires algorithms that run fast while being robust to noise. In this chapter, we employ Principal Component Analysis (PCA) [Pearson, 1901], an extensively used technique as a linear dimensionality reduction algorithm, to estimate optical flow. PCA estimates the principal axes, which are rapidly calculated for lower dimensions data (three-dimensional in our case). Moreover, we apply different regularization techniques to ensure a robust estimate to refine the estimated optical flow.

4.2 Optical flow Event-Based Approaches

Many algorithms using event-based optical flow have been developed as a solution for many applications during the past decade. These algorithms can be classified into three groups. The first group does not involve heavy optimization and either uses data correlation methods or adapts frame-based methods for optical flow estimation. The second group depends on minimizing – or maximizing – an optimization function optical flow estimation or scene reconstruction based on optical flow. The third group uses neural networks to predict event-based optical flow.

4.2.1 Non-Optimization-Based Methods

In one of the first attempts to tackle the challenge of event-driven optical flow estimation, [Delbrück, 2008] considers it as a problem of data association where events are correlated using their relative timestamps in a spatio-temporal neighborhood. This method can effectively augment the amount of information induced by each event and provides a highly discretized orientation and magnitude of optical flow but can not be considered a suitable or accurate optical flow estimation.

Since each event embeds in itself brightness intensity information, [Benosman et al., 2012] proposed a method where events in a spatio-temporal neighborhood are counted and considered as intensity-equivalent values. The events' count is inserted in an adapted Lucas-Kanade event-based scheme [Lucas and Kanade, 1981] to estimate the optical flow using the least-square method. This method can be considered the first event-based optical flow algorithm to achieve acceptable accuracy. However, the intensity-equivalent values are not the actual intensity values which means that the optical flow is not accurately represented.

Benefiting from the time being represented as a monotonically increasing function, generated events in a slight temporal shift will form approximately planar surfaces (see Figure 4.3). Based on this time property, [Benosman et al., 2013] proposes a local plane fitting method where an event is mapped as a time function t in the spatial 2D domain (x, y) and clustered events in a spatio-temporal neighborhood can be used to fit a plane. The x and y components of the estimated plane normal vector represent the optical flow estimate.

Instead of estimating the optical flow in the time-domain, [Barranco et al., 2015] estimate the optical flow using a Gabor filter on spatio-temporal neighboring events to obtain the optical flow at high contrast contours accurately. Although this method shows promising results, the use of frequency-domain filters can not provide real-time applicability for event-based cameras.

[Almatrafi et al., 2020] proposed a novel concept named the "distance surface" reformulating optical flow in a way more suited to event-based cameras. In addition, they provided a new formulation that satisfies the brightness constancy condition. Although sufficient improvements in accuracy were attained compared to non-optimization methods, no discussion about computational time or real-time applicability was disclosed in their

work.

Most of the non-optimisation based can perform in real-time up to a certain events' frequency. Although, the accuracy of the estimated optical flow is not sufficient to be used for 6-DoF motion estimation with acceptable quality.

4.2.2 Optimisation-Based Methods

Instead of using least square estimation or other straightforward approaches, complex optimization schemes are employed to obtain event-based optical flow. In order to refine the optical flow, [Mueggler et al., 2015] proposed to estimate the lifetime of events by employing a RANSAC scheme to refine the plane fitting output. RANSAC output is fed to an optimization scheme that minimizes estimated lifetime error and consequently improves optical flow estimation.

A contrast maximization optimization scheme is presented in [Stoffregen and Kleeman, 2019] where the optical flow is the motion vector that maximizes the variance of events in a spatio-temporal neighborhood. This method finds the optical flow of blocks of neighboring spatio-temporal events, which reduces the accuracy of each event individually because it finds the optimal optical flow vector for all the events contained in the block .

[Bardow et al., 2016] introduce a joint estimation optimization scheme to simultaneously obtain the optical flow field and the photometric grayscale intensity of all pixels to tackle the sparsity of event-based data. Although a GPU is used, the optimization method leads to a dense estimate of the optical flow at the price of high computational time.

With the help of grayscale blurred images provided by neuromorphic vision sensors,

[Pan et al., 2020] introduce another dense optical flow estimation using variational optimization. They prove that embedding the photometric information (even if blurred) in the optimization scheme improves the quality of optical flow estimation. This method requires strong texture in order to estimate an accurate optical flow.

Using a cost function for optimisation-based algorithms, improved the quality of the estimated optical flow to be used in motion estimation. Although, these methods require high computational power which is critical for event-based cameras since they provide a large amount of data with a varying frequencies. The asynchronous nature of event-based cameras suggests that an algorithm can perform in real-time in some scenarios and fails in other ones. For which reason, computational time should be kept as low as possible to guarantee real-time applicability in all cases.

4.2.3 Neural-Based methods

One of the first attempts to accommodate neural networks for event-based vision is presented in [Orchard et al., 2013]. For event-based sensors, it is found to be beneficial to use Spiking Neural Networks (SNN) instead of Convolutional Neural Networks. [Ghosh-Dastidar and Adeli, 2009] adopted an SNN to estimate a discretized optical flow orientation and amplitude. Although poor optical flow is estimated, this method can be considered a widely acceptable introduction of SNN to event-based vision.

Instead of obtaining a simple discretized flow field, a better adaptation of SNN is introduced in [Zhu et al., 2019] where three-channel images of positive and negative event count, as well as timestamps, are used to estimate dense optical flow, depth and ego-motion. They pass the images created by stacking the events' information to a neural network to predict the optical flow. The predicted optical flow is used to aid another

network in estimating the depth and ego-motion.

Spike-FlowNet [Lee et al., 2020] estimates the optical flow using events provided between two image frames. Although sufficiently accurate optical flow is estimated, this method estimates optical flow at the frequency of acquired image frames. No absolute computational power requirements are disclosed regarding their work.

Although, the usage of Neural networks provides, as expected, the best performance for optical flow estimation, neural networks require a meticulous training datasets so that it can cover all the possible scenarios without being overfit to the training set itself. The uncertainty about the outcome of neural networks, since it is not model-based, makes it difficult to rely on neural-based algorithms in large-scale scenarios. Moreover, neural networks require, in almost all cases, heavy computations and maybe GPU's, which hinders real-time applicability.

The work present in the state-of-the-art is always a trade-off between better accuracy and faster computation. In the context of our thesis, we are concerned with the possibility of providing acceptable accuracy in different scenarios while maintaining low computational power so that it can be integrated into other schemes without hindering real-time applicability. In the following section, we introduce how optical flow is modeled for event-based cameras then we present our PCA optical flow.

4.3 Event-Based Optical Flow

The brightness constancy condition[Horn and Schunck, 1981] indicates that under constant brightness, no change will be observed in the spatio-temporal domain (x, y, t) even

if it undergoes changes so that:

$$\mathbf{I}(x + \Delta x, y + \Delta y, t + \Delta t) = \mathbf{I}(x, y, t) \tag{4.1}$$

where $\mathbf{I}(x, y, t)$ is the brightness intensity in the 2D spatio-temporal domain (x, y, t) and Δx, Δy and Δt are the perturbations in x, y and t respectively. Approximating the right-hand side using the famous Taylor expansion [Taylor, 1717] leads to:

$$\mathbf{I}(x, y, t) + \frac{\partial}{\partial x}\mathbf{I}(x, y, t)\frac{\Delta x}{\Delta t} + \frac{\partial}{\partial y}\mathbf{I}(x, y, t)\frac{\Delta y}{\Delta t} + \frac{\partial}{\partial t}\mathbf{I}(x, y, t) + \mathcal{O}(\Delta t^2) = \mathbf{I}(x, y, t) \tag{4.2}$$

$$\nabla\mathbf{I}(x, y, t)\mathbf{V}(x, y, t) + \mathbf{I}_t(x, y, t) \approx 0 \tag{4.3}$$

where $\nabla\mathbf{I}(x, y, t)$ is the spatial first order partial derivative of the intensity function known as the image gradient, $\mathbf{I}_t(x, y, t)$ is the temporal first order partial derivative and $\mathbf{V}(x, y, t)$ is the optical flow field of each pixel. For event-based cameras, pixels responds independently and asynchronously to the change in the log of brightness intensity. The triggered events are encoded as a tuple of $\langle x, y, t, p\rangle$ created whenever the change exceeds a certain threshold δ_t [Gallego et al., 2015]:

$$\log(\mathbf{I}(x_i, y_i, t_i)) - \log(\mathbf{I}(x_i, y_i, t_i - \Delta t)) = p\delta_t \tag{4.4}$$

where the polarity p is the sign of the change δt that belongs to $\{1, -1\}$ and Δt is the time difference since the last event triggered by the same pixel. Applying Taylor's expansion to the log difference leads to the integration of the brightness constancy condition:

$$\Delta\mathbf{L}(x_i, y_i, t_i) = \frac{\partial L}{\partial t}(x_i, y_i, t_i)\Delta t + \mathcal{O}(\Delta t^2) \tag{4.5}$$

$$\Delta\mathbf{L}(x_i, y_i, t_i) \approx \left(\frac{\partial}{\partial x}\mathbf{L}(x_i, y_i, t_i)\frac{\Delta x}{\Delta t} + \frac{\partial}{\partial y}\mathbf{L}(x_i, y_i, t_i)\frac{\Delta y}{\Delta t} + \frac{\partial}{\partial t}\mathbf{L}(x_i, y_i, t_i)\right)\Delta t \tag{4.6}$$

Equation 4.6 shows that triggering an event encompasses in itself optical flow information integrated over time. Unlike frame-based cameras, the high temporal resolution (reaching microsecond level) of event-based cameras provides a more accurate and stable representation of Taylor's expansion. Based on this hypothesis, using optical flow estimation with event-based cameras for visual odometry is a good opportunity for its adaptation to event's creation model. The main challenges with event-based optical flow are: First, finding the best way for data association to solve for optical flow since only one event can not be used to find optical flow. Second, making sure that the chosen method operates as fast as possible to cope with the amount of data created and to achieve real-time applicability.

4.4 PCA Optical Flow

The Principal Component Analysis (PCA) [Pearson, 1901] was first introduced to the scientific community by Karl Pearson as a linear dimensionality reduction method. The concept is to map higher dimensional spaces $\mathbb{R}^n$ data to lower ones represented by the principal axes hence providing a hierarchical orthogonal coordinates system. PCA is done by changing the basis that spans data dimensions[1] (see Figure 4.2). Dimensionality reduction is made using PCA by maximizing the variance of the data around the principal axes. Variance maximization follows the same concept of contrast maximization [Gallego et al., 2018] where we try to find the axes around which the data (the events) provide the maximum spread. Consequently, the least principal axis (with the lowest variance around it) represents the velocity vector we are looking for. Since variance is maximized around the singular vectors representing the data, eigenvalue decomposition can be used to find the principal components of the data.

[1] PCA provides hierarchical principal axes where each axis is scaled to span a principal dimension of the data.

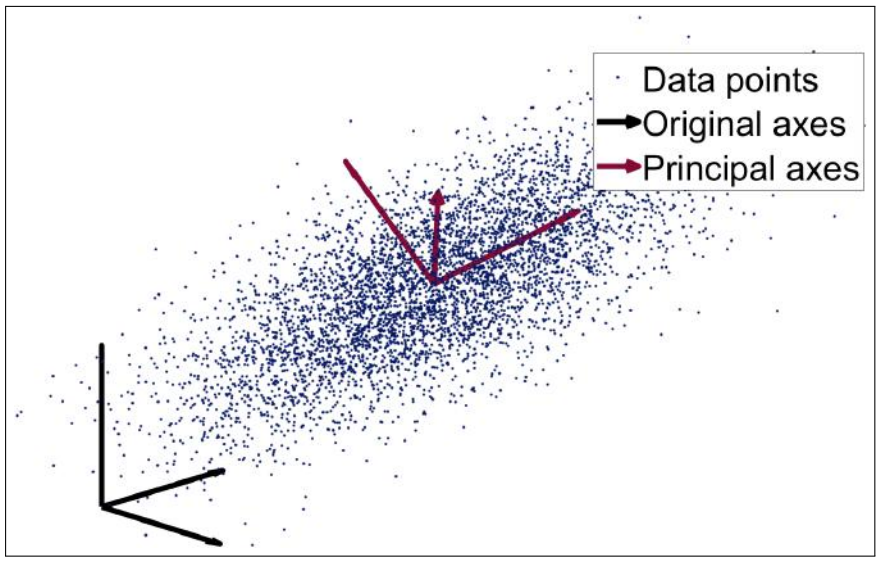


FIGURE 4.1: set of $\mathbb{R}^3$ scattered points with their original axes represented in the basis drawn in black and their normalized principal axes represented in the basis drawn in red.

Based on the hypothesis that triggered events from edges create a plane in a small neighborhood (see Figure 4.2), PCA can be employed to find the two orthogonal vectors spanning the created plane and the third will be the plane's normal[2] corresponding to the optical flow. PCA is found to be able to estimate the plane's normal and provide better accuracy than least-square plane fitting algorithms provided in the state-of-the-art [Benosman et al., 2013, Mueggler et al., 2015, Rueckauer and Delbruck, 2016] (see Figure 4.3). To estimate optical flow, our algorithm is implemented in three steps:

1. Events provided by the camera are conditioned to put aside redundant events or events created by noise or strong luminosity changes.
2. The optical flow is estimated using the PCA method on conditioned events.
3. Finally, the optical flow is regularized to ensure the robustness of the estimated optical flow.

These steps are detailed extensively in the following subsections.

[2]Plane normal corresponds to 0 variances for a perfect plane (no thickness for flat surfaces).

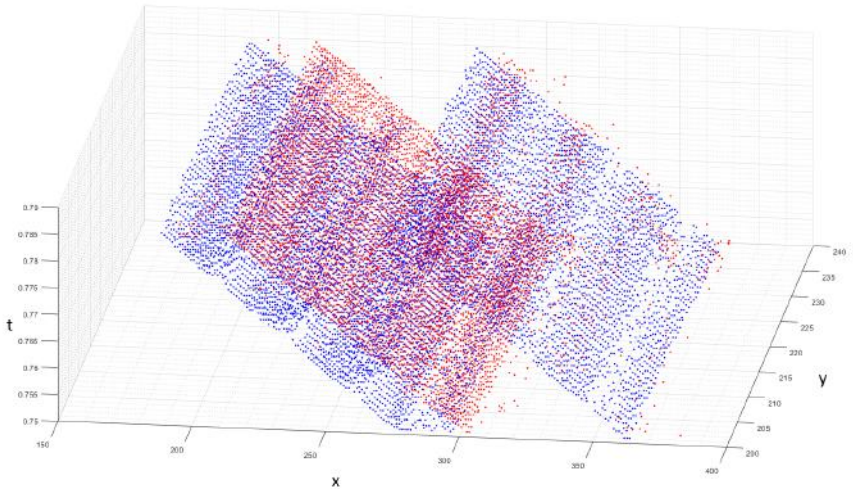


FIGURE 4.2: Positive and negative events created in spatio-temporal neighborhood form a Plane-like shape due to bar motion (planes correspond to different edges of the bar)

4.4.1 Events Conditioning

The independent nature of event-based pixels circuitry that provides signals due to changes in the environment makes event-based cameras highly prone to different noise sources. The events filtering step is required before applying any algorithm to ensure that triggered events correspond to actual changes in the environment. First, positive and negative events are separated and processed independently. A refractory filter marginalizes out events that may be fired as a result of the refractory effect due to sharp changes in intensity [Padala et al., 2018] where consecutive events may be triggered at the same pixel within an infinitesimal time interval. Based on the various scenarios of the used dataset, the limit is chosen to be 20 ms for events triggered with the same polarity and 1 ms for opposite polarity based on practical experiments where the refractory created events were within the chosen values. Next, an adaptive activity filter (see Section 3.3.1 "Page 54") is applied to make sure that all events belong to actual motion. At the end of this step, we obtain only the correctly created events which are then fed to PCA algorithm to estimate the optical flow.

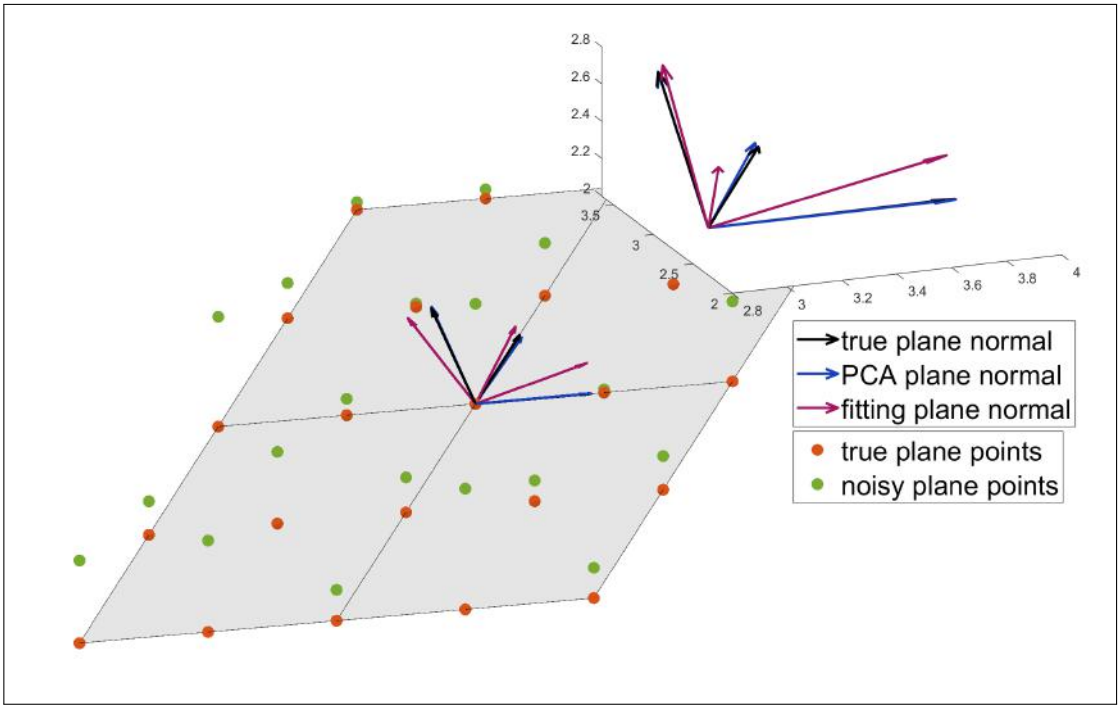


FIGURE 4.3: In red: points spanning the Actual plane. In green: noisy points representing a plane consensus. A closer look at all the coordinates systems shows up to the right, where PCA and the actual plane coordinates look almost identical. In black: the actual plane coordinates system. In blue, the estimated coordinates system estimated using PCA. In red: the coordinates system estimated using least square plane fitting.

4.4.2 PCA Optical Flow Estimation

PCA was first adapted to event-based nature in a line detection and tracking scheme [Everding and Conradt, 2018]. Here, PCA is adapted for optical flow estimation. We estimate the best plane fit of triggered events to obtain spatial components where the plane is expressed as:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} x & y & t & 1 \end{bmatrix} = 0 \tag{4.7}$$

where $(a\ b\ c\ d)$ are the plane parameters to be estimated. The principal components of a set of data points can be computed by finding the eigenvectors of the covariance matrix $\mathbf{\Sigma}$ on this set of centralized points. Points can be centralized by subtracting their

mean. The first principal component (vector) will be the one corresponding to the largest eigenvalue, and the next ones correspond to lower eigenvalues. A set of n polarized events $\{x_i, y_i, t_i\}_n$ created in a spatio-temporal neighborhood $\mathbf{N}$ in $\mathbb{R}^3$ around the event under test is centralized by subtracting the mean of each dimension such that:

$$\mathbf{N} = \begin{bmatrix} x_1 - \mu_{\mathbf{x}} & y_1 - \mu_{\mathbf{y}} & t_1 - \mu_{\mathbf{t}} \\ \vdots & \vdots & \vdots \\ x_n - \mu_{\mathbf{x}} & y_n - \mu_{\mathbf{y}} & t_n - \mu_{\mathbf{t}} \end{bmatrix} \tag{4.8}$$

The matrix $\mathbf{N}$ is used to construct the covariance matrix as:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_{\mathbf{xx}} & \sigma_{\mathbf{xy}} & \sigma_{\mathbf{xt}} \\ \sigma_{\mathbf{yx}} & \sigma_{\mathbf{yy}} & \sigma_{\mathbf{yt}} \\ \sigma_{\mathbf{tx}} & \sigma_{\mathbf{ty}} & \sigma_{\mathbf{tt}} \end{bmatrix} \tag{4.9}$$

where $\sigma_{\mathbf{ij}} = \sum_{k=0}^{n} i_k j_k$. The eigenvectors of $\mathbf{\Sigma}$ represent the principal components spanning the neighborhood. Two orthogonal vectors span the plane, and the third one corresponding to the smallest eigenvalue is perpendicular to the plane. Vector $\mathbf{V_p}$ corresponding to the smallest eigenvalue is considered the plane normal. This vector can be used directly to estimate the optical flow. However, an extra step is required to validate the accuracy of estimated parameters. First, we estimate the plane parameters as follows:

$$a = V_{px} \ , \ b = V_{py} \ , \ c = V_{pt} \tag{4.10}$$

$$d = -(V_{px}x + V_{py}y + V_{pt}t) \tag{4.11}$$

(V_{px}, V_{py}, V_{pt}) are the components of the vector V_p, (x, y, t) are the coordinates of the event under test. The plane parameters are used to evaluate t_{est} value for each pixel used to fit the plane where only the event spatial coordinates are used.

$$t_{est} = -\frac{ax + by + d}{c} \tag{4.12}$$

If the absolute difference between t_{est} and the actual triggering time of the event t exceeds a threshold δ, the event is considered an outlier. A consensus value for inliers should be more than $(1-\epsilon)N^2/2$ where ϵ is the accepted ratio of outliers. Otherwise, the estimated plane will be rejected. The estimation of optical flow is obtained as follows:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{-V_{pt}}{V_{px}^2 + V_{py}^2} \begin{bmatrix} V_{px} \\ V_{py} \end{bmatrix} \tag{4.13}$$

where $(-V_{pt})/(V_{px}^2 + V_{py}^2)$ represents the optical flow magnitude and (V_{px}, V_{py}) represents the normalised vector of optical flow orientation. Hence the duration of the appearance of each event, namely event's lifetime [Mueggler et al., 2015], is obtained:

$$t_{lifetime} = \frac{V_{px}^2 + V_{py}^2}{V_{pt}} \tag{4.14}$$

Algorithm 2 demonstrates a step-by-step pseudo-algorithm of our method.

4.5 Spatio-Temporal Optical Flow Regularization

Due to the sparse and noisy nature of events in neuromorphic sensors, it is required to go further and refine the estimated optical flow to ensure it depicts the actual optical flow. We tried two different spatio-temporal regularization techniques to see which would work best with regard to estimation quality and computation time.

Whilst methods in the state-of-the-art require solving a system of equations that depend on the chosen neighbourhood size, PCA-only[3] technique benefits from the fact that the neighbourhood's size will have little effect on computation time. Consequently, our first technique for regularization, PCA with levels regularization, uses different neighbourhood size levels to estimate the optical flow and take the mean as the best estimate to ensure

[3] we always compute eigenvectors of a 3×3 matrix, which would take most of computation time.

that extremes are filtered out (see Figure 4.4).

