UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

# Event-based egomotion estimation

**Bachelor's Degree Thesis**

Sergi Sánchez Orvay

Bachelor's degree in Electronic Engineering and Telecommunications

Barcelona, June 2024

Advisor: Juan Andrade-Cetto
Thesis completed at Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Submitted to the Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona of the Universitat Politècnica de Catalunya

# Resum

Les càmeres d'esdeveniments són sensors bio-inspirats que detecten canvis d'il·luminació logarítmics a nivell de píxel, generant esdeveniments asíncronament amb resolució de microsegons. Aquest principi els confereix avantatges significatius respecte a les càmeres convencionals com un alt rang dinàmic, baixa latència i baix consum de potència, ideals per a escenaris dinàmics i condicions lumíniques adverses.

Aquest treball aborda l'estimació del moviment d'una càmera d'esdeveniments mitjançant un enfocament basat en models geomètrics i tècniques d'optimització, prescindint de l'ús d'aprenentatge per ser aplicable en entorns amb recursos limitats i entendre el problema en profunditat. Es realitza un estudi detallat de les bases matemàtiques subjacents al problema, guiant el desenvolupament del mètode proposat.

La metodologia proposada està constituïda per tres blocs principals: l'estimació del flux normal, l'estimació de la inversa de profunditat i un solucionador lineal robust. El mètode d'estimació del flux normal es basa en l'ajust de plans locals en superfícies d'esdeveniments actius, mentre que la inversa de profunditat s'aborda mitjançant tècniques d'optimització per a alinear esdeveniments en un temps de referència. Finalment, s'implementa un solucionador lineal robust basat en el mètode iteratiu RANSAC que utilitza les estimacions de la inversa de profunditat i el flux normal per obtenir la velocitat lineal de la càmera.

S'avalua el funcionament del mètode en experiments amb dades sintètiques i reals. Amb dades sintètiques, l'algoritme d'optimització de la inversa de profunditat aconsegueix alinear els esdeveniments de manera efectiva i convergeix ràpidament al seu valor òptim. A més, l'estimació del flux normal recupera tant la seva orientació com la seva magnitud en aquests exemples. Això permet que, malgrat certes limitacions presents en les dades sintètiques, el solucionador lineal aconsegueixi obtenir la velocitat lineal. No obstant això, s'observen fluxos normals mal condicionats quan s'avaluen en dades reals. Per a intentar corregir aquests fluxos normals, s'implementa un mètode de maxpooling a escala multiespacial que, tot i millorar les orientacions estimades, no aconsegueix recuperar la magnitud. D'altra banda, l'algoritme d'estimació de la inversa de profunditat demostra ser altament efectiu amb dades reals, aconseguint alinear els esdeveniments correctament en poques iteracions. Finalment, es discuteixen les limitacions observades i es proposen alternatives per a futures investigacions.

# Resumen

Las cámaras de eventos son sensores bio-inspirados que detectan cambios de iluminación logarítmicos a nivel de píxel, generando eventos asíncronamente con resolución de microsegundos. Este principio les confiere ventajas significativas respecto a las cámaras convencionales como alto rango dinámico, baja latencia y bajo consumo de potencia, ideales para escenarios dinámicos y condiciones lumínicas adversas.

Este trabajo aborda la estimación del movimiento de una cámara de eventos mediante un enfoque basado en modelos geométricos y técnicas de optimización, prescindiendo del uso de aprendizaje para ser aplicable en entornos con recursos limitados y entender el problema en profundidad. Se realiza un estudio detallado de las bases matemáticas subyacentes al problema, guiando el desarrollo del método propuesto.

La metodología propuesta está constituida por tres bloques principales: la estimación del flujo normal, la estimación de la inversa de profundidad y un solucionador lineal robusto. El método de estimación del flujo normal se basa en el ajuste de planos locales en superficies de eventos activos, mientras que la inversa de profundidad se aborda mediante técnicas de optimización para alinear eventos en un tiempo de referencia. Finalmente, se implementa un solucionador lineal robusto basado en el método iterativo RANSAC que utiliza las estimaciones de inversa de profundidad y normal flow para obtener la velocidad lineal de la cámara.

Se evalúa el funcionamiento del método en experimentos con datos sintéticos y reales. Con datos sintéticos, el algoritmo de optimización de inversa de profundidad consigue alinear los eventos efectivamente y converge rápidamente a su valor óptimo. Además, la estimación de flujo normal recupera tanto su orientación como su magnitud en estos ejemplos. Esto permite que, a pesar de ciertas limitaciones existentes en los datos sintéticos, el solucionador lineal consiga obtener la velocidad lineal. No obstante, se observan flujos normales mal condicionados cuando se prueban en datos reales. Para tratar de corregir estos flujos normales se implementa un método de maxpooling a escala multiespacial que, a pesar de mejorar las orientaciones estimadas, no consigue recuperar la magnitud. Por otra parte, el algoritmo de estimación de inversa de profundidad demuestra ser altamente efectivo con datos reales, consiguiendo alinear los eventos correctamente en pocas iteraciones. Finalmente, se discuten las limitaciones observadas y se proponen alternativas para investigaciones futuras.

# Abstract

Event cameras are bio-inspired sensors that detect logarithmic changes in pixel-level illumination, generating events asynchronously with microsecond resolution. This principle provides significant advantages over conventional cameras such as high dynamic range, low latency, and low power consumption, making them ideal for dynamic scenarios and challenging lighting conditions.

This work addresses the egomotion estimation of an event camera using a geometric model-based approach and optimization techniques, avoiding the use of learning methods to make it applicable in resource-constrained environments and to deeply understand the problem. A detailed study of the underlying mathematical foundations guides the development of the proposed method.

The proposed methodology consists of three main blocks: normal flow estimation, inverse depth estimation, and a robust linear solver. The normal flow estimation method is based on fitting local planes on surfaces of active events, while inverse depth is tackled using optimization techniques to align events at a reference timestamp. Finally, a robust linear solver based on the iterative RANSAC method is implemented, using the inverse depth and normal flow estimates to obtain the camera's linear velocity.

The method's performance is evaluated through experiments with synthetic and real data. With synthetic data, the inverse depth optimization algorithm effectively aligns events and converges rapidly to its ground truth value. Additionally, the normal flow estimation recovers both its orientation and magnitude in these examples. Despite certain limitations in synthetic data, this allows the linear solver to obtain the linear velocity. However, poorly conditioned normal flows are observed when tested with real data. To correct these normal flows, a multi-spatial scale maxpooling method is implemented, which improves estimated orientations but fails to recover magnitude. Conversely, the inverse depth estimation algorithm proves highly effective with real data, correctly aligning events in few iterations. Finally, limitations observed are discussed, and alternatives for future research are proposed.

# Acknowledgements

I would like to express my gratitude to my supervisor, Juan Andrade-Cetto, for his constant support and invaluable guidance throughout the development of this thesis. His expertise and patience were crucial as I navigated through uncharted territory. I am truly grateful for his mentorship, which has been pivotal in shaping this work.

I also wish to express my sincere gratitude to the staff at IRI for giving me the opportunity to participate in this project and for making this work possible.

I am also deeply thankful to my family for their unconditional support and encouragement. Their belief in me has been a ceaseless source of strength and motivation.

Lastly, I want to extend a special acknowledgement to my beloved Andrea. Her unwavering support and companionship have been my rock, making even the toughest moments more manageable. Her encouragement is essential in motivating me to wake up every day with the determination to pursue my goals.

# Revision history and approval record

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 1.0 | 25/05/2024 | Sergi Sánchez Orvay | Document creation |
| 1.1 | 09/06/2024 | Sergi Sánchez Orvay | First version |
| 1.2 | 11/06/2024 | Juan Andrade-Cetto | First review |
| 2.0 | 12/06/2024 | Sergi Sánchez Orvay | Correction after review |
| 3.0 | 13/06/2024 | Juan Andrade-Cetto | Second review |
| 3.1 | 14/06/2024 | Sergi Sánchez Orvay | Correction after review |
| 4.0 | 16/06/2024 | Juan Andrade-Cetto | Last review |
| 4.1 | 16/06/2024 | Sergi Sánchez Orvay | Last version |

## Document distribution list

| Role | Name and surname(s) |
|---|---|
| Student | Sergi Sánchez Orvay |
| Project Supervisor | Juan Andrade-Cetto |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| Date | 15/06/2024 | Date | 16/06/2024 |
| Name | Sergi Sánchez Orvay | Name | Juan Andrade-Cetto |
| Position | Project Author | Position | Project Supervisor |

# Contents

# Figures

# Tables

# Abbreviations

**2D** 2 Dimensions

**3D** 3 Dimensions

**AER** Address Event Representation

**ANN** Artificial Neural Network

**CMOS** Complementary Metal Oxide Semiconductor

**$CO_{2e}$** Carbon dioxide emissions

**CSIC** Consell Superior d'Investigacions Científiques

**DVS** Dynamic Vision Sensor

**ECTS** European Credit Transfer and Accumulation System

**ETSETB** Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

**GPU** Graphics Processing Unit

**HDR** High Dynamic Range

**IMU** Inertial Measurement Unit

**IRI** Institut de Robòtica i Informàtica Industrial

**IWE** Image of Warped Events

**LiDAR** Light Detection and Ranging

**RANSAC** RANdom SAmpling Consensus

**RGB-D** Red Green Blue - Depth

**RMSProp** Root Mean Square Propagation

**SAE** Surface of Active Events

**SDGs** Sustainable Development Goals

**SLAM** Simultaneous Localization and Mapping

**SNN** Spiking Neural Network

**SVD** Singular Value Decomposition

**UPC** Universitat Politècnica de Catalunya

**VO** Visual Odometry

**WP** Work Package

# Nomenclature

### General convention

| | |
|---|---|
| $v$ | Scalar |
| $v(\cdot)$ | Function |
| $\mathbf{V}$ | Matrix |
| $\mathbf{v}$ | Vector |
| $\hat{\mathbf{v}}$ | Normalized vector |
| $\tilde{\mathbf{v}}$ | Estimate of vector $\mathbf{v}$ |
| $\mu_{\mathbf{A}}$ | Mean value of matrix $\mathbf{A}$ |
| $\sigma(\mathbf{A})$ | Variance of matrix $\mathbf{A}$ |

### Specific convention

| | |
|---|---|
| $\mathbf{x}$ | Pixel coordinates on the image plane |
| $\dot{\mathbf{x}}$ | Optical flow vector |
| $\dot{\mathbf{x}}_n$ | Normal flow vector |
| $B$ | Body frame |
| ref | Reference viewpoint |
| $\mathbf{v}^B$ | Linear velocity expressed in body frame coordinates |
| $\mathbf{w}^B$ | Angular velocity expressed in body frame coordinates |
| $Z$ | Depth |
| $\rho$ | Inverse depth |
| $\mathbf{T}(t)$ | Linear transformation of the camera pose in timestamp $t$ expressed in world frame coordinates |

$\mathbf{T}_i^{\text{ref}}$ Transformation matrix from camera pose in $t_i$ to camera pose in $t_{\text{ref}}$

$\mathbf{R}_i^{\text{ref}}$ Rotation matrix from camera pose in $t_i$ to camera pose in $t_{\text{ref}}$

$\mathbf{t}_i^{ref}$ Translation vector from camera pose in $t_i$ to camera pose in $t_{\text{ref}}$

$c_x$ Principal point, x coordinate

$c_y$ Principal point, y coordinate

$f_x$ Focal length, x direction

$f_y$ Focal length, y direction

$(k_1, k_2, k_3)$ Radial distortion coefficients

$\mathbf{K}$ $\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$ Camera's intrinsic parameters matrix

$\mathbf{H}_{\rho_j}$ Homography matrix from camera pose in $t_i$ to camera pose in $t_{\text{ref}}$ with $\rho_j$ inverse depth

$\mathbf{G}_{\rho_j}$ Projective homography matrix from camera pose in $t_i$ to camera pose in $t_{\text{ref}}$ with $\rho_j$ inverse depth

$\mathbf{G}_{\rho_j \to \rho_n}$ Projective homography matrix from $\rho_j$ to $\rho_n$ inverse depth parallel planes to camera pose in $t_{\text{ref}}$

$\mathbf{W}$ Image of warped events with the scores expressed in pixel terms

$\mathbf{W}_p$ Image of warped events with the scores expressed in patch terms

Chapter 1

# Introduction

Event cameras are relatively novel sensors that offer several advantages over traditional cameras, especially in high dynamic situations and resource-limited systems due to their high dynamic range, low latency, and low power consumption. Therefore, these new sensors hold great potential for perception applications in fields like mobile robotics, autonomous navigation, augmented reality, and virtual reality.

As will be explained in the following chapters, event cameras provide data differently from traditional cameras, opening up a wide range of possibilities for new research proposals in this field. This work aims to investigate the egomotion estimation of an event camera.

The project has been carried out at the Mobile Robotics and Intelligent Systems group at the Institut de Robòtica i Informàtica Industrial (IRI) a Joint University Research Institute participated by the Spanish National Research Council and the Technical University of Catalonia that conducts basic and applied research in human-centered robotics and automatic control.

Within the Mobile Robotics and Intelligent Systems group at IRI, Borinot was recently built [56]. Borinot stands as an open-source flying robotic platform crafted for agile manipulation and motion. Its adaptable extremities serve as legs for contact-based locomotion, as arms for object manipulation, and as tails during free flight, aiding in dynamic control akin to various animal movements. Currently, Borinot's operation is possible in controlled indoor environments where its positioning is provided by a dedicated multicamera inferred positioning system (optitrack). However, the aim is to extend this operation to unknown outdoor environments. Attempting to solve this problem with traditional perception algorithms becomes nearly impossible given Borinot's highly dynamic agile movements. The Mobile Robotics and Intelligent Systems group has a research line in vision with event cameras. Given the advantages offered by event cameras, they are a promising sensor to work under conditions such as those provided by Borinot. Thus, research is being conducted with event cameras to try to achieve a perception system that, with low resources (without the need for many sensors), can be mounted on Borinot in the future.

In previous works, methods have been implemented that, with event-to-edge associations, have achieved tracking and mapping under very aggressive motion conditions at a kilohertz frame rate [17, 18]. However, these methods are limited to operating in man-made scenes. Currently, the motivation is to extend these functionalities to unknown environments.

## 1.1 Work goals

The aim of the project is to investigate a method for estimating the motion of an event camera (egomotion) in unknown environments and to identify the challenges it faces in order to determine the direction for future research. Therefore, the main purpose is to gain an in-depth understanding of the mathematics and geometry underlying the problem. For this reason, a purely geometric model-based method is studied, without resorting to Artificial Neural Networks (ANNs) as is the current trend in robotics research. ANNs often act as black boxes that cannot be interpreted, and methods incorporating them are typically high power consuming, making them unsuitable for low-resource systems.

## 1.2 Requirements and specifications

As this is the first model-based egomotion estimation for unknown environments method with an event camera to be implemented at IRI-CSIC, there are no specific accuracy or performance requirements. The requirement is to develop a method that allows us to understand the problems and limitations in order to propose future directions by comprehensively understanding the problem from its foundations.

## 1.3 Methods and procedures

As previously mentioned, there is an ongoing research line on event cameras at IRI-CSIC. However, previous methods implemented required the knowledge of the scene. In this case, the goal is to achieve functionality in unknown environments, so the work has been approached and developed from scratch.

Previously, the student completed a curricular internship at IRI-CSIC working with event cameras. Among other tasks, a method for estimating normal flows was implemented in Python. Although this method has been utilized for this work, it has been redeveloped to work with MATLAB and integrate it with the rest of the system.

As explained in detail in Chapter 3, the method comprises three main stages in which inverse depth estimation, normal flow estimation, and a robust linear solver are implemented to achieve linear velocity estimation. Additionally, it is assumed that estimates of angular velocity and camera pose are available, provided by the preintegration of an inertial measurement unit (IMU). All these scenarios are supported by mathematical and geometric foundations that allow for interpretation of the method and understanding of

the problems and limitations it faces.

Since all the work is built from the ground up, tests are conducted for each part of the system to verify its functionality. For this purpose, synthetic examples are used initially, increasing in complexity from the most basic, and finally, a real dataset recorded in the laboratory with Borinot. However, since the project's requirements do not aim to carry out an extensive study of the achieved accuracy and performance, the results analysis (Chapter 4) consists of a more qualitative than quantitative assessment.

## 1.4 Work plan

### 1.4.1 Work plan modifications

Due to time constraints, the work plan has been modified from the one proposed in the bachelor's thesis project proposal. Considering that this is a first implementation of a long-term project, it was difficult to estimate the duration of the tasks. Time has shown us that we were overly optimistic in planning the tasks to be carried out, so the work plan has been adapted.

From Figure 1.2, it can be observed how initially it was intended not only to perform egomotion estimation based on events but also to fuse this estimation into a Kalman filter with measurements made by an IMU. Additionally, once this work in MATLAB was completed, the system was to be implemented in Python to evaluate the algorithm's efficiency and computational cost quantitatively.

As mentioned, the proposal has been developed from scratch, and the need to test the functionality incrementally and address the problems encountered along the way has resulted in more time being taken than initially planned. Nevertheless, the work plan has been adapted without modifying the project's objectives and requirements.

On one hand, the planning and design of the Kalman filter were not part of the tasks to be carried out by the student, but only the implementation in the code to make it work with the rest of the system. Therefore, it was not an essential task, and it was decided to dispense with it. On the other hand, the task of implementing the system in Python consisted only of translating the code done in MATLAB to perform quantitative evaluations. As explained, the objectives of this work do not include performance or precision specifications or requirements since the aim is to understand the problem. For this reason, preference has been given to the development and study of the proposed method, conducting more qualitative evaluations rather than quantitative ones of the results.

### 1.4.2 Final work plan

Just as the system's operation can be divided into three distinct stages, the task organization has been carried out in the same manner. Tasks are divided according to whether they are dedicated to inverse depth estimation, normal flow estimation, or the integration of both estimations into a linear solver to estimate the camera's linear velocity. Additionally, a previous work package is added to these mentioned system stages to research

the state-of-the-art and decide which methods will be attempted for implementation.

Thus, tasks are divided into four distinct work packages (WP): state-of-art research, inverse depth estimation, normal flow estimation, and linear solver (see Figure 1.1). Additionally, as various challenges were encountered during the project, extensive literature research was continuously conducted to identify and implement appropriate solutions.

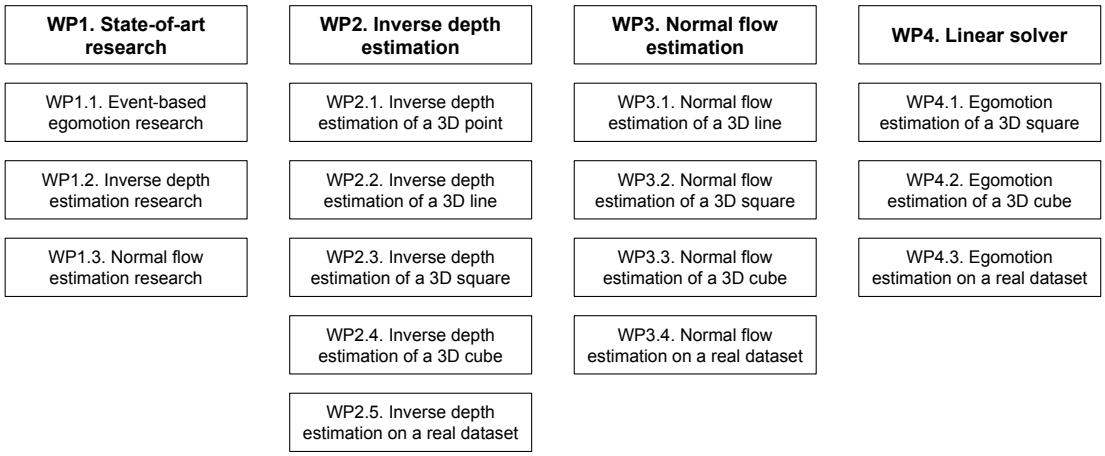



Figure 1.1: Work packages for the tasks organization of the project.

### 1.4.3 Tasks definition

In Tables 1.1, 1.2, 1.3, and 1.4, the definition of the tasks to be carried out during the development of the project can be observed.

| **WP1. State-of-art research** | | |
|---|---|---|
| **Task** | **Description** | **Dates** |
| WP1.1. Event-based egomotion research | Firstly, the state-of-the-art in egomotion estimation with event cameras has been researched. From this task, the proposal to be implemented and the necessary steps to achieve functionality are extracted. | Start: 12/02/2024 End: 16/02/2024 |
| WP1.2. Inverse depth estimation research | As explained, for the egomotion estimation method proposed inverse depth estimation is needed. Therefore, state-of-art research for inverse depth estimation has been conducted. | Start: 17/02/2024 End: 21/02/2024 |
| WP1.3. Normal flow estimation research | Apart from inverse depth, normal flows also need to be estimated. Thus, different alternatives to implement this stage have been studied. | Start: 22/02/2024 End: 24/02/2024 |

Table 1.1: Tasks definition of WP1: State-of-art research.

| WP2. Inverse depth estimation | | |
|---|---|---|
| **Task** | **Description** | **Dates** |
| WP2.1. Inverse depth estimation of a 3D point | To verify that the algorithm is being implemented correctly, we start with the simplest example: implementing inverse depth estimation for a 3D point. | Start: 25/02/2024 End: 10/03/2024 |
| WP2.2. Inverse depth estimation of a 3D line | Once the functionality is verified for a 3D point, we add a bit more complexity by implementing the algorithm for a 3D line. | Start: 11/03/2024 End: 22/03/2024 |
| WP2.3. Inverse depth estimation of a 3D square | In this task, a square is generated to attempt to estimate the inverse depth with more than one edge in the image. | Start: 23/03/2024 End: 29/03/2024 |
| WP2.4. Inverse depth estimation of a 3D cube | Once the functionality of the inverse depth estimation method for more than one edge has been verified, the complexity is increased by implementing the algorithm for a cube. | Start: 30/03/2024 End: 12/04/2024 |
| WP2.5. Inverse depth estimation on a real dataset | Finally, the functionality is verified with a real dataset recorded during a flight with Borinot. | Start: 15/05/2024 End: 20/05/2024 |

Table 1.2: Tasks definition of WP2: Inverse depth estimation.

| WP3. Normal flow estimation | | |
|---|---|---|
| **Task** | **Description** | **Dates** |
| WP3.1. Normal flow estimation of a 3D line | We begin by verifying the simplest example: estimating the normal flow of a single edge. | Start: 13/04/2024 End: 20/04/2024 |
| WP3.2. Normal flow estimation of a 3D square | In this task, the algorithm for normal flow is implemented for more than one edge, with flows in different perpendicular directions. | Start: 21/04/2024 End: 24/04/2024 |
| WP3.3. Normal flow estimation of a 3D cube | The estimation of normal flow is implemented for data coming from a moving cube, so that the directions and magnitudes of the normal flows are more varied. | Start: 25/04/2024 End: 29/04/2024 |
| WP3.4. Normal flow estimation on a real dataset | The performance of the implemented algorithm is verified on a real dataset recorded during a flight with Borinot. | Start: 21/05/2024 End: 03/06/2024 |

Table 1.3: Tasks definition of WP3: Normal flows estimation.

| WP4. Linear solver | | |
|---|---|---|
| **Task** | **Description** | **Dates** |
| WP4.1. Egomotion estimation of a 3D square | The linear solver is tested using the estimates of normal flow and inverse depth to obtain egomotion. | Start: 30/04/2024 End: 07/05/2024 |
| WP4.2. Egomotion estimation of a 3D cube | Since the example with the square is limited to very basic motion patterns, the linear solver is implemented using the example of the cube. | Start: 08/05/2024 End: 14/05/2024 |
| WP4.3. Egomotion estimation on a real dataset | Finally, the algorithm's performance is tested on a real dataset recorded during a flight with Borinot. | Start: 22/05/2024 End: 03/06/2024 |

Table 1.4: Tasks definition of WP4: Linear solver.

### 1.4.4 Initial Gantt diagram

In Figure 1.2, the initial Gantt diagram proposed at the beginning of the project can be visualized. As it can be observed, the final work packages were different since the intention was to implement the egomotion estimation method in Python as well.

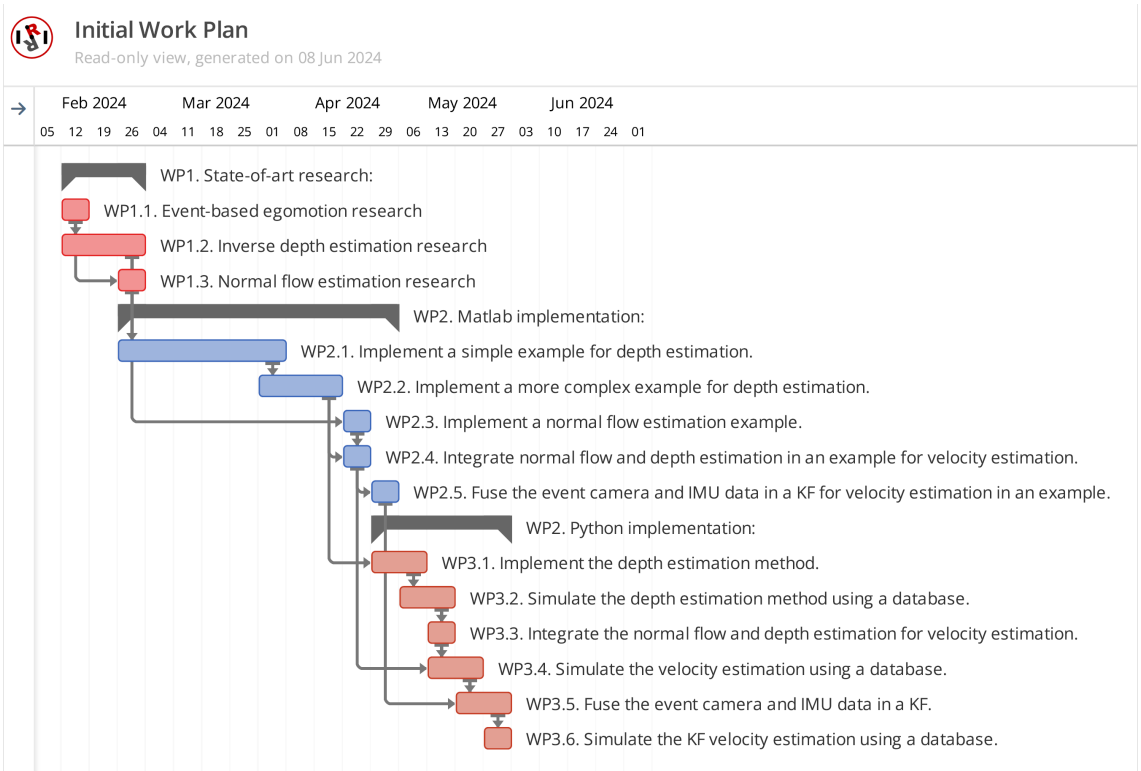



Figure 1.2: Gantt diagram corresponding the initial work plan.

### 1.4.5 Final Gantt diagram

Once the modifications explained to adapt the work plan have been applied, the Gantt diagram is shown in Figure 1.3. It can be observed that the MATLAB implementation work package from the initial Gantt chart (Figure 1.2) has been divided into three according to the stage of the system to which the tasks correspond. Additionally, it can be seen that the tasks are scheduled to finish in early June, leaving the remaining period for the writing of the project report.

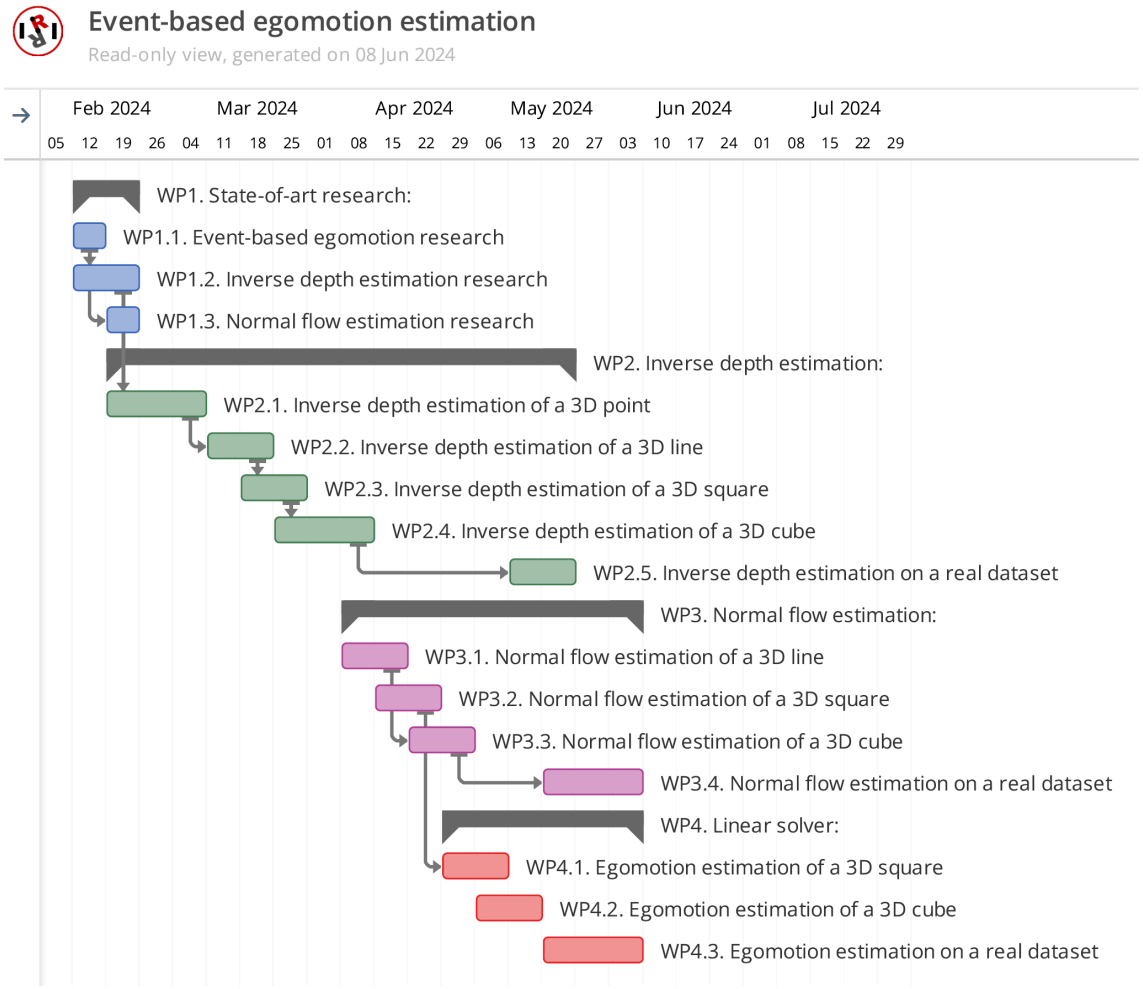



Figure 1.3: Gantt diagram corresponding the final work plan.

