Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(37^{th}cycle)

Visual Perception for Service Robotics: from Frame-based to Event-based sensors

By

Chiara Boretti

******

Supervisor(s):
Prof. Gianluca Setti, Supervisor
Prof. Marcello Chiaberge, Co-Supervisor

Politecnico di Torino
2026

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Chiara Boretti
2026

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Abstract

Autonomous service robotics represents a rapidly advancing field of research. The primary goal of a service robot is to assist humans and enhance the quality of their everyday lives. These robots are specifically designed to perform complex, dangerous, or physically demanding tasks, enabling greater safety and convenience for users in a fully autonomous manner. A critical aspect of a robotic platform's autonomy is its ability to perceive and interpret the environment effectively. Various sensors can be used to gather information about the surroundings, with vision-based sensors being among the most widely utilized. Visual data acquisition and processing play an essential role in enabling robots to understand and interact with the environment successfully, allowing tasks such as navigation and human-robot interaction. This data can be captured using two primary types of vision systems: traditional frame-based cameras and neuromorphic-inspired vision sensors, known as event-based cameras. Frame-based cameras process sequential images to extract visual cues. Alternatively, event-based vision sensors offer an innovative approach where each pixel operates independently, detecting and transmitting events triggered by brightness changes. This new working principle offers several advantages, including low latency, low power consumption, high dynamic range, and reduced redundancy in information. Neuromorphic vision has emerged as a promising foundation for developing more efficient and lightweight navigation strategies. Inspired by the way biological systems process visual stimuli, these approaches enable faster and more energy-efficient decision-making.

This dissertation explores the usage of biologically inspired strategies and of cutting-edge sensing technologies to develop efficient and versatile visual perception systems for autonomous robotics. The research is divided into two main parts: one focusing on the use of conventional frame-based cameras and the other on exploring the potential usage of event-based sensors. In detail, frame-based cameras are utilized to derive biologically inspired visual cues that facilitate autonomous navigation in

GPS-denied, unexplored environments while minimizing the sensor payload. Central to this approach is the concept of time-to-transit, which represents the time an autonomous platform requires to reach a specific feature detected in the image and present in its surroundings. Extensive experimentation has been conducted to improve the estimation of this quantity by leveraging various deep learning models, effectively addressing the limitations inherent in traditional computer vision techniques. Both simulation and real-world testing validate the feasibility of using time-to-transit as a signal for simple yet effective control strategies. The second part shifts the focus to novel event-based cameras, exploring their potential applications in service robotics. This includes contributing to the enrichment of the number of existing event datasets for tasks such as person detection and tracking, through the development of a new person-based dataset called PEDRo. Additionally, a new pre-processing pipeline, called Memory of Events through Spatial Attention (MESA), is introduced to organize events while preserving the temporal characteristics of event streams. The proposed methodology aims to expand the utility of event-based sensors within contemporary deep learning frameworks without the need to employ more complex neural networks such as recurrent architectures. Results on object classification and detection demonstrate that this technique improves what is achievable using state-of-the-art event representations alone, all while maintaining computational efficiency.

The findings discussed in this dissertation highlight the potential of biologically inspired vision strategies and innovative sensor technologies to advance the efficiency of service robotics platforms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots are increasingly entering our daily environments, from homes and offices to streets and factories. To operate safely, they must recognize obstacles, interact with people, and adapt to unpredictable changes around them. Achieving this requires not only mechanical strength but also sophisticated perception. Such machines are known as service robots, defined as *"a semi-automatic or a fully automatic machine designed to perform useful tasks for humans or equipment. These tasks include handling or serving items, transportation, physical support, guidance and information provision, inspection, surveillance, cooking and food handling, grooming, and cleaning".* The field of service robotics has gained significant attention in research, demonstrating how advanced technologies from various disciplines converge into practical applications. A particularly important milestone within this field is the integration of artificial intelligence (AI) algorithms with mobile robotic platforms. Deep learning-based methods, applied to platforms designed to assist humans and enhance their quality of life, have led to applications in diverse domains such as household management [53], smart cities [133], precision agriculture [150], defense [154], healthcare [84], logistics [221], industrial inspections [231], planetary exploration [152], and even leisure or recreation [200].

Service robots are specifically designed to execute complex, hazardous, or physically demanding tasks, thereby improving safety and convenience for users. In mobile robotics, autonomy is a fundamental aspect. It requires robots to navigate the environment, understand surroundings, recognize objects, obstacles and people, reach the designated area of operation, and execute tasks efficiently. These processes

Fig. 1.1 Comparison between frame-based (on the left) and event-based cameras (on the right): the event-based output highlights motion while reducing redundancy. Picture taken from the website of the Robotics and Perception Group, University of Zurich, Switzerland.

must be carried out with minimal human intervention while respecting predefined time constraints. A crucial aspect to do so and to enhance robotic autonomy is the ability to perceive and interpret the environment effectively. Various sensors can be used to perform environmental perception, with vision-based sensors being among the most widely utilized. Since vision is a primary sense for both humans and animals, visual data acquisition and processing play a pivotal role in enabling autonomous platforms to interact effectively with their surroundings, navigate dynamic environments, and adapt to changes in real-time.

Visual information can be acquired through two primary types of vision systems: traditional frame-based cameras and neuromorphic-inspired vision sensors, also known as event-based cameras. A comparison of the output of both sensors is shown in Figure 1.1. Frame-based cameras provide sequential images at a fixed rate, which can then be processed to extract visual cues, while event-based vision sensors offer an innovative approach where each pixel operates independently, detecting and transmitting events triggered by brightness changes. This novel working principle provides several advantages, including low latency, low power consumption, high dynamic range, and reduced redundancy in information.

Inspired by the way biological systems process visual stimuli, neuromorphic vision has emerged as a promising foundation for developing more efficient, lightweight navigation algorithms and decision-making strategies. Particularly, in this dissertation, we focus on both types of visual data acquisition sensors and we span from the usage of biologically inspired approaches to the exploration of

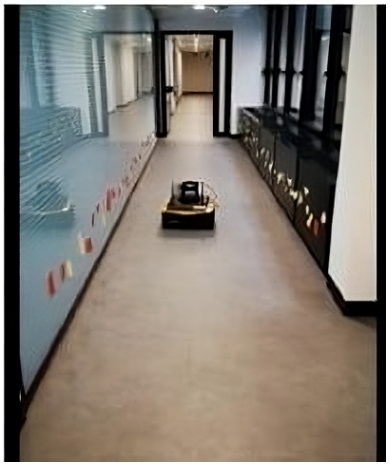


Fig. 1.2 A Jackal robot autonomously navigating in an office-like environment.

cutting-edge sensing technologies to develop efficient and versatile visual perception systems for autonomous robotics.

1.1 Contributions

This doctoral thesis explores the integration of various visual inputs into both state-of-the-art deep learning architectures and traditional computer vision techniques to develop efficient and adaptable visual perception systems for autonomous robotics. The research is divided into two main parts: the first one focuses on conventional frame-based camera outputs, while the second investigates the potential of novel event-based sensors. The findings presented in this dissertation underscore the advantages of biologically inspired vision strategies and cutting-edge sensor technologies in enhancing the efficiency and capabilities of service robotics platforms in multiple tasks.

In summary, the key contributions of this work are:

- A study that aims to enable autonomous navigation in GPS-denied, unexplored environments while minimizing sensor payload by exploiting biologically inspired visual cues obtained from sequences of images captured by a monocular frame-based camera. At the core of this approach is the concept of time-to-transit (TTT), which can be defined as the estimated time an autonomous platform requires to reach a specific visual feature detected in an image. To

obtain reliable TTT values, the study first leverages traditional computer vision techniques, such as Lucas-Kanade optical flow [28], and introduces a strategy to mitigate estimation errors. Then it advances to explore the usage of deep learning models to address the limitations of conventional methods and further improve TTT computation. Finally, through extensive experimentation conducted both in simulated and real-world environments (an example is presented in Figure 1.2), we confirm the feasibility of using accurate time-to-transit values as a reliable signal for simple yet effective control strategies [26].

- An analysis on how to effectively use event-based camera sensors in service robotic, since they offer a promising solution for enhancing perception, particularly in dynamic and complex environments. This work focuses on a key application, namely person detection, which plays a crucial role in various service robotics tasks such as social navigation, intruder detection, and collision avoidance in crowded spaces. To advance research in this field, we introduce PEDRo [23], an event-based dataset specifically designed for person detection. PEDRo is manually labeled and entirely collected using a moving Dynamic Vision Sensor (DVS). By contributing a high-quality dataset tailored to real-world service robotics applications, this work aims to expand the availability of event-based data for critical tasks like person tracking and detection, ultimately improving robotic perception and interaction in dynamic environments.

- The development of a novel pre-processing pipeline to organize events while preserving the temporal characteristics of event streams. The goal is to enhance the usability of event-based sensors within modern deep learning frameworks without relying on complex neural networks, such as recurrent architectures. The initial approach, Memory in Motion (MeMo) [24], is introduced for pupil detection. Building upon this, an improved method, Memory of Events through Spatial Attention (MESA) [20], is developed to make the methodology more dynamic. MESA proves effective in tasks such as object detection and classification, demonstrating that it enhances state-of-the-art event representations while maintaining computational efficiency.

This dissertation presents the research I conducted during my PhD at Politecnico di Torino and at PIC4SeR (Politecnico di Torino Interdepartmental Center for Service Robotics). The thesis encompasses the projects that leads to the publication of several

papers [26, 27, 23, 24, 216, 20] to which I contributed both methodologically and experimentally, either as a principal investigator or as an active collaborator alongside my fellow PhD colleagues.

Beyond the scope of this thesis, I participated in additional research projects that enriched my experience as a researcher and expanded my expertise in machine learning. For instance, I contributed to the evaluation of a novel pruning technique, namely a Multiply-And-Max/Min Neuron Paradigm for Aggressively Prunable Deep Neural Networks (MAM) [19, 160], applied to state-of-the-art architectures such as Vision Transformer [54].

1.2 Thesis Organization

The thesis is structured in three main parts, each of which collects similar content according to the type of visual input analyzed and the research context.

Part I provides the essential background knowledge for this dissertation. Chapter 2 introduces key principles of Computer Vision, highlighting classical algorithms, particularly those relevant to later sections. In addition, the chapter explores widely used neural network architectures for computer vision tasks, providing an overview of the specific models adopted in this dissertation. Chapter 3 discusses the principles behind neuromorphic sensor technology, covering both hardware and software perspectives. It also examines the most common event-based data representations and gives an overview of some of the event-based data applications in computer vision and robotics.

Part II focuses on the analysis and retrieval of time-to-transit (TTT) from sequences of RGB and grayscale images. Chapter 4 introduces the concept of TTT, its origins, and methods for obtaining it through both geometrical computation and image-based approaches, highlighting the challenges associated with the latter. Next, Chapter 5 explores an initial attempt to estimate TTT using a traditional computer vision technique, the Lucas-Kanade optical flow method. It then discusses efforts to refine this estimation to ensure its reliability as a control signal for robotic navigation. Building on this, Chapter 6 presents an enhanced methodology for TTT estimation, leveraging deep learning architectures to improve accuracy and robustness. A series of experiments are conducted, with both qualitative and quantitative results analyzed

in detail. Finally, real-world tests are performed using an actual robotic platform to validate the proposed approach.

Part III explores various approaches to leveraging event-based sensor visual information. Chapter 7 introduces PEDRo, an event-based dataset specifically designed for person detection in service robotics applications. Following this, the Memory in Motion (MeMo) pre-processing pipeline is presented in Chapter 9. MeMo organizes event representations to capture temporal dynamics without relying on complex architectures like recurrent neural networks. An enhanced version of this approach is then introduced in Chapter 10, featuring a more dynamic pre-processing pipeline and expanding its applications to object detection and classification beyond pupil detection. Finally, the part concludes with Chapter 11, where an exploration of integrating event-based sensors with spiking neural networks (SNNs) deployed on a microcontroller (MCU) is proposed. The advantages of the approach are presented, particularly in term of energy efficiency.

Part I

Fundamentals

Chapter 2

Computer Vision

Traditionally, Computer Vision (CV) has been defined, according to [81], as a combination of image processing methods, pattern recognition, and artificial intelligence focused on computer-based analysis of one or more images. These images can be captured by one or several sensors, either at a single moment or over a period of time. Classical approaches strongly rely on geometric modeling and complex mappings for pattern recognition and object detection. Various strategies can be applied, but the objective is always the same: to create a sophisticated network of elements that are able to analyze images or videos and identify either all objects or specific ones. In many ways, these systems attempt to mimic the behavior of the human eye by replicating how we perceive and interpret visual information.

However, with the advent and rapid development of neural networks and deep learning, the definition and capabilities of computer vision have expanded considerably. Today, computer vision can be seen as a subfield of Artificial Intelligence (AI) that enables computers and machines to derive key information from visual data such as images and videos. In other words, while AI allows machines to think, computer vision enables them to see, observe, and understand. Modern computer vision systems are trained using large datasets and neural networks, allowing them to learn complex visual patterns and perform tasks with high accuracy, sometimes achieving better performance than human capabilities. Recent breakthroughs in deep learning have pushed computer vision accuracy from around 50% to over 99% in less than a decade. This progress has transformed computer vision from a research field into a main part of modern intelligent systems.

The applications of computer vision span along multiple fields. In healthcare, it is of paramount importance in medical imaging and diagnostics [95]. In the automotive sector, it's essential for autonomous driving technologies [190]. Manufacturing uses it for quality control and defect detection [184], and energy, utilities, and logistics benefit from its monitoring and automation capabilities [52]. Robotics, particularly service robotics, leverage the CV for multiple tasks, including navigation [210], mapping [3], and place recognition [135], but also for object detection and recognition [182], and human-machine interaction [163]. Regardless of the specific applications, whether applied for classical image analysis or powered by modern AI, computer vision continues to redefine how machines understand and interact with the visual world.

2.1 Computer Vision: A Quick Historical Overview

The early foundations of computer vision were significantly influenced by the pioneering work of neurophysiologists David Hubel and Torsten Wiesel during the 1950s and 1960s [223, 90]. In their experiments, they presented arrays of images to cats and monkeys while recording neural activity. Their research revealed the presence of neurons that respond selectively to specific visual features, demonstrated hierarchical processing of visual information from simple to complex elements, and highlighted sensitivity to orientation. These findings represented the starting points for the development of multiple computer vision techniques, inspiring key methods such as edge detection, feature extraction, and hierarchical processing algorithms.

At the beginning of the 1960s, Lawrence G. Roberts launched the "Blockworld" project [167], aimed at deriving 3D representations from 2D images. His work underscored the importance of edge detection and 3D reconstruction by using visual cues and geometric hypotheses to recreate simple block-based scenes. This research became crucial for modern computer vision approaches that are capable of recognizing and interpreting complex objects and environments.

In 1966, Marvin Minsky examined the limitations of single-layer neural networks in the work "Perceptrons" [144] that he co-authored. He highlighted the difficulty of such models in handling complex, non-linear problems. This analysis motivates the development of more advanced neural architectures featuring multiple layers and

sophisticated training techniques, principles that now form the basis of deep learning in computer vision and AI.

By the late 1970s, Kunihiko Fukushima introduced the "Neocognitron" [63], a novel neural network architecture inspired by the human visual system. It was designed to extract local features and detect intricate patterns and edges. Additionally, the Neocognitron demonstrated the important capability of recognizing objects regardless of their position or orientation. This concept of spatial invariance became a foundational idea in the development of Convolutional Neural Networks (CNNs), which are now among the most widely used and effective architectures in the computer vision research area. In the early 2000s, research in computer vision primarily focused on object recognition. In 2001, the first real-time face recognition applications emerged, notably the system proposed by Viola and Jones [207].

With the growing interest in deep learning, particularly in its applications to computer vision, new datasets such as ImageNet [48] and COCO [126] became available. Alongside these datasets, new CNN-based architectures began to appear. One of the first was AlexNet [107], followed by more advanced models like VGG [181], ResNet [83], InceptionNet [191], EfficientNet [192], MobileNet [86], and YOLO [195]. In recent years, the introduction of Transformer-based architectures has marked a significant shift in the design of computer vision models. While CNNs dominated the field for years due to their effectiveness, attention mechanisms and Transformer architectures have started to gain traction, offering improved performance and accuracy across a wide range of tasks. These include object detection [236], optical flow estimation [178, 227], and object segmentation [118].

This chapter introduces the foundations of computer vision, beginning with its historical development, then reviewing a key task (optical flow estimation), and concluding with modern deep learning architectures widely used throughout this thesis.

2.2 Optical Flow Estimation: an example of a computer vision task

Optical Flow (OF) is defined as the pattern of apparent motion of image objects between two consecutive frames caused by movement. The concept of optical flow

was first introduced in 1950 by James Jerome Gibson [73] who provided many contributions to the field of visual perception. In the 1970s and 1980s, further developments were made by multiple researchers to estimate the optical flow field. Notable among these are the approaches proposed presented by Horn and Schunck [85], Lukas and Kanade [130], and Shi and Tomasi [177]. These classical methods rely on traditional computer vision techniques such as edge detection and feature tracking. However, with the rise of deep learning, optical flow estimation has seen significant progress. Modern neural network architectures have improved accuracy and robustness. Among the most recent and influential models are FlowFormer++ [178], GMFlow [226], and RAFT [194], which represent the cutting edge in optical flow estimation.

2.2.1 Optical Flow "Classical" Computation

Optical flow can be computed in two forms: Sparse OF which is obtained only for relevant features in the image and Dense OF which, on the contrary, is computed for all the pixels in the image. The difference is shown in Figure 2.1.

Fig. 2.1 An example of a Sparse OF field (on the left) and a Dense OF field (on the right).

The optical flow field is, by its nature, able to sense relative flow rates between frames but it can't be used to compute either the velocity of the observer or the distance observer-objects in the scene without adding more information like the exact size of those objects. Additionally, in a scenario where the observer is moving but there are also dynamic elements in the scene, the resulting field is a superposition of the flows of the observer and the object itself, as can be seen in Figure 2.2.

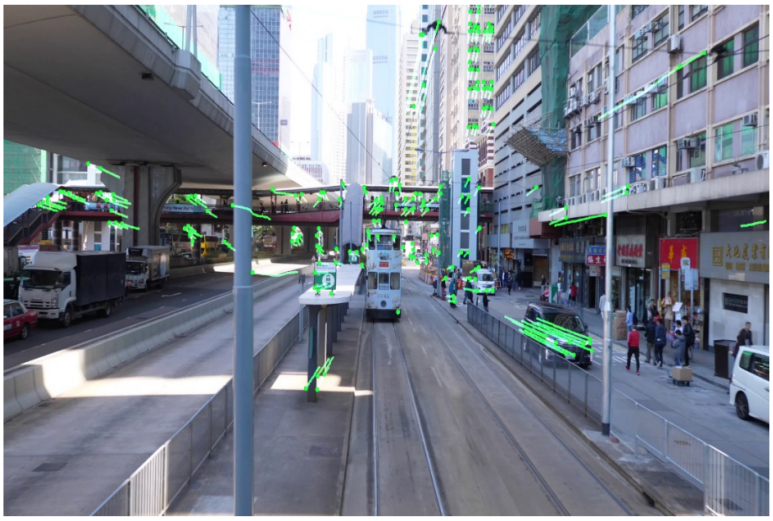


Fig. 2.2 The superposition of sparse OF fields when the camera is moving and there are dynamic objects in the scene.

As discussed at the beginning of this subsection, there are two main forms of optical flow and for each of them, there are strategies that can be used to obtain the desired optical flow field.

Dense Optical Flow: Horn-Schunck method

Horn and Schunck introduced a method for computing dense optical flow by assuming that images are continuously differentiable and that intensity changes smoothly over space and time. The proposed algorithm tries to minimize distortions in the flow, selecting the solution that presents a higher smoothness. The core assumption is the brightness constancy constraint, which states that the intensity of a pixel remains constant over time as it moves:

$$\frac{dI(x,y,t)}{dt} = 0 \tag{2.1}$$

Applying the chain rule to this expression gives:

$$\frac{\partial I(x,y,t)}{\partial x}\frac{dx}{dt} + \frac{\partial I(x,y,t)}{\partial y}\frac{dy}{dt} + \frac{\partial I(x,y,t)}{\partial t} = 0 \tag{2.2}$$

By assuming small motion constraints, the motion can be represented by its first-order approximation.

Horn and Schunck formulate the problem as a variational optimization, where the objective is to minimize the following energy functional:

$$J = \iint [(I_x v_x + I_y v_y + I_t)^2 + \alpha^2(||\nabla v_x||^2 + ||\nabla v_y||^2)]dxdy \tag{2.3}$$

The first term, where I_x, I_y, I_t are the derivatives of the image intensity values along the x, y, and t dimensions, enforces the brightness constancy constraint. The second term imposes smoothness on the flow field. α is a regularization term that controls the trade-off between these two first terms. ∇ is the Laplace operator encouraging spatial coherence in the flow, and v_x, v_y are the components of the optical flow vector. In practice, this equation represents a balance between preserving brightness constancy and ensuring smoothness across the image. Although this method provides a dense optical flow field with velocity vectors for each pixel in the image, it is sensitive to noise and computationally expensive, making it not ideal for real-time applications.

Sparse Optical Flow: Lucas-Kanade method

Bruce D. Lucas and Takeo Kanade proposed a method to estimate a sparse optical flow field by focusing only on relevant features within an image. These features are typically identified using a feature detector such as the Harris Corner Detector [82], SIFT [129], or FAST [170], and then described with a feature descriptor, such as BRIEF [33] or ORB [171], to facilitate reliable tracking.

The Lucas-Kanade approach estimates optical flow by optimizing an energy function at specific pixels, using predetermined feature points. It assumes that feature displacements between two consecutive frames are small and constant within a local neighborhood of size n around a given point p. Using the standard optical flow constraint equation, as in 2.2, the method expresses the constraint for each pixel p_i in the neighborhood as:

$$\frac{\partial I}{\partial x}(p_i)v_x + \frac{\partial I}{\partial y}(p_i)v_y + \frac{\partial I}{\partial t}(p_i) = 0, \quad where \quad i = 1, ..., n \tag{2.4}$$

To solve for the optical flow vector components, a least-squares optimization can be set up. This leads to the matrix formulation:

$$v = A^+ b \tag{2.5}$$

where A^+ is the Moore-Penrose inverse of matrix A, and:

$$A = \begin{bmatrix} \frac{\partial I}{\partial x}(p_1) & \frac{\partial I}{\partial y}(p_1) \\ \vdots & \vdots \\ \frac{\partial I}{\partial x}(p_n) & \frac{\partial I}{\partial y}(p_n) \end{bmatrix}, \quad v = \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \quad b = \begin{bmatrix} \frac{\partial I}{\partial t}(p_1) \\ \vdots \\ \frac{\partial I}{\partial t}(p_n) \end{bmatrix} \tag{2.6}$$

A drawback of the original Lucas-Kanade method is that it fails when a large pixel motion occurs. To address this, later approaches introduced a pyramidal implementation. In this multi-scale strategy, the image is processed at progressively lower resolutions through an image pyramid. Tracking begins at the top of the pyramid, where the resolution is lower and pixel motions appear smaller, making it easier to estimate motion. The results are then propagated down through successive pyramid levels, refining the estimates at higher resolutions until convergence.

2.2.2 Optical Flow "Neural" Computation

The advent of deep learning influenced the way in which the optical flow is computed. In recent years, numerous algorithms have been proposed to estimate a dense optical flow field, ranging from convolutional [80] to transformer-based structures [178, 226]. In the following, I'm going to explain a well-known method briefly. Although it is not among the most recent, it remains highly influential, serving both as inspiration for newer models, like FlowDiffuser [131], and as a benchmark for evaluation [227].

RAFT: Recurrent All-Pairs Field Transforms for Optical Flow

RAFT is a deep neural network designed to estimate dense optical flow between a pair of sequential images. It predicts a displacement field that maps each pixel in the first image to its corresponding location in the second image. The algorithm follows three main steps: feature extraction from both images, computation of pixel-wise correlations to capture visual similarity, and iterative refinement of the flow field using a mechanism inspired by optimization algorithms.

Figure 2.3 illustrates the overall architecture, which consists of three key components:

- *Feature Encoder Block*: It is designed to perform feature extraction on the two input images using a convolutional neural network composed of six residual

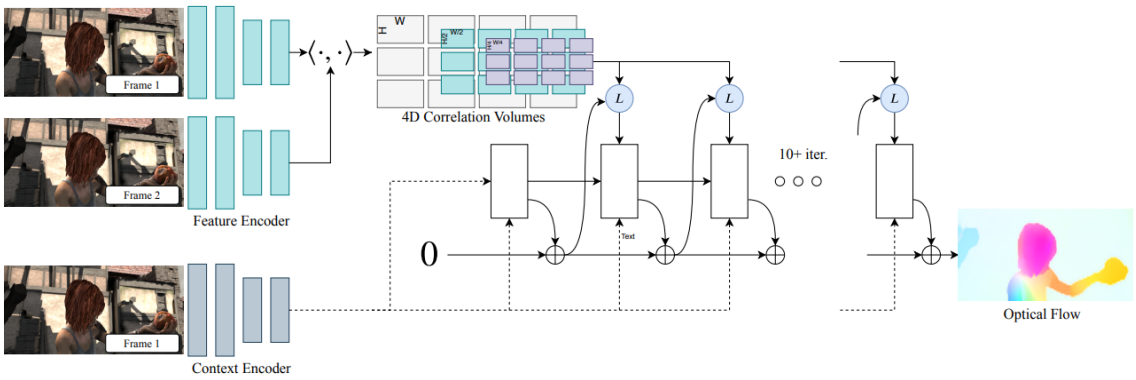


Fig. 2.3 General overview of the RAFT architecture, which extracts features, computes all-pairs correlations, and iteratively refines optical flow estimates. Picture from [194]

blocks. The output is a downsampled feature map with a resolution reduced to $1/8$ of the original image and a dimensionality D. As illustrated in Figure 2.3, there are two feature encoders that operates on both input images and share the weights. Additionally, a context encoder considers only the first input image and extracts feature that serve as a primary reference for the flow estimation. The context encoder outputs $C = c + h$ feature maps, which are the sum of the context feature maps c and the hidden features map h. These last h maps are used to initialize the hidden state in the iterative update block.

- *Visual Similarity Block*: Visual similarity is computed as the inner product of all pairs of feature maps, resulting in 4D correlation volumes that encapsulate key information about small and large pixel displacements. The all-pairs correlation of two feature maps are calculated without any fixed-sized window. To enhance this further, a 4-layer correlation pyramid is constructed by progressively pooling the last two dimensions of the 4D tensor with kernels of different sizes. This pyramid captures multi-scale similarity features, making the abrupt movements more noticeable. It also provides information of large and small displacements.

- *Iterative Update Block*: It estimates a sequence of optical flow fields starting from an initial point f_0, which can be all 0's or the forward projected previous flow estimation. At each iteration k, the block produces a flow update direction Δf to be added to the current flow estimate. This iterative process mimics an optimization algorithm, and the model is trained to produce updates that guide the flow sequence toward convergence at a fixed point. The block is composed

of Gated Recurrent Unit (GRU) [42] cells. Since the network outputs optical flow at a lower resolution than the original image, an upsampling strategy is required. The authors found that using a convex upsampling module yields the most accurate final optical flow estimates.

The loss function is defined as a simple L1 distance between the predicted optical flow and the ground truth. The total loss is computed as the sum of the L1 losses across all recurrent block outputs, comparing each upsampled prediction to the ground truth. The results on very popular datasets, such as the Sintel [31] and the KITTI [70], show performance outperforming other existing methods.

2.3 Computer Vision and Deep Learning: Neural Networks

Nowadays, Machine Learning and Deep Learning represent the core of almost all Computer Vision tasks. The development of neural architectures such as Convolutional Neural Networks (CNNs), and more recently, Transformer-based models, has greatly enhanced the ability of CV systems to automatically learn, extract, and process meaningful features from visual data. These advances have led to significant progress across a wide range of applications in the field. In the following sections, we will provide an overview of some of the most prominent and widely adopted neural architectures in computer vision research field, focusing on the architecture components and their working principles. This overview serves as a foundation, as these models will be employed as estimators in the subsequent parts of this dissertation.

2.3.1 ResNet: Residual Network

ResNet [83] is a neural network architecture introduced in 2015 by researchers at Microsoft Research. It was designed to address the vanishing and exploding gradient problems commonly encountered when training very deep neural networks. The key innovation in ResNet is the use of Residual Blocks, an example of which is illustrated in Figure 2.4. The most important feature of these blocks is the presence of a skip connection, which bypasses one or more layers in the network. This skip connection is fundamental to residual learning.

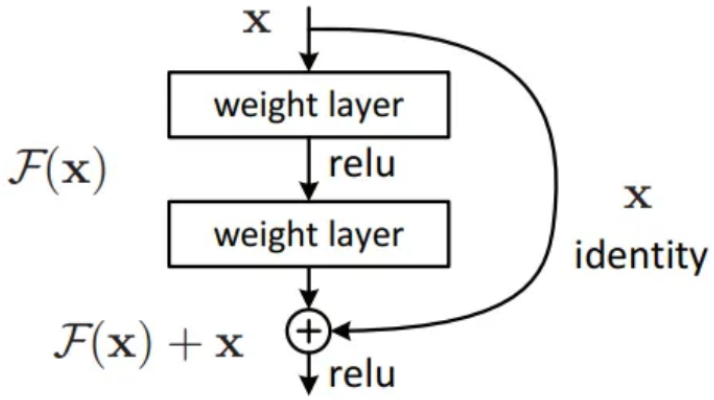


Fig. 2.4 Components of a Residual Block in the ResNet Neural Network.

In a traditional feedforward network, the output of a layer is typically represented as $H(x) = f(x)$, where $f(x)$ is a transformation of the input x involving weights, biases, and a non-linear activation function. However, in ResNet, the skip connection modifies this to $H(x) = f(x) + x$. This addition allows the network to learn residual functions with respect to the input, rather than trying to learn the full transformation from scratch. Moreover, these skip connections provide two major benefits: they mitigate the vanishing gradient problem by offering an alternative path for gradients to flow during backpropagation, and they enable the network to learn identity mappings more easily, ensuring that deeper layers perform at least as well as shallower ones.

Notably, it is important to mention that in networks that are composed on convolutional layers and pooling layers, there could be a problem of matching dimensions between the input and output of these layers. This could represent a challenge when applying the skip connections, therefore two strategies can be used: Zero-padding, where the input is padded with zeros to match the dimensions of $f(x)$, or Projection shortcut, where a 1×1 convolution is applied to the input to match the output dimensions.

Multiple ResNet architectures are available in the literature, depending on the number of layers they are composed of. Among the most popular we can mention ResNet-18, ResNet-50, and ResNet-101.

2.3.2 MobileNet

The first MobileNet network was developed by a team of researchers at Google in 2017 [87]. The objective was to create a CNN able to run on mobile and embedded devices, meaning that it must be smaller, efficient, but also comparable in terms of performance with others existing models.

The key contribution introduced with MobileNet is the concept of *Depthwise Separable Convolutions*. This technique basically splits the traditional convolution operation into two separate steps: a depthwise convolution where filters are applied to each input channel separately, and a pointwise convolution, which uses a 1×1 convolution to combine the outputs from the previous step. This approach reduces the number of parameters and computations, making the model more suitable for resource-constrained environments. A visual representation of the working principle and the structure of the depthwise and pointwise convolutions are shown in Figure 2.5.

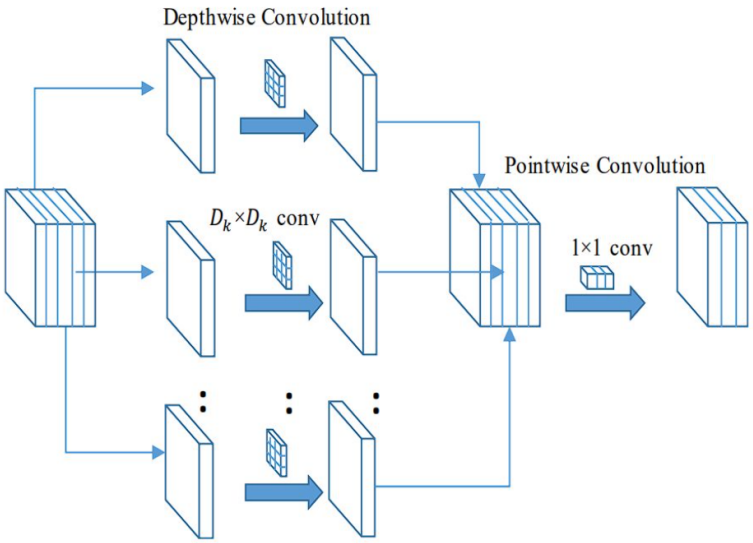


Fig. 2.5 Depthwise and Pointwise convolutions structures and working principle.

Following the original version, MobileNetV2 introduced *Inverted Residual Blocks* with bottlenecking. These blocks use linear bottlenecks between layers to reduce the number of channels processed and further enhance efficiency. Additionally, these blocks also add short connections between bottlenecks to improve

gradient flow, and substitute the last layer with a linear activation instead of the classical non-linear ReLU function.

In 2019, the last version [86] of the MobileNet family was released. It combines the best of both previous versions and introduces some improvements to further enhance the old architectures:

- Platform-aware Neural Architecture Search, which is an automated search technique to find the best configuration design for mobile platforms.
- Squeeze-and-Excite (SE) modules, which analyze the feature maps produced by the convolutional layers and highlight the most important features.
- Hard-Swish activation function, which offers a less complex alternative to ReLU that balances between accuracy and computational efficiency.