Algorithm 2 PCA Optical Flow Estimation

Input: $e\{x, y, t, p\}_{i=0}^{N-1}$, ϵ, N $t_s = 20\ ms$, $t_o = 1\ ms$
Output: $e\{x, y, t, p, v_x, v_y\}_{i=0}^{N-1}$

$ev \leftarrow zeros(col, row, 2)$
$flow \leftarrow zeros(col, row, 2, 3)$
for $i \leftarrow 0 : N-1$ **do** ▷ Events acquisition loop
  $flag \leftarrow 0$
  $x, y, t, p \leftarrow e_i$
  $p_o \leftarrow opposite\ polarity$
  if $(t_i - ev(x_i, y_i, p_i) < t_s)$ & $(t_i - ev(x_i, y_i, p_o) < t_o)$ **then** ▷ Events filtering condition
    $T_f \leftarrow adaptive\ time\ eqn(3.2)$
    if in neigh $(t_i - t_{neigh}) < T_f$ **then**
      $flag \leftarrow 1$
      $ev(x_i, y_i, p_i) = t_i$
    end if
  end if
  if $flag = 1$ **then** ▷ Optical flow estimation
    $neigh \leftarrow \{e\}_1^{n^2}$ in ev in $n \times n$ neighborhood
    if $size(neigh) > 3$ **then**
      $\mu_{\mathbf{x}} \leftarrow mean(neigh_x)$
      $\mu_{\mathbf{y}} \leftarrow mean(neigh_y)$
      $\mu_{\mathbf{t}} \leftarrow mean(neigh_t)$
      $\mathbf{N} \leftarrow Equation(4.8)$
      $\Sigma \leftarrow eqn(4.9)$
      $[V, E] \leftarrow eig(\Sigma)$
      if $E(0,0) < \epsilon$ **then**
        $\{t_{est}\}_{n^2} \leftarrow eqn(4.12)$
        if $\sum_1^{n^2} |t_{est} - t_i| \leq \delta$ **then**
          $[v_{xi}, v_{yi}] \leftarrow Equation(4.13)$
        else
          $[v_{xi}, v_{yi}] \leftarrow 0$
        end if
      else
        $[v_{xi}, v_{yi}] \leftarrow 0$
      end if
    else
      $[v_{xi}, v_{yi}] \leftarrow 0$
    end if
  end if
end for

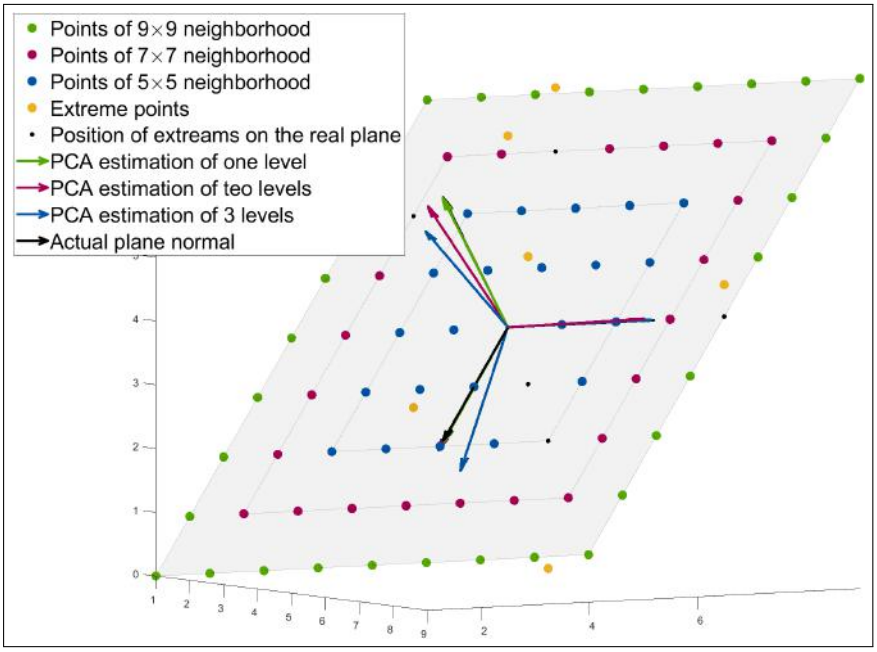


FIGURE 4.4:]
Three levels of neighborhood are shown, 5×5 neighborhood (in blue), 7×7 neighborhood (in red), 9×9 neighborhood (in green) and the extreme points (outliers, in yellow). PCA estimations on the three levels are shown where a larger neighborhood ensures better estimation of plane normal.

The second technique we experimented, PCA with weights regularization, consists in storing the optical flow in a buffer called "active optical flow", then weightings are applied to the previously estimated optical flow in a specific neighbourhood of the event under test. Weights θ_i are chosen to be inversely proportional to the timestamp differences between the event under test and events triggered in the neighbourhood so that the final optical flow estimation will be:

$$\begin{bmatrix} v_x \\ \\ v_y \end{bmatrix} = \sum_{i=1}^{n} \theta_i \begin{bmatrix} v_{xi} \\ \\ v_{yi} \end{bmatrix} \tag{4.15}$$

Choosing the inverse of exponential would be a reasonable weighting function choice for its vanishing values for higher time differences. This function is found to significantly suppress the effect of older optical flow in the neighbourhood (see Figure 4.5). However, among all possible weighting functions, we chose to use the inverse of the difference because it gives relatively correctly distributed weightings while being the fastest to compute. It has been

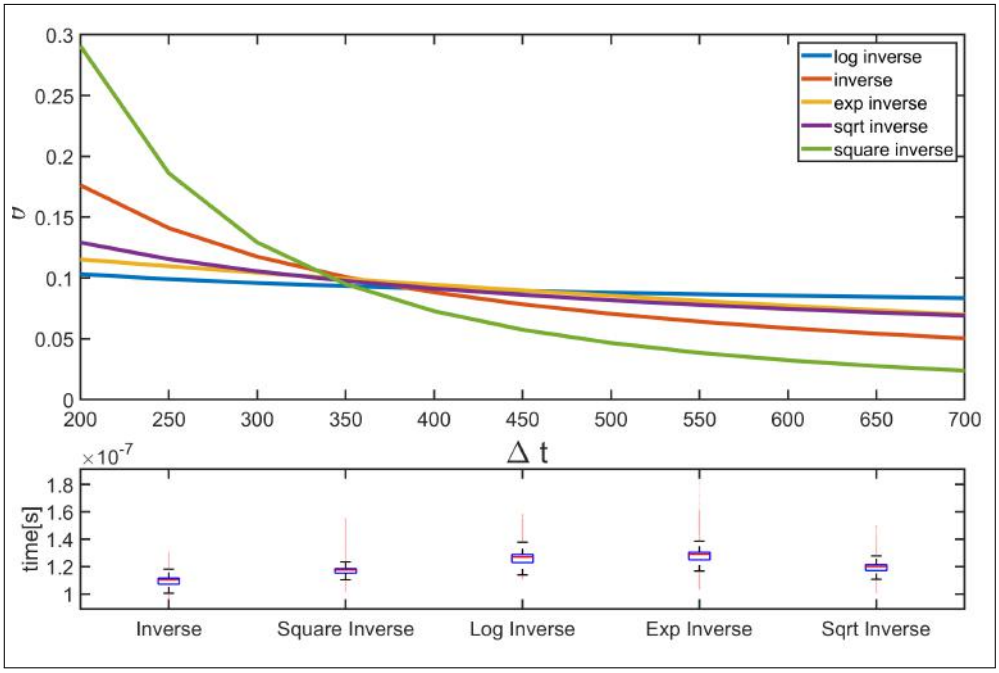


FIGURE 4.5: Top: Variation of weighting functions plotted according to actual collected data. The inverse of square (in green) gives extreme weights change, inverse logarithmic gives averaging-like effect (in blue) and the inverse gives weightings to be distributed in between (in red). Down: the average computation time of each weighting method where the inverse gives the lowest computational time.

found that choosing a smaller neighbourhood for weightings[4] than the neighbourhood chosen to estimate the optical flow provides better results. The regularization process is shown in Algorithm 3.

4.6 Experimental Setup for Validation

To validate the reliability and accuracy of the proposed algorithm in different cases where no specific kind of motion is favored, we used different datasets with various conditions. Other event-based optical flow datasets were introduced in the state-of-the-art by the time we finished the development phase of our algorithm and were available

[4] i.e. 5×5 for a 7×7 neighbourhood to estimate optical flow.

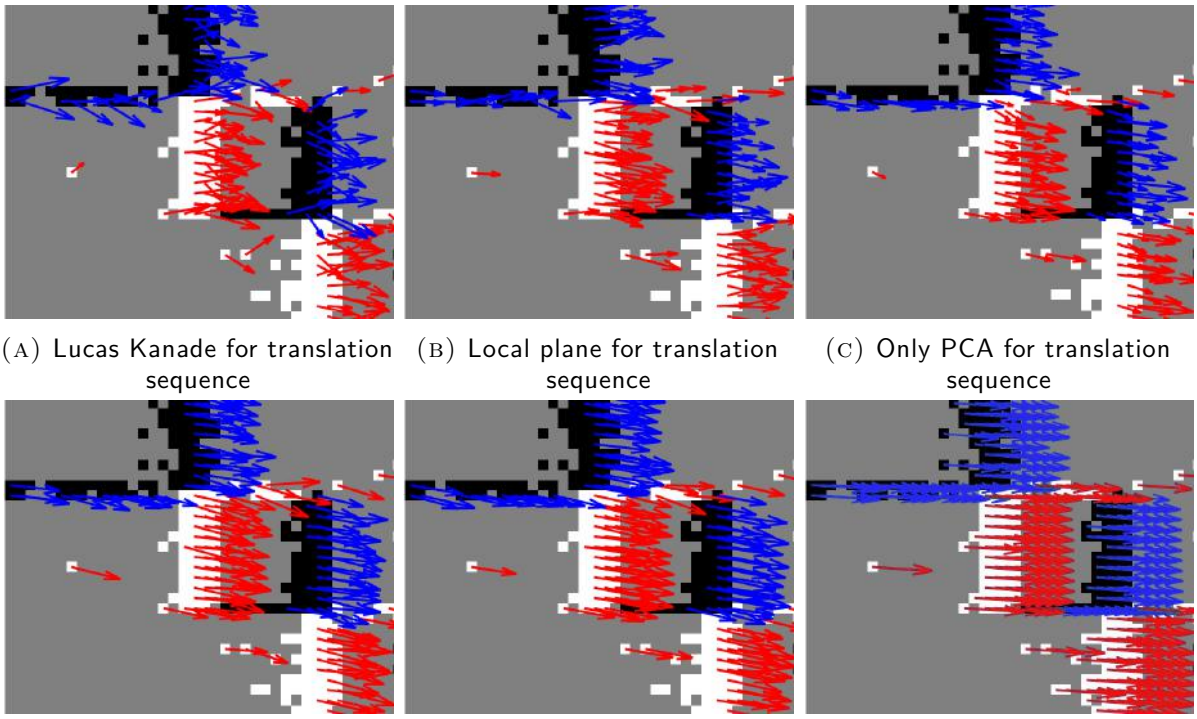
Algorithm 3 PCA Optical Flow Regularization

Input: $[v_x, v_y, t],\ flow$
reg, n_l
Output: regularized $[v_x, v_y]$
if $reg = weighting$ **then**
  $t_n \leftarrow$ times in $n \times n$ neighborhood in $flow$
  $v \leftarrow [v_x, v_y]$ in $n \times n$ neighborhood in $flow$
  $W = \frac{1}{t - t_n}$
  $W = \frac{W}{norm(W)}$
  $[v_x, v_y]_{reg}$ calculated using Equation(4.15)
else
  $v \leftarrow [v_x, v_y]$
  for $i \leftarrow 1 : n_l$ **do**
    Recall: Algorithm(2) with $n = n - 1$
    $v \leftarrow v + [v_x, v_y]$
    $[v_x, v_y]_{reg} = \frac{v}{n_l}$
  end for
end if

for the validation phase. Besides our dataset (see Chapter 3), we used DVSMOTION20 dataset [Almatrafi et al., 2020] and the dataset of moving lines introduced in [Mueggler et al., 2015]. In our own dataset, the usage of VICON motion capture system constricted us to only obtaining optical flow for planar surfaces, hence the usage of checkerboards. Therefore, the choice of other datasets allowed us to yield a comprehensive assessment of our algorithm.

4.6.1 DVSMOTION20 Dataset

DVSMOTION20 [Almatrafi et al., 2020] recorded long sequences using DAViS346 346×260 camera and provided ground truth at camera framerate. They recorded four different sequences named *Checkerboard*, *Classroom*, *Conference Room* and *Conference Room Translation* from which we chose the second and third sequences. Moreover, the recorded sequences feature random motion in different environments where occlusions may occur. As a result, these sequences provide a better assessment of the actual flow and not only the normal flow.



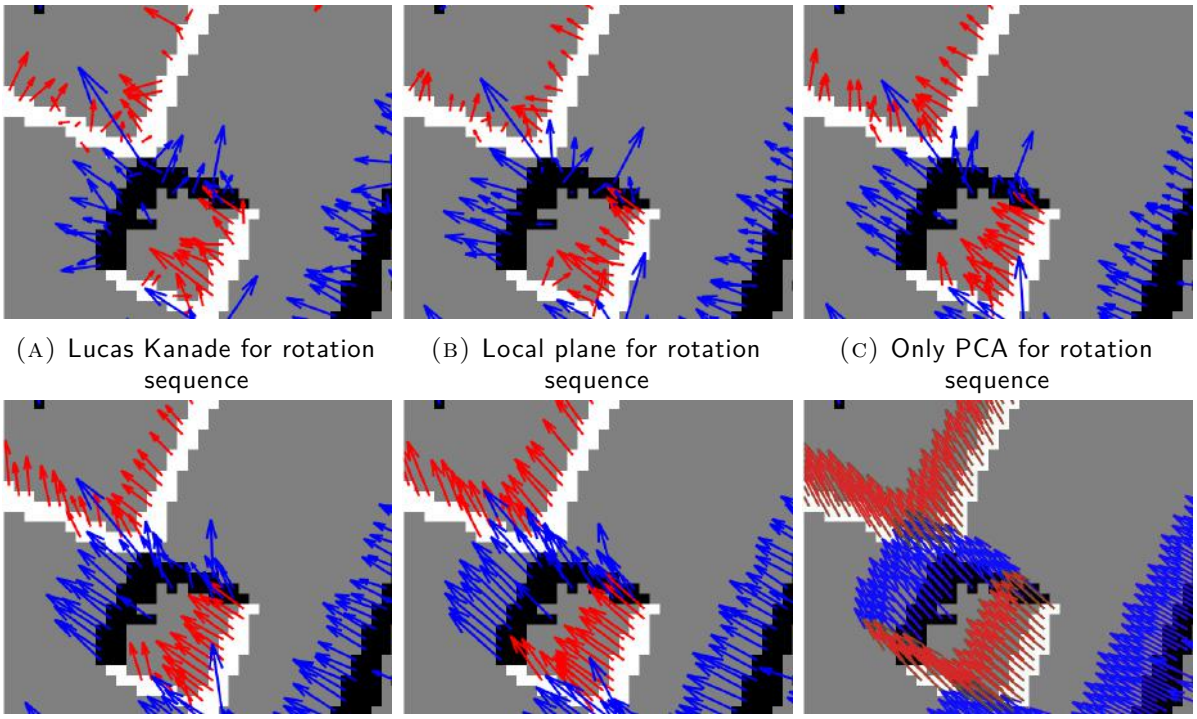
(A) Lucas Kanade for translation sequence (B) Local plane for translation sequence (C) Only PCA for translation sequence

(D) Weight regularization PCA for translation sequence (E) Levels regularization PCA for translation sequence (F) Ground truth for translation sequence

FIGURE 4.6: Results for the checkerboard translation sequence. Only small parts of each sequence is presented for better visualization of results.

4.6.2 Moving Line pattern Dataset

Some neuromorphic vision algorithms construct event frames to use them in frame-based schemes. The event frames require accurate event lifetime estimation to avoid blurry image construction. To test the ability to estimate correct event lifetime, we used the line pattern sequence introduced in [Mueggler et al., 2015]. This sequence is recorded using a 128×128 DVS camera [Lichtsteiner et al., 2008] mounted on a slider that moves in front of lines drawings featuring different depths so that the borders of lines move with different speeds due to the parallax effect.



(A) Lucas Kanade for rotation sequence

(B) Local plane for rotation sequence

(C) Only PCA for rotation sequence

(D) Weights regularization PCA for rotation sequence

(E) Levels regularization PCA for rotation sequence

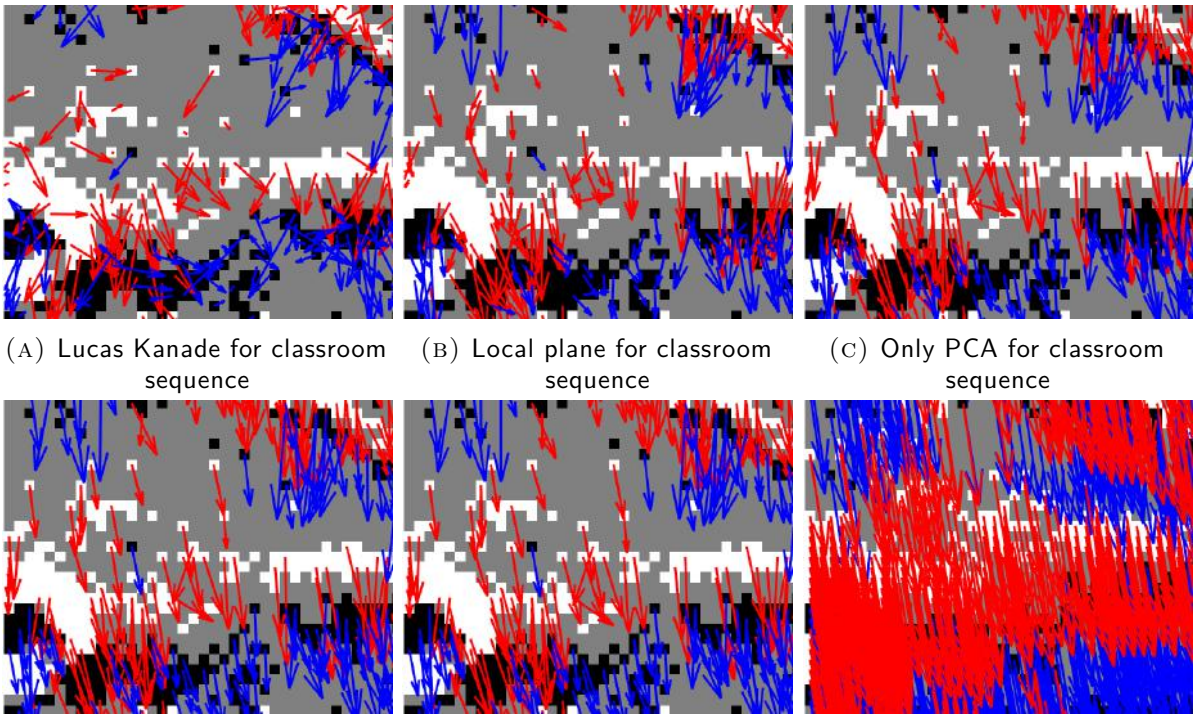
(F) Ground truth for rotation sequence

FIGURE 4.7: Results for the checkerboard rotation sequence. Only small parts of each sequence is presented for better visualization of results.

4.7 Results

Four metrics are used to provide a quantitative and comprehensive comparison. The Average End Point Error (AEPE) measures differences in optical flow magnitude. The Average Angular Error (AAE) measures differences in optical flow orientation. Lifetime error is defined as the difference between the estimated and the actual lifetime of each event. Finally, the computational time is used to assess the opportunity to use these algorithms in real-time applications.

To provide a fair and comprehensive assessment of our algorithm's quality, we compared our algorithm's performance with algorithms that do not require a lot of computational resources or implementation complexity. In the context of our thesis, this choice is made since optical flow is used as a core algorithm of a bigger scheme which implies low



(A) Lucas Kanade for classroom sequence

(B) Local plane for classroom sequence

(C) Only PCA for classroom sequence

(D) Weights regularization PCA for classroom sequence

(E) Levels regularization PCA for classroom sequence

(F) ground truth for classroom sequence

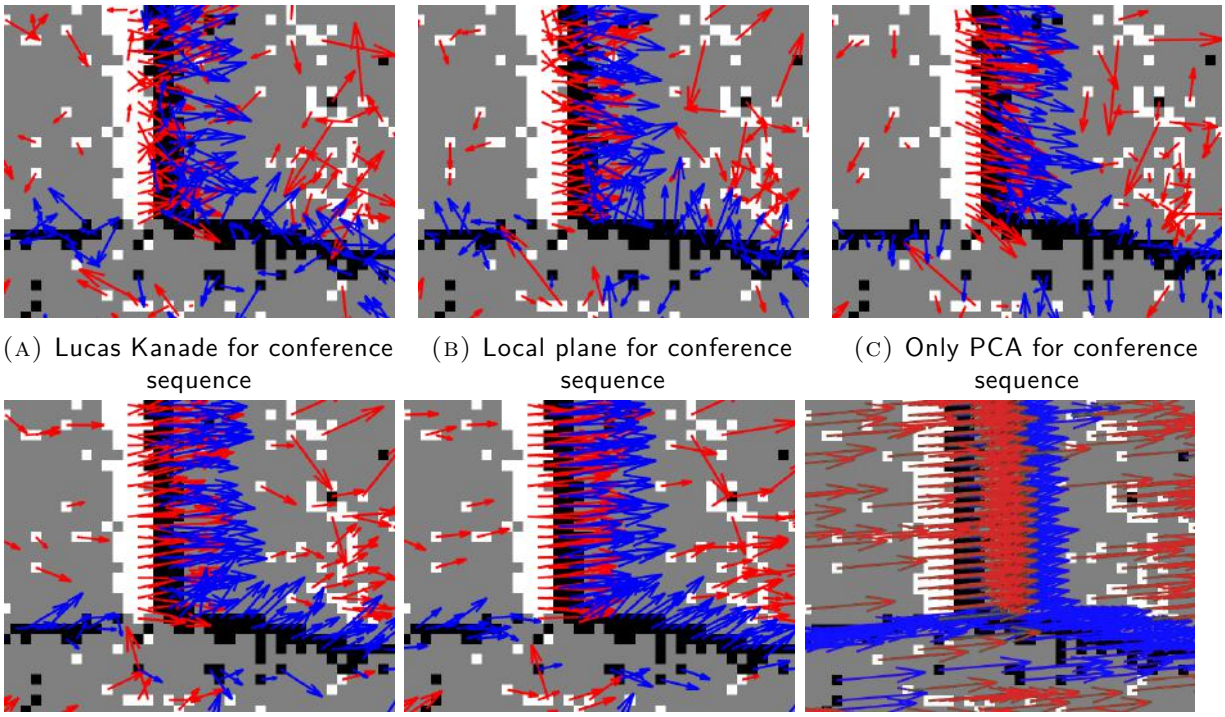
FIGURE 4.8: Results for the classroom sequence. Only small parts of each sequence is presented for better visualization of results.

computational time and acceptable accuracy to spare resources for the whole system. Amongst the event-based optical flow estimation methods in the state-of-the-art (see section 4.2), we chose event-based Lucas-Kanade algorithm [Benosman et al., 2012] and local plane fit algorithm [Benosman et al., 2013] for their low computational power. Figures 4.6, 4.7, 4.8 and 4.9 show a visualization of the obtained results, Table 4.4 summarizes the computation time required to estimate optical flow for each algorithm. Tables 4.1 and 4.2 shows AEPE and AAE performance metrics evaluated using our recorded sequences.

4.7.1 Average End-Point Error

The average end-point error measuring magnitude errors between estimated and ground-truth flow is defined as:

$$AEPE = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{u}_i - \hat{\mathbf{u}}_i|| \tag{4.16}$$



(A) Lucas Kanade for conference sequence

(B) Local plane for conference sequence

(C) Only PCA for conference sequence

(D) Weights regularization PCA for conference sequence

(E) levels regularization PCA for conference sequence

(F) Ground truth for conference sequence

FIGURE 4.9: Results for the conference sequence. Only small parts of each sequence is presented for better visualization of results.

where $\mathbf{u}_i$ and $\hat{\mathbf{u}}_i$ are the estimated and the ground truth optical flow respectively.

