TRABAJO FIN DE MÁSTER

# Enhancing Event-Based Vision for Logistics: Data Processing, Prediction Models Analisys and 3D Dataset Creation

**TRABAJO FIN DE MÁSTER** PARA LA OBTENCIÓN DEL TÍTULO DE **GRADUADO EN ROBÓTICA Y AUTOMÁTICA**

JUNIO **2025**

**Tomás Sánchez Villaluenga**

**DIRECTORES DEL TRABAJO FIN DE MÁSTER:**

**Pascual Campoy Cervera**

**Pedro Arias Pérez**

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

*This project was developed during a research stay in Dortmund, Germany. Conducted entirely under the supervision of the Chair of Material Handling and Warehousing (FLW) at Technische Universität Dortmund. In particular, the work was carried out under the academic guidance of Shrutarv Awasthi, who served as the principal supervisor there, along with the collaboration and support of the laboratory team throughout the development of the thesis.*

# RESUMEN EJECUTIVO

La visión basada en eventos ofrece ventajas en entornos dinámicos gracias a su adquisición de datos asíncrona y procesamiento de baja latencia. Esta tesis presenta un análisis detallado del algoritmo de Detección de Objetos con Etiquetado Eficiente (LEOD), evaluando su rendimiento sobre un conjunto de datos logísticos personalizado. Para garantizar la compatibilidad, LEOD ha sido modificado, abordando ineficiencias heredadas y optimizando su manejo de datos basados en eventos en aplicaciones logísticas.

Una parte fundamental de este trabajo consiste en la automatización del preprocesamiento y la anotación de datos, agilizando la creación de conjuntos de datos, la generación de etiquetas y la representación de eventos. Estas mejoras reducen la intervención manual y mejoran la reproducibilidad, mientras que las optimizaciones en el almacenamiento de datos disminuyen los requerimientos de espacio para anotaciones en un 90%, permitiendo una gestión más eficiente de conjuntos de datos a gran escala.

Los resultados experimentales confirman que LEOD cumple con las expectativas en aprendizaje semisupervisado y supera los valores de referencia previos en Detección de Objetos Débilmente Supervisada (WSOD), logrando una mayor precisión de detección que los conjuntos de datos *Gen1* y *1Mpx* en todos los tamaños de entrenamiento.

Además, esta investigación desarrolla un conjunto de datos 3D utilizando la tecnología de Campos de Radiancia Neuronal (NeRF), reconstruyendo entornos a partir de datos RGB para extender la percepción basada en eventos al espacio tridimensional. Las investigaciones futuras se centrarán en la integración de LEOD con percepción 3D basada en eventos y en la mejora de técnicas de segmentación para una mayor precisión en la detección.

### Palabras clave

Visión basada en eventos, Aprendizaje semi-supervisado, Aprendizaje auto-supervisado, Detección de objetos, Logística inteligente, Pseudoetiquetas, Reconstrucción 3D

### Código UNESCO

1203.04 Visión por computador

1203.09 Aprendizaje automático

3311.01 Ciencia y tecnología de sensores

3325.01 Ingeniería de control y sistemas inteligentes

3313.02 Robótica

3308.11 Eficiencia energética

# ABSTRACT

Event-based vision offers advantages in dynamic environments due to its asynchronous data acquisition and low-latency processing. This thesis presents an in-depth analysis of the Label-Efficient Object Detection (LEOD) algorithm, evaluating its performance on a custom logistics dataset. To ensure compatibility, LEOD has been modified, addressing inherited inefficiencies and optimizing its handling of event-based data in logistics applications.

A critical part of this work involves the automation of data preprocessing and annotation, streamlining dataset creation, label assignment, and event representation. These enhancements reduce manual intervention and improve reproducibility, while optimizations in data storage cut annotation space requirements by 90%, making large-scale dataset handling more efficient.

Experimental results confirm that LEOD meets expectations in semi-supervised learning and outperforms previous benchmarks in Weakly-Supervised Object Detection (WSOD), achieving higher detection precision than the *Gen1* and *1Mpx* datasets across all training sizes.

Additionally, this research develops a 3D dataset using Neural Radiance Fields (NeRF) technology, reconstructing environments from RGB data to extend event-based perception into three dimensions. Future research will focus on integrating LEOD with 3D event-based perception and improving segmentation techniques for enhanced detection precision.

### Key Words

Event-based vision, Semi-supervised learning, Self-supervised learning, Object detection, Smart logistics, Pseudolabels, 3D reconstruction

### UNESCO Code

1203.04 Computer Vision

1203.09 Machine Learning

3311.01 Sensor Science and Technology

3325.01 Control Engineering and Intelligent Systems

3313.02 Robotics

3308.11 Energy Efficiency

# TABLE OF CONTENT

# LIST OF TABLES

# LIST OF FIGURES

# GLOSSARY

**FLW** Lehrstuhls für Förder und Lagerwesen (Specific laboratory at TU Dortmund)

**RVT** Recurrent Vision Transformer

**LEOD** Learnable Event Object Detection

**WSOD** Weakly Supervised Object Detection

**SSOD** Semi-Supervised Object Detection

**2D** Two-Dimensional

**3D** Three-Dimensional

**API** Application Programming Interface

**DVS** Dynamic Vision Sensor

**fps** Frames Per Second

**dB** Decibels

**RGB** Red Green Blue

**RGB-D** RGB + Depth

**SIFT** Scale-Invariant Feature Transform

**SURF** Speeded-Up Robust Features

**SVM** Support Vector Machine

**CNN** Convolutional Neural Network

**R-CNN** Region-based Convolutional Neural Network

**LSTM** Long Short-Term Memory

**SLAM** Simultaneous Localization and Mapping

**RAM** Random Access Memory

**ROM** Read-Only Memory

**CUDA** Compute Unified Device Architecture

**HDF5** Hierarchical Data Format version 5

**SAM2** Segment Anything Model v2

**AOD** Agentic Object Detection

**LLM** Large Language Model

**BLOSC** Blocked Serial Compression

**NeRF** Neural Radiance Fields

**GT** Ground Truth

**AP** Average Precision

**mAP** Mean Average Precision

**AP-50** Average Precision at 50% IoU

**AP-75** Average Precision at 75% IoU

**AP-L** Average Precision for Large Objects

**AP-M** Average Precision for Medium Objects

**AP-S** Average Precision for Small Objects

**RVT-s** Recurrent Vision Transformer - Small

**RVT-b** Recurrent Vision Transformer - Base

**RVT-t** Recurrent Vision Transformer - Tiny

# 1 INTRODUCTION

The present project is a continuation of previous work carried out in the Lehrstuhls für Förder und Lagerwese (FLW) laboratory, focused on the study of RVT algorithm [1]. In this study, RGB and event cameras were used to capture information about the scenes and record the positions of the cameras and logistic elements within the shooting space.

The process of labeling the obtained data was carried out using a Roboflow model, which generated detections from event reconstructions accumulated in predefined time intervals. To facilitate its application, the Roboflow API was used, allowing a more structured workflow in the detection and annotation of elements present in the scenes.

## 1.1 Problem Statement

After an initial review of the methods used in the previous project, several limitations in data pre-processing were identified. First, a low level of process automation was detected, which generated inefficiencies and recurrent execution failures due to incomplete information in the code and the use of outdated and not fully functional code versions. In addition, it was evident that the segmentation model used for label generation introduced a high level of errors and noise in the annotations, which could lead to detection problems, false positives, and poor training of future models.

Another critical aspect identified was the complexity and time required for the creation of new scenes and pre-processing for the generation of data ready to train artificial intelligence models. The need to optimize this process is essential to reduce the costs of time and effort in the construction of new data sets, thus improving the overall efficiency of the workflow.

## 1.2 Objectives

The fundamental objectives of this project are to analyze the performance of the LEOD model compared to previous models. To do this, it is necessary to create a new method of pre-processing the data obtained so far, as well as seeking to automate it to save time and create new possibilities.

The objectives established for this project are as follows:

- **Optimize and automate data preprocessing** in order to improve label accuracy and reduce the time required for manual operation.
- **Develop a new three-dimensional dataset** that contains relevant and high-quality information for future applications in the field of machine learning. That would be an approximation to EventNeRF [2].
- **Analyze learning models with less supervision**, with the aim of obtaining good results by reducing the number of annotations required and, therefore, the effort of manual labeling.

## 1.3 Approach

To address the optimization and automation of data preprocessing, a process of improvement and unification of the previously used scripts has been carried out. Errors have been corrected, improvements have been made to file compression to optimize storage space, and procedures have been modified to minimize the required manual interaction. These improvements aim to ensure a more efficient and reproducible workflow, thus reducing the possibility of errors and failures in the execution of pre-processing.

In relation to the creation of a new dataset, new scenes have been recorded to complement those obtained previously. The scenes generated in this project have focused on the use of the 'Zivid' box (product brand) carried by a person. Aditionally, variations in the composition of the scenes have been introduced, including the 'Zivid' box together with other elements carried by a cart ('wagen'), in the person's arms, or through an AGV (Autonomous Guided Vehicle). In addition, to improve the quality and precision of the data collected, information on the position of the box and the different parts of the human body (hands, feet, trunk and head) has been incorporated using marker technology detected by the Optitrack system[1].

Another key aspect of the methodological approach has been the analysis of strategies for segmenting elements in images. Segmentation has been studied from both RGB images and reconstructions generated from events, evaluating different segmentation models with the aim of identifying those that perform better in terms of accuracy and robustness. In addition, as a complement to the 2D dataset, the creation of a new 3D dataset has been proposed, incorporating three-dimensional models of the elements present in the scenes. This approach would improve the detection and segmentation of objects in scenarios that use multiple cameras or 3D sensors.

Finally, in the context of the analysis of learning models with less supervision, the LEOD model has been explored, which allows the generation of pseudo-labels from a reduced set of initial annotations. The application of this approach could significantly reduce the number of manual annotations needed for pre-processing, decreasing the time required for this task by more than 50%. In addition, a comparison will be made between the LEOD model and the RVT model, previously studied in the original project, in order to evaluate their respective advantages and limitations in the context of the present work.

## 1.4 Document Structure

- **Chapter 1** provides an introduction to the objectives, the reference frame, and the expected impact of this work.
- **Chapter 2** sets out the fundamentals and state of the art, reviewing the technology used in the project and the theoretical concepts necessary to understand the general framework.
- **Chapter 3** provides a brief analysis of the technology used in the project and its relationship with the current research scene and its progress over time. In addition, projects related to the framework of this work are analyzed.
- **Chapter 4** explains the internal characteristics of the LEOD model. It covers everything

[1] OptiTrack is an advanced motion capture technology based on infrared cameras that track reflective markers placed on people or objects.

from the options it offers, its configuration and how to get the most out of it, to the modifications that would have to be made to its code to adjust it to a specific dataset.