Chapter 2

# Background

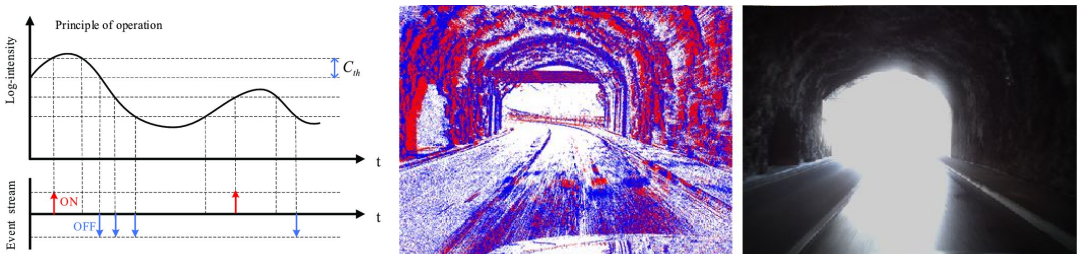
Event cameras are truly innovative sensors that first appeared on the market in 2008 [49]. Therefore, it is not common to be familiar with their operation. This chapter aims to provide the reader with the necessary preliminary knowledge to understand how event cameras work, their advantages, and the state of art in research with these sensors. Within the wide range of applications for event cameras, we will focus on those related to the work developed: optical flow (Section 2.2), depth estimation (Section 2.3), and egomotion estimation (Section 2.4).

## 2.1 Event cameras

Event cameras are bio-inspired sensors that capture visual information asynchronously, responding to the dynamics of the scene. They are designed to respond to changes in brightness in the scene with very low latency (around $1\,\mu s$). The captured events are transmitted as a stream of information with address event representation (AER). In the AER representation, each event contains information about the $x$,$y$ position of the pixel, a timestamp $t$, and the polarity $p$. It is worth mentioning that unlike traditional cameras, there is no transmission of brightness value information for each pixel. However, the polarity provides binary information about whether the brightness change that triggered the event was positive or negative.

When the received events are integrated over a period of time, the data can be easily visualized to create an image-like representation with microsecond resolution (see Figure 2.1). Thus, events are triggered when a logarithmic change in brightness exceeds a threshold with polarity ON ("1") for increments and OFF ("0") for decrements. This way, event cameras avoid redundant information and facilitate real-time operation in applications for robotics, drones, navigation, and autonomous vehicles, among others.

Although event cameras have become available relatively recently [49], latter literature on these new sensors, as well as mass production plans by some companies, highlight a great interest in exploiting these novel vision sensors. Since the information provided by event cameras differs from that of traditional cameras, traditional computer vision algorithms cannot be directly applied. This opens up the possibility of a new and exten-




(a) (b) (c)

Figure 2.1: Event generation principles. (a) Event generation of events with ON or OFF polarity; (b) Event integration during some ms, ON events represented in red and OFF events represented in blue; (c) Traditional image frame of the scene. Figure adapted from [31, 34].

sive research area aimed at surpassing the milestones achieved with classical computer vision algorithms.

### 2.1.1 Event camera bio-inspiration

In the literature, event cameras are often referred to as bio-inspired sensors. This is because their operation is inspired by how the human eye perceives and processes light [54]. Like the human eye, these cameras are designed to detect changes in the scene asynchronously and respond only to relevant stimuli, rather than capturing and transmitting complete images at regular time intervals.

In the human eye, photoreceptor cells called rods and cones are responsible for detecting light and sending signals to the brain through the optic nerve. These photoreceptor cells are highly sensitive to changes in light intensity and respond quickly to visual stimuli, allowing for real-time visual perception and efficient detection of movements and changes in the environment.

Similarly, event cameras use a bio-inspired approach by employing a Dynamic Vision Sensor (DVS) that functions analogously to rods and cones in the human eye. Instead of capturing complete images at fixed time intervals, the DVS sensor detects changes in the scene pixel by pixel and sends signals asynchronously and selectively when a significant change in light intensity occurs.

This capacity to detect events in real-time and respond efficiently only to relevant stimuli makes event cameras particularly useful in applications where high sensitivity to rapid movements and changes in the scene is required.

### 2.1.2 Benefits of event cameras

Event cameras offer several significant advantages compared to traditional cameras, making them especially useful in a variety of applications. These benefits can be categorized into three main aspects: high dynamic range (HDR), low latency, and low power consumption.

Event cameras inherently possess a HDR, which enables them to operate effectively in challenging lighting conditions where conventional cameras often struggle. Traditional cameras capture images at fixed intervals, leading to either overexposure or underexposure in scenes with varying light intensities. In contrast, event cameras detect changes in the intensity of individual pixels asynchronously, allowing them to adapt to both extremely bright and dark regions within the same scene. This capability ensures that details are preserved across a wide range of lighting conditions, making event cameras particularly advantageous in environments with significant contrasts in illumination and adverse lighting conditions.

Another significant benefit of event cameras is their asynchronous operation and low latency. Event cameras record changes in the scene at the exact moment they occur. This continuous event-driven acquisition results in extremely low latency, enabling the capture of rapid movements at $\mu$s rate without motion blur where conventional cameras often fail.

Additionally, the asynchronous nature of event cameras also leads to significantly lower power consumption compared to conventional cameras. Traditional cameras consume substantial power due to the need to process entire frames at regular intervals, regardless of whether there is significant information change in the scene. Event cameras, on the other hand, only transmit information when there is a change in the pixel intensity, resulting in much lower data rates and reduced computational requirements. This efficiency makes event cameras particularly suitable for embedded systems and applications where power consumption is a critical concern. The reduced need for constant data processing not only conserves energy but also facilitates real-time processing on platforms with limited computational resources.

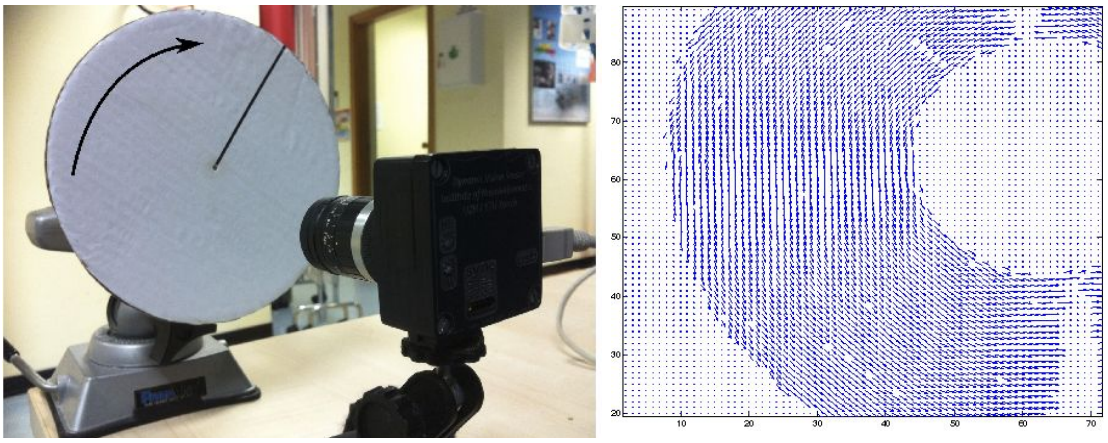
## 2.2 Optical flow

The analysis of motion within image sequences is a cornerstone in the field of computer vision, enabling a wide range of applications such as object tracking and robotic navigation. Optical flow, a field deeply rooted in computer vision, plays a pivotal role in this endeavor. It refers to the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between the observer (camera) and the scene (see Figure 2.2). Understanding optical flow provides crucial insights into the dynamics of a scene, facilitating the extraction of valuable information from image sequences.

### 2.2.1 Aperture problem

Despite its importance, optical flow estimation encounters challenges, one of which is known as the aperture problem. The aperture problem refers to the inherent limitation in optical flow estimation, where only component-wise motion perpendicular to the local edge orientation (namely normal flow) can be accurately recovered. Understanding the aperture problem is crucial for developing robust optical flow algorithms capable of handling complex visual scenes effectively.

The aperture problem arises from the ambiguity in estimating motion along an edge when only local information is available (Figure 2.3). In regions where the visual stim-



(a) (b)

Figure 2.2: Optical flow representation. (a) Scene in movement of which optical flow is determined, a line rotating; (b) Representation of the optical flow of the line where the blue arrows indicate the direction estimated. Figure adapted from [10].

ulus is highly directional, such as edges or contours, traditional optical flow methods struggle to accurately determine the true motion direction. This is because the local image gradient, which serves as the basis for optical flow computation, provides information only about the component of motion perpendicular to the edge orientation. As a result, estimating motion parallel to the edge becomes inherently challenging, leading to inaccuracies and ambiguities in optical flow estimation. Given that event cameras naturally respond to edges, this problem is even more limiting.

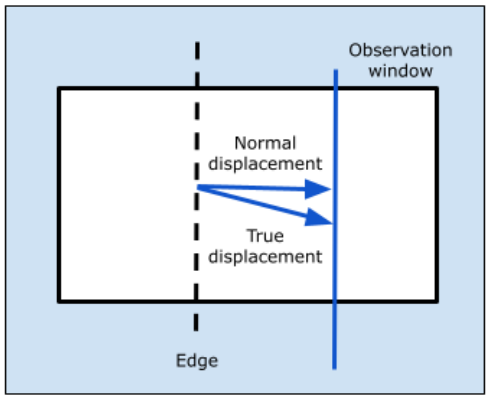



Figure 2.3: Representation of the aperture problem. It can be seen that due to the local information of the edge, only the normal displacement can be estimated from the observation window.

### 2.2.2 Event-based optical flow

Within the literature, various methods have been proposed for event-based optical flow estimation. In general, event-based optical flow methods can be classified as packet-based or event-by-event, depending on how events are processed and how the estimated

value is updated.

Packet-based methods process a batch of events within a fixed time window or a fixed number of events, requiring a waiting time before initiation. Previous works have proposed adaptations of classic frame-based methods with block-matching [50] or Lucas-Kanade [11]. Others have sought to leverage the characteristics of data provided by event cameras with spatio-temporal plane-fitting [10], time-surface matching [58], and contrast maximization methods [29, 73]. These are model-based methods that achieve their results by interpreting the problem and solving it through optimization. However, there are also packet-based methods with ANNs [85, 86, 32, 23, 45]. The current state of the art is led by ANNs due to their accuracy in optical flow estimation [73, 32], but on the contrary they are highly power-consuming methods and lack interpretability.

On the other hand, event-by-event methods process each event incrementally as it arrives, aiming to leverage the low latency of event cameras. Many event-by-event methods are inspired by the brain, as visual processing circuits are believed to be event-by-event [72]. For this type of methods, Spiking Neural Networks (SNNs) [77, 36, 62, 64, 76, 83] are commonly used, which attempt to simulate the brain's neuronal architecture and contain parameters that are difficult to interpret. However, their performance is still uncomparable with that of ANNs as they suffer from training difficulties.

## 2.3 Depth estimation

In the field of computer vision, depth estimation plays a pivotal role in extracting three-dimensional (3D) information from two-dimensional (2D) images or video sequences. Depth estimation involves determining the distance of objects to the camera. Depth estimation provides crucial spatial information that enhances the understanding and interpretation of visual data. By estimating depth, computer vision systems can perceive the geometric layout of a scene, identify object boundaries, assess spatial relationships between objects, and facilitate tasks such as object tracking, scene reconstruction, and environment mapping.

Traditional depth estimation methods rely on stereo matching, structure from motion, or depth sensors such as LiDAR or RGB-D cameras. While these methods have been successful in many scenarios, they often suffer from limitations such as sensitivity to lighting conditions, textureless regions, and computational complexity. Additionally, depth sensors can be bulky, expensive, and power-hungry, restricting their use in certain applications.

The ability to estimate depth from monocular images or video streams has gained significant attention due to its applicability in scenarios where only a single camera is available or where depth sensors are not feasible. This capability is particularly valuable for applications such as autonomous driving, robotics navigation, virtual reality content creation, and medical imaging, where precise depth information is essential for accurate perception and decision-making.

### 2.3.1 Event-based depth estimation

The emergence of event-based cameras presents a promising alternative for depth estimation tasks. Due to their unique operating principles, event cameras are particularly well-suited for depth estimation in challenging conditions, such as high-speed motion, dynamic scenes, and varying lighting conditions.

Most depth estimation works based on event data use two or more fixed event cameras, namely stereo methods. In these approaches, events are accumulated over a short period of time to exploit the advantages of event cameras in reconstructing dynamic scenes at high speed. Generally, they follow the classical two-step paradigm [37], where event correspondences across different image planes are sought, followed by triangulation to obtain the 3D point position [43, 70, 13]. Matching events across different sensors typically involve imposing constraints, often seeking temporal coherence by exploiting simultaneity and temporal correlation between sensors [41, 65, 28, 63]. Given the rather null distinctiveness of single events, it is extremely difficult to establish event correspondances from two separate cameras.

On the other hand, there are also some monocular approaches (with a single event camera). Since there is no simultaneity of events, there is no temporal coherence. For this reason, in these cases, it is necessary to know the camera motion, and these methods are usually used to address the problem of egomotion estimation or SLAM (Simultaneous Localization and Mapping). In some works, additional sensors are used to solve the problem, such as a CMOS camera [15], or an RGB-D camera [78]. In [42], a method based on a pipeline using three filters operating in parallel is proposed; however, it also needs to estimate image intensity to resolve depth. Adapting the classical space-sweep method [21] for event data, [66, 68] achieves 3D scene reconstruction in real time. However, the resulting depth is discretized between the minimum and maximum values, limiting it to known environments. Finally, in [29], depth estimation is proposed by aligning events to the trajectory, aiming to maximize the contrast of an image of warped events (IWE) created by translating sets of events from their individual timestamps to a common reference timestamp.

On the other hand, depth estimation methods with ANNs have also been implemented [39, 82], which achieve denser depth maps than model-based methods. Once more, methods using ANNs are difficultly interpretable geometrically and highly power-consuming. In our work we will study the use of contrast maximization for depth estimation.

## 2.4 Egomotion estimation

Egomotion estimation, also known as visual odometry (VO), is a crucial task in computer vision and robotics. It involves estimating the motion of a moving camera relative to its surroundings using visual information. Traditionally, egomotion estimation has been tackled using a combination of sensors, including cameras, inertial measurement units (IMUs) and LiDAR sensors to combine 2D data from the image plane with 3D information.

Achieving efficient egomotion estimation is crucial in various applications such as autonomous navigation, augmented reality, virtual reality, and robotics. This is because by accurately tracking the camera's motion, systems can make informed decisions and navigate through complex environments.

### 2.4.1 Event-based egomotion estimation

In recent years, there has been growing interest in leveraging event-based cameras for egomotion estimation. These novel sensors offer several advantages, including high temporal resolution, low latency, and low power consumption. However, integrating event-based cameras into egomotion estimation frameworks poses unique challenges due to differences in data representation and processing compared to traditional cameras.

While there are methods for camera tracking [30, 12] that have proven to be effective in achieving aggressive egomotion estimation, VO estimation systems have not always yielded good results. Early works using event cameras [42, 69] require very smooth movements for initializing local 3D maps. In order to address this restriction, stereo vision with event cameras has been proposed in [84] to improve efficiency and accuracy, but it fails in violent movements. To overcome this limitation, visual-inertial fusion and feature-based methods have been employed. The IMU can provide a prior estimation of the state. In [67], this motion prior is used to obtain a corrected IWE based on the estimated motion. However, since the warp operation depends on the median depth of the local map, it may fail if there are large depth variations. This dependency is addressed in some works [35, 20] by establishing feature association in image-like representations of events [44]. However, these representations may be erroneous when there are sudden linear velocity changes.

Using first-order kinematic dependency, [55, 33] proposes a set of velocity-invariant representations leading to edge patterns of the same thickness regardless of the camera velocity. However, despite many attempts of event-based feature detection and tracking [33, 3, 2, 22], many of them can not perform in real time or do not achieve comparable results to their traditional vision counterparts. Nevertheless, a few works [17, 18, 19] have succeeded in establishing event-to-edge associations in situations of aggressive movements at an ultra-frame rate but are limited to known environments.

On the other hand, [81] proposes a velometer that estimates linear velocity from angular velocities provided by an IMU and the estimation of normal flows and depth. This proposal has achieved good results on synthetic datasets. However, this method requires two event cameras to estimate depth in stereo. Finally, in a very recent work [74], a framework based on contrast maximization is proposed, achieving competitive results in the state of the art in terms of accuracy for egomotion estimation.

Thus, it can be said that except for the recent work [74], there are no methods that achieve effective egomotion estimation in unknown environments with a single event camera using real data.

Chapter 3

# Methodology

As previously mentioned, the main aim of this project is to explore the egomotion estimation from an event camera for unknown environments and to comprehend the underlying mathematical principles governing this challenge. Therefore, rather than resorting to neural networks that function as black boxes, the methodology employed is grounded in geometric principles based on the model. This approach ensures a transparent understanding of the processes involved, steering away from opaque solutions and delving into the geometric foundations that drive the problem-solving framework.

The methodology outlined in this chapter is structured to provide a comprehensive breakdown of the steps taken to derive egomotion from the event camera data. It begins with an overview of the general framework (Section 3.1), followed by a detailed exposition of each stage in the methodology (Sections 3.2, 3.3, 3.4 and 3.5).

## 3.1 General framework

This section aims to explain the general structure of the method employed. To understand the proposed framework, it is important to present the motion field equation since the entire algorithm revolves around solving this equation to obtain egomotion.

### 3.1.1 Motion field equation

The motion flow equation, first introduced in [51], demonstrates that the optical flow of a pixel at a given timestamp is determined solely by the first-order kinematics of the camera and the 3D information.

$$\begin{aligned}\begin{pmatrix}\dot{x}(t)\\ \dot{y}(t)\end{pmatrix} &= \begin{pmatrix}\frac{v_z x' - f v_x}{Z} + \frac{\omega_x}{f} x' y' - \omega_y \left(f + \frac{x'^2}{f}\right) + \omega_z y' \\ \frac{v_z y' - f v_y}{Z} + \omega_x \left(f + \frac{y'^2}{f}\right) - \frac{\omega_y}{f} x' y' - \omega_z x'\end{pmatrix} \\ &= \frac{1}{Z(\mathbf{x})} \mathbf{A}(\mathbf{x}) \mathbf{v}^B(t) + \mathbf{B}(\mathbf{x}) \boldsymbol{\omega}^B(t)\end{aligned} \tag{3.1}$$

where $\dot{\mathbf{x}} = [\dot{x}, \dot{y}]^\top$ denotes the optical flow at the pixel $\mathbf{x} = [x, y]^\top$, $\mathbf{x}' = [x', y']^\top = [x - c_x, y - c_y]^\top$ the relative image coordinates with respect to the principal point $[c_x, c_y]^\top$, $Z(\mathbf{x})$ the depth of the pixel, $f$ the focal length, $\boldsymbol{\omega}^B = [\omega_x, \omega_y, \omega_z]^\top$ the angular velocity and $\mathbf{v}^B = [v_x, v_y, v_z]^\top$ the linear velocity.

However, as mentioned in Section 2.2.1, only the normal component to the edge of the optical flow can be obtained due to the aperture problem. To address this issue, in Section 3.5, the motion flow equation is adapted for normal flows. Thus, assuming that the camera's angular velocity $\boldsymbol{\omega}^B$ is obtained from an IMU, only the depth $Z(\mathbf{x})$ and normal flow $\dot{\mathbf{x}}_\mathbf{n}$ of at least three pixels are needed to calculate the linear velocity.

### 3.1.2 System diagram

In Figure 3.1, a general overview of the system operation is depicted. Firstly, the camera calibration parameters are utilized to rectify event distortions (Section 3.2). Subsequently, events are used for estimating an inverse depth map (as discussed in Section 3.3, requiring pose estimations from an IMU preintegration stage) and normal flows (Section 3.4). Finally, leveraging the inverse depth map, normal flows, and angular velocity provided by an IMU, a linear solver (Section 3.5) is implemented to yield the camera's linear velocity.

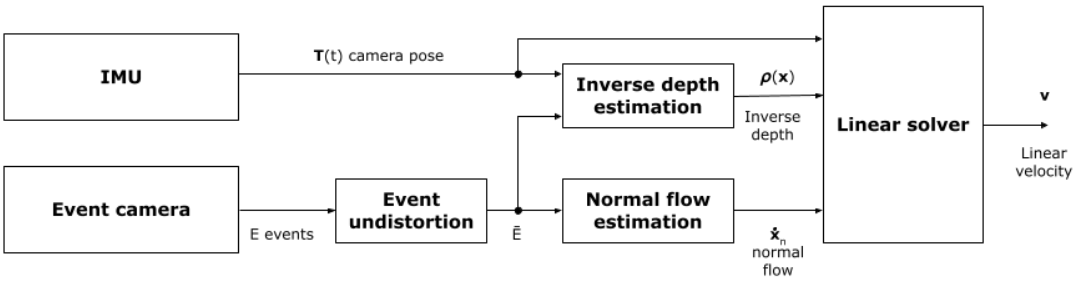



Figure 3.1: System diagram of the proposed linear velocity solver.

### 3.1.3 Implementation

For the implementation and testing of the method, MATLAB has been utilized. This choice stems from the objective of gaining a deep understanding of the mathematics underlying the problem. Consequently, the entire implementation has been carried out from scratch, leveraging geometry without resorting to learning. This approach ensures that each equation used is thoroughly validated. MATLAB is a versatile software that facilitates testing every aspect of the algorithm, from simpler to more complex examples. Hence, it has been deemed the optimal choice for method development.

## 3.2 Event undistortion

Similarly to traditional cameras, event cameras also exhibit distortion due to their lenses. With default 1:1:6 CS mount lenses, the distortion is primarily radial.

From camera calibration, focal length parameters $(f_x, f_y)$, the camera's principal point $[c_x, c_y]^\top$, and radial distortion coefficients $(k_1, k_2, k_3)$ are obtained. Utilizing these parameters, an exact formula is proposed in [25] to eliminate radial distortion for each pixel. This enables the creation of a lookup table containing the undistorted coordinates of each pixel. The process of eliminating distortion for each event with coordinates $\mathbf{x} = [x, y]^\top$ occurs in three steps:

1. Depixelation:

$$\mathbf{x}_{\text{dpx}} = \begin{cases} x_{\text{dpx}} = (x - c_x)/f_x \\ y_{\text{dpx}} = (y - c_y)/f_y \end{cases} \tag{3.2}$$

2. Undistortion:

$$\mathbf{x}_{\text{und}} = \mathbf{x}_{\text{dpx}}(1 - k_1||\mathbf{x}_{\text{dpx}}||^2 + (3k_1^2 - k_2)||\mathbf{x}_{\text{dpx}}||^4 + (8k_1k_2 - 12k_1 - k_3)||\mathbf{x}_{\text{dpx}}||^6) \tag{3.3}$$

3. Pixelation:

$$\mathbf{x}_{\text{new}} = \begin{cases} x_{\text{new}} = f_x \cdot x_{\text{und}} + c_x \\ y_{\text{new}} = f_y \cdot y_{\text{und}} + c_y \end{cases} \tag{3.4}$$

In the following sections, it is assumed that the events used are undistorted.

## 3.3 Inverse depth estimation

As discussed in Section 3.1, it is necessary to determine depth to solve the motion field equation. In other words, the aim is to obtain the distance of each pixel from the camera. However, as demonstrated in [16] (*Chapter 4, pages 122-123*), the depth optimization problem is better conditioned if instead of estimating depth $Z$, its inverse $\rho = \frac{1}{Z}$ is estimated. Therefore, the inverse depth will be estimated.

### 3.3.1 Contrast maximization approach

The estimation of inverse depth is carried out following the approach proposed in [29]. In this work, various computer vision problems with event cameras are addressed by maximizing image contrast.

Event cameras only report brightness change at pixels. Assuming constant illumination in the scene, for the camera to trigger events, there must be relative motion between the camera and the objects in the scene, along with sufficient texture for a brightness gradient to exist. Thus, event cameras respond to the apparent motion of edges. Consequently, edges depict trajectories in the image plane, and events are fired along these trajectories. The contrast maximization approach aims to find the trajectory that best fits the event data. If the trajectory is parameterized in terms of the variable to be estimated, the optimal value is the one that best aligns the events at a specific timestamp (maximizing image contrast).

### 3.3.2 Contrast maximization for inverse depth estimation

Consider an event camera moving in a static scene. As the camera moves, a set of events $\mathrm{E} = \{e_k\}_{k=1}^{N_e}$ is generated. Additionally, it is assumed that the linear transformation matrix $\mathbf{T}(t)$ describing the pose of the event camera with respect to the coordinate axis is known after an IMU preintegration stage. The objective is to estimate the inverse depth at a reference time.

The method iterates on the following three steps starting with an initial guess for the inverse depth at each pixel:

1. (a) Transfer events triggered at different timestamps to the reference time using the current inverse depth estimate for each pixel.
   (b) Create an IWE by counting the events that have fallen on each pixel at the reference time.
2. Measure the contrast of the IWE.
3. Revise candidate inverse depth of each pixel to maximize the contrast.

In Figure 3.2, a simple example of the proposed scenario is depicted: a camera moving in front of an object. Two events corresponding to the same texture are transferred from their timestamps to the reference time, $t_{\text{ref}}$, using three candidate inverse depths $(\rho_1, \rho_2, \rho_3)$. It can be observed that in the correct inverse depth $(\rho_2)$, the points align correctly.

#### Event transfer to reference time

The first step of the algorithm involves transferring the events occurred at different timestamps to the reference time for a candidate inverse depth. The events are transferred using the planar homography [37] (*Chapter 13*) induced by a plane parallel to the image plane of the reference view at the candidate inverse depth (see Figure 3.2).

Since the poses $\mathbf{T}_i = \mathbf{T}(t_i)$ are known, the linear transformation between each pose and the reference pose $\mathbf{T}_{\text{ref}} = \mathbf{T}(t_{\text{ref}})$ can be determined:

$$\mathbf{T}_i^{\text{ref}} = \mathbf{T}_{\text{ref}}^{-1}\mathbf{T}_i = (\mathbf{R}_i^{\text{ref}}|\mathbf{r}_i^{\text{ref}}) \tag{3.5}$$

where $\mathbf{T}_i^{\text{ref}}$ denotes the linear transformation matrix between the pose at time $t_i$ and the reference time $t_{\text{ref}}$, $\mathbf{R}_i^{\text{ref}}$ is the rotation matrix, and $\mathbf{t}_i^{\text{ref}}$ is the translation vector.