Translation scenario: Considered the easiest sequence, the best results were attained. The usage of PCA, in general, shows better accuracy in magnitude estimation (see Figure 4.6). Applying PCA-only achieved a relative error of 6.9%. Applying PCA with levels regularization provides the best magnitude estimation and reduces the overall estimation error to 4.6% (see Table 4.1). PCA with weights regularization shows a slight improvement in magnitude estimation compared to PCA only.

Rotation scenario: Estimating optical flow in rotation motion is considered more critical because magnitude would vary in a small neighbourhood. For this reason, AEPE is shown to be slightly higher than in the translation scenario (see Table 4.1). PCA with levels regularization gives the best results (see Figure 4.7).

Conference scenario: A real life scenario but less cluttered compared to the conference sequence, PCA-only attained 20.7% error while PCA with levels regularization improved the quality to only 18% error (see Figure 4.9).

Classroom sequence: Densely cluttered environment providing the most challenging scenario amongst all sequences. PCA-only achieved 22.1% error and PCA with levels regularization attained 18.7% with a performance similar to the conference sequence (see Figure 4.8).

In all of the cases, PCA with levels regularization provided 1.2 times less AEPE than the next best non-PCA algorithm provided in the comparison (see Table 4.1).

4.7.2 Average Angular Error

The metric to measure the differences in orientation between estimated and ground truth flow is the average angular error, defined as:

$$AAE = \frac{1}{N}\sum_{i=1}^{N} cos^{-1}\left(\frac{\hat{\mathbf{u}}_i^T \mathbf{u}_i}{||\hat{\mathbf{u}}_i||||\mathbf{u}_i||}\right) \tag{4.17}$$

Translation scenario: The choice of neighbourhood size deeply affects the accuracy of orientation estimation, which is why PCA with weights regularization always provides better results. PCA with weights regularization uses a smaller neighbourhood level for regularization along with a neighbourhood optical flow consensus, which creates

Algorithm	`translation` AEPE (%)	`rotation` AEPE (%)	`conference` AEPE (%)	`classroom` AEPE (%)
Lucas Kanade	13.1 ± 4.6	17.1 ± 7.3	27.1 ± 10.2	28.7 ± 14.3
Local Plane Fit	10.3 ± 3.2	13.8 ± 7.4	22.9 ± 10.4	25.6 ± 11.7
PCA-only	6.9 ± 2.4	8.1 ± 3.2	20.7 ± 5.7	22.1 ± 3.2
PCA with weights	6.1 ± 2.3	7.5 ± 4.0	18.2 ± 6.9	19.6 ± 4.1
PCA with levels	$\mathbf{4.6 \pm 1.5}$	$\mathbf{7.1 \pm 4.8}$	$\mathbf{18.0 \pm 5.8}$	$\mathbf{18.7 \pm 3.7}$

TABLE 4.1: Relative average end point error.

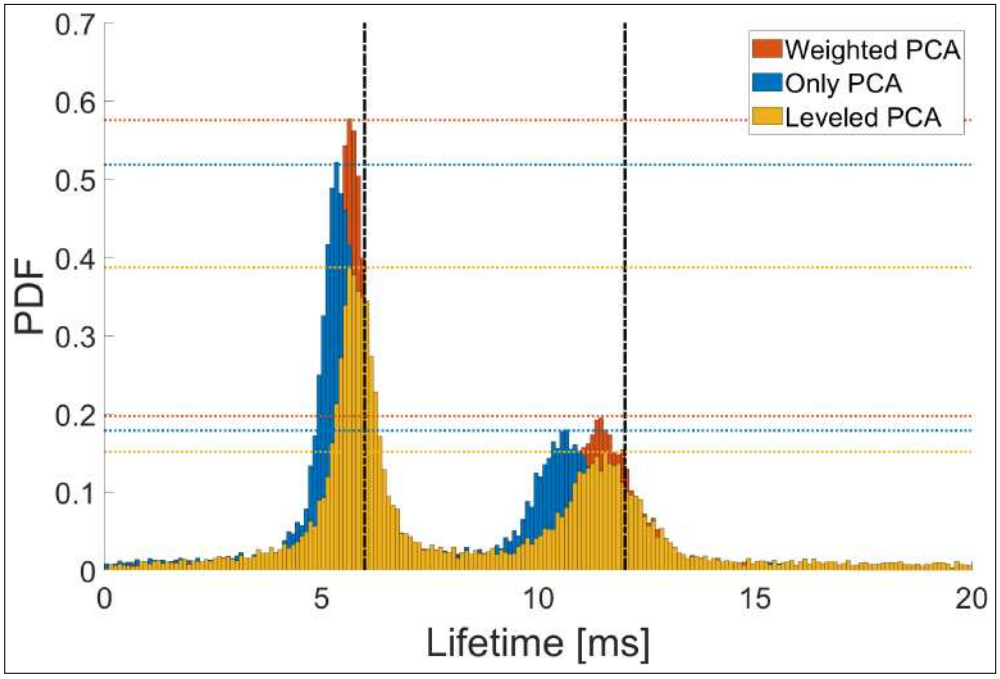


FIGURE 4.10: Probability Density Function (PDF) of the estimated lifetime for only PCA (Blue), weights regularization PCA (Red) and levels regularization PCA (Yellow) of the "moving lines" sequence in [Mueggler et al., 2015].

a smoothing effect. On the other hand, PCA with levels regularization improved the orientation estimation and boosted the accuracy but did not maintain maximum accuracy (see Table 4.2).

Rotational scenario: As expected, rotational scenarios are harder to estimate and accuracy may be reduced. Results provided by the three PCA approaches did not show much variation from each other. However, PCA with weights regularization attained the best results (see Figure 4.7).

Algorithm	`translation` AAE (°)	`rotation` AAE (°)	`conference` AAE (°)	`classroom` AAE (°)
Lucas Kanade	15.6 ± 3.8	19.6 ± 7.1	23.6 ± 11.1	26.0 ± 9.3
Local Plane Fit	12.5 ± 5.1	15.3 ± 6.1	19.7 ± 10.7	22.8 ± 10.2
PCA-only	7.8 ± 4.6	12.8 ± 4.5	16.4 ± 5.8	14.2 ± 6.8
PCA with weights	$\mathbf{5.7 \pm 4.6}$	$\mathbf{11.2 \pm 3.3}$	$\mathbf{13.7 \pm 6.2}$	$\mathbf{12.9 \pm 5.2}$
PCA with levels	6.6 ± 5.1	11.6 ± 4.9	14.1 ± 5.4	13.0 ± 5.2

TABLE 4.2: Average angular error for rotational and translational sequences.

Conference scenario: The motion of this sequence (as well as the classroom sequence) is considered a good test to validate the ability to estimate correct orientation. PCA-Only achieved an error of $16.4°$ and PCA with weights regularization provided – as expected – the smallest estimation error with only $13.7°$.

Classroom sequence: PCA-only achieved $14.2°$ error and PCA with weights regularization attained $12.9°$ with a better performance than on the conference sequence which may be due to the existence of many edges perpendicular to motion direction (see Figure 4.8).

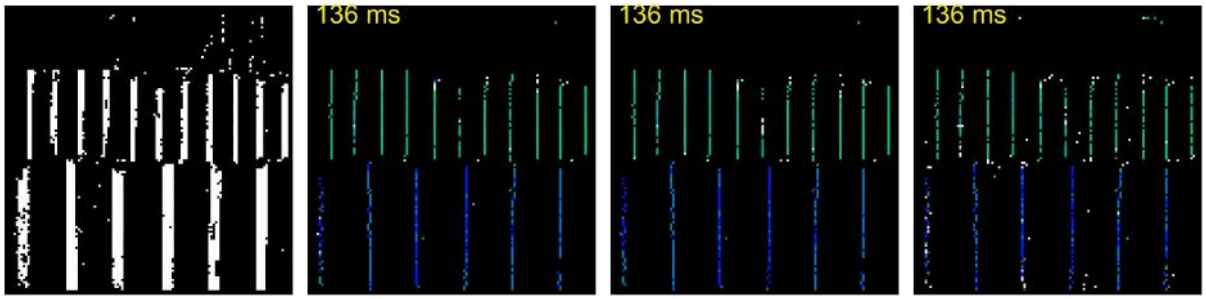
While the differences between levels and weights regularization is not significant, weights regularization provided at least 1.3 better AAE than the next non-PCA algorithm in the comparison (see Table 4.2).

4.7.3 Lifetime Estimation Error

Lifetime is considered one of the most important feature assigned to event-based cameras. It helps sharpen frames created from stacked events used in event-based visual odometry and SLAM algorithms [Mueggler et al., 2015]. All PCA algorithms provided reliable lifetime estimation compared to the state-of-the-art [Mueggler et al., 2015, Low et al., 2020]. We used the "moving lines" sequence where two sets of lines move at different constant speeds, resulting in a constant lifetime for each event generated by these lines ($6ms$ for fast lines and $12ms$ for slow lines). Figure 4.10 shows the probability density function (PDF) of the estimated lifetime. Using PCA only, 52.19% of events were assigned a $5.35\ ms$

Algorithm	fast_stripes			slow_stripes		
	max bin (ms)	%	error %	max bin (ms)	%	error %
PCA Only	5.3	52.2	10.8	10.7	17.9	11.3
Weighted PCA	5.5	57.7	7.5	11.1	19.44	7.91
Levelled PCA	5.7	38.6	5.8	11.5	15.1	4.6

TABLE 4.3: The maximum bin value for the life time estimation of the dataset provided in [Mueggler et al., 2015] and the percentage of each bin with the error percentage.



(A) accumulation of events for 20 ms (B) Lifetime for PCA only (C) Lifetime for weights regularized PCA (D) Lifetime for levels regularized PCA

FIGURE 4.11: From left to right, Events stacked during $20ms$ for the moving lines sequence, only active events are shown according to PCA only algorithm, PCA with weights regularization algorithm and PCA with levels regularization algorithm all at $136ms$. Events color represent their life time as a gradient from dark blue to white

lifetime with an error of 10.83% on the fast stripes and 17.95% were assigned $10.65\ ms$ lifetime with an error of 11.25% on the slow lines. PCA with weights regularization gave the thinnest distribution around the maximum bin value (least variance). The nearest large bin to fast stripes has the value of $5.55\ ms$ with 57.72% of all events and 7.5% error. For slow stripes, the largest bin's lifetime is $11.05\ ms$ with 19.44% of all events, which translates to an error of 7.91% of the actual lifetime. PCA with levels regularization gives the best results but with the largest variance around actual lifetime with a percentage of 38.58% and 15.11% around the fastest and slowest stripes respectively and errors of 5.83% and 4.58% (see Table 4.3). The high variance of PCA with levels regularization creates many falsely estimated lifetimes (see Figure 4.11).

Algorithm	Computational Time per Event (μs)
Lucas Kanade	3.32 ± 1.05
Local Plane	1.18 ± 0.76
PCA only	$\mathbf{0.29 \pm 0.05}$
PCA with weights	0.51 ± 0.06
PCA with Levels	0.78 ± 0.07

TABLE 4.4: Computation times required per event.

4.7.4 Computational Time

The PCA method is based on computing the eigenvectors of a 3×3 matrix constructed incrementally by the chosen neighborhood. Consequently, a much lower computation time is required compared to other techniques in teh state-of-the-art. For the same reason, the change in the neighborhood size, contrary to other methods, does not affect the computation time much. As a result, the PCA-only method is shown to be able to process around $3.5\ Mevent/sec$. PCA with levels regularization increased the computation time as expected. However, it is still able to process about $1.3\ Mevent/sec$ which remains acceptable for the provided events frequency. Although these algorithms are implemented on a Linux machine with a Core i5 3.10 GHz processor, the provided computation time of PCA variants is shown to be competitive to be selected for real-time applications.

4.8 Conclusion

We presented a novel method for event-based optical flow estimation based on PCA. We have shown that the proposed method is more adapted to the sparse nature of event-based cameras and produces significantly less noise than other methods. Although performance improvements always come at the cost of computational power, our proposed method spares the processor resources and drastically reduces computation time (about 2 times faster than the next fastest non-PCA compared algorithm) while keeping acceptable performance. The straightforward and non-complex procedure of the PCA event-optical flow approach is the method's strength besides providing accurate and fast computed estimation. The improvements made using PCA for event-based optical flow allow us to involve it as the base algorithm for event-based visual odometry. Despite the gain we attained in computational time, incorporating a few millions of events' optical flow information per second in an optimization scheme would undermine the real-time capabilities. For this reason, it would be beneficial to find a better representation of events that would provide robust data reduction without loss of information. Since the provided events are mainly

the contours where the change occurs ans since these contours can represent lines in a structured environments, we chose to encode events as lines (whenever possible) and feed them to the optimization scheme instead.

5 Neuromorphic Line Detection and Tracking

Chapter abstract

One of the challenges in computer vision, such as line-based SLAM (Simultaneous Localization and Mapping) and visual odometry schemes, is line detection and segmentation. In this chapter, we address this problem and provide a fast method to compute line detection by avoiding time-consuming search algorithms and limiting the use of any complex implementation. Our algorithm exploits the geometrical features of the scene and applies simple heuristics to ensure correct line detection and segmentation. As a result, the inherent line detection problems that lead to heavy computation or false estimation, like the use of search algorithms or wrong line connection, are addressed and tackled. Furthermore, the results are accurate in different motion scenarios (pure rotational, pure translational and free motion) and robust against false line segmentation.

5.1 Introduction

A reconstructed 3D point, despite being straightforward and sufficiently represented as a tuple (x, y, z), using feature points for SLAM systems is costly and requires attentive consideration for errors and outliers. Moreover, besides the fact that SLAM optimization requires, in many cases, several point features for robust estimation, a point

feature can be falsely described. Instead of depending on feature points, many works ([Lemaire and Lacroix, 2007], [Ruifang et al., 2017] and [Hirose and Saito, 2012] for example) shifted their point of view toward line features for their data reduction capability and for being less prone to the reprojection and matching errors, which leads to easier tracking.

In the context of our thesis, for event-based visual-inertial odometry, since a single event contributes very little information about the environment, we adopt the same methodology of depending on line features instead of point features. Representing structured environments (indoor environments) using line features helps solve the problem of data association for event-based cameras and provides data reduction for the massive stream of events supplied. Moreover, lines help speed up the optimization process for visual odometry and can construct a better and more intuitive map of structured environments. In this chapter we introduce a fast method to detect and track line segments for neuromorphic vision sensors based on optical flow.

5.2 Neuromorphic Line Detection and Tracking

Detecting and tracking objects (lines included) in the scene is one of the most studied fields in computer vision since it may provide information about the geometric structure of the scene. Lines are considered one of the distinct and repetitive geometric structures that exist in the world around us, for which reason, extracting lines is convenient to reconstruct the scene using either image frames or events. Consequently, after the emergence of neuromorphic vision sensors, many attempts have been conducted to adapt existing conventional frame-based Line detection and tracking algorithms, while others contributed with novel ideas that appropriately fit the event-based model for better representation.

In an attempt to produce an event-based line segment detector, [Brändli et al., 2016] adapted a frame-based algorithm to neuromorphic vision sensors and presented ELiSeD

(an Event-based adapted version of LSD algorithm [Von Gioi et al., 2008]). The proposed ELiSeD can be considered the first event-based algorithm to accurately detect lines in the scene. The algorithm starts by separately storing the negative and positive events in an activity map[1]. A Sobel filter is applied to the activity maps to detect the orientation of each event. Adjacent events with the same orientation (the output of the Sobel filter) within specific time intervals are clustered together to produce line segments. This method provides lines with discretized orientation according to sobel filter output. It would require heavy computations to apply a Sobel filter accompanied by a trigonometric function to track lines in an event-based scheme.

Hough transform, one of the most robust computer vision algorithms, is used to transform shapes – lines included [Duda and Hart, 1972] – in image space into a unique point in some parameter space. [Seifozzakerini et al., 2018] discussed the challenges and opportunities of applying Hough transform on event-based cameras. They employed a spiking neural network (SNN) to overcome the computational power needed for Hough transform. As a result, their event-based Hough transform implementation can provide accurate line segments – or shapes. However, the use of SNN requires good training and meticulous tuning.

A novel method that does not rely on frame-based reproduction of algorithms for neuromorphic vision sensors has been introduced in [Everding and Conradt, 2018]. In this work, they exploited the spatio-temporal planes constructed by events triggered from moving lines in (x, y, t) domain where any moving object creates extrusion of its contours in the time direction (see Figure 4.2 "Page 79"). They adapted PCA method [Pearson, 1901] to represent the principal 3D bases and estimate line parameters (length, vector and center) afterward. Using PCA notably enhanced the computational time to achieve real-time performance if the frequency of events was not extremely large. Nonetheless, it did not cope with rotating lines where triggered events do not create planes but spirals in (x, y, t)

[1] A map where only the timestamp of the last triggered event is stored in each pixel

space. Moreover, distinctive lines in the scene may also be falsely connected if their ends were detected adjacently at a certain moment and it would be difficult to separate them later.

Similar to the method we introduce in this chapter, [Valeiras et al., 2018] rely on the presence of optical flow for events and apply least-square weighted fitting on events clusters to detect line segments. Using optical flow to estimate line segment clusters improved line detection and estimation for both translating and rotating lines. However, the complexity of the optimization scheme and the tuning required for the decaying functions fed to the optimization scheme make this scheme difficult to master. Moreover, the computational time required for line detection, besides the time required for optical flow estimation, constrains the usage in real time.

5.3 Flow-Based Line detection and tracking

The methods discussed in the previous section propose novelties to tackle event-based line detection and tracking problem. Moreover, they suffer from drawbacks in either providing accurate estimation or requiring computational resources. This section presents a method to solve event-based line detection and tracking problem aided by optical flow to help yield data reduction and better representation of the environment. We tackle the problem of event-based line detection and segmentation while avoiding the issues encountered with the methods mentioned above. Henceforth, we approach the problems of event-based output refinement, algorithm complexity, real-time applicability and accuracy of tracked line segments. The improvements we present in this chapter can be summarized according to the following items:

- We use geometric properties inherent to lines and points where no trigonometric or complex functions are used to maintain a low complexity scheme and fast computational time.

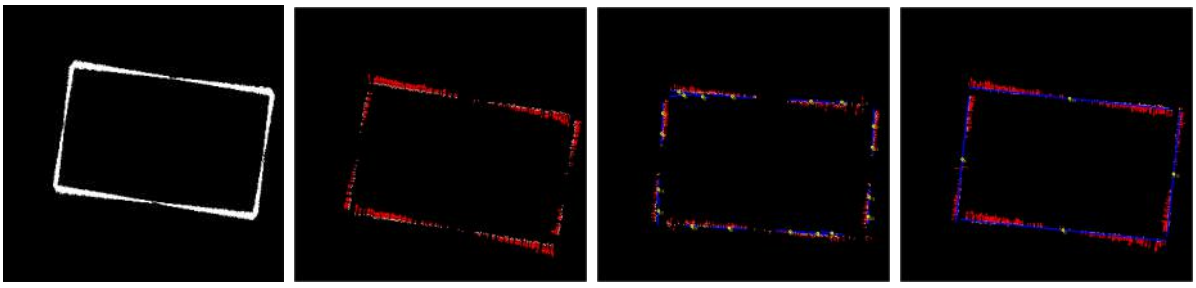
- Search algorithms are not employed to associate adjacent events to line clusters[2].
- Optical flow consensus is used to ensure no lines are falsely stitched, which improves tracking quality and the reusability of this scheme.
- We can detect all kinds of line motion (translations and rotations) and also use line clusters to refine the optical flow.

Neuromorphic vision sensors operate in consistency with the brightness constancy condition and provide a tuple of asynchronous data (see Section 4.3 "Page 75"). Following this model, packets of events created during a specific time interval are used to cluster events corresponding to a line segment or a portion of it. Our approach starts with events preprocessing and ensuring that accurate optical flow is assigned to each event (see Chapter 4). The preprocessed events are then used to create kernels (candidate events that may be promoted to a line) where certain checks (discussed in the following subsections) should be verified, and if a kernel exceeds a predefined threshold, it would be considered a line kernel. A kernel that gets upgraded to a line is characterized by specific parameters that help highlight the line and decide whether or not to include any upcoming event. Nearby lines with the same parameters can then get stitched together. Finally, global reset and cleaning are carried out to remove old and outlier events recorded in the activity map, buffer, kernels and lines to ensure the algorithm's robustness. Each step is explained thoroughly in the following subsections and summarized in Figure 5.1.

5.3.1 Preprocessing and Flow Stamping

The goal of our algorithm is to detect lines knowing that each event is stamped with its own optical flow. To achieve this goal, we use a method of PCA event-based optical flow (see Chapter 4) which also ensures that falsely created events are suppressed by using the adaptive activity filter. This method ensures that the estimation is performed

[2]Even the binary search algorithm where ordered lists are searched would be a burden in most cases where either the number of lines or the frequency of the events is high, and this process needs to be repeated many times.



(A) Denoised events (B) Optical flow (C) Small line segments (D) Stitched lines

FIGURE 5.1: The steps of our proposed algorithm, from left to right, (5.1a) a packet of events is denoised to use only truly triggered events, (5.1b) the optical flow is estimated and kernels are created (kernels are not drawn), (5.1c) small lines appears in the scene (shown here for visualisation purpose only), (5.1d) stitched lines represent actual lines in the scene.

efficiently while keeping the processing time low. For lines detection, we used the method of PCA only to maintain low computational power[3]. At the end, each event is altered from $\langle x, y, t, p \rangle$ to $\langle x, y, t, p, \nu_x, \nu_y, t_l \rangle$ (see Algorithm 4 Line 5) where ν_x, ν_y are the optical flow in x and y directions respectively and t_l is the lifetime of each event as defined in [Mueggler et al., 2015]. Aside from suppressing false positives, this method also provides a sharper image by suppressing the noise generated by the events. After all events have been altered, they are then grouped into packets and sent to the line detector scheme. The line detector scheme first creates kernels for possible line segments then these kernels gets upgraded to line segments whenever certain conditions are met.