- **Chapter 5** details the methodology used to create the data sets (2D and 3D), explains the files needed to represent the event data in such a way that a learning model can work with them, the preprocessing of the information obtained from the recorded scenes, and the processes evaluated for the generation of annotations.
- **Chapter 6** presents an in-depth analysis of the improvements promised by LEOD, its relationship and differences with RVT, and the data obtained when applying the prediction model to the created dataset, showing the differences between WSOD (*Weakly-Supervised Object Detection*) and SSOD (*Semi-Supervised Object Detection*) mode.
- **Chapter 7** presents the conclusions derived from the completion of this work and the analysis carried out, as well as possible ideas generated as future development lines for the project.

- **Appendix A:** outlines the hardware and software pack required to replicate and setup the project.
- **Appendix B** lists dataset folder conventions, default training hyper-parameters for WSOD / SSOD and notes on how mean-AP evolves with different step counts.
- **Appendix C** gathers quantitative tables for multiple WSOD and SSOD configurations and highlights the accuracy trends for each data ratio.
- **Appendix D** gives a high-level snapshot of the project schedule (WBS), Gantt overview and a concise estimate of human effort and material costs.
- **Appendix E** summarizes the project's social-ethical footprint and environmental benefits, linking the work to current regulations and its contribution to selected UN 2030 Sustainable Development Goals.
- **Appendix F** links the public GitHub repository *LEOD_FLW*, which hosts all code, datasets and reproducibility instructions.

# 2 FUNDAMENTALS

## 2.1 Event-Based Sensors and Cameras: Concept and Features

Event cameras, also known as event-based vision sensors or Dynamic Vision Sensors (DVS), represent a disruptive innovation in computer vision. Unlike traditional cameras, which capture full images at fixed time intervals, event cameras operate asynchronously and only register changes in the visual scene, specifically variations in light intensity at individual sensor pixels. Their unique characteristics make them highly suitable for applications that require high-speed motion detection and efficient data processing.

Event cameras offer a novel approach to capturing visual information. Whereas conventional cameras rely on frame-based acquisition that often produces redundant data, event cameras detect and report individual intensity changes, enabling more efficient and real-time processing.

### Concept and Operating Principle

The fundamental principle of event cameras is based on how they capture visual information. Instead of producing a continuous stream of full images, these sensors detect and report discrete events. An event occurs when a significant change in light intensity is detected in a pixel. Each event is recorded as a spatial coordinate $(x, y)$, a timestamp $t$, and a polarity value ($+$ or $-$), indicating whether the intensity increased or decreased.

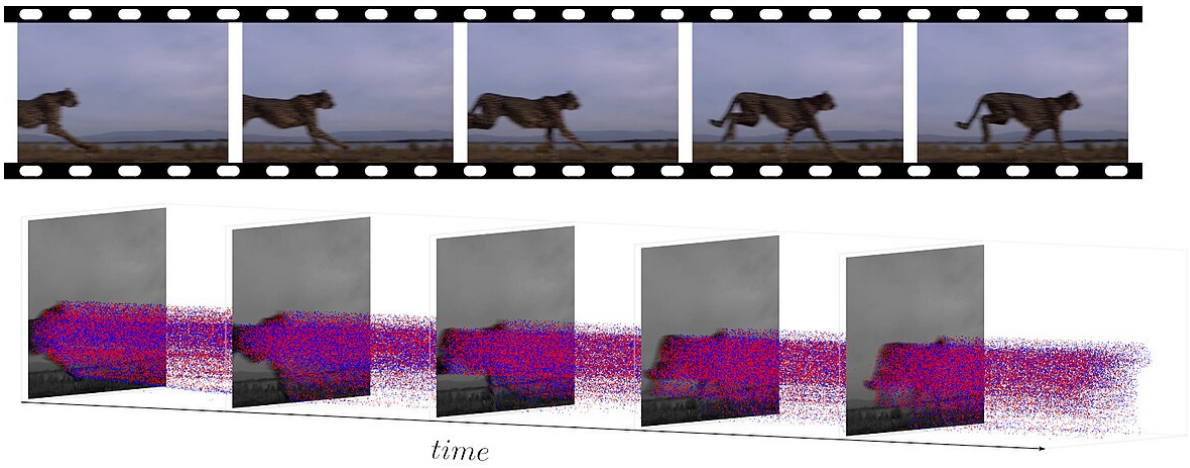



Figure 2.1: Comparison of data flow between a conventional camera and an event camera. [3]

### Key Features and Technical Aspects

Event cameras have several technical features that make them especially valuable for dynamic applications:

1. **High Temporal Resolution:** Event cameras can detect changes in the microsecond range, allowing them to capture extremely fast movements with high precision. This level of temporal resolution far exceeds that of conventional cameras, which typically operate at fixed frame rates, such as 30 or 60 fps.

2. **Low Power Consumption:** By processing only significant changes in a scene, event cameras generate substantially fewer data, reducing the need for computational resources and lowering energy consumption. This makes them ideal for battery-powered autonomous devices, such as drones and mobile robots.

3. **Extended Dynamic Range:** These cameras can operate with a dynamic range exceeding 120 dB, enabling them to operate in highly variable lighting conditions. Unlike conventional cameras, which may lose information in low-light environments, event cameras maintain high sensitivity and precision.

4. **Noise Reduction and High Efficiency:** Focusing exclusively on intensity changes allows event cameras to naturally filter out static noise, improving their performance in complex visual scenarios.

Figure 2.2: Event-based camera [4]

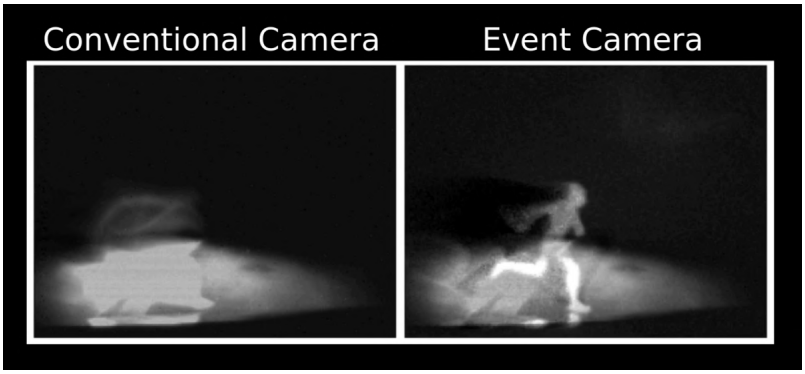



Figure 2.3: Comparison of information obtained in low-light scenarios [5][6]

### Limitations

Despite their advantages, event cameras present several challenges. Processing event-based data requires specialized algorithms that can handle asynchronous information. Additionally, as an emerging technology, their commercial adoption is still limited due to high initial costs and the need for further hardware development.

### Advantages in Robotics

Event cameras provide unique benefits that make them a preferred choice for mobile systems such as drones and autonomous robots. Due to their low power consumption, they improve battery efficiency, which is a critical factor in autonomous applications. Their high-speed response enables for real-time obstacle detection and navigation.

Another key advantage is their ability to operate effectively in variable lighting conditions. This feature makes them particularly useful in autonomous driving, where real-time perception is crucial. Although event cameras have been applied primarily in automotive vision systems, this study explores their potential use in logistics, aiming to optimize real-time inventory tracking and object recognition in dynamic environments.

## 2.2 Object Recognition and Detection: Traditional vs. Event-Based Methods

Object recognition and detection are fundamental in computer vision, with applications in robotics, logistics, and autonomous transportation. Traditionally, these tasks have relied on RGB cameras and frame-based analysis, but event cameras have introduced a new approach to dynamic scenarios. While techniques for RGB cameras have advanced significantly, algorithms specific to event cameras are still under development, presenting a challenge and a opportunity for research and industry.

### Traditional Methods for Detection and Recognition

RGB cameras have been the standard for computer vision for decades. These cameras capture static images or frame sequences at a fixed rate (30, 60 fps, etc.), and object analysis is performed by processing these images using either classical algorithms or modern deep learning techniques. Some of the traditional approaches include the following.

1. **Feature-Based Methods:** Techniques such as Scale-Invariant Feature Transform (*SIFT*) and Speeded-Up Robust Features (*SURF*) analyze key points in images and extract descriptors to identify and locate objects.
2. **Machine Learning Algorithms:** Classifiers like Support Vector Machines (SVM) and Random Forests are trained with specific features extracted from images.
3. **Convolutional Neural Networks (CNNs):** In recent years, CNNs have revolutionized the field, with models such as YOLO and Faster R-CNN leading in object detection and recognition due to their high precision and speed.

However, these methods have inherent limitations, particularly in scenarios where frame-based capture is inadequate, such as high-speed motion environments or adverse lighting conditions. Additionally, the constant processing of full images generates large volumes of data, increasing computational and energy demands.

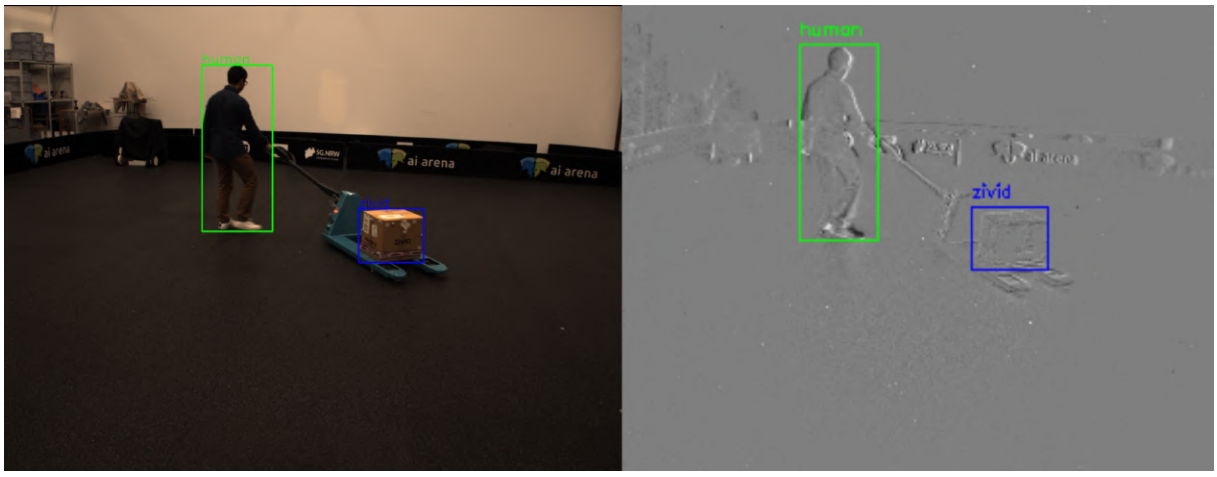



Figure 2.4: Comparison of traditional segmentation vs. event-based segmentation at the same time instant.