Thus, we can obtain the homography matrix $\mathbf{H}_{\rho_j}$ and the projective homography matrix $\mathbf{G}_{\rho_j}$:

$$\mathbf{H}_{\rho_j} = \mathbf{H}_i^{\text{ref}}(\rho_j) = \mathbf{R}_i^{\text{ref}} + \rho_j \mathbf{t}_i^{\text{ref}} \cdot \mathbf{n}^\top \tag{3.6}$$

$$\mathbf{G}_{\rho_j} = \mathbf{G}_i^{\text{ref}}(\rho_j) = \mathbf{K}\mathbf{H}_{\rho_j}\mathbf{K}^{-1} \tag{3.7}$$

where $\mathbf{n} = [0, 0, 1]^\top$ is the normal vector to the image plane at the reference time, and $\mathbf{K}$ is the calibration matrix containing the intrinsic parameters of the event camera. The homography matrix $\mathbf{H}_{\rho_j}$ transfers the event that occurred in the image plane with

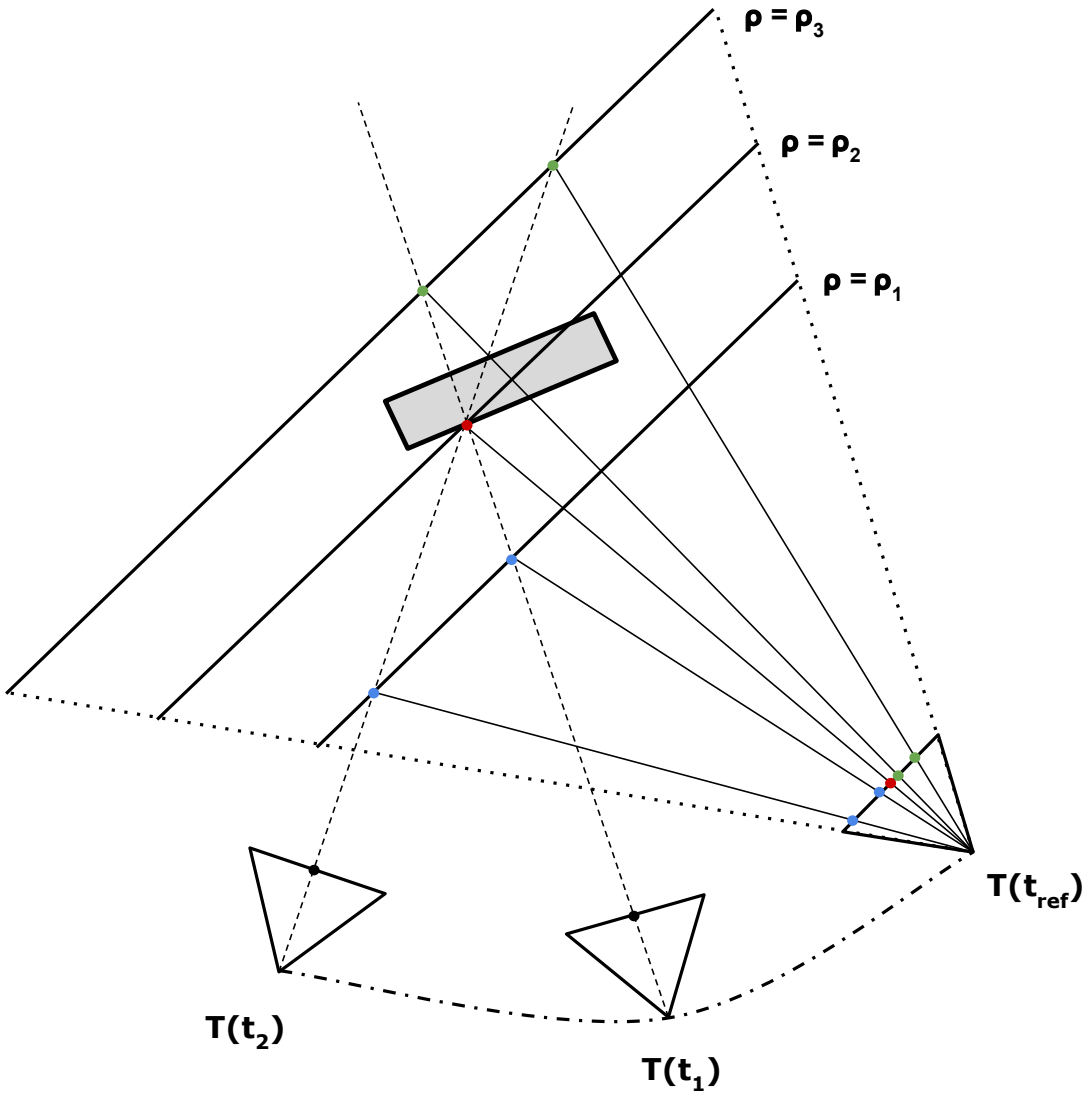



Figure 3.2: Simple example in which the alignment of warped events for three different inverse depth candidates can be observed. For the near ($\rho_1$) and far ($\rho_3$) cases, the events are misaligned (blue and green points). When the inverse depth is correct, the events are aligned and fall in the same red point.

coordinates $\mathbf{x} = [x, y]^\top$ at time $t_i$ to the plane parallel to the image plane at time $t_{\text{ref}}$ with a distance $1/\rho_j$. In other words, it converts the 2D coordinates from the image plane to 3D coordinates. The projective homography matrix projects the 3D point onto the image plane at the reference time to obtain the 2D coordinates of the warped event, $\mathbf{x}_w = [x_w, y_w]^\top$, using the camera calibration matrix $\mathbf{K}$.

Hence, to find the coordinates of the warped event:

$$\underline{\mathbf{x}}_w = \mathbf{G}_{\rho_j} \underline{\mathbf{x}} \tag{3.8}$$

where the operator $\underline{\cdot}$ denotes the homogeneous coordinates of the pixel $\underline{\mathbf{x}} = [x, y, 1]^\top$.

The optimization method used is iterative, which means it is necessary to transfer

events to other planes with different inverse depths. In [21], a mathematical arrangement is proposed that allows transferring events from the plane with distance $1/\rho_j$ to others with distance $1/\rho_n$:

$$\mathbf{G}_{\rho_j \to \rho_n} = \mathbf{K}\mathbf{H}_{\rho_n}\mathbf{H}_{\rho_j}^{-1}\mathbf{K}^{-1} \tag{3.9}$$

The homography matrices $\mathbf{H}_{\rho_n}$ and $\mathbf{H}_{\rho_j}$ can be calculated as in Eq. 3.6. Expressing the homography matrices in column-wise form:

$$\mathbf{G}_{\rho_j \to \rho_n} = \mathbf{K}\left[\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3 + \rho_n \mathbf{t}_i^{\text{ref}}\right]\left[\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3 + \rho_j \mathbf{t}_i^{\text{ref}}\right]^{-1}\mathbf{K}^{-1} \tag{3.10}$$

where $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$ are the column vectors of the rotation matrix $\mathbf{R}_i^{\text{ref}}$. Utilizing the property of orthonormal vectors $[\mathbf{a}, \mathbf{b}, \mathbf{c}]^{-1} \sim [(\mathbf{b} \times \mathbf{c}), (\mathbf{c} \times \mathbf{a}), (\mathbf{a} \times \mathbf{b})]^{\top}$, the result is:

$$\mathbf{G}_{\rho_j \to \rho_n} = \begin{pmatrix} 1 & 0 & \delta C_x \\ 0 & 1 & \delta C_y \\ 0 & 0 & 1 + \delta C_z \end{pmatrix} \tag{3.11}$$

where $\delta = \frac{\rho_n - \rho_j}{1 - \rho_i C_z}$ and $\mathbf{C} = \left[C_x, C_y, C_z\right]^T = -\mathbf{R}_{\text{ref}}^{i\top}\mathbf{t}_{\text{ref}}^{i} = \mathbf{t}_i^{\text{ref}}$. Consequently, events can be transferred from one plane to another (see Figure 3.3) very quickly, making the algorithm more efficient than if we had to perform the homography for each of the planes.

$$\underline{\mathbf{x}}_w(\rho_n) = \mathbf{G}_{\rho_j \to \rho_n}\underline{\mathbf{x}}_w(\rho_j) \tag{3.12}$$

where $\underline{\mathbf{x}}_w(\rho_n)$ denotes the warped event for a plane with depth $1/\rho_n$, and $\underline{\mathbf{x}}_w(\rho_j)$ denotes the warped event for a plane with depth $1/\rho_j$.

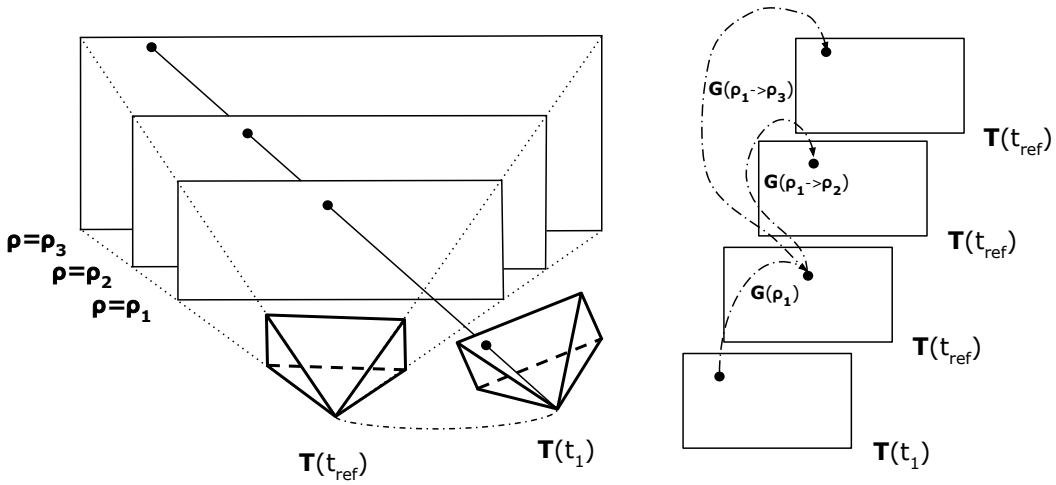



Figure 3.3: Ray back-projection process. First, an event triggered in $t_1$ is transferred to the reference time $t_{\text{ref}}$ using the projective homography $\mathbf{G}_{\rho_1}$ in the $Z = 1/\rho_1$ plane as explained in Eq. 3.8. Then, the warped event is back-projected to other planes using $\mathbf{G}_{\rho_1 \to \rho_{2,3}}$ (Eq. 3.12).

Even so, a large number of events arrive in a single message, and optimizing the inverse depth for each pixel is computationally expensive. For this reason, patches of $5 \times 5$ pixels are created such that pixels in the same patch have the same assigned inverse depth value. With this, the spatial resolution of the inverse depth map is reduced and the algorithm becomes more efficient.

### Pose interpolation

Event cameras are sensors with extremely high temporal resolution (approximately $1\,\mu s$). Consider an IMU preintegration that provides camera poses every $\sim 100\,\mathrm{ms}$. In order to implement the proposed algorithm, it is necessary to calculate intermediate poses between two poses estimated by the IMU to assign a pose to each timestamp.

Aiming to reduce the number of intermediate poses to calculate and avoid interpolating as many poses as events we have, timestamps for inverse depth estimation are discretized. Each timestamp within the window of $[t_i - t_{\text{disc}}/2, t_i + t_{\text{disc}}/2]$ is assigned the value $t_i$ and the pose correspondent to this timestamp. From this section onwards, it is assumed that the timestamps of the events are discretized for the inverse depth estimation algorithm. For more information on how the interpolation between two known poses is computed, please refer to Appendix A.

### Image of warped events

Once the needed tools to transfer events from their corresponding timestamps to the reference time are clear, the next step is to create the image of warped events. As explained in Section 3.3.2, this involves counting the warped events that have fallen on each pixel. In other words, each warped event casts a vote at its assigned pixel coordinate.

However, a voting scheme where each warped event only votes for a single pixel complicates the optimization problem. It only reports whether the vote is in the correct pixel or not, rather than how close it is to the correct pixel. Therefore, in [29], it is proposed to vote with a bivariate Gaussian distribution based on the distance to the warped event assigned pixel.

The Gaussian function is flat at points far from its maximum and steeper at points closer to it. This means that the gradient is large for points close to the maximum and small for points that are far away. For gradient-based optimization algorithms like the one explained in the next section, we have witnessed that this poses a problem when trying to converge to the maximum of the function.

For this reason, we propose instead to vote between 0 and 1 with a quadratic distribution in a $5 \times 5$ pixel patch with the pixel assigned to the warped event in the center. As can be seen in Figure 3.4, unlike the Gaussian distribution, this function is steeper at points far from the maximum and flatter at points close to it.

Thus, we define the bidimensional voting function $v(x_w, y_w, x_i, y_i)$:

$$\begin{cases} v_x(x_w, x_i) = -\left[\frac{1}{5/2}(x_i - x_w)\right]^2 + 1 \\ v_y(y_w, y_i) = -\left[\frac{1}{5/2}(y_i - y_w)\right]^2 + 1 \end{cases}$$

$$v(x_w, y_w, x_i, y_i) = \begin{cases} v_x(x_w, x_i) v_y(y_w, y_i) & \text{if } |x_i - x_w| < 5/2, |y_i - y_w| < 5/2 \\ 0 & \text{if others} \end{cases}$$

where $v(\mathbf{x}_w, \mathbf{x}_i)$ indicates the score assigned to the pixel with coordinates $\mathbf{x}_i = [x_i, y_i]^\top$ when the warped event is assigned the pixel with coordinates $\mathbf{x}_w = [x_w, y_w]^\top$.

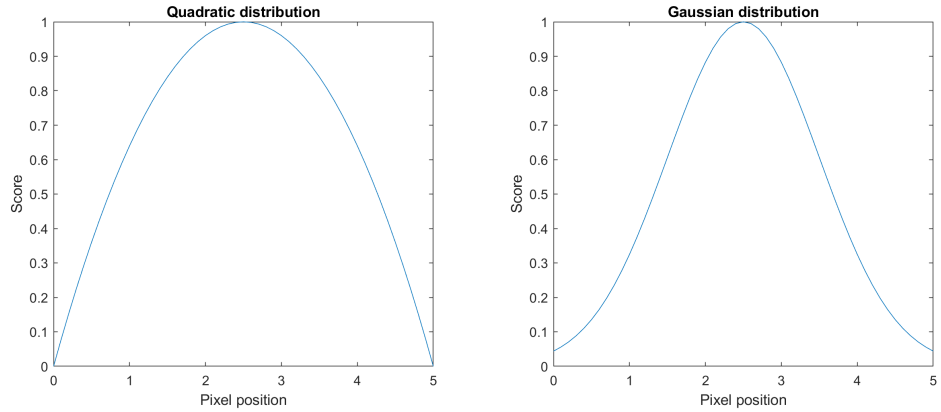



Figure 3.4: Gaussian and quadratic distributions for voting. The distributions are represented in 2 dimensions for simplicity.

In this way, following the steps described in Algorithm 1, an IWE can be created.

**Algorithm 1** Creation of an IWE.

```
for every patch in which an event has been triggered do
    for every event e_k in E do
        Warp e_k using planar homography
        Vote with quadratic distribution in the IWE
    end for
end for
```

### Optimization

For the optimization of the inverse depth of each $5 \times 5$ patch, the Root Mean Square Propagation (RMSProp) [40] gradient-based algorithm is used. The objective is to find the inverse depth of each patch that maximizes the contrast of the IWE. Therefore, the gradient of the contrast of the IWE with respect to the inverse depth of the patch, $\frac{\partial C_{\mathbf{W}}}{\partial \rho}$, must be calculated.

Firstly, the contrast is defined as the variance of the IWE:

$$C_{\mathbf{W}} = \sigma^2(\mathbf{W}) = \frac{1}{N_{pix}} \sum_{i=1}^{N_{pix}} (w_i - \mu_{\mathbf{W}})^2 \tag{3.13}$$

$$\mu_{\mathbf{W}} = \frac{1}{N_{pix}} \sum_{j=1}^{N_{pix}} w_j \tag{3.14}$$

where $\mathbf{W}$ denotes the matrix containing the scores of each pixel in the IWE, $\sigma^2(\mathbf{W})$ is the variance of $\mathbf{W}$, $N_{\text{pix}}$ is the number of pixels in the image, $w_i$ is the score of pixel $i$ in the IWE, and $\mu_{\mathbf{W}}$ is the mean value of $\mathbf{W}$.

However, these expressions are given in terms of the score and inverse depth of each pixel. Recall that it has been decided to optimize the inverse depth assigned to patches

of $5 \times 5$ pixels, so the expression of $C_{\mathbf{W}}$ must be adapted to express it in terms of patches. With this objective in mind, the resolution of the IWE is reduced such that each patch is assigned the mean value of the scores of the pixels contained within it.

$$w_{p,i} = \frac{1}{N} \sum_{p \in P_i} w_p \tag{3.15}$$

where $w_{p,i}$ is the score of the patch $P_i$, $N$ the number of pixels in the patch $P_i$ and $p$ iterates each pixel in $P_i$.

By reducing the resolution, the expression for the contrast of the IWE remains the same as in Eq. 3.13 but expressed in terms of the scores of the patches:

$$C_{\mathbf{W}_p} = \sigma^2(\mathbf{W}_p) = \frac{1}{N_p} \sum_{i=1}^{N_p} (w_{p,i} - \mu_{\mathbf{W}_p})^2 \tag{3.16}$$

$$\mu_{\mathbf{W}_p} = \frac{1}{N_p} \sum_{j=1}^{N_p} w_{p,j} \tag{3.17}$$

where $\mathbf{W}_p$ denotes the matrix containing the scores of each of the patches, $\sigma^2(\mathbf{W}_p)$ is the variance of $\mathbf{W}_p$, $N_p$ is the number of patches in the image, $w_{p,i}$ is the score of patch $P_i$, and $\mu_{\mathbf{W}_p}$ is the mean value of $\mathbf{W}_p$.

Substituting the expression for the mean value (Eq. 3.17) into Eq. 3.16:

$$C_{\mathbf{W}_p} = \frac{1}{N_p} \sum_{i=1}^{N_p} (w_{p,i} - \frac{1}{N_p} \sum_{j=1}^{N_p} w_{p,j})^2 = \frac{1}{N_p} \sum_{i=1}^{N_p} \left[ (1 - \frac{1}{N_p}) w_{p,i} - \frac{1}{N_p} \sum_{j \neq i} w_{p,j} \right]^2 \tag{3.18}$$

It is observed that the expression for $C_{\mathbf{W}_p}$ does not have a direct dependence on $\rho$. However, the score of each patch is related to the inverse depth. Therefore, we suggest splitting the gradient into two parts using the chain rule:

$$\frac{\partial C_{\mathbf{W}_p}}{\partial \rho_i} = \frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}} \frac{\partial w_{p,i}}{\partial \rho_i} \tag{3.19}$$

We propose to perform the first derivative ($\frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}}$) analytically and the second derivative ($\frac{\partial w_{p,i}}{\partial \rho_i}$) numerically. By analytically differentiating the expression for $C_{\mathbf{W}_p}$ in Eq. 3.18 with respect to the score of patch $P_i$ and grouping terms, we obtain a very simple expression (for more information on the gradient development, please refer to Appendix B):

$$\frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}} = \frac{2}{N_p} (w_{p,i} - \mu_{\mathbf{W}_p}) \tag{3.20}$$

Thus, knowing the IWE for the previous candidate inverse depth, the gradient corresponding to each patch can be calculated as follows:

$$\frac{\partial C_{\mathbf{W}_p}(k)}{\partial w_{p,i}(k)} = \frac{2}{N_p} (w_{p,i} - \mu_{\mathbf{W}_p}(k)) \cdot \frac{w_{p,i}(k) - w_{p,i}(k-1)}{\rho_i(k) - \rho_i(k-1)} \tag{3.21}$$

where $k$ and $k-1$ indicate the optimization iteration number corresponding to the parameter.

Once known how to calculate the gradients, the implementation the RMSProp optimization algorithm [71, 40] follows, which, as its name suggests, is based on the propagation of the root mean square value of the gradient. The steps to implement the algorithm are described in Algorithm 2.

**Algorithm 2** Inverse depth optimization for a single patch using RMSProp.

**Initialize** `RMSProp parameters:` $\delta$, $p$, $r_0$, $\alpha$, $n_{max}$
**while** `number of iterations performed` $n < n_{max}$ **do**
  **Accumulate** `the gradient` $g_n$ `of the patch in` $r_{n+1}$`:`
  $r_{n+1} = p \cdot r_n + (1-p) g_n^2$
  **Compute** `the next increment of inverse depth of the patch:`
  $\triangle\rho_{n+1} = \frac{\alpha}{\delta + \sqrt{r_{n+1}}} g_n$
  **Update** `the inverse depth value:`
  $\rho_{n+1} = \rho_n + \triangle\rho_{n+1}$
  **Increment** `number of iterations performed:`
  $n = n + 1$
**end while**

The algorithm described in Algorithm 2 for a single patch is easily scalable to all patches in the image where an event has occurred implementing the parameters $r_{n,n+1}$, $g_n$, $\rho_{n,n+1}$ and $\triangle\rho_{n+1}$ as vectors. Hence, the inverse depths of all patches can be optimized simultaneously.

### Iterative execution of the algorithm

In the previous section, it is explained how to optimize the inverse depth of each patch in an arbitrary iteration, given undistorted events and two camera poses. However, before starting the optimization algorithm, two candidate inverse depths are needed to create two IWEs so that the gradient can be calculated.

In the first iteration of all (with the first two poses), two IWEs are created by initializing the depth inverse map to two different values (all patches with the same value). From here, the inverse depth values of each patch are optimized to maximize contrast. The first time the depth inverse map is initialized, it is sufficient to do it with a commonsense value (so the warped events vote in the image and generate a non-zero gradient), since as will be seen in Chapter 4, the algorithm converges quickly with few iterations.

When more than one optimization process has been completed (beyond the first two poses), the same process is decided to be carried out. However, instead of initializing the depth inverse map with two arbitrary values, they will be initialized with the minimum and mean value of the previous estimated inverse depth map.

Another alternative would be to translate the inverse depth map given the camera poses using planar homographies. However, it has been observed that when changing the perspective from which the scene is viewed, there are few previous patches that

remain in the image plane. The patches that remain in the image plane are assigned the inverse depth estimated in the last pose, but the rest must be initialized. It has been seen that this actually does not improve the efficiency of the algorithm and entails a greater computational cost than directly performing the initialization of all patches. For this reason, this alternative has been discarded.

Finally, it is worth mentioning that, as explained in Algorithm 3, the timestamp of the last received pose is used as the reference time to have the most recent inverse depth map possible. Additionally, only the patches in which an event has occurred in the final pose (reference time) are optimized.

**Algorithm 3** Inverse depth estimation.

**if** a new pose $\mathbf{T}(t_i)$ has arrived from the IMU preintegration **then**
  **Update** the reference time $t_{ref} = t_i$.
  **Collect** all the events $\mathrm{E} = \{e_k\}_{k=1}^{N_e}$ triggered in $[t_{i-1}, t_i]$.
  **Collect** all the patches $\mathrm{P} = \{p_n\}_{n=1}^{N_p}$ where events in $\mathrm{E}$ have been triggered.
  **Discretize** the event timestamps and get the intermediate poses between $\mathbf{T}(t_{i-1})$ and $\mathbf{T}(t_i)$.
  **if** $t_i = t_1$ (it is the first iteration) **then**
    **Initialize** two initial inverse depth maps with two arbitrary values $\rho_0$ and $\rho_1$.
  **else**
    **Initialize** two initial inverse depth maps with the values $\rho_0 = \min(\boldsymbol{\rho}(i-1))$ and $\rho_1 = \mathrm{mean}(\boldsymbol{\rho}(i-1))$.
  **end if**
  **Warp the events** in $\mathrm{E}$ to $t_{ref}$ using $\rho_0$ for all the patches in $\mathrm{P}$ using Eq. 3.8, save the warped events in $\mathrm{E}_w = \{e_{w,k}\}$ and **create an IWE** (Algorithm 1).
  **Warp the events** in $\mathrm{E}_w$ using $\rho_1$ for all patches in $\mathrm{P}$ using Eq. 3.12 and **create an IWE**.
  **Compute** the gradients $\mathbf{g}_0$ for every patch in $\mathrm{P}$.
  **Optimize** the inverse depth map as explained in Algorithm 2.
**end if**

## 3.4 Event-based normal flow

As mentioned in Section 2.2, different methods have been proposed for estimating normal flow. For this work, we have chosen to calculate them using local-plane fitting of Surfaces of Active Events [10]. This is because it is a low-cost method with a clear geometric interpretation.

### 3.4.1 Surface of Active Events

The Surface of Active Events (SAE) is defined in the 3D space-time domain, consisting of the two dimensions of the image plane of the event camera and an additional dimension

representing time. Each incoming event generates or updates a point on the surface so that each pixel is assigned the value of the timestamp of the last event that occurred at this position. Therefore, we can define the mapping function $S(\mathbf{x})$ that returns the timestamp of the last event that occurred at $\mathbf{x}$:

$$S(\mathbf{x}) = t_{\text{last}} \tag{3.22}$$

where $\mathbf{x} = [x, y]^\top$ is the position of the pixel and $t_{\text{last}}$ is the timestamp of the last event that occurred at $\mathbf{x}$.

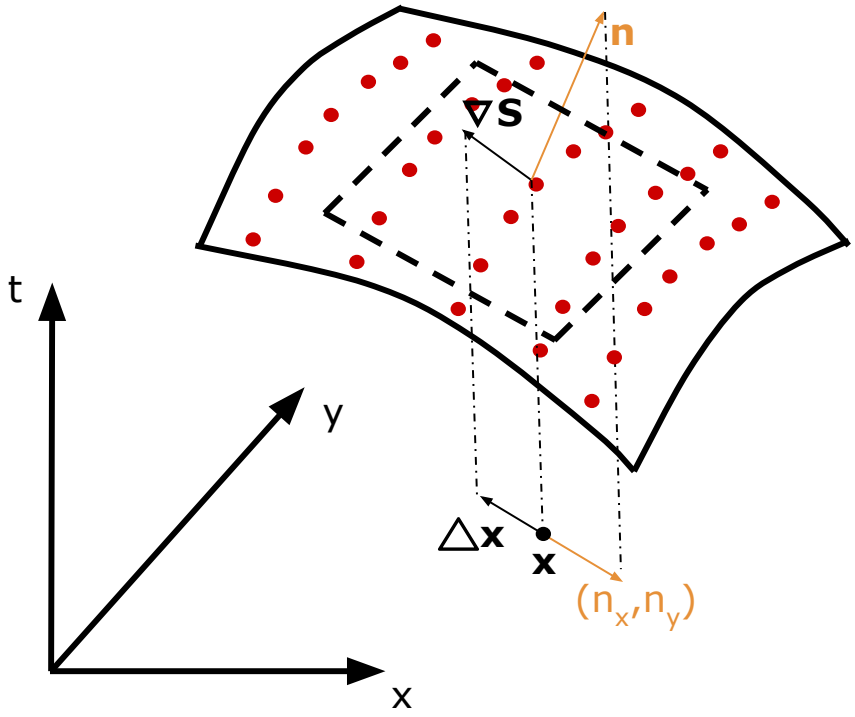



Figure 3.5: Representation of a SAE formed by the events generated by an edge depicted as red points and its planar approximation.

The planar approximation of the SAE at an event with position $\mathbf{x}$ is defined by its first-order Taylor polynomial:

$$S(\mathbf{x} + \triangle\mathbf{x}) \approx S(\mathbf{x}) + \nabla S^\top(\mathbf{x})\triangle\mathbf{x} \tag{3.23}$$

where $\triangle\mathbf{x} = [\triangle x, \triangle y, 0]$ denotes the pixel displacement, and $\nabla S(\mathbf{x}) = \left[\frac{\partial t}{\partial x}, \frac{\partial t}{\partial y}, 1\right]^\top = \left[S_x, S_y, 1\right]^\top$ is the gradient of $S(x, y)$ with respect to $[\mathbf{x}, t]^\top$. Thus, the normal flow $\dot{\mathbf{x}}_n = [\dot{x}_n, \dot{y}_n]$ is defined by the following expression:

$$\mathbf{x}_n = \frac{\triangle\mathbf{x}}{\partial t} \tag{3.24}$$

In Figure 3.5, the vector $\mathbf{n} = \left[n_x, n_y, n_t\right]^\top$ is represented, which is the normal vector to the planar approximation of the SAE. Thus, $S_x$ and $S_y$ can be expressed in terms of

the components of the normal vector:

$$S_x = -\frac{n_x}{n_t} \tag{3.25}$$

$$S_y = -\frac{n_y}{n_t} \tag{3.26}$$

Therefore, if the components of the normal vector to the plane are found, the normal flow can be calculated. To do this, we need to find the plane that best fits the events generated by the edge.

### 3.4.2 Local-plane fitting algorithm

To perform local-plane fitting and find the normal vector $\mathbf{n}$ to the SAE, a least squares minimization implementation is proposed. To begin with, an event $\mathbf{e} = [x, y, t]$ belongs to the plane if it satisfies the condition shown in Eq. 3.27.

$$n_x x + n_y y + n_t t + d = \mathbf{\Pi}^\top \begin{pmatrix} x \\ y \\ t \\ 1 \end{pmatrix} = 0 \tag{3.27}$$

Where $\mathbf{\Pi} = \begin{bmatrix} n_x, n_y, n_t, d \end{bmatrix}^\top$ denotes the vector containing the parameters of the plane.

The objective is to find the vector $\tilde{\mathbf{\Pi}}$ that minimizes the distance from the events to the plane:

$$\tilde{\mathbf{\Pi}} = \text{argmin}_{\mathbf{\Pi} \in \mathbb{R}^4} \sum_{i=1}^{N_e} |\mathbf{\Pi}^\top \begin{pmatrix} x_i \\ y_i \\ t_i \\ 1 \end{pmatrix}|^2 = \text{argmin}_{\mathbf{\Pi} \in \mathbb{R}^4} f(\mathbf{\Pi}, \mathbf{x}, \mathbf{t}) \tag{3.28}$$

where $\tilde{\mathbf{\Pi}} = \begin{bmatrix} \tilde{n}_x, \tilde{n}_y, \tilde{n}_t, \tilde{d} \end{bmatrix}$ is the vector of plane parameters that minimizes the distance to the plane of the events in $\mathrm{E} = \{e_i\}_{i=1}^{N_e}$.