5.3.2 Kernel Creation

A kernel is used to cluster adjacent events in the scene to later serve as a possible line candidate. At this step, no consensus is taken into account, and the only condition applied is that the events are sufficiently near each other along a line. Given the high number of events that are produced by event-based sensors (which would reach 1 Mevents/S of preprocessed events in our case), we use these checks to avoid costly search engines. Geometrically, if three points on a line segment are not perfectly aligned, then a triangle

[3] In this chapter, we already correct optical flow with a consensus of events that belong to the same rigid body

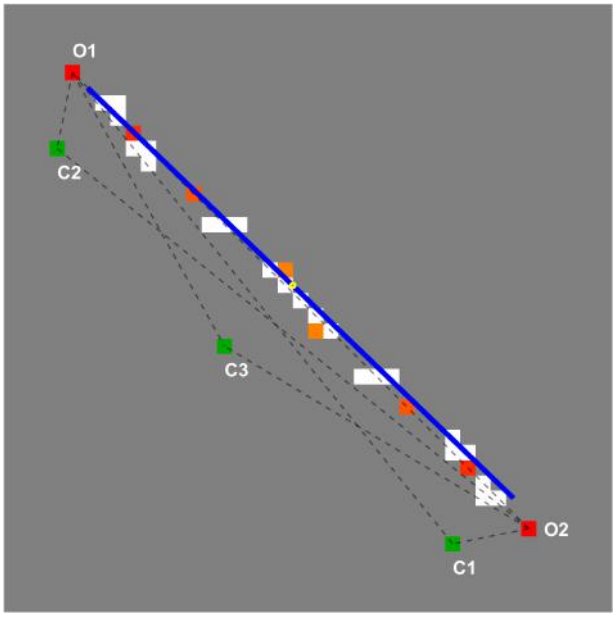


FIGURE 5.2: Events created by a line with some of the saved line tips o_1, o_2 (orange, the innermost tips, red, the outermost tips) with three possible candidates c_1, c_2, c_3 to be joined in green (either between the existing line tips, or replacing one of them) and the estimated line in blue.

can be constructed. For event-based cameras, if two events happen to define a line segment's tips, then a kernel will be created if the third new event occurs to define the triangle's third tip. The first two tips of the triangle represent the base and the event under test undergoes a height that should be less than a predefined threshold σ_l. To determine if the event is less than the threshold, we check the equality of the triangle's area using the base l, the height h and the lengths of its sides. The triangle perimeter is defined by $p_t = d_1 + d_2 + l$ where d_1 and d_2 are the distances between the event under test and the line kernel tips and l is the length of the line kernel. Using the lengths of the triangle sides we can calculate its area accordingly:

$$a_t = \sqrt{p_t(p_t - d_1)(p_t - d_2)(p_t - l)} \tag{5.1}$$

If we reformulate equation (5.1) and equate it with the base-height area, we deduce its height (normal distance between the event and the line kernel):

$$h^2 = \frac{4a_t^2}{l^2} \tag{5.2}$$

If $h^2 < \sigma_l^2$ then the event can be further examined to determine if it will create a new line tip or remain between the existing line tips (see Figure 5.2). An event candidate c_3 can remain between existing line tips o_1 and o_2 if $d_1 \& d_2 < l$, c_2 can be an new tip outside and near the first tip o_1 if $d_2 > l$ and $d_1 < \sigma_l$ or c_1 an new tip outside and near the second tip o_2 if $d_1 > l$ and $d_2 < \sigma_l$. The new tip is added on top of its nearest tip as a stack data structure where the oldest ones can be deleted to shrink the line if any of its events disappear from the scene (see Algorithm 4 Lines 15:22). If an event exceeds its lifetime t_l, it will also be deleted. Unmatched events are then added to the buffer for future kernels. A newly created kernels would keep growing in size (or length) until it meets the criteria to be upgraded as a line segment. Unless, it would be deleted after a certain time if not upgraded.

5.3.3 Kernel Upgrade and Line Segmentation

A kernel can collect events that are not part of line, such as a blob or an undefined shape. For a kernel to gain a "line status" and be upgraded, the number of events that can be collected should be greater than the predefined size n_k. Furthermore, the distance between the kernel's outermost tips should be larger than the predefined length l_k. If those conditions are met, a kernel gains the "line status" upgraded to a line segment and gets assigned the following parameters (see Algorithm 4 Lines 12 and 17):

- The mean $\mu_s = (\mu_{\nu_x}, \mu_{\nu_y})$ and variance $\sigma_s = (\sigma_{\nu_x}, \sigma_{\nu_y})$ of all events optical flow as the principal optical flow parameters of the line.

- The line equation $y = ax + b$ using line regression [Edwards, 1984] where a and b are computed using the mean of x and y events coordinates: μ_x and μ_y as follows:

$$a = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{n}(x_i - \mu_x)^2} \tag{5.3}$$

$$b = \mu_y - a\mu_x \tag{5.4}$$

  then a and b are used to transform the slope-intercept equation to a vector equation for better representation of the line segment.

- The center of the line as the mean of all the collected events

$$(\bar{x}, \bar{y}, \bar{t}) = \frac{1}{n}\left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i, \sum_{i=1}^{n} t_i\right) \tag{5.5}$$

  where $\bar{t}$ and the principal optical flow of the line can be used to change the 2D position of the line as follows;

$$(\bar{x}, \bar{y}) = (t_p - \bar{t})(\bar{x}, \bar{y})(\nu_x, \nu_y) \tag{5.6}$$

  where t_p is the present time when the center position is being updated.

- The line length, defined the distance between the two outermost tips of the line.
- A unique ID to the kernel to help tracking lines.

Each of the uniquely identified line segments are then submitted at each time step to line tracking scheme to monitor their motion.

5.3.4 Line Tracking

Line segments are created to maintain the detected lines until they disappear from the scene. Doing so will prevent unnecessary computation to re-cluster triggered events ensure that the line does not get falsely deleted. Also, event-based cameras are designed (benefiting from their quasi-continuous nature) to address the issue of tracking triggered

events which are created as a flow of sufficiently close events and facilitates clustering and rejecting old events. When a kernel is upgraded to a line, the triangle area test is performed to ensure that the event-to-line-adjacency is maintained (see Equation 5.2). A rigid body can create events that move with varying optical flow when rotating or roughly constant optical flow when translating. Hence, the incorporation of event's optical flow into the admission criteria for lines. New events optical flow should be bounded within the mean and variance of the existing line's principal optical flow $\mu_\nu - 2\sigma_\nu < \nu < \mu_\nu + 2\sigma_\nu$. This condition prevents new events from joining the line and causing the false line stitching. To minimize the effects of events sparsity on the optical flow estimation, optical flow denoising is performed. We apply a simple first-order filter [Ellis, 2012] using the event optical flow and the line principal optical flow:

$$(\nu_{x_{new}}, \nu_{y_{new}}) = \delta\mu_{\nu_{(x,y)}} + (1-\delta)(\nu_{x_{old}}, \nu_{y_{old}}) \tag{5.7}$$

where δ is adaptively chosen as a function of the mean and variance of lines optical flow[4]:

$$\delta = \delta_1 + \frac{\sigma_\nu^2}{\mu_\nu}(\delta_2 - \delta_1) \tag{5.8}$$

Here we have δ_1 and δ_2 as the minimum and maximum bounds of δ respectively. Instead of recalculating the optical flow mean and variance of all events of the line segment we use the recursive mean rule to avoid high computational time as:

$$\mu_n = \frac{n-1}{n}\mu_{n-1} + \frac{1}{n}x_n \tag{5.9}$$

$$\sigma_n^2 = \frac{n-1}{n}\sigma_{n-1} + \frac{1}{n-1}(x_n^2 - \mu_n^2) \tag{5.10}$$

μ_n and σ_n^2 are then used to re-estimate lines principal optical flow, slope and center. As part of lines update process, the number of events that each line can have is set to a maximum number n_{max}. This ensures that the most recent events are kept in the line

[4]The ratio between the mean and variance for translational motion would be relatively smaller than the ratio for rotational motion

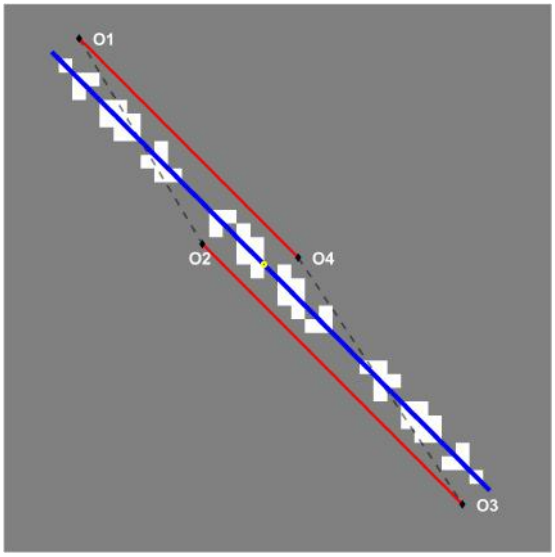


FIGURE 5.3: Two lines created of the same physical line are stitched, the area contained by their four vertices O1,O2,O3 and O4 (in red) and the resulting stitched line (in blue)

segment. This approach also ensures that the buffer containing the line's events does not overflow. It helps in enhancing the estimation of the line parameters: if a tip is removed from the list of updated events, its length can shrink or expand during the process. The previous event saved in the tips buffer will be considered the new tip. Lines may be upgraded early before they reach their actual length which may lead to many small broken lines. Hence the final step of our line detector is to ensure correct stitching of adjacent detected lines that belong to the same actual line.

5.3.5 Line Stitching and Global Restoration

After assigning an events packet to a line segment, it is possible that one physical line may create multiple smaller adjacent or duplicated lines in the environment. We then validate the results of the process by merging the duplicated lines and ensuring that the ones that are discontinued are correctly stitched and identified. The lines segments that are created by the same physical boundaries share the same characteristics, such as the slope and the principal optical flow, while being close enough to each other. Therefore, in order to merge these lines, we first need to check if they have the same characteristics. If

the cross-product of lines vector components is less than δ_s and the ratios between optical flow direction and magnitude are less than θ_ν and r_ν respectively, we consider these two lines mergeable and pass them to a final adjacency check: similar to the event's adjacency check, we avoid expensive search algorithms by considering that the tips of any two lines construct (in the general case) an irregular quadrilateral (see Figure 5.3). The area of an irregular quadrilateral that has sufficiently close sides will be proportional to its longest side. Hence, we check if the quadrilateral area constructed by any two detected lines does not exceed a threshold proportional to the longest line's length. We compute the area using the coordinates of the quadrilateral edges (namely the two lines tips) as follows:

$$A = x_1y_2 + x_2y_3 + x_3y_4 + x_4y_1 \tag{5.11}$$

$$B = x_2y_1 + x_3y_2 + x_4y_3 + x_1y_4 \tag{5.12}$$

$$\square_{area} = \frac{A + B}{2} \tag{5.13}$$

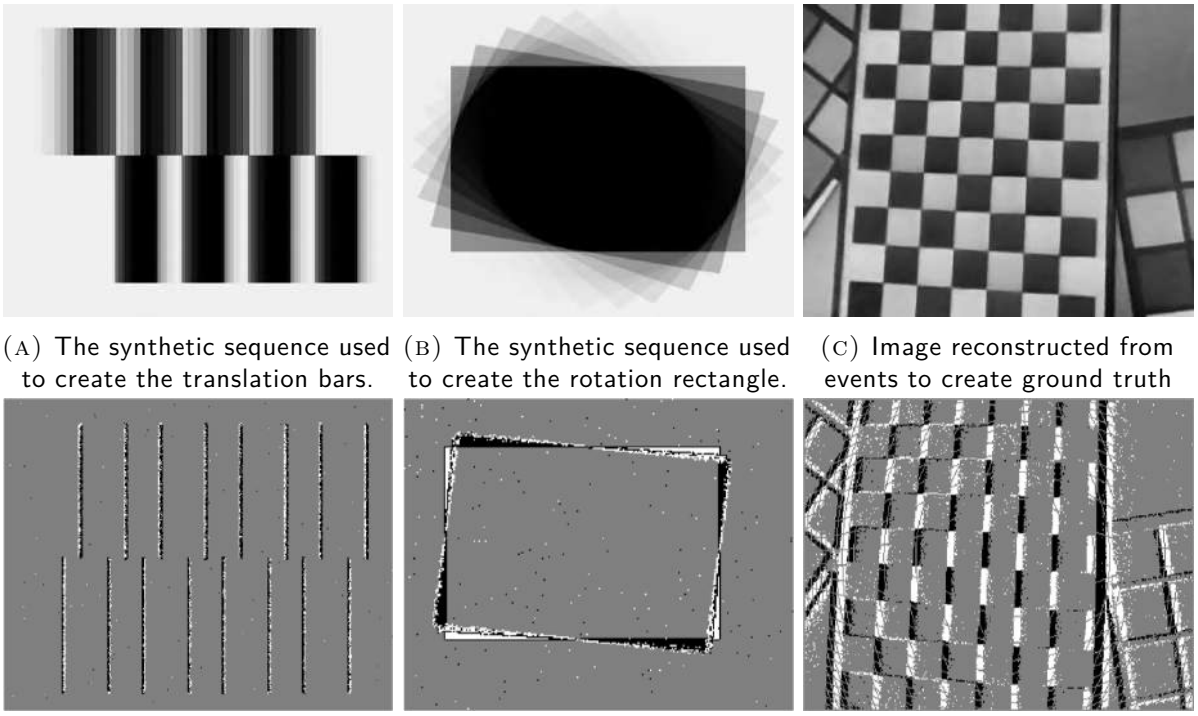
where $\Big\{(x_i, y_i) : i \in \{1, 2, 3, 4\}\Big\}$ are the lines tips coordinates arranged in order to create a sequence and $\square_{area}$ is the area of the quadrilateral. If two lines have an area less than a ratio ϕ of the longest line, then these two lines are stitched and considered one line. The four chains of lines tips are connected to create only two with the two outermost chains to the two innermost ones and then recompute the line parameters after deleting duplicate events. The newly stitched line takes the ID of the longest line to maintain better tracking quality (see Algorithm 4 Lines 37:43).

Finally, to avoid false data association or memory saturation, a check is performed to ensure the recency of all saved data after each time interval δ_s. If the lines and kernels in the scene contain too old events (the sum of its timestamp and lifetime is less than the present time), they are considered obsolete and get deleted from the memory. Events in the nuclei buffer which contains unclustered events that have been created δ_b milliseconds earlier are also considered obsolete and are deleted from the buffer.

Algorithm 4 Line Detection and Segmentation

Input: $e\{x, y, t, p\}_{i=1}^{N}$
Output: $e\{x, y, t, p, \nu_x, \nu_y, t_l\}_{i=1}^{N}$, $Lines\{l, c, v_l, \nu_l\}_{i=1}^{M}$

Initialize: $Lines$, $Kernels$, $Buffer$, $position$
$t_p = \delta_p$, $t_s = \delta_s$
for $i \leftarrow 1 : N$ **do** ▷ Events acquisition loop
    $e\{x, y, t, p, \nu_x, \nu_y, t_l\}_i$ =PCAFlow($e\{x, y, t, p\}_i$) ▷ see Algorithm 2
    $position$=updatePosition(e_i)
    if $t > t_m$ **then**
        e_packet=active events in $position$
        for $j \leftarrow 1 :$ #events in e_packet **do**
            (f,$line$)=verifyConnection(e_j,$Lines$)
            if f **then**
                $line$=updateParametersL($line, e_j$)
                $Lines.update(line)$
            else
                (f,$kernel$)=verifyConnection(e_j,$Kernels$)
                if f **then**
                    $kernel$=updateParametersK($kernel$)
                    if $n > n_k$ & $l > l_k$ **then**
                        $Lines.insert(kernel)$
                    else
                        $Kernels.update(kernel)$
                    end if
                else
                    for $k \leftarrow 1 :$ #events in $Buffer$ **do**
                        $dist$=distance(e_j,e_k)
                        if $dist < \sigma_l$ **then**
                            $Kernels.insert(\{e_j, e_k\})$
                        else
                            $Buffer.insert(e_j)$
                        end if
                    end for
                end if
            end if
        end for
        t_p=t_p+δ_p
    end if
end for
if $t > t_s$ **then**
    $Lines$=stitchLines($Lines$)
    deleteOldLines($Lines$)
    deleteOldKernels($Kernels$)
    deleteOldEvents($Buffer$)
    t_S=t_s+δ_s
end if



(A) The synthetic sequence used to create the translation bars.

(B) The synthetic sequence used to create the rotation rectangle.

(C) Image reconstructed from events to create ground truth

(D) Events created by the translating bars

(E) Events created by the rotating rectangle

(F) Events created by the moving checkerboard

FIGURE 5.4: Lines detection and tracking test sequences

5.4 Experimental Setup

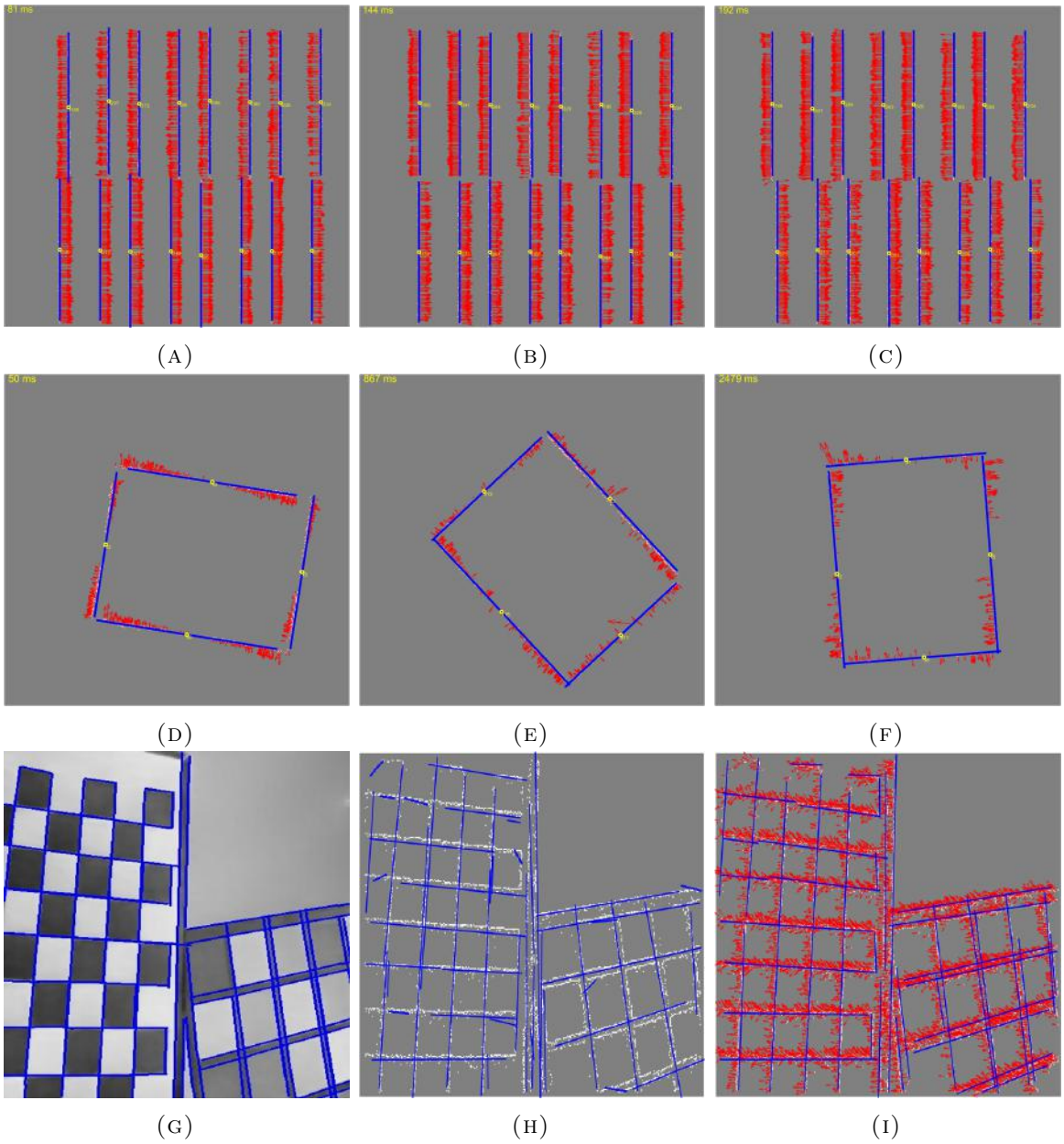
To test the performance of our event-based line detection and tracking system, we needed to build a set of sequences providing the ground truth for the lines in the environment. The event-based benchmarking sequence we created (see chapter 3) only provides the estimation of optical flow and 6-DOF pose. Therefore, to make sure our algorithm performs robustly in various cases, we created the ground truth with several synthetic and natural scenarios. The synthetic scenarios feature two sequences where patterns are created using Matlab: One illustrating pure rotational and the other pure translational motion displayed on a screen recorded using a 480×360 neuromorphic sensor [Posch et al., 2010a]. The first sequence of synthetic data consists of a black rectangle that continuously changes its rotational speed. The second synthetic sequence set contains two rows of vertical bars moving horizontally in opposite directions while touching each other. For the natural scenario, we used the checkerboard sequence of handheld random camera motion from the

DVSMOTION20 dataset [Almatrafi et al., 2020] featuring a large number of lines. For the checkerboard sequence, in order build a ground truth, we need to detect lines using well-known state-of-the-art algorithms such as the standard pair of Canny edge detector [Canny, 1986] / Hough transform [Duda and Hart, 1972] which requires full frame images. To create frames, the ground truth frames are generated by passing the recorded sequences to the E2VID neural network [Rebecq et al., 2019]. To have maximum distinction between edges, the sequence images are then thresholded to maximally distinguish edges and lines in the environment[5]. Next, the standard pair of Canny edge detector/Hough transform is applied to the thresholded images to detect lines. To ensure no duplicated line is detected, the events generated between images are then used to consume the events around the lines, which eliminates the duplicated lines that did not consume any event. A ground truth optical flow (provided with the sequence) is used in order to interpolate detected lines between images. The timestamps of detected lines can then match the corresponding time steps in the algorithms. Tracking lines consists in matching each line with a line in the upcoming timestamp based on their midpoint (with a 2 pixels tolerance), length and angle. The ground truth of the first two scenarios does not require any previous steps. Since no lines have changed in length nor have been occluded, the tracking line does not need to be reconstructed.

5.5 Results

To verify the reliability and correctness of our algorithm, we performed a comparison between the various parameters (Length, tracking and computational time) of our algorithm and the ELiSeD [Brändli et al., 2016] algorithm. We tested our algorithm on a QuadCore i7 1.9 GHz machine where most of the executed code consists in C++ (computations including optical flow estimation, kernel update, line parameters estimation and stitching) with some parts written with MATLAB (data handling, buffers restoration and results display).

[5] Benefiting from the high contrast of the checkerboard



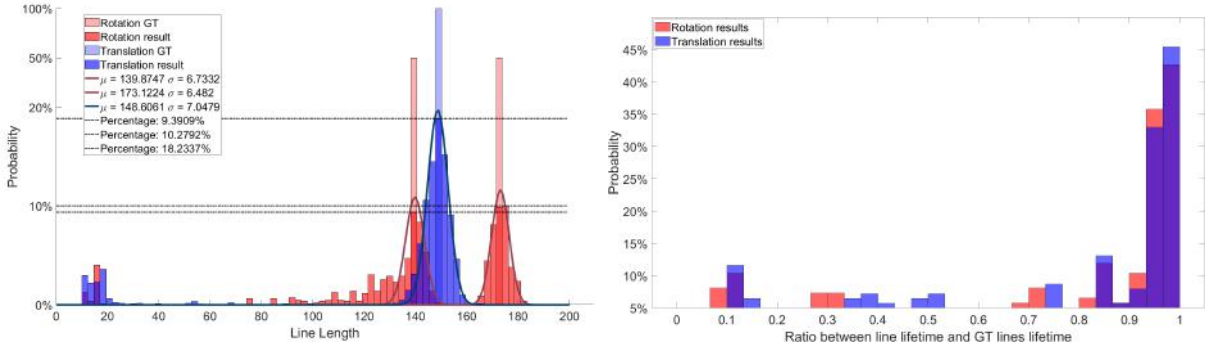
(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

FIGURE 5.5: Top row: results of three instances of two groups of eight lines touching each other and moving in opposite directions with no false stitching. Middle row: three instances of a rectangle rotating clock-wise with a varying optical flow around each line. Bottom row: (from left to right) Ground truth obtained using the standard Canny / Hough algorithm on the checkerboard sequence, results of the ELiSeD algorithm and results of our algorithm.

No optimization or explicit parallelization has been applied. Besides the comparison of lines' length, lifetime and computational time to assess our algorithm, estimated lines position is shown also in Figure 5.5.