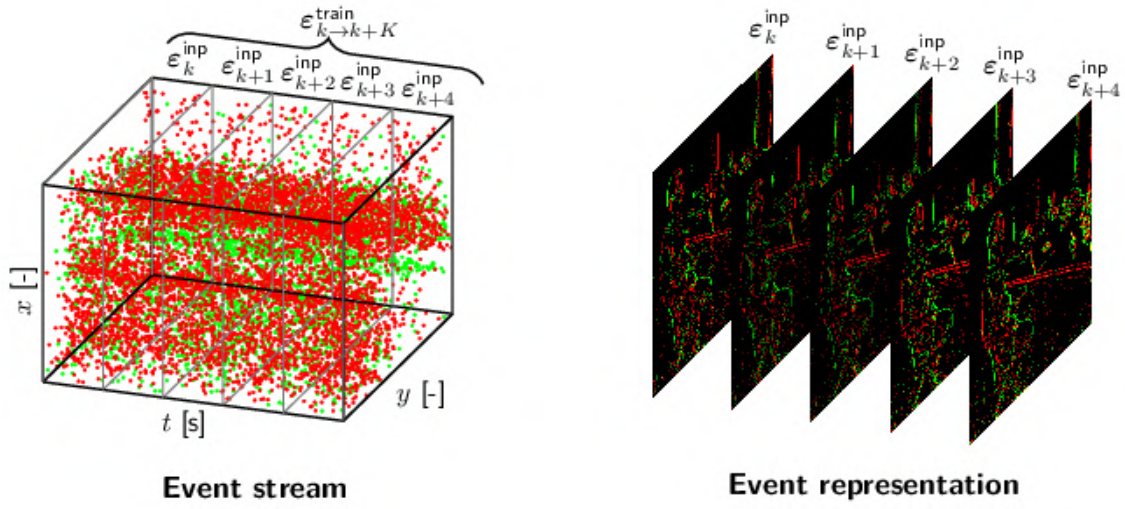
**Event-Based Object Detection Methods**

As explained in Section 2.1 and based on the literature review in Section 3, event-based segmentation methods are still in an early stage. Primary challenges include the lack of large annotated datasets, the absence of unified evaluation metrics, and the complexity of processing asynchronous data. These factors indicate that research in this area is still in a phase of consolidation and growth.

Current algorithms can be broadly categorized into two approaches:

1. **Event Accumulation Methods:** These methods aggregate events over time intervals to generate synthetic images, facilitating the application of existing learning techniques at the cost of losing temporal resolution.
2. **Asynchronous Processing Approaches:** These methods operate directly on the asynchronous event stream using neural networks, using the event structure for faster detection.

However, the scarcity of specialized datasets and the difficulty in designing robust networks still limit their practical application.




(a) Event stream (b) Event accumulation into synthetic images

Figure 2.5: Difference between types of event use. [7]

## 2.3 RVT: Recurrent Vision Transformer Algorithm

The Recurrent Vision Transformer algorithm [1] is a model designed to process data from event cameras, improving object detection in dynamic and real-time environments. Its architecture combines global and local attention mechanisms with a temporal memory structure, optimizing the analysis of asynchronous and high-temporal-resolution information.

### RVT Main Components

The model is based on three main components, designed to extract and process the spatial and temporal characteristics of event data efficiently:

1. **Convolutional Prioritization:** In the first stage, light convolutions are used to extract spatial features and generate a positional representation of events. This process facilitates the identification of relevant patterns before applying attention mechanisms.
2. **Dilated Self-Attention:** Incorporates a hierarchical attention scheme combining:
   - **Dilated Global Attention:** Captures large-scale spatial relationships without significantly increasing computational cost.
   - **Local Attention:** Focuses on detecting specific details and nearby relationships in the scene.

   The combination of both mechanisms allows for the analysis capability to balance between large-scale structures and local details.
3. **Recurrent Temporal Aggregation (LSTM):** To integrate the temporal dimension of events, the RVT uses a memory network based on Long Short-Term Memory (LSTM). This module allows combining information from past and current events, which is useful for tracking moving objects and interpreting dynamic sequences.

### Technical Advantages

The RVT design offers several advantages in detecting objects with event cameras:

- **Optimized Inference Time:** It can process information in less than 12 milliseconds on standard hardware, allowing its use in robotic systems and autonomous vehicles.
- **Detection Accuracy:** Has achieved a 47.2% mAP in tests on the Gen1 automotive dataset, with competitive performance compared to previous models.
- **Resource Efficiency:** Uses five times fewer parameters than similar models, reducing hardware requirements and facilitating deployment on resource-constrained devices.
- **Adaptability to Dynamic Scenarios:** Its combination of spatial attention and temporal memory allows it to detect multiple moving objects and operate in varying lighting conditions.

### Limitations and Improvement Areas

Despite its advantages, the RVT faces certain challenges in its implementation and application:

- **Model Complexity:** The integration of multiple advanced modules, such as hierarchical attention and recurrent memory, increases the difficulty of model design and tuning.
- **Training Data Dependency:** Its performance depends on varied and well-structured data sets. In data-limited scenarios, its generalization capability may suffer.

- **Applications in Other Domains:** Although it has shown good results in automotive environments, its performance in other areas, such as surveillance or industrial robotics, still requires experimental validation.

## 2.4 LEOD: Label-Efficient Object Detection Algorithm

Label-Efficient Object Detection [8] is a method designed for object detection in event camera data, with a focus on reducing the need for labeled data. To this end, it employs pseudo-labeling and self-training techniques, allowing the use of unannotated data and decreasing the dependence on manual labeling. Its operation is based on strategies such as soft anchor assignment, bidirectional use of time, and detection-based tracking, which aim to improve the accuracy of pseudo-labeling and optimize model performance.

In this context, LEOD aligns with partially supervised learning approaches such as Semi-Supervised Object Detection (*SSOD*) and Weakly-Supervised Object Detection (*WSOD*). SSOD is based on training the model with a combination of fully labeled and unlabeled sequences, generating pseudolabels in the latter. WSOD, on the other hand, distributes the labels sparsely over time, allowing the model to learn the continuity of moving objects without relying on dense labels at every instant.

### Key Concepts

In order for LEOD to achieve efficient training with less annotated data, there are several principles that allow it to maximize the use of available information. These principles are as follows.

- **Efficient Labeling:** LEOD minimizes the use of labeled data through self-training, a strategy especially useful in event cameras, where manual annotation of asynchronous data is costly and complex.
- **Pseudo-labeling:** A pre-trained detector generates estimated labels for unannotated data, expanding the available dataset. In SSOD, these pseudolabels reinforce unlabeled sequences, while in WSOD, they complement time-sparse supervision.
- **Iterative self-training:** The model is retrained in cycles, refining the pseudo-labels with each iteration and adjusting its parameters to improve accuracy on unlabeled data.

### Technical Innovations

To ensure that semi-supervised learning in LEOD is stable and effective, strategies are applied that reduce the impact of noise on pseudo-labels and strengthen the consistency of detections over time. Using advanced refinement and tracking techniques, the model improves its accuracy even with partially annotated data. These techniques are as follows.

- **Smooth Anchor Assignment:** LEOD dynamically adjusts anchor assignment to avoid errors in pseudo-labels, improving training stability.

- **Bidirectional Use of Time:** It analyzes events in both forward and backward directions, verifying the temporal consistency of pseudo-labels and reducing false positives.
- **Detection-Based Tracking:** A tracking module is incorporated that improves the continuity of detections over time, reinforcing coherence in SSOD, where labels are scarce.

#### Relationship to RVT

LEOD complements the Recurrent Vision Transformer (RVT) by focusing on labeling efficiency and semi-supervised learning. While RVT is designed to process real-time event data using a hierarchical attention scheme that optimizes feature extraction, LEOD improves model performance in scenarios with limited labeled data. This is achieved through the use of pseudo-labeling and semi-supervised learning strategies, allowing the model to generalize better without relying exclusively on large volumes of manually annotated data. Thus, while RVT prioritizes accuracy in event detection through its recurrent and self-attenuating architecture, LEOD optimizes the training process in partially annotated environments, facilitating its application in cases where label collection is costly or impractical.

# 3 LITERATURE REVIEW

## 3.1 Analysis

The study of event cameras has increased significantly in recent years. A total of 882 relevant documents were identified in the bibliographic search carried out in the *Web of Science* [9] database under the term "*event-based cameras*" in the abstract and title. Of these, 762 were published in the last five years. Likewise, when analyzing the evolution over time of scientific production, a linear growth can be seen in the last five years, acquiring a behavior that tends towards the exponential when considering the complete period from the first studies to the present. These figures reflect the rapid consolidation of the subject in the research community, as well as the growing interest it is arousing in multiple fields, particularly in computer vision, robotics, and artificial intelligence.

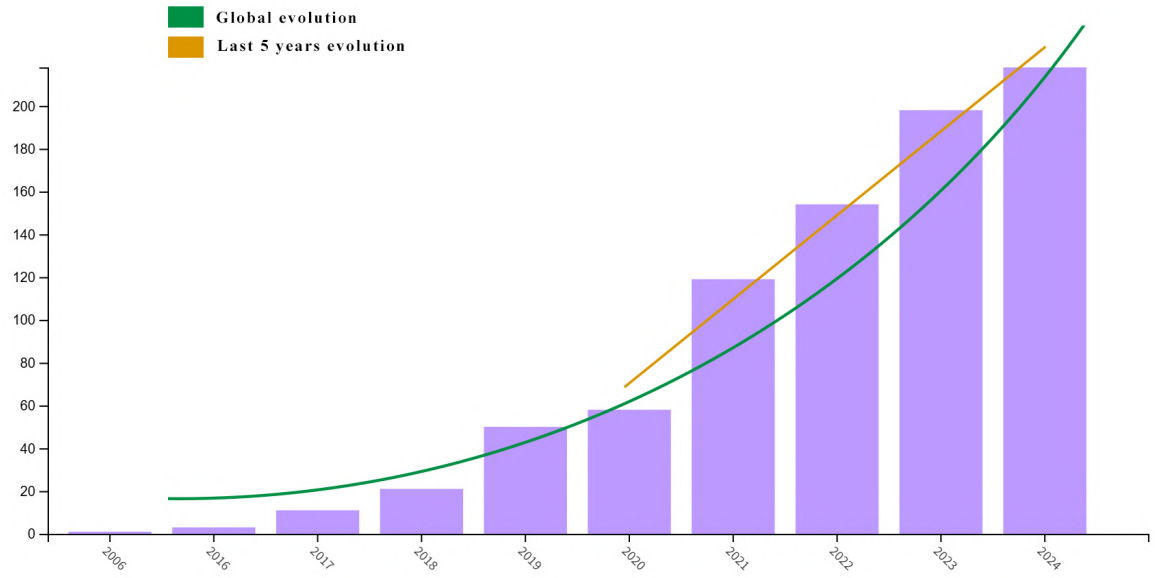



Figure 3.1: Evolution of the number of publications on event cameras. [9]

The first specialized articles on event cameras appeared around 2016. During that year, initial contributions were presented at prominent conferences in the field of computer vision, such as the Conference on Computer Vision and Pattern Recognition ($CVPR$), the Winter Conference on Applications of Computer Vision ($WACV$), and the European Conference on Computer Vision ($ECCV$). At these events, the first formal descriptions of the advantages of event cameras over traditional cameras were offered, along with basic algorithms to take advantage of their asynchronous nature.