2.3.3 MobileViT

MobileViT [137], introduced in 2022 by researchers from Apple, is a lightweight and general-purpose vision transformer designed for mobile vision tasks. It combines the strengths of traditional convolutional neural networks (CNNs), such as spatial inductive biases and robustness to data augmentation, with the advantages of Vision Transformers (ViTs) [54], including adaptive input processing and global context modeling. The core idea behind MobileViT is to learn global representations using transformers while preserving convolutional characteristics. By treating transformers as convolutions, the network can incorporate spatial biases and learn effective image representations with fewer parameters and simpler training procedures. An overview of the general architecture is shown in Figure 2.6. It is composed of two main block type:

- *MobileNetV2 (MV2) Block*: This is an inverted residual block introduced in the second version of the MobileNet architecture. It consists of two 1×1 pointwise convolutions with a depthwise convolution layer in between. A residual connection is also added between the input and the output of the block to facilitate better gradient flow and improve performance.
- *MobileViT Block*: The purpose of this block is to effectively capture both local and global information while keeping the number of parameters low. The

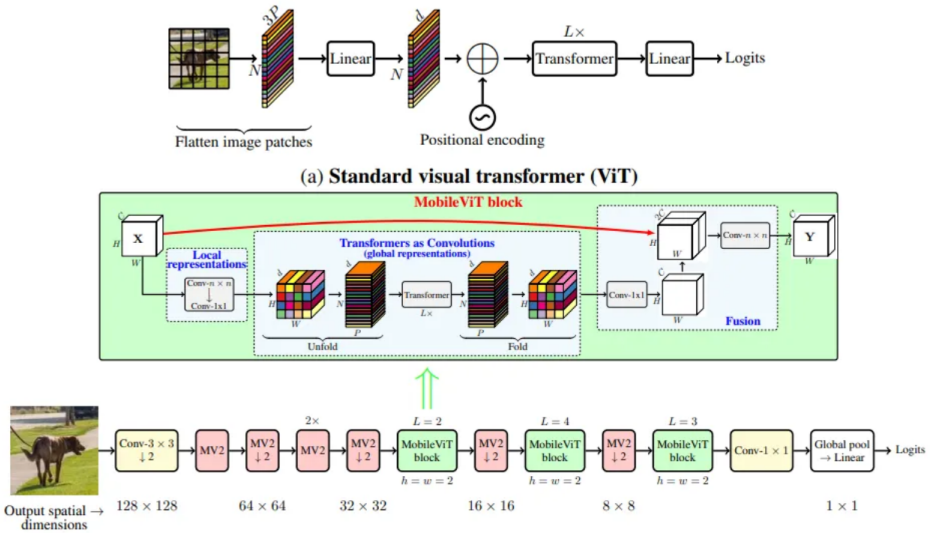


Fig. 2.6 General overview of MobileViT architecture. It is characterized by the presence of both CNN architectures and attention modules. Picture from [137]

block begins by applying an initial $n \times n$ convolution, followed by a 1×1 convolution that projects the features into a higher-dimensional space. The resulting feature map is then divided into N non-overlapping flattened patches, which are passed to a transformer module. The transformer learns global representations with built-in spatial inductive biases, producing an output of shape $P \times N \times d$, where P is the flattened size of each patch (height × width), and d is the feature dimension. To merge the transformer's output with the original input tensor, the output is reshaped back to a tensor of size $H \times W \times d$, where H and W are the height and width of the original input. This transformed output is passed through a 1×1 convolution and then concatenated with the input tensor. Finally, a $n \times n$ convolution is applied to fuse the concatenated features effectively.

MobileVit has been presented in multiple variations, depending on the size of the model, i.e., small, extra small or extra extra small.

2.3.4 GMDepth

GMDepth was presented in [227], where the authors proposed a unified transformer-based model that estimates optical-flow, depth, and stereo matching. We will focus

just on the depth estimation part, but it is important to mention that for performing the three vision tasks, the authors developed a unique model that features just a single task-dependent block. An overview of the architecture is illustrated in Figure 2.7.

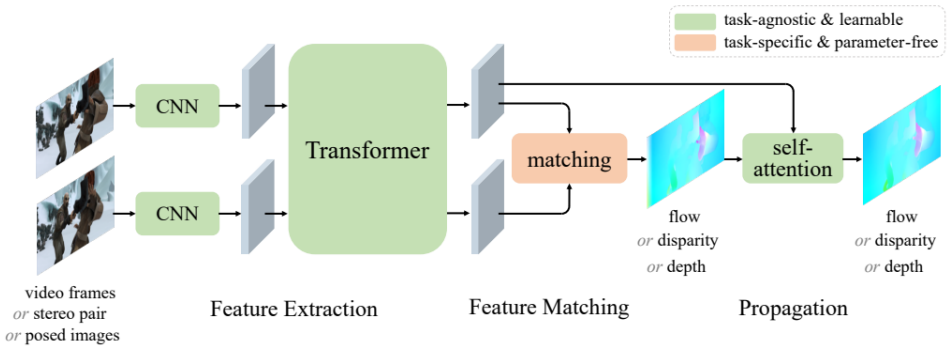


Fig. 2.7 Overview of the architecture presented in [227]. Picture from [227].

Starting by considering two images as input, I_1 and I_2, the fundamental steps in the GMDepth methodology are the following:

Feature Extraction. In this step, high-quality discriminative features are extracted using a combination of CNNs with Transformers. Specifically, a weight-sharing ResNet architecture is employed to extract feature maps from each of the two input tensors independently, with the spatial resolution reduced by a factor of eight relative to the input. Then, to capture relationships between the two sets of features and integrate knowledge from the potential matching candidates in another image, a cross-attention mechanism is introduced. In addition to that, a self-attention layer is applied to further improve the feature's quality by incorporating a broader contextual understanding than what convolutional layers alone can provide. This is followed by a two-layer feed-forward network (FFN), in line with the original Transformer architecture [205], to enhance even more the network's representational capacity. All these blocks together, the self-attention, cross-attention and FFN, form a Transformer block. The final model counts six of these blocks stacked, which refine the feature representations and gradually improve the performance.

Depth Matching. This is the only step that is task-specific and does not require any learnable parameters. For depth estimation, we assume that the camera intrinsics and extrinsics are known. The proposed method is inspired by the plane-sweep stereo technique [43]. It begins by discretizing the depth range into a set of candidate depth

values. For each candidate depth, the authors compute the projection of every pixel in image I_1 into image I_2, using the known camera parameters. This results in a grid of 2D coordinates corresponding to where each pixel the first image would appear in the second one at that specific depth. Using these coordinates, features from I_2 are sampled via bilinear interpolation. Next, for each depth candidate, the authors compute a similarity map by taking the dot product between the features of I_1 and the sampled features from I_2. These similarity maps are then stacked along the depth dimension to form a 3D cost volume: $C_{depth} = [C^1, C^2, ..., C^N]$, where N is the number of discretized depth points. To estimate the final depth, the cost volume is normalized along the depth dimension to produce a matching distribution M_{depth}. The depth at each pixel is then obtained by computing the weighted average of the depth candidates, using M_{depth} as weights.

Prediction Propagation. This step is fundamental to overcoming the problem of occluded or out-of-boundary pixels, which violate the assumption that corresponding pixels are always visible in both input images. Observing that the depth field in the image itself shares high structure similarity, the authors propose to propagate high-quality predictions to unmatched regions by leveraging feature self-similarity. This can be easily implemented through one self-attention layer.

Refinement. This is a step introduced by the authors to further improved the performances of the model. The addition also leads to an increase in the computational speed of the process; thus it is important to find a trade-off between accuracy and speed. The paper proposes two different refinement strategies: hierarchical matching refinement, which is task-agnostic, and local regression refinement, which, on the contrary, is task-specific. Since this step is not essential for understanding the working principle of the entire system, for a more detailed explanation of these refinement methodologies, please refer to the original paper [227].

Training Loss. The training loss is defined to supervise all predictions (N is the number of predictions), including intermediate network outputs and final ones, with the ground truth:

$$L = \sum_{i=1}^{N} \gamma^{N-i} \ell(V_i, V_{gt}) \tag{2.7}$$

Here, γ is a weight that increases exponentially to give higher importance to later predictions. For depth estimation, the loss function ℓ is defined as the L_1 loss on the

inverse depth [201]. Additionally, a gradient-based loss term is incorporated into the overall loss.

2.3.5 YOLO: You Only Look Once

YOLO (You Only Look Once) is a widely used object detection neural network, first introduced by Redmon et al. in 2015 [166]. It formulates object detection as a single regression problem, directly predicting bounding boxes and class probabilities from an input image in a single forward pass. This design enables YOLO to achieve real-time performance without sacrificing accuracy.

The core idea behind YOLO is to divide the input image into a grid of dimensions $S \times S$. Each cell in this grid is responsible for predicting a fixed number of bounding boxes, along with their confidence scores and class probabilities, for any object whose center falls within the cell. Figure 2.8 illustrates this grid-based prediction strategy.

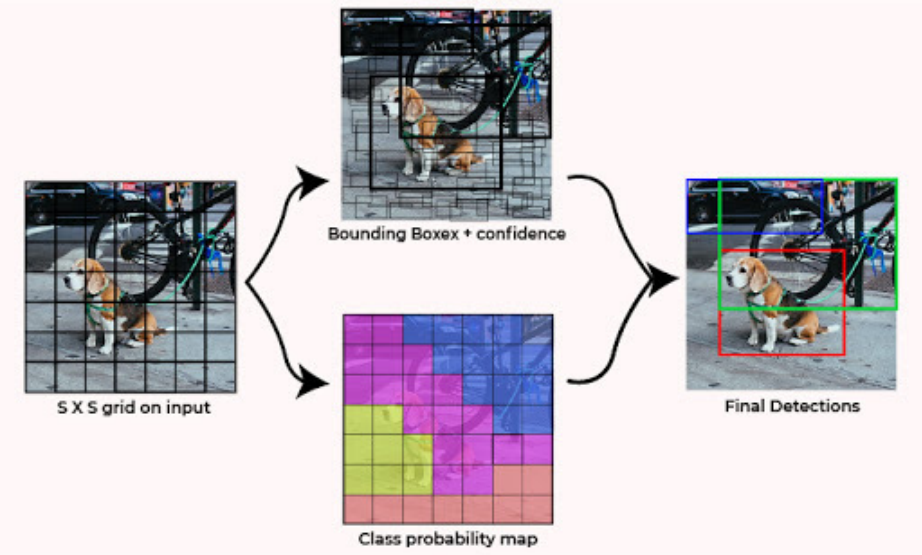


Fig. 2.8 Visual representation of the grid-based prediction strategy of YOLO.

The base architecture of YOLO consists of several convolutional layers for feature extraction, followed by fully connected layers that produce the final detections. The framework offers several advantages: it can be trained end-to-end on detection data, it learns global contextual information about the image, and it generates fewer false positives in background regions compared to region-based methods. This

combination results in highly accurate, real-time performance. However, the first version of YOLO also has some limitations. In particular, it struggles to detect multiple small or closely spaced objects because each grid cell can predict only a limited number of bounding boxes. These issues have been progressively addressed in subsequent versions of YOLO through a variety of improvements, including:

- Anchor Boxes, which improve bounding box priors and allow the model to better handle objects of varying scales.
- Batch Normalization and higher-resolution input images, which enhance detection accuracy.
- Feature Pyramid Networks, which improve the detection of small objects.

Today, YOLO remains one of the most widely adopted neural networks for object detection and even tracking, in real-time applications. The architecture continues to evolve, with continuous improvements in both structure and performance. The latest released version, YOLO v12 [196], incorporates attention-based mechanisms to further enhance detection accuracy and robustness.

2.3.6 DETR

DETR (DEtection TRansformer) is a transformer-based neural network introduced by Carion *et al.* [38] for the object detection task. DETR formulates the object detection problem as a direct set prediction, meaning that it predicts a fixed-size set of bounding boxes and their associated class labels in a single, end-to-end pass. The architecture

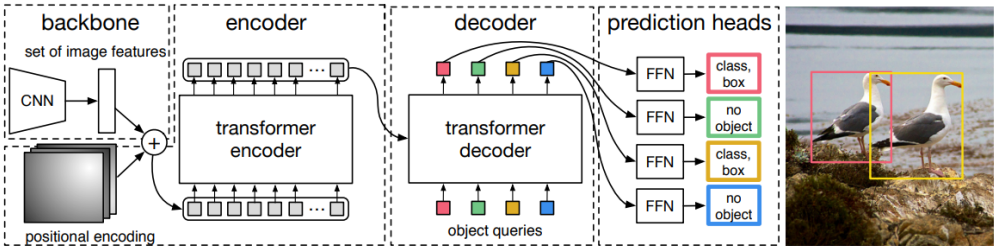


Fig. 2.9 General Overview of DETR architecture. Picture from [38].

of the network is presented in Figure 2.9, and it consists of a convolutional backbone,

usually a ResNet, for extracting image features, followed by a transform encoder-decoder module. The encoder part processes the flattened feature map to capture global relationships between image regions, while the decoder takes a fixed set of learned object queries and reasons about which regions correspond to objects. Each query estimates a prediction containing the class label and the bounding box parameters. In order to associate each ground truth object with only one prediction, a Hungarian algorithm is employed during the training phase. This allows the model to learn object localization and classification in a unified framework.

Compared to other approaches, DETR presents several advantages, such as:

- No need for anchors, region proposals or post-processing mechanisms to discard predictions.
- Through the usage of transformer modules, the framework is able to capture information about the global context and the relationships between objects.

However, DETR also presents some limitations; indeed, it struggles in detecting small objects due to the coarse resolution of the feature maps, and most notably, it requires long training times and a lot of data to converge. Recent improvements address these limitations, in the variants named Deformable DETR [243] or Conditional DETR [138], the introduction of the **deformable attention** allows the model to focus only on relevant regions, improving efficiency and small object detections. Moreover, the **multi-scale feature processing** and the **improved training strategies** enable finer-grained localization and accelerate convergence.

Nowadays, the DETR family has become an important benchmark in object detection frameworks, combining the power of transformers with the convolutional neural architectures.

Chapter 3

Event-based Vision

In recent years, event-based vision has gained more and more attention in the field of visual sensing technology. It takes inspiration from the biological working principle of the human eye, particularly the retina [158], and proposes a new methodology for capturing visual information. Unlike traditional frame-based imaging systems that capture whole scenes at fixed time rates, event-based vision systems focus on the dynamic elements of a visual scene, offering a novel and efficient alternative for acquiring motion and change.

The core elements of this approach are event-based sensors, also known as event cameras or neuromorphic sensors. These advanced sensors have pixels that operate asynchronously and independently. Instead of continuously recording frames, pixels generate an array of discrete data points, called "events", only when changes in light intensity are detected. Each event is composed of four critical information: the time of occurrence, the pixel's location, and the polarity of the brightness change ($+1$ for positive and 0 or -1 for negative variations). This continuous and asynchronous stream of data is acquired with a time resolution in the order of microseconds, enabling systems to respond rapidly to changes. Because event cameras are sensitive only to motion or variations in brightness, they avoid capturing redundant information, making them highly efficient and suitable for environments where fast decision-making and low latency are essential.

The multiple advantages of event-based vision are unlocking new possibilities in computer vision [64, 4, 239]. Applications span a variety of fields, including robotics [2, 233, 148, 149], autonomous vehicles [176], surveillance [61, 1], and augmented

reality [91, 216], where traditional frame-based methods may be not enough. In the research community, event-based vision is increasingly recognized as a promising solution to challenging visual processing issues, offering faster, more efficient, and biologically inspired ways to perceive the world.

3.1 Event-based Sensors: a historical overview

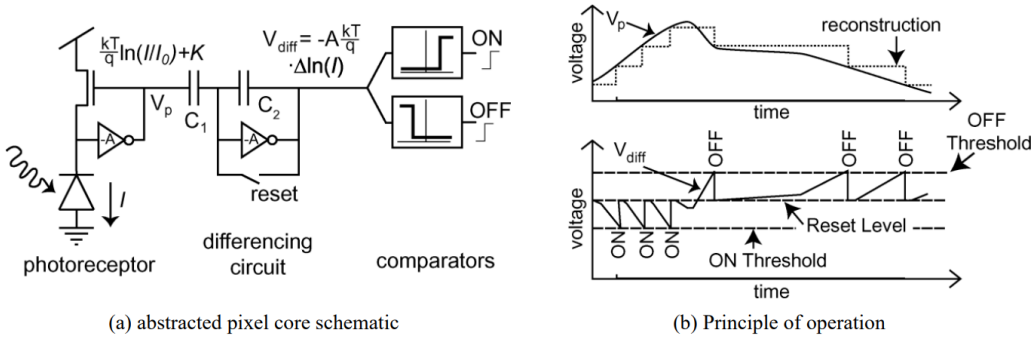
Initially, the research in the domain of event-based vision mainly focused on creating and validating biologically inspired models, with little attention to possible real-world applications. However, as the drawbacks of the well-known frame-based sensors become more evident, the application of neuromorphic imaging sensors in practical perception problems has emerged as a concrete option.

The advancements in neuromorphic engineering, a field that tries to replicate neural systems' structure and functionality, constitute a first step for developing event-based sensors [127]. The first commercially available event-based sensor was made in 2008 and it is called the Dynamic Vision Sensor (DVS) 128 [125]. The DVS-128 has a resolution of 128×128 pixels and a dynamic range of 120 dB. After that, other versions of neuromorphic sensors were invented. In 2010, the Asynchronous Time-Based Image Sensor (ATIS) was introduced [157], and four years later, Brandli *et al.* presented the Dynamic and Active-Pixel Vision Sensor (DAVIS) [30]. Since these initial prototypes, the improvements in the development of event-based sensors have advanced rapidly, mainly focusing on augmenting the pixel resolution, the dynamic range, the latency, and the energy efficiency [124, 125, 157, 30, 189].

The evolution of event-based cameras highlights the technological progress in the field but also emphasizes the fact that these vision sensors are recognized as possible objects to revolutionize how machines, like robotic platforms, perceive and interact with the environment.

3.2 Working Principle: Technical Foundations

The key principle of event-based neuromorphic sensing is the ability to capture only the dynamic changes in the environment. Each pixel operates independently and asynchronously and reacts individually to light changes. When the light intensity



(a) abstracted pixel core schematic (b) Principle of operation

Fig. 3.1 Circuit of a single DVS-128 pixel and its operating principle. Picture from [125]

change overcomes a certain threshold, the pixel generates an event that encodes the time of occurrence, the position of the pixel, and the polarity of the variation (whether the light intensity increased or decreased).

Functional Component of a DVS Pixel

The general DVS design and the components of a DVS pixel are shown in Figure 3.1. The main components are three, and they are the following:

- *The Photoreceptor*: it is responsible for converting the incident light into an electrical signal V_p. The response is logarithmic, and it automatically controls the pixel's gain while responding quickly to changes in illumination. This design introduces a DC mismatch between pixels, which requires calibration for direct use of the output.
- *The Differencing Circuit*: it is a circuit connected to the photoreceptor and it amplifies changes in the signal with high precision. It plays a crucial role in removing the DC mismatch described before by balancing the output to a reset level after an event is generated, as presented in Figure 3.1(b). The gain of this change amplification is determined by the ratio of the capacitors $\frac{C_1}{C_2}$ and it helps reduce the impact of comparator mismatch.
- *The Comparators*: they compare the output V_{diff} of the differencing circuit against a threshold level to generate events. The comparators are fundamental for converting the analog input signal into a digital output - ON or OFF events - defining the change in the scene illumination.

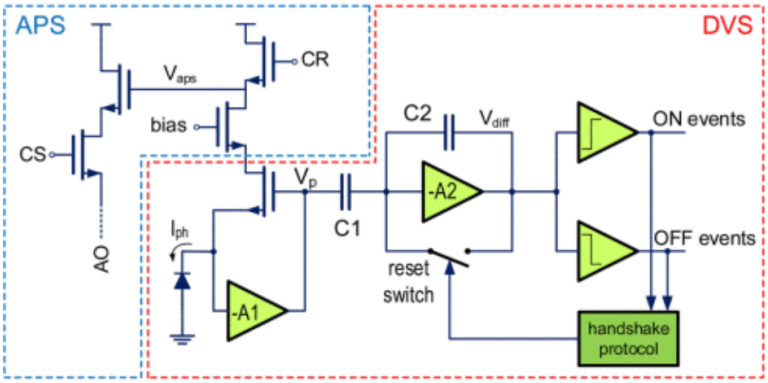


Fig. 3.2 Simplified circuit of a single DAVIS pixel. Picture from [158]

All these components make the DVS pixel sensitive to temporal contrast rather than absolute light levels, enabling it to detect and respond to dynamic changes in the scene efficiently.

Combining DVS with Active Pixel Sensor (APS)

The DVS working principle described in the previous paragraph works effectively in high dynamic and high-speed scenes but it lacks in static applications, where it is also important to provide information on the static light intensity. The ATIS version of the neuromorphic vision sensor has been developed to address this issue since it added an APS circuitry to capture static light intensity. A major drawback of this approach, however, is represented by the possibility of introducing motion artifacts [32] due to the ATIS's dependency on the DVS to trigger light-intensity updates which result in asynchronous and not uniform exposure times.

Another approach to solving the DVS limitations is represented by the development of the DAVIS. As shown in Figure 3.2, these sensors combine the DVS with the APS at the pixel level, sharing the same photodiode. In particular, the DVS operates as described before, while the APS functions independently by continuously measuring and integrating the photocurrent over time to produce a voltage signal representing the light intensity. This process employs differential double sampling during the APS readout phase to mitigate fixed pattern noise. Differential double sampling consists of capturing two voltage readings from each pixel, the reset voltage and the signal voltage. The first one is taken immediately after resetting the pixel while the latter is sampled at the end of the exposure period. The difference between these two values is proportional to the amount of light that has struck the pixel during

the exposure time, allowing the system to calculate the light intensity. The resulting analog signal is then converted to a digital value to construct an image.

The integration of DVS and APS in a shared architecture enables the DAVIS sensor to capture simultaneously event-based data and standard image frames. Both components use the same photodiode with the APS just requiring the addition of a small readout circuit that slightly increases the pixel area. This dual-output capability makes the DAVIS suitable for a wide range of computer vision tasks by combining high-speed dynamic sensing with traditional frame-based imaging. In the initial design, the DAVIS utilizes a rolling shutter, which results in motion artifacts during the capture of fast-moving scenes. A later improvement was performed to solve this issue, a global shutter was incorporated into the DAVIS, taking inspiration from commonly used CMOS-based imaging sensors. This solution effectively minimizes motion artifacts, especially compared to earlier DAVIS designs or ATIS sensors. Moreover, unlike ATIS, the DAVIS provides a simultaneous output of asynchronous events and traditional frames in both rolling and global shutter modes, allowing the sensor to accurately capture both dynamic and static elements in the scene. In summary, the DAVIS architecture expands the range of applications for the event-based imaging technology.

3.3 Advantages and Key Characteristics

Event-based sensors have many unique characteristics that are not present in the standard frame-based vision sensors. In particular:

- **High Temporal Resolution**: Neuromorphic vision sensors work with a high temporal resolution. In theory, each event is acquired every microsecond (1 µs), but more practically the value is around 100 µs. This high temporal resolution effectively eliminates motion blur issues, usually present in traditional cameras due to prolonged exposure times.
- **High Dynamic Range (HDR)**: The dynamic range of event-based cameras is approximatly 120 dB. The logarithmic response of photoreceptors to the varying light intensities allows the compression of a wide range of light levels. As a result, the event detection is performed also in very dark and very bright

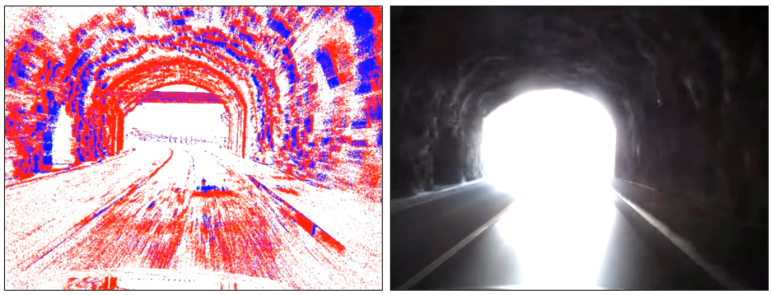


Fig. 3.3 The high dynamic range of event-based cameras enables detection also in very bright or very dark situations, such as a car exiting a tunnel. On the left is the output of a Prophesee Gen 3.1 event-based camera, while on the right is the output of a traditional frame-based camera. Picture from the DSEC dataset [68].

areas, something that standard frame-based sensors struggle with, as shown in Figure 3.3, due to the limited dynamic range (around 60 dB).

- **Low Latency**: This advantage is a consequence of the high temporal resolution. It is the result of fast photoreceptor circuits, efficient processing, independent pixel operation, and optimized data transmission protocols.
- **Low Power Consumption**: The power consumption of event cameras is typically in the milliwatt range and it depends on the DVS pixels' activity. This power efficiency is strongly related to the fact that the event-based sensors generate data only when dynamic objects are present in the scene, so no redundant information is provided.

3.4 Events representation

Event-based cameras generate as output a continuous stream of events, where each event can be expressed as $e = (t, x, y, p)$. Here t is the timestamp of the event, (x, y) is the pixel position that registers the event, and p is the event's polarity. The polarity is determined based on changes in the logarithmic intensity L at the pixel, and it is defined as:

$$p = \begin{cases} +1, & L(x,y,t) - L(x,y,t-\Delta t) \geq C \\ -1, & L(x,y,t) - L(x,y,t-\Delta t) \leq -C \\ 0, & other \end{cases} \tag{3.1}$$

where C is a threshold, and Δt is the time interval between two events at the same pixel position. The output stream of events can be expressed as:

$$\mathcal{E} = \{e_i\}_{i=1}^{N} = \{x_i, y_i, t_i, p_i\}, \quad i \in N \tag{3.2}$$

While this event-driven representation offers several advantages, it is different from the frame-based data that most of the conventional deep neural networks are used to process. In recent years, researchers have been focusing on taking the raw events stream provided by the camera and exploiting its advantages by feeding architectures that process it directly, such as spiking neural networks or recurrent architectures. However, a popular approach to bridge the gap between event-based sensing and conventional deep learning is to convert the raw stream of events into representations that are "understandable" by state-of-the-art deep neural networks. This transformation enables the use of standard neural network architectures, allowing researchers to benefit from event-based cameras without the need for a complete rethink of existing models or methodologies.

In the following, a detailed explanation of the different representations is provided, focusing on the ones that are useful for the scope of this work. They can be summarized in four categories: image-based, surface-based, voxel-based, and graph-based [64, 239].

3.4.1 Image-based Representation

The simplest solution to use the event stream with existing DL approaches is to convert it into a 2D image-like structure. The frame-like structures are usually multi-channel frames and aggregate information including timestamps, polarities, spatial coordinates, and events counts [134, 241, 51, 9, 49]. Considering the event's information used to obtain the representation, we can categorize:

- **Stack based on polarity**: In 2018, Maqueda *et al.* [134] proposed a 2-channel representation where they set up two separate channels to evaluate the histograms for positive and negative events.
- **Stack based on timestamps**: It considers the importance of events timestamps but also events count. In [241], Zhu *et al.* presented a four-channel representa-

tion where the first two channels are event-counts channels while the others contained the timestamps of the most recent event based on the polarity.

- **Stack based on the number of events**: It stacks the events in a fixed constant number [88, 139].
- **Stack based on timestamp and polarity**: In EV-gait [214], the authors converted the events into a four-channel frame-like representation, containing the positive or negative polarities in two channels and the temporal characteristics in the other two channels.

3.4.2 Surface-based Representation

A Time Surface (TS) is a 2D map where each pixel stores a single value, such as the timestamp of the last event at that location [47, 109]. This surface captures the motion history and the temporal dynamics of a scene, updating asynchronously as new events are received. It is commonly computed over a specific time interval. TSs emphasize recent events over older ones through the application of an exponential temporal kernel. However, there is also another type of representation called Surface of Active events (SAE) that simply assigns values based on the time difference between the most recent t_i and the first event t_0 at a given pixel location, considering the specified time interval T:

$$SAE(x_i, y_i) = \frac{t_i - t_0}{T} \tag{3.3}$$

To preserve invariance to motion speed, various normalization techniques have been proposed for these surface representations [5, 132]. Additionally, to make TSs less sensitive to noise, each pixel value may be computed by filtering the events in a space-time window [183].

One drawback of these types of representation is related to the fact that TS or SAE highly compress the information since they consider only the most recent event for each pixel location. This leads to a degradation in their effectiveness when the scene is particularly textured and the pixels spike frequently.

3.4.3 Voxel-based Representation

The voxel-grid is a 3D space-time histogram that represents event data, where each voxel corresponds to a specific pixel location and a defined time interval. This structure maps raw events onto a discrete temporal grid by assigning them to temporal bins, preserving temporal information more effectively than traditional 2D projections.

The first voxel-grid representation was introduced in [245], where events were inserted into volumes using a linearly weighted accumulation strategy to enhance temporal resolution. Subsequent works have built upon this foundation [229, 165]. More recently, the Time-Ordered Recent Event (TORE) volume was proposed in [12], where the main goal is to compactly maintain raw spike temporal information with minimal information loss.

3.4.4 Graph-based Representation

The graph-based representation converts raw event data within a specified time window into a set of connected nodes, aiming to preserve the inherent sparsity of event streams. In [18, 17], Bi *et al.* introduced a residual graph convolutional neural network to derive compact graph representations for object detection. Two years later, Deng *et al.* proposed EV-VGCNN [50], a lightweight voxel-based graph convolutional network designed to leverage the sparsity of event data for classification tasks.

3.5 Event-based examples in Computer Vision

Event-based vision has gained more and more space in the deep learning research field, particularly for tackling a wide range of computer vision tasks, such as surveillance and monitoring [61, 162], object detection [34, 155], tracking [145], obstacle avoidance [58, 209], deblurring videos [101], place recognition [60, 111] and advanced Simultaneous Localization and Mapping (SLAM) for robotics [77]. The reasons are multiple, indeed the unique capabilities of these sensors are well-suited to address complex challenges including high variations in illumination, extreme dynamic scenarios, and applications where the hardware requires energy constraints.

Object Classification

Object classification is a fundamental task in computer vision, aiming to categorize objects based on their visual features. When performed using event-based sensors, the classification can benefit from the low latency and the high temporal resolution of the camera, making it ideal for applications in autonomous driving, robotics, and, more generally, those that involve the usage of mobile platforms.

In 2019, Gehrig *et al.* [66] introduced the first end-to-end framework for learning event representations for object classification. While this approach shows high accuracy, it comes with significant computational costs and latency. After that, Cannici *et al.* [35] presented an approach that aggregates temporal information of events efficiently leveraging Long-Short Term Memory (LSTM) cells. Most recently, as detailed in Chapter 10, we propose a novel pre-processing pipeline that aggregates events into *memory* structures. This enables compatibility with state-of-the-art neural network architectures, allowing for both object classification and detection with high accuracy and minimal overhead.

It is also important to mention that, in the literature, several works have aimed to fully exploit the advantages of events, including their asynchronicity. In [139], the authors proposed a methodology that convert trained classification models with frame-like event representations into models that accept as input asynchronous events. In 2022, Schaefer *et al.* [174] introduced a graph-based approach to process events sparsely and asynchronously as a temporally evolving graph. That same year, Deng *et al.* presented EV-VGCNN [50] that utilizes voxel-wise vertices instead of point-wise inputs, effectively balancing accuracy and model complexity.

Object Detection

Object detection is the task of localizing objects in the scene, typically also providing their scale. The introduction of event-based sensors has opened new possibilities in this field, addressing some of the challenges faced with traditional frame-based methods [36] such as motion blur, occlusions, and extreme lighting conditions.

In the literature, some approaches perform object detection by organizing events into frame-like structures [123, 69] or event volumes [155, 88]. While effective, these methods have the drawback of losing part of the temporal information inherent in the event stream. To overcome this limitation, Li *et al.* recently presented ASTMNet [121], an architecture that directly processes asynchronous events. Other methods

explore the fusion of RGB frames and events to improve detection accuracy, both in normal and extreme conditions [97, 120, 197].

Recurrent neural network layers have been proposed to fully leverage the spatio-temporal properties of the event streams, resulting in significant improvements in detection accuracy [121, 69]. Also, for the task of object detection, an alternative strategy is presented in Chapter 10, with the introduction of the pre-processing pipeline based on *memory* of events.

Object Tracking

Tracking dynamic objects is a fundamental capability in robotics, supporting a wide range of applications such as obstacle avoidance [58, 209] and thus leading to safe navigation in cluttered environments, but also in fields like augmented and virtual reality [216, 91], as well as in monitoring driver health and behavior [98]. The characteristics of event-based vision sensors make them a promising approach to obtain real-time, high-performance tracking task, especially in scenarios involving fast object movements (e.g., pupil tracking), platform motion, or applications in dynamic environments.

An interesting aspect to consider, particularly when dealing with mobile platforms, is the fact that a moving event-based camera will produce events not only from dynamic objects in the scene but also from static elements due to the camera's own motion. A more detailed explanation of this phenomenon and how to tackle it is given in the following section.

3.6 Event-based examples in Mobile Robotics

Modern mobile platforms must operate and interact across a wide range of dynamic and unstructured environments, posing significant challenges for perception and decision-making. Event-based cameras offer a promising new sensing modality that mimics biological vision, aiming to replicate nature's proven efficiency in handling such complexity [149]. The application of event-based sensors to key tasks in robotic perception and navigation has become a hot topic in the field of neuromorphic engineering. Researchers have proposed numerous methodologies, leveraging both traditional artificial neural networks and more recent spiking neural network architectures. In the following sections, we explore several commonly

studied tasks in mobile robotics, highlighting approaches that utilize event-based data within conventional neural architectures.

Ego-motion estimation

To perform fast and efficient trajectories, a mobile platform must accurately determine its position relative to the environment during navigation. The collection of techniques used to estimate a robot's motion with respect to a reference frame is known as Visual Odometry (VO) or ego-motion estimation.

Brain-inspired approaches are particularly well-suited for onboard processing due to their low computational complexity and energy efficiency. These methods also enhance the robot's agility by minimizing perception latency, a critical factor for achieving real-time navigation at high speeds [58]. Ego-motion estimation techniques can be categorized into two main types: monocular and stereo algorithms.