To attempt to minimize the function $f(\mathbf{\Pi}, \mathbf{x}, \mathbf{t})$, its derivative is calculated and equated to 0:

$$\frac{\partial f}{\partial \tilde{n}_x} = 2(\sum_{i=1}^{N_e} \tilde{n}_x x_i^2 + \tilde{n}_y x_i y_i + \tilde{n}_t x_i t_i + \tilde{d} x_i) = 0 \tag{3.29}$$

$$\frac{\partial f}{\partial \tilde{n}_y} = 2(\sum_{i=1}^{N_e} \tilde{n}_x x_i y_i + \tilde{n}_y y_i^2 + \tilde{n}_t y_i t_i + \tilde{d} y_i) = 0 \tag{3.30}$$

$$\frac{\partial f}{\partial \tilde{n}_t} = 2(\sum_{i=1}^{N_e} \tilde{n}_x x_i t_i + \tilde{n}_y y_i t_i + \tilde{n}_t t_i^2 + \tilde{d} t_i) = 0 \tag{3.31}$$

$$\frac{\partial f}{\partial \tilde{d}} = 2(\sum_{i=1}^{N_e} \tilde{n}_x x_i + \tilde{n}_y y_i + \tilde{n}_t t_i + \tilde{d}) = 0 \tag{3.32}$$

From Eq. 3.32, it can be deduced that $\tilde{d} = -\frac{1}{N_e} \sum_{i=1}^{N_e} \tilde{n}_x x_i + \tilde{n}_y y_i + \tilde{n}_t t_i$. Defining the mean value of each dimension as $\mu_x = \frac{1}{N_e} \sum_{i=1}^{N_e} x_i$, $\mu_y = \frac{1}{N_e} \sum_{i=1}^{N_e} y_i$, and $\mu_t =$

$\frac{1}{N_e}\sum_{i=1}^{N_e} t_i$, and substituting into Eqs. 3.29, 3.30, and 3.31, obtain the following system of equations is obtained:

$$\begin{cases} \tilde{n}_x \sum_{i=1}^{N_e}(x_i^2 - x_i\mu_x) + \tilde{n}_y \sum_{i=1}^{N_e}(x_i y_i - x_i\mu_y) + \tilde{n}_t \sum_{i=1}^{N_e}(x_i t_i - x_i\mu_t) = 0 \\ \tilde{n}_x \sum_{i=1}^{N_e}(x_i y_i - y_i\mu_x) + \tilde{n}_y (\sum_{i=1}^{N_e}(y_i^2 - y_i\mu_y) + \tilde{n}_t \sum_{i=1}^{N_e}(y_i t_i - y_i\mu_t) = 0 \\ \tilde{n}_x \sum_{i=1}^{N_e}(x_i t_i - t_i\mu_x) + \tilde{n}_y \sum_{i=1}^{N_e}(y_i t_i - t_i\mu_y) + \tilde{n}_t \sum_{i=1}^{N_e}(t_i^2 - t_i\mu_t) = 0 \end{cases} \tag{3.33}$$

Thus, the matrix $\mathbf{A}$ and the vector $\tilde{\mathbf{n}}$ are defined such that:

$$\mathbf{A}\tilde{\mathbf{n}} = \begin{pmatrix} \sum_{i=1}^{N_e}(x_i^2 - x_i\mu_x) & \sum_{i=1}^{N_e}(x_i y_i - x_i\mu_y) & \sum_{i=1}^{N_e}(x_i t_i - x_i\mu_t) \\ \sum_{i=1}^{N_e}(x_i y_i - y_i\mu_x) & \sum_{i=1}^{N_e}(y_i^2 - y_i\mu_y) & \sum_{i=1}^{N_e}(y_i t_i - y_i\mu_t) \\ \sum_{i=1}^{N_e}(x_i t_i - t_i\mu_x) & \sum_{i=1}^{N_e}(y_i t_i - t_i\mu_y) & \sum_{i=1}^{N_e}(t_i^2 - t_i\mu_t) \end{pmatrix} \begin{pmatrix} \tilde{n}_x \\ \tilde{n}_y \\ \tilde{n}_t \end{pmatrix} = 0 \tag{3.34}$$

To solve this problem and find the vector $\tilde{\mathbf{n}}$ that best fits the plane, we perform Singular Value Decomposition (SVD) of $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. The vectors are arranged so that the third row vector of $\mathbf{V}^\top$ is associated with the smallest singular value, and therefore, it is the best solution for the minimization problem. Thus, $\tilde{\mathbf{n}}$ is the third row vector of $\mathbf{V}^\top$.

To perform plane fitting, spatiotemporal windows of 5pix$\times$ 5pix$\times$30ms are used. That is, for each event that has been the last to occur in its pixel, we construct a SAE. This event is centered in the spatio-temporal window and, along with the other events within the window, apply the local-plane fitting algorithm. To make this algorithm more robust against noise, it is only attempted to fit the plane if the spatio-temporal window contains a number of events greater than the parameter *minhits*, as a significant number of events should be triggered if an edge has indeed passed through that window. Additionally, it is verified that the solution of the SVD is well-conditioned by checking that the magnitude of the singular values does not vary significantly, as significant variation would indicate numerical instability.

### 3.4.3 Normal flow estimation

Finally, once the normal vector to the planar approximation of the SAE, $\mathbf{n}$, has been found, the normal flow vector $\dot{\mathbf{x}}_n$ can be obtained. Substituting the expressions from Eq. 3.25, 3.26 into the planar approximation of the SAE (Eq. 3.23) and isolating $n_t$:

$$n_t = -(n_x\dot{x}_n + n_y\dot{y}_n) \tag{3.35}$$

We have one equation and two unknowns, $\dot{\mathbf{x}}_n = [\dot{x}_n, \dot{y}_n]$. To calculate the magnitude of the normal flow, we need to project the normal vector $\mathbf{n}$ onto the image plane, as depicted in Figure 3.5.

The vector connecting $\mathbf{x}$ and $\triangle\mathbf{x}$ is perpendicular to the edge projected onto the image plane and has a slope of $\frac{n_y}{n_x}$ (see Figure 3.5). Therefore, from the equation of this vector and scaling by $\partial t$:

$$(y + \triangle y) - y = \frac{n_y}{n_x}\left[(x + \triangle x) - x\right] \tag{3.36}$$

$$\frac{1}{\partial t}\left[\triangle y = \frac{n_y}{n_x}\triangle x\right] \Longrightarrow \dot{y}_n = \frac{n_y}{n_x}\dot{x}_n \tag{3.37}$$

Substituting equation 3.37 into equation 3.35 and isolating terms yields the expression for the normal flow to the edge:

$$\begin{aligned} \dot{x}_n &= -\frac{n_t n_x}{n_x^2 + n_y^2} \\ \dot{y}_n &= -\frac{n_t n_y}{n_x^2 + n_y^2} \end{aligned} \tag{3.38}$$

With all that explained, Algorithm 4 describes step by step the algorithm for estimating normal flows.

**Algorithm 4** Normal flow estimation.

**if** a new message of events $\mathrm{E} = \{e_k\}_{k=1}^{N_e}$ arrives **then**
  **Save** the timestamp of the last event triggered in each pixel in $\mathrm{E}_{\mathtt{last}} = \{e_{\mathtt{last,k}}\}$.
  **for** every event $e_{\mathtt{last,k}}$ in $\mathrm{E}_{\mathtt{last}}$ **do**
    **Count** the number events in $\mathrm{E}_{\mathtt{last}}$ contained in a window of 5pix×5pix×30ms with $e_{\mathtt{last,k}}$ in the center.
    **if** the number of events in the window $>$ minhits **then**
      **Build** a SAE and compute its normal vector with local-plane fitting.
      **if** the SVD solution is well-conditioned **then**
        **Compute** the normal flow associated to $e_{\mathtt{last,k}}$.
      **end if**
    **end if**
  **end for**
**end if**

### 3.4.4 Multi-spatial scale maxpooling

In Section 4.1.2 it is observed that poorly conditioned normal flows are obtained when tested on real data. To address this issue, an attempt is made to improve the consistency of the estimated normal flows using the multi-spatial scale maxpooling approach proposed in [1].

For the implementation of this method, the first step is to estimate the normal flows in the manner proposed in Algorithm 4. Next, a set of neighborhoods $S = \{\sigma_k\}$ is defined, centered on the spatiotemporal coordinates of the normal flow to be corrected $[x, y, t]^\top$, with an increasing spatial neighborhood size and a temporal window $\left[t - t_{past}, t + t_{past}\right]$. For each neighborhood, the mean value of the normal flow magnitude and the direction angle $\theta$ within it are calculated, as shown in Eq. 3.39 and 3.40.

$$\mu_{||\dot{\mathbf{x}}_n||_k} = \frac{\sum_{i \in \sigma_k} ||\dot{\mathbf{x}}_n||_i}{n_k} \tag{3.39}$$

$$\mu_{\theta_k} = \frac{\sum_{i \in \sigma_k} \theta_i}{n_k} \tag{3.40}$$

where $n_k$ is the number of normal flows present in the neighborhood $\sigma_k$, and $\theta_i$ is the angle indicating the direction of the estimated normal flow, given by $\arctan(\dot{y}_{n,i}/\dot{x}_{n,i})$.

Thus, the normal flow value considered most consistent is that of the neighborhood with the highest average magnitude. Finally, knowing the average direction corresponding to that magnitude, the best-conditioned normal flow is obtained using Eq. 3.41.

$$\dot{\mathbf{x}}_{n_{\sigma_{max}}} = \left[\mu_{||\dot{\mathbf{x}}_n||_{max}} cos(\mu_{\theta_{max}}), \mu_{||\dot{\mathbf{x}}_n||_{max}} sin(\mu_{\theta_{max}})\right]^\top \tag{3.41}$$

where the subscript $max$ refers to the best-conditioned normal flow.

This way, Algorithm 5 summarizes the normal flow estimation method using multi-spatial scale maxpooling.

**Algorithm 5** Normal flow estimation with multi-spatial scale maxpooling.

**for** each event $e = [x, y, t]^\top$ in E **do**
  **Compute** local flow (Algorithm 4)
  **Multi-spatial scale maxpooling:**
  **Define** $S = \{\sigma_k\}$, the set of neighborhoods, centered on $[x, y, t]^\top$, $\sigma_k$ with increasing radius and temporal window $\left[t - t_{past}, t + t_{past}\right]$.
  **if** $\dot{\mathbf{x}}_n \neq [0, 0]^\top$ **then**
    **for** each $\sigma_k \in S$ **do**
      **Compute** mean normal flow magnitude and direction angle, Eq. 3.39 and 3.40.
    **end for**
    $\sigma_{max} = argmax_{\sigma_k \in S}(\mu_{||\dot{\mathbf{x}}_n||_k})$
  **end if**
  **Update** normal flow, Eq. 3.41.
**end for**

For the neighborhoods in the multi-spatial scale maxpooling method, the parameters proposed in [1] are used: a spatial range $\sigma$ from 0 to 100 pixels in steps of 10 and a temporal limit $t_{past}$ of 5 ms.

## 3.5 Linear solver for egomotion estimation

As explained in previous sections, the ultimate goal is to estimate the camera egomotion by solving the motion field equation 3.1. However, instead of having optical flow, we have normal flow. In [81], it is proposed to adapt the motion field equation for normal flows. This is achieved by multiplying both sides of the equation by the normalized normal flow vector $\hat{\dot{\mathbf{x}}}_n(t) = \frac{\dot{\mathbf{x}}_n(t)}{||\dot{\mathbf{x}}_n(t)||}$, where the operator $||\cdot||$ refers to the Euclidean norm.

$$\hat{\dot{\mathbf{x}}}_n(t)^\top \dot{\mathbf{x}}(t) = \rho(\mathbf{x}) \hat{\dot{\mathbf{x}}}_n(t)^\top \mathbf{A}(\mathbf{x}) \mathbf{v}^B(t) + \hat{\dot{\mathbf{x}}}_n(t)^\top \mathbf{B}(\mathbf{x}) \boldsymbol{\omega}^B(t) \tag{3.42}$$

The optical flow vector can be decomposed into the sum of the normal component and the component parallel to the edge, $\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}_n(t) + \dot{\mathbf{x}}_{\|}(t)$. Therefore, the adapted motion field scalar equation for normal flows is as follows:

$$||\dot{\mathbf{x}}_n(t)|| = \rho(\mathbf{x}) \hat{\dot{\mathbf{x}}}_n(t)^\top \mathbf{A}(\mathbf{x}) \mathbf{v}^B(t) + \hat{\dot{\mathbf{x}}}_n(t)^\top \mathbf{B}(\mathbf{x}) \boldsymbol{\omega}^B(t) \tag{3.43}$$

Now, instead of having two constraints (one for each optical flow direction) per pixel, we only have one. The equation is scalar.

As mentioned earlier, we only have three unknowns: the three components of linear velocity. From each event message, many estimates of normal flow are extracted. Furthermore, most of them belong to a patch where the inverse depth has been estimated. Therefore, an overdetermined system of equations could be set up with all this data to find the egomotion. However, the estimated normal flows are subject to noise due to method limitations, and in some cases, they may be ill-conditioned. To address this issue, a linear solver with robust outlier removal based on RANSAC (Random Sampling Consensus) [80] is implemented for more robust sampling.

RANSAC is an iterative method for estimating the parameters of a mathematical model from a set of observed data containing outliers. The data is classified into inliers, i.e., data whose distribution is explained by a set of model parameters, and outliers, which are data points that do not fit the model. Algorithm 6 outlines the linear solver based on RANSAC for finding egomotion.

**Algorithm 6** Linear solver for egomotion estimation.

**RANSAC parameters**:
k: maximum number of iterations.
n: minimum number of normal flow and depth estimates to fit the model (3 in our case as we have three unknown parameters).
t: threshold value to determine if a point fits the model.
d: number of close data values to assert that a model fits well the data.
iterations = 0
bestVel = Null
bestErr = $\infty$
**while** iterations $<$ k **do**
  maybeInliers = n randomly selected values from data.
  maybeVel = velocity fitted to maybeInliers with least squares.
  alsoInliers = empty set.
  **for** every point in data not in maybeInliers **do**
    **if** the pointError < th **then**
      **Add** the point to alsoInliers.
    **end if**
  **end for**
  **if** the number of points in alsoInliers is > d **then**
    betterVel = velocity fitted to maybeInliers and alsoInliers with least squares.
    thisErr = measure how well the model fits these points.
    **if** thisErr $<$ bestErr **then**
      bestVel = betterVel
      bestErr = thisErr
    **end if**
  **end if**
  iterations = iterations + 1
**end while**

Chapter 4

# Results

In this chapter, the results obtained from the various experiments conducted are presented and analyzed. The analysis of these results enables decisions to be made regarding future directions of work.

## 4.1 Experiments and Tests

The performance of the proposed methods for each stage of the system has been tested in various experiments, ranging from basic examples to understand all underlying principles to more complex ones using synthetic data, as well as a dataset recorded during a manual free flight with Borinot. Both the implementation of the methods and the data reading and visualization of results were conducted using MATLAB.

### 4.1.1 Inverse depth estimation

The inverse depth method has been implemented from scratch, and therefore, it has been tested with different examples starting from the most basic:

- Inverse depth estimation for a point.
- Inverse depth estimation for a line.
- Inverse depth estimation for a polyhedron.
- Inverse depth estimation using a dataset recorded during a manual free flight with Borinot.

For evaluating the performance of the inverse depth estimation method, the evolution of the inverse depths will be visualized across the optimization iterations of RMSProp, along with a heatmap of the IWE to observe how the votes align.

#### RMSProp parameters

Before presenting the obtained results, it is important to outline the values assigned to the parameters used in RMSProp Algorithm 2. Firstly, $\delta$ is given the typical value of

$10^{-8}$. Instead of the typical value of $0.9$, $p$ is assigned a value of $0.7$ to place slightly more importance on the most recent accumulated gradient. Lastly, the learning rate $\alpha$ is set to $0.01$.

$$\begin{cases} \delta = 10^{-8} \\ p = 0.7 \\ \alpha = 0.01 \end{cases}$$

### Inverse depth estimation for a point

First, the inverse depth estimation for a point placed at a known distance in 3D space is implemented. This is a synthetic example, so the data to be processed must be generated. To generate the events, two poses are defined: the initial pose and the final pose of the camera. This allows defining a motion pattern. The events are generated by interpolating $k$ poses between the two defined poses and projecting the 3D point whose coordinates are known onto the image plane of each of the generated poses. To do this, it is necessary to calculate the projection matrix of each pose:

$$\mathbf{T}_{3\times4} = \begin{pmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_1 \\ R_{2,1} & R_{2,2} & R_{2,3} & t_2 \\ R_{3,1} & R_{3,2} & R_{3,3} & t_3 \end{pmatrix} \tag{4.1}$$

$$\mathbf{P} = \mathbf{K}\mathbf{T}_{3\times4} \tag{4.2}$$

where $\mathbf{T}_{3\times4}$ is the linear transformation matrix of the pose, $R_{i,j}$ are the elements of the rotation matrix, $t_i$ are the elements of the translation vector, $\mathbf{K}$ is the camera calibration matrix, and $\mathbf{P}$ is the perspective projection matrix of the pose.

Thus, knowing the coordinates $\mathbf{p} = \begin{bmatrix} p_x, p_y, p_z \end{bmatrix}^\top$ of the 3D point for which the inverse depth is to be estimated, its position on the image plane for a specific pose can be obtained.

$$\underline{\mathbf{x}} = \mathbf{P}\underline{\mathbf{p}} \tag{4.3}$$

where $\underline{\mathbf{x}}$ denotes the position of the point projection on the image plane in homogeneous coordinates. Therefore, events are generated by saving each of the obtained projections for each pose and assigning them the corresponding timestamp of the associated pose.

To demonstrate the functionality of the method, it is tested on two different motion patterns and placing the point at different 3D coordinates (see Table 4.1). Additionally, the algorithm's robustness is demonstrated by testing it with both favorable and unfavorable initial conditions ($\rho_0$ and $\rho_1$).

| Example | Inverse depth | Linear velocity | Angular velocity | num. events |
|---|---|---|---|---|
| 1 | 0.1667 | $(10, 18, -15)$ m/s | $(0, 2.618, -1.745)$ rad/s | 200 |
| 2 | 0.25 | $(-5, 10, -5)$ m/s | $(1.745, 2.618, -0.873)$ rad/s | 200 |

Table 4.1: Table with the examples information of the inverse depth estimation for a 3D point.

The camera poses and the point in the 3D space are illustrated in Figure 4.1. The cameras are depicted as polyhedrons, the 3D point as a cross, and its projection onto the image planes as circular points. It is worth noting that in these examples, the origin pose is used as the reference time, meaning the votes should accumulate at the image plane coordinates where we see the point projected. Since these are synthetic examples we have created, the ground truth value for each patch being optimized can be determined. In this example, since only one point is projected, a single patch is optimized. As mentioned earlier, the inverse depth estimation algorithm is tested on two different motion patterns and point positions. Additionally, it is demonstrated that the proposed gradient information achieves good results both in cases with favorable initial conditions (providing a correct and close initial direction to the ground truth, as seen in Figures 4.2 and 4.4) and unfavorable initial conditions (Figures 4.3 and 4.5).

In all cases (Figures 4.2 to 4.5), the algorithm successfully oscillates around the ground truth value of inverse depth for the patch and aligns all the votes at the correct coordinates (where the 3D point is projected onto the reference image plane, as seen in Figure 4.1).

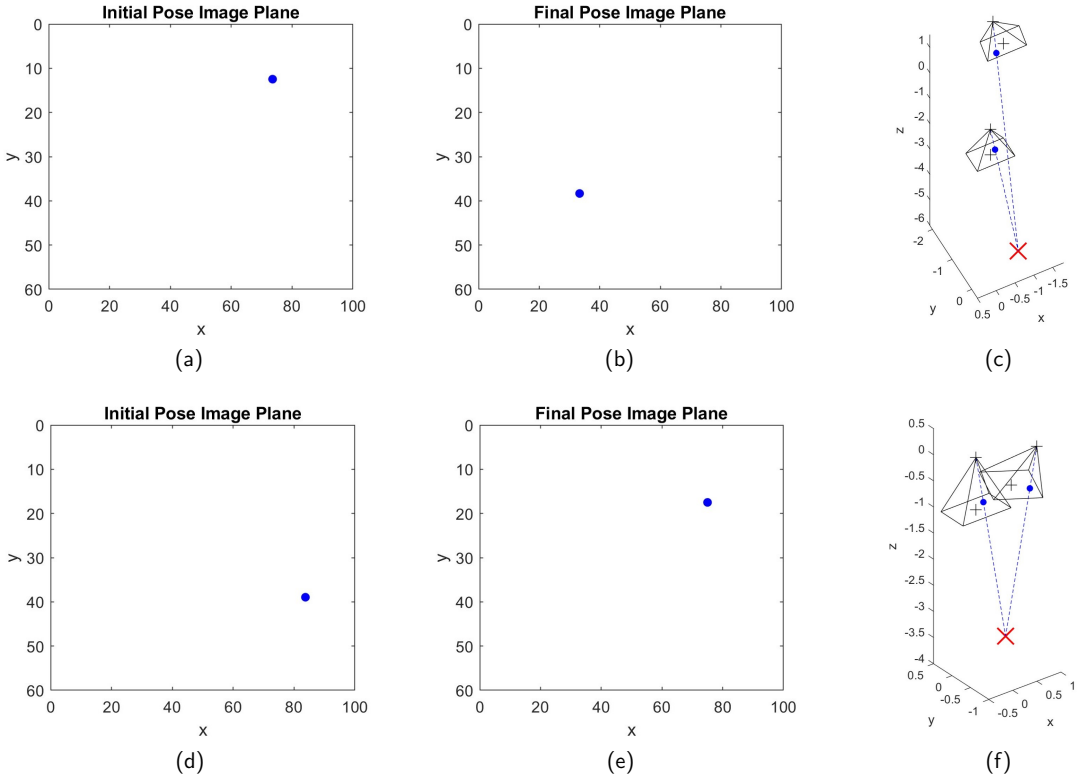



Figure 4.1: Example 1 and 2, 100 ms sequence. (a),(b) and (c) Example 1 point projection onto image planes and camera poses; (d),(e) and (f) Example 2 point projection onto image planes and camera poses.

Additionally, the results highlight two relevant observations: the gradient is normalized between -1 and 1, and the inverse depth value consistently oscillates around the ground truth in increments of 0.01. This occurs because the gradient can only provide information when the patch has experienced some change in score. Therefore, the

gradient offers ambiguous information, making it challenging to distinguish between situations where the inverse depth value is far from optimal (with small changes due to few votes) and when it is near optimal (with small changes as all events that should vote in that patch are already voting). This can lead to a very small gradient resulting in an insignificant inverse depth increment. When close to the optimal value, since the numerically calculated part of the gradient has the inverse depth increment in the denominator (Eq. 3.21), a small change in voting could spike the gradient value, causing the inverse depth to deviate significantly from the near-optimal value. Normalizing the gradient mitigates this situation, and imposing a minimum inverse depth increment of 0.01 completely avoids it. The trade-off is a lower accuracy in the final result, but the results indicate that this is sufficient for aligning the event votes effectively. These measures are maintained for all subsequent examples presented.

As the Figures 4.2-4.5 illustrate, for the initial value assigned to inverse depth $\rho_0$ the initial IWE shows misaligned votes and hence a lower contrast in the image of warped events. After the optimization process, it is evident that the votes align closely to the projection coordinates of the 3D point in the reference image, maximizating the contrast. Furthermore, despite executing 15 optimization iterations to show how the estimated values remain around the ground truth, rapid convergence to nearby values is achieved.

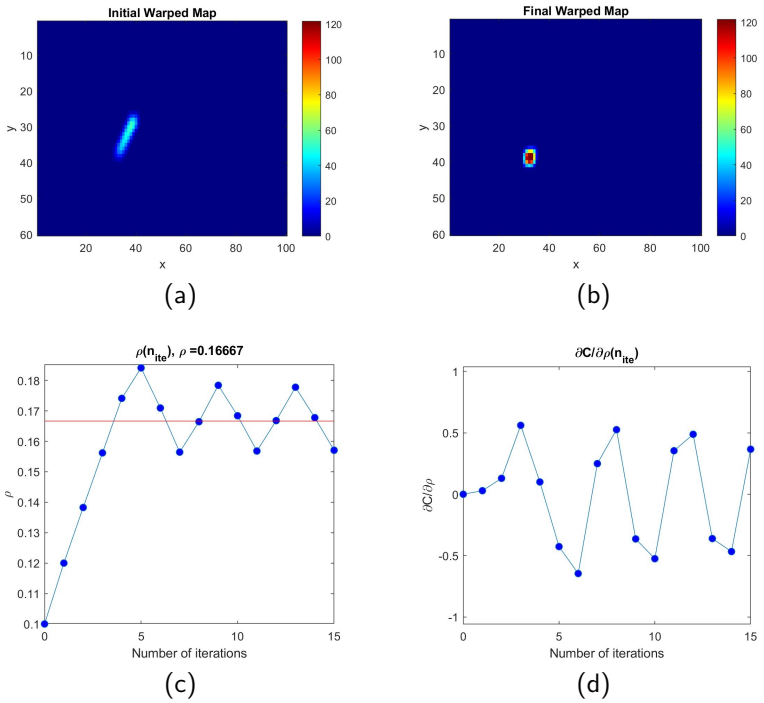



Figure 4.2: Example 1 results for initial inverse depth measures $\rho_0 = 0.1$ m$^{-1}$, $\rho_1 = 0.12$ m$^{-1}$. (a) Initial IWE; (b) Final IWE;(c) Inverse depth values assigned to each iteration process and the ground truth value in red; (d) Gradients computed.

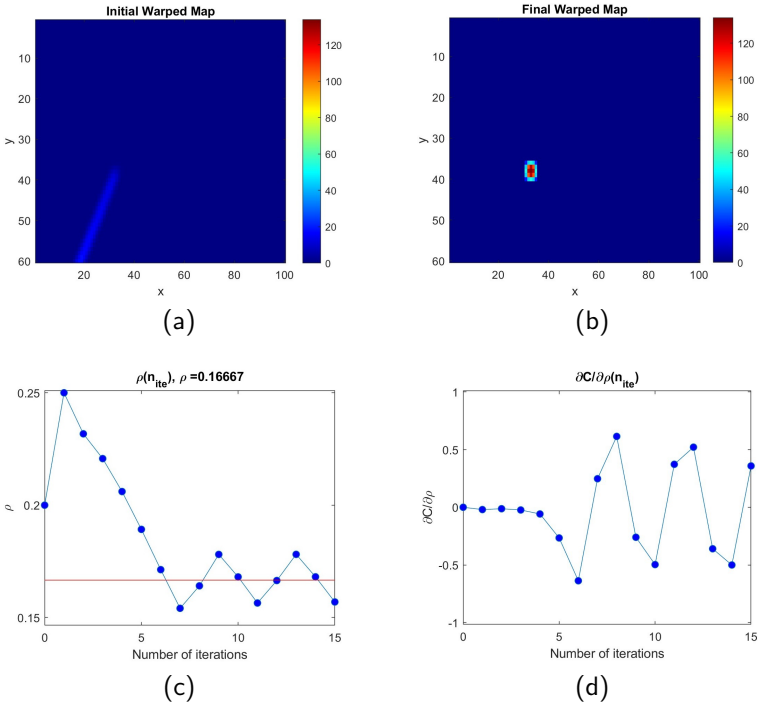



Figure 4.3: Example 1 results for initial inverse depth measures $\rho_0 = 0.2$ m$^{-1}$, $\rho_1 = 0.25$ m$^{-1}$. (a) Initial IWE; (b) Final IWE;(c) Inverse depth values assigned to each iteration process and the ground truth value in red; (d) Gradients computed.

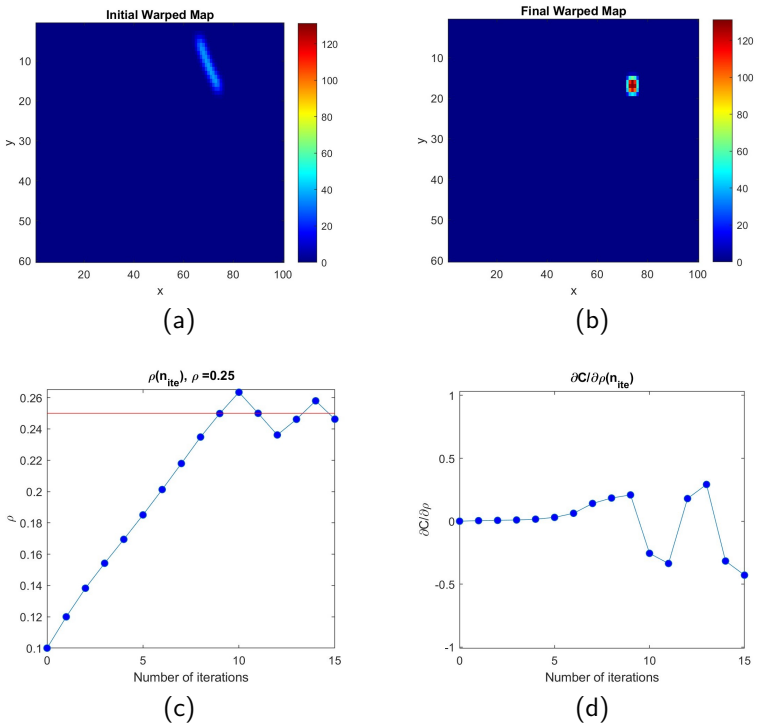



Figure 4.4: Example 2 results for initial inverse depth measures $\rho_0 = 0.1$ m$^{-1}$, $\rho_1 = 0.12$ m$^{-1}$. (a) Initial IWE; (b) Final IWE;(c) Inverse depth values assigned to each iteration process and the ground truth value in red; (d) Gradients computed.