Although this technology is in an early stage, some of the published papers already have a high number of references and citations. Some of these publications are as follows:

**"Event-Based Vision: A Survey" (2022) [10][11]**
This work, published in 2022, has established itself as the most cited reference to date, reaching 1123 citations and including 241 bibliographic references. It can be considered as the one that has laid the theoretical and technical foundations of this field. It provides an overview of event camera technology, describes its fundamentals and applications, and provides future lines of research. It also details technical aspects of data acquisition, event processing and evaluation methodologies.

**"The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM" (2017) [12][13]**
With 390 citations and 21 references, it represents a decisive contribution to the consolidation of research into event cameras. This work focuses on the creation of a dataset and a simulator specialized in the generation of event sequences, facilitating the validation and comparison of algorithms in pose estimation, visual odometry and SLAM tasks. It establishes evaluation methodologies that have served as a standard for subsequent studies.

**"Intelligent Machinery Fault Diagnosis With Event-Based Camera" (2024)[14][15]**
Recently published, this article has accumulated 75 citations and 35 references, reflecting the emerging interest in industrial areas. Its main contribution lies in the integration of event camera technology into machinery diagnostic systems, demonstrating improvements in fault identification through the continuous and high temporal resolution capture offered by these devices. This approach represents a significant advance in the field of smart industry and predictive maintenance.

**Conclusion**

From the analysis carried out, it can be concluded that, although research into event cameras has gained relevance at a remarkable pace, it is still at an early stage of development. The linear increase in publications over the last five years, added to the trend towards exponential expansion when considering the entire period, is indicative of the growing number of research groups that are directing their efforts towards this field. Special attention is being paid to the design of efficient algorithms that can process large volumes of event data in real time, as well as to the development of more robust and lower-cost hardware. The consolidation of evaluation methodologies, the emergence of software toolkits and the existence of public databases specially designed for working with this type of information are promoting the adoption of event cameras in different disciplines. Currently, research is increasingly oriented towards practical application: new possibilities are being explored to improve the detection and classification of objects in rapidly changing scenarios and the aim is to integrate the technology into industrial supervision systems, autonomous vehicles or drones. However, although there are already useful and functional implementations, progress continues to be made in understanding the limitations of event cameras, for example, in relation to the inherent noise of the sensor and the need for more robust processing algorithms in extreme lighting conditions.

## 3.2 Relevant articles in relation to the project

In addition to the RVT and LEOD algorithms, there are articles that can be used as a basis and help to understand part of the current project. These are as follows:

**"Quantitative Evaluation of a Multi-Modal Camera Setup for Fusing Event Data with RGB Images" [16]**
In this work, the authors propose a camera setup that combines two high-resolution event sensors with a global shutter RGB camera. The main motivation is to align and accurately fuse data from both types of sensors for computer vision tasks. They address issues of calibration and temporal synchronization (both at the hardware and software level) and analyze the accuracy of their results. The calibration achieves a very small alignment error, making it easier for conventional image-based vision tools to take advantage of event data without losing fidelity. This combination opens up opportunities for automatic label creation, object detection in difficult conditions and detailed analysis of high dynamic range scenes.
Furthermore, this is a good starting point for the present project. Based on this information, problems can be solved and the preprocessing of the information can be fine-tuned when creating the dataset.

**"EventNeRF: Continuous-time Neural Radiance Fields from a Moving Event Camera" [2]**
This paper addresses the challenge of reconstructing 3D scenes from event cameras, which asynchronously record changes in intensity. Unlike traditional cameras, which generate static images or discrete sequences, event cameras produce continuous streams of data every time a significant change in brightness is detected in a pixel. This work combines the asynchronous nature of these sensors with the flexibility of NeRF techniques, proposing a model that, instead of receiving specific frames, integrates the events in a continuous time representation. In this way, the geometry and appearance of the scene can be estimated simultaneously, even in conditions where conventional frame cameras encounter limitations (for example, low-light environments or very fast movements).
The main contribution of EventNeRF lies in its ability to continuously update the radiance and density of each point in space, taking into account the high temporal frequency of events. Thus, the method reconstructs the scene as an implicit field trained by ray tracing that is compared with the recorded events, adjusting an MLP that describes the density and color. The result is a precise and robust 3D reconstruction in scenarios of movement and changes in lighting, opening up new possibilities in applications such as augmented reality, robotics and artificial vision, where the speed of capture and tolerance to environments with a high dynamic range are crucial.

# 4 METHODOLOGY

## 4.1 Label-Efficient Object Detection (LEOD)

The objective of this project is to analyze machine learning models for the prediction of 2D images, with the aim of identifying the most suitable model for working with the available data set. To this end, the research focuses on the analysis of the LEOD model and its comparison with other alternatives. LEOD was developed to address the limitation of event-based object detection methods, which depend on large volumes of manually annotated data. Since labeling this data is impractical on a large scale, LEOD seeks to make the most of available labels and improve performance with unlabeled data. It is the first method to integrate self-training strategies into event-based detection, maximizing the use of unannotated data. In terms of its scope of use, it is a general event-based object detection algorithm. Although its evaluation has focused on traffic objects, such as cars and pedestrians. It was tested on two specific data sets:

- **Gen1**, which contains urban traffic scenes with annotations of cars and pedestrians.
- **1Mpx**, which includes a wider range of environmental conditions and more object classes, such as bicycles and motorcycles.

Although it has been implemented and evaluated in the context of vehicles and pedestrians, the algorithm framework is applicable to other domains provided that event camera data is available.

### 4.1.1 Processes offered

#### Model training

In this phase, the model learns to identify events in time series from labeled data. Its parameters are optimized using a training set, and the model weights are stored for later use.

Throughout the training, the model optimizes its parameters using backpropagation and gradient descent algorithms. During this process, different *checkpoints* of the model are saved at different moments of the optimization, which allows previous versions to be recovered if necessary.

**Requirements:** Annotated data.
**Output:**

- Model trained with optimized parameters.
- Model weights saved in checkpoints (`.ckpt`).
- Performance metrics such as accuracy (AP) during training.
- Records of loss and learning evolution for later analysis.

### Training with Pseudo-labels

If the model is being used for the first time with pseudo-labels, changes must be made to the training model. In this case, `model=rnndet-soft` will be used instead of `model=rnndet`. This model is used in the self-learning process and allows working with **soft labels**, which are probabilistic labels instead of rigid binary values.

In self-learning, the model can make mistakes when generating pseudo-labels. If hard labels such as "*rnndet*" (fixed and deterministic labels) are used, incorrect information could be propagated in successive iterations, making it difficult to improve the model. In contrast, with soft labels (*rnndet-soft*), the model weights the predictions with a degree of confidence, allowing instances with high certainty to have greater influence in the next round of training.

### Pseudo-label generation

When there are insufficient annotated data, the trained model is used to predict events in unlabeled data, generating pseudolabels that can be used in future training iterations. Although these pseudo-labels may contain a certain margin of error, their iterative use in new training rounds contributes to progressively improving the model's performance.

**Requirements:** Previously trained model, checkpoint of the trained model and "*ratio*" or "*train_ratio*" configuration (depending on *WSOD* or *SSOD*).
**Output:**

- New dataset with the pseudo-labels.[a]

[a]The only file is not modified is `event_representations.h5`.

### Model Evaluation

Once a model has been trained, it is essential to evaluate its performance to determine how well it detects events in the time series. In this phase, the model is tested on a validation dataset, and its predictions are compared with the actual labels.

During the evaluation, key metrics are generated such as the average precision (AP), which indicates the proportion of correctly detected events. In addition, specific errors can be analyzed, such as false positives (incorrect detections) and false negatives (omitted events).

**Requirements:** Trained model and test data (val) with real annotations.
**Output:**

- Quantitative model accuracy metrics (*AP, AP-50, AP-75, AP-L, AP-M, AP-S* )

### Results Visualization

Viewing the results allows the model's predictions to be inspected manually and their accuracy to be verified. Through automatically generated graphs and videos, it is possible to observe the

events detected by the model in the time series and compare them with the real annotations.

**Requirements:** Model weights saved as checkpoints (`.ckpt`) and the number of videos we want to obtain.
**Output:**

- Videos with a visual comparison of predictions and actual annotated graphs, allowing for more precise adjustments.

### 4.1.2 Configuration, implementation and operation

To use LEOD, you must follow the installation instructions available at the *LEOD Github Repository*[17]. The repository is structured in 11 folders, each with specific functions, although some are more relevant than others. The function of each folder and file will be described below, as well as the possible modifications required to adapt the model to a dataset of your own.[2]

#### 4.1.2.1 Model configuration

When introducing a dataset for the first time in LEOD, some of the information that the new dataset works with must be understood. Therefore, the main modifications and steps to be followed to ensure that the model understands it and can work with it are explained below.

These are the main steps needed to run the algorithm with external datasets masking it as one of the pre-established ones (*Gen1* or *Gen4*). If you need to adapt all the algorithm code to the new dataset, follow the steps explained in Section 4.1.3.

**Dataset**

Once the dataset has been configured, all that remains is to tell the model where it is located. To do this, the "*path*" parameter of the file "`/config/dataset/genX.yaml`" is modified with the path of the main folder of the dataset used. It doesn't really matter which folder it is in as long as it is indicated in the *config* directory. The dataset should be divided into 3 folders: val, test and train.

**`.pkl` file**

This type of file stores the encoding and configuration of labels, features from raw data and metadata about the dataset. This file is necessary for the model to understand the dataset, and will be created at the start of model training if it does not exist.

[2]The model can actually be used with any dataset just by changing that is indicated in Section 4.1.2.1 and the `path` in `/config/dataset`. Executing it with the name **gen1** (as long as it has only two types of labels) or **gen4** (as long as it has pedestrian, cycle and car types of labels). In this project a new repository has been created where everything is adjusted for a new dataset called "*flw_dataset*".