(A) Translation and rotation scenarios length distribution

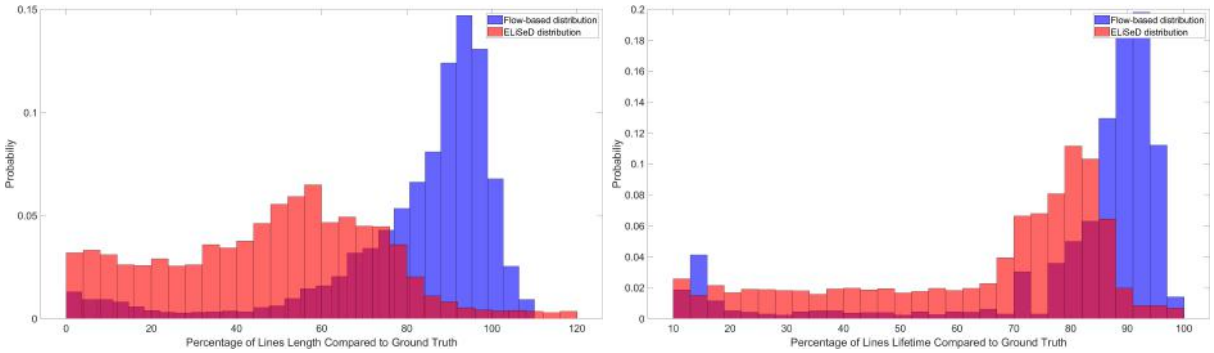
(B) Translation and rotation scenarios lifetime distribution

FIGURE 5.6: Synthetic scenarios distributions.

5.5.1 Length of Detected Lines

Correctly estimating the detected line length is essential to ensure the correct line is detected and tracked with no false stitching. All lines have the same length for the synthetic translation scenario, and the synthetic rotation scenario has two distinctive line lengths. At the beginning, small portion of lines are being detected before being correctly stitched. This approach results in the distribution of lines' small parts being visible. Aside from having these short lines, our algorithm can also correctly estimate the length of the lines where the peak of the distribution matches the actual length (see Figures 5.6a and 5.7a).

Because lines are tracked in event-based nature by registering newly triggered events to already detected lines, falsely stitched lines may cause critical problems. The challenge in the translation scenario was to ensure that the lines were properly stitched since neither the slope nor the optical flow of the lines changed much. Based on optical flow magnitude and direction to distinguish each line, most of the detected lines had the length normally distributed with $\mu = 148.61$ pixel and $\sigma = 7.08$ pixel with no false stitching (see Figure 5.6a)



(A) Estimated lines length distribution (B) Estimated lines lifetime distribution

FIGURE 5.7: Distribution of the percentage of estimated lines length and lifetime matched to ground truth of our algorithm (in blue) and ELiSeD algorithm (in red) of the Checkerboard sequence.

For a rotational scenario, the quality of lines detection is expected to be less satisfying because the slope and optical flow vary at each point of the line. In such case, an event may not be correctly assigned to the right cluster. However, the length distribution of the detected lines features a distribution with two distinctive peaks, each of which can be approximated to normal distribution with $\mu = 139.704$, $\sigma = 6.756$ and $\mu = 173.004$, $\sigma = 6.369$ where the expected lengths were 141 and 174 pixels respectively (see Figure 5.6a).

The checkerboard sequence being a real life motion scenario, lines lengths and orientations may vary if occluded or left the scene gradually. We calculated the distribution of correctly estimated length as the ratio $\frac{detected\ line\ length}{matched\ ground{-}truth\ line\ length}$ where a detected line is matched to a line in the ground-truth if the midpoint distance and line orientation do not exceed an acceptable threshold. The distribution of our method is a skewed distribution of low variance peaked between 80 and 105% of the ground-truth length. ELiSeD distribution has a more flattened distribution with a small peak around 60% due to the existence of many broken lines (see Figure 5.7a)

5.5.2 Amount of Detected Lines

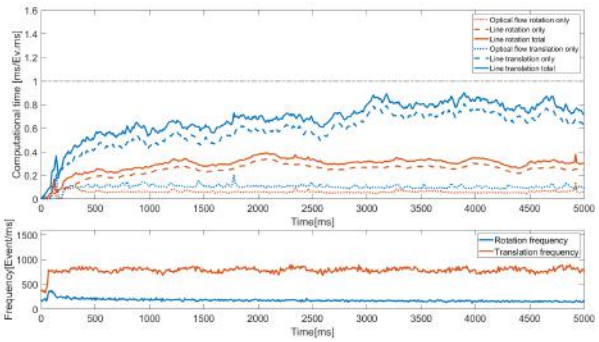
In order to properly track the lines in a scene, it is important that the number of detected lines matches the actual amount of lines and that no duplicate lines are created. When a kernel collects several events that exceed the threshold, it will be upgraded to a short line in the scene, and these will then be stitched to the correct line. This process ensures that the number of detected lines will increase slightly, but then decrease again once the stability of tracking has reached its limit (see Figure 5.8c). The number of lines for the checkerboard sequence saturates slightly above the number of lines detected in the ground truth. Unlikely, ELiSeD algorithm shows a higher number of detected lines since not all detected lines are stitched correctly(see Figure 5.8d).

5.5.3 Lines Tracking

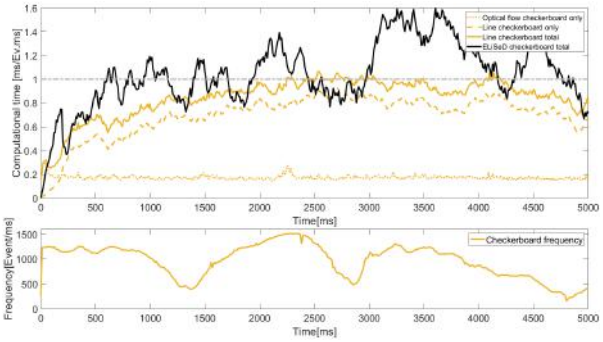
Line segments detection in a scene is very helpful for robotic applications but only if we can keep track of the detected lines. To assess the ability of our algorithm to track lines, we compare the lifetime of the detected line with the lifetime of its matched line in the ground truth. In the rotation and translation sequences, all lines were clearly visible at all time. This suggests that the lines lifetime is equal to the sequence time. Most of the lines detected by our algorithm were apparent in the scene during runtime (see Figure 5.6b). Some lines have a shorter lifetime due to the appearance of small line segments after the upgrade process before being stitched. For the checkerboard sequence, our algorithm kept accurate tracking where the lifetime of most of the detected lines exceeded 90% of the ground truth lines lifetime, while EliSeD algorithm features a lifetime distribution around 75% (see Figure 5.7b) with much more line disappearing at lower lifetime.

5.5.4 Computational Power

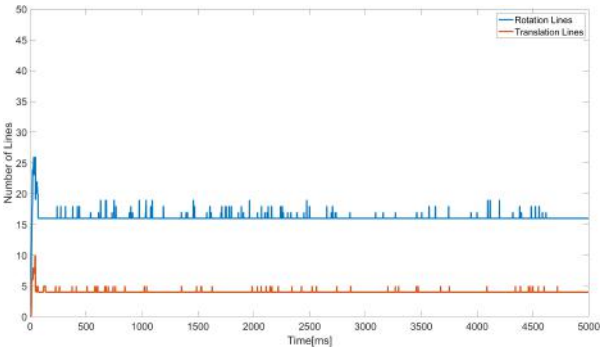
Since event-based cameras work asynchronously, the computational power required to achieve real-time processing will always depend on the number of events emitted per



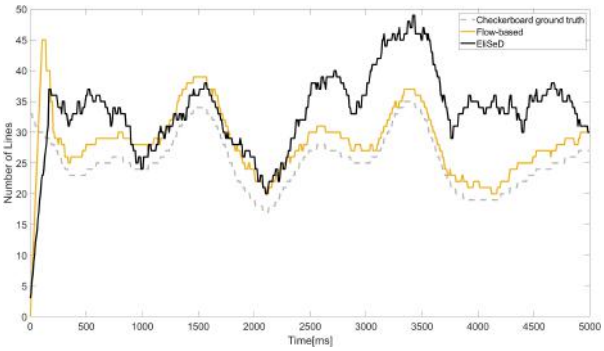
(A) Translation and rotation sequences optical flow(dotted), lines estimation (dashed) and total computational time (continuous) of our flow-based algorithm



(B) checkerboard sequences optical flow (dotted), lines estimation (dashed) and total computational time (continuous) and ELiSeD computational time (continuous black)



(C) Number of estimated lines of the rotation (red) and translation (blue) sequences of our flow-based algorithm



(D) Number of estimated lines of the checkerboard sequence of our flow-based algorithm (yellow), ELiSeD algorithm (black) and the lines found in the ground truth (grey)

FIGURE 5.8: Results for computational time parameters. (a) Computational time and events frequency, (c) number of lines in the scene for the rotational and translational sequences. (b) Computational time and events frequency, (d) number of lines in the scene for the checkerboard sequences of our algorithm, ELiSeD and the ground truth.

time step. To make a reasonable judgment on how fast our algorithm can work, Figures 5.8a and 5.8b show the frequency of events provided by the camera in all scenarios, and Figures 5.8c and 5.8d show how many lines are found in each scene, as these are the two main parameters that directly affect computation time. The computational time in translation and rotation scenarios remained below acquisition time with a peak value of around $0.9ms$ for each $1ms$ events packet for the translation and rotation sequences (see Figure 5.8a). The event frequency of the translational scenario is about 900 events/ms which is lower than the frequency in most ordinary scenes. For the checkerboard sequence, although the events frequency and the number of lines are much higher than in the other

sequences, our algorithm attained real-time computation except in minimal time intervals (around 80 milliseconds) by a slightly extra amount of time (about 50 microseconds for each millisecond) (see Figure 5.8b). The number of lines in the scene and the frequency of events can each be set to a maximum that cannot be exceeded to ensure real-time applicability.

5.6 Conclusion

In this chapter, we introduced a comprehensive scheme for line detection and segmentation that takes into account the various characteristics of a scene. We used simple heuristics to avoid costly implementations and complex designs. As a result, we managed to acquire adequately competitive results that provide accurate line detection and segmentation. Our algorithm performed correctly in different cases like rotating lines or lines coinciding with each other while moving in opposite directions, and no false line detection was reported. The computational capabilities of our algorithm can allow it to perform well in various scenarios. One of the main advantages of our algorithm is that it can detect lines and enhance optical flow quality by rejecting extreme values. It can also improve the accuracy of optical flow by providing a consensus flow for each line. The ability to perform segmentation and line detection in event-based systems can be beneficial for various tasks, such as data association and visual-inertial odometry. In data association, for instance, the segmentation and line detection can help improve the system efficiency by identifying and grouping events that are related to the data. Having all the elements put together, we show, in the next chapter, how we employ them to introduce our Visual-inertial scheme.

6 Flow-Based Neuromorphic visual Inertial Odometry

Chapter abstract

In this chapter, we present our flow-based visual-inertial odometry scheme and validate it using different scenarios. To our knowledge, the algorithm introduced in this chapter is the first event-based visual inertial algorithm which uses optical flow error for optimisation. We present a novel method for initialization exploiting geometric information about the environment obtained by lines detection. Our algorithm works without being restricted by key-frame initialisation or maximum events frequency.

6.1 Introduction

By providing frame-free asynchronous data, event-based cameras are designed to trigger events and react to changes in brightness in the scene whenever detected. These sensors are designed to mimic the activities of the biological retina and do not depend on any artificial clock signals. The asynchronous nature of event-based cameras enables them to suppress redundant data and provide high temporal resolution, high dynamic range with low power consumption. With the advantages of event-based cameras, these sensors provide a convenient replacement for frame-based vision sensors in scenarios where high

maneuverability occurs leading to high dynamic visual environment.

For the past decade, many solutions have been introduced to integrate event-based cameras in robotic applications: for instance, [Kim et al., 2008], [Mueggler et al., 2014], [Rebecq et al., 2016b] and [Mueggler et al., 2018] provide accurate motion estimation. Amongst the approaches adopted to solve this problem, probabilistic filtering methods have been introduced in [Kim et al., 2008], [Kim et al., 2016], [Weikersdorfer and Conradt, 2012] and [Weikersdorfer et al., 2013]. Other methods presented in [Mueggler et al., 2014], [Kueng et al., 2016] and [Weikersdorfer et al., 2014] used different optimisation schemes to benefit from their higher accuracy to provide motion estimation (see Section 2.5 "Page 35").

Event-based cameras' ability to provide asynchronous data with significantly low temporal resolution leads to better continuous representation compared to frame-based cameras, besides eliminating other problems such as motion blur and low dynamic range. Furthermore, this ability provides a more stable mathematical modeling of the brightness constancy condition, which describes the apparent pixels motion known as the optical flow. In this chapter, we introduce, to the extent of our knowledge, the first visual-inertial odometry algorithm that jointly optimizes the optical flow with the inertial measurements for neuromorphic vision sensors.

6.2 Related Work

In order to have accurate 6-DoF motion estimation, researchers adopt two main approaches. The first approach is using filtering based on Bayes theorem [Konrad and Dubois, 1992] which is used for standard cameras ([Weiss and Siegwart, 2011] and [Mourikis et al., 2007]

for example). This approach is adopted also for event-based cameras to provide accurate motion estimation. In one of the first trials to assess filtering approaches, [Weikersdorfer and Conradt, 2012] used a particle filter to estimate the rotation vector by stitching mosaic images of events. [Weikersdorfer et al., 2013] estimated 2D motion of event-camera using also a particle filter. To reduce the required computational time of particle filters, [Kim et al., 2016] adopted an extended Kalman filter to estimate 6-DoF motion and the scene gradient intensity. [Zihao Zhu et al., 2017] aided an event-based camera with an IMU to have accurate scale factor estimation. The IMU measurements is fused with the camera output in an extended Kalman filter scheme to have 6-DoF motion estimation.

To alleviate the constraints of filtering approach such that the condition of linearity should not be violated, the approach of non-linear unconstrained optimization is introduced. [Mueggler et al., 2014] used non-linear optimisation to estimate 6-DoF pose of a quadcopter that undergoes aggressive maneuvers which would be difficult to estimate using a linear estimator like the Kalman filter. Using keyframes, [Rebecq et al., 2016b] estimate a 3D map to augment the accuracy of the estimated 6-DoF pose. [Vidal et al., 2018] augmented the event-based camera with a CMOS and depth sensors and jointly optimized the measurments of each sensor to provide highly accurate 6-DoF pose estimation (see Section 2.5 "Page 35").

The filtering-based approaches have leverages the computational constraint and runs faster than optimization-based ones whilst the optimization-based approaches achieve excellent localization accuracy and require less memory [Chen et al., 2018]. In our approach, we adopt non-linear unconstrained optimization scheme for motion estimation for its higher accuracy and propose solutions to reduce the computational requirements.

6.3 Flow-Based Visual-Inertial Odometry

6.3.1 Preliminaries

6-DoF Pose

In this section, we show how we exploit optical flow information and inertial measurements to incrementally estimate a 6-DOF pose $\boldsymbol{T} \in SE(3)$ defined as:

$$\boldsymbol{T}_{ij} = \begin{bmatrix} R_{ij} & \boldsymbol{t}_{ij} \\ \mathbf{0} & 1 \end{bmatrix} \tag{6.1}$$

Where $R_{ij} \in SO(3)$ is the rotation matrix and $\mathbf{t}_{ij} \in \mathbb{R}^3$ is the translation vector. A rigid body transformation $\boldsymbol{T}_{ij}$ expressed as a Lie group $\mathcal{L}$ differentiable on manifold with the Lie algebra $\mathcal{A}$ as its tangent space at the identity called twist $\boldsymbol{\zeta}_{ij}$:

$$\boldsymbol{\zeta}_{ij} = \begin{bmatrix} \lfloor \boldsymbol{\Omega}_{ij} \rfloor_{\times} & \boldsymbol{V}_{ij} \\ \mathbf{0} & 1 \end{bmatrix} \tag{6.2}$$

where $\lfloor \boldsymbol{\Omega}_{ij} \rfloor_{\times} \in \mathfrak{so}(3)$ is the skew symmetric matrix of the angular velocity vector and $\boldsymbol{V}_{ij} \in \mathbb{R}^3$ is the linear velocity vector. The logarithmic map $Log(.) : \mathcal{L} \rightarrow \mathcal{A}$ is used to obtain the twist $\boldsymbol{\zeta}_{ij}$ of $\boldsymbol{T}_{ij}$ at the identity space and its inverse can be found using the exponential map $Exp(.) : \mathcal{A} \rightarrow \mathcal{L}$ [Chirikjian, 2011]. The vector space representing the rigid body transformation (group and algebra) is represented by the vee sign $.^{\vee}$ and is reversed by the hat sign $.^{\wedge}$ as:

$$.^{\vee} : \mathcal{L}, \mathcal{A} \rightarrow \mathbb{R}^d \quad , \quad .^{\wedge} : \mathbb{R}^d \rightarrow \mathcal{L}, \mathcal{A} \tag{6.3}$$

$$\boldsymbol{T}_{ij}^{\vee} = \mathcal{P}_{ij} \quad , \quad \mathcal{P}_{ij}^{\wedge} = \boldsymbol{T}_{ij} \tag{6.4}$$

$$\boldsymbol{\zeta}_{ij}^{\vee} = \mathcal{V}_{ij} \quad , \quad \mathcal{V}_{ij}^{\wedge} = \boldsymbol{\zeta}_{ij} \tag{6.5}$$

where $\mathcal{P}_{ij} = [\Delta x_{ij}, \Delta y_{ij}, \Delta z_{ij}, \Delta\phi_{ij}, \Delta\theta_{ij}, \Delta\psi_{ij}]$ is the incremental pose vector and $\mathcal{V}_{ij} = [v_{xij}, v_{yij}, v_{zij}, \omega_{xij}, \omega_{yij}, \omega_{zij}]$ is the incremental velocity vector between the time steps i and j respectively.

Pinhole Model

Event-based cameras uses the pinhole model [Cyganek and Siebert, 2011] (or any projection model according to the used lens) to describe the 3D/2D projection $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ of any 3D point $\boldsymbol{X}_c = [X_c, Y_c, Z_c]^T \in \mathbb{R}^3$ in the camera frame to a 2D point $\boldsymbol{x}_c = [x_c, y_c]^T \in \mathbb{R}^2$ on the image plane as:

$$\pi\left(\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}\right) = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} f_u \frac{X_c}{Z_c} + c_u \\ f_v \frac{Y_c}{Z_c} + c_v \end{bmatrix} \tag{6.6}$$

where (f_u, f_v) are the lens focal length values and (c_u, c_v) are the the principal point coordinates in x and y directions respectively. The pinhole model is a planar model which requires each pixel (event in our case) to be undistorted for accurate 3D/2D reprojection.

Optical Flow Representation

Optical flow describes the pixels apparent motion [Longuet-Higgins and Prazdny, 1980] which can be approximated as the perspective projection of a 3D point X_c moving freely with linear velocity $\boldsymbol{V}_c = [v_{xc}, v_{yc}, v_{zc}]^T$ and angular velocity $\boldsymbol{\Omega}_c = [\omega_{xc}, \omega_{yc}, \omega_{zc}]^T$ so that the point's 3D velocity is described as:

$$\dot{\boldsymbol{X}}_c = -(\boldsymbol{\Omega}_c \times \boldsymbol{X}_c + \boldsymbol{V}_c) = \begin{bmatrix} \dot{X}_c \\ \dot{Y}_c \\ \dot{Z}_c \end{bmatrix} = -\left(\begin{bmatrix} \omega_{yc} Z_c - Y_c \omega_{zc} \\ \omega_{zc} X_c - Z_c \omega_{xc} \\ \omega_{xc} Y_c - X_c \omega_{yc} \end{bmatrix} + \begin{bmatrix} v_{xc} \\ v_{yc} \\ v_{zc} \end{bmatrix}\right) \tag{6.7}$$

2D point velocities (optical flow approximation) corresponding to the optical flow can be obtained by the derivative of equation (6.6) incorporating (6.7):

$$\dot{\boldsymbol{x}}_c = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{\dot{\boldsymbol{X}}_c}{Z_c} - \frac{\dot{Z}_c}{Z_c}\boldsymbol{x}_c = \frac{1}{Z_c}\boldsymbol{A}(x_c, y_c)\mathbf{V}_c + \boldsymbol{B}(x_c, y_c)\boldsymbol{\Omega}_c \tag{6.8}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are:

$$\boldsymbol{A} = \begin{bmatrix} -f & 0 & (x_c - c_u) \\ 0 & -f & (y_c - c_v) \end{bmatrix} \tag{6.9}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{(x_c - c_u)(y_c - c_v)}{f} & -\left(f + \frac{(x_c - c_u)^2}{f}\right) & (y_c - c_v) \\ \left(f + \frac{(y_c - c_v)^2}{f}\right) & -\frac{(x_c - c_u)(y_c - c_v)}{f} & -(x_c - c_u) \end{bmatrix} \tag{6.10}$$

Hence, estimating the optical flow, if the velocity vector $\boldsymbol{\zeta}_c^{\vee} = [v_{xc}, v_{yc}, v_{zc}, \omega_{xc}, \omega_{yc}, \omega_{zc}]$ is known, would also require knowledge about the depth Z_c of each point.