- *Monocular Ego-Motion*: These pose estimation methods rely on a single monocular camera. Some approaches exploit event accumulation for pose estimation, such as the method presented in [99], where the authors propose a real-time joint estimation of 6-DOF camera motion, 3D structure, and intensity using three decoupled Extended Kalman Filter (EKF)-based probabilistic filters. Other methods leverage event volumes, for example, EVIO [244], the first event-based feature tracker that fuses event streams with inertial measurement unit (IMU) data. More recently, the authors of [229] introduced an approach that relies solely on events, employing an Evenly-Cascaded Convolutional Network (ECN) to estimate depth and a separate neural network to estimate camera pose.

- *Stereo Ego-Motion*: These pose estimation methods aim to mimic the dual-eye perception capabilities of humans. One of the earliest approaches was ESVO [240], which utilizes time surface representations of events. Moreover, in [78], the authors introduced a method that matches features across consecutive left and right event time surfaces. Unlike traditional methods that operate at fixed frequencies, this approach estimates pose dynamically, triggered by the density of incoming events to ensure sufficient visual information for reliable feature detection.

Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) algorithms aim to produce a globally consistent estimate of a mobile platform's trajectory while concurrently mapping the environment. A critical component in achieving global consistency, especially in long trajectories, is loop closure, which involves recognizing previously visited locations and performing corrective updates to reduce drift errors.

In [220], the authors introduced vSLAM, an event-based 2D SLAM approach extending the tracking algorithm proposed in [219]. In this method, the tracking process uses a set of particles to compute a multivariate probabilistic estimate of the current system state. Each particle represents a possible state of the system and carries an associated likelihood score. The feature-tracking map is dynamically updated during localization and built as an occupancy map, where each pixel indicates the likelihood of events occurring at that location. Building on this work, the same authors later developed a 3D SLAM [218] framework that addresses the limitations of 2D navigation. This new approach combines data from both RGB-D sensors and event-based cameras. However, the usage of frame-based sensors limits the low latency of event-based systems. In [141], the first purely event-based SLAM method incorporating loop closure was proposed. This initial solution includes three core modules: a loop closure algorithm based on SeqSLAM [143], a visual odometry module constrained to the ground plane, and a semi-metric, topologically consistent mapping tool derived from the experience mapping algorithm [142].

More recently, in [217], a method that employs an asynchronous Kalman Filter using purely event-based data was presented. This technique has been demonstrated to be effective in various applications, including the deblurring of conventional image frames under conditions of rapid motion.

IMOs Detection

The goal of Independently Moving Object (IMO) detection is to segment dynamic objects in a scene while the sensor platform itself is also in motion during environment exploration. This task is particularly challenging, as it typically involves estimating both the pose and velocity of dynamic objects. A major difficulty lies in distinguishing motion caused by the moving objects from motion induced by the ego-motion of the sensor.

Early approaches attempted to compensate for camera motion to isolate independently moving objects. This was often done by subtracting the estimated camera motion from the optical flow. Other methods leveraged ego-motion estimation to

project 3D motion into the 2D image plane, iteratively removing the ego-motion component from the estimated motion field. The residual motion was then used to segment independently moving objects, repeating the process until convergence. More recent methods have shifted towards deep learning techniques for improved accuracy and robustness. In [153], the authors introduced 0-MMS, a monocular motion compensation approach that utilizes event-based features. These features are extracted and tracked over time to perform motion compensation. By analyzing motion clusters, the method can effectively segment independently moving objects. In [146], another strategy was proposed based on two convolutional neural networks (CNNs). The first CNN estimates scene depth, while the second predicts the position and orientation of each pixel using sequences of event data within temporal windows. Combining the outputs of these networks provides both optical flow and segmented IMOs. A key innovation of this approach is the use of a lightweight "shadow network" with only 40 000 parameters. Additionally, the authors introduced a new event-based dataset called EV-IMO to support this research.

Part II

Frame-based Bio-inspired perception in Service Robotics

Chapter 4

Time-To-Transit: Definiton and Usage

Nature has long been a source of inspiration for robotics, from flexible soft robots that move like living organisms [8, 136], to fish-inspired platforms that glide smoothly through water [238] to small drones that agily navigate confined spaces with the maneuverability of birds [238, 29]. A crucial aspect in the advancements of biomimetic robots is represented by their ability to sense the surroundings, with vision as one of the most valuable means of perceiving the environment, as highlighted in the pioneering work of Lee and Reddish [113]. Indeed, by leveraging visual sensing, robots can achieve a detailed understanding of their nearby context, allowing them to navigate, interact and make decisions.

Several studies have been conducted to understand how animals maximize the use of visual cues, with particular attention given to birds and bats [45, 104, 186]. These animals possess a highly sophisticated sensorimotor system, enabling them to navigate complex environments and to react to various stimuli, ranging from stationary objects to the presence of other animals. A fascinating idea emerged among researchers at Cornell University, following the foundational work of J.Gibson which laid the basis for understanding the role of optical information in control [73]. In their work, Lee and Reddish [112, 113] examined how diving sea birds rely on a visual cue known as Time-To-Contact (TTC) as part of their behavioral strategy. Around the same period, Wang and Frost [215] discovered that neural mechanisms in the nucleus rotundus of pigeons play a significant role in signaling TTC. These studies suggest that animals like birds and bats use TTC to estimate the time remaining before encountering an obstacle in their flight path. Since then, time-to-contact has

been a topic of significant research interest in perceptual psychology [199, 57, 202], and it has been extensively studied for its application in navigation [96, 122]. In the early 2000s, Srinivasan *et al.* [186] showed that insects strongly rely on optical flow visual cues, generated by the movement of image patterns across their photoreceptors, to gauge distances to the surroundings and navigate through the environment.

In recent years, a concept closely related to time-to-contact, called Time-To-Transit (TTT), or τ (tau), has been introduced [10]. Intuitively, TTT represents the amount of time a moving robot, if it continues at its current speed and direction, would take to cross an imaginary plane passing through a visual feature in the environment. The research observed that this visual cue can be incorporated into steering laws to safely steer a mobile robot, mimicking the navigation strategies of bats and bees [104]. A key aspect in replicating the visual proficiency of animals consists in reliably estimating the TTT. State-of-the-art approaches typically considered the optical flow (OF) field, obtained from sequential pairs of images, as the starting point for the computation of TTT. The OF can be estimated through biologically plausible techniques, such as the correlational Elementary Motion Detector (EMD) model, as demonstrated by Shoemaker *et al.* in [179]. However, major advancements in optical flow estimation have been driven by computer vision algorithms, progressing from traditional approaches to deep neural networks. In 1981, Lucas and Kanade introduced an algorithm to compute sparse optical flow fields [130] while, around the same period, Horn and Schunk proposed a technique for obtaining a dense OF field [85]. More recently, with the advent of deep learning, novel techniques have been introduced to compute OF, leveraging the power of neural networks. Transformer-based architectures, such as FlowFormer++ [178] and GMFlow [226], have further enhanced the accuracy in dense OF estimation.

Precise optical flow values lead to reliable TTT quantities that can be used in steering control laws. Early research [175, 237] shows that simple control laws leveraging τ values as input can qualitatively synthesize bat-like trajectories. However, as firstly introduced by Kang and Baillieul [11], TTT is badly affected by rotational motions. Indeed, one of the main problems of TTT is that it can be correctly computed under the assumptions of:

- Constant speed
- No rotational motions

- Static scenario

A first attempt to overcome some of these limitations, particularly the no rotation condition, was made in [26] where we proposed the introduction of a sense-act cycle strategy to mitigate the effect of rotational motions in the optical flow and thus in the time-to-transit computation. By applying this strategy, we also demonstrate that a mobile robot could safely navigate an unknown environment using simple control laws, given a correct estimation of τ visual cues. Unfortunately, even if the sense-act cycle strategy proves effective, it clearly limits the rate at which the steer command can be issued and forces the control to be applied in open loop. In subsequent research, we improved upon our previous work [26] by developing a technique that does not require explicit optical flow calculation. This approach decouples motion and perception, thereby bypassing the limitations of the sense-act cycle. Moreover, the lightweight DNN we presented in this research, achieves accurate TTT estimation even in dynamic and featureless environments.

The following sections present an in-depth analysis of the two methodologies proposed by us. The work presented in [26] is analyzed in Chapter 5, while the improved research deriving from it, is explained in Chapter 6, highlighting the advantages and limitations of each approach, as well as the key advancements and innovations proposed.

4.1 Geometric and Perceived Time-To-transit

In this section, the concept of time-to-transit is presented. To better understand how it can be perceived and used to guide navigation, we consider the simplest possible context: a monocular camera-equipped mobile platform that is moving in a planar space with unicycle kinematics. The time-to-transit for the monocular camera can be computed by defining the kinematics of the vehicle as:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ u \end{pmatrix}, \tag{4.1}$$

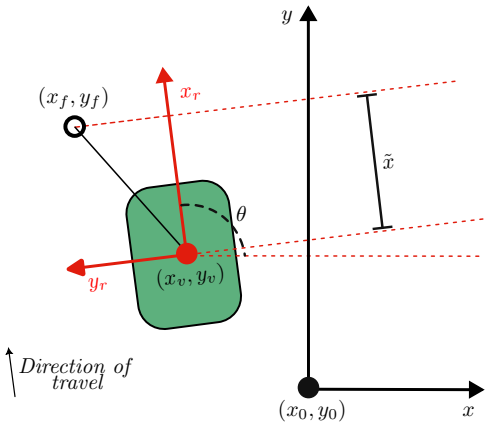


Fig. 4.1 Simple plane geometry representing a vehicle centered in (x_v, y_v), moving at a constant speed v. The global reference frame is fixed in (x_0, y_0). The point (x_f, y_f) indicates the position of a feature whose time-to-transit value measured by the vehicle is $\tau_g = \frac{\tilde{x}}{v}$.

Here v is the speed of the vehicle, θ is the angle between the x_r-axis of the reference frame fixed to the robot and the x-axis of a global reference frame, and u is the turning rate. Figure 4.1 gives a graphical representation of the considered situation.

The visual cue time-to-transit has a simple geometric interpretation. It represents the time required for a vehicle, moving at a constant speed along a straight-line path, to reach a specific plane in space. This plane is defined as passing through a given feature point in the environment while being perpendicular to the vehicle's direction of motion. Specifically, as discussed in [10, 175, 237, 11], if a feature point lies somewhere ahead of the vehicle/camera, possibly to the left or right in the environment, this plane can be used to obtain the time-to-transit value. Following the time-to-transit definition and by considering simple plane geometry, as depicted in Fig. 4.1, τ can be expressed as:

$$\tau_g(t) = \frac{\tilde{x}(t)}{v} = \frac{cos\theta(x_f - x_v(t)) + sin\theta(y_f - y_v(t))}{v}, \tag{4.2}$$

where (x_f, y_f) is the position of a static feature in the global reference frame.

Since in the equation 4.2, the time-to-transit can be computed by accessing to some geometrical quantities such as the relative position between the feature and the moving vehicle, and the vehicle's heading with respect to a global reference frame, we can name it *geometric* time-to-transit, or τ_g. Nevertheless, TTT is of interest as a

navigation signal considering that, under specific conditions, the geometric value of τ can also be perceived on the image plane. Specifically, imagine a vehicle, moving at a constant speed v, equipped with a front-looking camera. The platform's heading is maintained at a constant $\theta(t) = \theta_0$ to sustain a fixed lateral distance $d(t) = D$ from a designated feature point. Figure 4.2 gives a graphical representation of the analyzed case. Exploiting the similarity of triangles, it is possible to derive that:

$$\frac{d(t)}{\tilde{x}(t)} = \frac{d_i(t)}{f}. \tag{4.3}$$

From here, under the given assumptions, the following can be obtained

$$\dot{d}_i(t)\tilde{x}(t) - d_i(t)v = 0, \tag{4.4}$$

and by rearranging terms, the geometric τ value can be derived by leveraging perceived quantities from image point movement on the optical sensor:

$$\tau_g(t) = \frac{\tilde{x}(t)}{v} = \frac{d_i(t)}{\dot{d}_i(t)}. \tag{4.5}$$

Since now the TTT is computed from perceived measurements, we will define it as *perceived* time-to-transit $\tau_p = \frac{d_i(t)}{\dot{d}_i(t)}$. When the assumptions of constant speed v and heading θ_0 are respected, it follows that $\tau_p = \tau_g$, i.e., perceived and geometric time-to-transit coincide.

Unfortunately, these strong assumptions clearly limit the usage of time-to-transit in robotics applications, where the platforms must operate in real-world scenarios, taking real-time actions to, for example, avoid static and dynamic obstacles and explore unknown zones. In such scenarios, it is highly likely that the observed time-to-transit, denoted as τ_p, deviates from the ideal geometric value τ_g, as illustrated in Fig. 4.3, mainly due to unexpected rotations.

4.1.1 Measuring the distortion between τ_g and τ_p

When the fundamental assumption of constant heading is violated, the perceived time-to-transit differs from the geometric one. In order to quantify this difference,

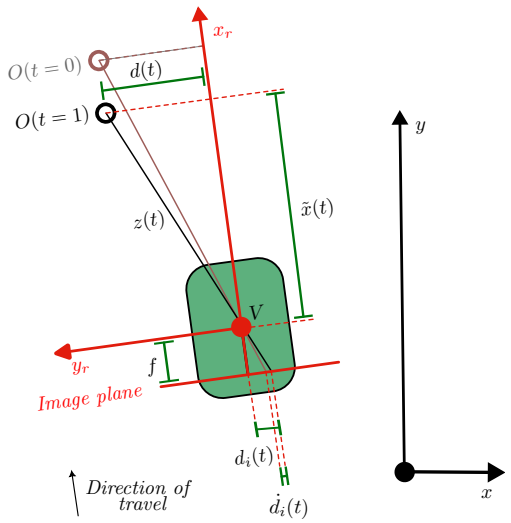


Fig. 4.2 A vehicle V is detecting a feature O whose distance from the vehicle is described over time by $z(t)$. If the direction of travel of the vehicle is kept constant, the distance $d(t)$ does not change over time and the geometric time-to-transit $\tau_g(t)$ given by feature O can be correctly estimated by the moving vehicle equipped with a monocular camera with focal length f using the perceived $\tau_p(t)$ since $\tau_g(t) = \tau_p(t) = \frac{d_i(t)}{\dot{d}_i(t)}$.

we can consider equation 4.3, and we can rewrite it as:

$$\dot{d}_i(t)\tilde{x}(t) + d_i(t)\dot{\tilde{x}}(t) - \dot{d}(t)f = 0 \tag{4.6}$$

Using the kinematics in (4.1) and the definition of τ_g in (4.2), equation 4.6 can be rearranged leading to the following geometric time-to-transit annotation:

$$\tau_g(t) = \tau_p(t) + \frac{\dot{d}(t)f + d_i(t)\delta(t)}{v\dot{d}_i(t)} \tag{4.7}$$

where $\tau_p(t) = \frac{d_i(t)}{\dot{d}_i(t)}$ and

$$\begin{aligned}\delta(t) = &+v(\cos^2\theta(t) - \sin^2(\theta)) + \\ &+u(y_f\cos\theta(t) - x_f\sin\theta(t)) + \\ &+\sin^2\theta(t) - \cos^2\theta(t)\end{aligned} \tag{4.8}$$

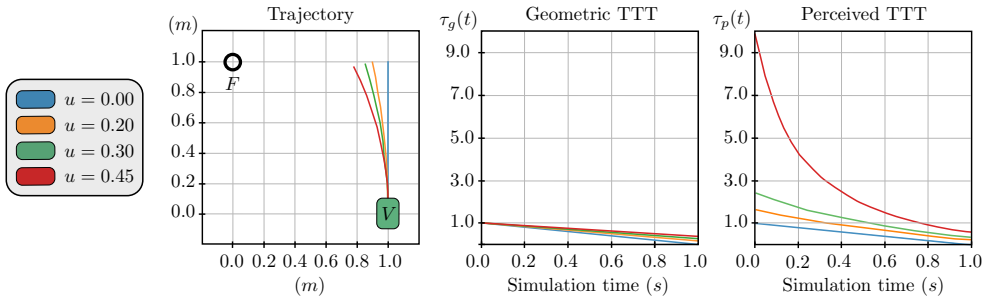


Fig. 4.3 Effect of rotation on TTT estimation: the unit-speed unicycle vehicle V moves to transit a point feature F that is a unit distance to its left and a unit distance ahead along its current heading. The vehicle turning rate toward the feature is $u = \dot{\theta}(t)$. While the geometric $\tau_g(t)$ is essentially the same under all motions, the perceived $\tau_p(t)$ value is badly distorted even for modest turn rates.

with (x_f, y_f) being the feature's position in the global reference frame. The effect of this distortion $\delta(t)$ and the subsequent discrepancy between perceived and geometric τ is shown in Figure 4.3.

In [26], we proposed a straightforward solution to mitigate the distortion by introducing the sense-act cycle strategy, where sensing and action phases are interleaved to ensure the τ computation only during linear motion. Building on this work, we further explore the use of deep neural networks to estimate the correct TTT even when rotations are present. In the next chapters, a detailed analysis of these two approaches is provided, describing their advantages and limitations.

Chapter 5

Monocular Visual Navigation Using Optical Flow and Time-to-Transit

5.1 Methodology

Monocular vision is lightweight and inexpensive. By combining optical flow with time-to-transit (TTT), we can derive temporal cues that serve as effective control signals for robot motion. In this work, we develop a complete pipeline for controlling a robot using control laws that leverage TTT values, starting with image acquisition from an RGB monocular camera. These images are processed to compute the sparse optical flow and to extract τ values, which are then integrated into simple control laws to navigate a mobile platform, specifically a Jackal robot, across various simulated and real-world scenarios. Building upon a previous work [11], we present an initial attempt, called *sense-act* cycle, to mitigate the effects of rotational motion and to extract reliable visual cues from image sequences. This approach enhances the robot's ability to navigate effectively by continuously processing and responding to visual input. An overview of the proposed methodology is shown in Figure 5.1. The following sections provide a detailed explanation of the *sense-act* cycle strategy and of the control laws used. Then, the results of multiple experiments conducted in both simulated and real-world environments are presented.

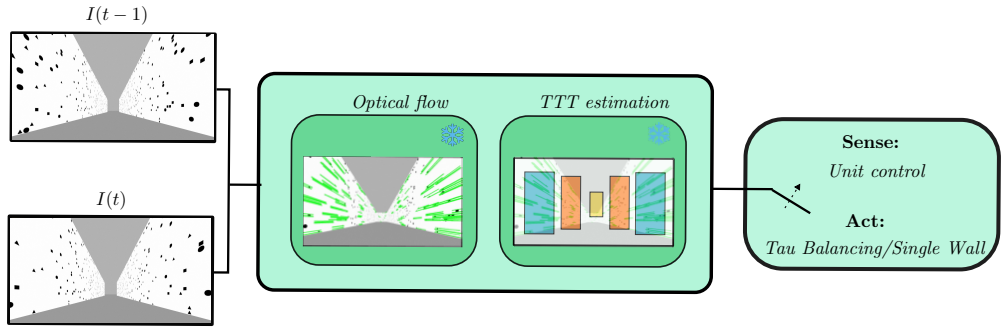


Fig. 5.1 Graphical representation of the pipeline used in this paper. Navigation is accomplished by processing sequences of images to compute the Lucas-Kanade sparse optical flow field and to extract reliable time-to-transit values, through the *sense-act* cycle.

5.1.1 Sense-Act Cycle

Following the pioneering works [104, 10], Baillieul and Kang [11] introduced a τ-based navigation strategy in 2020, leveraging the so-called Eulerian optical flow which assumes that the TTT values are continuously available at each photoreceptor. On the contrary, in this work, we adopt a more practical, less idealized implementation based on Lagrangian optical flow computation. Specifically, the perceived TTT values are obtained by processing images to compute a sparse optical flow field using the well-known Lucas-Kanade algorithm in its pyramidal version. Since this algorithm computes optical flow vectors for selected features in the image, it is both computationally efficient and fast enough to run in real-time without requiring high processing power. From the resulting optical flow field, we extract the TTT values necessary to compute the control actions for accurate navigation of the mobile platform. To enhance the robustness of these values, we define multiple Regions of Interest (ROIs) and use the average TTT value within each ROI as input to the control law. However, as discussed in Chapter 4, the reliability of these values, and consequently the control actions, significantly deteriorates when the robot performs rotational motion. To address this limitation, we introduce the *sense-act* cycle strategy.

The *sense-act* cycle consists of segmenting the motion of the platform by alternating straight and curved segments. During the sense phase, the robot moves

in a straight line while acquiring environmental data, computing the optical flow field, and estimating τ values. In the subsequent act phase, the control action is applied based on the values obtained in the previous step. An important aspect of this approach is to balance the duration of both phases: a sensing phase that is too long could lead to missing rapid variations in the optical flow field, potentially overlooking important environmental cues, whereas a too short phase may not provide sufficient data for robust control signals.

Figure 5.2 illustrates the improvement in TTT estimation when the *sense-act* cycle is applied, with the displayed values representing the average of the visual cues within a selected region of interest. To generate these results, we conducted a simulation in Gazebo using a Jackal robot equipped with a monocular camera. The robot collected data in the same environment (shown in the images in Figure 5.1) over three runs: (a) moving straight, (b) turning right, and (c) turning left. TTT values were computed both with and without the *sense-act* cycle. The results show that in cases (b) and (c), without the *sense-act* cycle, the perceived TTT signals are highly noisy and deviate significantly from the actual geometric values. In contrast, when the *sense-act* cycle is applied, the perceived and geometric values align more closely, demonstrating the effectiveness of this approach in improving TTT estimation and the overall navigation accuracy.

5.1.2 Control Laws

Reliable time-to-transit values can be used in simple control laws to control a mobile platform and enable it to navigate safely in multiple environments. One such control law, known as *Tau Balancing*, was first introduced in [10], and it consists in balancing TTT values from left and right portions of the visual field. This strategy draws inspiration from bats and bees, which, when flying through narrow passages, position themselves at the center of it [186].

In this work, we adapt the *Tau Balancing* strategy in order to use it with values obtained from different regions of interest (ROIs), specifically five, as illustrated in the central panel of Figure 5.1. The motion primitive is defined as follows:

$$u(t) = k_f(\tau_{fl} - \tau_{fr}) + k_m(\tau_l - \tau_r) \tag{5.1}$$

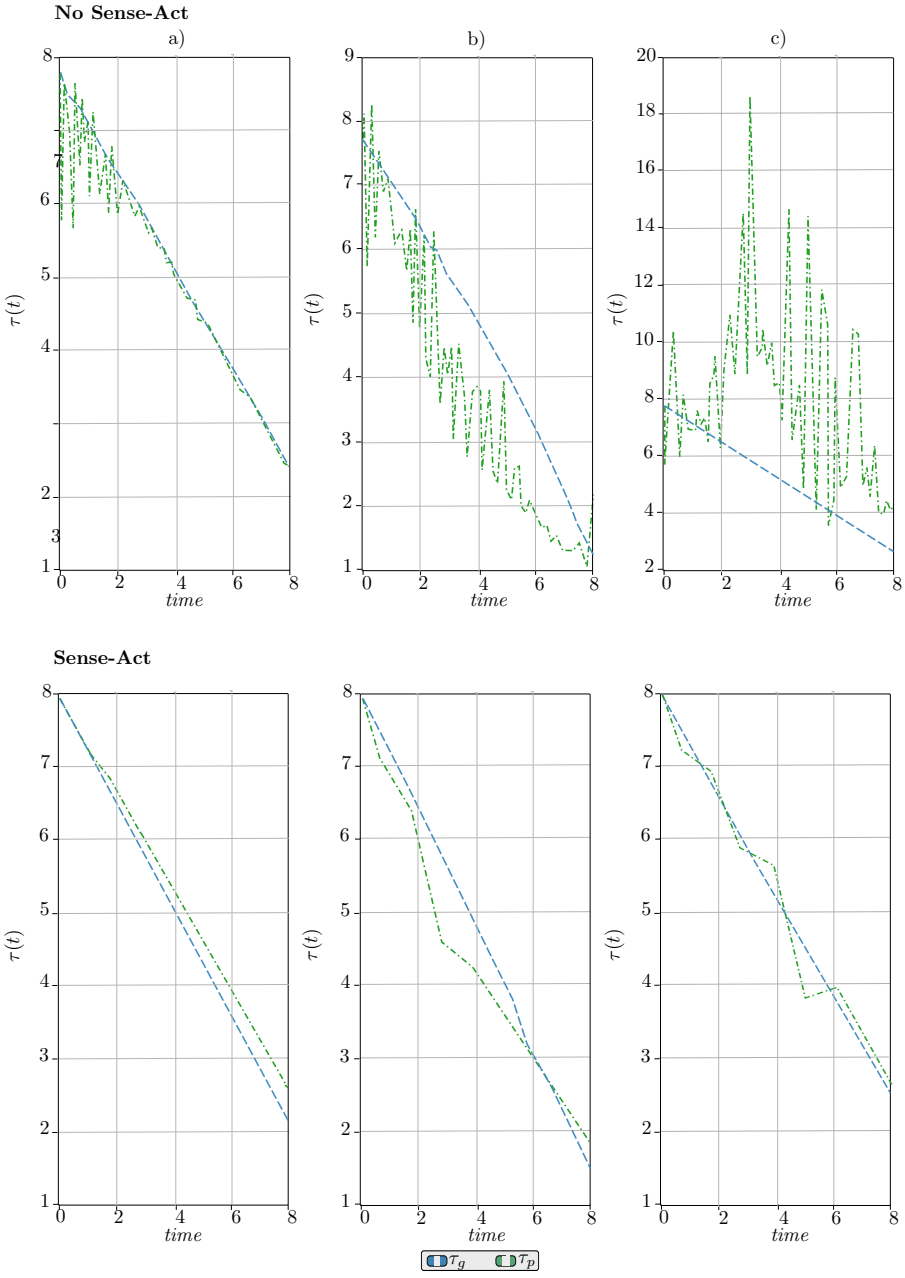


Fig. 5.2 Comparison between *geometric* (blue line) and *perceived* (green line) time-to-transit values during three tests: (a) moving straight, (b) turning right, and (c) turning left. The robot velocity is set to $v = 0.5m/s$, the sense phase lasts $0.4s$ and the act lasts $0.25s$. The cycle reduces noise and aligns perceived τ values with geometric ones.

where τ_{fl}, τ_{fr}, τ_r and τ_l are the average time-to-transit values computed from different ROIs of the image, excluding the central one. The constants k_f and k_m are empirically determined based on the application scenario. Importantly, following the methodology outlined in [10], it can be demonstrated that this control law remains stable for any choice of gains $k > 0$.

When a mobile platform navigates through an environment, there could be instances where visual cues from one side of the scene are not available. In such cases, the *Tau Balancing* control law becomes ineffective. To address this limitation, we introduce an alternative motion primitive called the *Single Wall Strategy*:

$$u(t) = \pm k(\tau_x - c) \tag{5.2}$$

where $\tau_x \in \{\tau_{fl}, \tau_l, \tau_r, \tau_{fr}\}$ and c is a constant [26] that can be set to maintain a specific distance from the wall that generates the detected features in the environment. Similar to *Tau Balancing*, the stability of this motion primitive is guaranteed for any value of $k > 0$. Detailed stability proofs can be found in the extended version of the paper [25].

During the experiments presented in the next section, the mobile platform autonomously switches between the two motion primitives just considering the available visual cues. This ensures an adaptive and robust navigation strategy for a wide range of scenarios.

5.2 Experiments and Results

Multiple experiments have been conducted to evaluate the reliability of the proposed methodology. Initially, we created various environments in Gazebo to assess the robot's navigation capabilities in simulation. Subsequently, we performed several tests in real-world scenarios to further validate the approach.

Simulation Results

The testing environments were designed with diverse geometric characteristics, including smooth curves, 90-degree turns, and corridors with one or two walls to assess both control strategies. Figure 5.3 illustrates examples of these environments, along with the paths followed by the robot using the proposed navigation pipeline.

The results demonstrate that the platform successfully navigates both artificial and more realistic environments, confirming that the TTT estimation is performed accurately and that the robot effectively applies the appropriate control law based on the encountered scenario.

Real World Results

Building upon the promising results obtained in simulations, several tests were also performed in real case scenarios. A Jackal UGV[1], equipped with a Stereolabs ZED camera[2](used in monocular mode), was utilized for these experiments. The platform is shown in Figure 5.4.

During each test run, the robot's position was computed by combining odometry and IMU data, referencing a coordinate frame with its origin at the point where the Jackal was first powered on. Figure 5.5 presents the results of these experiments. The environments tested featured different geometries that the robot successfully navigated without colliding with walls. The algorithm relied mainly on features represented by the edges and the corners of the post-it notes to compute τ values. However, since the tests were conducted in real-world settings, additional features from surrounding objects were also utilized.

5.3 Conclusions

In this work, we demonstrated the feasibility of using bio-inspired signals, called time-to-transit values, for reliable real-time navigation of a mobile platform in unknown environments. By adopting a *sense-act* strategy, we addressed the challenge of corrupted TTT values arising from rotational motions, enabling the effective use of these visual cues in simple yet robust control laws, namely the Tau Balancing and Single Wall strategies. Extensive experiments, conducted both in simulation and on real robotic platforms, validate the effectiveness of the proposed approach.

[1]Clearpath Jackal UGV: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/
[2]Stereolabs ZED camera: https://www.stereolabs.com/en-it

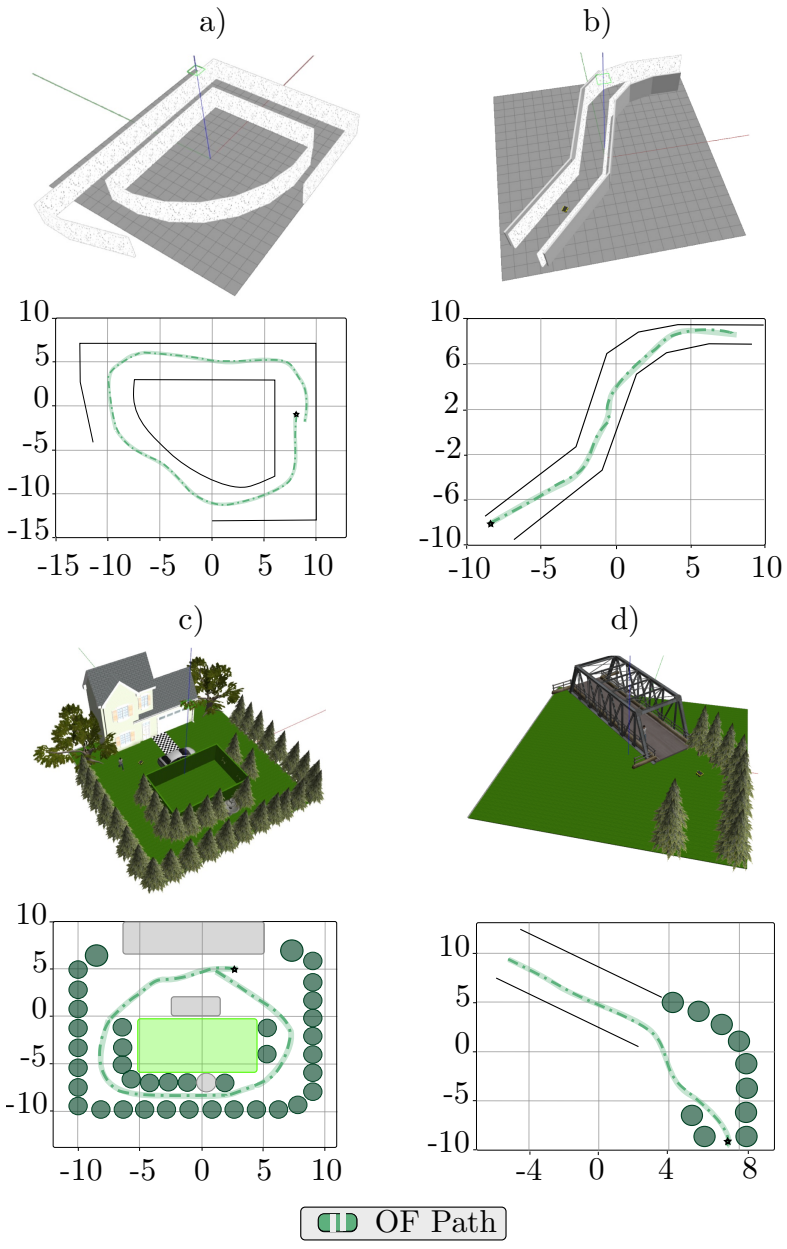


Fig. 5.3 An example of artificial (a, c) and realistic (b, d) environments together with the trajectories followed by the robot during Gazebo simulations when the proposed pipeline is applied. The robot's starting position is marked with a black star.

Fig. 5.4 The Jackal platform, equipped with a ZED2 stereo camera, used in this work.