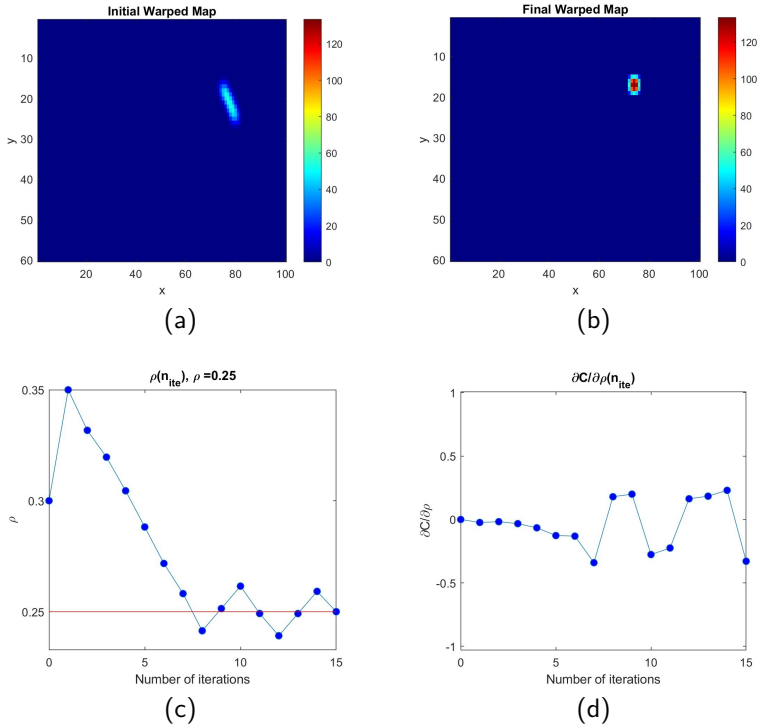



Figure 4.5: Example 2 results for initial inverse depth measures $\rho_0 = 0.3$ m$^{-1}$, $\rho_1 = 0.35$ m$^{-1}$. (a) Initial IWE; (b) Final IWE;(c) Inverse depth values assigned to each iteration process and the ground truth value in red; (d) Gradients computed.

### Inverse depth estimation for a line

In this example, the experiment is repeated but this time with a 3D line instead of a point. For event generation, the same procedure as in the previous example is followed but for two points (start and end of the line). Once the projection of the two points is obtained, the coordinates of intermediate pixels are also saved to obtain the events generated by a line.

Thus, different examples with different motion patterns are generated. We present the results of two examples in which the line is placed in different positions and with different camera movements. Additionally, to demonstrate the robustness of the optimization method, different initial conditions will be shown to demonstrate that the information provided by the proposed gradient gives the direction that the optimization algorithm should take. Table 4.2 shows the characteristics of the proposed examples.

| Example | Inverse depth | Linear velocity | Angular velocity | num. events |
|---|---|---|---|---|
| 1 | 0.1667 to 0.25 | $(10, 18, -15)$ m/s | $(0, -2.618, 1.745)$ rad/s | 5713 |
| 2 | 0.2 to 0.25 | $(-5, 10, -5)$ m/s | $(-4.363, -2.618, 0.873)$ rad/s | 6328 |

Table 4.2: Table with the examples information of the inverse depth estimation for a 3D line. The inverse depth ground truth includes the values of the two line endpoints.

Again, different camera motion patterns are performed (Figure 4.6), the line is placed at varying distances, and the performance is evaluated under different initial conditions.

As shown in Figures 4.7 and 4.8, the inverse depth is successfully optimized, remaining around its optimal value, and the event votes align correctly to form the line. Additionally, it is worth noting that the initial and final IWE representations are displayed with different scales. This approach ensures visibility in the initial image, where events are misaligned and votes are more dispersed, resulting in lower scores.

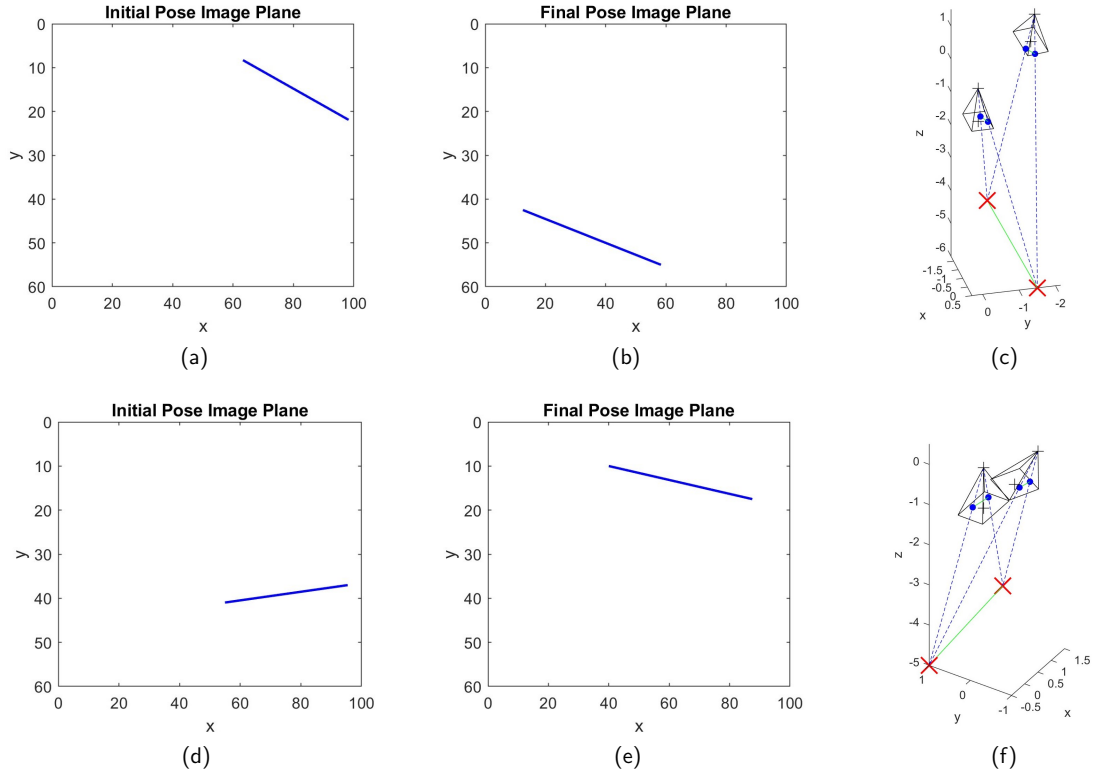



Figure 4.6: Example 1 and 2, 100 ms sequence. (a),(b) and (c) Example 1 line projection onto image planes and camera poses; (d),(e) and (f) Example 2 line projection onto image planes and camera poses.

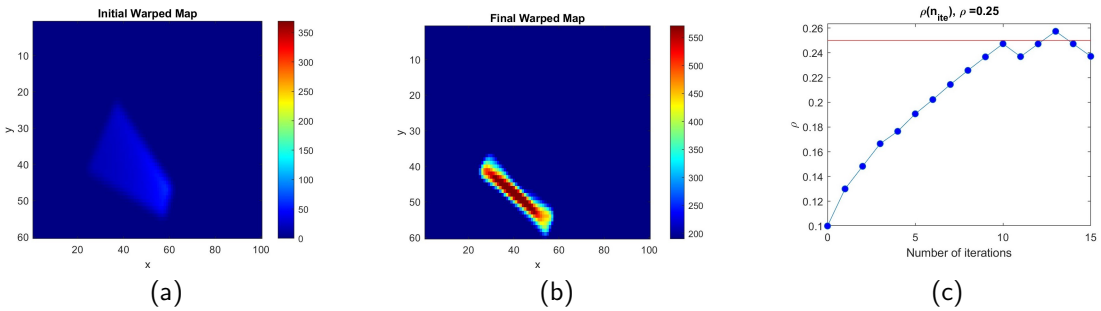



Figure 4.7: Example 1 results for initial inverse depth measures $\rho_0 = 0.1$ m$^{-1}$, $\rho_1 = 0.13$ m$^{-1}$. (a) Initial IWE; (b) Final IWE; (c) Inverse depth values assigned to each iteration process and the ground truth value in red. In order to appreciate the initial IWE scores they are plotted with a different scale.

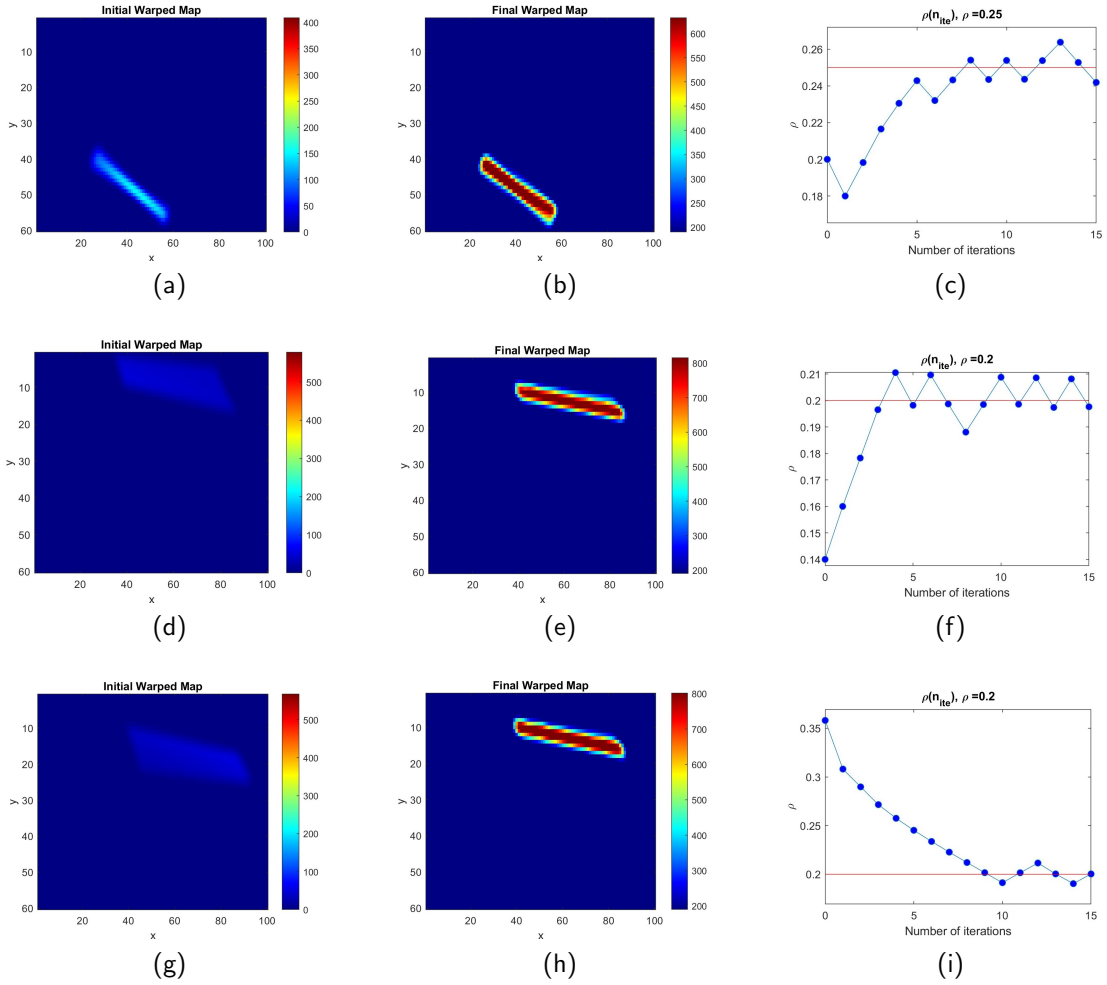



Figure 4.8: Results of inverse depth optimization process for Example 1 for initial inverse depth measures $\rho_0 = 0.2$ m$^{-1}$, $\rho_1 = 0.18$ m$^{-1}$ (a)-(c) and Example 2 for initial inverse depth measures $\rho_0 = 0.14$ m$^{-1}$, $\rho_1 = 0.165$ m$^{-1}$ (d)-(f) and $\rho_0 = 0.36$ m$^{-1}$, $\rho_1 = 0.31$ m$^{-1}$ (g)-(i). Initial IWE (a), (d) and (g) and final IWE (b), (e) and (h) are represented; (c), (f) and (i) Inverse depth values assigned to each iteration process and the ground truth value in red. In order to appreciate the initial IWE scores they are plotted with a different scale.

### Inverse depth estimation for a polyhedron

As with the previous example, events are generated for various camera movement patterns. To increase the number of edges and the overall quantity of events, a polyhedron is placed in the scene (Figure 4.9). The characteristics of the two examples are shown in Table 4.3.

| Example | Inverse depth | Linear velocity | Angular velocity | num. events |
|---|---|---|---|---|
| 1 | 0.2 to 0.25 | $(5, 20, -5)$ m/s | $(-1.745, 1.222, 0.391)$ rad/s | 24163 |
| 2 | 0.16 to 0.2 | $(10, -10, 20)$ m/s | $(-1.745, -0.873, 0.873)$ rad/s | 22498 |

Table 4.3: Table with the examples information of the inverse depth estimation for a 3D polyhedron.

It is important to note that the data generated for the examples with the polyhedron have certain limitations. For simplicity, a polyhedron with opacity was not generated; instead, only its edges were created. This means that edges behind one of the faces are also visible due to the transparency. For this particular problem, such transparency could limit the results obtained. This occurs because edges from both the front and rear faces can be very close or even overlap, despite each edge generating events corresponding to significantly different depths.

Therefore, this implies that there are events being aligned despite having different depths. These events might be located within the same patch, leading to ambiguous results during the optimization of that patch. For these examples, where the scene is simply defined by a few edges, this can have a more pronounced effect, especially when estimating egomotion later on. Generally, this issue will not occur with real data since the scenes projected onto the camera are much richer. Additionally, there will be many more patches with events being optimized, so the overall computation should yield a good inverse depth map.

In Figure 4.10 it can be observed that despite this limitations, once again, the events are correctly aligned and oscillate around the optimal value for both examples and given different initial conditions. It is worth noting that in the IWE representations, a different scale is used for the initial IWE compared to the final IWE to better visualize how the image contrast has been maximized. Additionally, unlike the previous examples, the initial pose is assigned to the origin since the goal is to obtain the most up-to-date inverse depth map possible.

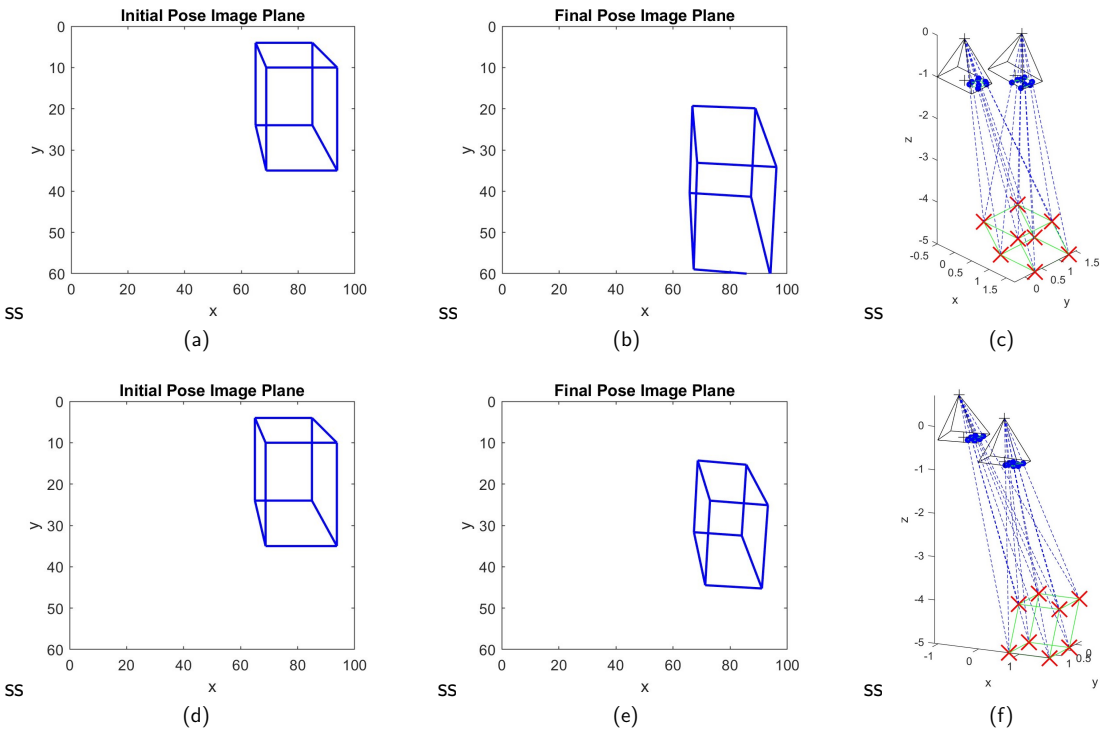



Figure 4.9: Example 1 and 2, 100 ms sequence. (a),(b) and (c) Example 1 polyhedron projection onto image planes and camera poses; (d),(e) and (f) Example 2 polyhedron projection onto image planes and camera poses.

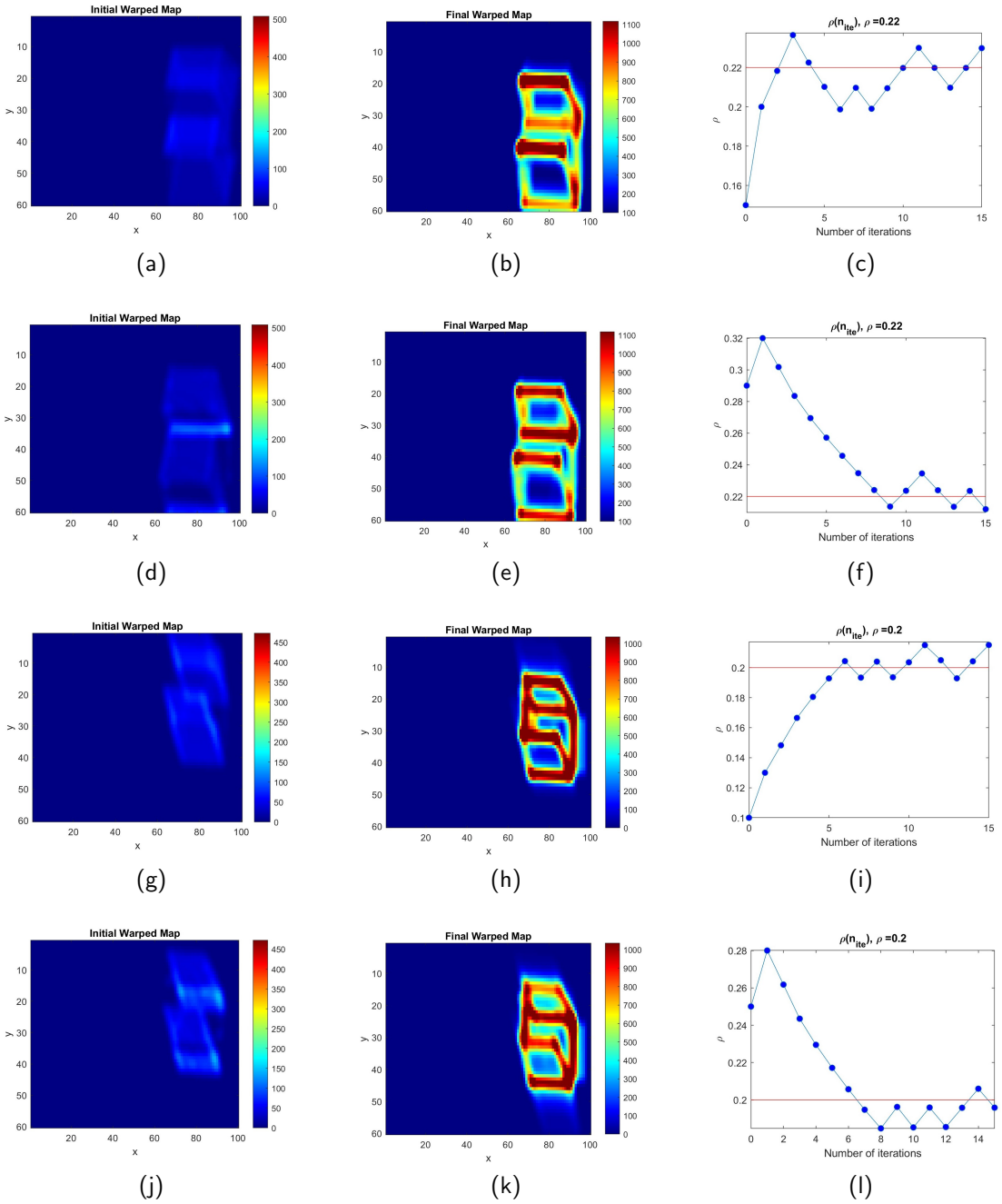



Figure 4.10: Results of inverse depth optimization process for Example 1 for initial inverse depth measures $\rho_0 = 0.15$ m$^{-1}$, $\rho_1 = 0.2$ m$^{-1}$ (a)-(c) and $\rho_0 = 0.29$ m$^{-1}$ (d)-(f), $\rho_1 = 0.32$ m$^{-1}$ and Example 2 for initial inverse depth measures $\rho_0 = 0.1$ m$^{-1}$, $\rho_1 = 0.13$ m$^{-1}$ (g)-(i) and $\rho_0 = 0.25$ m$^{-1}$, $\rho_1 = 0.28$ m$^{-1}$ (j)-(l). Initial IWE (a), (d), (g) and (j) and final IWE (b), (e), (h) and (k) are represented; (c), (f), (i) and (l) Inverse depth values assigned to each iteration process and the ground truth value in red. In order to appreciate the initial IWE scores they are plotted with a different scale.

### Inverse depth estimation using a real dataset

Once the proposed method has been tested with various synthetic data in simple examples, it is applied to a dataset recorded during an indoor free flight with Borinot. The

events are provided by a DAVIS346 camera, and ground truth poses are available from an optitrack system. Additionally, raw images are provided by an RGB camera. For this dataset, there is no ground truth for depth, so the evaluations will be qualitative, based on how well the events align in the final IWE. It is important to mention that before processing the events to estimate the inverse depth, the distortion has been removed. The algorithm is tested on two different sequences from the dataset (see Table 4.4).

| Example | Linear velocity | Angular velocity | num. events |
|---|---|---|---|
| 1 | $(-0.432, -0.147, 0.245)$ m/s | $(-0.773, -0.428, -0.239)$ rad/s | 65217 |
| 2 | $(0.046, 0.223, 0.346)$ m/s | $(-0.059, -0.038, 0.008)$ rad/s | 91049 |

Table 4.4: Table with the examples information of the inverse depth estimation using a real dataset.

To evaluate the alignment of the events, the final IWE is displayed alongside the image provided by the RGB camera. It is important to note that the event camera used has a resolution of 346 $\times$ 260 pixels, while the RGB camera used has a resolution of 892 $\times$ 676 pixels. Therefore, the field of view of the RGB camera is larger than that of the event camera. It is worth noting that despite the removal of radial distortion from the events, it is evident that the correction has not been applied to the RGB camera.

In Figures 4.11, it is clearly visible how the event votes align properly, making the scene elements identifiable in the IWE. In this case, the IWE is represented in grayscale to better visualize the image. Thus, despite the discretization of event timestamps and the resolution reduction to patches, good results for the contrast maximization and inverse depth estimation are achieved with real data.

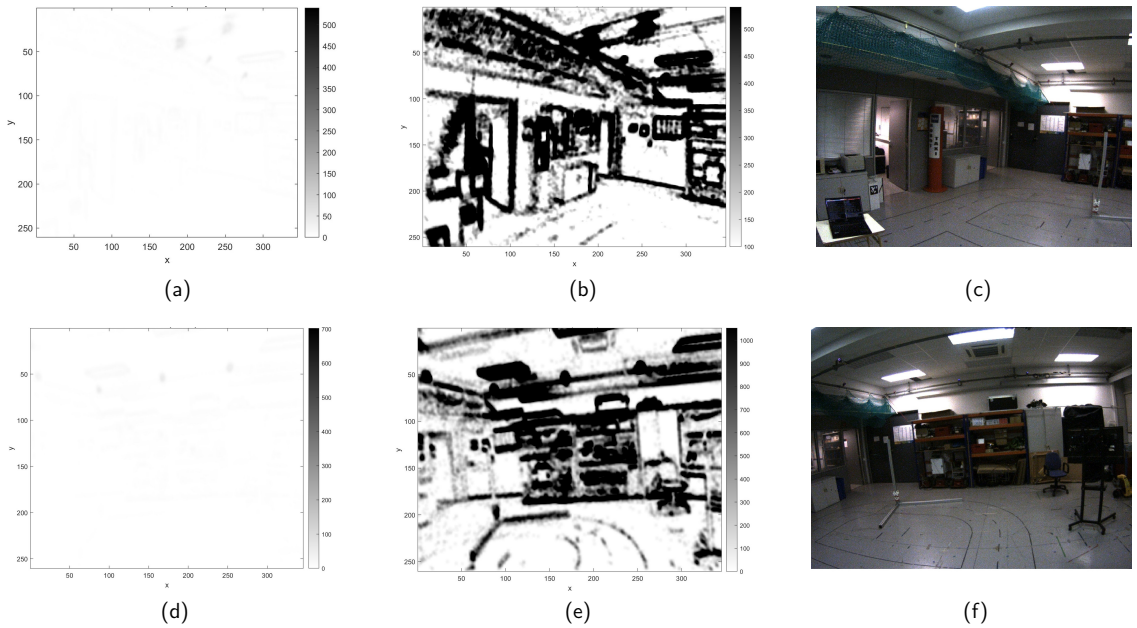


Figure 4.11: Examples 1 (a)-(c) and 2 (d)-(e) with real data, 50 ms sequences. (a) and (d) Initial IWE; (b) and (e) Final IWE after 5 optimization iterations represented in gray scale;(c) and (f) Raw image from RGB camera.

### 4.1.2 Normal flows

In order to verify the correct functioning of the normal flow estimation method, we begin with basic examples, such as estimating the normal flow of a single edge, and gradually progress to testing with real data.

- Normal flow estimation for an edge.
- Normal flow estimation for a polygon.
- Normal flow estimation for a polyhedron.
- Normal flow estimation for a real dataset recorded during a manual free flight with Borinot.

To evaluate the performance of the normal flow estimation, we will examine the coherence of the calculated normal flows in a quiver plot, comparing them to the observed movement in the image plane. In cases where necessary, a scatter plot of the vector directions will be shown to verify the sensibility of the results. It is worth mentioning that the generation of events for synthetic examples follows the same method described in the previous section. Additionally, an example with a single point is not conducted since the normal flow is defined as the component normal to an edge of the apparent motion vector in the image plane, making it irrelevant to calculate it at a single point.

In Section 3.4.4, a method for rectifying normal flows using multi-spatial scale maxpooling is explained. However, it was not necessary to implement this method until the normal flow estimation algorithm was tested with real data. Therefore, the only section where the correction of the estimated normal flows is performed is the final section, where the estimation algorithm is tested with real data. Additionally, this section also highlights the differences in the results obtained using multi-spatial scale maxpooling compared to the previous results.

#### Normal flow estimation for an edge

In this first example, the simplest case is considered: estimating the normal flow of an edge moving in the image plane. To demonstrate the method's functionality, two examples are executed under the conditions shown in Table 4.5.

| Example | Linear velocity | Angular velocity | num. events |
|---|---|---|---|
| 1 | $(0, -10, 0)$ m/s | $(0, 0, 0)$ rad/s | 17000 |
| 2 | $(10, -10, 0)$ m/s | $(0, 0, 0)$ rad/s | 17000 |

Table 4.5: Table with the examples information of the normal flow estimation for an edge.

In Figure 4.12, it can be observed that in the first example, the camera moves downwards, while in the second example, it also moves to the right. As expected, due to the aperture problem, only the normal component of the edge is recovered (see 4.13).

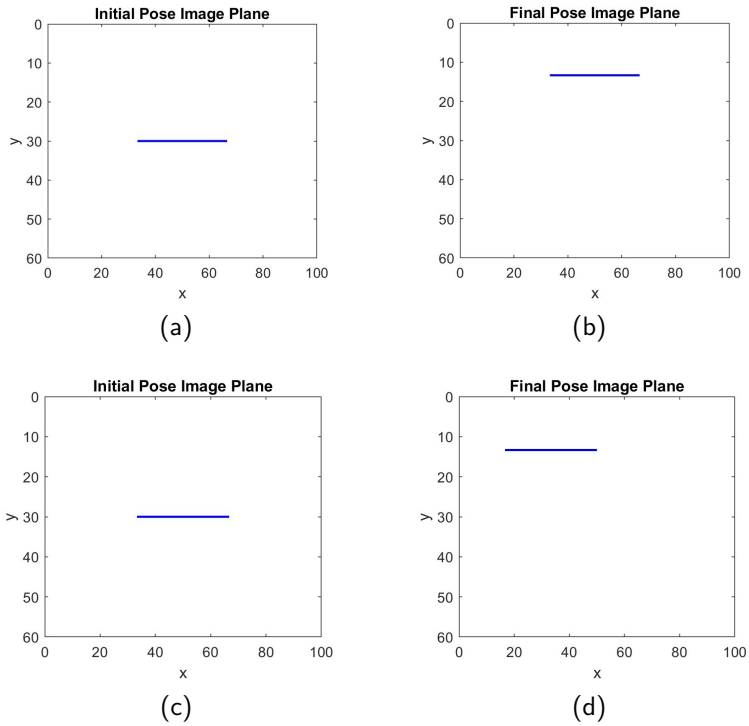



Figure 4.12: Example 1 and 2, 100 ms sequence. (a) and (b) Projection of the edge onto the initial and final image planes from Example 1;(c) and (d) Projection of the edge onto the initial and final image planes from Example 2.

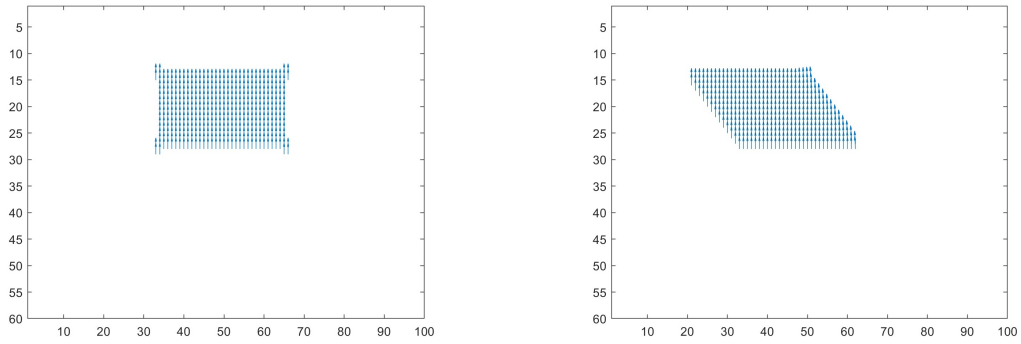



Figure 4.13: Quiver plot of Examples 1 and 2. (a) Normal flow vectors estimated from Example 1; (b) Normal flow vectors estimated from Example 2.