IMU preintegration measurements

An inertial measurement unit provides proprioceptive information as the linear acceleration $\tilde{\mathbf{a}}_b(t)$ and angular velocity $\tilde{\boldsymbol{\Omega}}_b(t)$ expressed in the body frame and influenced by different noise sources described as:

$$\tilde{\boldsymbol{\Omega}}_b(t) = \boldsymbol{\Omega}_b(t) + \mathbf{b}_g(t) + \boldsymbol{\eta}_g(t) \tag{6.11}$$

$$\tilde{\mathbf{a}}_b(t) = \mathbf{a}_b(t)) + R_{wb}^T\mathbf{g} + \mathbf{b}_a(t) + \boldsymbol{\eta}_a(t) \tag{6.12}$$

where $\boldsymbol{\eta}_g(t)$ and $\boldsymbol{\eta}_g(t)$ are the Gaussian white noise characterised as $\mathcal{N}(0, \sigma_g)$ and $\mathcal{N}(0, \sigma_a)$ respectively. $\boldsymbol{\Omega}_b(t)$ and $\mathbf{a}_b(t)$ are the actual angular velocity and linear acceleration of the IMU, R_{wb} is the transformation matrix between the body frame and the world frame and $\mathbf{g}$ is the gravity vector. $\mathbf{b}_g(t)$ and $\mathbf{b}_a(t)$ are the slowly varying random walk noise of the sensors:

$$\dot{\mathbf{b}}_g(t) = \boldsymbol{\eta}_{bg} \quad , \quad \dot{\mathbf{b}}_a(t) = \boldsymbol{\eta}_{ba} \tag{6.13}$$

Estimating the states of motion from an instant i to the instant j is done by integrating the linear acceleration and angular velocity:

$$R_{wb}(t_j) = R_{wb}(t_i)Exp\left(\int_{t_i}^{t_j}(\tilde{\mathbf{\Omega}}(\tau) - \mathbf{b}_g(\tau) - \boldsymbol{\eta}_g(\tau))d\tau\right) \tag{6.14}$$

$$\boldsymbol{V}_b(t_j) = \boldsymbol{V}_b(t_i) + \int_{t_i}^{t_j}(R_{wb}(\tilde{\mathbf{a}}(\tau) - \mathbf{b}_a(\tau) - \boldsymbol{\eta}_a(\tau)) - \mathbf{g})\,d\tau \tag{6.15}$$

$$\boldsymbol{P}_b(t_j) = \boldsymbol{P}_b(t_i) + \boldsymbol{V}_b(t_{ij})\Delta t + \int\int_{t_i}^{t_j}(R_{wb}(\tilde{\mathbf{a}}(\tau) - \mathbf{b}_a(\tau) - \boldsymbol{\eta}_a(\tau)) - \mathbf{g})\,d\tau^2 \tag{6.16}$$

In this chapter, we choose to work with preintegration representation of IMU measurements introduced in [Lupton and Sukkarieh, 2011] and modified for representation on manifolds in [Forster et al., 2016] to avoid recomputation of parameters that may increase errors propagation. The increments of motion states in discrete time assuming that the bias is constant between two time steps are derived as:

$$\begin{aligned}\Delta R_{wb}(t_{ij}) &\simeq \prod_{k=i}^{j-1}\left(Exp((\tilde{\mathbf{\Omega}}(t_k) - \mathbf{b}_{gi})\Delta t_{kk+1})Exp(-\mathbf{J}_{rk}\boldsymbol{\eta}_{gk}\Delta t_{ik}\right)\\ &= \Delta\tilde{R}_{wb}(t_{ij})\prod_{k=i}^{j-1}Exp(-\tilde{R}_{wb}^T(t_{k+1j})\mathbf{J}_{rk}\boldsymbol{\eta}_{gk}\Delta t_{ik})\\ &= \Delta\tilde{R}_{wb}(t_{ij})Exp(-\delta\phi_{ij})\end{aligned} \tag{6.17}$$

$$\begin{aligned}\Delta\boldsymbol{V}_b(t_{ij}) &\simeq \sum_{k=i}^{j-1}(\tilde{R}_{wb}^T(t_{ik})(\mathbf{I} - \lfloor\delta\phi_{ik}\rfloor_\times)(\tilde{\mathbf{a}}_b(t_k) - \mathbf{b}_{ai}) - \tilde{R}_{wb}(t_{ij})\boldsymbol{\eta}_{ak})\Delta t_{kk+1}\\ &= \Delta\tilde{\boldsymbol{V}}_b(t_{ij}) + \sum_{k=i}^{j-1}(\tilde{R}_{wb}(t_{ik})\lfloor\tilde{\mathbf{a}}_b(t_k) - \mathbf{b}_{ai}\rfloor_\times - \tilde{R}_{wb}(t_{ik})\boldsymbol{\eta}_{ak})\Delta t_{kk+1}\\ &= \Delta\tilde{\boldsymbol{V}}_b(t_{ij}) - \delta\boldsymbol{V}_b(t_{ij})\end{aligned} \tag{6.18}$$

$$
\begin{aligned}
\Delta \boldsymbol{P}_b(t_{ij}) &\simeq \sum_{k=i}^{j-1} (\Delta \tilde{\boldsymbol{V}}_b(t_{ik}) - \delta \boldsymbol{V}_b(t_{ik})) \Delta t_{kk+1} \\
&+ \frac{1}{2} \left(\tilde{R}_{wb}(t_{ik}) (\mathbf{I} - \lfloor \delta \phi_{ik} \rfloor_\times) (\tilde{\mathbf{a}}_b(t_k) - \mathbf{b}_{ai}) - \tilde{R}_{wb}(t_{ik}) \boldsymbol{\eta}_{ak} \right) \Delta t_{kk+1}^2 \\
&= \Delta \tilde{\mathbf{P}}_b(t_{ij}) + \sum_{k=i}^{j-1} -\delta \mathbf{V}_b(t_{ik} \Delta t_{kk+1} \\
&+ \frac{1}{2} \left(\tilde{R}_{wb}(t_{ik}) \lfloor \tilde{\mathbf{a}}_b(t_k) - \mathbf{b}_{ai} \rfloor_\times - \tilde{R}_{wb}(t_{ik}) \boldsymbol{\eta}_{ak} \right) \Delta t_{kk+1}^2 \\
&= \Delta \tilde{\mathbf{P}}_b(t_{ij}) - \delta \mathbf{P}_b(t_{ij})
\end{aligned}
\tag{6.19}
$$

Where each state vector is defined as its mean value $\Delta(\tilde{.})$ to be estimated initially plus a perturbation value $\delta(.)$. $\mathbf{J}_{rk}$ is the right Jacobian of angular velocity defined as $\mathbf{J}_r(\tilde{\boldsymbol{\Omega}}(t_k)) - \mathbf{b}_{gi}) \Delta t$. To avoid recomputing the state estimates during the optimisation process, bias is not considered constant and is accounted for (see supplementary material of [Forster et al., 2016]).

6.3.2 Optimisation Scheme

Using the optical flow for accurate motion estimation is a complex problems which requires decoupling the translational and rotational motion and have a prior knowledge of depth in some cases ([Zucchelli, 2002], [Liu et al., 2017]). In a different way, our approach exploits the geometric characteristics of the environment besides augmenting the optical flow with IMU measurements to obtain accurate ego-motion and depth estimation (see Figure 6.1). We show in red the blocks representing the raw data used in our scheme, in blue, the blocks providing the processed data required for optimisation and in green the optimisation scheme and the initialization conditioning outputting 6DoF, twist and depth.

Event-Based Cameras provide signals due to change in the environment which would occur at contours of objects. This makes Event-Based sensors suitable for semi-dense SLAM and visual odometry algorithms. The high temporal resolution of Event-Based cameras puts into question the availability to use a flow-based visual-inertial optimisation scheme. In our scheme, we follow a probabilistic approach [Furgale et al., 2012] where

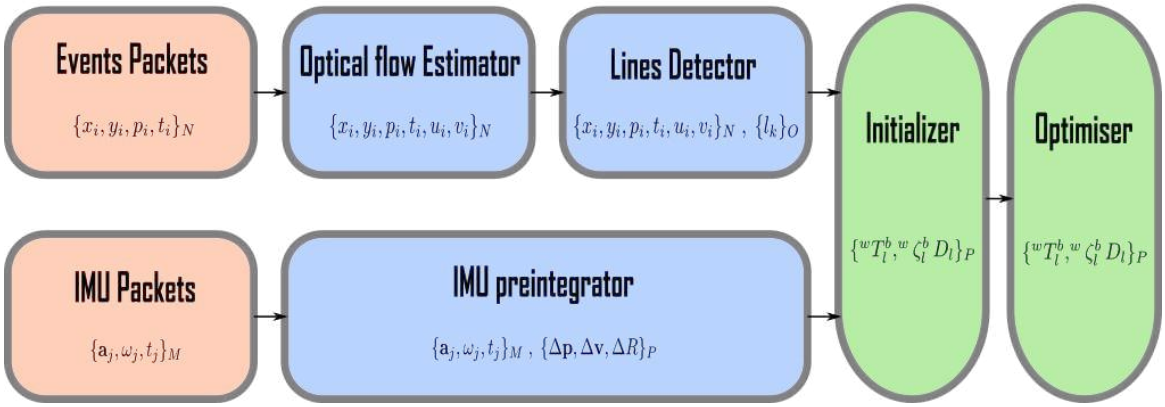


FIGURE 6.1: Our flow-based visual-inertial odometry scheme where each block shows its expected output and.

we try to get optimal state estimates $\mathcal{X}(t)$ within a time interval $[t_0, t_f]$. Using a set of measurements $\mathcal{Z}(t)$ where the environment has the structure $\mathcal{S}$ in a joint posterior estimate $p(\mathcal{X}(t)|\mathcal{Z}(t))$ where no map or prior belief are provided. The set of measurements consist of the measured optical flow given the position of each event $\mathcal{U}_m(t)$, the accelerometer measurements $\mathcal{A}(t)$ and gyroscope measurements $\mathcal{W}(t)$. With no prior belief, we try to find a maximum likelihood of measurements using the estimated states as:

$$p(\mathcal{U}_m(t), \mathcal{A}(t), \mathcal{W}(t)|\mathcal{X}(t)) = p(\mathcal{U}_m(t)|\mathcal{X}(t))p(\mathcal{A}(t)|\mathcal{X}(t))p(\mathcal{W}(t)|\mathcal{X}(t)) \tag{6.20}$$

where the conditional probability of Equation 6.20 consists of the multiplication of conditional probabilities of measurements given that each set of measurements is independent of the other. We assume that each conditional probability is described as a Gaussian probability distribution with zero mean and variance σ. Obtaining the maximum likelihood is equivalent to estimating the minimum of the log function which translates to minimizing the following cost function:

$$F = \frac{1}{N}\sum_{i=1}^{N}\Delta_{iu} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jimu} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jba} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jbw} \tag{6.21}$$

where N is the number of events providing optical flow during optimisation span and M

is the number of IMU measurements used. Δ_{iu}, Δ_{jimu} are the error terms corresponding to optical flow estimation and the IMU measurements respectively. Δ_{jba} and Δ_{jbw} are the error terms corresponding to the accelerometer and gyroscope bias. In order to enhance the optimisation process we added a twist error term responsible for refining the twist used for optical flow estimation. The new enhanced cost function is defined as follows:

$$F = \frac{1}{N}\sum_{i=1}^{N}\Delta_{iu} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jimu} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jba} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{jbw} + \frac{1}{M}\sum_{i=1}^{M}\Delta_{j\zeta} \tag{6.22}$$

The optical flow error Δ_{iu} is defined as:

$$\Delta_u = \rho\left((\mathbf{u}_e(t) - \mathbf{u}_m(d(t)))^T \mathbf{\Sigma}_u (\mathbf{u}_e(t) - \mathbf{u}_m(d(t)))\right) \tag{6.23}$$

where $\Sigma_u$is the covariance matrix associated with the optical flow. $\mathbf{u}_e(t)$ is the estimated optical flow that we obtained using PCA (see Chapter 4), $\mathbf{u}_m(d(t))$ is the measured optical flow using the IMU readings to obtain the twist vector $\boldsymbol{\zeta}_c^{\vee}$ (see Equation 6.8) where the initial estimate of depth d is shown in the Optimization Conditioning step. To alleviate the problem of estimating the depth of each event independently – which would require heavier computations – and since the provided events are created due to the motion of contours of objects, we assumed that the environment contains a sufficient amount of contour lines that can be used to estimate the depth of the used events. ρ is the Huber loss function defined as:

$$L(\Delta) = \begin{cases} \Delta^2 & \text{if } |\Delta| \leq \delta \\ \delta(|\Delta| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{6.24}$$

where δ is a predefined threshold designed to reject extreme outliers.

The IMU measurements error term Δ_{imu} is defined as:

$$\Delta_{imu} = \rho\left([\Delta_{Rij}^T, \Delta_{vij}^T, \Delta_{pij}^T]^T \boldsymbol{\Sigma}_{imu} ([\Delta_{Rij}^T, \Delta_{vij}^T, \Delta_{pij}^T]\right) \tag{6.25}$$

where Σ_{imu} is the IMU covariance matrix and the preintegration error terms are:

$$\Delta_{Rij} = Log\left(\left(\Delta\tilde{R}_{wb}(t_{ij}) Exp\left(\frac{\partial \tilde{R}_{wb}}{\partial \mathbf{b}_g}\partial \mathbf{b}_g\right)\right)^T R_{wb}(t_i)^T R_{wb}(t_j)\right) \tag{6.26}$$

$$\begin{aligned}\Delta_{vij} = & R_{wb}(t_i)\left(\boldsymbol{V}_b(t_j) - \boldsymbol{V}_b(t_i) - \mathbf{g}\Delta t_{ij}\right) \\ & - \left(\Delta\tilde{\boldsymbol{V}}_b(t_{ij}) + \frac{\partial \tilde{\boldsymbol{V}}_b}{\partial \mathbf{b}_a}\partial \mathbf{b}_a + \frac{\partial \tilde{\boldsymbol{V}}_b}{\partial \mathbf{b}_g}\partial \mathbf{b}_g\right)\end{aligned} \tag{6.27}$$

$$\begin{aligned}\Delta_{pij} = & R_{wb}(t_i)\left(\boldsymbol{P}_b(t_j) - \boldsymbol{P}_b(t_i) - \boldsymbol{V}(t_i)\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\right) \\ & - \left(\Delta\tilde{\boldsymbol{P}}_b(t_{ij}) + \frac{\partial \tilde{\boldsymbol{P}}_b}{\partial \mathbf{b}_a}\partial \mathbf{b}_a + \frac{\partial \tilde{\boldsymbol{P}}_b}{\partial \mathbf{b}_g}\partial \mathbf{b}_g\right)\end{aligned} \tag{6.28}$$

where the partial derivatives $[\frac{\partial \tilde{R}_{wb}}{\partial \mathbf{b}_g}, \frac{\partial \tilde{\mathbf{V}}_b}{\partial \mathbf{b}_a}, \frac{\partial \tilde{\mathbf{V}}_b}{\partial \mathbf{b}_g}, \frac{\partial \tilde{\mathbf{P}}_b}{\partial \mathbf{b}_a}, \frac{\partial \tilde{\mathbf{P}}_b}{\partial \mathbf{b}_a}]$ are to be calculated using the supplementary materials of [Forster et al., 2016].

Finally the error terms for the biases Δ_{ba} and Δ_{bw} are defined as:

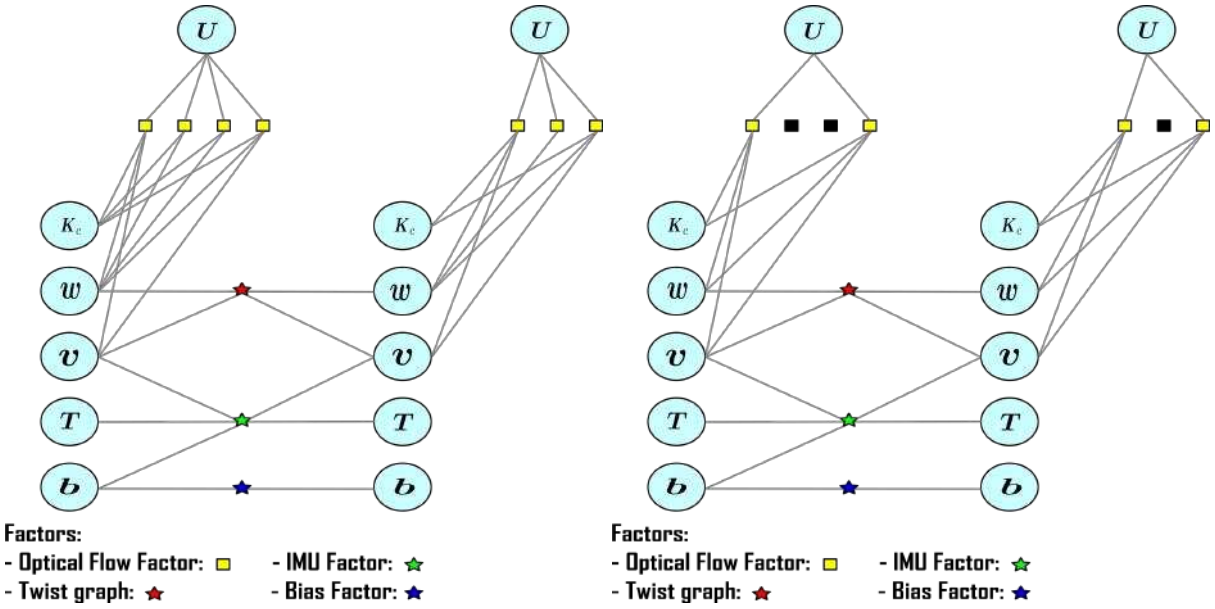
$$\Delta_{ba} = \rho\left((\mathbf{b}_{aj} - \mathbf{b}_{ai})^T \boldsymbol{\Sigma}_{ba} (\mathbf{b}_{aj} - \mathbf{b}_{ai})\right) \tag{6.29}$$

$$\Delta_{bw} = \rho\left((\mathbf{b}_{wj} - \mathbf{b}_{wi})^T \boldsymbol{\Sigma}_{bw} (\mathbf{b}_{wj} - \mathbf{b}_{wi})\right) \tag{6.30}$$

The twist error term:

$$\Delta_{\zeta ij} = \rho\left(\left(\left(\frac{1}{\Delta t}\hat{\boldsymbol{T}}_i^{-1}\hat{\boldsymbol{T}}_j\right) \ominus \boldsymbol{\zeta}_{ij}\right)^T \boldsymbol{\Sigma}_\zeta \left(\left(\frac{1}{\Delta t}\hat{\boldsymbol{T}}_i^{-1}\hat{\boldsymbol{T}}_j\right) \ominus \boldsymbol{\zeta}_{ij}\right)\right). \tag{6.31}$$

Frame-based optimisation schemes using features such as [Rebecq et al., 2017] choose certain key-frames to achieve triangulation with low uncertainty. Conversely, using optical flow allows to ignore key-frames and freely choose the time steps for optimisation



(A) (B)

FIGURE 6.2: A) Factor graph with no dropped events between two optimisation time steps, B) Factor graph where some events are dropped

depending on either the number of events N or the number of IMU readings M. Moreover, having rich events optical flow and lines ensure that we can drop events whenever events frequency exceeds a threshold in order to attain real-time proceseeing. Figure 6.2 depicts a factor graph showing how factors are connected to construct the cost function.

The state vector we optimise contains the position, rotation quaternion, velocity, IMU bias vectors and the camera intrinsic parameter (f_u, f_v, c_u, c_v) $\{\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{V}, \boldsymbol{b}_a, \boldsymbol{b}_g, K_c\}$. Our cost function is solved as a non-linear unconstrained leas square problem using Levenberg-Marquadrt method.

6.3.3 Optimization Conditioning

Being a nonlinear unconstrained optimisation problem, our scheme requires initial values that are close enough to the optimal values to ensure convergence(see Figure 6.1). Event-based cameras provide information about contours and since the lines are one of the repetitive geometric patterns in the indoor environments, we exploit the line detection

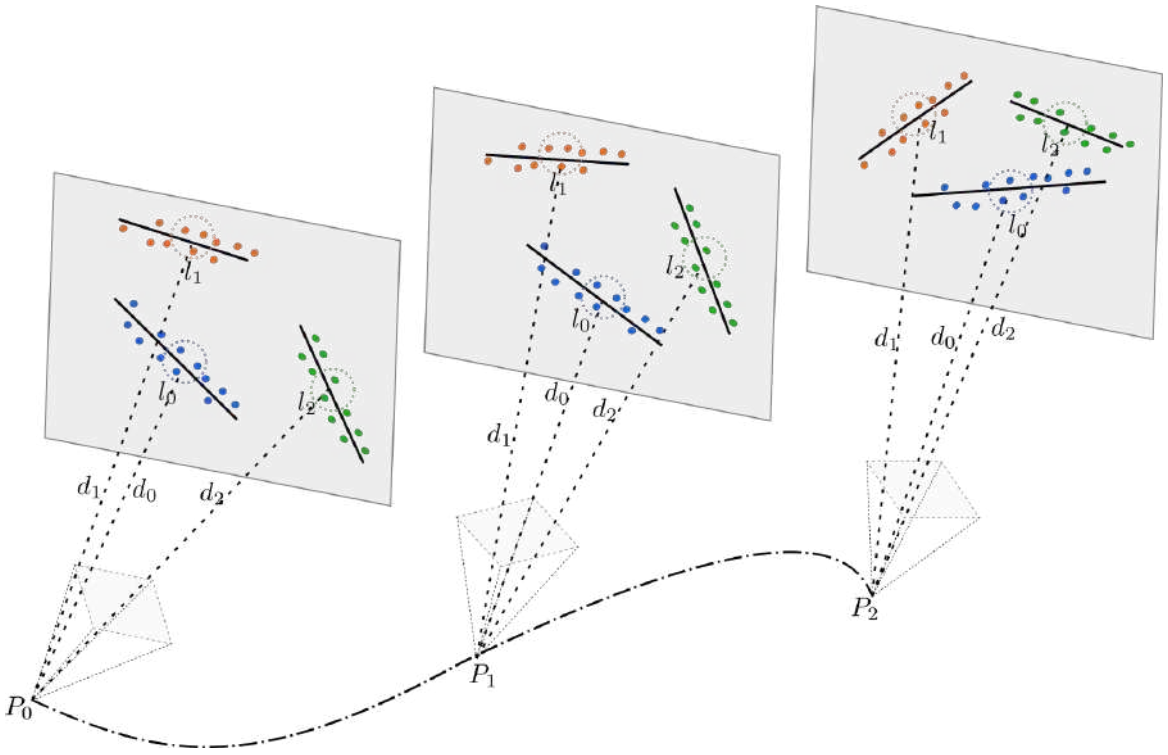


FIGURE 6.3: Scheme of different detected lines of different time steps with their assigned events with a small radius around the center point.

algorithm presented in Chapter 5 to augment the prior information we know about the environment. We assume that the IMU and the camera are calibrated and the extrinsic transformation $\mathcal{T}_{ic}$ between the IMU and the camera is known (illustrated in Figure 6.4). To find 6-DoF initial pose using optical flow, we need to know the depth of events and to estimate depth we need the 6-DoF pose. We iteratively estimate an initial depth then use it to correct for accurate pose and twist estimation.

Initial Depth Estimation

The line detection algorithm provides the line parameters (center point, line vector and Principal optical flow) and the events assigned to each line. A 2D projected line on the image plane may have varying depth in 3D. However, the depth of events around the line's center point would have too little variations (see Figure 6.3). We choose only events around the center with their optical flow to participate in the initial depth estimation assuming small depth variation. Using the estimated optical flow and the IMU

measurements to estimate the depth according to Equation 6.8. Since the linear velocity is computed using a single integration of the IMU readings and the angular velocity is directly provided from it, we use a moving average window to alleviate the effect of the accelerometer white noise without removing the gravity vector offset. For the gyroscope we use a band pass filter to alleviate the bias offset and the white noise. This conditioning step improves the quality of the estimated linear and angular velocity. For each set of events around a line, we use Equation 6.8 where the only unknown is the inverse depth so each optical flow vector for an event gives two values of depth and the equation becomes:

$$\begin{bmatrix} \frac{1}{Z_{cx}} \\ \frac{1}{Z_{cy}} \end{bmatrix} = \left(\dot{\boldsymbol{x}} - \boldsymbol{B}(x_c, y_c)\Omega\right) / \left(\boldsymbol{A}(x_c, y_c)\mathbf{V}_c\right) \tag{6.32}$$

where the division here is element-wise division. The depth ratio $\left(\frac{1}{Z_{cx}} / \frac{1}{Z_{cy}}\right)$ should be identity because they belong to the same event. If the depth ratio is not in a bounded interval $[th_1, th_2]$, this implies that the estimated optical flow is highly corrupted and will be rejected. The initial depth assigned to all the events of the line is the mean of the estimated depth around the center after rejecting outliers.This initialization method is only effective if the depth does not vary much along each line, i.e. downward facing cameras of drones, cameras moving indoor in front of walls.