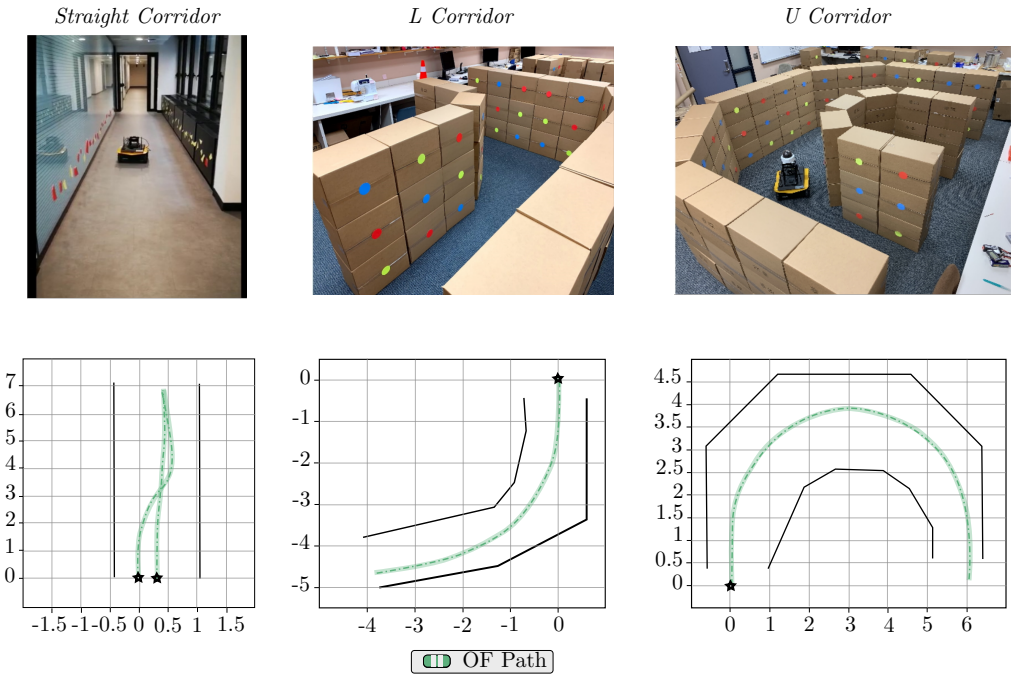


Fig. 5.5 Results of the experiments conducted in real-world scenarios. A variety of environments were constructed using moving boxes and the trajectories of the robot (starting from the point marked with a star) were collected by using a combination of IMU and wheel odometry.

Chapter 6

MobileNeTTT: a Lightweight DNN-based Time-To-Transit Estimator for Visual Navigation

Time-To-Transit (TTT) is a visual cue that can be computed from sequences of frames and used in simple control laws to navigate a robot in unknown environments, as demonstrated in Chapter 5. However, as discussed in Section 4.1, certain conditions must be guaranteed to obtain the correct TTT value, such as constant vehicle heading and speed. These constraints significantly limit the usage of time-to-transit in real-world applications, necessitating the development of methodologies that mitigate the effects of violating the restrictions. This work builds upon the previous research discussed in Chapter 5, proposing a novel approach to estimate time-to-transit values using a lightweight deep neural network. In particular, a modified version of the MobileNetV3 Small [86] is employed to eliminate the need for optical flow computation. By reinterpreting the problem as a variant of the traditional depth estimation task, the proposed approach offers several advantages: the real-time estimation of TTT without requiring a sense-act cycle, the improved estimation also in dynamic and/or featureless environment and the broader coverage of TTT values by considering a wider grid instead of just few regions of interest which leads to a better understanding of the environment.

The following sections detail the methodology and experimental results, including both network training and navigation experiments. The findings prove that the

proposed method is effective either in simulation and in real-world scenarios and it can be successfully implemented on a physical robotic platform.

6.1 Methodology

6.1.1 Deep Neural Network for time-to-transit estimation

In this work, the problem of estimating time-to-transit values has been reframed as a variant of the traditional depth estimation problem. Specifically, we use a deep neural network to obtain the depth $x(\tilde{t})$ and then compute the geometric TTT as $\tau_g = \frac{x(\tilde{t})}{v}$. Instead of estimating a per-pixel value, we adopt a coarser strategy by defining a mesoscale grid that aggregates areas of the original input image. This method reduces the resolution of the input and therefore significantly decrease the computational load of the depth estimator. The selected grid size balances a more biologically plausible coarse time-to-transit estimation with respect to per-pixel depth estimation, while still enabling a flexible definition of regions of interest (ROIs), as needed for different applications. To improve the reliability of the depth estimation, we also introduce a confidence mask with the same resolution as the depth map. The confidence helps identify the trustworthiness of each depth value by enabling the model to focus only on areas where meaningful depth information is present and avoiding regions, such as the sky in the image, where the depth value is infinite or not defined. The adoption of a depth estimation rather than directly predicting time-to-transit introduces several advantages:

- The model is encouraged to estimate a strong geometric quantity which is independent from the vehicle's speed.
- The model can be trained by exploiting the already present in the literature depth estimation datasets, enhancing the model's training, performance and generalization capabilities.

The objective of this research is to develop a lightweight model that can operate in real-time and that can be deployed on different robotic platforms. The depth estimation process takes two consecutive images, $I(t-1)$ and $I(t)$, as input. It generates a two-channel output tensor where the first channel is the estimated mean

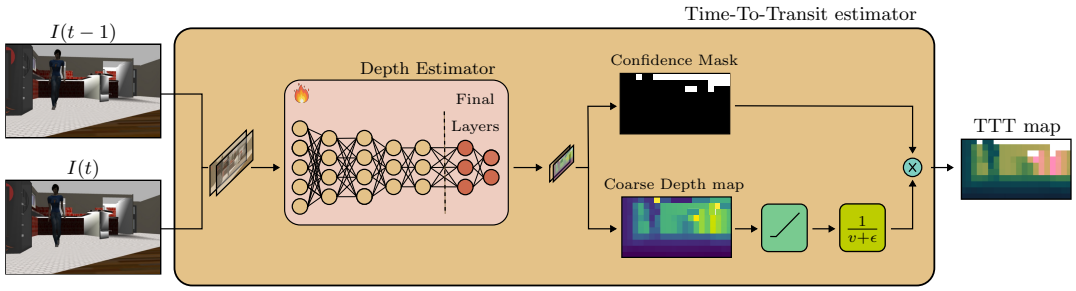


Fig. 6.1 Overview of the pipeline used to estimate Time-To-Transit (TTT). Starting from sequential RGB images $I(t-1)$ and $I(t)$, we train a depth estimator to produce a two-channel tensor containing an extremely coarse depth map and a binary mask indicating which area of the depth map contains reliable values. Depth values are then converted to time-to-transit values by dividing them by the robot's speed v, with a small constant ε added to avoid division by zero when stationary. The resulting time-to-transit values are then passed through a ReLU function to ensure all values remain non-negative.

depth values across the different image regions while the second channel is the binary confidence value indicating the reliability of the associated depth. To ensure non-negative depth values, the first channel is passed through a ReLU activation function. Finally, to obtain the geometric time-to-transit values, the grid of estimated $\tilde{x(t)}$ is divided by the platform velocity v, where a small ε is added to avoid division by zero, and multiplied by the confidence mask. Figure 6.1 shows the entire pipeline.

In order to reach the final goal, we compare different well-established neural network architectures, customized to estimate $\tilde{x(t)}$, and we consider their accuracy, memory footprint, and inference time to select the best option for our purpose. The architectures we analyze are MobileNet-v3 Small [86] with Channel attention [224], MobileNet-v3 Large [86], GMDepth [227], MobileViT-xs [137], ResNet-50 [83]. Each of these architectures has been modified to accept a six-channel input tensor and to produce a two-channel output. In the following a brief overview of the architectures selected for this work is reported, for a more detailed explanation please refer to Chapter 2.

MobileNetV3 Small and MobileNetV3 Large

MobileNetV3 models, introduced in 2019 [86], build upon previous MobileNet versions [87] with the objective of developing the next generation of high-accuracy efficient neural network models for on-device computer vision. Like previous versions, MobileNetV3 uses depthwise separable convolutions to significantly reduce the number of parameters and computational cost, along with inverted residual

blocks with linear bottlenecks to improve efficiency. However, it incorporates several key improvements, including Neural Architecture Search (NAS), the Hard-Swish activation function, and Squeeze-and-Excitatory (SE) blocks to improve channel-wise attention.

In this work, we enhance the performance of the MobileNetV3 Small model, by adding a simplified version of the Channel attention module described in [224]. Specifically, the six-channel input is max-pooled and then flattened, the resulting tensor is fed into a three-layer MLP, with 6, 3, 6 neurons layers respectively, to produce a 1×6 tensor containing the attention scores for each input channel. These scores are multiplied by the corresponding input channel and the result is finally passed to the traditional MobileNetV3 Small. The resulting model is referred to as MobileNeTTT.

ResNet-50

ResNet-50 is a deep neural network from the ResNet family [83], consisting of 50 convolutional layers. It is characterized by the presence of shortcut connections that enable residual learning, effectively addressing the vanishing gradient problem. This allows for the successful training of very deep neural networks. ResNet-50 is widely used in various computer vision tasks thanks to its ability of generalize well across different datasets.

MobileViT-xs

MobileViT was introduced in [137], where the authors present this new architecture as the transform-like version of the MobileNet family. It combines the local feature extraction capabilities of convolutional neural networks with the global context modeling of transformers, enhancing its effectiveness for computer vision tasks, such as image classification and image segmentation. In [137], the authors propose multiple model sizes, and we choose the MobileViT-xs variant due to its comparable number of parameters to MobileNetV3 Small. Additionally, MobileViT-xs is designed for efficiency on mobile and edge devices, making it suitable for applications in resource-constraint environments that require a balance between performance and computational cost.

GMDepth

GMDepth is the name given to the model presented in [227] when it is used for depth estimation task. Indeed, the authors introduce a unified model capable of

estimating optical flow, rectified stereo matching, and per-pixel depth. The model features a transformer-based feature extraction layer, which is crucial for obtaining discriminative features for matching. In particular, the cross-attention mechanism plays a fundamental role in integrating knowledge from another viewpoint through the cross-view interaction, enhancing the quality of the extracted features. Once the features are extracted, the model performs the feature matching, using a task-dependent parameter-free matching mechanism, which produces the selected output (optical flow, stereo matching or depth). Additionally, the architecture includes also a final self-attention layer, which refines predictions by propagating high-quality estimates to unmatched regions by measuring self-similarity. In this work, we use the network for depth estimation, feeding it with two consecutive RGB images.

6.2 Datasets and Metrics

6.2.1 Datasets

All the selected models must then be trained to obtain the depth values. Training a model to predict depth rather than directly estimating time-to-transit values enables the possibility of exploiting well-established state-of-the-art datasets. Among the various datasets available in the literature, we have chosen the KITTI [70] and the RBGD-SLAM [188] datasets because they can prove the effectiveness of our methodology in real-world application scenarios. The KITTI dataset provides data from complex urban environments, while the RGBD-SLAM dataset includes indoor office settings. Additionally, we have created a custom dataset specifically designed for our task. It consists of synthetic data collected with a Jackal robot equipped with a monocular camera and a 3D LiDAR sensor navigating in simulated Gazebo environments.

To generate ground-truth data that matches the requirement for the specific task of this work, the data in each dataset has been structured to link consecutive images together with a coarse depth map and a corresponding binary confidence mask. Each depth map cell contains the mean depth value (Kinect depth values for the RGBD-SLAM dataset and 3D LiDAR values for the KITTI and custom datasets) among all the pixels within it, normalized between 0 and 1, reflecting the range of the LiDAR sensor used. The confidence mask assigns a value of 0 where the LiDAR

data are not reliable, i.e unavailable, and a value of 1 otherwise.
Below are the details of each dataset:

- **KITTI Dataset**: The KITTI dataset is widely used in the field of robotics and computer vision. It contains a large amount of data collected from a vehicle equipped with a 3D LiDAR, stereo and monocular cameras, and a GPS/IMU sensor. This dataset serves as a benchmark for multiple tasks, including depth estimation, optical flow computation, object detection, and segmentation. In our work, we utilize only a portion of the KITTI dataset, specifically the *Residential* section, as it includes recordings from urban environments with a high presence of dynamic obstacles such as pedestrians, bicycles, and cars. These scenarios are more relevant to our study, and they are more challenging compared to other recordings taken on the highway, where the scene has more uniform conditions. The final dataset consists of 18 248 images, split into 14 320 for training (80%) and validation (20%) and 3928 for testing.

- **RGBD-SLAM Dataset**: The RGBD-SLAM dataset consists of indoor videos paired with depth maps captured using a Kinect sensor. We focus only on portions of the data available, in particular the recordings from the *Handled* and *Robot SLAM* categories, for a total of 15 videos. Among them, we excluded 4 that are not relevant to our purpose - those collected with a fixed rotating camera or those primarily focusing on the floor. The remaining sequences are then divided into 10 803 images for training, 2700 for validation, and 3456 for testing.

- **Custom Dataset**: Our custom dataset has been created using a robotic platform equipped with a 3D LiDAR and a monocular RGB camera. The two sensors were mounted on the robot at the same **xy** position but at different heights. The data was collected in simulated indoor and outdoor environments created in Gazebo, featuring both dynamic and static objects. Some environments in our collection are original, while others have been previously utilized in the literature[1]. Examples of these environments are illustrated in Figure 6.2.

  To ensure accurate image-label pairing, a spatial and temporal alignment between the outputs of the two sensors must be performed. Specifically,

[1]The environments not developed by us are sourced from https://github.com/leonhartyao/gazebo_models_worlds_collection.

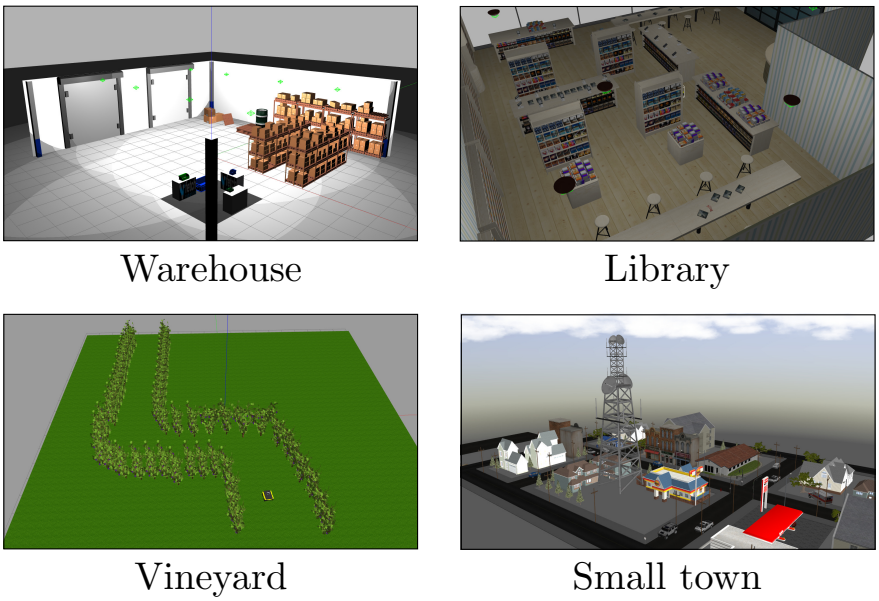


Fig. 6.2 Four of the different simulated environments used in our custom dataset.

temporal alignment was achieved through precise timestamping, while spatial alignment was guaranteed by projecting the 3D LiDAR points into the camera reference frame. The transformation between the two sensors was calibrated following the process outlined in [103]. The dataset obtained counts 17 612 images of size 640×480, with 14 612 images used for training and validation (80% and 20%, respectively) and 3000 images reserved for testing. The test images were collected in different environments from those used for training and validation.

6.2.2 Metrics

In this work, the different neural networks are evaluated using metrics commonly adopted in depth estimation tasks, as outlined in [235]. These include the Root Mean Squared Error (RMSE), the Absolute Relative Error (ARE) and the Threshold Accuracy. In addition, we assess the models' ability to estimate the binary confidence mask using the Macro-averaged F1 score. In more detail:

- **Root Mean Square Error**: This metric measures the square root of the mean squared differences between the predicted and ground-truth values. It quantifies how closely the estimated values align with the actual values.

- **Absolute Relative Error**: This metric calculates the absolute difference between the actual and predicted values, normalized by the actual value. It provides a quantitative indication of the prediction accuracy.

- **Threshold Accuracy**: This metric represents the percentage of predictions that fall within a specified range of the ground-truth values. Formally, considering the ground-truth value y_i and the predicted value $\hat{y}_i$ the threshold accuracy is defined as:

$$\delta = \max\left(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}\right) < \varepsilon \tag{6.1}$$

  Following standard practice in the literature [74], we evaluate this metric at three predefined thresholds: $\varepsilon = 1.25$, $\varepsilon = 1.25^2$, and $\varepsilon = 1.25^3$. These thresholds are associated to the accuracy metrics δ_1, δ_2, δ_3, and define acceptable error margins within which the model's predictions are considered accurate.

- **Macro-averaged F1 [185]**: The problem of estimating the values of the confidence mask is treated as a multi-label classification problem where class labels represent *'confident'* (1) and *'not confident'* (0) predictions. The Macro-averaged F1 score computes the F1 score separately for each class and then averages them, ensuring a balanced evaluation regardless of class distribution. This metric effectively captures the model's accuracy in estimating the binary confidence mask, considering both precision and recall equally across classes.

6.3 Results

6.3.1 Comparison of Neural Network-based TTT Estimators

In order to train all the neural network architectures described in Section 6.1.1, in this work we utilized a common loss function, adjusting only the hyperparameters based on the selected network. Our loss function is inspired by the well-known YOLO [195] loss, which effectively handles multi-channel outputs and the usage of

a confidence mask. The loss function is defined as follows:

$$\begin{aligned}\mathcal{L} = \lambda_{TTT} \sum_{i=0}^{H} \sum_{j=0}^{W} C_{ij}^{gt} (\tilde{X}_{ij}^{gt} - \tilde{X}_{ij}^{est})^2 + \\ + \lambda_{data} \sum_{i=0}^{H} \sum_{j=0}^{W} C_{ij}^{gt} (C_{ij}^{gt} - C_{ij}^{est})^2 + \\ + \lambda_{nodata} \sum_{i=0}^{H} \sum_{j=0}^{W} (1 - C_{ij}^{gt}) (C_{ij}^{gt} - C_{ij}^{est})^2\end{aligned} \tag{6.2}$$

where the output dimensions are defined as $H = 10$ and $W = 20$. The ground-truth labels include $C^{gt} \in 0, 1^{H \times W}$ for the confidence mask and $\tilde{X}^{gt} \in \mathbb{R}^{H \times W}$ for the depth values grid. The estimation error on $\tilde{x}$ values is weighted by the constant λ_{TTT}, while the confidence mask prediction error is weighted by two different constants, λ_{data} and λ_{nodata}. This distinction takes into account the fact that typically there is an imbalance between the number of zeros and the number of ones in the confidence mask. Without separate weighting, an incorrect estimation of zeros could disproportionately impact the loss function. An alternative approach to address this issue is the use of focal loss, which dynamically adjusts the contribution of each term in the loss based on prediction confidence.

All neural network models were trained on our custom Gazebo dataset, as described in Section 6.2.1. Starting from pretrained models[2], we trained all networks using the Adam optimizer [102] for 100 epochs with an initial learning rate of 0.001. The results on the test set are shown in Table 6.1. In addition, we use the other two real scenes datasets (KITTI and RGBD-SLAM), to finetune the best-performing model from the initial training of each network. The fine-tuning process involved training for 50 epochs using the same optimizer and the same starting learning rate of the training phase. Table 6.1 illustrates the finetuning results on the test set.

Finally, since real-time operation on a robotic platform is a key requirement in this work, we also evaluated each network's computational efficiency. Specifically, we analyzed the number of parameters and inference time for each model. These evaluations were conducted on an NVIDIA Jetson Orin Nano[3], an edge computing

[2]Pretrained models used in this work, where available, were obtained from https://huggingface.co/. For GMDepth, we utilized the pretrained model from https://github.com/autonomousvision/unimatch.

[3]NVIDIA Jetson Orin Nano website: https://www.nvidia.com/it-it/autonomous-machines/embedded-systems/jetson-orin/

		RMSE ↓	ARE ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	F1 Macro ↑
Synthetic	ResNet-50	0.129 ± 0.003	0.170 ± 0.002	76.2% ± 0.3%	88.8% ± 0.1%	93.9% ± 0.1%	84.1% ± 0.1%
	MobileViT-xs	0.128 ± 0.004	0.165 ± 0.005	80.2% ± 0.1%	90.7% ± 0.2%	94.6% ± 0.1%	85.9% ± 0.1%
	MobileNet-Large	**0.113 ± 0.004**	0.158 ± 0.002	79.9% ± 0.3%	90.5% ± 1.0%	94.5% ± 0.6%	**86.9% ± 0.6%**
	GMDepth	0.144 ± 0.005	0.213 ± 0.014	71.1% ± 1.3%	85.9% ± 1.1%	93.4% ± 0.3%	76.0% ± 1.6%
	MobileNeTTT	**0.114 ± 0.004**	*__0.152 ± 0.003__	*__80.8% ± 0.4%__	*__92.1% ± 0.5%__	*__95.8% ± 0.4%__	**86.4% ± 0.1%**
RBGD-SLAM	ResNet-50	0.044 ± 0.001	0.219 ± 0.003	73.7% ± 0.4%	89.2% ± 0.3%	93.8% ± 0.3%	83.0% ± 0.1%
	MobileViT-xs	0.038 ± 0.001	0.189 ± 0.002	*__79.4% ± 0.2%__	**92.3% ± 0.2%**	**96.2% ± 0.5%**	83.9% ± 0.7%
	MobileNet-Large	0.038 ± 0.001	0.190 ± 0.005	77.4% ± 0.9%	91.3% ± 0.3%	94.9% ± 0.3%	84.5% ± 0.1%
	GMDepth	0.065 ± 0.003	0.379 ± 0.008	54.8% ± 0.7%	74.9% ± 1.2%	85.5% ± 1.7%	78.8% ± 1.2%
	MobileNeTTT	*__0.035 ± 0.001__	*__0.185 ± 0.002__	76.9% ± 0.6%	**91.7% ± 0.6%**	**95.6% ± 0.2%**	*__84.8% ± 0.1%__
KITTI	ResNet-50	0.333 ± 0.010	0.113 ± 0.003	89.2% ± 0.3%	94.7% ± 0.2%	95.8% ± 0.3%	95.0% ± 0.1%
	MobileViT-xs	0.299 ± 0.003	0.114 ± 0.001	**90.1% ± 0.2%**	94.5% ± 0.1%	95.6% ± 0.1%	94.5% ± 0.2%
	MobileNet-Large	0.316 ± 0.008	0.114 ± 0.003	**90.6% ± 0.5%**	94.7% ± 0.2%	96.2% ± 0.2%	95.1% ± 0.1%
	GMDepth	0.355 ± 0.011	0.142 ± 0.002	84.2% ± 0.3%	92.6% ± 0.5%	94.9% ± 0.4%	93.8% ± 0.2%
	MobileNeTTT	*__0.292 ± 0.002__	*__0.103 ± 0.003__	**90.2% ± 0.3%**	*__95.2% ± 0.1%__	*__96.7% ± 0.2%__	*__95.8% ± 0.1%__

Table 6.1 Evaluation and comparison of the performance of different neural models with different architectures and sizes when tested on our custom synthetic dataset, on the RGBD-SLAM dataset as well as on the KITTI dataset. In **bold** the best model is reported, considering models equally best if their standard deviations overlap. When the best model is unique, we mark it with a *. In light gray the model with fewer parameters is shown.

device optimized for AI tasks on resource-constrained platforms. The results are reported in Table 6.2.

Depth Estimator	Param. (M)	Memory (MB)	Inf. Time (ms)
ResNet-50	26.62	114.56	58.51
MobileNet-Large	5.22	20.97	26.75
GMDepth	4.68	18.72	112.96
MobileViT-xs	2.48	10.03	50.42
MobileNeTTT	**1.95**	**7.86**	**22.48**

Table 6.2 Number of parameters, memory occupation and inference time (on a Nvidia Jetson Orin Nano) of the different neural models. In **bold** the best model and in light gray the estimator selected for the remainder of the work is highlighted.

Upon analyzing both the performance metrics and computational efficiency, we observed that larger models, such as GMDepth and ResNet-50, exhibited inferior performance overall. This is probably related to the fact that they need more data to be properly trained. Moreover, these models demand significant computational resources and time for inference. Conversely, smaller models performed well, with minimal differences among them. MobileNeTTT slightly outperformed the others, achieving the best results in ARE, δ_1, δ_2, and δ_3, while matching the performance of

MobileNet-Large in terms of RMSE and F1 macro on the custom Gazebo dataset. Additionally, as shown in Table 6.1, MobileNeTTT achieved the best RMSE and ARE values on both KITTI and RGBD-SLAM, while surpassing or matching the performance of larger models in terms of δ_1, δ_2, and δ_3. It is also important to mention that this MobileNet modified version we proposed has the smallest number of parameters and operates in real-time, achieving an inference time of approximately 20 ms, as shown in Table 6.2. Considering all the results, we selected the MobileNeTTT network as the architecture for comparison with the TTT estimation technique proposed in Chapter 5.

6.3.2 Comparison MobileNeTTT vs OF-based Estimator

The quantitative analysis performed in the previous section was really useful in identifying which model, among the different ones considered, was the best one to estimate $x(\tilde{t})$ values, and consequently the time-to-transit. However, to fully demonstrate the effectiveness of the proposed methodology and its improvements over the approach presented in Chapter 5, a further analysis is necessary between the MobileNeTTT and the OF-based TTT estimators. In Chapter 5 we obtained TTT values using optical flow, following the equation $\tau = d_i(t)/\dot{d}_i(t)$. The quality of this estimation strongly depends on feature detection and on the ability to compute their displacements in the subsequent frames. With this information, and given the velocity of the robot (which must remain constant) and the time interval between two consecutive frames, the time-to-transit values can be computed by following the equations presented in Section 4.1. Since this technique does not require training data and can only be applied under the constraint of no rotational motion, the test datasets utilized in Section 6.3.1 must be enlarged by including also situations where only straight motions are performed. Thus, to ensure a fair comparison, we designed new environments where the robot moves without rotational motion.

The evaluation of TTT estimations between MobileNeTTT and the methodology proposed in [26] is performed by providing two different analysis:

- **The mean squared error (MSE) and the Wasserstein distance**. These metrics are obtained by considering mean TTT values along the horizontal axis, as the TTT-based control primarily focuses on lateral balancing. The results, summarized in Table 6.3, are derived from the test set of our custom

Scenario	**Metric**	**OF-based** [26]	**MobileNeTTT**
w/o rotations	MSE ↓	1.40 ± 0.75	$\mathbf{0.01} \pm \leq 0.01$
	Wasserstein ↓	1.59 ± 0.72	$\mathbf{0.10} \pm 0.07$
with rotations	MSE ↓	7.42 ± 2.18	$\mathbf{0.02} \pm \leq 0.01$
	Wasserstein ↓	2.28 ± 1.12	$\mathbf{0.31} \pm 0.11$

Table 6.3 Comparison of TTT estimation quality between the OF-based method and MobileNeTTT on our synthetic dataset test set.

dataset and the additional recordings of the robot moving in a straight motion. In both metrics, MobileNeTTT demonstrates significantly better performance compared to the OF-based method.

- **Regions of interest (ROIs)**. The ROIs are the same for both approaches. For MobileNeTTT, they are obtained by aggregating cells of the predicted coarse grid. The TTT estimation was analyzed in three key scenarios[4], considering different regions of the image: (1) the lateral region, (2) the central region when trackable features are present in the environment, and (3) the central region when features are removed. The results for the three situations are shown in Figure 6.3. When using optical flow, the TTT estimation is generally noisier, particularly in the central area of the image, near the focus of expansion (FOE). In this region, feature displacement is minimal, making it more challenging to compute. The third environment clearly demonstrates that, as expected, TTT estimation from optical flow in a featureless environment is not possible. In contrast, MobileNeTTT provides more reliable estimates across all environments and regions of interest, and this is a straightforward consequence of the accuracy in estimating $x(\tilde{t})$ already discussed in the previous subsection.

A possible explanation for the improvement brought by MobileNeTTT estimation is that the neural estimator processes the entire image as input rather than relying on sparse, independently detected features. This broader perspective allows it to extract more comprehensive scene information, leading to more stable predictions. Additionally, MobileNeTTT offers another important advantage; indeed, it eliminates the usage of the *sense-act* cycle introduced in Chapter 5. This enables the estimation of time-to-transit values even when the robot is rotating and navigating dynamic

[4]To ensure fair comparisons, the environment selected for the experiments was new to MobileNeTTT (e.g. not present in its training set).

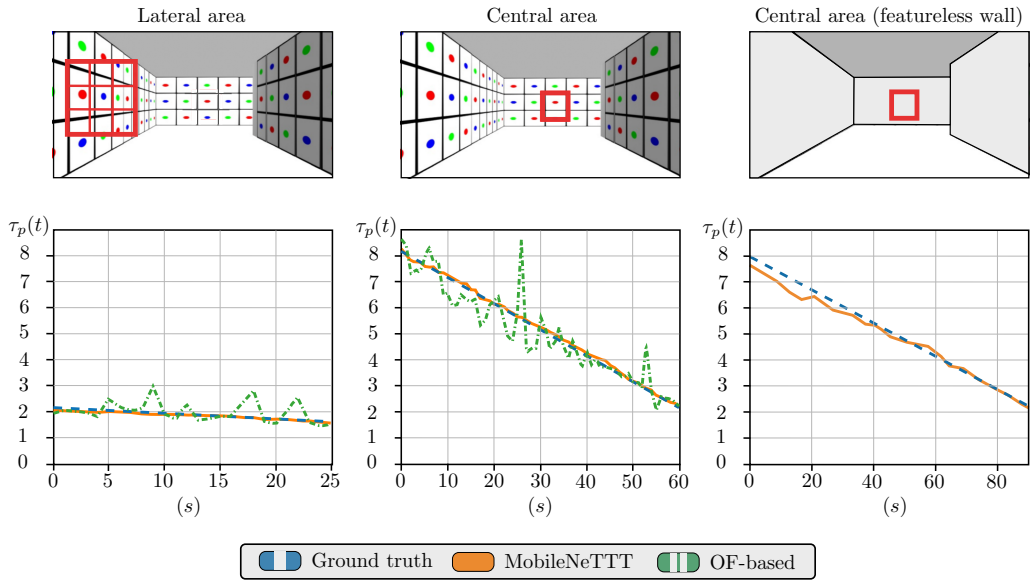


Fig. 6.3 Comparison of time-to-transit estimation using MobileNeTTT and Optical Flow feature tracking across various scenarios. In the three scenarios, the robot moves without rotations. On the left and in the center, the environment is a corridor with a high number of features, while on the right, the environment is the same but with a lower number of trackable features on the walls. The red outer area represents the Region of Interest (ROI) where features are selected and then tracked using Optical Flow. The ROIs are combinations of the grid cells provided by MobileNeTTT. MobileNeTTT offers superior, less noisy estimation quality, particularly in the central regions of the image, and provides an acceptable estimate even in areas with a low number of trackable features where Optical Flow estimation is not possible.

environments, significantly improving its applicability in real-world scenarios. As further proof, we tested both TTT estimation approaches on the KITTI dataset, where data are recorded using an RGB camera mounted on a vehicle moving in dynamic environments. As illustrated in Figure 6.4, MobileNeTTT provides accurate TTT estimations, while the optical flow-based method fails to compute meaningful results, mainly because the fundamental assumption of no rotational motions is violated.

6.4 Time-to-transit Vision-based Navigation using MobileNeTTT

A correct and reliable estimate of time-to-transit values is crucial for effectively controlling a mobile platform's movement. This section presents simple examples of

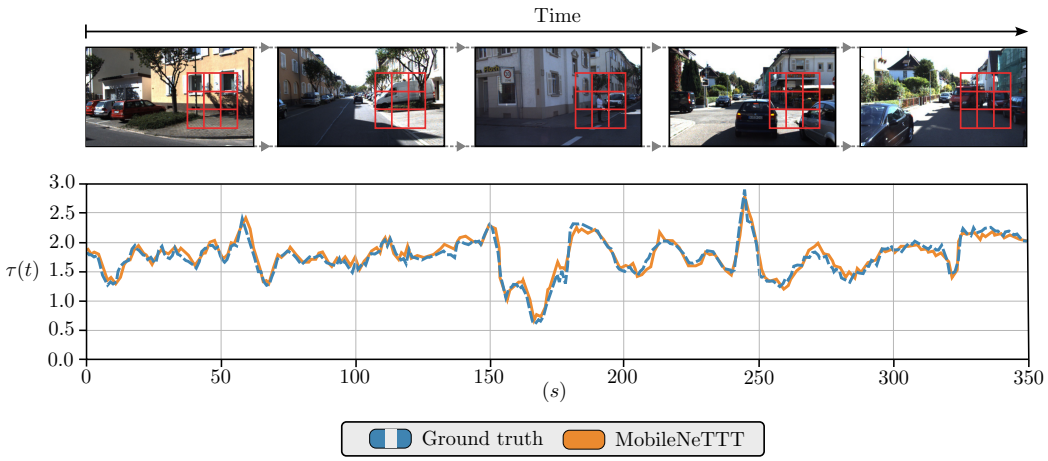


Fig. 6.4 TTT estimation quality of MobileNeTTT dealing with real data coming from the KITTI dataset. The plot shows the mean of time-to-transit values coming from the image's selected red ROI.

how the improved estimation can positively impact navigation. In particular, the grid estimation of time-to-transit leads to a better understanding of the environments and to the possibility of switching more accurate among the control laws.

All the experiments presented in this section, either conducted in simulations and in real-world environments, leverage the two steering laws proposed in [10] and in Chapter 5. Specifically, considering a camera-equipped mobile platform with kinematics as the one expressed in Equation 4.1, moving at a constant velocity v, the two control laws considered are:

- **Tau Balancing**:

$$u(t) = k(\tau_l(t) - \tau_r(t))$$

  where τ_r and τ_l represent the average TTT values computed from regions on the right and left of the input image, respectively. In a corridor-like environment, this control law helps guide the vehicle toward the corridor's center.

- **Single Wall strategy**:

$$u(t) = \pm k(\tau_x - c)$$

  where τ_x is the available TTT value in cases where the estimation technique fails to provide values for some visual regions. The sign is defined according to the side of the image from where the τ_x value is coming from and c is

a parameter that controls the vehicle's distance from the wall on which the detected feature is located.

6.4.1 Better understanding of the environments

In the OF-based estimation strategy, due to the estimation noise of TTT, τ_r and τ_l represent the average time-to-transit values from fixed lateral areas of the image. This strongly limits the understanding of the environment's geometry, leading to two interconnected issues for navigation: 1) incorrect switching between control laws and 2) reliance on fixed ROIs for control. In Figure 6.5, we show an example of the input image on the left, with the output of the OF-based method and the vertically averaged output of MobileNeTTT on the right. Thanks to improved estimation, it is possible to generate an output that offers more detailed information about the surrounding environment. Furthermore, existing control laws can still be applied, while leveraging the discontinuities in TTT values to create dynamic left and right ROIs that adapt to the environment. In the experiments conducted in this work, as well as in Figure 6.5, we start by identifying the column with the TTT peak. To define the right area, we select the column with the highest TTT estimate that is at least 10% lower than the maximum value. In the image, everything to the right of this point is considered the right area. A similar approach is applied to define the left area.