Therefore, the first time it is trained or if the dataset data is modified (partial information is added, deleted or modified), it is necessary to delete the file "`ssod_XX-off0.pkl`" from the folder `/data/genx_utils/splits/dataset_name`[3].

### Label identification

Depending on the dataset and the elements to be predicted, there will be a number and types of classes of labels or other classes. Therefore, these must be recognized by their names indicated in the annotations to make it possible for the model to understand them and subsequently visualize them.

The files that need to know these names are:

`utils/evaluation/prophesee/evaluation.py`: Where the exact names of the classes indicated in the annotations must be indicated with the same values given in them. For example, if it has been indicated that "human" is identified as 0 and "pallet" as 1, it should be indicated in the form "*classes = ("human", 'pallet')*".

`utils/evaluation/prophesee/evaluator.py`: The same steps should be followed as in the previous case for the "evaluation.py" file, but in this case, instead of entering the full name as indicated in the annotations, an abbreviation should be given. For example, if it has been indicated that "human" is identified as 0 and "pallet" as 1, it should be indicated in the form: *("hum", 'pal')*.

`/utils/evaluation/prophesee/visualize/vis_utils.py`: The exact same steps are followed as in the case of *evaluation.py* for the corresponding LABELMAP.

### Checkpoints

In case that the model is to be retrained with the same dataset and the same high-level parameters (steps, model, dataset name, radius), it is necessary to delete the folder `./checkpoint/rnndet_YY`[4]. Otherwise, the model will understand that the model is already trained with that configuration and will not retrain it.

[3]Wherec"*XX*" corresponds to the radius with which the model is trained.
[4]Where "*YY*" represents specific values of the configuration of the finished training

### 4.1.3 File management

When adjusting the code of the LEOD algorithm for use with a specific new dataset, the number of classes, the exact names of the classes implemented in their annotations and the dimensions of the images used must first be known. Once this is known, it is necessary to modify and create several files that fit this new dataset. The main folders that are interesting to know about when working with this model are the following:

*First of all, we must bear in mind that we must always use the same name for the new dataset to be configured.*

**CREATE: config/dataset/*DatasetName*.yaml:** This YAML file contains the specific configuration for the *DatasetName* dataset.

> ***MODIFICATIONS:*** *Configuration of the main parameters and assignment of the path where the dataset is located.*

**CREATE: config/experiment/*DatasetName*:** It contains experiment-specific configurations for the dataset and is used to run experiments with different configurations without having to modify the main code.

> ***MODIFICATIONS:*** *This folder should contain the files "`base.yaml`", "`small.yaml`", "`tiny.yaml`" and "`default.yaml`" (which can be copied from other datasets already created and the parameter values modified if necessary)*

**CREATE: /data/genx_utils/splits/*DatasetName***: Stores divisions of the dataset into subsets such as training, validation and testing in order to load only the samples corresponding to each phase of the training. All autogenerated `.pkl` files will be saved in this folder.

**data/genx_utils/labels.py:** This is where the structure that the image labels of any dataset that is to be used with this model must follow is found. It specifies the assignment of numerical identifiers to class names and manages the conversion between them.

> ***MODIFICATIONS:*** *There is a variable in which the coordinates of the expected images in the dataset must be indicated. This variable is called "`input_size`" and is found in the main of the code.*

**utils/evaluation/prophesee/evaluator.py:** Implements functions to evaluate the performance of the model in event detection tasks using *Prophesee* specific metrics (such as precision, recall or F1-score).

> ***MODIFICATIONS:*** *Create a new* `LABELMAP` *with the abbreviations of the classes in the new dataset and identify the number of classes in the variable "*`num_cls`*". It is also useful to add the name of the new dataset in "*`assert dataset`*" to evaluate possible errors.*

**utils/evaluation/prophesee/evaluation.py:** Contains functions for detailed evaluation of the results obtained with the model, including advanced metrics and statistics on the predictions.

> ***MODIFICATIONS:*** *A new camera type should be added with the name given to the new dataset (*`camera == 'NewDataset'`*) specifying the annotation classes. In addition, it is ideal to add the name of the new dataset to the "*`assert camera`*" error evaluation.*

**data/utils/spatial.py:** Contains functions for manipulating and transforming spatial coordinates in the data, which is useful in the representation and processing of detections. It is used in the data preprocessing phase and also in the post-processing of model predictions.

> ***MODIFICATIONS:*** *Add a new "*`DatasetType.DatasetName`*" with the dimensions of the images in the new dataset.*

**utils/evaluation/prophesee/visualize/vis_utils.py:** Contains functions to visualize events and predictions in images or videos, helping to interpret the model results.

> ***MODIFICATIONS:*** *Create a new LABELMAP with the abbreviations of the classes in the new dataset. In addition, a conditional must be added when evaluating the variable "*`dst_name.lower()`*" to evaluate with the dataset to be worked with and therefore choose which type of* `LABELMAP` *should be used in each case (Example: elif* `dst_name.lower() == 'DatasetName': return LABELMAP_DatasetName`*).*

On the other hand, this is where the color of the bounding boxes is configured for display.

**config/modifier.py:** Allows dynamic modification of model and dataset configurations, adjusting parameters for different experiments or datasets. It is executed before training to adapt the configuration to the user's needs.

> ***MODIFICATIONS:*** *Define the number of classes according to the dataset. For example:* `num_classes = 3 if dst_name == 'gen4'`*.*

**modules/utils/ssod.py:** Implements functions related to semi-supervised training (SSOD), including pseudo-label generation and detection refinement. It is used in the semi-supervised training phase to improve model performance by combining labeled and pseudo-labeled data.

> ***MODIFICATIONS:*** *Define the dimensions of the images to be used in the new dataset by indicating the name of the dataset for the height and width within the macros* `DATASET2HEIGHT` *and* `DATASET2WIDTH`*.*

**config/model/pseudo_labeler.yaml:** This file configures the confidence thresholds for object detection and for the classification of detected objects for the generation of pseudo-labels in semi-supervised training.

***MODIFICATIONS:***

- `obj_thresh:` *This suggests that the detection of certain objects requires a higher level of confidence to be accepted as a pseudo-label, while other objects can be accepted with a lower threshold. If the model's confidence in the detection of an object is lower than these values, that detection is discarded. Each row represents a class in the dataset. Example:* `obj_thresh: [0.6, 0.3]`.
- `cls_thresh:` *This parameter prevents the model from generating incorrect pseudo-labels with low classification probabilities. Example:* `cls_thresh: [0.6, 0.3]`.

It should be borne in mind that each vector must have as many columns as there are classes in the dataset to be used.

### 4.1.4 Parameter management

The LEOD model can be configured for your dataset based on a large number of parameters. There are some that define the general configuration of the dataset and others that define the specific configuration of a particular experiment or dataset. The former are found in the directory "*config/dataset/*" and the latter are found in the directory "*config/experiment/*", respectively.

#### 4.1.4.1 General configuration

**Model configuration**

**embed_dim:** Refers to the representation size of the backbone[5] model. It defines how many features it should learn at each step.

***MODIFICATIONS:***

- *If the value is high (e.g. 64, 128), the model can capture more details and represent richer information.*
- *If the value is low (e.g. 16, 32), the amount of memory and computation is reduced, but so is the model's ability to learn complex patterns.*

**dim_head:** Represents the attention in the network. It defines how the model distributes the information among different parts of the image or sequence of events.

[5]The base network or main structure of a deep learning model

***MODIFICATIONS:***

- *If `dim_head` is high (e.g. 32, 48), each "attention head" can focus on more information at a time.*
- *If `dim_head` is low (e.g. 8, 16), processing is lighter, but with less capacity to understand complex relationships.*

**fpn.depth:** This refers to the Depth of the Feature Pyramid, used by LEOD to improve event detection, allowing events to be detected on different scales. It's like adjusting the number of zoom levels on a camera (a deeper pyramid allows you to see fine details, while a shallower one only captures the most obvious).

***MODIFICATIONS:***

- *If `fpn.depth` is high (e.g. 1.0), the model will be able to detect small and distant events with greater precision.*
- *If `fpn.depth` is low (e.g. 0.33), the model will be faster, but may ignore details in small objects.*

**Training Configuration**

**Precision:** Represents the type of numerical precision used during training (float8, float16, float32, etc.). It has a great impact on the efficiency and stability of the model.

***MODIFICATIONS:***

- *If `precision: 16 (float16)`, lower precision is used for calculations on the GPU, reducing memory and speeding up training.*
- *If `precision: 32 (float32)`, higher precision is used, which can improve stability in more complex models.*

**max_epochs**[6] **and max_steps**[7] Represent the maximum time the model is trained (maximum number of epochs and maximum number of steps). These are the main and most critical factors in machine learning algorithms.

***MODIFICATIONS:***

- *If we train too many epochs, the model could overlearn (overfitting), losing its capacity for generalization.*

[6] An **epoch** represents a complete pass of the model over the entire training data set (if there are 100 images, it trains with all 100 images).

[7] A **step** represents an update of the model weights after processing a minibatch of data. When the model trains, it does not process all the data at once, but divides it into small batches (batch size). Each time the model processes a batch and adjusts its weights, it counts as one step. **LEOD is configured according to the maximum number of steps, not epochs.**

- *If we train too few, the model does not reach its best performance.*

**learning_rate:** Controls how much the model weights are adjusted in each iteration.

**MODIFICATIONS:**

- *If `learning_rate` is high (e.g. 0.01), the model learns quickly, but could become unstable.*
- *If `learning_rate` is low (e.g. 0.0001), the model learns more stably, but will need more iterations.*

#### Learning Rate Configuration (`lr_scheduler`)

**pct_start:** This parameter indicates what percentage of the training will be dedicated to the warmup phase, in which the learning rate starts low and then progressively increases.

***MODIFICATIONS:***

- *If `pct_start` is high (e.g. 0.1 a $\rightarrow$ b 10% of training), the model will start slowly for a longer period of time before reaching its maximum rate.*
- *If `pct_start` is low (e.g. 0.005 a $\rightarrow$ b 0.5% of training), the learning rate will increase rapidly.*

**div_factor:** This parameter reduces the initial learning rate compared to the maximum learning rate (*learning_rate*). A low learning rate at the beginning helps stabilize learning before making more aggressive adjustments.

$$learning\_rate_{init} = \frac{learning\_rate}{div\_factor}$$

**final_div_factor:** This parameter defines how much the learning rate will be reduced at the end of training. When the model is almost trained, reducing the learning rate allows for fine-tuning without changing the weights too much, which helps improve final accuracy and avoid overfitting.

$$learning\_rate_{end} = \frac{learning\_rate}{final\_div\_factor}$$

#### Hardware and Computational Resources Configuration

These are the parameters that dictate the hardware how much power it should use to execute the algorithm. In the case of having little data in the dataset, it is possible that few resources

should be chosen (small `batch_size` and `num_workers`), since there may not be enough data to execute at the same time (in the case of executing in parallel).