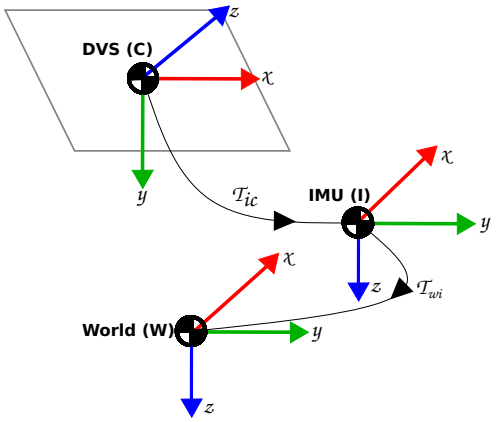
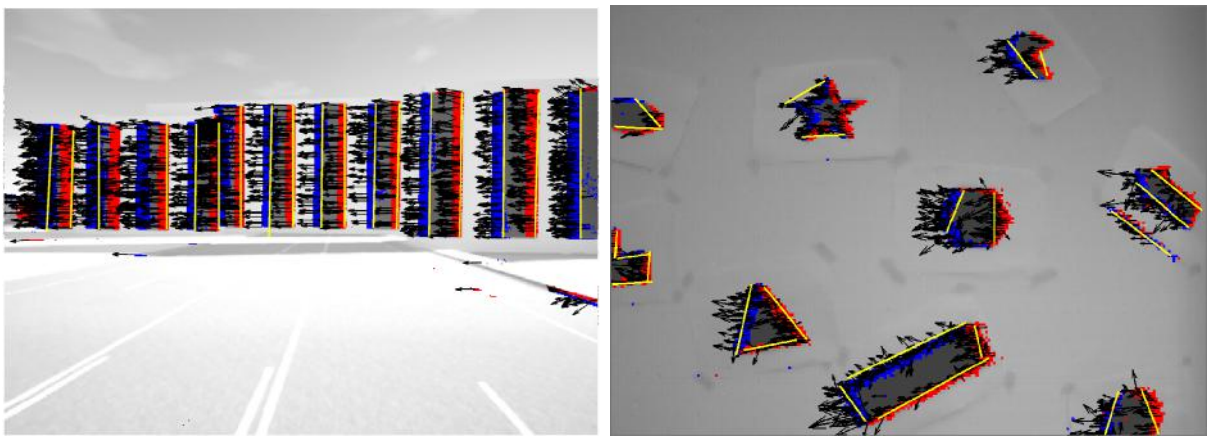


FIGURE 6.4: The coordinates frames of the IMU i, event-based camera c and the world w



(A) (B)

FIGURE 6.5: Grayscale images of the sequences used to test our algorithm with the triggered events (red for positive polarity and blue for negative polarity). The estimated optical flow arrows in black and the detected lines in yellows

Initial Pose and Twist Estimation

Using estimated depth of all events around center point of detected lines and after rejecting outlier optical flow, we re-inject the depth values into equation 6.8 after modifying it so that it becomes (for a single event):

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{-f}{Z_c} & 0 & \frac{(x_c-c_u)}{Z_c} & \frac{(x_c-c_u)(y_c-c_v)}{f} & -\left(f+\frac{(x_c-c_u)^2}{f}\right) & (y_c-c_v) \\ 0 & \frac{-f}{Z_c} & \frac{(y_c-c_v)}{Z_c} & \left(f+\frac{(y_c-c_v)^2}{f}\right) & -\frac{(x_c-c_u)(y_c-c_v)}{f} & -(x_c-c_y) \end{bmatrix} \boldsymbol{\zeta}^{\vee} \tag{6.33}$$

$$\dot{\boldsymbol{x}}_c = \boldsymbol{C}(x_c, y_c, Z_c)\boldsymbol{\zeta}^{\vee}$$

In Equation 6.33, the twist vector $\boldsymbol{\zeta}^{\vee}$ is the only unknown. We can stack the optical flow information for all events as:

$$\begin{bmatrix} \boldsymbol{C}_1(x_c, y_c, Z_c) \\ \vdots \\ \boldsymbol{C}_n(x_c, y_c, Z_c) \end{bmatrix} \boldsymbol{\zeta}^{\vee} = \begin{bmatrix} \dot{\boldsymbol{x}}_{c1} \\ \vdots \\ \dot{\boldsymbol{x}}_{cn} \end{bmatrix} \tag{6.34}$$

This equation can be solved for $\boldsymbol{\zeta}^{\vee}$ using least square method for $\boldsymbol{A}x = \boldsymbol{b}$ where the solution would be $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. Estimating the depth and twist is repeated iteratively

until convergence to make sure initialized depth and twist are correctly estimated. The initial pose is estimated by integrating the twist vector (Equations 6.14, 6.15 and 6.16).

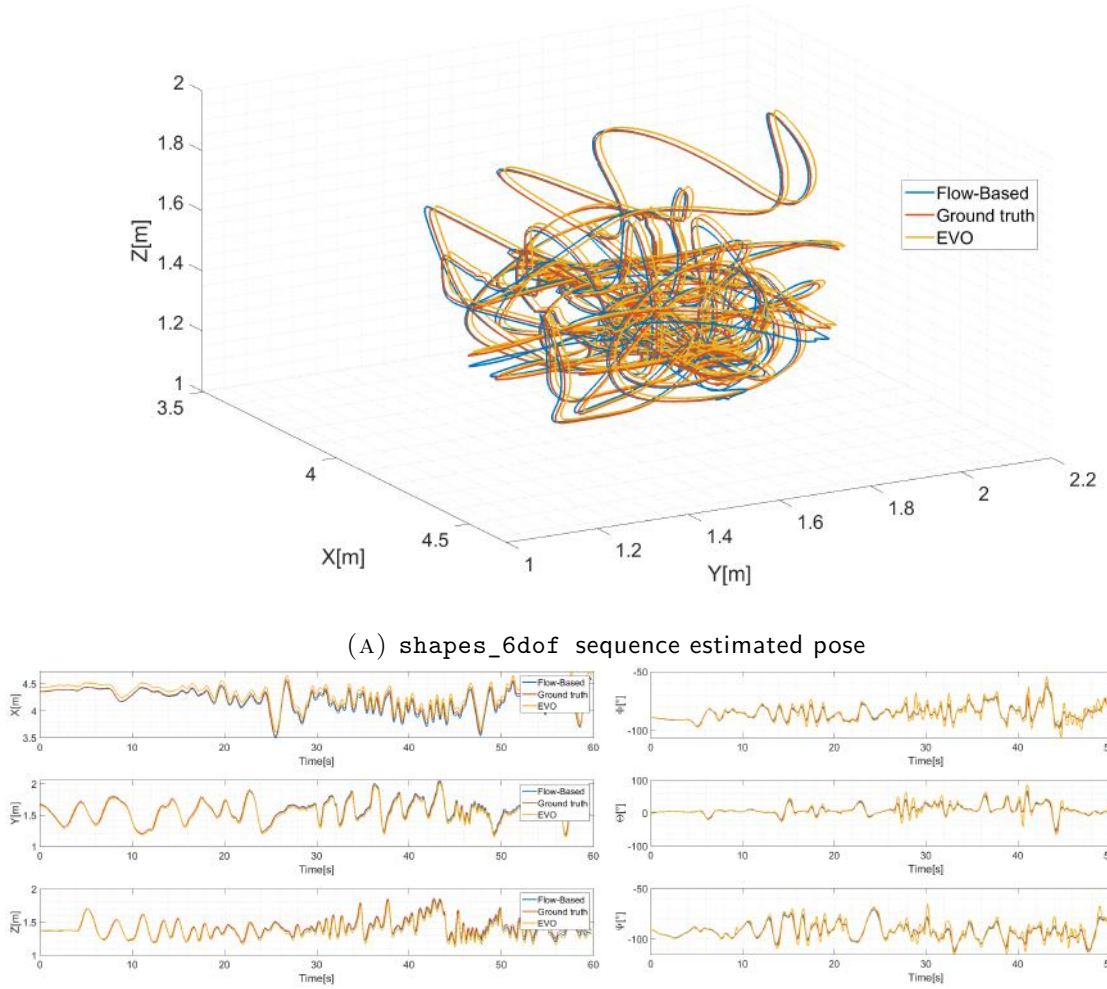
6.4 Experimental Setup

Our proposed visual-inertial odometry scheme performs in structured environments containing lines with low depth variations. For this purpose, we choose sequences fulfilling these criteria in order to provide a fair assessment. We used one of `IBISCape` sequences provided in [Soliman et al., 2022] of a car moving in an environment augmented with white walls and black rectangles at different depths. Additionally, we used the sequence of `shapes_6dof` provided in [Mueggler et al., 2017] of a handheld camera moving randomly in front of different geometric shapes depicted on a wall. These sequence were, first, passed through the optical flow estimator then the lines detector to have all the required information for optimization (see Figure 6.5).

`shapes_6dof`				
Method	ARMS [m]		ARMS [°]	
	μ	σ	μ	σ
EVO [Rebecq et al., 2016b]	0.09103	0.0051	5.0217	0.9851
$Flow-Based$ (All events)	**0.0802**	**0.0043**	**2.5791**	**1.9732**
$Flow-Based$ (25% dropped)	0.0841	0.0094	2.8041	1.8541
$Flow-Based$ (50% dropped)	0.0971	0.0158	2.8460	1.9471
$Flow-Based$ (75% dropped)	–	–	–	–
`IBISCape`				
Method	ARMS [m]		ARMS [°]	
	μ	σ	μ	σ
EVO [Rebecq et al., 2016b]	0.1369	0.0082	1.7840	0.6214
$Flow-Based$ (All events)	**0.1204**	**0.0079**	1.5602	0.7683
$Flow-Based$ (25% dropped)	0.1231	0.0117	1.5874	0.8024
$Flow-Based$ (50% dropped)	0.1217	0.0172	**1.4272**	**0.8401**
$Flow-Based$ (75% dropped)	–	–	–	–

TABLE 6.1: Average Root Mean Square Error of `shapes_6dof` and `IBISCape` sequence.

We use Ceres solver [Agarwal et al., 2022] as an optimizer for its automatic differentiation



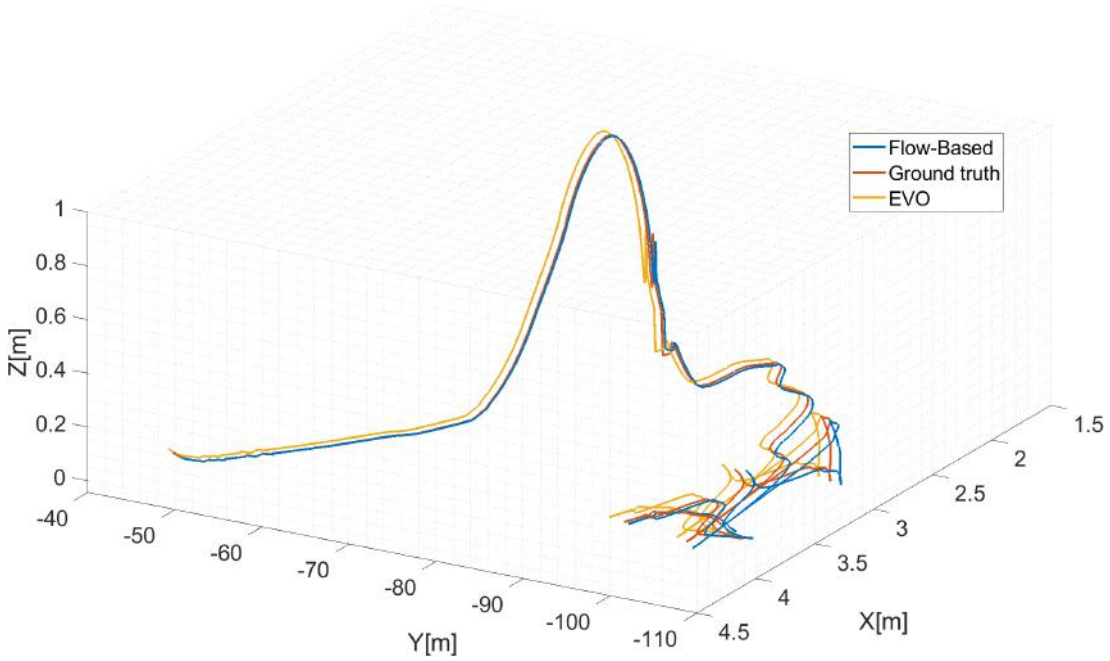
(A) `shapes_6dof` sequence estimated pose

(B) `shapes_6dof` position in each axis

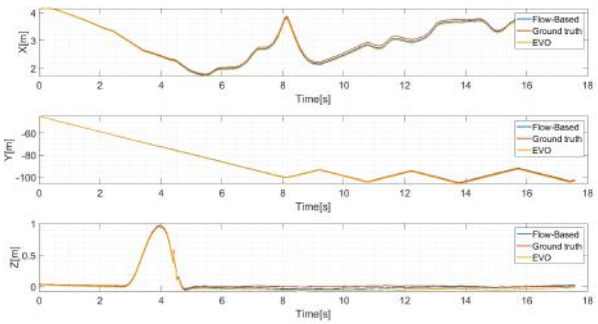
(C) `shapes_6dof` angles in each axis

FIGURE 6.6: The estimated pose, position and angles of `shapes_6dof` Flow-Based method in blue, the ground truth in red and EVO in yellow

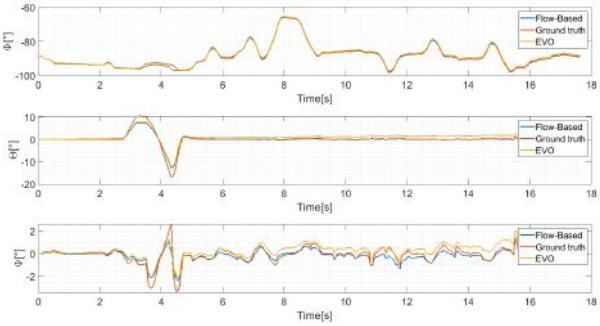
capability. Our algorithm run on a $3GHz$ Core $i7$ 16 core Linux machine. We have set our time step to 0.025 s where 5 IMU measurements are preintegrated for `IBISCape`'s sequence and 25 measurements are preintegrated for the `shapes_6dof` sequence. Being recorded with a handheld camera, `shapes_6dof` sequence undergoes high rotational speed and relatively low translational speed while `IBISCape`'s sequence have the opposite characteristics since it is recorded as a car's onboard camera.



(A) IBISCape sequence estimated pose



(B) IBISCape position in each axis

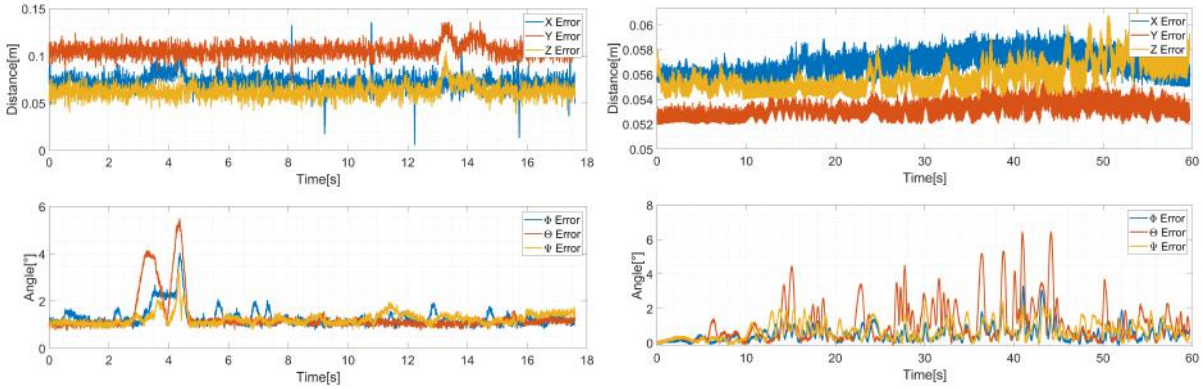


(C) IBISCape angles in each axis

FIGURE 6.7: The estimated pose, position and angles of IBISCape Flow-Based method in blue, the ground truth in red and EVO in yellow

6.5 Results

We compare the results of our flow-based approach with the EVO method [Rebecq et al., 2016b]. We use the Average Root Mean Square error (ARMS) to show the accuracy of our algorithm (see Tables 6.1 and Figures 6.6 and 6.7). IBISCape's sequence had a higher ARMS for translation because of its high translational speed. Unlikely, shapes_6dof sequence attained a lower ARMS for translation for the same reason. The rotational ARMS error is maintained relatively small because of the accuracy of the IMU measurements (see Figure 6.8).



(A) Position and angle errors of `IBISCape` sequence

(B) Position and angle errors of `shapes_6dof` sequence.

FIGURE 6.8: Errors of flow-based visual-inertial odometry method.

We ran many experiments to check for the accuracy of our system with and without dropping events to alleviate for real-time computation. The assumption that our scheme will still work in case of events being dropped is made since it only depends on optical flow (and not tracked features) and that the number of optimization residuals is always much lower than the amount of events at each time step. We found that our system can hold accurate results until we reach around 50% of dropped events for `shapes_6dof` sequence and about 60% (see Table 6.1) of dropped events for `IBISCape`'s sequence[1]. The amount of events that can be dropped depends on events frequency. `IBISCape`'s sequence maintained good results while more events were dropped. The accuracy did not vary much before failure occurred which validates the assumption that events can be dropped with a threshold depending on events frequency and camera resolution. Dropping the events can also be improved to maintain accuracy by choosing the dropped events being assigned to lines where each line should have a minimum amount of events to avoid failure.

To measure the computational time of our scheme, measurements to be optimized are placed in a sliding window where previously optimized poses are considered constant and only the sliding window is optimized. Table 6.2 shows the computational time of different windows with different percentages of dropped events where no events. The

[1]Table 6.2 shows results for 25, 50, and 75 percent of dropped events as milestones for brevity

drop $[\%]$	packet size $[-]$	packet time $[s]$	residual and jacobian time $[s]$	linear solver $[s]$	Total time $[s]$
–	50	0.25	0.420759	0.304085	0.724844
–	100	0.5	0.624733	0.496576	1.121309
–	150	0.75	0.956266	0.912470	1.868736
–	200	1	1.059416	0.998935	2.058351
25	50	0.25	0.262560	0.091245	0.353805
25	100	0.5	0.545977	0.215199	0.761176
25	150	0.75	0.729159	0.275548	1.004707
25	200	1	0.845035	0.317980	1.163015
50	50	0.25	0.235112	0.082975	0.318087
50	100	0.5	0.345446	0.104557	**0.450003**
50	150	0.75	0.465738	0.133461	**0.599199**
50	200	1	0.627081	0.188407	**0.815488**
75	50	0.25	0.691566	0.113761	0.805327
75	100	0.5	0.997071	0.208451	1.205522
75	150	0.75	1.285245	0.418131	1.703376
75	200	1	1.375911	0.537240	1.913151

TABLE 6.2: Average Root Mean Square Error of `shapes_6dof` sequence.

high computational time for `IBISCape`'s simulator sequence is due to the very high events frequency (data were generated for a 1024×1024 camera resolution). Although, `shapes_6dof` sequence attained real-time for all the sliding windows with no dropped events.

The number of IMU measurements and the amount of events to be dropped defines the compromise to achieve real-time applicability (see Figure 6.9). We should keep the smallest possible sliding window with the maximum amount of events to be dropped which leads to a trade-off between computational time and accuracy (sliding windows allowing real-time performance are shown in bold within Table 6.2).

6.6 Conclusion

We introduce a flow-based visual-inertial odometry algorithm for neuromorphic vision sensors. The algorithm corrects optical flow information using IMU measurements in environments where lines can be detected. We run our algorithm without the need of triangulation or key-frame estimation which provides a liberty to choose the size of our

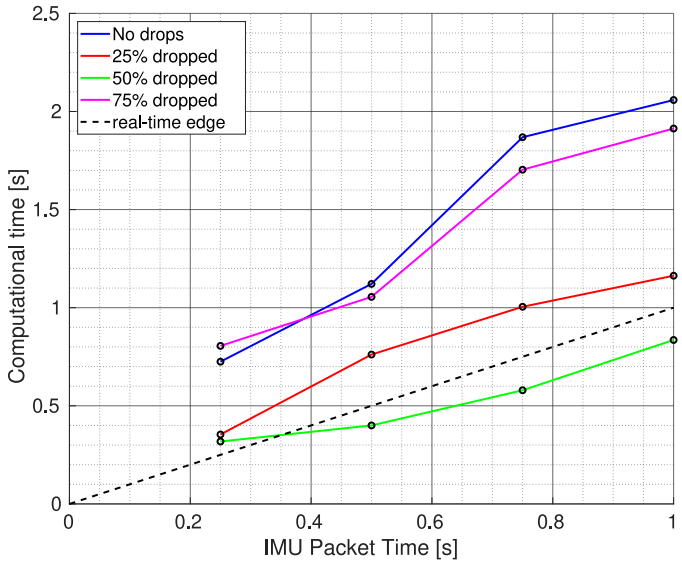


FIGURE 6.9: The computational time of our algorithm

sliding window for optimization.

Instead of running for only scenarios where the depth of lines does not vary much, the introduced algorithm can run freely if backed with a depth sensor. Integrating a depth sensor can also be used to estimate more accurate optical flow. Other improvement to our system would be place recognition in order to have the ability to close the loop in a complete SLAM system.

7 Conclusion

We started this thesis to study the capabilities of neuromorphic vision sensors and to provide a better understanding of their capabilities and restrictions. Keeping in mind our philosophy when we first started this research topic, we tried to maintain equilibrium between delving into details and keeping the big picture in mind. Therefore, we approached and improved different algorithms that can be used independently to solve problems related to neuromorphic vision while introducing a comprehensive scheme that encapsulates all the developed algorithms to serve as a solution for the main challenge of this thesis.

We provided an optical flow algorithm capable of delivering competent accuracy compared to the family of optical flow algorithms corresponding to it. Moreover, we kept improving the quality of estimated optical flow whenever possible after having the initial estimate of optical flow. Nevertheless, keeping the same notion of alleviating the complexity of solutions we provide, our line detection and segmentation algorithm offers highly accurate results based on straightforward conditions.

We introduce the first neuromorphic visual-inertial odometry algorithm that exploits optical flow information as a parameter for optimization, which provides sufficient accuracy. Our algorithm is liberated from triangulation or keyframe selection which spares more time for the optimization process. Furthermore, it can be tuned to optimize different

sizes of sliding windows for optimization based on the dynamics of the application and can drop unnecessary events to maintain real-time applicability.

At each phase of this thesis, we would discover that improvements can be achieved if either more sensors were added to our system or more complex techniques were adopted. Although, we preferred to follow the same philosophy we embraced from the beginning to evaluate the capabilities of event-based cameras in a minimal system. In the last phase of this thesis, where we stand on solid ground and understand the advantages and constraints of event-based cameras, we come to the conclusion that to use event-based cameras in versatile scenarios with varying conditions, other sensors should reinforce event-based cameras. Although we provide accurate outcomes, our scheme is restrained to certain scenarios where the environment should be rich in lines with depth that does vary much.

Therefore, we believe that reinforcing our system with a depth sensor would much improve the quality of odometry estimation. Integrating depth information would be the most convenient in the optimization scheme to tackle the problem of optimization conditioning. However, this would contradict the way we adopted to deliver this research. In the future, we aspire, with the integration of a depth sensor, to revisit the optical flow estimation and determine how depth information can improve the optical flow estimation and provide 3D visual flow information instead of 2D optical flow. Additionally, integrating a depth sensor would liberate us from the necessity of detecting or tracking lines which would save computational resources. Even if depth information would be used to improve optical flow estimation, we aim to embed the depth information in the optimization cost function and evaluate how it would enhance the optimization output. Furthermore, we target completing our scheme to provide a full SLAM system which requires place recognition techniques to help close the loop and correct for drift for long-range scenarios. Developing a place recognition algorithm for event-based cameras would require algorithms capable of classifying repetitive patterns in hierarchies such as the bag of visual words, which is

still an unexplored area for event-based cameras.

Bibliography

[Agarwal et al., 2022] Agarwal, S., Mierle, K., and Team, T. C. S. (2022). Ceres Solver.

[al Jazarī, 1973] al Jazarī, I. a.-R. (1973). *The Book of Knowledge of Ingenious Mechanical Devices:(Kitāb Fī Ma'rifat Al-hiyal Al-handasiyya)*. Springer.

[Almatrafi et al., 2020] Almatrafi, M., Baldwin, R., Aizawa, K., and Hirakawa, K. (2020). Distance surface for event-based optical flow. *IEEE transactions on pattern analysis and machine intelligence*, 42(7):1547–1556.

[Almatrafi and Hirakawa, 2019] Almatrafi, M. and Hirakawa, K. (2019). Davis camera optical flow. *IEEE Transactions on Computational Imaging*, 6:396–407.

[Amir et al., 2017] Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., et al. (2017). A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252.

[Baker and Matthews, 2004] Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255.

[Baker et al., 2011] Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M. J., and Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31.

[Bardow et al., 2016] Bardow, P., Davison, A. J., and Leutenegger, S. (2016). Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition, pages 884–892.

[Barranco et al., 2015] Barranco, F., Fermuller, C., and Aloimonos, Y. (2015). Bio-inspired motion estimation with event-driven sensors. In *International Work-Conference on Artificial Neural Networks*, pages 309–321. Springer.

[Barranco et al., 2016] Barranco, F., Fermuller, C., Aloimonos, Y., and Delbruck, T. (2016). A dataset for visual navigation with neuromorphic methods. *Frontiers in neuroscience*, 10:49.