### Normal flow estimation for a polygon

Since the normal flow estimation examples for a single edge only yield vectors in one direction, the method is tested on data generated from the movement of a polygon to observe a more varied normal flow (see Table 4.6).

| Example | Linear velocity | Angular velocity | num. events |
|---|---|---|---|
| 1 | $(0, 5, 5)$ m/s | $(0, -0.873, 0.174)$ rad/s | 11022 |
| 2 | $(10, -10, 10)$ m/s | $(0, 0.873, 0.873)$ rad/s | 10630 |

Table 4.6: Table with the examples information of the normal flow estimation for a polygon.

In Figure 4.14, the movement of the camera and the motion of the polygon in the image plane are illustrated for the proposed examples. In Example 1, the camera moves upward and slightly to the right, whereas in Example 2, it moves downward and slightly to the left.

The results can be observed in Figure 4.15. It is evident that we now have apparent motion vectors normal to the various edges. However, a small portion of these vectors are outliers, failing to accurately recover the direction or magnitude (Figure 4.15 (a), (c)). Nevertheless, as shown in Figure 4.15 (b), (d), the majority of the vectors provide correct direction.

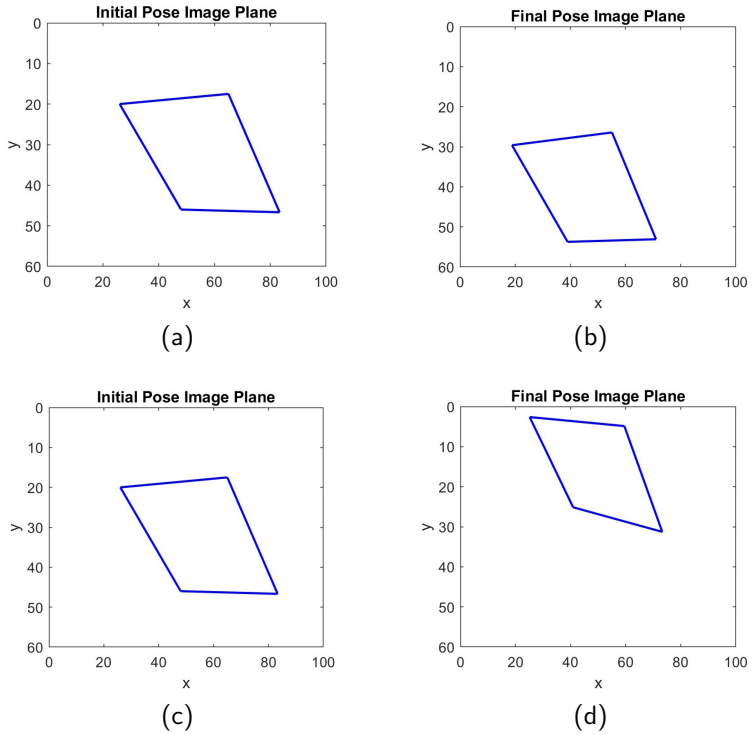
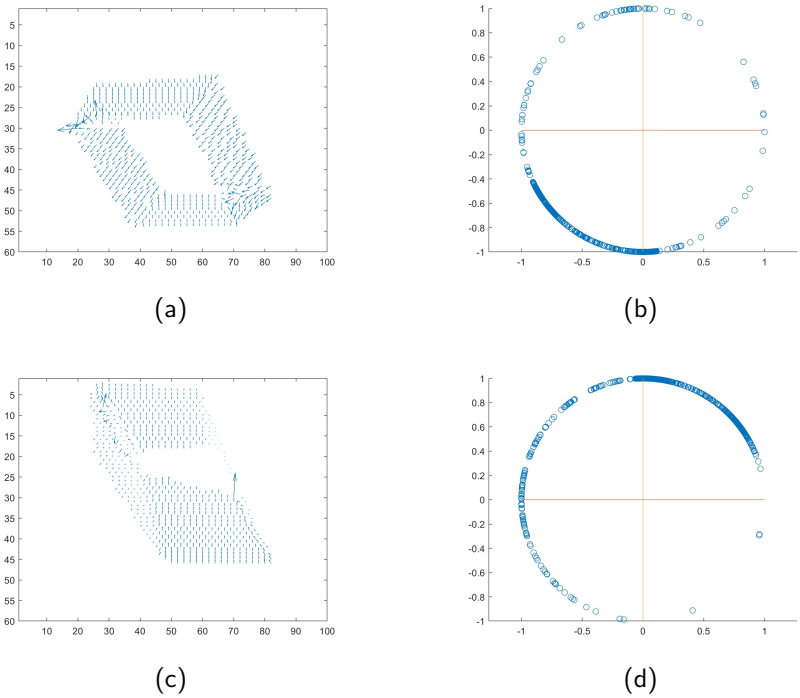



Figure 4.14: Example 1 and 2, 100 ms sequences. The projection of the polygon onto the initial and final image planes is represented for Examples 1 (a) and (b) and 2 (c) and (d).



(a) (b)

(c) (d)

Figure 4.15: Normal flow estimation results of Examples 1 and 2. (a) Normal flow vectors estimated from Example 1; (b) Normal flows direction vectors (normalized) from Example 1; (c) Normal flow vectors estimated from Example 2; (d) Normal flows direction vectors (normalized) from Example 2.

### Normal flow estimation for a polyhedron

Next, the synthetic example is made slightly more complex by calculating the normal flow of a polyhedron moving in the image plane. Table 4.7 shows the different movement patterns used to evaluate the normal flow estimation.

| Example | Linear velocity | Angular velocity | num. events |
|---|---|---|---|
| 1 | $(-10, -7, -2)$ m/s | $(0.349, 0.174, -0.174)$ rad/s | 11626 |
| 2 | $(5, 5, 0)$ m/s | $(-0.174, 0.174, 0.174)$ rad/s | 11948 |

Table 4.7: Table with the examples information of the normal flow estimation for a polyhedron.

In Figure 4.16, the various camera poses and the movement of the polyhedron's projection in the image plane can be observed. The results of running both examples can be seen in Figure 4.17. In this case, there are more outliers than before. This is because, in cases where two edges of the polyhedron are very close, the events generated by one edge are overwritten by the next, causing timestamps from different edges to mix within the same spatiotemporal window. Additionally, in these examples, the synthetic data was constructed in a way that the polyhedron is considered transparent, meaning there is no opacity between the edges and faces of the polyhedron, which exacerbates this limitation of the method. However, as shown in Figure 4.17 (b) and (d), most of the

calculated normal flows still align correctly with the normals of the polyhedron's edges. Therefore, the RANSAC in the linear solver should be able to handle this situation.

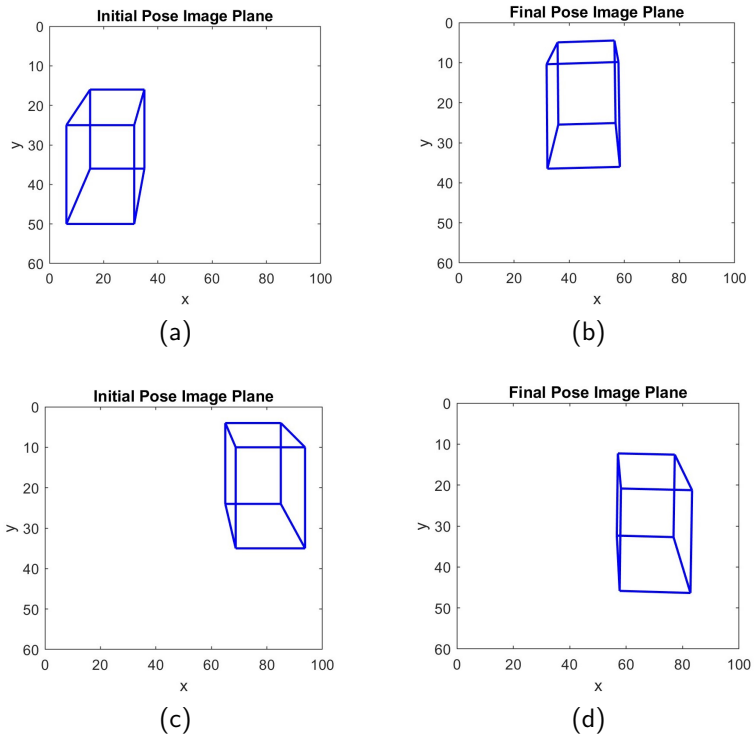



Figure 4.16: Example 1 and 2, 100 ms sequence. (a) and (b) Projection of the polyhedron onto the initial and final image planes from Example 1;(c) and (d) Projection of the polyhedron onto the initial and final image planes from Example 2.

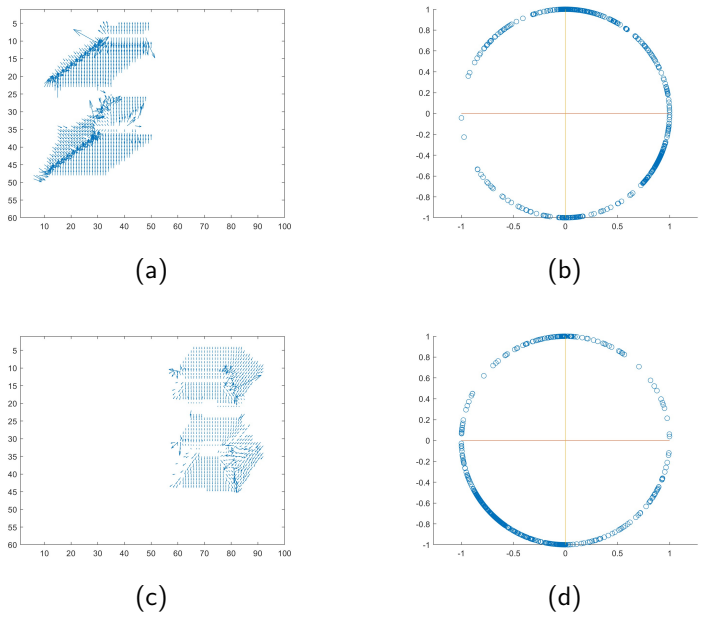



Figure 4.17: Normal flow estimation results of Examples 1 and 2. Normal flow vectors estimated from Examples 1 (a) and 2 (b). Normal flows direction vectors (normalized) from Examples 1 (c) and 2 (d).

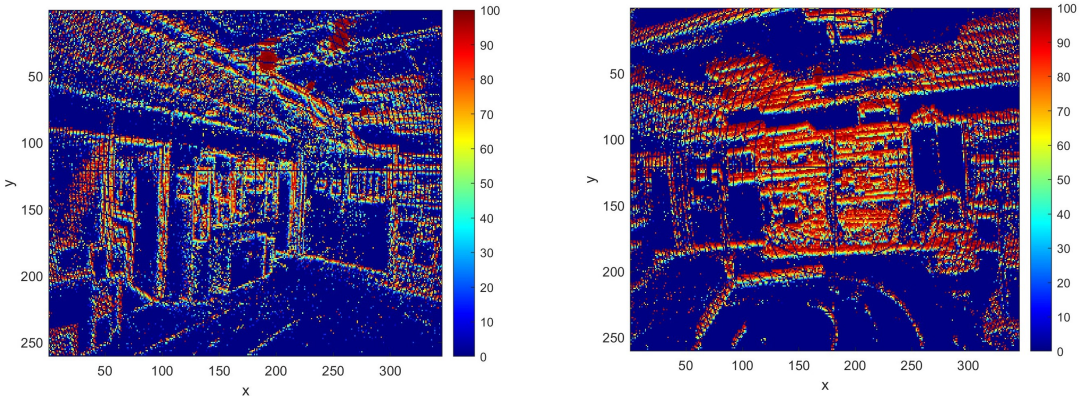
### Normal flow estimation using a real dataset

Once the functionality of the normal flow method has been verified on synthetic data, it is tested on the dataset recorded with Borinot during a manual free-flight. The sequences used for testing the algorithm are the same as those used for the inverse depth test, so the data characteristics are identical (see Table 4.8).

| Example | Linear velocity | Angular velocity | num. events |
|---|---|---|---|
| 1 | $(-0.432, -0.147, 0.245)$ m/s | $(-0.773, -0.438, -0.239)$ rad/s | 65217 |
| 2 | $(0.046, 0.223, 0.346)$ m/s | $(-0.059, -0.038, 0.008)$ rad/s | 91049 |

Table 4.8: Table with the examples information of the normal flow estimation using a real dataset.

In case of real data, we have no ground truths for normal flow and, to understand the direction in which the scene is moving in the image plane, we plot the timestamps of the events generated during the 50 ms sequences (see Figure 4.18). Events generated with the farthest timestamps are shown in blue, while those with the most recent timestamps are in brown. From the plots, it can be deduced that in Example 1, the camera is moving downwards and to the left with respect to the image plane, and in Example 2, it is moving downwards.




(a) (b)

Figure 4.18: Representation of the events and their timestamps assigned in the sequences. (a) Sample sequence corresponding to camera motion downwards and slightly to the left; (b) Sample sequence with mainly downward camera motion with respect to the image plane that can be attributed either for camera rotation or camera displacement or a combination of both.

In Figure 4.19, it is observed that the solutions for normal flow estimation are poorly conditioned. In the quiver plots (a) and (c), it is observed that the solutions are also poorly conditioned in terms of magnitude. Additionally, in the scatter plots shown in frames (b) and (d), although many normal flow vectors accumulate in the correct directions, there are many outliers. This is a significant issue because to recover the linear velocity, the linear solver needs well-conditioned normal flow vectors in both magnitude

and direction.

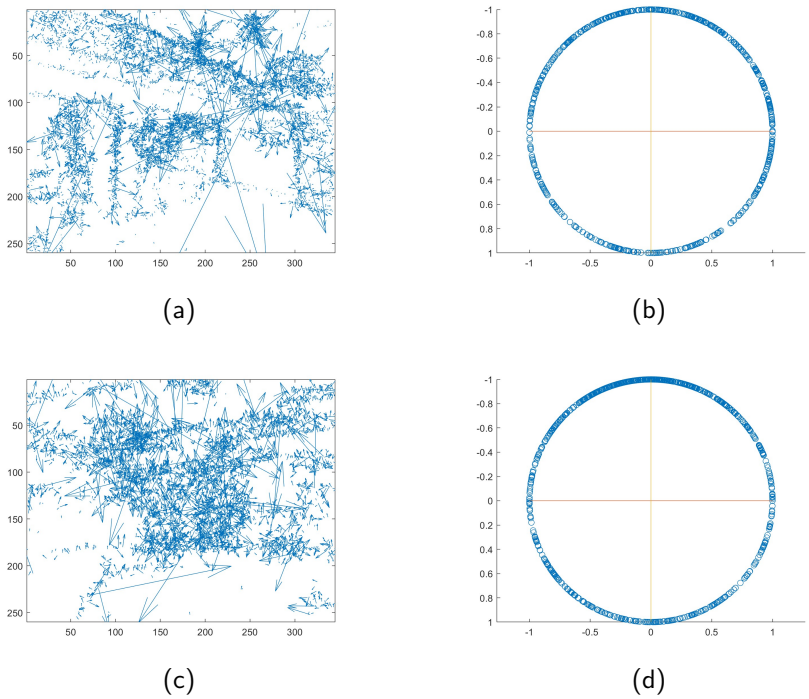



Figure 4.19: Normal flow estimation results of Examples 1 and 2. (a) Normal flow vectors estimated from Example 1; (b) Normal flows direction vectors (normalized) from Example 1; (c) Normal flow vectors estimated from Example 2; (d) Normal flows direction vectors (normalized) from Example 2. Just 40% of the flows are represented in order to ease the results visualization.

In order to analyze what is happening in the estimation of the normal flow with real data, we analyze the plane-fitting of SAEs for each example. Figure 4.20 illustrates how the plane has been fitted in two SAEs and the resultant normal vector, one for the first sequence (a),(b) and another for the second (c),(d). The algorithm adjusts by minimizing the distance of the events in the spatiotemporal window to the plane (Section 3.3.2). In these examples, it is observed that the events of the SAEs do not show a clear direction in the image plane over time. For this reason, although the plane fits well to the events in the SAE and the normal vector is computed correctly, the set of events used does not reflect the normal direction of motion to a specific edge.

The explanation for this lies in how the data is generated. In previous examples with synthetic data, events are generated uniformly wherever an edge passes, but textures do not exist. However, in reality, events are generated by changes in brightness in pixels, and these changes occur in pixels of the image where there is also texture. If the area around an edge has different textures (as is often the case in reality), it will generate different events in different pixels and timestamps. This generates the effect observed in the image where the SAEs do not show a clear planar pattern. This problem is similar to what was observed in the example of the wireframe polyhedron when there were two nearby edges, in which one edge of the polyhedron was overwriting the events left by the other. With real data, the problem is more serious because textures are not as

uniform as an edge, and therefore, the way events are overwritten is also less consistent. Additionally, it does not occur only in isolated areas of the image as in the example of the polyhedron but occurs in practically the entire image. This limitation complicates the possibility of recovering linear velocity with real data using this method of estimating normal flows.

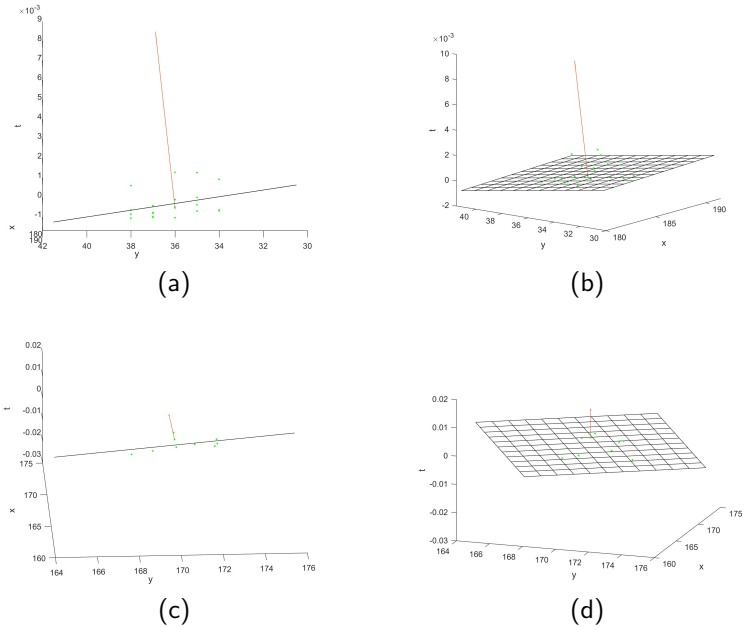



Figure 4.20: Plane-fitting results for two SAEs created from the data in Examples 1 (frames a and b) and 2 (frames c and d). Events are shown in green and the fitted plane is shown in black. The resultant normal vector is shown in red.

To improve the condition of the estimated normal flows, the multi-spatial scale max-pooling method explained in Section 3.4.4 is implemented. This rectification aims to achieve more consistent normal flow estimations that better correspond to the actual flow.

In this way, it can be observed in Figure 4.21 that the normal flows obtained are more consistent in their direction, with fewer poorly conditioned estimations. However, as seen in Figure 4.22, the magnitude of the estimated vectors remains poorly conditioned. It is important to remember that to accurately recover the linear velocity, it is not sufficient to only have a well-estimated direction (as is the case here); a well-conditioned magnitude is also required. Therefore, although the multi-spatial scale maxpooling method achieves a direction more consistent with the real flow, it fails to recover the magnitude accurately, and hence the linear velocity, when using real data.

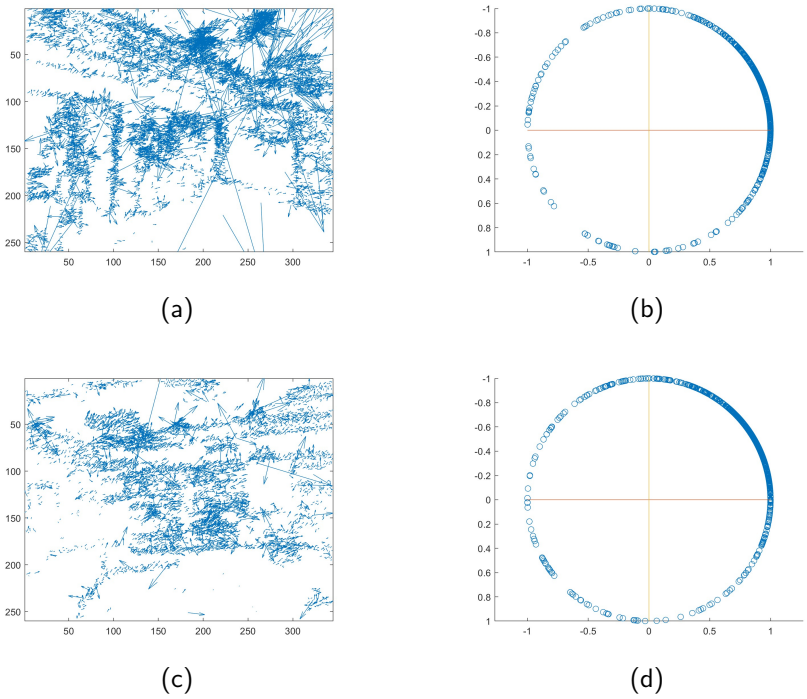



Figure 4.21: Normal flow estimation results of Examples 1 and 2 applying multi-spatial scale maxpooling rectification. (a) Normal flow vectors estimated from Example 1; (b) Normal flows direction vectors (normalized) from Example 1; (c) Normal flow vectors estimated from Example 2; (d) Normal flows direction vectors (normalized) from Example 2. Just 40% of the flows are represented in order to ease the results visualization.

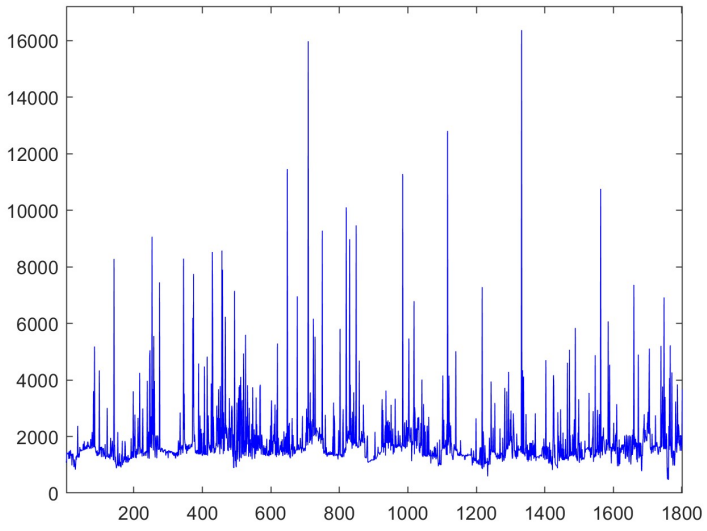



Figure 4.22: Representation of the flow vectors magnitude estimated with real data.

### 4.1.3 Egomotion estimation

Similar to the approach taken with the other stages of the system, the performance of the linear solver is verified step by step using various examples with synthetic data. The case of real data is left for future work once the issues with normal flow with real data are solved. Thus, two different scenes are presented:

- Egomotion estimation for a polygon.
- Egomotion estimation for a polyhedron.

In this section, the objective is to estimate the egomotion of the camera (linear velocity) given the angular velocity (from the IMU), the inverse depth map, and the normal flow at each pixel.

To evaluate the performance of the egomotion estimation, the estimated result will be compared with the ground truth value, and the relative error will be calculated as shown in Equation 4.4.

$$\epsilon_{\text{rel}} = \frac{\sqrt{(\mathbf{v}_{\text{gt}} - \mathbf{v}_{\text{est}})^2}}{||\mathbf{v}_{\text{gt}}||} \tag{4.4}$$

where $\mathbf{v}_{\text{est}}$ is the estimated linear velocity vector, $\mathbf{v}_{\text{gt}}$ is the ground truth vector, and $||\mathbf{v}_{\text{gt}}||$ is its Euclidean norm.

#### Egomotion estimation for a polygon

Firstly, the egomotion for examples involving a polygon are estimated. It is important to note that in these examples, since the estimated normal flows are well-conditioned, the system is solved directly using a least squares method applied to all pixels where the depth and normal flow are known. By utilizing the motion field equation adapted for normal flows 3.43, we can solve the different examples. In all proposed examples, the polygon is initially centered in the image (see Figure 4.23) and a movement is executed from that position.

In Table 4.9 the different examples executed and the results can be seen.

| **Example** | $\mathbf{v}_{\text{gt}}$ [m/s] | $\mathbf{v}_{\text{est}}$ [m/s] | $\epsilon_{\text{rel}}$ (%) | **num. events** |
|---|---|---|---|---|
| 1 | (10,10,0) | (10.0812,10.061,0.000) | 0.718 | 6732 |
| 2 | (-20,10,0) | (-21.975,12.722,0.000) | 15.041 | 6951 |
| 3 | (12,-5,0) | (10.738,-5.383,2.487) | 21.653 | 6701 |
| 4 | (-30,-10,2) | (-29.952,-7.835,-2.737) | 16.439 | 6881 |
| 5 | (10,-10,0) | (10.076,-10.076,0.000) | 0.764 | 6732 |
| 6 | (-10,10,0) | (-10.088, 10.088,0.000) | 0.880 | 6732 |

Table 4.9: Results of egomotion estimation for a polygon. The results of the estimated velocity $\mathbf{v}_{\text{est}}$ and relative error $\epsilon_{\text{rel}}$ have been estimated to the third decimal value.

Overall, the estimation of linear velocity is achieved. Observing the results, the examples with the smallest error are those where the camera moves the same distance

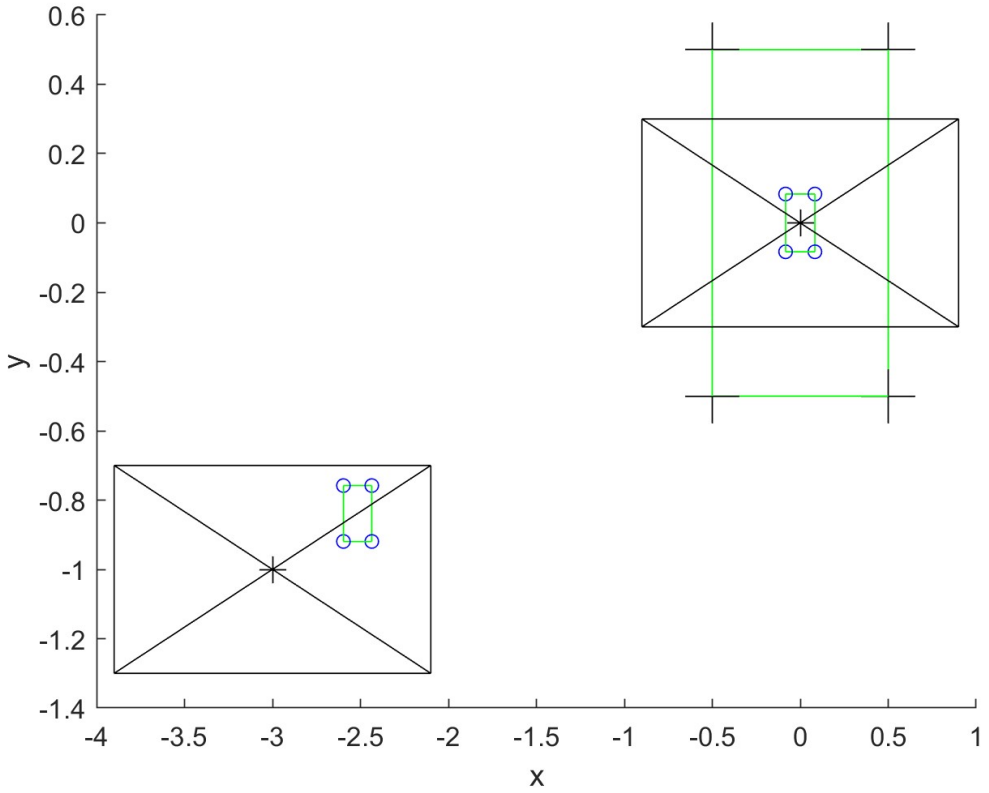



Figure 4.23: Example 4 visualization. In the image the camera poses from example 4 can be seen in addition to the polygon that projects onto the center of the initial image plane.