**batch_size:** Indicates how many examples are processed simultaneously in each iteration of the training or evaluation.

***MODIFICATIONS:***

- *If `batch_size` is large (e.g. 64), training is more stable and faster, but requires more GPU memory.*
- *If `batch_size` is small (e.g. 2), it is possible to train on computers with less memory, but model updates will be noisier.*

**num_workers:** Defines how many parallel processes are used to load this data.

***MODIFICATIONS:***

- *If `num_workers` is high (e.g. 8), data loading will be faster, but CPU usage may increase.*
- *If `num_workers` is low (e.g. 2), data loading will be slower, but resource consumption will be reduced.*

#### 4.1.4.2 Specific dataset configuration

**Configuring the Location and Loading of the Dataset**

**path:** This parameter defines the path where the dataset to be used in the experiment is located. If the path is changed, it is necessary to ensure that the data is organized correctly so that the model can access it.

**only_load_labels:** Indicates whether only the labels should be loaded without loading the complete events.

***MODIFICATIONS:***

- *If `True`, memory and processing are saved by avoiding loading the raw events.*
- *If `False`, both the labels and the original events are loaded.*

**only_load_end_labels:** Defines whether only the final labels of the events should be loaded, instead of all the instances annotated in the time sequence. This can be useful in tasks where only the detection of events at a specific instant is of interest.

**reverse_event_order:** Allows the order of the events in the time series to be changed.

> ***MODIFICATIONS:*** *If `True`, the most recent events will appear first, which can be useful for certain experiments that analyze trends in reverse. Data Subsampling Configuration*

### Data Subsampling Configuration and Model Choice (SSOD or WSOD)

**ratio:** Control the fraction of labels available within each training sequence. This parameter is associated with the study of the **WSOD** model, as it ensures that the model only sees the indicated percentage of labels for each sequence of the dataset without eliminating them (configures the corresponding split file with the established configuration).

> ***MODIFICATIONS:***
>
> - *A value of `-1` means that all the available annotations are used per sequence.*
> - *A value `0.1` indicates that only 10% of the original annotations will be kept in each sequence. Thus, if a sequence had 100 annotated frames, only 10 of them will be kept as "labeled frames" for training.*
> - *For **Semi-Supervised Object Detection (SSOD)**, the **ratio** is usually kept at **`-1`**, as all annotations are used in the selected sequences.*

**train_ratio:** Controls how many sequences (or samples) from the dataset are used for training. Unlike `ratio`, which affects how many annotations are taken for each sequence, `train_ratio` affects the number of sequences that are included in the training phase. It is associated with the **SSOD** model, as it allows the sequences of a dataset to be divided into labeled and unlabeled sequences, only by taking their labels or not when creating the `.pkl` file (*splits*).

> ***MODIFICATIONS:***
>
> - *A value of `-1` indicates that all of the available sequences are used for training.*
> - *A value of 0.1 indicates that only 10% of the existing sequences in the dataset will be used in the training. Thus, if the dataset consists of 50 sequences, only 5 of them will be used for training (without deleting the data from the remaining sequences).*
> - *For **Weakly-Supervised Object Detection (WSOD)**, the `train_ratio` is usually kept at `-1`, as all existing sequences are used.*

**val_ratio** and **test_ratio:** These parameters control how much data is used in each of the specified phases depending on the depth we seek to give to the verification in each of them.

- **val_ratio:** Proportion of data used in model validation.
- **test_ratio:** Proportion of data reserved for the final test of the model.

***MODIFICATIONS:*** *Modifications work the same as in the case of the "`ratio`" and "`train_ratio`" parameters. If a value of `-1` is set, the entire available data set will be used. Reducing these values speeds up the process, but may affect the quality of the results.*

### Data Representation Configuration

**ev_repr_name:** Defines the format in which the events will be represented before being entered into the model. For LEOD, the name "*stacked_histogram_dt=50_nbins=10*" is used by default, although it can be modified depending on the use. This indicates that the events will be grouped into histograms with intervals of 50 time units and 10 bins (channels).

**sequence_length:** Defines the number of time steps considered in each sample.

***MODIFICATIONS:*** *High values allow more context to be captured, but increase computational complexity.*

**resolution_hw:** Specifies the resolution of the image representing the events. In the case of the dataset created for this project (`FLW_dataset`), dimensions of [240, 304] are used.

***MODIFICATIONS:*** *Modifying it will affect the amount of visual information the model receives.*

**downsample_by_factor_2:** Reduces the input resolution by a factor of 2.

***MODIFICATIONS:*** *If `True`, the size of the image is reduced, which speeds up processing, but can lose details in the events.*

### Training and Sampling Mode Configuration

During training, data can be selected in different ways:

**train.sampling:** Defines how samples are chosen for training.

***OPTIONS:***

- **'random':** Selects events randomly. Concurrent execution.
- **'stream':** Uses continuous sequences of events. Execution is sequential
- **'mixed':** Mixes both strategies. Concurrent execution.

**train.mixed.w_stream** and **train.mixed.w_random:** Adjust the weight of each strategy in

mixed sampling.

***MODIFICATIONS:***

- *If `w_stream` is higher, the selection of continuous sequences will be favored.*
- *If `w_random` is higher, more weight will be given to the random selection of events.*

**Data Augmentation Configuration**

Data augmentation is a key technique for improving model generalization. In LEOD, there are several ways to modify the input data before training:

### Random Transformations

**prob_hflip:** Probability of mirroring the image horizontally.

EXAMPLE: If it is `0.5`, it indicates that half of the samples will be mirrored.

**prob_tflip:**Probability of reversing the events in time.

EXAMPLE: If it is `0.5`, there is a 50% probability that the events will be ordered in reverse.

**rotate.min_angle_deg** and **max_angle_deg:** Define the rotation range in degrees.

EXAMPLE:
rotate:

prob: 0.5
min_angle_deg: 2
max_angle_deg: 6

This means that there is a 50% probability that the image will be rotated between 2 and 6 degrees.

### Zoom Transformations

Zoom transformations are part of data augmentation techniques, the aim of which is to make the model more robust in the face of variations in the scale of the events detected. In addition, these data augmentations help the model to be more robust in the face of changes in the scale and orientation of events.

In other words, by zooming in on the training images, the model learns to recognize events in different sizes, which makes it more accurate in real-world situations. The parameters that can

be controlled in LEOD are as follows:

**zoom.prob:** Ensures that the model sees enough zoomed images to learn how to detect events at different scales, but without altering all the images.

***MODIFICATIONS:***

- *If `prob: 1.0` (100%), all training images are zoomed in.*
- *If `prob: 0.5` (50%), only half of the images are zoomed in.*
- *If `prob: 0` (0%), no images are zoomed in.*

**zoom.zoom_in.factor:** Defines the magnification of the image. Helps the model to detect events in large and nearby objects.

***MODIFICATIONS:***

- *If `min: 1` and `max: 1.5`, it means that the image can increase its size up to 50% larger.*
- *If `min: 1` and `max: 2`, it means that the image can double in size.*
- *If `prob: 0` (0%), no images are zoomed in.*

**zoom.zoom_out.factor:** Controls the reduction of the image.

***MODIFICATIONS:***

- *If `max: 1.2`, it means that the image can be reduced to 80% of its original size.*
- *If `max: 1.5`, it means that the image can be reduced by half.*

### 4.1.5 Workflow

The use of LEOD for event detection in time series follows a structured process that allows the performance of the model to be progressively improved as new data is incorporated. The process followed to make the most of LEOD's capabilities in event detection in time series is detailed below.

#### Initial Model Training with Annotated Data

The first step in implementing LEOD consists of training the model with a manually annotated data set (using `train.py model=rnndet`). This phase is fundamental, as it provides the basis on which the model will learn to identify events within the time series.

To do this, it is necessary to have a dataset in which each relevant event has been previously marked by a human or extracted from reliable sources. This labeled data is divided into three

subsets: **training, validation and test**. During training, the model adjusts its internal parameters using optimization algorithms that seek to minimize the difference between its predictions and the real labels.

Throughout the process, the model not only learns to recognize specific events, but also adjusts its weights to be able to generalize to new situations. Once the initial training is complete, the model's performance is evaluated using the validation and test data, which allows for the identification of possible errors and the adjustment of hyperparameters if necessary. [8] At this point, a base model is obtained that is capable of detecting events with reasonable accuracy, although its capacity for generalization may still be limited due to the amount of data used in training.

#### Incorporation of New Data and Generation of Pseudo-labels

Once the model has been trained with the initial data, the next step is to **introduce new data into the system**. This data can come from new captures, additional sensors or simply from information that was not previously available. However, the main challenge at this point is that this new data is not manually annotated, so the model cannot use it directly for learning.

To address this limitation, LEOD incorporates the **pseudo-labeling mechanism** (`predict.py`), which consists of using the previously trained model to predict events in this new data and generate labels automatically. That is, instead of a human manually annotating the events within the new scenes, the model performs this task automatically based on what it learned in the previous phase.

This pseudo-labeling process is crucial because it allows the labeled data set to be significantly expanded without incurring the high cost of manual or semi-manual annotation. However, the generated pseudo-labels may contain a certain margin of error, so it is necessary to evaluate their quality before using them in a new training phase.

#### Pseudo-Label Verification and Quality Control

Before using the generated pseudo-labels in a new training cycle, it is essential to verify their quality and ensure that the information the model is learning is accurate and useful. If the pseudo-labels contain too many errors, these will be propagated in the following iterations of the training, negatively affecting the performance of the model instead of improving it.

Pseudo-label validation can be carried out in several ways. A common strategy is **visual comparison** (`vis_pred.py`), in which graphical representations of the events detected by the model are generated and superimposed on the original data. This makes it possible to identify whether the model is detecting events in the correct positions or whether it is generating false positives and negatives.

Another technique used to evaluate the quality of the pseudo-labels is the **analysis of thresholds**

[8] There are different configurations that can be modified. Ideally, you should check which LEOD model best suits your needs (`base, small or tiny`). Then choose the number of steps (related to epochs) that you want to occur during training, depending on the level of refinement you are looking for. And finally, the algorithm's hyperparameters (*bach_size, num_workers, radio, etc.*).