[Benosman et al., 2013] Benosman, R., Clercq, C., Lagorce, X., Ieng, S.-H., and Bartolozzi, C. (2013). Event-based visual flow. *IEEE transactions on neural networks and learning systems*, 25(2):407–417.

[Benosman et al., 2012] Benosman, R., Ieng, S.-H., Clercq, C., Bartolozzi, C., and Srinivasan, M. (2012). Asynchronous frameless event-based optical flow. *Neural Networks*, 27:32–37.

[Bi et al., 2019] Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., and Andreopoulos, Y. (2019). Graph-based object classification for neuromorphic vision sensing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 491–501.

[Binas et al., 2017] Binas, J., Neil, D., Liu, S.-C., and Delbruck, T. (2017). Ddd17: End-to-end davis driving dataset. *arXiv preprint arXiv:1711.01458*.

[Boahen, 2000] Boahen, K. A. (2000). Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5):416–434.

[Bolles et al., 1987] Bolles, R. C., Baker, H. H., and Marimont, D. H. (1987). Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision*, 1(1):7–55.

[Brandli et al., 2014] Brandli, C., Berner, R., Yang, M., Liu, S.-C., and Delbruck, T. (2014). A 240×180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341.

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

[Calabrese et al., 2019] Calabrese, E., Taverni, G., Awai Easthope, C., Skriabine, S., Corradi, F., Longinotti, L., Eng, K., and Delbruck, T. (2019). Dhp19: Dynamic vision sensor 3d human pose dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

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

[Chamorro Hernández et al., 2020] Chamorro Hernández, W. O., Andrade-Cetto, J., and Solà Ortega, J. (2020). High-speed event camera tracking. In *Proceedings of the The 31st British Machine Vision Virtual Conference*, pages 1–12.

[Chen et al., 2018] Chen, C., Zhu, H., Li, M., and You, S. (2018). A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives. *Robotics*, 7(3):45.

[Chen et al., 2014] Chen, M., Yang, Q., Li, Q., Wang, G., and Yang, M.-H. (2014). Spatiotemporal background subtraction using minimum spanning tree and optical flow. In *European Conference on Computer Vision*, pages 521–534. Springer.

[Chirikjian, 2011] Chirikjian, G. S. (2011). *Stochastic models, information theory, and Lie groups, volume 2: Analytic methods and modern applications*, volume 2. Springer Science & Business Media.

[Comport et al., 2007] Comport, A. I., Malis, E., and Rives, P. (2007). Accurate quadrifocal tracking for robust 3d visual odometry. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 40–45. IEEE.

[Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223.

[Cottini et al., 2013] Cottini, N., Gottardi, M., Massari, N., Passerone, R., and Smilansky, Z. (2013). A 33μ w 64×64 pixel vision sensor embedding robust dynamic background subtraction for event detection and scene interpretation. *IEEE Journal of Solid-State Circuits*, 48(3):850–863.

[Cyganek and Siebert, 2011] Cyganek, B. and Siebert, J. P. (2011). *An introduction to 3D computer vision techniques and algorithms*. John Wiley & Sons.

[de Tournemire et al., 2020] de Tournemire, P., Nitti, D., Perot, E., Migliore, D., and Sironi, A. (2020). A large scale event-based detection dataset for automotive. *arXiv preprint arXiv:2001.08499*.

[Delbrück, 2008] Delbrück, T. (2008). Frame-free dynamic digital vision. In *Proceedings of International Symposium on Secure-Life Electronics, Advanced Electronics for Quality Life and Society, Univ. of Tokyo, Mar. 6-7, 2008*, pages 21–26. nternational Symposium on Secure-Life Electronics, Advanced Electronics for ....

[Delbruck, 2008] Delbruck, T. (2008). Frame-free dynamic digital vision. In *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, volume 1, pages 21–26. Citeseer.

[Drazen et al., 2011] Drazen, D., Lichtsteiner, P., Häfliger, P., Delbrück, T., and Jensen, A. (2011). Toward real-time particle tracking using an event-based dynamic vision sensor. *Experiments in Fluids*, 51(5):1465.

[Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

[Edwards, 1984] Edwards, A. L. (1984). An introduction to linear regression and correlation. Technical report.

[El-Diasty and Pagiatakis, 2010] El-Diasty, M. and Pagiatakis, S. (2010). Calibration and stochastic m odelling of i nertial na vigation s ensor errors. *Positioning (POS) Journal Information*, page 80.

[El-Sheimy et al., 2007] El-Sheimy, N., Hou, H., and Niu, X. (2007). Analysis and modeling of inertial sensors using allan variance. *IEEE Transactions on instrumentation and measurement*, 57(1):140–149.

[Ellis, 2012] Ellis, G. (2012). *Control system design guide: using your computer to understand and diagnose feedback controllers*. Butterworth-Heinemann.

[Engel et al., 2016] Engel, J., Koltun, V., and Cremers, D. (2016). Direct sparse odometry. *CoRR*, abs/1607.02565.

[Engel et al., 2018] Engel, J., Koltun, V., and Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625.

[Engel et al., 2013] Engel, J., Sturm, J., and Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456.

[Engelberger, 2012] Engelberger, J. F. (2012). *Robotics in practice: management and applications of industrial robots*. Springer Science & Business Media.

[Everding and Conradt, 2018] Everding, L. and Conradt, J. (2018). Low-latency line tracking using event-based dynamic vision sensors. *Frontiers in neurorobotics*, 12:4.

[Fanuc, 2022] Fanuc (2022). Fanuc. https://www.fanuc.com/.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[Forster et al., 2016] Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2016). On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21.

[Forster et al., 2014] Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE.

[Furgale et al., 2012] Furgale, P., Barfoot, T. D., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *2012 IEEE International Conference on Robotics and Automation*, pages 2088–2095. IEEE.

[Furgale et al., 2014] Furgale, P., Maye, J., Rehder, J., and Schneider, T. (2014). Kalibr: A unified camera/imu calibration toolbox.

[Gallego et al., 2019] Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., and Scaramuzza, D. (2019). Event-based vision: A survey. *CoRR*.

[Gallego et al., 2022] Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., and Scaramuzza, D. (2022). Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180.

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

[Gallego et al., 2018] Gallego, G., Rebecq, H., and Scaramuzza, D. (2018). A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876.

[Gálvez-López and Tardos, 2012] Gálvez-López, D. and Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

[Gao et al., 2018] Gao, X., Wang, R., Demmel, N., and Cremers, D. (2018). LDSO: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

[Ghosh-Dastidar and Adeli, 2009] Ghosh-Dastidar, S. and Adeli, H. (2009). Spiking neural networks. *International journal of neural systems*, 19(04):295–308.

[Gibson, 1950] Gibson, J. J. (1950). *The Perception Of The Visual World*. Boston: Houghton Mifflin.

[Graeter et al., 2018] Graeter, J., Wilczynski, A., and Lauer, M. (2018). Limo: Lidar-monocular visual odometry. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 7872–7879. IEEE.

[Gumbs et al., 2021] Gumbs, A. A., Frigerio, I., Spolverato, G., Croner, R., Illanes, A., Chouillard, E., and Elyan, E. (2021). Artificial intelligence surgery: How do we get to autonomous actions in surgery? *Sensors*, 21(16):5526.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[Heeger and Jepson, 1992] Heeger, D. J. and Jepson, A. D. (1992). Subspace methods for recovering rigid motion i: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117.

[Hirose and Saito, 2012] Hirose, K. and Saito, H. (2012). Fast line description for line-based slam. In *BMVC*, pages 1–11.

[Honour, 2009] Honour, J. (2009). A brief history of principles used in high speed cameras. *The Imaging Science Journal*, 57(6):303–316.

[Horn, 1987] Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4):629–642.

[Horn and Negahdaripour, 1987] Horn, B. K. and Negahdaripour, S. (1987). Direct passive navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:168–176.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, 17(1-3):185–203.

[Horn and Weldon, 1988] Horn, B. K. and Weldon, E. (1988). Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76.

[Hu et al., 2016] Hu, Y., Liu, H., Pfeiffer, M., and Delbruck, T. (2016). Dvs benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in neuroscience*, 10:405.

[Incisio, 2022] Incisio, V. (2022). Virual incisio. https://virtualincision.com/.

[Intelligence, 2021] Intelligence, M. (2021). Robotic vision market - growth, trends, covid-19 impact, and forecasts (2022 - 2027). Technical report, Mordor Intelligence.

[Jacobs, 2001] Jacobs, D. W. (2001). Linear fitting with missing data for structure-from-motion. *Computer Vision and Image Understanding*, 82(1):57–81.

[Kanade and Morris, 1998] Kanade, T. and Morris, D. D. (1998). Factorization methods for structure from motion. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1740):1153–1173.

[Kim et al., 2008] Kim, H., Handa, A., Benosman, R., Ieng, S.-H., and Davison, A. J. (2008). Simultaneous mosaicing and tracking with an event camera. *J. Solid State Circ*, 43:566–576.

[Kim et al., 2016] Kim, H., Leutenegger, S., and Davison, A. J. (2016). Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer.

[Kitt et al., 2010] Kitt, B., Geiger, A., and Lategahn, H. (2010). Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *2010 ieee intelligent vehicles symposium*, pages 486–492. IEEE.

[Klenk et al., 2021] Klenk, S., Chui, J., Demmel, N., and Cremers, D. (2021). Tum-vie: The tum stereo visual-inertial event dataset. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8601–8608. IEEE.

[Koenderink and Van Doorn, 1991] Koenderink, J. J. and Van Doorn, A. J. (1991). Affine structure from motion. *JOSA A*, 8(2):377–385.

[Konrad and Dubois, 1992] Konrad, J. and Dubois, E. (1992). Bayesian estimation of motion vector fields. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 14(09):910–927.

[Krizhevsky et al., 2009] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[Kueng et al., 2016] Kueng, B., Mueggler, E., Gallego, G., and Scaramuzza, D. (2016). Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 16–23. IEEE.

[Lacroix et al., 1999] Lacroix, S., Mallet, A., Chatila, R., and Gallo, L. (1999). Rover self localization in planetary-like environments. In *Artificial Intelligence, Robotics and Automation in Space*, volume 440, page 433.

[Le Gentil et al., 2020] Le Gentil, C., Tschopp, F., Alzugaray, I., Vidal-Calleja, T., Siegwart, R., and Nieto, J. (2020). Idol: A framework for imu-dvs odometry using lines. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5863–5870. IEEE.

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

[Lemaire and Lacroix, 2007] Lemaire, T. and Lacroix, S. (2007). Monocular-vision based slam using line segments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2791–2796. IEEE.

[Lepisk, 2005] Lepisk, A. (2005). The use of optic flow within background subtraction. *Master's Degree Project, Royal Institute of Technology of Stockholm, Sweden*.

[Li et al., 2017] Li, H., Liu, H., Ji, X., Li, G., and Shi, L. (2017). Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309.

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128$times$128 120 db 15μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576.

[Lin et al., 2021] Lin, Y., Ding, W., Qiang, S., Deng, L., and Li, G. (2021). Es-imagenet: A million event-stream classification dataset for spiking neural networks. *Frontiers in Neuroscience*, page 1546.

[Liu et al., 2017] Liu, M.-y., Wang, Y., and Guo, L. (2017). 6-dof motion estimation using optical flow based on dual cameras. *Journal of Central South University*, 24(2):459–466.

[Longuet-Higgins, 1981] Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135.

[Longuet-Higgins and Prazdny, 1980] Longuet-Higgins, H. C. and Prazdny, K. (1980). The interpretation of a moving retinal image. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 208(1173):385–397.

[Low et al., 2020] Low, W. F., Gao, Z., Xiang, C., and Ramesh, B. (2020). Sofea: A non-iterative and robust optical flow estimation algorithm for dynamic vision sensors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 82–83.

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, page 674–679. Vancouver.

[Lupton and Sukkarieh, 2011] Lupton, T. and Sukkarieh, S. (2011). Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76.

[Mahowald and Mead, 1991] Mahowald, M. A. and Mead, C. (1991). The silicon retina. *Sci. Am.*, 264(5):76–83.

[Matthies and Shafer, 1987] Matthies, L. and Shafer, S. (1987). Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248.

[Miao et al., 2019] Miao, S., Chen, G., Ning, X., Zi, Y., Ren, K., Bing, Z., and Knoll, A. (2019). Neuromorphic vision datasets for pedestrian detection, action recognition, and fall detection. *Frontiers in neurorobotics*, 13:38.

[Minaee et al., 2021] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., and Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*.

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

[Mob, 2022] Mob, W. (2022). https://www.widsmob.com/tips/ldr.html.

[Moravec, 1980] Moravec, H. P. (1980). *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, Stanford University.

[Mourikis and Roumeliotis, 2007] Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572.

[Mourikis et al., 2007] Mourikis, A. I., Roumeliotis, S. I., et al. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *ICRA*, volume 2, page 6.

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

[Mueggler et al., 2018] Mueggler, E., Gallego, G., Rebecq, H., and Scaramuzza, D. (2018). Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440.

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

[Mueggler et al., 2017] Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., and Scaramuzza, D. (2017). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149.

[Mur-Artal et al., 2015a] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015a). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.

[Mur-Artal et al., 2015b] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015b). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[Mur-Artal and Tardós, 2014] Mur-Artal, R. and Tardós, J. D. (2014). Fast relocalisation and loop closing in keyframe-based slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853.

[Mur-Artal and Tardós, 2015] Mur-Artal, R. and Tardós, J. D. (2015). Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *Robotics: Science and Systems*, page 9, Rome, Italy.

[Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

[Myers, 1980] Myers, W. (1980). Industry begins to use visual pattern recognition. *Computer*, 13(05):21–31.

[Nister, 2003] Nister, D. (2003). An efficient solution to the five-point relative pose problem. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern*

Recognition, 2003. Proceedings., volume 2, pages II–195. IEEE.

[Nistér et al., 2004] Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee.

[Orchard et al., 2013] Orchard, G., Benosman, R., Etienne-Cummings, R., and Thakor, N. V. (2013). A spiking neural network architecture for visual motion estimation. In *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 298–301. IEEE.

[Orchard et al., 2015] Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437.

[Padala et al., 2018] Padala, V., Basu, A., and Orchard, G. (2018). A noise filtering algorithm for event-based asynchronous change detection image sensors on truenorth and its implementation on truenorth. *Frontiers in neuroscience*, 12:118.

[Pan et al., 2020] Pan, L., Liu, M., and Hartley, R. (2020). Single image optical flow estimation with an event camera. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1669–1678. IEEE.

[Parra et al., 2011] Parra, I., Sotelo, M. A., Llorca, D. F., Fernández, C., Llamazares, A., Hernández, N., and García, I. (2011). Visual odometry and map fusion for gps navigation assistance. In *2011 IEEE International Symposium on Industrial Electronics*, pages 832–837. IEEE.

[Paul et al., 2020] Paul, A., Ghosh, S., Das, A. K., Goswami, S., Das Choudhury, S., and Sen, S. (2020). A review on agricultural advancement based on computer vision and machine learning. In *Emerging technology in modelling and graphics*, pages 567–581. Springer.

[Pearson, 1901] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

[Photography, 2022] Photography, H. (2022). https://www.hackingphotography.com/motion-blur/.

[Posch et al., 2010a] Posch, C., Matolin, D., and Wohlgenannt, R. (2010a). 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*, 46(1):259–275.

[Posch et al., 2010b] Posch, C., Matolin, D., and Wohlgenannt, R. (2010b). 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*, 46(1):259–275.

[Qin, 2022] Qin, M. (2022). https://maryqin.com/college-night-at-the-getty/dsc01353/.

[Qin et al., 2018] Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.

[Rebecq et al., 2018] Rebecq, H., Gallego, G., Mueggler, E., and Scaramuzza, D. (2018). Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *International Journal of Computer Vision*, 126(12):1394–1414.

[Rebecq et al., 2016a] Rebecq, H., Gallego, G., and Scaramuzza, D. (2016a). Emvs: Event-based multi-view stereo.

[Rebecq et al., 2017] Rebecq, H., Horstschaefer, T., Gallego, G., and Scaramuzza, D. (2017). Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600.

[Rebecq et al., 2017] Rebecq, H., Horstschaefer, T., and Scaramuzza, D. (2017). Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization.

[Rebecq et al., 2016b] Rebecq, H., Horstschäfer, T., Gallego, G., and Scaramuzza, D. (2016b). Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600.

[Rebecq et al., 2019] Rebecq, H., Ranftl, R., Koltun, V., and Scaramuzza, D. (2019). Events-to-video: Bringing modern computer vision to event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3857–3866.

[Rosheim, 2006] Rosheim, M. (2006). *Leonardos Lost Robots*. Springer Science & Business Media.

[Rubio et al., 2019] Rubio, F., Valero, F., and Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596.

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

[Ruifang et al., 2017] Ruifang, D., Frémont, V., Lacroix, S., Fantoni, I., and Changan, L. (2017). Line-based monocular graph slam. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 494–500. IEEE.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[School, 2017] School, N. F. (2017). https://nofilmschool.com/2017/11/5000-ftsec-25-million-fps-mechanics-ultra-high-speed-cameras.

[Schraml et al., 2015] Schraml, S., Nabil Belbachir, A., and Bischof, H. (2015). Event-driven stereo matching for real-time 3d panoramic vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 466–474.

[Schuman et al., 2017] Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., and Plank, J. S. (2017). A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*.

[Seifozzakerini et al., 2018] Seifozzakerini, S., Yau, W.-Y., Mao, K., and Nejati, H. (2018). Hough transform implementation for event-based systems: Concepts and challenges. *Frontiers in computational neuroscience*, page 103.

[Serrano-Gotarredona and Linares-Barranco, 2013] Serrano-Gotarredona, T. and Linares-Barranco, B. (2013). A 128\\$times$128 1.5% contrast sensitivity 0.9% fpn 3 μs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, 48(3):827–838.

[Shashua and Hanna, 1995] Shashua, A. and Hanna, K. J. (1995). The tensor brightness constraints: Direct estimation of motion revisited. Technical report, Technion Technical Report, Haifa, Israel.

[Silva et al., 2018] Silva, R. L., Rudek, M., Szejka, A. L., and Junior, O. C. (2018). Machine vision systems for industrial quality control inspections. In *IFIP International Conference on Product Lifecycle Management*, pages 631–641. Springer.

[Sironi et al., 2018] Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., and Benosman, R. (2018). Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740.

[Sola, 2017] Sola, J. (2017). Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*.

[Soliman et al., 2022] Soliman, A., Bonardi, F., Sidibé, D., and Bouchafa, S. (2022). IBISCape: A simulated benchmark for multi-modal slam systems evaluation in large-scale dynamic environments. *arXiv:2206.13455v1*.

[Souissi et al., 2013] Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N., and Feyzeau, P. (2013). Path planning: A 2013 survey. In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1–8. IEEE.

[Stein and Shashua, 1996] Stein, G. P. and Shashua, A. (1996). Direct methods for estimation of structure and motion from three views.

[Steinbrücker et al., 2011] Steinbrücker, F., Sturm, J., and Cremers, D. (2011). Real-time visual odometry from dense rgb-d images. In *2011 IEEE international conference on computer vision workshops (ICCV Workshops)*, pages 719–722. IEEE.

[Stoffregen and Kleeman, 2019] Stoffregen, T. and Kleeman, L. (2019). Event cameras, contrast maximization and reward functions: An analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12300–12308.

[Taylor, 1717] Taylor, B. (1717). *Methodus incrementorum directa et inversa*. Innys.

[Tykkälä et al., 2011] Tykkälä, T., Audras, C., and Comport, A. I. (2011). Direct iterative closest point for real-time visual odometry. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2050–2056. IEEE.

[Valeiras et al., 2018] Valeiras, D. R., Clady, X., Ieng, S.-H., and Benosman, R. (2018). Event-based line fitting and segment detection using a neuromorphic visual sensor. *IEEE transactions on neural networks and learning systems*, 30(4):1218–1230.

[Vargas-Sierra et al., 2014] Vargas-Sierra, S., Liñán-Cembrano, G., and Rodríguez-Vázquez, Á. (2014). A 151 db high dynamic range cmos image sensor chip architecture with tone mapping compression embedded in-pixel. *IEEE Sensors Journal*, 15(1):180–195.

[Vidal et al., 2018] Vidal, A. R., Rebecq, H., Horstschaefer, T., and Scaramuzza, D. (2018). Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001.

[Von Gioi et al., 2008] Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. (2008). Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732.

[Wahba, 1965] Wahba, G. (1965). A least squares estimate of satellite attitude. *SIAM review*, 7(3):409–409.

[Wanlass and Sah, 1991] Wanlass, F. M. and Sah, C. T. (1991). Nanowatt logic using field-effect metal-oxide semiconductor triodes. In *Semiconductor devices: pioneering papers*, pages 637–638. World Scientific.

[waymo, 2022] waymo (2022). waymo. https://waymo.com/.

[Weikersdorfer et al., 2014] Weikersdorfer, D., Adrian, D. B., Cremers, D., and Conradt, J. (2014). Event-based 3d slam with a depth-augmented dynamic vision sensor. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 359–364.

[Weikersdorfer et al., 2014] Weikersdorfer, D., Adrian, D. B., Cremers, D., and Conradt, J. (2014). Event-based 3d slam with a depth-augmented dynamic vision sensor. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 359–364. IEEE.

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

[Weikersdorfer et al., 2013] Weikersdorfer, D., Hoffmann, R., and Conradt, J. (2013). Simultaneous localization and mapping for event-based vision systems. In *International Conference on Computer Vision Systems*, pages 133–142. Springer.

[Weiss et al., 2012] Weiss, S., Achtelik, M. W., Chli, M., and Siegwart, R. (2012). Versatile distributed pose estimation and sensor self-calibration for an autonomous mav. In *2012 IEEE International Conference on Robotics and Automation*, pages 31–38. IEEE.

[Weiss and Siegwart, 2011] Weiss, S. and Siegwart, R. (2011). Real-time metric state estimation for modular vision-inertial systems. In *2011 IEEE international conference on robotics and automation*, pages 4531–4537. IEEE.

[Xiao et al., 2002] Xiao, F., DiCarlo, J. M., Catrysse, P. B., and Wandell, B. A. (2002). High dynamic range imaging of natural scenes. In *Color and imaging conference*, volume 2002, pages 337–342. Society for Imaging Science and Technology.

[Yaqoob et al., 2019] Yaqoob, I., Khan, L. U., Kazmi, S. A., Imran, M., Guizani, N., and Hong, C. S. (2019). Autonomous driving cars in smart cities: Recent advances, requirements, and challenges. *IEEE Network*, 34(1):174–181.

[Zhu et al., 2018] Zhu, A. Z., Thakur, D., Özaslan, T., Pfrommer, B., Kumar, V., and Daniilidis, K. (2018). The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039.

[Zhu et al., 2019] Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2019). Unsupervised event-based learning of optical flow, depth, and egomotion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 989–997.

[Zhu et al., 2021] Zhu, L., Spachos, P., Pensini, E., and Plataniotis, K. N. (2021). Deep learning and machine vision for food processing: A survey. *Current Research in Food Science*, 4:233–249.

[Zihao Zhu et al., 2017] Zihao Zhu, A., Atanasov, N., and Daniilidis, K. (2017). Event-based visual inertial odometry. In *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition, pages 5391–5399.

[Zohaib et al., 2013] Zohaib, M., Pasha, M., Riaz, R., Javaid, N., Ilahi, M., and Khan, R. (2013). Control strategies for mobile robot with obstacle avoidance. *arXiv preprint arXiv:1306.1144*.

[Zucchelli, 2002] Zucchelli, M. (2002). *Optical flow based structure from motion*. PhD thesis, Numerisk analys och datalogi.

[Zuo et al., 2022] Zuo, Y.-F., Yang, J., Chen, J., Wang, X., Wang, Y., and Kneip, L. (2022). Devo: Depth-event camera visual odometry in challenging conditions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2179–2185. IEEE.