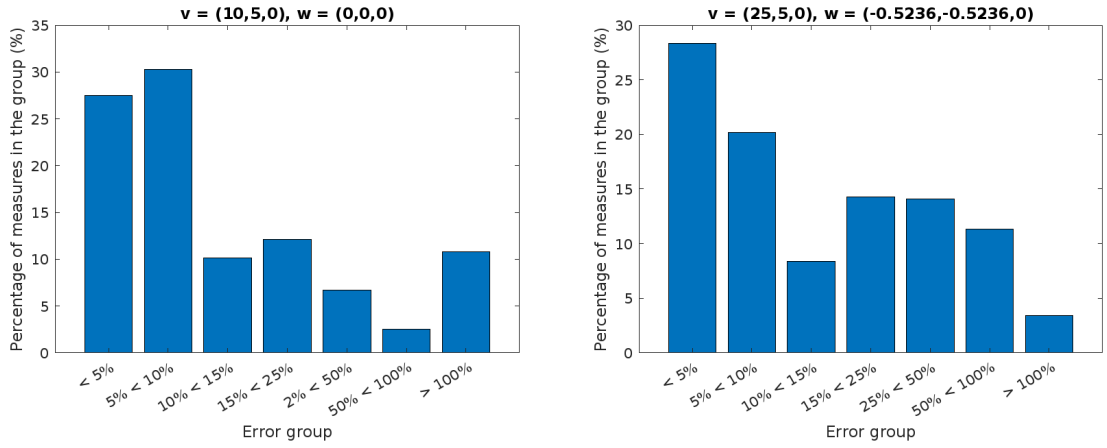
in both the $x$ and $y$ directions (Examples 1, 5, and 6). This is because, given the polygon, we only have normal flow vectors with purely vertical and horizontal directions. Consequently, there is not a wide variety of directions, and in cases where the vector is not entirely horizontal, vertical, or the vertical component does not equal the horizontal component, the system is not as well-conditioned. Moreover, the examples with the highest error (Examples 3 and 4) primarily fail in estimating the velocity in the $z$ direction. This is likely due to the lack of parallax and the discretization of the depth bins performed using the contrast maximization method, resulting in a much lower resolution in the $z$ dimension compared to $x$ and $y$.

### Egomotion estimation for a polyhedron

In the previous example, there is a lack of variety in normal flow orientations, having only vertical and horizontal axes. To have richer event flow directions, various examples involving a polyhedron are implemented. As seen in the normal flow estimation section (Section 4.1.2), increasing the scene complexity induces poor conditioning in flow estimation. Consequently, a RANSAC-based linear solver is implemented as explained in Section 3.5.

To determine the appropriate values for the RANSAC parameters, a study was conducted on how well the motion field equations adapted for normal flow fit the ground

truth model. Figure 4.24 shows the percentage of pixels with known inverse depth and normal flow falling within each error range for two different examples.




(a) (b)

Figure 4.24: Relative error distribution of the pixels used to solve the linear velocity for two examples: (a) $\mathbf{v} = [10, 5, 0]^\top$ m/s, $\boldsymbol{\omega} = [0, 0, 0]$ rad/s and (b) $\mathbf{v} = [25, 5, 0]^\top$ m/s, $\boldsymbol{\omega} = [-0.5236, -0.5236, 0]$ rad/s.

Based on the error distribution presented in Figure 4.24, we decided to consider any pixel with a relative error less than 20% compared to the proposed model as an inlier. Additionally, a minimum of 60% of the pixels must be inliers for the model to be considered valid. To ensure a good estimation in over 99% of the cases, 500 RANSAC iterations are performed, based on the method for computing the necessary number of iterations explained in [80]. Finally, the minimum number of equations required to solve the system is one for each parameter to be estimated, thus we need three equations to solve for the three dimensions of the linear velocity vector. Therefore, the RANSAC parameters mentioned in Section 3.5 are assigned the following values:

$$\begin{cases} k = 500 \\ n = 3 \\ t = 0.2 \\ d = 0.6 \cdot pix \end{cases}$$

where $pix$ is the number of pixels for which the inverse depth and normal flow are known.

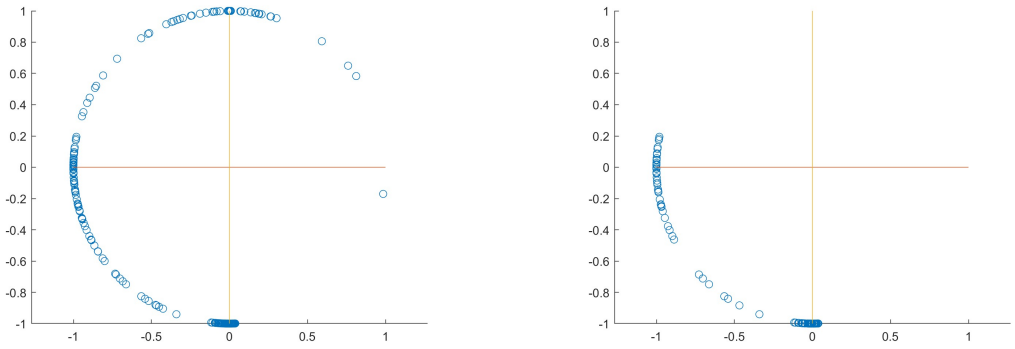
The results of the various examples executed with the polyhedron are presented in Table 4.10. It can be observed that in these examples, the linear velocity is estimated correctly. However, despite seeing how in some examples the velocity in $z$ axis is recovered well (Examples 5 and 6), it is still observed to be the main cause of most of the error in other cases (Examples 1, 2, and 3). On the other hand, in Example 4, a good estimation of the normal flow orientation is observed, but there is an error in the scale.

As previously mentioned, RANSAC should be capable of selecting pixels where the normal flow is well-conditioned. To verify this, we refer to Example 6 (4.25) and examine the scatter plots of normal flow directions before and after applying RANSAC. It can

| **Example** | $\omega$ [rad/s] | $\mathbf{v}_{gt}$ [m/s] | $\mathbf{v}_{est}$ [m/s] | $e_{rel}$ (%) | **num. events** |
|---|---|---|---|---|---|
| 1 | (0,0,0) | (10,10,0) | (11.647,10.872,1.263) | 15.919 | 18612 |
| 2 | (-0.523,0,0) | (10,5,0) | (10.017,5.003,-1.514) | 13.544 | 9637 |
| 3 | (0.873,0.523,0.349) | (20,10,0) | (19.268,9.791,-2.919) | 13.490 | 9685 |
| 4 | (-0.523,0,0) | (20,20,10) | (24.457,23.664,13.096) | 21.782 | 8671 |
| 5 | (-0.523,0.349,0.174) | (20,15,10) | (22.637,18.110,10.232) | 15.169 | 8680 |
| 6 | (-0.349,1.222,0) | (10,10,-5) | (9.067,11.466,-5.456) | 11.980 | 10326 |

Table 4.10: Results of egomotion estimation for a polyhedron.

be observed how RANSAC retains only those normal flows that provide well-conditioned information about the downward and leftward directions.



(a) (b)

Figure 4.25: Scatter plots of the normal flow orientations before (a) and after (b) the RANSAC algorithm. It can be seen that the RANSAC retains only the well-conditioned normal flows.

To this point, each example has been executed independently. However, in reality, motion patterns will occur one after another. In these cases, the previous results can be leveraged to provide initial values for inverse depth estimation, as explained in Section 3.3.2. To verify this functionality, three linked motion sequences are performed (see Table 4.11). In these example, it can be seen that for each of the different motion patterns, the velocity is estimated with a relative error of less than 15%. Hence, we can conclude that the estimation error computed in a previous sequence does not undermine the performance of contrast maximization for depth estimation in the next sequence.

| **Sequence** | $\omega$ [rad/s] | $\mathbf{v}_{gt}$ [m/s] | $\mathbf{v}_{est}$ [m/s] | $e_{rel}$ (%) | **num. events** |
|---|---|---|---|---|---|
| 1 | (0.349,-1.222,0) | (10,10,-5) | (8.393,9.019,-6.0905) | 14.504 | 12250 |
| 2 | (-0.349,1.222,0) | (-10,-10,5) | (-8.501,-10.222,6.5029) | 14.228 | 12151 |
| 3 | (0,0,0) | (5,5,0) | (4.082,4.978,-0.518) | 14.904 | 20794 |

Table 4.11: Results of a sequence of egomotion estimation examples for a polyhedron.

Chapter 5

# Conclusions and future work

## 5.1 Conclusions

This work explores model-based egomotion estimation using an event camera in unknown environments. The workflow and each experiment conducted during the project's development are designed to understand the problem from its fundamental aspects. Furthermore, the approach aims to exploit the unique features of event cameras through geometric proposals without relying on machine learning, allowing execution on low-resource systems. After extensive and continuous exploration of the state of the art, a system divided into three main blocks has been implemented: inverse depth estimation, normal flow estimation, and a robust linear solver. Specifically, a contrast maximization approach has been applied for inverse depth estimation, planar approximations of SAEs has been used for normal flow estimation, and a RANSAC-based solver has been implemented to solve the scalar motion field equation.

The methodology was tested with simple experiments using synthetic data and then applied to sequences with real data. In synthetic data examples, the inverse depth optimization algorithm effectively aligned the events and rapidly converged to the ground truth value, even with unfavorable initial conditions. Additionally, the normal flow estimation recovered both the direction normal to the edges and their magnitude in these examples. The successful performance of these two blocks with synthetic data enabled the linear solver to recover the linear velocity, despite some limitations in the generated data. However, when testing normal flow estimation with real data, poorly conditioned results were observed. To address this issue, the obtained normal flows were corrected using a multi-spatial scale maxpooling approach which, despite significantly improving the orientation of the estimated vectors, failed to recover their magnitude. Nevertheless, the inverse depth estimation method proved to be highly successful. When tested on sequences with real data, it correctly aligned the events with very few iterations. This optimization process, being gradient-based, achieved efficiency through the gradient computation and the voting function we proposed.

As mentioned at the beginning of the document, it is important to highlight that this is a preliminary work aimed at addressing this type of problem. Therefore, the objectives

were to identify the limitations encountered and to deeply understand the underlying mathematics and geometry. The project successfully implemented a monocular inverse depth estimation method, demonstrated that the robust linear solver effectively solves the motion field equation when flows are well-conditioned, and identified a limitation in recovering the scale of normal flows. Furthermore, the following section describes the steps for future research based on the results obtained. Consequently, it can be concluded that the project's objectives have been met.

## 5.2 Future directions

This work represents an initial step towards a future research line. Thus, it is crucial to identify the encountered limitations and propose new tasks to continue this project.

Firstly, an alternative to normal flow estimation, identified as a limitation, is proposed. One possible solution involves attempting to recover optical flow instead of normal flow. As discussed throughout the project, the aperture problem presents a significant challenge. However, there is an option to use the contrast maximization approach, as used for inverse depth estimation, for optical flow estimation. This solution is considered promising due to its demonstrated effectiveness in inverse depth estimation. In this case, instead of parameterizing the IWE with inverse depth, it would be parameterized with the pixel displacement over a short period, aiming to align the events at a common reference timestamp. This approach is elegant and coherent since both blocks preceding the linear solver would operate under the same principle. Additionally, the linear solver should be adapted to solve the original vectorial motion field equation rather than the scalar equation adjusted for normal flows.

Moreover, the algorithm has been fully tested and implemented in MATLAB. While this facilitated exploring the method from its basic principles and understanding the mathematical foundations, it imposes a limitation on evaluating the computational cost and execution time of the algorithm. Therefore, future work should involve implementing the developed method in Python or C++. Considering the ultimate goal of this research line is to integrate the perception system into Borinot, achieving real-time execution of the method will be essential.

Finally, another proposal for the future is the integration of event camera and IMU data using a factor graph. A factor graph is a bipartite graphical model that represents the factorization of a global function into a product of local functions, facilitating efficient computation and inference. IMUs offer high-frequency motion data, but they are not ideal for long-term navigation on their own due to the rapid accumulation of errors and the challenging issue of measurement bias that drifts over time. Factor graphs provide a highly adaptable framework for integrating these two complementary sources of information. An IMU can provide subtle contributions through linear acceleration, which can help correct the metric scale as the z-axis acceleration reflects the Earth's gravity in $m/s^2$. This data, when double integrated, can assist in refining the scale. Therefore, utilizing a factor graph for data fusion between an IMU and an event camera can significantly enhance the accuracy, robustness and computational effectiveness.

Chapter 6

# Sustainability analysis and ethical implications

## 6.1 Introduction

In the current context, characterized by rapid technological growth and increasing global awareness of the environmental crisis, drafting sustainability analysis reports and their ethical implications has become fundamental in the field of engineering. These reports not only allow for the assessment of the environmental and social impact of technological projects but also guide decision-making towards more responsible and sustainable practices. Modern engineering can no longer be considered in isolation from its effects on the environment and society. Engineers have the responsibility to design solutions that are not only efficient and cost-effective but also minimize negative environmental impacts and promote social well-being.

Furthermore, the growing demand for transparency and corporate accountability from society and regulatory bodies makes these reports essential for maintaining trust and the social license to operate. Incorporating ethical and sustainability principles into engineering projects is not just a recommended practice, but a necessity to ensure technological development contributes positively to the future of our society and planet.

This report conducts an analysis of the sustainability and ethical implications related to the work carried out. First, the environmental, economic, and social impact is identified using the sustainability matrix as a guide (Section 6.2). Next, the ethical implications related to the project development are analyzed (Section 6.3). Finally, it is evaluated whether the work contributes to any of the 17 Sustainable Development Goals (SDGs) adopted by the United Nations in 2015 (Section 6.4).

## 6.2 Sustainability matrix

The sustainability matrix facilitates the organization of various concepts to be addressed in the analysis. Using this matrix as a guide (see Table 6.1) allows for the evaluation of

the work's impact from environmental, economic, and social perspectives. Additionally, these elements are analyzed both during the development of the work and its operation once the project is implemented. Furthermore, risks and limitations related to the work in these areas are also assessed.

| | **Development of Bachelor's Thesis** | **Project execution** | **Risks and limitations** |
|---|---|---|---|
| **Environmental perspective** | Environmental impact | Environmental impact | Environmental risks and limitations |
| **Economic perspective** | Cost | Viability analysis | Economic risks and limitations |
| **Social perspective** | Personal impact | Social impact | Social risks and limitations |

Table 6.1: Sustainability matrix.

Since this project does not aim to present a product, system, or service that will be operational once its development is completed, an analysis of project execution is not conducted as it is non-existent. However, the impact related to the execution of the work and the associated risks are addressed.

### 6.2.1 Environmental impact

#### Development of Bachelor's Thesis

For the development of this project, previously used equipment was utilized. Both the iPad and the laptop are the student's personal devices, so there was no need to purchase new materials for this project. For the mathematical development of the methodology, an iPad Air (4th generation) was used, and for the implementation and execution of examples in MATLAB, an LG gram 15Z95P laptop was utilized. This resulted in a reduced environmental impact as the $CO_{2e}$ associated with the manufacturing and transportation of new materials were avoided.

As reported in Apple's Product Environmental Report [9] for the iPad Air (4th generation) used, the carbon emissions associated are 82 kg $CO_{2e}$, with the energy distribution for the production and transport phase of the product shown in Table 6.2.

| | | |
|---|---|---|
| **Production** | 83% | 68.06 kg $CO_{2e}$ |
| **Transport** | 5% | 4.10 kg $CO_{2e}$ |
| **Total** | - | **72.16 kg $CO_{2e}$** |

Table 6.2: iPad Air life cycle carbon emissions.

On the other hand, LG does not provide detailed information about each of its products. However, they do publish a Sustainability Report. Since the LG laptop was purchased in early 2020, we refer to the report published for 2019/2020 [48]. This report

does not include information about laptops as they are not among the 7 categories in which they produce the most. Therefore, we use an embodied carbon emission of 200 kg $CO_{2e}$ as suggested in [52].

Thus, considering a lifespan of 5 years for these devices and the weekly usage hours shown in Table 6.3, an estimated savings of 2.16 kg $CO_{2e}$ can be calculated.

| **Device** | **h/week** | **n° weeks** | **kg $CO_{2e}$/year** | **g $CO_{2e}$/h** | **Total** |
|---|---|---|---|---|---|
| iPad | 10 | 20 | 14.43 | 1.65 | **0.33 kg $CO_{2e}$** |
| Laptop | 20 | 20 | 40 | 4.57 | **1.83 kg $CO_{2e}$** |
| **Total** | - | - | - | - | **2.16 kg $CO_{2e}$** |

Table 6.3: Estimated saving of carbon emissions.

On the other hand, the emissions related to the energy consumption during the hours of device usage must be considered. From the product specifications, the average energy consumption during use can be determined. Table 6.4 shows the total hours of usage for each device and the associated kg $CO_{2e}$, considering an average electricity impact of 350 g $CO_{2e}$/kWh.

| **Device** | **h/week** | **n° weeks** | **g $CO_{2e}$/kWh** | **W during usage** | **Total** |
|---|---|---|---|---|---|
| iPad | 10 | 20 | 350 | 10 [8] | **700 g $CO_{2e}$** |
| Laptop | 20 | 20 | 350 | 25 [60] | **3.5 kg $CO_{2e}$** |
| **Total** | - | - | - | - | **4.2 kg $CO_{2e}$** |

Table 6.4: Carbon emissions related to the usage of the laptop and iPad during the development of the project.

Additionally, the transportation of the student from home to the workplace must be considered. The transportation was done by metro, covering a distance of 7 km (3.5 km one way and 3.5 km return). Using the consumption data proposed in [14], 50.13 g $CO_{2e}$/(passenger$\times$km), the total emissions associated with this project are calculated in Eq. 6.1.

$$20 \text{ weeks} \times 5 \text{ workdays/week} \times 7 \text{ km} \times 50.13 \text{ g } CO_{2e}/\text{km} = 35.09 \text{ kg } CO_{2e} \quad (6.1)$$

Thus, by adding the carbon emissions related to transportation and the use of electronic devices, a total of 39.29 kg $CO_{2e}$ is associated with the development of the project. On average, a tree absorbs 25 kg of $CO_2$ per year [26], which amounts to 8.33 kg of $CO_2$ every 4 months. Therefore, it would take 5 trees to offset the emissions produced during the 4 months of project development.

Additionally, the electronic devices come from manufacturers that are environmentally responsible. Both Apple and LG demonstrate a strong commitment to sustainability and the circular economy in their operations and supply chains. Apple [53, 6, 5] publishes annual sustainability reports, uses 100% renewable energy in its facilities, and has set a

goal to be carbon neutral by 2030. Its Code of Conduct imposes strict environmental standards on its suppliers, with regular audits to ensure compliance. Apple also promotes reuse and recycling through programs like "Apple Trade In" and innovations in recycling technologies, such as the Daisy robot, which recovers valuable materials from old devices.

Similarly, LG [59, 47] is committed to sustainability through its annual reports, use of renewable energy, and its goal of being carbon neutral by 2050. LG's Code of Conduct establishes strict environmental standards for its suppliers and conducts regular audits to ensure compliance. LG promotes the circular economy through electronic waste management programs and the development of technologies to recover valuable materials from their old products, aligning with their long-term environmental goals.

### Risks and Limitations

The method implemented is model-based and does not use neural networks. However, the limitations of these methods may require resorting to the use of ANNs. ANNs require training, which typically consumes a significant amount of energy.

Consider the example of training a large neural network like ResNet-50 [38] on a typical GPU setup. The energy consumption depends on the type of GPU and the duration of training. For instance, a NVIDIA V100 GPU [61], commonly used for neural network training, has a maximum power consumption of 300 W. Assuming a training duration of 48 hours, this scenario would lead to an increase of approximately 5.04 kg $CO_{2e}$ in carbon emissions (representing a 7.8% increase in the estimated carbon footprint for the project), as indicated in Eq. 6.2.

$$48 \text{ h} \times 0.3 \text{ kW} \times 350 \text{ g } CO_{2e}/\text{kWh} = 5.04 \text{ kg } CO_{2e} \tag{6.2}$$

Additionally, a very pessimistic scenario could be considered where the metro used for transportation to the workplace is out of service during the project development period. In this case, the student would need to commute to the workplace by car. Using data provided in [24], assuming the use of a medium-efficiency car with emissions of 150 g $CO_{2e}$/km, emissions would be three times higher than those calculated for metro transportation (see Eq. 6.3).

$$20 \text{ weeks} \times 5 \text{ workdays/week} \times 7 \text{ km} \times 150 \text{ g } CO_{2e}/\text{passenger} \times \text{km} = 105 \text{ kg } CO_{2e} \tag{6.3}$$

Apart from these situations, no other environmental risks are identified for this project, as factors such as needing more working hours than necessary or using additional material are entirely dependent on the project author and therefore not considered risks.

However, it is important to mention that the analysis conducted has certain limitations. Firstly, the lack of specific information on carbon emissions for the exact model of laptop used limits the precision of calculating the carbon footprint associated with the device. Additionally, while a lifespan of 5 years is estimated for the devices used, actual durability could vary significantly depending on maintenance, usage patterns, and future technological updates, introducing uncertainty into carbon emission projections. Finally, it is worth noting that the precise measurement of $CO_2$ emissions associated with electricity consumed during device use may be influenced by factors such as regional energy mix, which can vary temporally and geographically.

### 6.2.2 Economic impact

#### Development of Bachelor's Thesis

The project cost is determined by both human resources and the materials used. Human resources include the student's working hours. The Bachelor's thesis consists of a total of 18 ECTS credits, equivalent to 450 hours of dedication (1 ECTS = 25 hours). According to the agreement established by ETSETB, the cost per hour of student work is 10€. Therefore, human resources entail a total cost of 4500€.

Additionally, the cost of the equipment used must be considered. In this case, it involves depreciating these devices over the 4-month duration of the project, assuming a lifespan of 5 years for each device, as shown in Table 6.5.

| Device | Price | Amort./month | Amort./4 months |
|---|---|---|---|
| iPad Air (4th Generation) | 649€ (release date [79]) | 10.82€ | **43.27€** |
| LG Gram 15Z95P | 930€ [4] | 15.50€ | **62.00€** |

Table 6.5: Amortization for 4 months of usage of the electronic devices utilized for the project development. The abbreviation Amort. is used for the term Amortization.

Furthermore, the cost associated with power consumption from using the devices is considered (Table 6.6), although it is minimal and negligible.

| Device | h/week | nº weeks | €/kWh | W during usage | Total |
|---|---|---|---|---|---|
| iPad | 10 | 20 | 0.12 | 10 | **0.24€** |
| Laptop | 20 | 20 | 0.12 | 25 | **1.20€** |
| **Total** | - | - | - | - | **1.44€** |

Table 6.6: Cost associated with the power consumption of the electric devices used. For the €/kWh price, the mean value throughout the day in Spain has been considered [27].

Finally, the MATLAB license must also be considered. If the student did not have a license through UPC due to their student status, it would be necessary to purchase one. In such a case, an academic license can be obtained for a price of 262€ [57] per year, equivalent to 87.33€ for four months of use during the project.

Thus, by adding up the costs associated with labor, the depreciation over 4 months of the electronic equipment used, the electricity consumption of the devices and the MATLAB license, the total cost amounts to 4694.04€ (Eq. 6.4).

$$4500.00€ + 43.27€ + 62.00€ + 1.44€ + 87.33€ = 4694.04€ \tag{6.4}$$

#### Risks and limitations

As seen in the previous section, project costs are difficult to reduce given that few resources have been used and most of the costs come from the student's work hours.

The only scenario where project costs could potentially increase is as previously mentioned: if model-based methods are limited and there is the need to resorting to the use of ANNs. Using the same example of energy consumption for training a neural network and the hourly electricity cost proposed earlier, this would result in an additional cost of 1.73€ per training session (Eq. 6.5). Therefore, this would not increase significantly the project cost, and there is no scenario considered where it would become economically unfeasible.

$$48 \text{ h} \times 0.3 \text{ kW} \times 0.12 \text{ €/kWh} = 1.73\text{€} \tag{6.5}$$

However, it is worth noting that the analysis conducted has limitations. Estimates were used for calculating the cost per hour of student work and the total hours dedicated. Additionally, since the exact hours worked each day are not known, the electricity cost was approximated, although it has been observed that this has a minimal impact.

### 6.2.3 Social impact

#### Development of Bachelor's Thesis

The project undertaken is purely academic and aimed solely at advancing scientific research. Additionally, event cameras are sensors that have recently entered the market, primarily used for research purposes at present. For these reasons, this project has not had a significant impact on ethical standards at a personal or professional level.

On the other hand, it is important to mention that the manufacturers of the equipment used adhere to ethical codes of conduct. Both Apple and LG have established codes of conduct and business ethics [7, 46] that underpin their operations and guide the behavior of their employees and business partners. This code focuses on fundamental principles such as integrity, promoting honesty and transparency in all corporate interactions and communications. It also emphasizes confidentiality, urging employees to protect sensitive company and customer information. Both companies promote an inclusive and respectful work environment, valuing diversity and condemning any form of discrimination or harassment. Furthermore, all employees are expected to strictly comply with local and international laws and regulations, maintaining rigorous ethical standards in all company activities. Lastly, Apple and LG are committed to environmental and social responsibility, seeking to minimize their environmental impact and make positive contributions to the communities in which they operate.

#### Risks and limitations

Despite being a project developed exclusively for academic purposes and the advancement of scientific knowledge, the implementation of perception methods, such as the one used in this study, carries social risks and potential consequences.

Perception methods serve as foundational technologies across diverse sectors including robotics, navigation, aviation, and drones, among others. These methods empower systems to operate autonomously, thereby potentially displacing human labor that can be substituted by machines. This displacement poses risks of economic insecurity and job loss, particularly for roles that are repetitive or easily automated. While these technologies may phase out certain jobs, they also have the capacity to create new opportunities, requiring proactive education and training initiatives to equip the workforce with skills

aligned with emerging technological demands.

Moreover, accessibility to these technologies varies globally, leading to disparities where some regions benefit disproportionately from advancements while others lag behind. This digital divide exacerbates socio-economic inequalities and underscores the importance of equitable distribution and inclusive development of technological capabilities.

Furthermore, the ethical implications of perception technologies extend beyond economic impacts. Concerns arise regarding privacy, surveillance, and misuse. In military contexts, for instance, these technologies can be deployed for reconnaissance and weaponry, altering the dynamics of warfare and potentially leading to ethical dilemmas concerning civilian casualties and the conduct of armed conflicts. The dual-use nature of these technologies necessitates robust ethical frameworks and international agreements to mitigate risks and ensure responsible deployment.

On the other hand, it should be noted that the analysis of social impact conducted is constrained. This limitation arises because in fields where technologies similar to those studied in this project could be employed for malicious purposes, such as defense systems and military applications, secrecy prevails, making it difficult to ascertain if event cameras are being utilized for such uses.

## 6.3 Ethical implications

As mentioned throughout the work, event cameras are relatively new sensors with significant potential due to the advantages they offer over traditional cameras. However, the information they provide differs from that captured by conventional cameras, needing further research and development of new methods. Furthermore, as explained, this project stems from the need for a perception system for Borinot that can operate in highly dynamic situations. However, it is important to note that the development of Borinot does not address any specific social need; its primary aim is to contribute to the advancement of scientific knowledge.

As explained in the section on social risks, working on perception algorithms can lead to uncontrollable consequences. In addition to the outcomes explained previously, two more aspects must be mentioned: privacy concerns, accountability and transparency.

The deployment of advanced perception systems in public and private spaces could lead to increased surveillance, raising concerns about privacy and data security. The ability to monitor and record activities in real-time might be misused, leading to unauthorized data collection and potential violations of individuals' privacy rights.

Implementing complex autonomous systems raise questions about accountability and transparency. It is essential to establish clear guidelines and frameworks to ensure that developers, users, and organizations deploying these technologies are accountable for their impacts. First and foremost, clear attribution of responsibility is vital. Developers and researchers, who are at the forefront of creating and testing these systems, must adhere to stringent ethical standards. They have a duty to ensure that their work does

not inadvertently cause harm.

Transparency in operation involves making the workings of these technologies, the data they collect, and the uses of that data clear to the public. Organizations and developers should provide accessible and understandable information, which helps build trust with users and the public by demonstrating a commitment to ethical practices and accountability. Moreover, transparency in intelligent systems becomes more challenging when using ANNs, as they often function as black boxes that are difficult to interpret. In this work, a model-based method is implemented specifically to interpret the results and understand the system's operation.

Finally, it is important to state that this work is in compliance with the principles of the UPC Code of Ethics, which emphasizes honesty, integrity, and accountability, and aligns with the UPC Code of Research Integrity, which ensures transparency, reliability, and ethical conduct in research.

## 6.4 Relation to the SDGs

To conclude the sustainability analysis, it is assessed whether the work contributes to any of the 17 Sustainable Development Goals (SDGs). Adopted by the United Nations in 2015, the SDGs consist of 169 targets related to sustainability. This work has identified contributions to two of these goals: SDG 8 (Decent Work and Economic Growth) and SDG 9 (Industry, Innovation, and Infrastructure).

On one hand, the work contributes to SDG 8 by advancing knowledge in perception systems, commonly used in autonomous systems. These autonomous systems can assist people in industrial environments, substituting them in more costly tasks and enhancing work quality. Moreover, this contributes to increasing economic productivity through innovation and technological modernization.

On the other hand, this work contributes to SDG 9 by advancing knowledge and scientific research, utilizing innovative technology such as event cameras. This research not only explores new avenues in perception systems but also promotes sustainable development by fostering technological innovation in sensing technologies.

# Bibliography


[1] H. Akolkar, S. Ieng, and R. Benosman. "Real-Time High Speed Motion Prediction Using Fast Aperture-Robust Event-Driven Visual Flow". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.01 (2022), pp. 361–372. DOI: 10.1109/TPAMI.2020.3010468.

[2] Ignacio Alzugaray and Margarita Chli. "ACE: An efficient asynchronous corner tracker for event cameras". In: *2018 International Conference on 3D Vision (3DV)*. 2018, pp. 653–661. DOI: 10.1109/3DV.2018.00080.

[3] Ignacio Alzugaray and Margarita Chli. "Asynchronous corner detection and tracking for event cameras in real time". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3177–3184. DOI: 10.1109/LRA.2018.2849882.

[4] Amazon. *LG gram 15Z95P laptop 15.6" ultra-lightweight, IPS, FHD (1920 x 1080), Intel Core i7, 16GB RAM, 512GB SSD, Windows 11 Home, 80Wh battery, Alexa built-in, 2X USB-C, HDMI, USB-A – black*. URL: https://www.amazon.com/pulgadas-ultraligero-Windows-bater%C3%ADa-integrado/dp/B09G3HG3N1?language=en_US&currency=EUR.