**of confidence** (`val_dst.py`). In LEOD, each prediction of the model is accompanied by a level of confidence (`pseudo_label.obj_thresh`[9] and `pseudo_label.cls_thresh`[10]) which indicates how certain the system is that an event actually occurred at a certain point in the time series. If pseudo-labels are generated with too low a confidence threshold, they are likely to include errors. On the other hand, if a very high threshold is set, real events could be lost by discarding uncertain predictions. Finding a balance in this threshold is key to ensuring the quality of the pseudo-labels.

In some cases, it may also be useful to perform a partial manual review of the pseudo-labels, especially in applications where accuracy is critical. Evaluating a sample of the generated data allows the pseudo-labeling parameters to be adjusted and possible model biases to be corrected before moving on to the next phase of training.

#### Retraining the Model with Pseudo-Labels and Original Data

After validating the quality of the pseudo-labels, the next step is to **retrain the model** (`train.py model=rnndet-soft`) using both the original labels and the generated pseudo-labels. This technique, known as **self-training**, allows the model to refine its detection capability based on new data, improving its accuracy and generalization without the need for a manual labeling process.

During this phase, the model is exposed to a greater diversity of data and learns to identify more complex patterns. As the data set grows with the addition of pseudo-labels, the model can improve in detecting events that it previously did not recognize accurately.

However, it is important to maintain a balance between the original labels and the pseudo-labels during training. If the model is trained exclusively with pseudo-labels, it runs the risk of reinforcing its own errors instead of correcting them. For this reason, in LEOD it is recommended to combine both types of labels and, if possible, to introduce new samples annotated manually in each iteration of the process.

#### Final Evaluation and Implementation of the Improved Model

Once the pseudo-label retraining has been completed, a final evaluation (`val.py`) is performed to check if the model has improved in event detection. Its predictions are compared with the real labels in a test set and metrics such as accuracy (`AP`) calculated to determine its performance.

If the model achieves a satisfactory level of accuracy, it can be implemented in production or used in real-world event detection scenarios. Otherwise, the cycle can be repeated by incorporating more data and refining the pseudo-labeling process.

This iterative approach allows LEOD to learn and improve continuously, reducing the need for manual labeling and adapting to new conditions without losing accuracy.

[9] `pseudo_label.obj_thresh`: Defines the threshold of confidence for accepting a detection as valid.

[10] `pseudo_label.cls_thresh`: Defines the confidence threshold for the classification of the detected object. It indicates how certain the model must be about the class of the object before generating a pseudo-label.

# 5 RESULTS

## 5.1 Bidimensional Dataset

### 5.1.1 Data Pre-processing

#### 5.1.1.1 Data Structure

To create a dataset compatible with the algorithms used, the main structure required for each sequence of data is as follows:

- ***sequence_name***
  - **event_representations_v2**
    - stacked_histogram_dt=50_nbins=10
      - event_representations.h5
      - objframe_idx_2_repr_idx.npy
      - timestamps_us.npy
  - **labels_v2**
    - labels.npz
    - timestamps_us.npy

Each of these files has the following functions:

**event_representations_v2**

- **event_representations.h5**: This file stores event histograms in HDF5 format and represents the events detected in different time windows as three-dimensional images. It has a structure of the type:

  $$(\text{Number of histograms}, \text{Channels}, \text{Height}, \text{Width})$$

  where:

  - ***Number of histograms***: Each event histogram represents a specific time interval and serves as input to the model for processing event information instead of RGB images. In general terms, each histogram shows the frequency with which certain values occur within a given range.
  - ***Channels or bins***: This is a discretization of the histograms. The number of bins indicates how many subintervals the histogram will be divided into, in order to calculate how many different events occur in each subinterval and compare them. This reduces the amount of data to be processed and helps to compress patterns more quickly.
  - ***Height***: This is the height of the event images.

  - ***Width***: This is the width of the event images.

- **timestamps_us.npy**: This file stores the time intervals covered by each histogram (event histogram timestamps).

- **objframe_idx_2_repr_idx.npy**: This file matches the labeled images with the event histograms. Therefore, it must have the same length as `labels_v2/timestamps_us.npy` and its values cannot exceed the length of the file `event_representations_v2/.../timestamps_us.npy`.

**labels_v2**

- **labels.npz**: This is a Python structure made up of two labels: `"labels"` and `"objframe_idx_2_label_idx"`.

  - ***labels***: A structure of the type:

    $$[\text{timestamp}, x, y, \text{weight}, \text{height}, \text{class}, \text{confidence}, \text{track\_id}]$$

    where both the bounding boxes of each label and the class of the label with its confidence interval are identified.

  - ***objframe_idx_2_label_idx***: A vector that matches the index of each label with the corresponding timestamp.

- **timestamps_us.npy**: This file contains the timestamps of the labeled images.

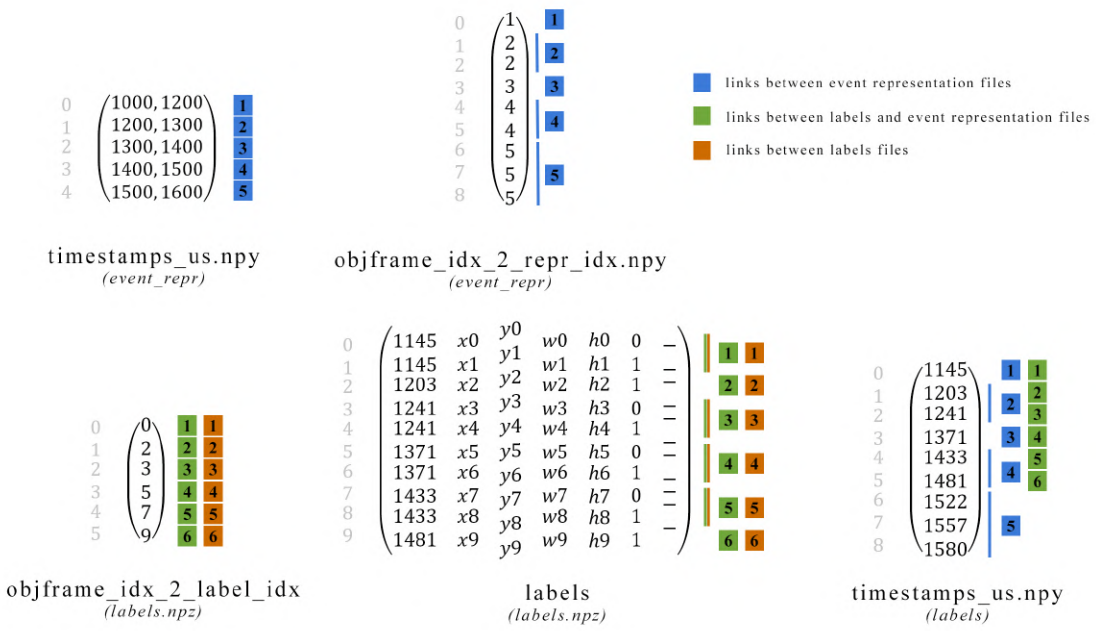



Figure 5.1: Representation of dataset files with the correct structure and their linking together

**Important Note**

It is essential that the data in each of the above files is in **ascending order**. Failure to maintain the correct order may cause mismatches in label representation, leading to **invalid or incorrect training**.

### 5.1.2 Pre-processing Flow

Before being able to pre-process data and convert it into structures that can be understood by the algorithms used, it is necessary to have data available. To achieve this, we start from the raw data obtained directly from the captures of both the event cameras and the RGB cameras.

Once we have the raw data, it is time to pre-process it to structure it in such a way that the algorithms can understand and work with it.

Based on the data structure, the flow is divided into two stages: labeling and representation of the events.

#### 5.1.2.1 Data Extraction

ROS Noetic is used to extract the data from each scene in our dataset. Using *ROSBags*, we can store the most important data that characterizes each scene. The following topics are recorded in these *ROSBags*:

- **RGB image:** (timestamp, pixels)
- **Left camera events:** (timestamp, events *(polarity and pixel)*)
- **Right camera events:** (timestamp, events *(polarity and pixel)*)
- **Position of the three cameras:** (Vicon)
- **Position of the element:** (Vicon)[11]

For newer scenes, additional tracked positions include human (head, trunk, feet, hands) and wagon (if it is used). This additional data increases the versatility of the dataset and enables new possibilities for future research.

Despite having a large amount of data, not all of it is necessary for this study. In this case, only the data from the RGB camera and the events from the left event camera have been used. The choice between the right and left cameras has no specific criteria.

[11]To simplify processing, the scenes used for this project contain only a single object and, in some cases, a single human.

Figure 5.2: Representation of the three-camera setup

One of the main challenges in synchronizing images generated from event cameras and RGB cameras is their inherent difference in synchronization. Event cameras operate asynchronously, while RGB cameras are synchronous. [12]

As a result, the timestamps of the RGB images differ entirely from the timestamps of the event data. Furthermore, if both event cameras (left and right) are used, the process becomes more complex, as each operates independently, leading to slight differences in their timestamps (although typically with minimal discrepancy).

When working with event cameras, another method that can be used to gain a better understanding of the information provided by these cameras and to provide more information for the segmentation of elements is the **reconstruction of events**.

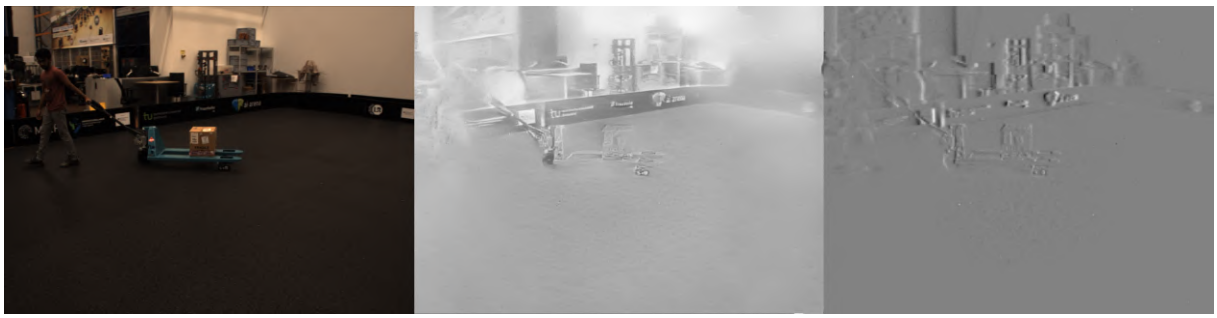


Figure 5.3: Different types of events reconstruction

The reconstruction of events is based on the information of each pixel that emits asynchronous impulses (events) when it detects an increase or decrease in illumination. To visualize the information as a conventional image, the complete cloud of events is taken and, at predefined time intervals (timestamps), each pixel is assigned the last recorded polarity value. In this way, an approximate intensity map is constructed that reflects the luminous variations that occurred in that interval. This is the simplest way of reconstructing events. However, there are more sophisticated methods [18][19] that, through more advanced learning models, allow images to be reconstructed with greater quality of detail. [13]

[12] Event cameras only provide data every time a pixel experiences a change in brightness (event). However, RGB cameras capture full images at fixed preset intervals.