6.4.2 Navigation improvements

Various tests have been conducted, and as described in the previous section, τ_r and τ_l represent the average time-to-transit values from the lateral areas of the image. However, for the OF-based estimator the two time-to-transit values are computed using fixed and pre-defined ROIs, while for the MobileNeTTT estimator, the average time-to-transit values are obtained from the dynamic ROIs. Figures 6.6 and 6.7 illustrate the results obtained in simulated environments. In Figure 6.6 a Jackal robot moves in an L-shaped scenario, starting from a point marked by a star. In this case, both estimation methods provide reliable TTT signals, allowing the robot to complete the path and reach the end of the environment without colliding with the walls. However, the optical-flow-based control strategy induces oscillatory behavior due to the *sense-act* cycle, where sensing must be performed in open loop,

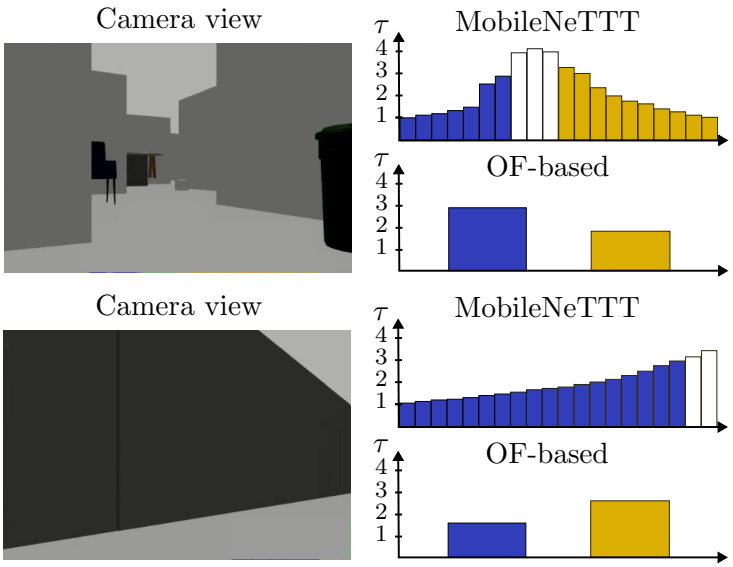


Fig. 6.5 An example of an environment and the TTT estimates produced by the OF-based method and MobileNeTTT. MobileNeTTT enables a better understanding of the environment and the definition of dynamic left (blue), right (yellow), and central (white) ROIs.

preventing real-time rotational adjustments. In contrast, the trajectory generated by MobileNeTTT exhibits minimal oscillations, enabling the robot to maintain a more stable position at the center of the corridor.

Two more complex scenarios are depicted in Figure 6.7. On the left, the robot operates in a corridor-like environment within a vineyard, while on the right, it navigates through an area featuring obstacles and walls with few distinguishable features. In the vineyard scenario, the robot collides into the plants when using the OF-based TTT estimation, probably due to the similarity and high sparsity of features. In contrast, it successfully completes the path when using MobileNeTTT, even if the trajectory of the robot suggests that the control laws and the controller could be further improved to achieve greater stability. In the second case, the "zigzag" corridor, when using the OF-based method, the robot is unable to complete the path, even starting from different positions, mainly because it does not receive enough information from the environment. Conversely, the neural network based approach makes the robot successfully navigate the corridor without collisions, indicating that the time-to-transit values are accurately estimated.

Preliminary results were also obtained in a real-world scenario, as shown in Figure 6.8. The experiments were conducted using an AgileX Limo rover moving at

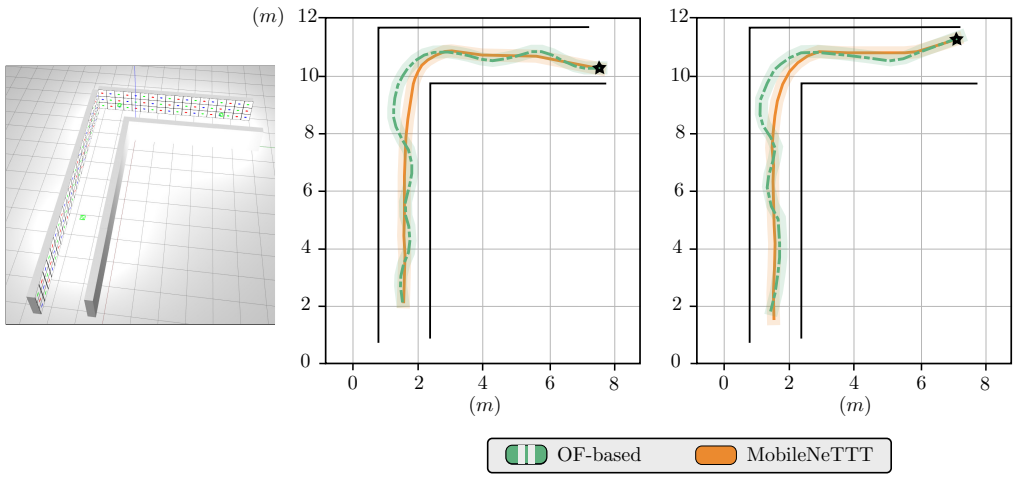


Fig. 6.6 A mobile robot navigates an L-corridor with no obstacles and walls with distinguishable features, starting from the point indicated by the black star. With both TTT estimation methods, the robot is able to reach the end of the corridor starting from different positions, but the OF-based method induces oscillations in the path due to the mandatory use of the *sense-act* cycle.

a constant speed, powered by a Quad-core ARM Cortex-A57 MPCore processor, in a corridor-like environment with static obstacles. Time-to-transit values were estimated using both MobileNeTTT running solely on the CPU, and the Lucas-Kanade optical flow method. The resulting trajectories were then compared. Notably, no fine-tuning was applied to the neural network-based approach. The results demonstrate that MobileNeTTT outperformed the methodology proposed in Chapter 5. Specifically, the robot successfully navigated the entire path using MobileNeTTT, whereas the optical flow-based method struggled due to the sparse feature set in the environment. The limitations of the Lucas-Kanade algorithm led to inaccurate τ estimations, causing the robot to rotate in place instead of progressing along the corridor, as illustrated in Figure 6.8.

6.5 Conclusion

In this chapter, we introduced MobileNeTTT, a lightweight DNN-based framework for real-time time-to-transit estimation in robotic navigation. By directly predicting TTT using a coarse depth estimator, our approach eliminates the need for optical flow computation. Despite its coarse nature, the depth estimation retains the robustness of

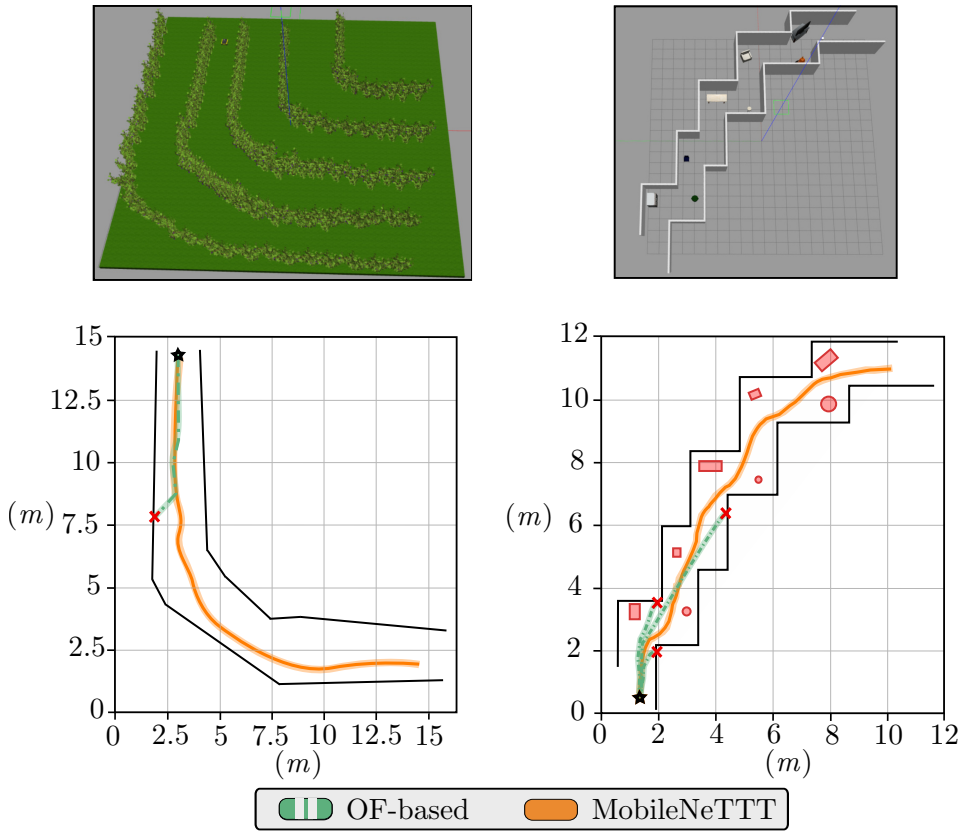


Fig. 6.7 A mobile robot navigates a vineyard environment (on the left) and a corridor with obstacles (marked in red) and featureless walls (on the right). In both scenarios, the starting point is indicated by the black star. Time-to-transit estimation using MobileNeTTT allows the robot to reach the end of the corridor (solid orange line) in both experiments. When using the Optical Flow-based method the robot fails to complete the path, leading to collisions.

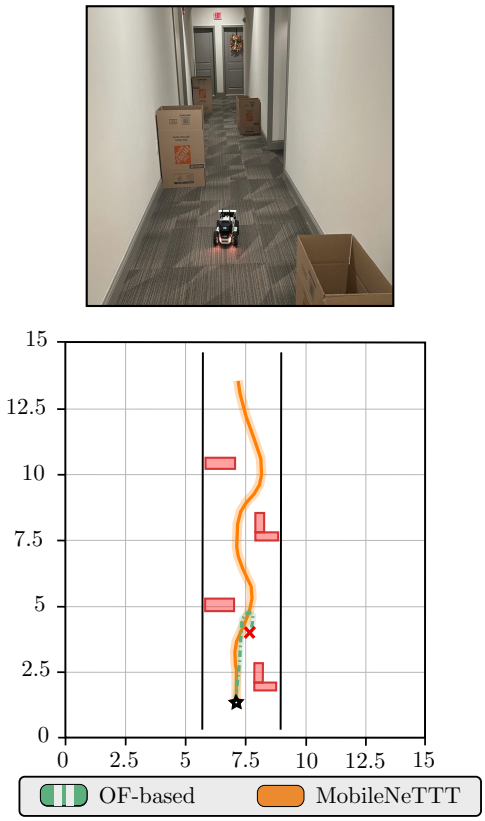


Fig. 6.8 An AgileX Limo rover navigates a hallway with obstacles (marked in red) starting from the point indicated by the black star. When the robot estimates time-to-transit using MobileNeTTT, Tau Balancing is applied correctly, allowing it to reach the end of the corridor (solid orange line). When using the Optical Flow-based method to estimate TTT, the robot fails to estimate the correct τ values, resulting in the robot rotating in place (green line).

depth-based methods, handling robot rotations, dynamic objects, and sparse visual regions. We also showed how this enhanced perception benefits navigation, with improved TTT estimates enabling dynamic adjustment of Regions-of-Interest (ROIs) for more stable control laws.

Part III

Event-based perception

Chapter 7

PEDRo: an Event-based Dataset for Person Detection in Robotics

Event-based cameras offer multiple advantages, making them an excellent tool for tasks that require fast and accurate motion detection and tracking across a wide range of scenarios with varying lighting conditions. In service robotics, one such task is person detection and tracking, which is particularly useful for navigation in crowded spaces or for monitoring applications such as intruder detection.

In the literature, the state-of-the-art approach for person detection, both with standard frame-based and event-based cameras, is the use of deep learning neural networks. However, training these architectures requires large amounts of data, which remains an open challenge when dealing with event-based cameras. This is primarily due to the limited availability of event-based datasets, as most existing datasets have been collected using frame-based cameras, given the still limited usage of event-based sensors. One possible solution is to generate synthetic event-based data by converting frame-based datasets using available image-to-event converters [164, 89]. However, this often leads to suboptimal performance due to differences between simulated and real event data [67, 187].

In recent years, several event-based datasets have been collected. In 2017, Bolten *et al.* [22] introduced a dataset designed for monitoring purposes, consisting of recordings collected in a public urban setting using three fixed CeleX4-DVS [76] sensors. Two years later, Shu *et al.* [140] presented a smaller dataset for pedestrian detection, action recognition, and fall detection. This dataset, collected using a

fixed DAVIS346 event camera [92], includes 4670 labeled individuals. In [206], the authors introduced eTRAM, an event-based dataset for object detection and tracking in traffic monitoring. The recordings were collected in three different environments (i.e., intersections, roadways, and local streets) using a fixed Prophesee EVK4 HD event camera [161]. This dataset contains 2 million instances of 2D bounding boxes for multiple subjects, including cars, pedestrians, and wheelchairs. In the same year, Wang *et al.* [211] introduced FELT-SOT, an event-RGB dataset specifically designed for object tracking. As an evolution of EventVOT [212], it comprises 742 videos of 45 target object classes, including pedestrians, recorded with a DAVIS346 event camera in both indoor and outdoor settings under varying lighting conditions.

However, all the already mentioned datasets and most of the existing event-based ones were collected using fixed sensors, which does not reflect the typical scenario in service robotics, where cameras are mounted on mobile platforms moving through the environment. Two examples of event-based datasets captured with moving sensors, primarily for autonomous driving applications, were released by Prophesee in 2020. The first, the GEN1 Automotive Detection Dataset [46], was collected over 39 hours using a Prophesee Gen1 event-based camera [157] (304×240 resolution) and includes 255 781 manually labeled bounding boxes (228 123 cars and 27 658 pedestrians). The second, the 1 Megapixel Automotive Detection Dataset [155], contains 15 hours of recordings obtained with a high-resolution event sensor (1280×720), providing a total of 25 million automatically labeled bounding boxes for various subjects, including pedestrians, cars, and bicycles. These datasets are not suitable for mobile robotic applications, since they primarily feature pedestrians walking on sidewalks in road environments. Mobile robots must operate in a wider range of scenarios, including both indoor and outdoor environments, under different lighting conditions, and in close proximity to people.

To better address these challenges, we present an event-based dataset specifically designed for person detection and tracking in robotics. Unlike previous datasets, ours focuses on scenarios that more accurately reflect the environments encountered by service robots and also takes into account that the platform is moving during navigation. Notably, at the time of publication in 2023, our dataset represents the largest manually annotated event-based dataset for person detection recorded with a moving camera. The following provides a comprehensive analysis of the presented dataset, including a detailed explanation of the data acquisition process and a statistical evaluation of its contents. Additionally, we demonstrate how this dataset

can be leveraged to improve the performance of event-based people detection neural networks. As a case study, we utilize the well-known YOLOv8[1] [204] architecture to demonstrate its effectiveness.

7.1 Dataset

PEDRo [23] is an event-based dataset specifically designed for person detection and tracking tasks. It completely focuses on human figures and it can be used in various service robotic applications, including intruder monitoring, navigation in crowded environments, and person following. The dataset is composed of 119 recordings of variable duration (an average length of approximately 18 s) acquired with a moving DAVIS346 event-based camera. It presents a huge variety of scenarios, spanning offices and house environments, as well as outdoor environments such as mountains, lakes landscapes, and seafronts. The recordings were conducted under different meteorological conditions such as sunny, rainy, and snowy during the day and night. The dataset was collected in 6 months from September 2022 to February 2023 and the people recorded in PEDRo range from 20 to 70 years of age. While most labeled subjects are walking, the dataset also contains examples of people standing still, sitting, or running. To ensure privacy, all participants provided written informed consent, and only event data and labels were published, excluding any identifiable visual information. An example of the dataset's recordings is illustrated in Figure 7.1.

7.1.1 Dataset collection and labeling

The entire dataset was collected using a DAVIS346 event-based camera. This sensor simultaneously provides both an events stream and greyscale images at a resolution of 346×260 pixels. During the acquisition the camera was in motion, hand-carried, and positioned at different heights, resulting in diverse viewpoints across recordings.

The labeling process was done manually. An initial attempt was made using the well-known YOLOv8x neural network to automatically obtain bounding boxes on the greyscale frames. However, probably due to the low resolution of these frames, the

[1] In this work we used the model available from Ultralytics https://docs.ultralytics.com/it/models/yolov8/

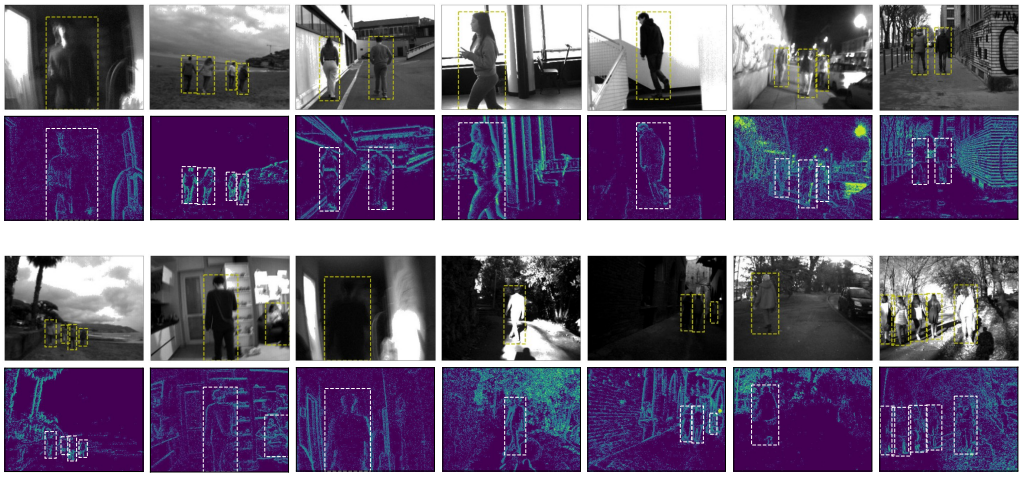


Fig. 7.1 Examples of recording contained in PEDRo. The dataset focuses on people and it presents recordings taken in a large variety of environments with diverse lighting and meteorological conditions.

results were inaccurate and unsuitable as ground-truth labels. Figure 7.2 illustrates an example of wrongly predicted bounding boxes. To better highlight this concept

Fig. 7.2 Examples of wrong predictions from Yolov8x neural network.

and quantify the accuracy of the YOLOv8x generated bounding boxes, we used the intersection over union (IoU) metric. The IoU is a criterion that measures the degree of overlap between two bounding boxes as the ratio of their intersection over their union area. For frames containing multiple objects, the optimal true-predicted pairings were determined by computing the IoU for all possible combinations and selecting the pairs with the highest IoU values. This process was repeated iteratively until no pairs remained. Any unpaired bounding boxes were assigned an IoU of zero. Then the average IoU among all the couples is computed and the results of the

analysis are shown in Figure 7.3. Using the largest pre-trained YOLOv8 model, 22% of the greyscale frames used for labeling failed to reach an average IoU of 0.85 and almost 45% did not reach the 0.90. These findings indicate that automatic labeling is not accurate enough for this particular setting, reinforcing the need for manual annotation.

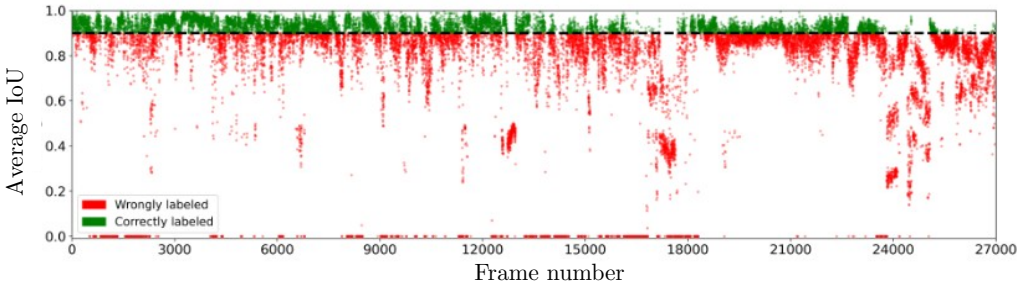


Fig. 7.3 Analysis results of the Yolov8x accuracy in predicting bounding boxes. The average IoU considering the predicted and the manually labeled boxes for each frame is presented.

7.1.2 Dataset format

The dataset is composed of 119 recordings, from which a total of 27 000 samples were extracted. Each sample corresponds to the stream of events collected during a 40 ms time interval preceding the timestamp of the frame used to obtain the corresponding labels. This time interval is determined by the acquisition rate of the grey-scale images used for the manual labeling process (25 fps). To prevent overlap between recordings, each one is assigned entirely to a single dataset split: train, validation and test set. Each sample is coupled with the corresponding label that containes the bounding boxes present in that sample. In total, the dataset includes 43 259 bounding boxes, of which 34 243 (79.2%) for training, 4372 (10.1%) for validation and 4179 (9.7%) for testing.

The events in a sample (both positive and negative polarity) are stored in a numpy structure while the corresponding labels are stored in Pascal VOC format [56]. Sample and label are matched by looking at the file name (e.g. file `frame0000001.npy` is associated with `frame0000001.xml`). Moreover, a text file is also provided listing the recordings along with the names of the samples they contain.

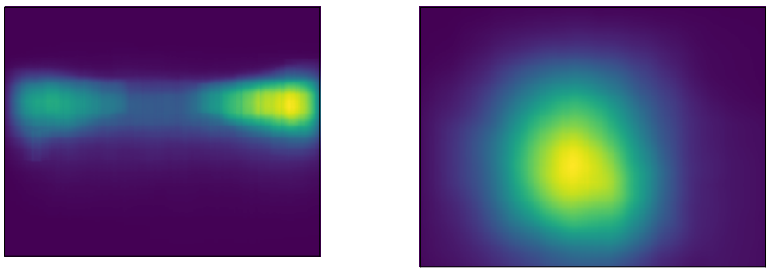


Fig. 7.4 On the left, the heatmap displaying labeled bounding boxes for pedestrians of the GEN1 dataset, while on the right, the heatmap for people in our dataset. Bounding boxes contained in PEDRo cover different areas compared to the ones in the GEN1 dataset. Picture taken from [23].

7.1.3 Dataset analysis and statistics

The PEDRo dataset is specifically designed for service robotic applications and exhibits key differences compared to other event-based datasets. In particular, we compare it with the GEN1 Automotive Detection dataset from Prophesee, which, among the event-based datasets with a moving camera, is the most relevant for our analysis. We selected GEN1 over the 1 Megapixel Automotive dataset because it includes hand-labeled annotations and has a spatial resolution similar to PEDRo. The differences we want to highlight are mainly two and are the following.

Bounding Box Distribution One fundamental difference lies in the distribution of bounding boxes. To analyze this, we count the number of bounding boxes covering each pixel across the entire dataset. By normalizing these counts (dividing by the total number of bounding boxes), we generate heatmaps with values ranging from 0 to 1, as shown in Figure 7.4. These heatmaps reveal that in GEN1, detected people are primarily pedestrians walking on sidewalks, with bounding boxes concentrated along the image margins. In contrast, PEDRo exhibits a higher concentration of bounding boxes in the center of the image, where subjects are closer to the camera.

Bounding Box Size Distribution Another important difference is the size distribution of bounding boxes, illustrated in Figure 7.5. This figure represents the distribution of bounding box diagonals for GEN1 and PEDRo. In the GEN1 automotive dataset, most bounding boxes are small, indicating that people are typically far from the camera. Conversely, PEDRo features a wider range of bounding box sizes, reflecting a greater variety of distances between subjects and the acquisition sensor.

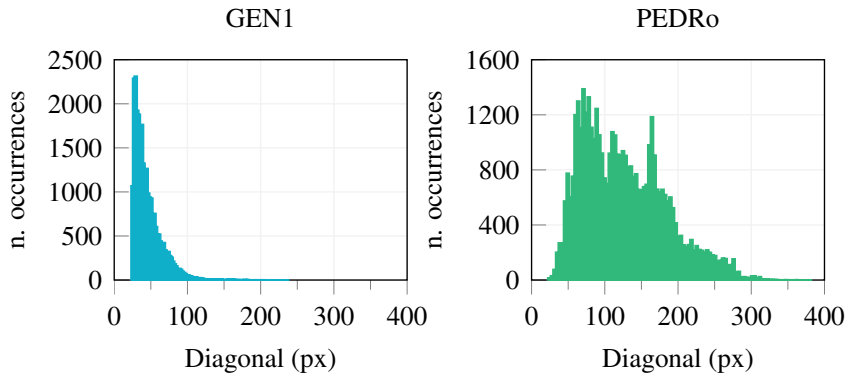


Fig. 7.5 On the left, the distribution of the length (in pixels) of the diagonals of bounding boxes for pedestrians in the GEN1 dataset, while on the right, the diagonals for people in our dataset. PEDRo features a larger number of bounding boxes with high diagonals. Picture taken from [23].

This simple analysis shows that the PEDRo dataset features properties that are missing in an automotive-focused dataset like the GEN1, making it a valuable addition for improving object detection performance, as it will be further discussed in the next section.

7.2 Experimental Results

In this section, we demonstrate how our newly introduced dataset, PEDRo could enhance the performance of a well-known object detection neural network such as YOLOv8x. Specifically, we evaluate the network's performance on three configurations: training on PEDRo alone, on the GEN1 dataset alone, and on a combination of both datasets.

Training Setup We fine-tuned a pre-trained YOLOv8 model for a total of five epochs using the Stochastic Gradient Descent (SGD) optimizer, with a learning rate of 0.01 and a batch size of 64. The input to the neural network consists of the widely used Surface of Active Events (SAE) [140, 147], which organizes event-based data into an image-like representation. Each pixel in the SAE is assigned a value based on the following formula:

$$SAE(x_i, y_i, t) = \frac{t - t_0}{T} \tag{7.1}$$

Test \ Train	GEN1	PEDRo	GEN1 + PEDRo
GEN1	0.716|0.341	0.487|0.205	0.718|0.342
PEDRo	0.437|0.237	0.895|0.586	0.794|0.504

Table 7.1 Results expressed as $mAP_{50}|mAP_{50:95}$ with YOLOv8x trained on different training sets and evaluated on various test sets.

where t and t_0 are the times at which the last and the first events are acquired in the (x_i, y_i) pixel location, while T is the time interval in which we consider the events to construct the SAE. This value is proportional to the timestamp of the most recent event, meaning that more recent events have higher values. Since events can be either positive or negative, we construct two separate SAE maps and stack them to form a two-channel tensor, which serves as input for YOLOv8. To ensure compatibility with YOLOv8x, we normalize SAE values between 0 and 255 (as it is standard for image processing). Additionally, since the pre-trained YOLOv8x model expects three-channel images, we removed the third input channel from the network. We evaluate the network's performance using Mean Average Precision (mAP) [126], a common metric for object detection accuracy [88, 155]. Specifically, we report results in terms of mAP_{50} (mAP at an IoU threshold of 50%) and $mAP_{50:95}$ (mAP averaged across IoU thresholds from 50% to 95%), evaluated on different test sets.

Results The results are presented in Table 7.1. We observe that the two datasets are uncorrelated since the mAP values are quite low when YOLOv8x is trained on one dataset and tested on the other one. GEN1, an automotive dataset focused on pedestrians, is not well-suited for general person detection, as it primarily captures individuals in road environments. PEDRo, on the other hand, enables the training of models capable of detecting people in closer proximity to the camera and in diverse environments, making it highly relevant for applications in robotics and surveillance. However, when both PEDRo and GEN1 are used for training, there is no loss in pedestrian detection performance. Instead, the model gains the ability to detect additional targets from PEDRo, demonstrating the dataset's potential in expanding YOLOv8's recognition capabilities.

7.3 Conclusions

In this work, we introduced PEDRo, an event-based dataset entirely collected with a moving sensor and specifically designed for person detection in service robotics applications. PEDRo is the largest manually annotated event-based dataset of this type. It features a diverse range of recording environments and characteristics, including varying lighting conditions, multiple scenarios, and different meteorological situations. The results of the experiments performed using the well-established YOLOv8 architecture for object detection demonstrate that the proposed dataset is a valuable addition to other existing event-based datasets to enhance detection performance. Therefore, it could pave the way for novel research avenues in the field of event-based vision.

Chapter 8

From Event Representations to Deep Learning Integration

In recent years, Deep Neural Networks (DNNs) have become the standard for tackling various computer vision tasks. Among these models, non-recurrent architectures, including convolutional [110, 107, 83, 86, 166] and transformer-based networks [54, 38, 137], have demonstrated remarkable scalability and performance in solving tasks ranging from simple image classification [110, 106, 48] to object detection [246], usually in dynamic scenarios where things changes quickly, such as egocentric vision [13] and autonomous driving [213].

Event-based cameras, or Dynamic Vision Sensors (DVS) [124, 117, 64], represents an interesting alternative to standard frame-based sensors, especially in scenarios where there is the need for high acquisition rates but with low-power consumptions and the reduction of redundant information, making them well-suited for a large variety of applications including surveillance [168, 169], robotics [21, 58], and biomedical science [14, 75, 156]. However, despite the multiple advantages, event-based cameras generate data that is not easily interpretable, requiring specialized representations to be effectively processed by computer vision models.

To bridge this gap, researchers have explored various strategies for transforming event streams into structured formats that contain meaningful information and are compatible with conventional deep-learning models. As discussed in Section 3.4, common approaches include time surfaces (TS) [109] and Surface of Active Events (SAE) [140] which collapse events within a time window into an image-like matrix,

and voxel grids [242] or TORE volumes [12], which organize events into three-dimensional spatial-temporal bins. While these representations facilitate the use of standard DNNs, they aggregate events over fixed time intervals, meaning that it could happen that not every piece of input contains enough information for the DNN models to generate accurate outputs. Expanding the time window could provide more contextual information, but it would also diminish one of the advantages of event-based cameras: their ability to capture fine-grained temporal dynamics with a very high temporal resolution. Thus, while event-based encoding is a promising approach for high-speed computer vision tasks, striking a balance between temporal resolution and data richness remains a key challenge in leveraging these sensors effectively within deep learning frameworks. Possible solutions to this problem are the usage of custom recurrent neural network models [40] which have the drawback of being typically more complex to train compared to non-recurrent ones or to leverage a hybrid RGB-DVS framework [7].

In the following chapters, I'm going to present two consecutive studies that utilize event-based camera outputs for eye-tracking and object classification/detection tasks. In the first study, we explore how event streams can be effectively leveraged for eye-tracking [24]. In the second, we extend this approach to object classification and detection. Both studies propose methods that transform event streams into meaningful representations suitable for widely used convolutional neural networks, eliminating the need for complex and resource-intensive neural architectures.

Chapter 9

Memory in Motion: Exploring Leaky Integration of Time Surfaces for Event-based Eye-tracking

The rapid development of Augmented Reality (AR) and Virtual Reality (VR) technologies in various industries has highlighted the importance of precise and accurate eye-tracking methods. These systems, along with all the correlated key aspects such as pupil position estimation, pupil shape detection, pupil tracking and gaze esimation, enable a wide range of applications, including human-machine interaction [2, 172], driver health status monitoring [98, 228], and the diagnosis of conditions such as Parkinson's or Alzheimer's diseases [159, 55, 115].

Two critical requirements that are usually important when implementing algorithms on mobile platforms are power consumption and the ability to operate in resource-constrained settings. Additionally, since eye movements occur within a fraction of milliseconds, the system must achieve a high sampling rate. Therefore, the final solution should be lightweight enough to integrate seamlessly into a headset while maintaining a high sampling rate. Dynamic Vision Sensor (DVS), also known as event-based cameras [105, 124, 64], represent an interesting option to meet these constraints in eye-tracking systems. Previous methods have attempted to leverage event streams to perform eye-tracking by employing custom recurrent neural networks [40] which are usually difficult to train compared to non-recurrent

counterparts [15]. Alternatively, there exists also solution [7] that has integrated a hybrid RGB-events system to obtain eye-tracking and gaze estimation.

In this work, we propose to use standard convolutional neural networks (CNNs) to estimate the pupil position starting from event-based data. Our approach introduces a pre-processing pipeline, called *Memory in Motion (MeMo)*, utilizing *memory channels* to structure event data into image-like representations. Unlike widely used techniques such as Surface of Active Events (SAE) and Time Surfaces (TS), our method ensures the generation of meaningful images even in scenarios where a low number of events would otherwise result in sparse or nearly empty structures.

9.1 Methodology

The method we propose is based on memory channels and employs a pre-processing pipeline that integrates events over time to obtain an enriched input data representation. As the base event representation, we utilize the time surface, computed using the *Tonic* framework [116]. Given that event polarity is dual (positive or negative), we generate two separate time surfaces for each predefined time interval Δ_t. These are denoted as $\boldsymbol{S}^n_{\mathrm{p}}$ for positive events and $\boldsymbol{S}^n_{\mathrm{n}}$ for negative events, where n represents discrete time steps at times $t = \Delta_t, 2\Delta_t, 3\Delta_t, \ldots$ corresponding to the associated time surfaces. A time surface can be computed using the following formula:

$$\mathbf{S}^n_p(x,y) = e^{-\frac{n\Delta_t - \mathcal{T}^n_p(x,y)}{\tau}} \tag{9.1}$$

which defines the pixel value of the time surface at position (x,y) at time $t = n\Delta_t$ with polarity $p \in \{\mathrm{p},\mathrm{n}\}$. Here $\mathcal{T}^n_p(x,y)$ is given by:

$$\mathcal{T}^n_p(x,y) = \max_{\boldsymbol{e} \in \mathcal{E}^n_p(x,y)} t \in \boldsymbol{e} \tag{9.2}$$

where $\boldsymbol{e} = (x,y,t,p)$ represents an event from the set $\mathcal{E}^n_p(x,y)$, which includes all the events occurring at pixel (x,y) within the time range $t \in [n\Delta_t - \Delta_t, n\Delta_t]$. In this work, we use a time constant $\tau = 7 \times 10^{-1}$ s.