[5] Apple. *Apple 2030: A plan as innovative as our products*. URL: https://www.apple.com/environment/.

[6] Apple. *Apple announces major progress toward climate goals ahead of Earth day*. URL: https://www.apple.com/sg/newsroom/2023/04/apple-announces-major-progress-toward-climate-goals-ahead-of-earth-day/.

[7] Apple. *Ethics and compliance*. URL: https://www.apple.com/compliance/policies/.

[8] Apple. *iPad Air (4th generation) - Technical specifications*. URL: https://support.apple.com/en-us/111905.

[9] Apple. *iPad Air (4th generation) product environmental report*. URL: https://www.apple.com/th/environment/pdf/products/ipad/iPadAir_PER_sept2020.pdf.

[10] Ryad Benosman et al. "Event-based visual flow". In: *IEEE Transactions on Neural Networks and Learning Systems* 25.2 (2014), pp. 407–417. DOI: 10.1109/TNNLS.2013.2273537.

[11] Ryad B. Benosman et al. "Asynchronous frameless event-based optical flow". In: *Neural networks : the official journal of the International Neural Network Society* 27 (2012), pp. 32–7. URL: https://api.semanticscholar.org/CorpusID:17407641.

[12] Samuel Bryner et al. "Event-based, direct camera tracking from a photometric 3D map using nonlinear optimization". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 325–331. DOI: 10.1109/ICRA.2019.8794255.

[13] Luis A Camuñas-Mesa et al. "On the use of orientation filters for 3D reconstruction in event-driven stereo vision". In: *Frontiers in Neuroscience* 8 (2014), p. 48.

[14] Generalitat de Catalunya. *Practical guide for calculating greenhouse gas (GHG) emissions*. URL: https://canviclimatic.gencat.cat/web/.content/04_ACTUA/Com_calcular_emissions_GEH/guia_de_calcul_demissions_de_co2/190301_Practical-guide-calculating-GHG-emissions_OCCC.pdf.

[15] Andrea Censi and Davide Scaramuzza. "Low-latency event-based visual odometry". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 703–710. DOI: 10.1109/ICRA.2014.6906931.

[16] William Chamorro. "Event based SLAM". PhD thesis. Universitat Politècnica de Catalunya, Feb. 2023. URL: http://hdl.handle.net/2117/396555.

[17] William Chamorro, Juan Andrade-Cetto, and Joan Solà. "High speed event camera tracking". In: *British Machine Vision Conference*. 2020. URL: https://arxiv.org/abs/2010.02771.

[18] William Chamorro, Joan Solà, and Juan Andrade-Cetto. "Event-based line SLAM in real-time". In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 8146–8153. DOI: 10.1109/LRA.2022.3187266.

[19] William Chamorro, Joan Solà, and Juan Andrade-Cetto. "Event-IMU fusion strategies for faster-than-IMU estimation throughput". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023, pp. 3976–3983. DOI: 10.1109/CVPRW59228.2023.00414.

[20] Peiyu Chen, Weipeng Guan, and Peng Lu. "ESVIO: Event-based stereo visual inertial odometry". In: *IEEE Robotics and Automation Letters* 8.6 (2023), pp. 3661–3668. DOI: 10.1109/LRA.2023.3269950.

[21] R.T. Collins. "A space-sweep approach to true multi-image matching". In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1996, pp. 358–363. DOI: 10.1109/CVPR.1996.517097.

[22] Laurent Dardelet, Ryad Benosman, and Sio-Hoi Ieng. "An event-by-event feature detection and tracking invariant to motion direction and velocity". In: Institute of Electrical and Electronics Engineers (IEEE), Nov. 2021. DOI: 10.36227/techrxiv.17013824.

[23] Ziluo Ding et al. "Spatio-Temporal Recurrent Networks for Event-Based Optical Flow Estimation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36 (June 2022), pp. 525–533. DOI: 10.1609/aaai.v36i1.19931.

[24] Instituto para la Diversificación y Ahorro de la Energía (IDAE). *Histórico de guía de consumo y emisiones*. URL: https://coches.idae.es/historico-emisiones-consumos.

[25] Pierre Drap and Julien Lefèvre. "An exact formula for calculating inverse radial lens distortions". In: *Sensors* 16.6 (2016). DOI: 10.3390/s16060807.

[26] ecotree. *How much CO2 does a tree absorb?* URL: https://ecotree.green/en/how-much-co2-does-a-tree-absorb.

[27] Jorge Morales Fernández. *¿Cuánto cuesta el kilovatio hora de luz (kWh) en España?* URL: https://tarifaluzhora.es/info/precio-kwh.

[28] Mohsen Firouzi and Jorg Conradt. "Asynchronous event-based cooperative stereo matching using neuromorphic silicon retinas". In: *Neural Processing Letters* 43 (Apr. 2016), pp. 311–326. DOI: 10.1007/s11063-015-9434-5.

[29] G. Gallego, H. Rebecq, and D. Scaramuzza. "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00407.

[30] Guillermo Gallego et al. "Event-based, 6-DOF camera tracking from photometric depth maps". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.10 (2018), pp. 2402–2412. DOI: 10.1109/TPAMI.2017.2769655.

[31] Mathias Gehrig et al. "DSEC: A stereo event camera dataset for driving scenarios". In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 4947–4954. URL: https://api.semanticscholar.org/CorpusID:232170230.

[32] Mathias Gehrig et al. "E-RAFT: Dense optical flow from event cameras". In: *International Conference on 3D Vision (3DV)*. 2021.

[33] Arren Glover et al. "LuvHarris: A practical corner detector for event-cameras". In: *CoRR* (May 2021).

[34] Guangsha Guo et al. "Event-guided image super-resolution reconstruction". In: *Sensors* 23 (Feb. 2023), p. 2155. DOI: 10.3390/s23042155.

[35] Antea Hadviger et al. "Feature-based event stereo visual odometry". In: *European Conference on Mobile Robots (ECMR) 2021*. July 2021.

[36] Jesse Hagenaars, Federico Paredes-Vallés, and Guido De Croon. "Self-supervised learning of event-based optical flow with spiking neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7167–7179.

[37] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. 2nd ed. Cambridge University Press, 2004.

[38] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[39] J. Hidalgo-Carrio, D. Gehrig, and D. Scaramuzza. "Learning Monocular Dense Depth from Events". In: *2020 International Conference on 3D Vision (3DV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2020, pp. 534–542. DOI: 10.1109/3DV50981.2020.00063.

[40] Geoffry Hinton, Nitish Srivastava, and Kevin Swersky. *Lecture 6a: Overview mini-batch gradient descent*. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[41] Sio-Hoi Ieng et al. "Neuromorphic event-based generalized time-based stereovision". In: *Frontiers in Neuroscience* 12 (2018). DOI: 10.3389/fnins.2018.00442.

[42] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. "Real-time 3D reconstruction and 6-DoF tracking with an event camera". In: *European Conference on Computer Vision*. 2016. URL: https://api.semanticscholar.org/CorpusID:26324573.

[43] Jürgen Kogler, Martin Humenberger, and Christoph Sulzbachner. "Event-based stereo matching approaches for frameless address event stereo data." In: *Advances in Visual Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2011, pp. 674–685.

[44] Xavier Lagorce et al. "HOTS: A hierarchy of event-based time-surfaces for pattern recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.7 (2017), pp. 1346–1359. DOI: 10.1109/TPAMI.2016.2574707.

[45] Chankyu Lee et al. "Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks". In: *European Conference on Computer Vision (ECCV) 2020*. arXiv: 2003.06696. URL: https://arxiv.org/abs/2003.06696.

[46] LG. *LG code of ethics*. URL: https://www.lgensol.com/en/company-management-ethics%20n.

[47] LG. *LG publica su informe global de sostenibilidad 2022-2023*. URL: https://www.lg.com/es/acerca-de-lg/comunicados-de-prensa/informe-global-sostenibilidad-2022-2023/.

[48] LG. *LG sustainability reports*. URL: https://www.lg.com/global/sustainability/reports.

[49] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128× 128 120 dB 15 $\mu$s latency asynchronous temporal contrast vision sensor". In: *Solid-State Circuits, IEEE Journal of* 43 (Mar. 2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[50] Min Liu and Tobi Delbrück. "Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors". In: *British Machine Vision Conference*. 2018. URL: https://api.semanticscholar.org/CorpusID:52283776.

[51] Hugh Christopher Longuet-Higgins and K. Prazdny. "The interpretation of a moving retinal image". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 208 (1980), pp. 385–397.

[52] N. Lövehagen et al. "Assessing embodied carbon emissions of communication user devices by combining approaches". In: *Renewable and sustainable energy reviews* 183 (2023), p. 113422. DOI: https://doi.org/10.1016/j.rser.2023.113422.

[53] MacRumors. *Apple shares 2023 environmental progress report ahead of Earth day*. URL: https://www.macrumors.com/2023/04/19/apple-shares-2023-environmental-progress-report/.

[54] Misha A. Mahowald. *An analog VLSI system for stereoscopic vision*. Springer New York, NY, 1994. URL: https://api.semanticscholar.org/CorpusID:60778343.

[55] Jacques Manderscheid et al. "Speed invariant time surface for learning to detect corner points with event-based cameras". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 10237–10246. URL: https://api.semanticscholar.org/CorpusID:85529508.

[56] Josep Martí-Saumell et al. *Borinot: an open thrust-torque-controlled robot for research on agile aerial-contact motion*. 2023. arXiv: 2307.14686 [cs.RO].

[57] MATLAB. *Precios y licencias*. URL: https://es.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=edu.

[58] Jun Nagata, Yusuke Sekikawa, and Yoshimitsu Aoki. "Optical flow estimation by matching time surface with event-based cameras". In: *Sensors* 21 (Feb. 2021), p. 1150. DOI: 10.3390/s21041150.

[59] LG Newsroom. *LG releases 2022-2023 sustainability report*. URL: https://www.lgnewsroom.com/2023/07/lg-releases-2022-2023-sustainability-report/.

[60] NOTEBOOKCHECK. *Análisis del LG gram 15Z90P: portátil de 15 pulgadas de 1,1 kg ( 2,4 libras)*. URL: https://www.notebookcheck.org/Analisis-del-LG-Gram-15Z90P-portatil-de-15-pulgadas-de-1-1-kg-2-4-libras.596131.0.html.

[61] NVIDIA. *NVIDIA TE SL A V100 GPU accelerator*. URL: https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf.

[62] Garrick Orchard et al. "A spiking neural network architecture for visual motion estimation". In: *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 2013, pp. 298–301. DOI: 10.1109/BioCAS.2013.6679698.

[63] Marc Osswald et al. "A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems". In: *Scientific Reports* 7 (2017). URL: https://api.semanticscholar.org/CorpusID:205698386.

[64] Federico Paredes-Valles, Kirk Yannick Willehm Scheper, and Guido C. H. E. de Croon. "Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: from events to global motion perception". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.8 (Aug. 2020), pp. 2051–2064. DOI: 10.1109/tpami.2019.2903179.

[65] Ewa Piatkowska, Ahmed Nabil Belbachir, and Margrit Gelautz. "Cooperative and asynchronous stereo vision for dynamic vision sensors". In: *Measurement Science and Technology* 25.5 (2014), p. 055108.

[66] Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. "EMVS: Event-based multi-view stereo". In: *British Machine Vision Conference*. 2016. URL: https://api.semanticscholar.org/CorpusID:9771557.

[67] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization". In: *British Machine Vision Conference*. 2017. URL: https://api.semanticscholar.org/CorpusID:30723444.

[68] Henri Rebecq et al. "EMVS: Event-based multi-view stereo—3D reconstruction with an event camera in real-time". In: *International Journal of Computer Vision* 126 (2017), pp. 1394–1414. URL: https://api.semanticscholar.org/CorpusID:255097813.

[69] Henri Rebecq et al. "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 593–600. DOI: 10.1109/LRA.2016.2645143.

[70] Paul Rogister et al. "Asynchronous event-based binocular stereo matching". In: *IEEE Transactions on Neural Networks and Learning Systems* 23 (2012), pp. 347–353. URL: https://api.semanticscholar.org/CorpusID:17693733.

[71] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[72] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. "Fast event-based optical flow estimation by triplet matching". In: *IEEE Signal Processing Letters* 29 (2022), pp. 2712–2716. DOI: 10.1109/lsp.2023.3234800.

[73] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. "Secrets of event-based optical flow". In: *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 628–645. ISBN: 9783031197970. DOI: 10.1007/978-3-031-19797-0_36.

[74] Shintaro Shiba et al. "Secrets of event-based optical flow, depth and ego-motion estimation by contrast maximization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), pp. 1–18. DOI: 10.1109/TPAMI.2024.3396116.

[75] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. "A micro Lie theory for state estimation in robotics". In: *arXiv preprint arXiv:1812.01537* (2018).

[76] Yi Tian and J. Andrade-Cetto. "Event transformer FlowNet for optical flow estimation". In: *British Machine Vision Conference*. 2022. URL: https://api.semanticscholar.org/CorpusID:254019864.

[77] Yi Tian and Juan Andrade-Cetto. "Egomotion from event-based SNN optical flow". In: *ICONS '23: 2023 International Conference on Neuromorphic Systems*. Aug. 2023, pp. 1–8. DOI: 10.1145/3589737.3605978.

[78] David Weikersdorfer et al. "Event-based 3D SLAM with a depth-augmented dynamic vision sensor". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 359–364. DOI: 10.1109/ICRA.2014.6906882.

[79] Wikipedia. *iPad Air (4th generation)*. URL: https://en.wikipedia.org/wiki/IPad_Air_(4th_generation).

[80] Wikipedia. *Random sample consensus*. URL: https://en.wikipedia.org/wiki/Random_sample_consensus.

[81] Shaojie Shen Xiuyuan Lu Yi Zhou. "Event-based visual inertial velometer". In: *arXiv preprint arXiv:2311.18189* (2023). DOI: https://doi.org/10.48550/arXiv.2311.18189.

[82] Chengxi Ye et al. "Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5831–5838. DOI: 10.1109/IROS45743.2020.9341224.

[83] Jiqing Zhang et al. "Spiking transformers for event-based single object tracking". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8791–8800. DOI: 10.1109/CVPR52688.2022.00860.

[84] Yi Zhou, Guillermo Gallego, and Shaojie Shen. "Event-based stereo visual odometry". In: *IEEE Transactions on Robotics* 37.5 (2021), pp. 1433–1450. DOI: 10.1109/TRO.2021.3062252.

[85] Alex Zhu et al. "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras". In: *Robotics: Science and Systems XIV*. RSS2018. Robotics: Science and Systems Foundation, June 2018. DOI: 10.15607/rss.2018.xiv.062.

[86] Alex Zihao Zhu et al. "Live Demonstration: Unsupervised Event-Based Learning of Optical Flow, Depth and Egomotion". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 1694–1694. DOI: 10.1109/CVPRW.2019.00216.

Chapter A

# Pose interpolation

Given an initial pose $\mathbf{T}(t_{\text{ini}})$ and a final pose $\mathbf{T}(t_{\text{fin}})$, we want to obtain $n$ intermediate poses following a geodesic path. To achieve this, tools from Lie algebra are utilized.

## A.1 Lie algebra introduction

A Lie group is a smooth manifold that also has a group structure such that the group operation (multiplication and taking inverses) is smooth (differentiable). A Lie algebra is an algebraic structure associated with a Lie group that captures the local properties of the group in terms of a vector space equipped with a Lie bracket (an antisymmetric bilinear product satisfying the Jacobi identity).

Without delving into too much detail (for further information, see [75]), it is important to understand the two main operations:

- The exponential operation in Lie algebra allows relating elements of the Lie algebra (which capture the local properties of a Lie group) to elements of the Lie group itself. The exponential of an element $\mathbf{X}$ in the Lie algebra is defined through the Taylor series, adapted for special matrices:

$$exp(\mathbf{X}) = \sum_{k=0}^{\infty} \frac{\mathbf{X}^k}{k!} \tag{A.1}$$

- The logarithm in Lie algebra provides the local inverse of the exponential operation, allowing the recovery of elements from the Lie algebra based on elements from the Lie group. Given $\mathbf{R}$ in the Lie group, the logarithm is computed using a special inverse series:

$$log(\mathbf{R}) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} (\mathbf{I} - \mathbf{R})^k \tag{A.2}$$

  where $\mathbf{I}$ is the identity matrix.

In Figure A.1, the relationship between Lie algebra and Lie group is depicted, illustrating the functionality of the described operations.

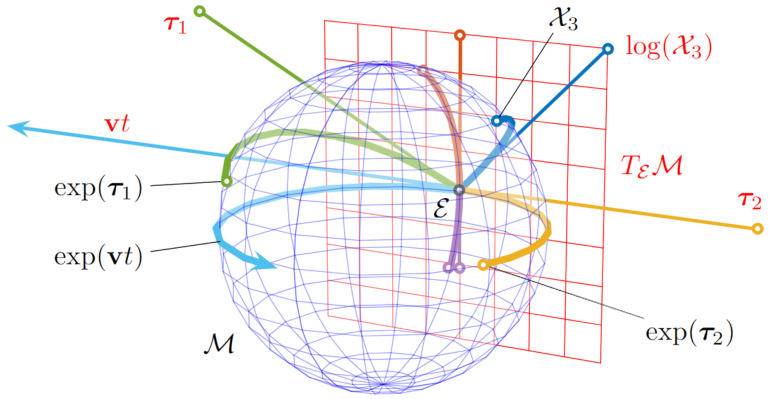



Figure A.1: Representation of the relationship between the Lie group and the Lie algebra. The Lie algebra $T\epsilon M$(red plane) is the tangent space to the Lie group's manifold $M$(here represented as a blue sphere) at the identity $\epsilon$. Through the exponential mapping, each straight path passing through the origin in the Lie algebra $\mathbf{v}t$ produces a geodesic path $exp(\mathbf{v}t)$ around the manifold. Conversely, each element of the group has an equivalent in the Lie algebra. Figure taken from [75].

## A.2 Pose interpolation

Pose interpolation between two rigid transformations $\mathbf{T}_{\text{ini}} = \mathbf{T}(t_{\text{ini}})$ and $\mathbf{T}_{\text{fin}} = \mathbf{T}(t_{\text{fin}})$ is a common problem in robotics and computer graphics. By using Lie algebra, we can perform this interpolation effectively.

First, the difference between the two poses is computed as an element of the Lie group.

$$\mathbf{T}_{\text{ini}}^{\text{fin}} = \mathbf{T}_{\text{fin}}\mathbf{T}_{\text{ini}}^{-1} \tag{A.3}$$

Next, the matrix $\mathbf{T}_{\text{ini}}^{\text{fin}}$ is transferred to the Lie algebra by taking its logarithm:

$$\mathbf{S}_{\text{ini}}^{\text{fin}} = log(\mathbf{T}_{\text{ini}}^{\text{fin}}) \tag{A.4}$$

From the matrix $\mathbf{S}_{\text{ini}}^{\text{fin}}$, we extract the components representing the angle $\theta$ and the unit vector $\hat{\mathbf{s}}$ that defines the axis of rotation, using $\mathbf{S}_{\text{ini}}^{\text{fin}} = \theta\hat{\mathbf{s}}$.

Since we want to interpolate $n$ poses $\mathbf{T}_i$ $(i = 1, 2, ..., n)$, we need to interpolate the angle $\theta$ as a function of $i$:

$$\theta_i = \frac{i}{n}\theta \tag{A.5}$$

$$\hat{\mathbf{s}}_i = \hat{\mathbf{s}} \tag{A.6}$$

Then, the rotation matrix for pose $i$ can be calculated by returning to the Lie group using the exponential and expressing the rotational transformation in terms of the initial pose:

$$\mathbf{R}_i = exp(\theta_i\hat{\mathbf{s}}_i)\mathbf{T}_{\text{ini}} \tag{A.7}$$

However, the linear displacement must also be interpolated in accordance with the defined geodesic path.

On one hand, we define $\mathbf{V}_i$ to provide the interpolated linear displacement.

$$\mathbf{V}_i = \frac{i}{n}\mathbf{I} + (\frac{1-cos(\theta_i)}{\theta^2})\mathbf{S} + \frac{i/n - sin(\theta_i)/\theta}{\theta^2}\mathbf{S}^2 \tag{A.8}$$

where $\mathbf{S}$ is the antisymmetric matrix associated with $\hat{\mathbf{s}}$.

On the other hand, this interpolated linear displacement must be adjusted to match the angular interpolation. To achieve this, the matrix $\mathbf{U}_{\text{ini}}^{\text{fin}}$ is defined as a function of $\theta$ and $\hat{\mathbf{s}}$, and is generally expressed as a series or an approximation that adjusts the linear displacement.

$$\mathbf{U}_{\text{ini}}^{\text{fin}} = exp(-\frac{\theta}{2}\hat{\mathbf{s}}) \cdot \left[\frac{sin(\theta/2)}{\theta/2}\mathbf{I} + (1 - \frac{sin(\theta/2)}{\theta/2})\hat{\mathbf{s}}\hat{\mathbf{s}}^\top\right] \cdot \mathbf{r}_{\text{ini}}^{\text{fin}} \tag{A.9}$$

where $\mathbf{r}_{\text{ini}}^{\text{fin}}$ denotes the linear displacement vector contained in $\mathbf{T}_{\text{ini}}^{\text{fin}}$.

Thus, the interpolated linear displacement for pose $i$ is calculated using the following expression:

$$\mathbf{r}_{\text{i}} = \mathbf{V}_i\mathbf{U}_{\text{ini}}^{\text{fin}} \tag{A.10}$$

Hence, the interpolated pose $i$ between $\mathbf{T}_{\text{ini}}$ and $\mathbf{T}_{\text{fin}}$ is obtained:

$$\mathbf{T}_i = (\mathbf{R}_i|\mathbf{r}_i) \tag{A.11}$$

Figure A.2 shows an example where the interpolation of 7 poses between the initial and final poses is performed.

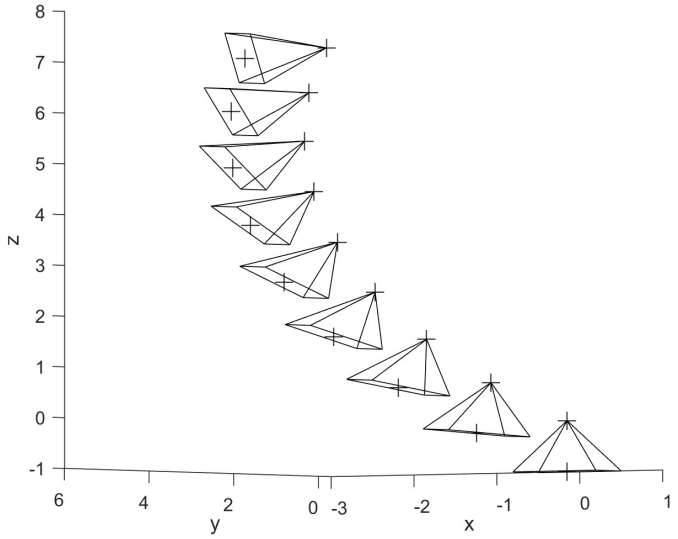



Figure A.2: Interpolation example of 7 poses between an initial and a final pose.

Appendix B

# Contrast maximization gradient development

The objective is to find the gradient of the contrast of the IWE with respect to the inverse depth of the patch, $\frac{\partial C_{\mathbf{W}}}{\partial \rho}$.

## B.1 Expressing the contrast in terms of patches

Firstly, the contrast is defined as the variance of the IWE:

$$C_{\mathbf{W}} = \sigma^2(\mathbf{W}) = \frac{1}{N_{pix}} \sum_{i=1}^{N_{pix}} (w_i - \mu_{\mathbf{W}})^2 \tag{B.1}$$

$$\mu_{\mathbf{W}} = \frac{1}{N_{pix}} \sum_{j=1}^{N_{pix}} w_j \tag{B.2}$$

where $\mathbf{W}$ denotes the matrix containing the scores of each pixel in the IWE, $\sigma^2(\mathbf{W})$ is the variance of $\mathbf{W}$, $N_{\text{pix}}$ is the number of pixels in the image, $w_i$ is the score of pixel $i$ in the IWE, and $\mu_{\mathbf{W}}$ is the mean value of $\mathbf{W}$.

However, these expressions are given in terms of the score and inverse depth of each pixel, so the expression of $C_{\mathbf{W}}$ must be adapted to express it in terms of patches. With this purpose in mind, the resolution of the IWE is reduced such that each patch is assigned the mean value of the scores of the pixels contained within it.

$$w_{p,i} = \frac{1}{N} \sum_{p \in P_i} w_p \tag{B.3}$$

where $w_{p,i}$ is the score of the patch $P_i$, $N$ the number of pixels in the patch $P_i$ and $p$ iterates each pixel in $P_i$.

By reducing the resolution, the expression for the contrast of the IWE remains the same as in Eq. B.1 but expressed in terms of the scores of the patches:

$$C_{\mathbf{W}_p} = \sigma^2(\mathbf{W}_p) = \frac{1}{N_p}\sum_{i=1}^{N_p}(w_{p,i} - \mu_{\mathbf{W}_p})^2 \tag{B.4}$$

$$\mu_{\mathbf{W}_p} = \frac{1}{N_p}\sum_{j=1}^{N_p} w_{p,j} \tag{B.5}$$

where $\mathbf{W}_p$ denotes the matrix containing the scores of each of the patches, $\sigma^2(\mathbf{W}_p)$ is the variance of $\mathbf{W}_p$, $N_p$ is the number of patches in the image, $w_{p,i}$ is the score of patch $P_i$, and $\mu_{\mathbf{W}_p}$ is the mean value of $\mathbf{W}_p$.

## B.2 Gradient computation

Substituting the expression for the mean value (Eq. B.5) into Eq. B.4:

$$C_{\mathbf{W}_p} = \frac{1}{N_p}\sum_{i=1}^{N_p}(w_{p,i} - \frac{1}{N_p}\sum_{j=1}^{N_p} w_{p,j})^2 = \frac{1}{N_p}\sum_{i=1}^{N_p}\left[(1-\frac{1}{N_p})w_{p,i} - \frac{1}{N_p}\sum_{j\neq i} w_{p,j}\right]^2 \tag{B.6}$$

It is observed that the expression for $C_{\mathbf{W}_p}$ does not have a direct dependence on $\rho$. However, the score of each patch is related to the inverse depth. Therefore, we suggest splitting the gradient into two parts using the chain rule:

$$\frac{\partial C_{\mathbf{W}_p}}{\partial \rho_i} = \frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}}\frac{\partial w_{p,i}}{\partial \rho_i} \tag{B.7}$$

We propose to perform the first derivative ($\frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}}$) analytically and the second derivative ($\frac{\partial w_{p,i}}{\partial \rho_i}$) numerically. By analytically differentiating the expression for $C_{\mathbf{W}_p}$ in Eq. B.6 with respect to the score of patch $P_i$:

$$\begin{aligned}\frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}} =& \frac{1}{N_p}\left[2(1-\frac{1}{N_p})\left((1-\frac{1}{N_p})w_{p,i} - \frac{1}{N_p}\sum_{j\neq i} w_{p,j}\right)\right] \\ &- \frac{2}{N_p}\sum_{k\neq i}\left[(1-\frac{1}{N_p})w_{p,k} - \frac{1}{N_p}\sum_{m\neq k} w_{p,m}\right]\end{aligned} \tag{B.8}$$

By adding $\frac{1}{N_p}w_{p,i}$ in the summation over $j$ and $\frac{1}{N_p}w_{p,k}$ in the summation over $m$, we obtain $\mu_{\mathbf{W}_p} = \frac{1}{N_p}\sum_{j=1}^{N_p} w_{p,j}$. Thus, by grouping terms:

$$\frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}} = \frac{2}{N_p}\left[(1-\frac{1}{N_p})(w_{p,i} - \mu_{\mathbf{W}_p}) - \frac{1}{N_p}\sum_{k\neq i}(w_{p,k} - \mu_{\mathbf{W}_p})\right] \tag{B.9}$$

By adding and subtracting $\frac{1}{N_p}(w_{p,i} - \mu_{\mathbf{W}_p})$ to the expression, the complete summation over $k$ is obtained:

$$\frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}} = \frac{2}{N_p}\left[(1-\frac{1}{N_p})(w_{p,i} - \mu_{\mathbf{W}_p}) + \frac{1}{N_p}(w_{p,i} - \mu_{\mathbf{W}_p}) - \frac{1}{N_p}\sum_{k=1}^{N_p}(w_{p,k} - \mu_{\mathbf{W}_p})\right] \tag{B.10}$$

Grouping terms and separating the summation over $k$ into two parts, we obtain the simplified expression:

$$\frac{\partial C_{\mathbf{W}_p}}{\partial w_{p,i}} = \frac{2}{N_p}\left[(1 - \cancel{\frac{1}{N_p}} + \cancel{\frac{1}{N_p}})(w_{p,i} - \mu_{\mathbf{W}_p}) - \cancel{\frac{1}{N_p}\sum_{k=1}^{N_p} w_{p,k}} + \cancel{\frac{1}{N_p} N_p \mu_{\mathbf{W}_p}}\right] = \\ = \frac{1}{N_p}(w_{p,i} - \mu_{\mathbf{W}_p}) \tag{B.11}$$

Thus, knowing the IWE for the previous candidate inverse depth, the gradient corresponding to each patch can be calculated as follows:

$$\frac{\partial C_{\mathbf{W}_p}(k)}{\partial w_{p,i}(k)} = \frac{2}{N_p}(w_{p,i} - \mu_{\mathbf{W}_p}(k)) \cdot \frac{w_{p,i}(k) - w_{p,i}(k-1)}{\rho_i(k) - \rho_i(k-1)} \tag{B.12}$$

where $k$ and $k-1$ indicate the optimization iteration number corresponding to the parameter.