To enhance temporal information retention, the time surfaces are integrated over time into multiple memory channels, defined as

$$\boldsymbol{M}_i^n = \left[k_i \boldsymbol{M}_i^{n-1} + \frac{\boldsymbol{S}_\mathrm{p}^n + \boldsymbol{S}_\mathrm{n}^n}{2} \right]_0^1 \tag{9.3}$$

where $\boldsymbol{M}_i^n$ represents the i-th memory channel at time-step n, $k_i \in (0,1)$ is the leakage/forgetting factor of the i-th channel. The operator $[\cdot]_0^1$ saturates the argument between 0 and 1. $\boldsymbol{M}_i^0$ are defined as all-zero matrices. The forgetting factor k_i determines how long the information is maintained, values close to 1 preserve information for a longer period of time, while lower values prioritize recent events by quickly discarding older data. Using multiple values of k_i generates multiple memory channels, enabling a combination of both short-term and long-term memory. These channels are then stacked to form a multi-channel tensor, serving as input to the estimation model.

The proposed input pipeline effectively shifts the time dependency of the model entirely to the input pipeline, simplifying optimization by ensuring that both forward and backward passes depend only on the current time step. Additionally, this approach allows Δ_t, the interval between discrete time steps n and $n+1$, to be taken as small as desired. This would not be possible by simply using time surfaces as inputs, since they would not contain enough information for the vision model to produce meaningful output, limiting the frequency at which the output could be generated.

The described methodology is summarized in Figure 9.1, and it serves as a pre-processing block that can generate input suitable for various state-of-the-art, non-recurrent, convolutional neural networks to solve the pupil estimation task.

9.2 Datasets and Metrics

In this section, we are going to present the dataset we used to train and test the various neural networks for pupil position estimation, but we also give a brief overview of the metrics we employed to evaluate the performance of the models.

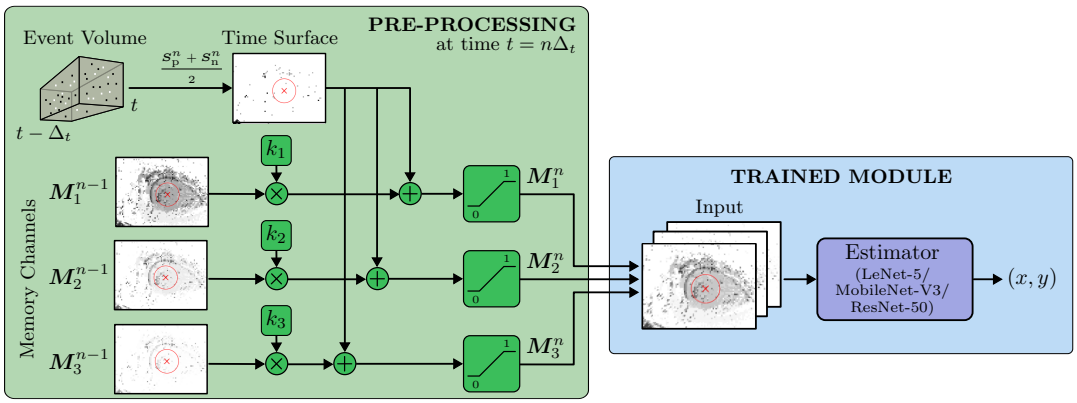


Fig. 9.1 Schematic of the method used to solve the event-based eye-tracking task. Starting from a volume of events collected in the time range $(t, t+\Delta_t)$, a time surface is generated and then enriched with older information within memory channels built starting from events that were collected in the time range $(0, t)$. From here, the input of the network is structured as the concatenation of three enriched time surfaces while the output is the pupil's position where $x, y \in [0, 1]$. In this work $k_1 = 0.8$, $k_2 = 0.6$ and $k_3 = 0.4$. Picture taken from [24].

9.2.1 Datasets

We decided to use two dataset specifically designed for eye-tracking purposes, the 3ET Dataset [40] and the 3ET+ Challenge Dataset [216].

The Efficient Event-based Eye-tracking (3ET) Dataset is derived from the RGB LPW dataset [198]. Using the V2E [89] event simulator, the original RGB images have been converted into event streams, resulting in a dataset containing recordings from 22 different subjects. The original dataset counts 62 event-based videos, but following the approach presented in [40], the authors discarded a significant portion of them due to long periods of time without recorded events. The refined dataset includes only 16 videos for training and 2 for validation. Each recording lasts approximatly 20 s, with target labels provided at a frequency of 100 Hz, representing the (x, y) coordinates of the center of the pupil. The resolution of the videos is 640×480 pixels. Since no separate test is available, we evaluate model accuracy on the validation set and additionally use two recordings from the training set as validation data.

The Event-based Eye-Tracking (3ET+) Dataset has been presented for the CVPR 2024 EET Challenge [216]. It counts 52 recordings entirely acquired with an event-based camera, specifically a DVXplorer Mini [93], at a resolution of 640×480 pixels. The involved subjects are 13, each of them having 2 to 6 recording session

performing different activities. These are divided in 5 classes, including random, saccade movements, read text, smooth pursuit, and blinking. The labels are the (x, y) position of the center of the pupil and a binary value that indicates whether there was an eye blink or not. The targets are provided at a frequency of 100 Hz for training and 20 Hz for testing. The dataset is split into 59% of the recordings for training, 18% for validation, and 23% for testing.

9.2.2 Metrics

The metrics we used to evaluate the performance of the different models in the eye-tracking, specifically the pupil estimation task, are the following:

- **Mean Euclidean Distance**: it is the distance between the estimated pupil center and the groud-truth one. Obviously the smaller the value the better the estimation.
- **P10 Accuracy**: it is an extension of the previous metric. Specifically it indicates whether the estimated position is below a radius of ten pixels from the ground-truth pupil position.

9.3 Training Neural Network Estimators

In this work, we employ various state-of-the-art convolutional neural networks to estimate pupil position. Our goal is to demonstrate that our pre-processing pipeline can be seamlessly integrated with a wide range of neural networks without introducing significant computational overhead or requiring modifications to the networks themselves. The models used in our study are the following:

- **LeNet-5 [110]**: it is a simple deep neural network consisting of two convolutional layers, followed by three fully connected layers with 200, 84 and 2 neurons, respectively. The total number of parameter of the model is 670 000. We selected LeNet-5 for its simplicity and as a baseline for comparison.
- **MobileNet-V3 [86]**: a well-known lightweight convolutional neural network designed for efficiency. We experiment with both the small variant, which has

approximately 2.5 million parameters, and the large variant, with around 5 million parameters. Both models are pre-trained on ImageNet-1K [48]. The MobileNet family is widely used in industry and is known for delivering high performance while being computationally efficient, making it well-suited for CPU-based edge devices.

- **ResNet-50 [83]**: the largest model used in our study, with approximately 25 million parameters. Like MobileNet-V3, we use a pre-trained version on ImageNet-1K.

Each of these networks has been trained for 200 epochs using a batch size of 32, the Adam optimizer [102], and an initial learning rate of 2.8×10^{-4}. The loss function employed is Mean Squared Error (MSE), and each model receives input downsampled to 60×80 resolution.

For training and validation, the multiple recordings and event streams have been structured as follows: each event stream is divided into small chunks defined by a time interval Δ_t. From each chunk, a time surface is extracted, and subsequences are generated. During training, these subsequences of consecutive time surfaces are fed in random order. However, for validation and testing, a single chronologically ordered sequence is used to mimic the behavior of a real eye-tracking system. Figure 9.2 presents graphically how the different set have been divided.

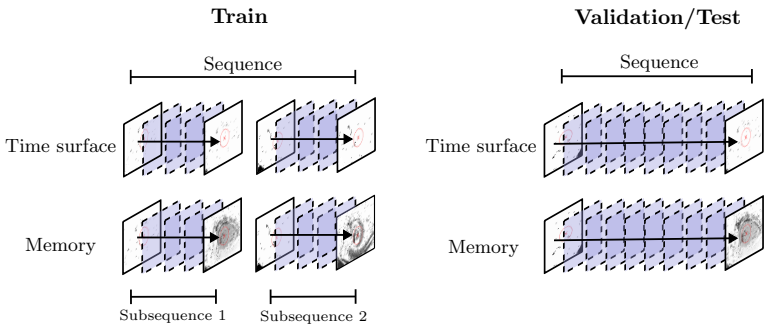


Fig. 9.2 For training, original sequences of time surfaces are divided into subsequences which are then fed to the DNN in a random order. For validation/test data, the original sequence is not subdivided to emulate the behavior of the final system.

To avoid overfitting and improve robustness, data augmentation has been applied to the training dataset. This includes randomly flipping the time surfaces horizontally

and vertically, as well as shifting them. Additionally, Gaussian noise is added to 10% of the input samples. Figure 9.3 shows examples of the data augmentation performed.

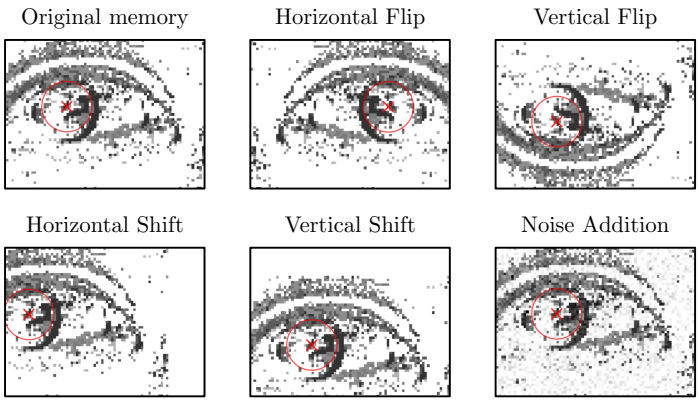


Fig. 9.3 The original memory of events and the obtained memory after the application of an augmentation technique.

9.4 Results

The results obtained using the proposed approach based on *Memory Channels* are reported, firstly in term of accuracy and then by looking also at the computational efficiency of each neural estimator evaluated.

A. Accuracy

To assess the effectiveness of our input pipeline compared to using simple time surfaces, we refer to Table 9.1. This table reports the performance in terms of mean Euclidean distance and P10 accuracy for all neural estimators on both the 3ET and 3ET+ datasets, with Δ_t =50 ms. When using simple time surfaces, the input tensor consists of three channels, the positive, the negative, and the average of the two time surfaces. On the contrary, our memory channels approach constructs the input using three different forgetting factors: $k_1 = 0.8$, $k_2 = 0.6$ and $k_3 = 0.4$. The results consistently demonstrate a significant improvement when using memory channels over simple time surfaces.

	3ET				**3ET+ (or EET)**			
	P10 accuracy		Mean Euclidean distance		P10 accuracy		Mean Euclidean distance	
	TS only	**Ours**	TS only	**Ours**	TS only	**Ours**	TS only	**Ours**
LeNet-5	81.1%	91.9%	7.2	4.9	89.8%	95.0%	5.3	4.3
MobileNet-V3S	84.9%	92.3%	6.4	4.6	93.8%	97.3%	4.3	3.5
MobileNet-V3L	86.5%	94.6%	5.6	4.6	94.9%	99.1%	3.7	3.2
ResNet-50	90.4%	96.9%	5.1	3.9	95.1%	98.9%	3.4	2.4

Table 9.1 P10 accuracy and mean Euclidean distance of different configurations of the eye-tracking system, with Δ_t = 50 ms. We use as input either the positive and negative time surfaces or three memory channels.

	$\Delta_t = 20$ ms		$\Delta_t = 40$ ms	
	TS only	**Ours**	TS only	**Ours**
LeNet-5	73.8%	**85.5%**	78.3%	**86.3%**
MobileNet-V3S	80.9%	**90.6%**	84.1%	**92.4%**
MobileNet-V3L	85.4%	**94.4%**	93.2%	**98.6%**
ResNet-50	84.1%	**93.1%**	94.8%	**98.8%**

Table 9.2 P10 accuracy of the eye-tracking system employing various DNNs on the EET dataset, both with the use of time surfaces only or incorporating the input pipeline based on three memory channels with different values of Δ_t.

Table 9.2 presents results obtained on the 3ET+ Dataset with different Δ_t, specifically 20 ms and 40 ms. Since labels are not available in the original test set for these time intervals, we use the validation set for evaluation and we take two recordings from the training set as validation. The results further confirm that our approach enhances estimation performance, particularly as the time interval decreases. Shorter time intervals reduce the amount of information contained in a simple time surface, but by integrating memory channels over time, this limitation is effectively mitigated.

Finally, we conducted a study to analyze the impact of the number of memory channels on performance. Using the LeNet-5 estimator on the 3ET+ dataset, we evaluated performance while varying the number of memory channels. The results are presented in Table 9.3 and they are obtained with the same k_1, k_2 and k_3 used for Table 9.1 and by adding a fourth leakage factor $k_4 = 0.2$. The findings indicate that the best performance is achieved when multiple memory channels with different forgetting factors are used, allowing the estimator to distinguish recent information

# of memory channels	P10 accuracy	Mean Euclidean distance
1	92.7%	4.6
2	93.4%	4.5
3	95.0%	4.3
4	95.1%	4.4

Table 9.3 Performance of the eye-tracking system employing LeNet-5 on the EET dataset, using as input an increasing number of memory channels with $\Delta_t = 50\,\text{ms}$.

	FLOPS	**Latency**
Memory update	24.00 k	0.23 ms (CPU)
LeNet-5	1.64 M	1.40 ms (GPU)
MobileNet-V3S	0.12 G	20.04 ms (GPU)
MobileNet-V3L	0.23 G	24.26 ms (GPU)
ResNet-50	4.00 G	23.59 ms (GPU)

Table 9.4 Number of FLOPS and computational time required by the memory channels update compared to the complexity and the inference time of the estimators. Results are computed using the Jetson Orin Nano single board computer system

from older data. However, even with a single memory channel, our approach outperforms the use of simple time surfaces, further validating its effectiveness.

B. Computational Efficiency

One key advantage of the proposed methodology is the introduction of a pre-processing pipeline that enriches event-based input information and can be applied to non-recurrent neural networks. However, this pipeline introduces some computational overhead to the deployed system. To quantify this, we evaluate the number of Floating Point Operations (FLOPs) required for memory channel updates, along with the inference time for each neural estimator. Table 9.4 shows the results, together with the latency measured on a Jetson Orin Nano single computer board. Notably, the inference time remains unaffected by the number of channels used in the memory pre-processing pipeline, as the Jetson's GPU processes them in parallel. Moreover, the additional latency introduced by memory updates is minimal compared to the total inference time. For instance, in the case of MobileNet-V3, the memory channel update accounts for only 0.9% of the network's inference time.

9.5 Conclusions

In this work, we introduced an input pre-processing pipeline called *MeMo* (Memory in Motion), designed to enhance non-recurrent neural estimators for event-based eye-tracking tasks. MeMo leverages multiple memory channels, each integrating a leaky multiple time surface of events over time. These channels are constructed using different forgetting constants, allowing the model to capture temporal dynamics at varying scales. Through a series of experiments on both synthetic and real event-based datasets, we demonstrate that our approach significantly improves performance on the target task.

Chapter 10

MESA: A Dynamical Attention-based Pre-processing Pipeline for Event-based Computer Vision Tasks

The methodology proposed in Chapter 9 has been shown to be effective, but it presents a significant limitation. The pre-processing pipeline relies on static forgetting factors, which are predefined and fixed throughout the entire operation. This approach results in a suboptimal solution, as the scaling factor remains constant across both time and space. Consequently, regions of the image with a high density of events are treated the same as areas with fewer events, which is far from ideal.

To address these issues, this work introduces an evolved pre-processing pipeline, shown in Figure 10.1, that retains the advantages of the original method: 1) its applicability to a wide range of convolutional neural networks and transformers without the need to modify their internal architectures, 2) the possibility of selecting very short temporal windows, greatly enhancing the performance of state-of-the-art estimators with minimal computational overhead and 3) the chance to avoid the use of recurrent neural networks. The new methodology incorporates pixel-wise adaptive forgetting factors, which are dynamically generated in real-time using a spatial attention module. We call this enhanced approach *Memory of Events through Spatial Attention (MESA)* [20]. Several tests have been conducted on various computer vision tasks, such as eye-tracking estimation, object classification and object detection, and with different event representations. The proposed pre-processing pipeline was

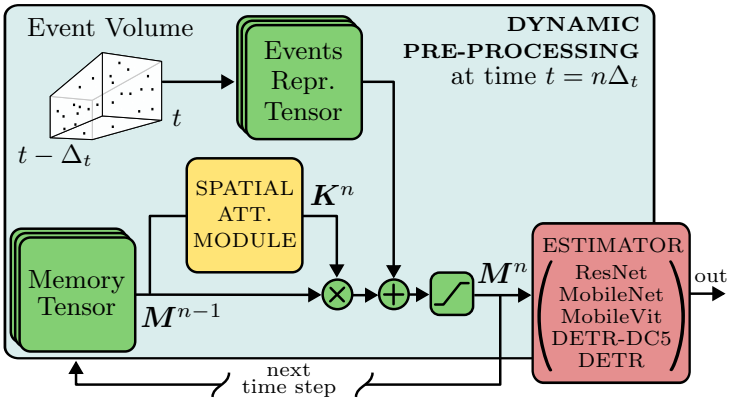


Fig. 10.1 Pre-processing pipeline. A tensor-like representation of an event stream, in the time range $(t, t+\Delta_t)$, is accumulated (with 0 to 1 clipping) over time and stored in a memory tensor $\boldsymbol{M}^n$. At each time step n, the values corresponding to each pixel of the memory tensor $\boldsymbol{M}^{n-1}$ are dampened with varying strength by a forgetting matrix $\boldsymbol{K}^n$, dynamically generated by the spatial attention module.

integrated into state-of-the-art neural networks, and the results demonstrate that performance is significantly improved compared to using single time surfaces alone, but also by using the Memory in Motion pipeline presented in the previous chapter.

10.1 Methodology

As already discussed, event-based sensors generate a stream of asynchronous events that encode pixel-local changes in scene brightness. Each event is characterized by its pixel position, time of occurrence, and polarity. In this work, events are organized using two different approaches: time surfaces (ts) and voxel grids (vg), both implemented with the *Tonic* [116] framework. Similar to the pipeline presented in Chapter 9, we define a time interval Δ_t and we collect events occurring within this interval to construct the event representations. A new representation is generated at each time step $n = 1, 2, 3, \ldots$, and encoded as a tensor $\boldsymbol{R}^n \in \mathbb{R}^{H \times W \times C}$, where H and W are the height and width of the DVS sensor, and C represents the number of channels, which varies based on the representation type:

- **Time surfaces**

Time surfaces are 2D image-like structures, defined as:

$$\boldsymbol{R}^n_{\mathrm{ts},p}(x,y) = \exp\left[-\frac{1}{\tau}\left(n\Delta_t - \max_{\boldsymbol{e}\in\mathcal{E}^n_p(x,y)} t_{\boldsymbol{e}}\right)\right] \tag{10.1}$$

where $\mathcal{E}^n_p(x,y)$ is the set of the events with polarity p located at pixel (x,y) within the time range $t \in ((n-1)\Delta_t, n\Delta_t]$, $t_{\boldsymbol{e}}$ is the timestamp of event $\boldsymbol{e}$ and τ is a time constant.

- **Voxel grid**

  Voxel grids extend event representation to 3D by discretizing time into B bins of size $\delta_t = \Delta_t/(B-1)$. The structure is defined as:

$$\boldsymbol{R}^n_{\mathrm{vg}}(x,y,z) = \sum_{\boldsymbol{e}\in\mathcal{E}^n(x,y)} p_{\boldsymbol{e}} \max\left(0, 1-|z\delta_t - t^*_{\boldsymbol{e}}|\right) \tag{10.2}$$

  where $t^*_{\boldsymbol{e}} = \left[t_{\boldsymbol{e}} - (n-1)\Delta_t\right]/\delta_t$ is the time of event $\boldsymbol{e}$ shifted to the event representation range and normalized to the bin size δ_t and $p_{\boldsymbol{e}}$ is its polarity. The max operator in Equation 10.2 functions as bilinear sampling kernel defined in [94], giving greater influence to events closer to the center of a temporal bin. A single event may influence two adjacent bins simultaneously.

At each time step n, the time surface or voxel grid representations are accumulated into a *memory tensor* $\boldsymbol{M}^n \in \mathbb{R}^{H\times W\times C}$ defined as

$$\boldsymbol{M}^n = \left[\boldsymbol{K}^n \odot \boldsymbol{M}^{n-1} + \boldsymbol{R}^n\right]^1_0 \tag{10.3}$$

where $\boldsymbol{K}^n \in (0,1)^{H\times W}$ is the *forgetting matrix* with the same spatial height and width of $\boldsymbol{M}^n$, $\odot$ is an element-wise multiplication operator that scales all the values corresponding to a spatial position in $\boldsymbol{M}^{n-1}$ by the corresponding value in matrix $\boldsymbol{K}^n$ and operator $[\cdot]^1_0$ clips the argument between 0 and 1. The processing starts at $n=1$, with $\boldsymbol{M}^0$ initialized as an all-zero tensor. Fig. 10.1 illustrates the proposed pre-processing pipeline.

The forgetting matrix $\boldsymbol{K}^n$, with values in the range $(0,1)$, is dynamically generated at each time step using a *spatial attention* module [224]. This module selects what parts of the memory tensor should be quickly "forgotten" (values close to 0) and which should be "retained" (scaling values close to 1). This is done by condens-

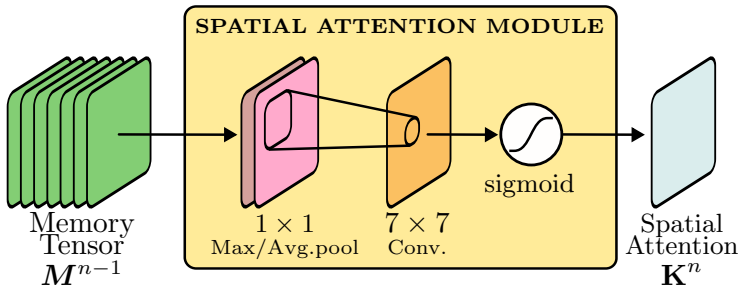


Fig. 10.2 Spatial attention module structure: a 2-channel tensor is generated by taking the average and the maximum of the memory tensor channels; then a 7×7 convolution filter is applied and a sigmoid function is applied to get the forgetting matrix $\boldsymbol{K}^n$ with values between 0 and 1.

ing all the channels of the memory tensor in 2 channels[1] and by applying a 7×7 convolutional filter followed by a sigmoid activation function[2]. This kind of approach selectively retains important pixel data in the memory tensor, optimizing the performance on the task. Figure 10.2 shows the spatial attention module structure.

This event-processing input pipeline allows the definition of arbitrarily small time intervals, increasing the throughput of the algorithm without significantly affecting the performance. On the contrary, using individual time surfaces or voxel grid representations alone would result in low information density for small Δ_t.

10.2 Datasets and Estimators

To demonstrate the versatility of our proposed methodology and its applicability across various non-recurrent neural network architectures and multiple tasks, we evaluate it using several well-known computer vision datasets. Specifically, we evaluate the performance on the eye-tracking task but we also explore tasks on object detection and object classification, leveraging the following datasets:

- **3ET+ [216]**: an event-based eye-tracking dataset. It comprises 52 videos entirely captured by means of an event-based camera from 13 different subjects performing different eyes activities.

[1] For each pixel we encode the channel-wise maximum and average value.
[2] We apply zero-padding to keep the spatial dimensions unchanged.

- **N-MNIST [151]**: a neuromorphic adaptation of the widely used MNIST [110] dataset, serving as a fundamental benchmark for object classification. It consists of 70 000 recordings, each representing a handwritten digit (0–9) captured as an event stream.
- **CIFAR-10 DVS [119]**: the neuromorphic version of the CIFAR-10 dataset, comprising 10 000 event streams across 10 distinct classes. These classes represent objects from different categories, such as animals and vehicles.
- **N-CALTECH101 [151]**: an event-based adaptation of the CALTECH101 [59] dataset, featuring 101 classes with an average of 50 event streams per class.
- **PEDRo [23]**: introduced in Chapter 8, this dataset is specifically designed for pedestrian detection. It contains 43 259 bounding boxes across 119 recordings, collected using a moving DVS sensor in dynamic environments.

To solve the aforementioned tasks, we fine-tuned and tested several neural network models in conjunction with our proposed pre-processing pipeline. The models employed include:

Eye Tracking and Object Classification

- **ResNet-18 [83]**: A well-known residual convolutional neural network that employs skip connections to mitigate the vanishing gradient problem. It is the smallest model in the ResNet family.
- **MobileNet-v3s [86]**: A lightweight convolutional neural network optimized for mobile devices, representing the smallest variant of the MobileNet architectures.
- **MobileViT-v2s [137]**: A transformer-based extension of MobileNet that retains computational efficiency while improving performance through the incorporation of transformer blocks.

Object Detection

- **DETR [38]**: A state-of-the-art object detection architecture that eliminates the need for region proposals and anchor boxes by directly predicting bounding boxes and class labels. We fine-tuned the pre-trained model available in the Hugging Face library.

- **DETR-DC5 [38]**: A variant of DETR designed to enhance performance by incorporating dilated convolutions into the network's backbone. We fine-tuned the pre-trained model available in the Hugging Face library.

This comprehensive evaluation highlights the adaptability of our methodology across diverse datasets, tasks, and network architectures.

10.3 Training and Results

A. Training

The training procedure follows a similar approach to the one described for the previous version of the pre-processing pipeline presented in Chapter 9, which was based on Memory Channels with constant forgetting factors. In this setup, event stream recordings are divided into small chunks of duration Δ_t to generate event representations, such as time surfaces or voxel grid tensors, denoted as $\boldsymbol{R}^n$. This process produces a sequence of tensors, which, for the training set, are split into sub-sequences, batched, and randomly fed into the selected pre-processing/estimator pair for optimization. For all datasets, training sub-sequences consist of 20 samples, each corresponding to a time interval of $\Delta_t = 5\,\text{ms}$, with an overlap of 3 samples. In contrast, validation and test sequences remain unaltered to simulate real-time system behavior. Each sample may consist of time surfaces with a time decay of $\tau = 3\,\text{ms}$ or voxel grids with $B = 3$.

All neural estimators were trained for 20 epochs, using the AdamW optimizer [128] with an initial learning rate of 1×10^{-4} and a weight decay of 1×10^{-4}. No data augmentation was applied, as the defined time interval is sufficiently small to ensure an adequate number of samples. The batch size varies depending on the task to be solved. For eye-tracking, it is set to 32, for object classification to 512, while for object detection to 258.

B. Results

The results of various experiments evaluating the pre-processing pipeline in combination with different estimators are presented in Table 10.1. These experiments assess the accuracy on the test set of different datasets for two event-based representations: the time surface and the voxel grid. The results compare three approaches:

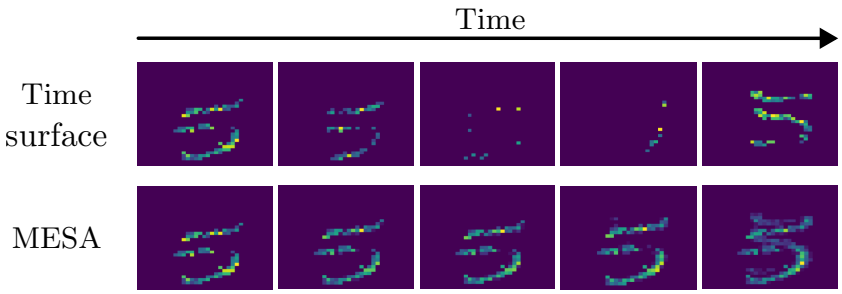


Fig. 10.3 Example of a sequence of time surfaces and a sequence of MESA representing the digit 5 from the N-MNIST dataset.

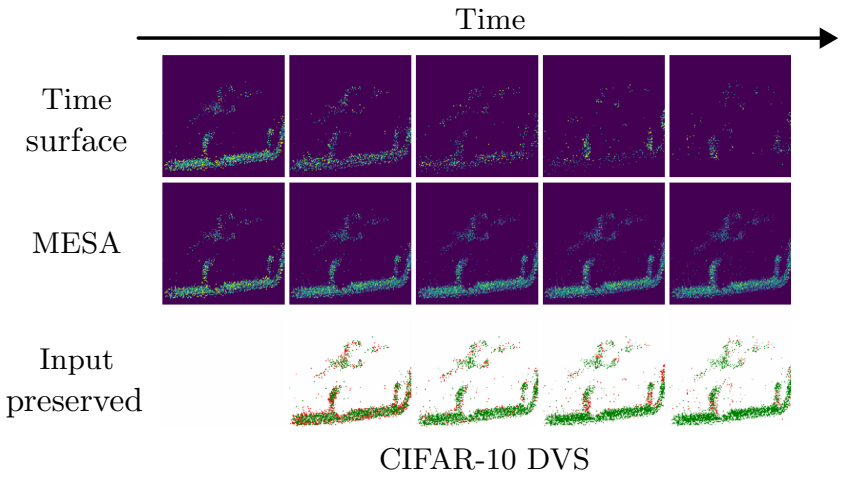


Fig. 10.4 Sequence of time surfaces from the CIFAR-10 DVS dataset (top), the corresponding MESA memory states (center), and (bottom) the retained information using MESA (where green=preserved, red=forgotten).

Dataset	DNN	Time surfaces			Voxel grids		
		without memory	*static memory*	*with MESA*	*without memory*	*static memory*	*with MESA*
Task: Regression		**Metric: Euclidean distance** (lower is better)					
3ET	ResNet-18	10.30 ± 0.85	**9.01 ± 0.14**	**8.90 ± 0.40**	10.02 ± 0.47	9.29 ± 0.22	**8.80 ± 0.19**
	MobileNet-v3s	11.26 ± 0.41	10.27 ± 0.49	**7.23 ± 0.17**	11.07 ± 0.52	9.06 ± 0.15	**7.15 ± 0.37**
	MobileViT-v2s	10.01 ± 0.42	9.06 ± 0.51	**7.08 ± 0.29**	9.57 ± 0.23	8.80 ± 0.25	**8.23 ± 0.12**
Task: Classification		**Metric: Accuracy (%)** (higher is better)					
N-MNIST	ResNet-18	82.45 ± 0.95	87.16 ± 0.93	**90.30 ± 0.93**	80.76 ± 0.17	89.60 ± 0.65	**92.97 ± 0.42**
	MobileNet-v3s	73.20 ± 2.01	76.63 ± 2.06	**79.16 ± 0.77**	72.96 ± 0.91	81.38 ± 1.17	**84.52 ± 0.25**
	MobileViT-v2s	82.66 ± 1.04	88.15 ± 0.46	**91.82 ± 0.48**	82.18 ± 0.12	88.93 ± 0.84	**92.67 ± 0.24**
CIFAR-10 DVS	ResNet-18	49.71 ± 0.25	52.70 ± 1.23	**58.91 ± 0.63**	44.90 ± 0.48	49.63 ± 0.81	**53.19 ± 0.08**
	MobileNet-v3s	47.15 ± 0.11	**52.61 ± 0.90**	**53.82 ± 0.48**	42.73 ± 0.18	44.85 ± 0.27	**50.02 ± 0.18**
	MobileViT-v2s	52.86 ± 0.31	58.16 ± 1.28	**65.24 ± 0.80**	47.72 ± 0.83	52.51 ± 0.63	**58.85 ± 0.71**
N-CALTECH101	ResNet-18	64.56 ± 0.52	68.65 ± 0.79	**70.52 ± 0.24**	63.46 ± 0.25	70.26 ± 0.76	**74.36 ± 0.12**
	MobileNet-v3s	58.62 ± 0.19	65.84 ± 0.91	**68.02 ± 0.97**	59.52 ± 0.27	**70.38 ± 1.84**	**72.38 ± 0.23**
	MobileViT-v2s	65.96 ± 0.20	72.93 ± 0.96	**74.83 ± 0.43**	65.39 ± 0.25	75.05 ± 0.66	**79.05 ± 0.23**
Task: Object detection		**Metric: mAP**$_{.5:.95}$ (higher is better)					
PEDRO	DETR-Resnet50	0.1195 ± 0.0008	0.3285 ± 0.0055	**0.3818 ± 0.0072**	0.2533 ± 0.0125	0.3923 ± 0.0183	**0.4421 ± 0.0243**
	DETR-Resnet101	0.2475 ± 0.0121	0.3965 ± 0.0234	**0.5557 ± 0.0113**	0.3843 ± 0.0211	0.4983 ± 0.0197	**0.5610 ± 0.0098**
	DETR-DC5-Resnet50	0.3423 ± 0.0305	0.5211 ± 0.0148	**0.5824 ± 0.0147**	0.4326 ± 0.0347	0.5070 ± 0.0301	**0.5833 ± 0.0102**

Table 10.1 Accuracy of the different models across the selected datasets. The best models are highlighted in **bold**. Best models with overlapping standard deviations are considered equally the best. In light gray, the results obtained with [24] and MESA.

- The MESA pipeline.
- The static memory approach (MEMO) presented in Chapter 9, where the $\boldsymbol{K}^n$ in (10.3) is replaced by a fixed value $k = 0.5$.
- The baseline approach using only a single time surface or voxel grid as input.

Figure 10.3 shows qualitatively the difference of adopting MESA or just single time surfaces as input tensor, while Figure 10.4 shows a sequence of time surfaces from the CIFAR-10 DVS dataset, highlighting the ability of MESA to maintain salient features over event representations.

In all the tested scenarios the dynamic pre-processing pipeline MESA proves to be highly effective. It yields a reduction of the Euclidean distance by up to 36% compared to using only simple representations in the regression task. Moreover, MESA leads also to a 10% increase in accuracy for classification and it can improve the mAP$_{.5:.95}$ by up to 3 times compared to the other solutions when dealing with object detection applications.

From a computational point of view, MESA employs channel reduction (both max and average) followed by a lightweight convolutional layer for the spatial attention module, an element-wise multiplication of $\boldsymbol{K}^n$ with $\boldsymbol{M}^{n-1}$ and an element-wise addition with $\boldsymbol{R}^n$. Under the assumption of a fixed number of channels and convolutional kernel size, all these operations exhibit a complexity of $O(HW)$,

	Module/model	*Memory (MB)*	*GFLOPs*
MESA (preprocessing)		0.55	0.009
backbones	ResNet-18	44.59	1.819
	MobileNet-v3s	9.70	0.060
	MobileViT-v2s	10.93	0.819
	DETR-based	158.32	5.458
	DETR-C5-based	158.32	9.084

Table 10.2 Absolute memory (MB) and computational overhead (GFLOPs) introduced by the MESA module and the requirements of each used backbone alone in terms of both memory and compute.

making their computational cost almost negligible compared to that of most neural estimator. Table 10.2 shows the memory consumption (in bytes) and the number of Floating Point Operations (FLOPs) introduced by MESA, together with the same quantities for all the employed architectures. The absolute overhead introduced by MESA remains constant across applications and it is equal to 392 bytes (32-bit floating-point parameters) and 0.018 GFPLOS. In relative terms, this overhead represents only a small fraction of the total computational complexity, peaking at 15% for MobileNet-v3s and being negligible for the object detection models. Overall, the findings demonstrate that the MESA approach significantly enhances performance while maintaining minimal computational cost and a very small increase in the memory footprint.

10.4 Conclusions

In this work, we presented *MESA*, a dynamic pre-processing input pipeline designed to enhance the performance of widely used neural architectures on event-based computer vision tasks. The key component of the proposed approach is the spatial attention module, which is used to update a memory tensor that accumulates multiple event representations over time. This attention mechanism allows the model to adapt dynamically to the incoming event data, retaining only the most important information. Experimental results on object classification and detection tasks demonstrate that our approach improves neural estimation performance, without introducing substantial computational overhead across all tested scenarios.

Chapter 11

Event-based Classification with Recurrent Spiking Neural Networks on Low-end Micro-Controller Units

Deep Neural Networks (DNNs) are powerful computational models capable of solving a wide range of complex tasks [208, 39]. In the field of Computer Vision, either with frame-based images or event-based data, they represent the state of the art, addressing challenges such as object detection, classification, gesture recognition, and 3D reconstruction. In the previous Chapters of this thesis, as in the literature, there are a lot of examples that employ various architectures to solve the aforementioned tasks, ranging from convolutional neural networks (CNNs)[86, 83, 195], transformer-based structures[137, 236, 54], and recurrent or non-recurrent models. All these architectures rely on trainable parameters that are optimized to achieve the desired performance.

Recently, a new paradigm in neuromorphic computing has gained prominence: Spiking Neural Networks (SNNs). These architectures are particularly well-suited for event-based data processing, as they leverage sparse representations in both time and space. Unlike traditional artificial neurons, spiking neurons operate as dynamic systems, maintaining a state variable and integrating inputs over time through their membrane potential. This characteristic makes SNNs highly efficient, especially when paired with event-based vision sensors [44, 72, 193]. SNNs have been successfully applied to tasks such as classification [173, 79], hand-gesture recogni-

tion [6, 71], object detection [37, 100], optical flow estimation [16], and visual place recognition [114]. Among SNN architectures, Recurrent Spiking Neural Networks (RSNNs)[15] are particularly noteworthy due to their ability to process temporal information over extended timescales, surpassing the intrinsic time constants of individual network elements[230].

In this work, we implement a RSNN for image classification using signals from an event-based sensor. A key contribution of our study is the deployment of this network on a highly resource-constrained Microcontroller Unit (MCU), which presents significant challenges in terms of memory footprint and power consumption. Inspired by previous research on RSNNs designed for hardware-constrained environments [62], we demonstrate the feasibility of running a RSNN on a low-cost, commercially available MCU. This approach enables rapid prototyping and deployment compared to more complex custom hardware implementations.

The following sections present our methodology, experimental results using the well-known N-MNIST event-based dataset [151], and a detailed analysis of the network's performance on the MCU. We evaluate classification accuracy as well as key deployment metrics such as computational time, energy consumption, and memory footprint, proving that software-based RSNNs can perform real-time inference in vision applications even on resource-limited hardware.

11.1 Methodology

The Recurrent Spiking Neural Network (RSNN) proposed in this work consists of Leaky Integrate-and-Fire (LIF) neurons and follows the time-discrete models introduced in [15]. It employs a custom implementation as proposed in [62]. An overview of the proposed methodology is illustrated in Figure 11.1.

A. RSNN Architecture

The RSNN consists of three layers: an input layer, a hidden recurrent layer, and an output layer. The input layer generates a sequence of input spikes, which are then propagated through interconnections to the recurrent layer. These input spikes are represented as vectors $x^1, \ldots, x^T$ defined at each discrete time step $t \in \{1, \cdots, T\}$, where $x^t \in \{0,1\}^N$ and N denotes the number of input neurons. A value of 1 signifies a spike at time t, while 0 indicates no activity.

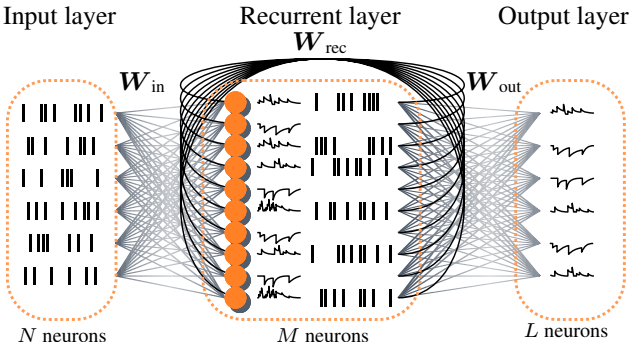


Fig. 11.1 Scheme of the RSNN model employed in this work, composed of an input layer (for feeding input spikes), a hidden recurrent layer composed of LIF neurons and an output layer composed of leaky integrators. Picture from [27].

The hidden recurrent layer consists of LIF neurons that maintain membrane potentials $v^t \in \mathbb{R}^M$ where M is the number of recurrent neurons. These neurons generate output spikes $z^t \in \{0,1\}^M$, based on their membrane potentials, which evolve according to the following update rule:

$$v_j^{t+1} = \alpha v_j^t + \sum_{i=1}^{N} w_{ji}^{\text{in}} x_i^t + \sum_{i=1, i\neq j}^{M} w_{ji}^{\text{rec}} z_i^t - \theta z_j^t \quad \text{for} \quad j \in 1,\ldots,M \tag{11.1}$$

where

- $\alpha \in (0,1)$ is a damping factor that governs the decay of the membrane potential,
- w_{ji}^{in} is the value at row j and column i of the input weights matrix W^{in},
- x_i^t is the i-th input spike,
- w_{ji}^{rec} is the value at row j and column i of the recurrent weights matrix W^{rec},
- z_i^t is the i-th value of output spikes vector z^t,
- θ is the firing threshold parameter.

Moreover, neurons in the hidden layer generate spikes based on the Heaviside step function:

$$z_j^t = \mathcal{H}\left(v_j^t - \theta\right) \quad \text{for } j \in 1,\ldots,M \tag{11.2}$$

where $\mathcal{H}(\cdot)$ is the step Heaviside function defined as

$$\mathcal{H}\left(v_j^t - \theta\right) = \begin{cases} 0 & \text{for } v_j^t \leq \theta \\ 1 & \text{for } v_j^t > \theta \end{cases} \tag{11.3}$$

Equation 11.1 consists of three terms: the first term, through the α parameter, applies exponential decay to the membrane potential when no spikes occur, the second term integrates input spikes, and the third term accounts for recurrent spikes from other neurons in the hidden layer. In detail, when a spike occurs, the value of the corresponding weight is added to the neuron potential. For the input layer the spikes come from an external source. In contrast, for the hidden layer the recurrent spikes are generated from Equation 11.2, then fed back to the hidden neurons through recurrent connections (excluding self-occurrence). When a neuron reaches its threshold θ, it generates a spike and resets its potential, as enforced by the last term of 11.1.

The output layer of the architecture consists of non-firing leaky integrator neurons, whose dynamics follow:

$$y_k^{t+1} = \kappa y_k^t + \sum_{i=0}^{L} w_{ji}^{\text{out}} z_i^t \quad \text{for } k \in 1, \ldots, L \tag{11.4}$$

where y_k^{t+1} is the k-th output of the network, $\kappa \in (0,1)$ is a damping factor and w_{ji}^{out} is the value at row j and column i of the output weights matrix W^{out}. The output neuron is fed by the output spikes of the hidden recurrent layer z^t. The damping factors α and κ determine the leakage time constants τ_{rec} and τ_{out}, as $\alpha = \exp\left(-\Delta t/\tau_{\text{rec}}\right)$ and $\kappa = \exp\left(-\Delta t/\tau_{\text{out}}\right)$, where Δt is the time interval between two discrete time-steps, i.e. the temporal resolution.

B. Training with BPTT

The RSNN is trained with the Back Propagation Through Time (BPTT) algorithm [222]. Since Equation 11.2 introduces a discontinuity, we employ a pseudo-derivative function, as suggested in [15]:

$$\frac{\partial \mathcal{H}}{\partial v_j^t} \triangleq \gamma \max\left(0, 1 - \left|v_j^t - \theta\right|\right) \tag{11.5}$$

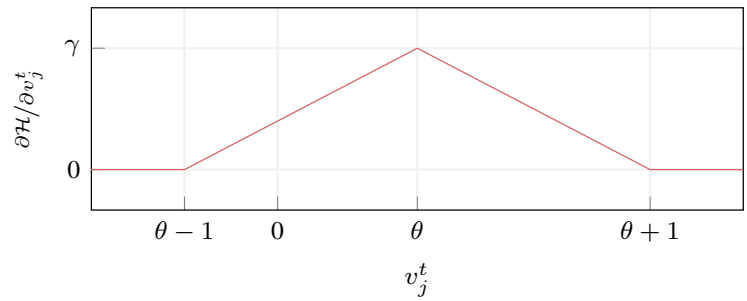


Fig. 11.2 Plot of the pseudo derivative function in (11.5). Picture from [27].

where γ controls the strength of the pseudo-derivative, as illustrated in Figure 11.2.

11.2 Dataset and Preliminary Results

The proposed methodology has been validated at first by running preliminary offline tests within the PyTorch-based framework *snnTorch*[1].

11.2.1 Dataset

The N-MNIST dataset [151] is a neuromorphic, event-based version of the widely known MNIST dataset [110]. It consists of 70 000 samples, split into 55 000 for training, 5000 for validation, and 10 000 for testing. Each sample contains event-based recordings spanning approximately 350 ms, with a temporal resolution of 1 µs and a spatial resolution of 34×34 pixels. The dataset represents handwritten digits (0-9), with each event labeled accordingly. Events are encoded as either positive ($+1$) or negative (-1), depending on the direction of pixel intensity change. The dataset acquisition process is inspired by the biological phenomenon of saccades (rapid, simultaneous eye movements between fixation phases [65]). Specifically, each MNIST digit is displayed on a monitor and recorded by a motor-driven Dynamic Vision Sensor (DVS) camera executing three "saccadic" movements in a triangular pattern.

In this work, to optimize computational efficiency and reduce memory footprint on the microcontroller unit (MCU), the event stream is downsampled. The spatial

[1]The snnTorch framework is available at: https://snntorch.readthedocs.io/en/latest/

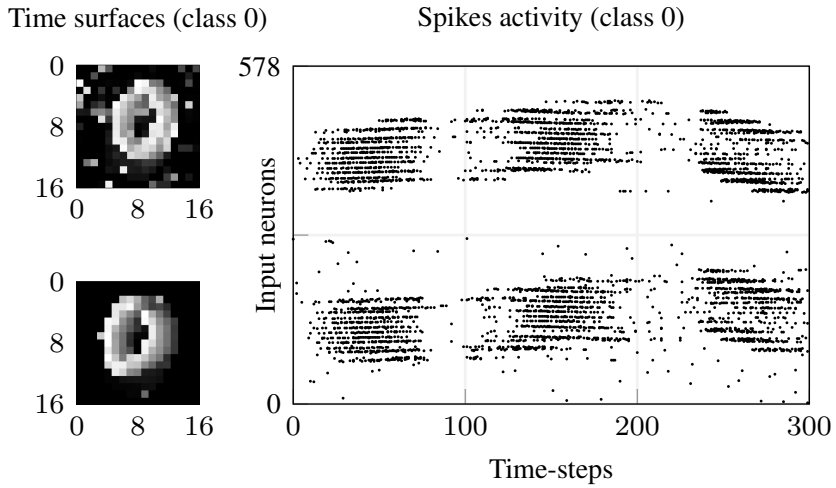


Fig. 11.3 An example of sample of class 0. On the left, the time surfaces for positive and negative events, obtained by integrating all the events over 300 ms. On the right, the spiking activity of the sample. Picture from [27].

resolution is reduced to 17×17 pixels, and all events are remapped accordingly. Additionally, the number of time steps (T) is limited to 300 by discarding events beyond 300 ms and downsampling the temporal resolution to $\Delta t = 1$ ms. Within each 1 ms bin, events are aggregated. This time resolution is commonly used in Spiking Neural Network (SNN) tasks, as finer resolutions typically do not yield significant benefits [62, 71, 225]. An example of an input sample is shown in Figure 11.3.

11.2.2 Preliminary Results

The Recurrent Spiking Neural Network (RSNN) consists of $M = 100$ recurrent neurons and $L = 10$ output neurons, corresponding to the classification of digits from 0 to 9. The network is configured with the following hyperparameters: threshold $\theta = 0.6$, recurrent time constant $\tau_{rec} = 250$ ms, output time constant $\tau_{out} = 20$ ms, and factor $\gamma = 0.3$. The input layer comprises $N = 578$ neurons, reflecting the event-based nature of the N-MNIST dataset. Each of the 17×17 pixels in the down-sampled images is represented by two input neurons, one for positive and one for negative events, leading to a total of 578 input neurons.

Training is performed using Backpropagation Through Time (BPTT) with the Adam optimizer[102]. The primary loss function is the Cross-Entropy (CE) loss, computed between the output potentials y^t and the ground-truth labels y^t_{true}, averaged

over all time steps $t \in 1, \ldots, T$. During validation and testing, the predicted class corresponds to the output neuron with the highest average potential. To encourage sparsity in the recurrent layer, an additional regularization term is introduced, computed as the L^2 norm of the spike trains z^t, averaged over time. This regularization reduces spiking activity by approximately 90% during inference while maintaining classification accuracy.

Quantization-aware training is employed to enable efficient deployment with quantized parameters. Specifically, fake quantization is applied during training, following the approach in [232], where dynamically quantized 8-bit weights are used in the forward pass. The network is trained with a batch size of 5 and a learning rate of 10^{-4}. After 50 epochs, the model achieves a test accuracy of 97.2%, which is competitive with state-of-the-art results on the N-MNIST dataset, even compared to more complex architectures [203, 41, 180, 234].

11.3 Implementation on MCU and Performance

11.3.1 Implementation on MCU

The Recurrent Spiking Neural Network (RSNN) described in the previous section is implemented in software on an ARM Cortex-M4-based microcontroller unit (MCU), the STM32F767ZI [2]. This MCU features 512 kB of SRAM and operates at a maximum frequency of 216 MHz.

Spike Processing and Network Execution

Spike updates are applied column-first, meaning that each spike's potential update is sequentially applied to all neuron potentials before processing the next spike. Additionally, leakage is applied to all neuron potentials at each time step. The C code implementation [3] is compiled using GCC with the -Ofast and -loop-unroll optimization flags. To maximize performance, the STM32 AXI interface is enabled, along with both data and instruction caches.

Data Encoding

[2] https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html
[3] GitHub repository at https://github.com/SSIGPRO/ucrsnn.

The different types of spikes present in the network are encoded differently, in the following ways:

- *Input Spikes:* Each input event is encoded in 32 bits, the first 16 bits specify the input neuron address while the last 16 bits indicate the number of time steps from the previous spike (that can be associated with any input neuron).
- *Recurrent Spikes:* These spikes are internally generated at each time step. They are stored in 8-bit value arrays which express the address of the corresponding recurrent neuron.

The weights of the system are encoded with 8 bits (sign bit with 7 fractional bits) while the neuron potentials are encoded with 32 bits (sign bit, 16 integer bits, and 15 fractional bits). Given the bit alignment used for neuron potentials, an overflow detection mechanism is unnecessary, as the numerical range is significantly larger than what can be encountered in practice.

11.3.2 Performances

The performance of the RSNN implemented on the MCU is evaluated by inferring all 10000 samples of the test set, each consisting of 300 time steps. The accuracy remains consistent with the one reported in Section 11.2. Moreover, to measure the computational time required for each operation we used an on-chip counter.

Figure 11.4 illustrates the computational time per time step (Δt_c) for different samples at an operating frequency of 144 MHz, alongside the model's spiking activity. The total computational time is broken down into its main components:

- Applying leakages to recurrent and output neurons.
- Applying input and recurrent spikes to recurrent and output neurons.
- Generating recurrent spikes.

The most time-consuming operation is the integration of spikes into neuron potentials, which scales proportionally with the number of spikes. In contrast, other computational contributions are minimal. Table 11.1 presents key performance

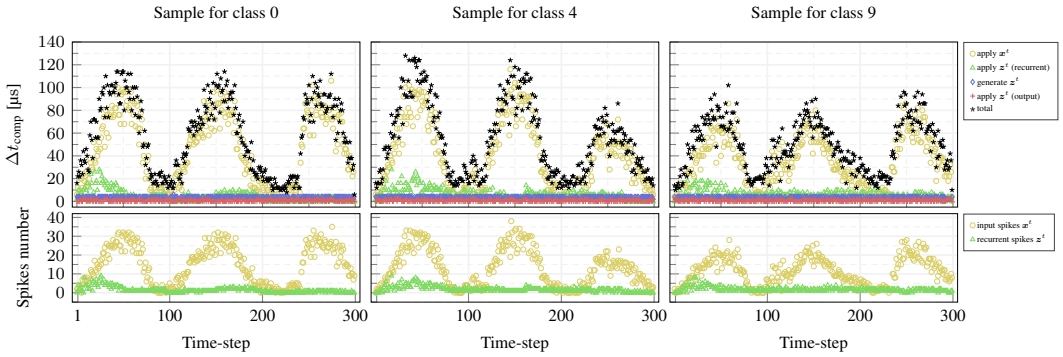


Fig. 11.4 Computational time per time-step Δt_c for example inferred samples with $f_{\text{CLK}} = 144\,\text{MHz}$, compared with the input and recurrent spiking activity. The computational time is split into different contributions, and the contributions due to the application of the leakage are not shown as they are below 1 µs. The greatest contributions are due to the input spikes whose number is high compared to the recurrent spikes. Their trend is coherent with the three movements performed by the DVS camera while recording N-MNIST. Picture from [27]

f_{CLK}	P_{typ}	P_{max}	$\Delta t_{\text{c, avg}}$	$\Delta t_{\text{c, worst-case}}$	$E_{\text{c, avg}}$	$E_{\text{c, worst-case}}$
25 MHz	13.1 mW	18.5 mW	292.5 µs	1082.4 µs	3.8 µJ	21.9 µJ
60 MHz	26.8 mW	34.1 mW	121.5 µs	444.7 µs	3.3 µJ	15.9 µJ
144 MHz	58.7 mW	70.1 mW	50.1 µs	207.0 µs	3.1 µJ	14.8 µJ
169 MHz	83.6 mW	98.1 mW	43.5 µs	161.3 µs	3.7 µJ	16.1 µJ
180 MHz	99.5 mW	116.5 mW	40.6 µs	150.4 µs	4.1 µJ	17.8 µJ

Table 11.1 Power consumption of STM32F767ZI, with the computational time and energy consumption for a single time-step update of the RSNN.

metrics, including the MCU's average and worst-case power consumption, the computational time per time step, and the corresponding energy consumption. Power consumption is estimated based on values provided in the datasheet, assuming an internal supply voltage of 1.2 V, with the STM32 AXI interface and cache enabled and all peripherals disabled. Computational times are averaged over multiple time steps across different samples, while energy consumption is calculated as the product of power and time. At the lowest tested operating frequency, the worst-case computational time is approximately 1 ms, whereas at 180 MHz, it decreases to just 150.4 µs. These results highlight the feasibility of running RSNN models on low-cost, resource-constrained MCUs, offering fast deployment compared to custom hardware implementations. Finally, the memory footprint of the RSNN model is mostly defined by its parameters, which occupy 68.8 kB in this work.

11.4 Conclusions

In this work, we present the implementation of a Recurrent Spiking Neural Network (RSNN) on a Microcontroller Unit (MCU) for solving an object classification task, specifically N-MNIST digit recognition. This approach offers several advantages, including the use of widely available commercial devices, rapid prototyping, and quick deployment. Additionally, by maintaining an accuracy of 97.2%, we achieved a worst-case computational time per step as low as 150.4 µs and average energy consumption of 4.1 µJ at a $f_{CLK} = 180\,\text{MHz}$. The proposed methodology shows that it is possible to use software-based RSNN for inference in real-time vision applications employing resource-constrained hardware.

Chapter 12

Conclusions and Future Works

Among the five senses, sight is one of the most crucial for both humans and animals to survive in their environments. It plays a vital role not only in navigation and spatial awareness but also in object recognition and interaction with the surrounding world. Furthermore, vision is a key component in effective decision-making processes. Given its importance, it is no surprise that researchers, especially in the field of robotics, are highly interested in leveraging visual sensors to enable robots to operate autonomously and perform a wide range of tasks across diverse application domains. Many of these methodologies are inspired by biological systems, particularly those related to visual perception.

The most commonly used visual sensor is the traditional frame-based camera, which captures grayscale or RGB images at a fixed rate. However, over the past decade, a new class of visual sensors has emerged: event-based cameras. These represent a significant shift in how visual information can be sensed and processed. Event-based cameras are inspired by the human visual system. Unlike conventional cameras that capture full frames at regular intervals, each pixel in an event-based sensor operates independently and asynchronously. These pixels detect changes in brightness at their location, and each change triggers an event. An event contains four key pieces of information: the timestamp of the change, the pixel's position, and the polarity (i.e., whether the brightness increased or decreased). The result is a continuous stream of events, rather than a sequence of full images. Compared to traditional cameras, event-based cameras offer numerous advantages: ultra-low latency, extremely high temporal resolution, low power consumption, high dynamic

range, and reduced redundant information. These characteristics make event-based sensors particularly well-suited for dynamic environments involving fast movements or limited computational resources, common in fields such as service robotics, where mobile platforms are often employed.

This dissertation has explored the intersection of event-based vision, bio-inspired navigation strategies, and lightweight neuromorphic computation, demonstrating both theoretical contributions and practical implementations. Several approaches have been presented for processing and leveraging visual information, along with examples of application scenarios where the proposed methods can be effectively employed. Part II focuses on the extraction of biologically plausible visual cues from sequences of RGB images. The concept of Time-to-Transit (TTT) is introduced as a means to enable autonomous navigation for mobile platforms in unknown environments, reducing reliance on extensive sensor arrays or overly complex algorithms. The proposed approach is a starting point with the aim of demonstrating that, when accurately estimated, TTT can function as a robust control signal in both simulated and real-world experiments. The methodologies presented are designed to enhance TTT estimation, also under critical conditions, such as during rotational motion or within dynamic environments. This is achieved through empirical strategies, including a sense-act cycle, as well as the integration of deep learning architectures to further boost performance and reliability.

On the other hand, the subject of Part III is the exploration and analysis of event-based sensors, with particular attention given to how this novel type of signal can be organized and processed for use in state-of-the-art deep learning techniques applied to common computer vision tasks such as object detection and classification. A first contribution is introduced, presenting a useful tool to the event-based vision community: PEDRo, an event-based dataset specifically designed for person detection, entirely collected using a moving sensor. The discussion then shifts to the development of a pre-processing strategy that takes event-based data as input and can be easily used with standard deep learning architectures, without introducing significant computational overhead or complexity. Initially, this strategy is applied to the problem of eye tracking, leveraging static *memory* structures to retain and utilize temporal information. Subsequently, the approach is extended to more general tasks like object detection and classification. To better capture the dynamic nature of scenes, the method is further enhanced through the introduction of dynamic memory structures. Extensive experiments shows the effectiveness of the methodology pro-

posed, by enhancing accuracy with respect to other work that used already existing techniques to organize raw-event data.

Finally, Chapter 11 presents a work that integrates event-based sensor data with a neuromorphic neural network, specifically, a Spiking Neural Network (SNN), everything deployed on an MCU. This research demonstrates that neuromorphic models can be effectively deployed on standard microcontrollers to solve computer vision tasks. A Recurrent SNN was developed for digit classification using event-based input data. The trained network was then deployed on a widely used microcontroller, and its performance was evaluated. Special emphasis is placed on energy efficiency: the system achieves high accuracy while maintaining remarkably low energy consumption, highlighting its potential for low-power, real-time applications.

Within the broader evolution of artificial intelligence, recent years have been marked by the rapid emergence of transformer-based architectures as dominant models in computer vision and, more generally, as unifying frameworks across multiple modalities. Attention mechanisms are increasingly being extended to event-based signals and multimodal perception systems. In this context, although the methodologies developed in this dissertation are primarily grounded in convolutional neural networks, with a strong emphasis on efficiency and deployability, they address fundamental representation challenges that remain central regardless of the architectural paradigm adopted. This is supported by preliminary experimental results presented in this work, including investigations combining MESA with DETR and MobileViT, as well as analyses exploring the use of MobileViT and GMDepth for TTT estimation. These observations suggest that the proposed representations and processing strategies could be further explored in conjunction with transformer-based architectures, particularly in scenarios where transformers are employed as unifying models for integrating heterogeneous sensing modalities.

12.1 Future Works

This dissertation presents different methodologies and experiments that represent an initial attempt to address challenges at the intersection of perception, service robotics, and bio-inspired signals. While these approaches show promise, several challenges remain, and further developments could enhance the research in these domains.

In Part II, even though the proposed methodology, particularly the latest version presented in Chapter 6, has proven effective for reliably estimating Time-To-Transit (TTT) to be applied in robotic control, there is still space for improvement. Firstly, additional experiments in real-world scenarios would strengthen the applicability of the approach. Then, new control strategies could be developed by leveraging the concept of estimating TTT on a coarse grid, which offers a finer resolution than traditional, large predefined regions of interest while still supporting real-time performance. Additionally, deep learning techniques could be explored to compute control signals based on the estimated TTT. This includes investigating reinforcement learning approaches, which would enable the platform to learn from its environment and adapt its decisions during navigation.

The usage of event-based sensors is a subject that is gaining more and more attention among researchers, leading to innovative discoveries and future developments. The number of publications on the topic has shown an increasing trend in the last two years, and this interest in the academic research community is also reflected in the industrial field. New collaborations have emerged between companies and event-based sensor manufacturers, such as the partnership between Prophesee and Sony, demonstrating the commercial viability and appeal of this technology.

Particularly referring to Part III of this dissertation, the PEDRo dataset introduced in Chapter 7 represents a valuable contribution, which can be useful for research areas such as mobile robotics and social navigation. Indeed, this dataset enables the exploration of neuromorphic sensors in challenging but realistic situations, like highly cluttered environments. Furthermore, the methodologies presented in Chapters 9 and 10 constitute an initial step toward several significant extensions. One important direction involves validating these approaches in real-world applications, where the inherent advantages of event-based sensors, such as robustness to highly dynamic scenes and resilience to extreme or rapidly changing lighting conditions, can be fully exploited to enhance system performance. Another relevant extension concerns broader benchmarking efforts, including systematic comparisons with recent transformer-based architectures for event-based vision. These models have demonstrated strong capabilities in capturing long-range spatio-temporal dependencies through attention mechanisms. Such comparative analyses would enable a deeper understanding of the trade-offs among architectural complexity, accuracy, computational cost, and energy efficiency.

Finally, Chapter 11 presents a promising foundation for future work that could involve integrating event-based vision with neuromorphic architectures for more complex visual tasks. An interesting direction is the development of a fully neuromorphic mobile platform to evaluate the practical benefits of such a setup. This also aligns with the research in neuromorphic computing, where the aim is to emulate the principles of the human brain to achieve more energy-efficient and scalable technologies. Technologies which, as stated in a recent paper published in Nature [108], are almost ready to be adopted in a wide range of applications, potentially even at the industrial production level.

References

[1] Ahmad, S., Morerio, P., and Del Bue, A. (2024). Event Anonymization: Privacy-Preserving Person Re-Identification and Pose Estimation in Event-Based Vision. *IEEE Access*, 12:66964–66980.

[2] Aitsam, M., Davies, S., and Di Nuovo, A. (2024). Event Camera-Based Real-Time Gesture Recognition for Improved Robotic Guidance. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

[3] Al-Tawil, B., Hempel, T., Abdelrahman, A., and Al-Hamadi, A. (2024). A review of visual SLAM for robotics: Evolution, properties, and future applications. *Frontiers in Robotics and AI*, 11:1347985.

[4] AliAkbarpour, H., Moori, A., Khorramdel, J., Blasch, E., and Tahri, O. (2024). Emerging Trends and Applications of Neuromorphic Dynamic Vision Sensors: A Survey. *IEEE Sensors Reviews*, 1:14–63.

[5] Alzugaray, I. and Chli, M. (2018). Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robotics and Automation Letters*, 3(4):3177–3184.

[6] Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., and Modha, D. (2017). A Low Power, Fully Event-Based Gesture Recognition System. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397.

[7] Angelopoulos, A. N., Martel, J. N., Kohli, A. P., Conradt, J., and Wetzstein, G. (2021). Event-Based Near-Eye Gaze Tracking Beyond 10,000 Hz. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2577–2586.

[8] Arguello, H. and Arce, G. R. (2013). Rank Minimization Code Aperture Design for Spectrally Selective Compressive Imaging. *Image Processing, IEEE Transactions on*, 22(3):941–954.

[9] Bai, W., Chen, Y., Feng, R., and Zheng, Y. (2022). Accurate and Efficient Frame-based Event Representation for AER Object Recognition. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6.

[10] Baillieul, J. (2019). Perceptual Control with Large Feature and Actuator Networks. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3819–3826, Nice, France.

[11] Baillieul, J. and Kang, F. (2020-07-12-17). Visual Navigation with a 2-pixel Camera—Possibilities and Limitations. In *Proceedings of the 21st IFAC World Congress*, Berlin, Germany.

[12] Baldwin, R. W., Liu, R., Almatrafi, M., Asari, V., and Hirakawa, K. (2023). Time-ordered recent event (TORE) volumes for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2519–2532.

[13] Bandini, A. and Zariffa, J. (2023). Analysis of the Hands in Egocentric Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):6846–6866.

[14] Becattini, F., Palai, F., and Bimbo, A. D. (2022). Understanding Human Reactions Looking at Facial Microexpressions With an Event Camera. *IEEE Transactions on Industrial Informatics*, 18(12):9112–9121.

[15] Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2020). A Solution to the Learning Dilemma for Recurrent Networks of Spiking Neurons. *Nature Communications*, 11(1):3625.

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

[17] Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., and Andreopoulos, Y. (2019). Graph-based object classification for neuromorphic vision sensing. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 491–501.

[18] Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., and Andreopoulos, Y. (2020). Graph-Based Spatio-Temporal Feature Learning for Neuromorphic Vision Sensing. *IEEE Transactions on Image Processing*, 29:9084–9098.

[19] Bich, P., Boretti, C., Prono, L., Pareschi, F., Rovatti, R., and Setti, G. (2024). Optimizing Vision Transformers: Leveraging Max and Min Operations for Efficient Pruning. In *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)*, pages 337–341.

[20] Bich, P., Prono, L., Boretti, C., Pareschi, F., Rovatti, R., and Setti, G. (2025). MESA: A Dynamical Attention-based Pre-processing Pipeline for High-throughput Event-based Computer Vision Tasks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 72(12):2057–2061.

[21] Blum, H., Dietmüller, A., Milde, M., Conradt, J., Indiveri, G., and Sandamirskaya, Y. (2017). A Neuromorphic Controller for a Robotic Vehicle Equipped with a Dynamic Vision Sensor. *Robotics Science and Systems, RSS 2017*.

[22] Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. D. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348–1357.

[23] Boretti, C., Bich, P., Pareschi, F., Prono, L., Rovatti, R., and Setti, G. (2023a). PEDRo: An Event-based Dataset for Person Detection in Robotics. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4065–4070.

[24] Boretti, C., Bich, P., Prono, L., Pareschi, F., Rovatti, R., and Setti, G. (2024). Memory in Motion: Exploring Leaky Integration of Time Surfaces for Event-based Eye-tracking. In *2024 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–5.

[25] Boretti, C., Bich, P., Zhang, Y., and Baillieul, J. (2021). Visual navigation using sparse optical flow and time-to-transit. *arXiv preprint arXiv:2111.09669*.

[26] Boretti, C., Bich, P., Zhang, Y., and Baillieul, J. (2022). Visual Navigation Using Sparse Optical Flow and Time-to-Transit. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9397–9403.

[27] Boretti, C., Prono, L., Frenkel, C., Indiveri, G., Pareschi, F., Mangia, M., Rovatti, R., and Setti, G. (2023b). Event-based Classification with Recurrent Spiking Neural Networks on Low-end Micro-Controller Units. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5.

[28] Bouguet, J.-Y. et al. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel corporation*, 5(1-10):4.

[29] Bouwmeester, R. J., Paredes-Vallés, F., and de Croon, G. C. H. E. (2023). NanoFlowNet: Real-time Dense Optical Flow on a Nano Quadcopter. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1996–2003.

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

[31] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *2012 European Conference on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625.

[32] Cain, S., Hayat, M., and Armstrong, E. (2001). Projection-based image registration in the presence of fixed-pattern noise. *IEEE Transactions on Image Processing*, 10(12):1860–1872.

[33] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *2010 European Conference on Computer Vision (ECCV)*, pages 778–792. Springer.

[34] Cannici, M., Ciccone, M., Romanoni, A., and Matteucci, M. (2019). Asynchronous convolutional networks for object detection in neuromorphic cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[35] Cannici, M., Ciccone, M., Romanoni, A., and Matteucci, M. (2020). A differentiable recurrent surface for asynchronous event-based data. In *2020 European Conference on Computer Vision (ECCV)*, pages 136–152. Springer.

[36] Cao, J., Zheng, X., Lyu, Y., Wang, J., Xu, R., and Wang, L. (2024). Chasing Day and Night: Towards Robust and Efficient All-Day Object Detection Guided by an Event Camera. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9026–9032.

[37] Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *International Journal of Computer Vision*, 113(1):54–66.

[38] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *2020 European Conference on Computer Vision (ECCV)*, pages 213–229. Springer.

[39] Chai, J., Zeng, H., Li, A., and Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134.

[40] Chen, Q., Wang, Z., Liu, S.-C., and Gao, C. (2023). 3ET: Efficient Event-based Eye Tracking using a Change-Based ConvLSTM Network. In *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–5.

[41] Cheng, X., Hao, Y., Xu, J., and Xu, B. (2020). LISNN: Improving Spiking Neural Networks with Lateral Interactions for Robust Object Recognition. In *Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 2, pages 1519–1525.

[42] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[43] Collins, R. (1996). A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–363.

[44] Dayan, P. and Abbott, L. F. (2005). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press.

[45] de Croon, G. C. H. E., Dupeyroux, J. J. G., De Wagter, C., Chatterjee, A., Olejnik, D. A., and Ruffier, F. (2022). Accommodating Unobservability to Control Flight Attitude with Optic Flow. *Nature*, 610(7932):485–490.

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

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

[48] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[49] Deng, Y., Chen, H., and Li, Y. (2021). MVF-Net: A multi-view fusion network for event-based object classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(12):8275–8284.

[50] Deng, Y., Chen, H., Liu, H., and Li, Y. (2022). A voxel graph cnn for object classification with event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1172–1181.

[51] Deng, Y., Li, Y., and Chen, H. (2020). Amae: Adaptive motion-agnostic encoder for event-based object classification. *IEEE Robotics and Automation Letters*, 5(3):4596–4603.

[52] Di Capua, M., Ciaramella, A., and De Prisco, A. (2023). Machine learning and computer vision for the automation of processes in advanced logistics: The integrated logistic platform (ilp) 4.0. *Procedia Computer Science*, 217:326–338.

[53] Doelling, K., Shin, J., and Popa, D. O. (2014). Service robotics for the home: a state of the art review. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*. Association for Computing Machinery.

[54] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of 2021 International Conference on Learning Representations (ICLR)*.

[55] Duan, H., Zhai, G., Min, X., Che, Z., Fang, Y., Yang, X., Gutiérrez, J., and Callet, P. L. (2019). A dataset of eye movements for the children with autism spectrum disorder. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 255–260.

[56] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338.

[57] Fajen, B. R. (2001). Steering toward a Goal by Equalizing Taus. *Journal of Experimental Psychology. Human Perception and Performance*, 27(4):953–968.

[58] Falanga, D., Kleber, K., and Scaramuzza, D. (2020). Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712.

[59] Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

[60] Fischer, T. and Milford, M. (2020). Event-based visual place recognition with ensembles of temporal windows. *IEEE Robotics and Automation Letters*, 5(4):6924–6931.

[61] Freeman, A. C., Mayer-Patel, K., and Singh, M. (2024). Accelerated event-based feature detection and compression for surveillance video systems. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 132–143.

[62] Frenkel, C. and Indiveri, G. (2022). ReckOn: A 28nm Sub-mm2 Task-Agnostic Spiking Recurrent Neural Network Processor Enabling On-Chip Learning over Second-Long Timescales. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 65, pages 1–3.

[63] Fukushima, K., Miyake, S., and Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (5):826–834.

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

[65] Gegenfurtner, K. R. (2016). The Interaction Between Vision and Eye Movements. *Perception*, 45(12):1333–1357.

[66] Gehrig, D., Loquercio, A., Derpanis, K. G., and Scaramuzza, D. (2019). End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5633–5643.

[67] Gehrig, D., Rüegg, M., Gehrig, M., Hidalgo-Carrió, J., and Scaramuzza, D. (2021a). Combining Events and Frames Using Recurrent Asynchronous Multimodal Networks for Monocular Depth Prediction. *IEEE Robotics and Automation Letters*, 6(2):2822–2829.

[68] Gehrig, M., Aarents, W., Gehrig, D., and Scaramuzza, D. (2021b). DSEC: A Stereo Event Camera Dataset for Driving Scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954.

[69] Gehrig, M. and Scaramuzza, D. (2023). Recurrent Vision Transformers for Object Detection with Event Cameras. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13884–13893.

[70] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

[71] George, A. M., Banerjee, D., Dey, S., Mukherjee, A., and Balamurali, P. (2020). A Reservoir-based Convolutional Spiking Neural Network for Gesture Recognition from DVS Input. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9.

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

[73] Gibson, J. J. (1950). *The Perception of the Visual World*. Houghton Mifflin.

[74] Godard, C., Aodha, O. M., and Brostow, G. J. (2017). Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611. IEEE.

[75] Gouda, M., Lugnan, A., Dambre, J., van den Branden, G., Posch, C., and Bienstman, P. (2023). Improving the Classification Accuracy in Label-Free Flow Cytometry Using Event-Based Vision and Simple Logistic Regression. *IEEE Journal of Selected Topics in Quantum Electronics*, 29(2: Optical Computing):1–8.

[76] Guo, M., Huang, J., and Chen, S. (2017). Live demonstration: A 768 × 640 pixels 200Meps dynamic vision sensor. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*.

[77] Guo, S. and Gallego, G. (2024). CMax-SLAM: Event-based rotational-motion bundle adjustment and SLAM system using contrast maximization. *IEEE Transactions on Robotics*, 40:2442–2461.

[78] Hadviger, A., Cvišić, I., Marković, I., Vražić, S., and Petrović, I. (2021). Feature-based event stereo visual odometry. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE.

[79] Han, B. and Roy, K. (2020). Deep Spiking Neural Network: Energy Efficiency Through Time Based Coding. In *2020 European Conference on Computer Vision (ECCV)*, pages 388–404.

[80] Han, J., Tao, J., and Wang, C. (2020). FlowNet: A Deep Learning Framework for Clustering and Selection of Streamlines and Stream Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 26(4):1732–1744.

[81] Haralick, R. M. and Shapiro, L. G. (1990). Glossary of computer vision terms. *Pattern Recognit.*, 24:69–93.

[82] Harris, C., Stephens, M., et al. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer.

[83] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[84] Holland, J., Kingston, L., McCarthy, C., Armstrong, E., O'Dwyer, P., Merz, F., and McConnell, M. (2021). Service robots in the healthcare sector. *Robotics*, 10(1):47.

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

[86] Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., and Le, Q. (2019). Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324. IEEE.

[87] Howard, A. G. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[88] Hu, Y., Delbruck, T., and Liu, S.-C. (2020). Learning to exploit multiple vision modalities by using grafted networks. In *2020 European Conference on Computer Vision (ECCV)*, pages 85–101. Springer.

[89] Hu, Y., Liu, S.-C., and Delbruck, T. (2021). v2e: From Video Frames to Realistic DVS Events. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1312–1321.

[90] Hubel, D. H. and Wiesel, T. N. (1963). Receptive fields of cells in striate cortex of very young, visually inexperienced kittens. *Journal of neurophysiology*, 26(6):994–1002.

[91] Iddrisu, K., Shariff, W., Corcoran, P., O'Connor, N. E., Lemley, J., and Little, S. (2024). Event Camera-Based Eye Motion Analysis: A Survey. *IEEE Access*, 12:136783–136804.

[92] Inivation (2019). DAVIS346. https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf.

[93] Inivation (2022). DVXplorer Mini. https://inivation.com/wp-content/uploads/2023/03/DVXplorer-Mini.pdf.

[94] Jaderberg, M., Simonyan, K., Zisserman, A., and kavukcuoglu, k. (2015). Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[95] Javaid, M., Haleem, A., Singh, R. P., and Ahmed, M. (2024). Computer vision to enhance healthcare domain: An overview of features, implementation, and opportunities. *Intelligent Pharmacy*.

[96] Jiang, X., Wang, W., and Bengler, K. (2015). Intercultural Analyses of Time-to-Collision in Vehicle–Pedestrian Conflict on an Urban Midblock Crosswalk. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):1048–1053.

[97] Jiang, Z., Xia, P., Huang, K., Stechele, W., Chen, G., Bing, Z., and Knoll, A. (2019). Mixed Frame-/Event-Driven Fast Pedestrian Detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8332–8338.

[98] Kielty, P., Dilmaghani, M. S., Shariff, W., Ryan, C., Lemley, J., and Corcoran, P. (2023). Neuromorphic Driver Monitoring Systems: A Proof-of-Concept for Yawn Detection and Seatbelt State Detection Using an Event Camera. *IEEE Access*, 11:96363–96373.

[99] Kim, H., Leutenegger, S., and Davison, A. J. (2016). Real-Time 3D reconstruction and 6-DoF tracking with an event camera. In *2016 European Conference on Computer Vision (ECCV)*.

[100] Kim, S., Park, S., Na, B., and Yoon, S. (2020). Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11270–11277.

[101] Kim, T., Cho, H., and Yoon, K.-J. (2024). Frequency-aware event-based video deblurring for real-world motion blur. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24966–24976.

[102] Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization.

[103] Koide, K., Oishi, S., Yokozuka, M., and Banno, A. (2023). General, Single-shot, Target-less, and Automatic LiDAR-Camera Extrinsic Calibration Toolbox. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11301–11307.

[104] Kong, Z., Özcimder, K., Fuller, N., Greco, A., Theriault, D., Wu, Z., Kunz, T., Betke, M., and Baillieul, J. (2013). Optical Flow Sensing and the Inverse Perception Problem for Flying Bats. In *52nd IEEE Conference on Decision and Control*, pages 1608–1615.

[105] Kramer, J. (2002). An on/off Transient Imager with Event-Driven, Asynchronous Read-Out. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings*, volume 2, pages II–II.

[106] Krizhevsky, A. and Hinton, G. (2009). Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto.

[107] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[108] Kudithipudi, D., Schuman, C., Vineyard, C. M., Pandit, T., Merkel, C., Kubendran, R., Aimone, J. B., Orchard, G., Mayr, C., Benosman, R., et al. (2025). Neuromorphic computing at scale. *Nature*, 637(8047):801–812.

[109] Lagorce, X., Orchard, G., Galluppi, F., Shi, B. E., and Benosman, R. B. (2017). HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346–1359.

[110] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[111] Lee, A. J. and Kim, A. (2021). EventVLAD: Visual place recognition with reconstructed edges from event cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2247–2252. IEEE.

[112] Lee, D. N. (1976). A Theory of Visual Control of Braking Based on Information about Time-to-Collision. *Perception*, 5(4):437–459.

[113] Lee, D. N. and Reddish, P. E. (1981). Plummeting Gannets: A Paradigm of Ecological Optics. *Nature*, 293(5830):293–294.

[114] Lee, H. and Hwang, H. (2023). Ev-ReconNet: Visual Place Recognition Using Event Camera With Spiking Neural Networks. *IEEE Sensors Journal*, 23(17):20390–20399.

[115] Lee, K.-H. and Williams, L. M. (2000). Eye Movement Dysfunction as a Biological Marker of Risk for Schizophrenia. *Australian & New Zealand Journal of Psychiatry*, 34(1_suppl):A91–A100. PMID: 11129321.

[116] Lenz, G., Chaney, K., Shrestha, S. B., Oubari, O., Picaud, S., and Zarrella, G. (2021). Tonic: Event-Based Datasets and Transformations. Zenodo.

[117] Li, C., Brandli, C., Berner, R., Liu, H., Yang, M., Liu, S.-C., and Delbruck, T. (2015). Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 718–721.

[118] Li, F., Zhang, H., Xu, H., Liu, S., Zhang, L., Ni, L. M., and Shum, H.-Y. (2023). Mask dino: Towards a unified transformer-based framework for object detection and segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3041–3050.

[119] Li, H., Liu, H., Ji, X., Li, G., and Shi, L. (2017). CIFAR10-DVS: An Event-Stream Dataset for Object Classification. *Frontiers in Neuroscience*, 11.

[120] Li, J., Dong, S., Yu, Z., Tian, Y., and Huang, T. (2019). Event-Based Vision Enhanced: A Joint Detection Framework in Autonomous Driving. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1396–1401.

[121] Li, J., Li, J., Zhu, L., Xiang, X., Huang, T., and Tian, Y. (2022). Asynchronous Spatio-Temporal Memory Network for Continuous Event-Based Object Detection. *IEEE Transactions on Image Processing*, 31:2975–2987.

[122] Li, J., Liao, B., LU, X., Liu, P., Shen, S., and Zhou, Y. (2024). Event-Aided Time-to-Collision Estimation for Autonomous Driving. In *2024 European Conference on Computer Vision (ECCV)*, pages 57–73.

[123] Liang, Z., Cao, H., Yang, C., Zhang, Z., and Chen, G. (2022). Global-local Feature Aggregation for Event-based Object Detection on EventKITTI. In *2022 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 1–7.

[124] Lichtsteiner, P., Posch, C., and Delbruck, T. (2006). A 128 X 128 120db 30mw Asynchronous Vision Sensor That Responds to Relative Intensity Change. In *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, pages 2060–2069.

[125] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128$\times$ 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576.

[126] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *2014 European Conference on Computer Vision (ECCV)*, pages 740–755. Springer.

[127] Liu, S.-C., Delbruck, T., Indiveri, G., Whatley, A., and Douglas, R. (2014). *Event-based neuromorphic systems*. John Wiley & Sons.

[128] Loshchilov, I. and Hutter, F. (2019). Decoupled Weight Decay Regularization. (arXiv:1711.05101).

[129] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110.

[130] Lucas, B. D. and Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, volume 2, pages 674–679.

[131] Luo, A., Li, X., Yang, F., Liu, J., Fan, H., and Liu, S. (2024). FlowDiffuser: Advancing Optical Flow Estimation with Diffusion Models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19167–19176.

[132] Manderscheid, J., Sironi, A., Bourdis, N., Migliore, D., and Lepetit, V. (2019). 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)*, pages 10237–10246. IEEE Computer Society.

[133] Manikanda Kumaran, K. and Chinnadurai, M. (2020). Cloud-based robotic system for crowd control in smart cities using hybrid intelligent generic algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 11(12):6293–6306.

[134] Maqueda, A. I., Loquercio, A., Gallego, G., García, N., and Scaramuzza, D. (2018). Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5419–5427.

[135] Masone, C. and Caputo, B. (2021). A survey on deep visual place recognition. *IEEE Access*, 9:19516–19547.

[136] McCandless, M., Gerald, A., Carroll, A., Aihara, H., and Russo, S. (2021). A Soft Robotic Sleeve for Safer Colonoscopy Procedures. *IEEE Robotics and Automation Letters*, 6(3):5292–5299.

[137] Mehta, S. and Rastegari, M. (2022). MobileViT: Light-weight, General-Purpose, and Mobile-Friendly Vision Transformer. In *Proceedings of 2022 International Conference on Learning Representations (ICLR)*.

[138] Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., and Wang, J. (2021). Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3651–3660.

[139] Messikommer, N., Gehrig, D., Loquercio, A., and Scaramuzza, D. (2020). Event-based asynchronous sparse convolutional networks. In *2020 European Conference on Computer Vision (ECCV)*, pages 415–431. Springer.

[140] Miao, S., Chen, G., Ning, X., Zi, Y., Ren, K., Bing, Z., and Knoll, A. (2019). Neuromorphic Vision Datasets for Pedestrian Detection, Action Recognition, and Fall Detection. *Frontiers in Neurorobotics*, 13.

[141] Milford, M., Kim, H., Leutenegger, S., and Davison, A. (2015). Towards visual slam with event-based cameras. In *The problem of mobile sensors workshop in conjunction with RSS*.

[142] Milford, M. J. and Wyeth, G. F. (2008). Mapping a Suburb With a Single Camera Using a Biologically Inspired SLAM System. *IEEE Transactions on Robotics*, 24(5):1038–1053.

[143] Milford, M. J. and Wyeth, G. F. (2012). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE International Conference on Robotics and Automation*, pages 1643–1649.

[144] Minsky, M. L. and Papert, S. A. (1988). *Perceptrons: expanded edition*. MIT Press, Cambridge, MA, USA.

[145] Mitrokhin, A., Fermüller, C., Parameshwara, C., and Aloimonos, Y. (2018). Event-Based Moving Object Detection and Tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9.

[146] Mitrokhin, A., Ye, C., Fermüller, C., Aloimonos, Y., and Delbruck, T. (2019). EV-IMO: Motion Segmentation Dataset and Learning Pipeline for Event Cameras. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6105–6112.

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

[148] Niu, J., Zhong, S., and Zhou, Y. (2024). IMU-Aided Event-based Stereo Visual Odometry. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11977–11983.

[149] Novo, A., Lobon, F., Garcia de Marina, H., Romero, S., and Barranco, F. (2024). Neuromorphic perception and navigation for mobile robots: a review. *ACM Computing Surveys*, 56(10):1–37.

[150] Oliveira, L. F., Moreira, A. P., and Silva, M. F. (2021). Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics*, 10(2):52.

[151] Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9.

[152] Papadopoulos, E., Aghili, F., Ma, O., and Lampariello, R. (2021). Robotic manipulation and capture in space: A survey. *Frontiers in Robotics and AI*, 8:686723.

[153] Parameshwara, C. M., Sanket, N. J., Singh, C. D., Fermüller, C., and Aloimonos, Y. (2021). 0-mms: Zero-shot multi-motion segmentation with a monocular event camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9594–9600. IEEE.

[154] Patil, D., Ansari, M., Tendulkar, D., Bhatlekar, R., Pawar, V. N., and Aswale, S. (2020). A Survey On Autonomous Military Service Robot. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–7.

[155] Perot, E., de Tournemire, P., Nitti, D., Masci, J., and Sironi, A. (2020). Learning to Detect Objects with a 1 Megapixel Event Camera. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, volume 33, pages 16639–16652. Curran Associates, Inc.

[156] Plou, C., Gallego, N., Sabater, A., Urcola, P., Montijano, E., Montesano, L., Martinez-Cantin, R., and Murillo, A. C. (2025). EventSleep: Sleep Activity Recognition with Event Cameras. In *2024 European Conference on Computer Vision (ECCV) Workshops*, pages 52–69, Cham. Springer Nature Switzerland.

[157] Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS. *IEEE Journal of Solid-State Circuits*, 46(1):259–275.

[158] Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., and Delbruck, T. (2014). Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output. *Proceedings of the IEEE*, 102(10):1470–1484.

[159] Pretegiani, E. and Optican, L. M. (2017). Eye movements in Parkinson's disease and inherited parkinsonian syndromes. *Frontiers in Neurology*, 8:592.

[160] Prono, L., Bich, P., Boretti, C., Mangia, M., Pareschi, F., Rovatti, R., and Setti, G. (2025). A Multiply-And-Max/Min Neuron Paradigm for Aggressively Prunable Deep Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 36(8):14414–14427.

[161] Prophesee (2022). Prophesee evaluation kit 4 HD. https://docs.prophesee.ai/stable/hw/evk/evk4.html.

[162] Pérez-Cutiño, M., Eguíluz, A. G., Dios, J. M.-d., and Ollero, A. (2021). Event-based human intrusion detection in UAS using Deep Learning. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 91–100.

[163] Qi, J., Ma, L., Cui, Z., and Yu, Y. (2024). Computer vision-based hand gesture recognition for human-robot interaction: a review. *Complex & Intelligent Systems*, 10(1):1581–1606.

[164] Rebecq, H., Gehrig, D., and Scaramuzza, D. (2018). ESIM: An Open Event Camera Simulator. In *Proceedings of The 2nd Conference on Robot Learning*, pages 969–982. PMLR.

[165] Rebecq, H., Ranftl, R., Koltun, V., and Scaramuzza, D. (2019). Events-To-Video: Bringing Modern Computer Vision to Event Cameras. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3852–3861.

[166] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

[167] Roberts, L. (1963). *Machine Perception of Three-dimensional Solids*. Its Technical report. M.I.T. Lincoln Laboratory.

[168] Rodríguez-Gomez, J., Eguíluz, A. G., Martínez-de Dios, J., and Ollero, A. (2020). Asynchronous event-based clustering and tracking for intrusion monitoring in UAS. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8518–8524.

[169] Rodríguez-Gómez, J. P., Eguíluz, A. G., Martínez-De Dios, J. R., and Ollero, A. (2021). Auto-Tuned Event-Based Perception Scheme for Intrusion Monitoring With UAS. *IEEE Access*, 9:44840–44854.

[170] Rosten, E., Porter, R., and Drummond, T. (2010). Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119.

[171] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*, pages 2564–2571. IEEE.

[172] Ryan, C., Elrasad, A., Shariff, W., Lemley, J., Kielty, P., Hurney, P., and Corcoran, P. (2023). Real-Time Multi-Task Facial Analytics With Event Cameras. *IEEE Access*, 11:76964–76976.

[173] Samadzadeh, A., Far, F. S. T., Javadi, A., Nickabadi, A., and Chehreghani, M. H. (2021). Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction. *Neural Process Letters*, 55:6979–6995.

[174] Schaefer, S., Gehrig, D., and Scaramuzza, D. (2022). AEGNN: Asynchronous Event-based Graph Neural Networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12361–12371.

[175] Sebesta, K. and Baillieul, J. (2012). Animal-Inspired Agile Flight Using Optical Flow Sensing. *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3727–3734.

[176] Shariff, W., Dilmaghani, M. S., Kielty, P., Moustafa, M., Lemley, J., and Corcoran, P. (2024). Event Cameras in Automotive Sensing: A Review. *IEEE Access*, 12:51275–51306.

[177] Shi, J. et al. (1994). Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE.

[178] Shi, X., Huang, Z., Li, D., Zhang, M., Cheung, K. C., See, S., Qin, H., Dai, J., and Li, H. (2023). FlowFormer++: Masked Cost Volume Autoencoding for Pretraining Optical Flow Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1599–1610.

[179] Shoemaker, P. A., Hyslop, A. M., and Humbert, J. S. (2011). Optic Flow Estimation on Trajectories Generated by Bio-Inspired Closed-Loop Flight. *Biological Cybernetics*, 104(4):339–350.

[180] Shrestha, S. B. and Orchard, G. (2018). SLAYER: Spike layer error reassignment in time. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*, pages 1419–1428.

[181] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*.

[182] Singh, K. J., Kapoor, D. S., Thakur, K., Sharma, A., et al. (2022). Computer-vision based object detection and recognition for service robot in indoor environment. *Computers, Materials & Continua*, 72(1).

[183] Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., and Benosman, R. (2018). HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1731–1740. IEEE.

[184] Sohal, J., Kumar, S., Vijay, S. A. A., Garg, P., et al. (2024). The Role of Computer Vision in Automating Defect Detection in Manufacturing. In *2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG)*, pages 1–6. IEEE.

[185] Sokolova, M. and Lapalme, G. (2009). A Systematic Analysis of Performance Measures for Classification Tasks. *Information Processing & Management*, 45(4):427–437.

[186] Srinivasan, M. V., Moore, R. J. D., Thurrowgood, S., Soccol, D., and Bland, D. (2012). From Biology to Engineering: Insect Vision and Applications to Robotics. In Barth, F. G., Humphrey, J. A. C., and Srinivasan, M. V., editors, *Frontiers in Sensing: From Biology to Engineering*, pages 19–39. Springer.

[187] Stoffregen, T., Scheerlinck, C., Scaramuzza, D., Drummond, T., Barnes, N., Kleeman, L., and Mahony, R. (2020). Reducing the Sim-to-Real Gap for Event Cameras. In *2020 European Conference on Computer Vision (ECCV)*, page 534–549. Springer-Verlag.

[188] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580.

[189] Suh, Y., Choi, S., Ito, M., Kim, J., Lee, Y., Seo, J., Jung, H., Yeo, D.-H., Namgung, S., Bong, J., Yoo, S., Shin, S.-H., Kwon, D., Kang, P., Kim, S., Na, H., Hwang, K., Shin, C., Kim, J.-S., Park, P. K. J., Kim, J., Ryu, H., and Park, Y. (2020). A 1280×960 Dynamic Vision Sensor with a 4.95-m Pixel Pitch and Motion Artifact Minimization. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5.

[190] Sun, Y., Shen, L., Wu, S., Yan, H., and Sheng, F. (2024). Research on Computer Vision-Based Environment Perception and Decision-Making System for Autonomous Vehicles. In *2024 14th International Conference on Information Technology in Medicine and Education (ITME)*, pages 82–85. IEEE.

[191] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

[192] Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

[193] Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks*, 111:47–63.

[194] Teed, Z. and Deng, J. (2020). RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *2020 European Conference on Computer Vision (ECCV)*, pages 402–419. Springer International.

[195] Terven, J., Córdova-Esparza, D.-M., and Romero-González, J.-A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine learning and knowledge extraction*, 5(4):1680–1716.

[196] Tian, Y., Ye, Q., and Doermann, D. (2025). YOLOv12: Attention-Centric Real-Time Object Detectors. *arXiv preprint arXiv:2502.12524*.

[197] Tomy, A., Paigwar, A., Mann, K. S., Renzaglia, A., and Laugier, C. (2022). Fusing Event-based and RGB camera for Robust Object Detection in Adverse Conditions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 933–939.

[198] Tonsen, M., Zhang, X., Sugano, Y., and Bulling, A. (2016). Labelled Pupils in the Wild: A Dataset for Studying Pupil Detection in Unconstrained Environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, pages 139–142. Association for Computing Machinery.

[199] Tresilian, J. R. (1995). Perceptual and Cognitive Processes in Time-to-Contact Estimation: Analysis of Prediction-Motion and Relative Judgment Tasks. *Perception & Psychophysics*, 57(2):231–245.

[200] Tussyadiah, I. (2020). A review of research into automation in tourism: Launching the Annals of Tourism Research Curated Collection on Artificial Intelligence and Robotics in Tourism. *Annals of Tourism Research*, 81:102883.

[201] Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., and Brox, T. (2017). DeMoN: Depth and Motion Network for Learning Monocular Stereo. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5622–5631.

[202] Vagnoni, E., Andreanidou, V., Lourenco, S. F., and Longo, M. R. (2017). Action Ability Modulates Time-to-Collision Judgments. *Experimental Brain Research*, 235(9):2729–2739.

[203] Vaila, R., Chiasson, J., and Saxena, V. (2019). Feature Extraction using Spiking Convolutional Neural Networks. In *Proceedings of the International Conference on Neuromorphic Systems (ICONS '19)*, pages 1–8.

[204] Varghese, R. and M., S. (2024). YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pages 1–6.

[205] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ukasz Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[206] Verma, A. A., Chakravarthi, B., Vaghela, A., Wei, H., and Yang, Y. (2024). etram: Event-based traffic monitoring dataset. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22637–22646.

[207] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition (CVPR)*, volume 1, pages I–I. IEEE.

[208] Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018(1):7068349.

[209] Walters, C. and Hadfield, S. (2021). Evreflex: Dense time-to-impact prediction for event-based obstacle avoidance. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1304–1309. IEEE.

[210] Wang, T., Chen, B., Zhang, Z., Li, H., and Zhang, M. (2022). Applications of machine vision in agricultural robot navigation: A review. *Computers and Electronics in Agriculture*, 198:107085.

[211] Wang, X., Huang, J., Wang, S., Tang, C., Jiang, B., Tian, Y., Tang, J., and Luo, B. (2024a). Long-term frame-event visual tracking: Benchmark dataset and baseline. *arXiv e-prints*.

[212] Wang, X., Wang, S., Tang, C., Zhu, L., Jiang, B., Tian, Y., and Tang, J. (2024b). Event Stream-Based Visual Object Tracking: A High-Resolution Benchmark Dataset and A Novel Baseline. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19248–19257.

[213] Wang, Y. and Bi, X. (2024). The Application of Computer Vision Target Recognition Technology in Autonomous Driving. In *2024 3rd International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS)*, pages 519–524.

[214] Wang, Y., Du, B., Shen, Y., Wu, K., Zhao, G., Sun, J., and Wen, H. (2019). EV-gait: Event-based robust gait recognition using dynamic vision sensors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 6358–6367.

[215] Wang, Y. and Frost, B. J. (1992). Time to Collision Is Signalled by Neurons in the Nucleus Rotundus of Pigeons. *Nature*, 356(6366):236–238.

[216] Wang, Z., Gao, C., Wu, Z., Conde, M. V., Timofte, R., Liu, S.-C., Chen, Q., Zha, Z.-j., Zhai, W., Han, H., Liao, B., Wu, Y., Wan, Z., Wang, Z., Cao, Y., Tan, G., Chen, J., Pei, Y. R., Brüers, S., Crouzet, S., McLelland, D., Coenen, O., Zhang, B., Gao, Y., Li, J., So, H. K.-H., Bich, P., Boretti, C., Prono, L., Lică, M., Dinucu-Jianu, D., Grîu, C., Lin, X., Ren, H., Cheng, B., Zhang, X., Vial, V., Yezzi, A., and Tsai, J. (2024c). Event-Based Eye Tracking. AIS 2024 Challenge Survey. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 5810–5825.

[217] Wang, Z., Ng, Y., Scheerlinck, C., and Mahony, R. (2021). An Asynchronous Kalman Filter for Hybrid Event Cameras. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 438–447.

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

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

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

[221] Wen, J., He, L., and Zhu, F. (2018). Swarm Robotics Control and Communications: Imminent Challenges for Next Generation Smart Logistics. *IEEE Communications Magazine*, 56(7):102–107.

[222] Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[223] Wiesel, T. N. and Hubel, D. H. (1963). Effects of visual deprivation on morphology and physiology of cells in the cat's lateral geniculate body. *Journal of neurophysiology*, 26(6):978–993.

[224] Woo, S., Park, J., Lee, J.-Y., and Kweon, I. S. (2018). CBAM: Convolutional Block Attention Module. In *2018 European Conference on Computer Vision (ECCV)*, pages 3–19. Springer International Publishing.

[225] Xing, Y., Di Caterina, G., and Soraghan, J. (2020). A New Spiking Convolutional Recurrent Neural Network (SCRNN) With Applications to Event-Based Hand Gesture Recognition. *Frontiers in Neuroscience*, 14.

[226] Xu, H., Zhang, J., Cai, J., Rezatofighi, H., and Tao, D. (2022). GMFlow: Learning Optical Flow via Global Matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8121–8130.

[227] Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Yu, F., Tao, D., and Geiger, A. (2023). Unifying Flow, Stereo and Depth Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13941–13958.

[228] Yang, X. and Yang, X. (2025). DriveGazen: Event-Based Driving Status Recognition using Conventional Camera. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 9283–9291.

[229] Ye, C., Mitrokhin, A., Fermüller, C., Yorke, J. A., and Aloimonos, Y. (2020). 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)*, pages 5831–5838. IEEE.

[230] Yin, B., Corradi, F., and Bohté, S. M. (2021). Accurate and Efficient Time-Domain Classification with Adaptive Spiking Recurrent Neural Networks. *Nature Machine Intelligence*, 3(10):905–913.

[231] Yuan, C., Xiong, B., Li, X., Sang, X., and Kong, Q. (2022). A novel intelligent inspection robot with deep stereo vision for three-dimensional concrete damage detection and quantification. *Structural Health Monitoring*, 21(3):788–802.

[232] Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. (2019). Q8BERT: Quantized 8Bit BERT. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, pages 36–39.

[233] Zhang, J., Yu, X., Sier, H., Zhang, H., and Westerlund, T. (2025). Event-based Sensor Fusion and Application on Odometry: A Survey. In *2025 IEEE 6th International Conference on Image Processing, Applications and Systems (IPAS)*, volume CFP2540Z-ART, pages 1–6.

[234] Zhang, W. and Li, P. (2020). Temporal Spike Sequence Learning via Backpropagation for Deep Spiking Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 12022–12033.

[235] Zhao, C., Sun, Q., Zhang, C., Tang, Y., and Qian, F. (2020). Monocular Depth Estimation Based on Deep Learning: An Overview. *Science China Technological Sciences*, 63(9):1612–1627.

[236] Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., and Chen, J. (2024). DETRs Beat YOLOs on Real-time Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974.

[237] Zhaodan Kong, Ozcimder, K., Fuller, N., Greco, A., Theriault, D., Zheng Wu, Kunz, T., Betke, M., and Baillieul, J. (2013). Optical flow sensing and the inverse perception problem for flying bats. *52nd IEEE Conference on Decision and Control*, pages 1608–1615.

[238] Zheng, J., Zhang, T., Wang, C., Xiong, M., and Xie, G. (2022). Learning for Attitude Holding of a Robotic Fish: An End-to-End Approach With Sim-to-Real Transfer. *IEEE Transactions on Robotics*, 38(2):1287–1303.

[239] Zheng, X., Liu, Y., Lu, Y., Hua, T., Pan, T., Zhang, W., Tao, D., and Wang, L. (2023). Deep learning for event-based vision: A comprehensive survey and benchmarks. *arXiv preprint arXiv:2302.08890*.

[240] Zhou, Y., Gallego, G., and Shen, S. (2021). Event-based stereo visual odometry. *IEEE Transactions on Robotics*, 37(5):1433–1450.

[241] Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2018). EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*.

[242] Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2019). Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 989–997.

[243] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.

[244] Zihao Zhu, A., Atanasov, N., and Daniilidis, K. (2017). Event-based visual inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5391–5399.

[245] Zihao Zhu, A., Yuan, L., Chaney, K., and Daniilidis, K. (2018). Unsupervised event-based optical flow using motion compensation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 711–714. Springer.

[246] Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*, 111(3):257–276.