[13] This could be interesting in order to be able to use the results obtained from events with traditional prediction and segmentation models in exchange for losing speed and lightness in processing.

#### 5.1.2.2 Labeling

When starting the project, the previous method of labeling scenes was based on the segmentation of elements using a `YOLO` detection model hosted on Roboflow that used the images resulting from the reconstruction of the events. As mentioned above, the images obtained from the reconstruction of the events lack much of the information that current image segmentation models are used to. Therefore, the labeling efficiency was very low.

After evaluating the labels obtained, it was concluded that this method had an efficiency of around 35%[14]. This is too great and unacceptable an error, so it would be necessary to manually eliminate those labels that do not meet expectations.

As a substitute, several modern segmentation models that offered greater efficiency were analyzed.

- **Segmentation methods evaluated:**

Since the main problem is the lack of data in event images when using segmentation algorithms, the idea was to use the data from the RGB images to create the labels and then transform them to the dimensions and times of the events. With this method, accuracy is initially improved, although when it comes to making the labels compatible with the events, some noise will be introduced. This noise will arise due to the desynchronization of the instants in which the data is obtained (it is extremely likely that the timestamps of the RGB images and the events are not the same) and to the difference in terms of the lens used (explained later in Section 5.1.2.2). In the quest to create labels with the least possible noise and error that could subsequently alter the result, several segmentation algorithms were analyzed. The more precise the algorithm, the less noise summation will be applied in the final label.

**Manual**

The process consists of assigning labels to each image or marking specific objects within them, which requires a detailed review by the annotators. Depending on the type of task —whether classification, object detection or segmentation— the level of detail and the time required can increase significantly. In large-scale projects, where thousands or even millions of annotated images are needed, manual labeling becomes unsustainable due to the human effort involved.

In addition to the time it consumes, this method is subject to human error and a lack of consistency between annotators, which can affect the quality of the dataset and, therefore, the performance of the trained model. For this reason, there are tools and techniques that seek to optimize this process, such as the use of artificial intelligence models for assisted labeling or semi-supervised learning strategies (LEOD).

Even so, manual labeling also depends heavily on the quality of the raw data obtained. In the case of the dataset used in this project, the only data used are RGB images and events. However, in the case of knowing the positions of the elements through markers at the time of recording scenes, this process can be simplified somewhat, since these positions could be extrapolated to the images obtained.[15] Therefore, semi-manual labels could be generated more directly and easily.

[14] As an average of the calculation of correct labels versus the total number of labels in several scenes.

[15] In the newest videos that have been recorded for the dataset, this data is already known, so this method could be evaluated for future labeling.

**Segment Anything Model 2 (*SAM2*)**

SAM2 [20] is an image segmentation model created by Facebook that has a universal segmentation architecture, designed to identify and segment objects in images and videos without the need for additional training for each specific data set. Unlike other traditional segmentation models, which require exhaustive training with specific data sets, SAM2 has been designed to generalize efficiently to new images and objects never seen before.

This is a model created mainly for RGB images, although, thanks to its highly optimized transformer that allows for rapid identification of contours and structures in images and videos, it can be effective in grayscale images or other types of images (although with less precision).

#### Segmentation process

This model is based on the idea of "*promptable segmentation*", where a user can provide specific indications, such as reference points or bounding boxes, to guide the segmentation of objects. This means that by only indicating the object to be segmented in the first frame of a video using the coordinates of a possible bounding box that surrounds it, the algorithm can create a mask that segments that element for all subsequent frames of the scene.

Once the mask/masks has/have been obtained, the model outputs an image for each frame (if it is a video) where the generated mask appears on top of the original image and an `.npy` file where the mesh that generates the mask in image size is found (for each image. The `.npy` file is an array of zeros and ones the size of the image where the ones represent the pixels where the mask is located. If it is necessary to know the coordinates of a possible bounding box surrounding the mask, it is necessary to obtain the maximum and minimum coordinates in the vertical and horizontal where a one appears.

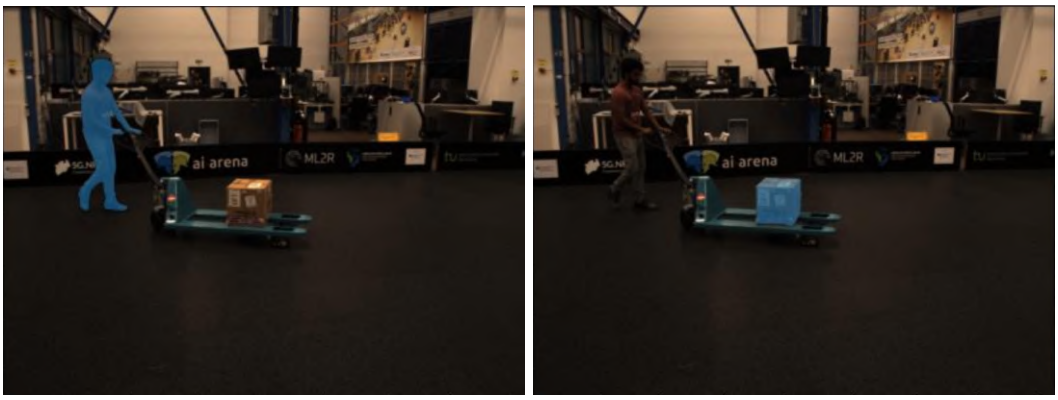
#### Experiments

When obtaining segmentations, the model has been tested using two types of images: RGB and gray-scale event reconstruction.

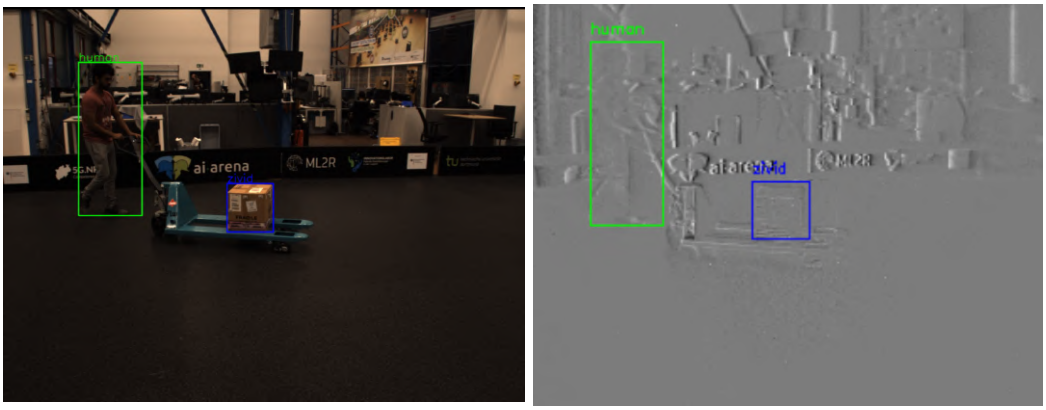
### A. Segmentation from RGB image

Object segmentation from RGB images has proven to be very effective with the model used. Thanks to this approach, fairly accurate segmentations and bounding boxes that clearly identify the objects in the image can be obtained. However, during its implementation, a technical problem was detected related to the conversion of coordinates between different image formats (RGB to event reconstruction image) as specified in Section 5.1.2.2.

The main challenge arises because RGB images have a resolution of *2048x1536* pixels, while event images, which are also used in this process, are much smaller (*304x240 pixels*). When converting coordinates from one format to another, precision is lost, as the points are not moved to exactly the same location in the new image.



(a) Segmentation mask using SAM2



(b) Resulting bounding box and transformation in the reconstruction of events

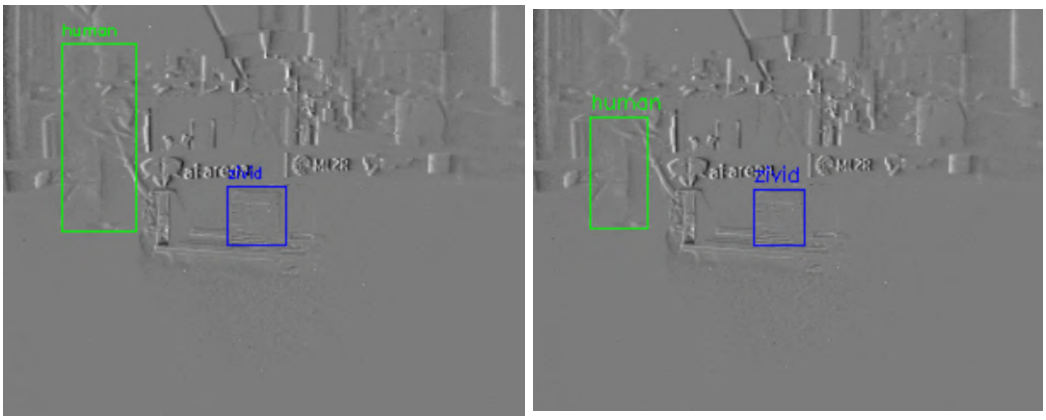
Figure 5.5: Comparative images mask, RGB bounding box and RGB to well-defined events.

As can be seen in Figure 5.5, the resulting labels after the transformation between images are quite good and meet all expectations. Despite this, all labels should always be inspected, even if only quickly, as erroneous annotations can always appear.

### B. Segmentation based on reconstruction of events in grayscale

In this case, the results obtained have been quite good and better than expected. However, inconsistencies in the quality of the generated labels have been identified. While in some frames the detection is accurate and the labels correctly cover the desired elements, in others the segmentation is not as effective, leaving some parts of the object without being correctly labeled.

This variability in the accuracy of the labels represents a significant challenge. Given that some labels are very accurate and others not so much, it is not possible to apply an automatic correction factor, as this could improve the incorrect labels, but at the same time deteriorate those that were already accurate. Due to this limitation, this method has not been selected as the main approach for segmentation.




(a) (b)

Figure 5.6: RGB to events bounding boxes (a) vs directly event segmentation (b) for the same timestamp

[16]

Unlike other methods, this one does not require a transformation of coordinates, which simplifies the processing. However, it introduces errors in the labels, albeit in a small number. To correct these errors, there are two main options:

- **Eliminating the incorrect labels:** This would reduce the amount of data available for training or analysis.
- **Manually correcting the faulty labels:** This would involve greater time consumption and a loss of efficiency in the process.

In cases where Semi-Supervised or Weakly-Supervised (specially) training is sought, this approach may still be a viable option. However, manual processing will still be necessary to eliminate labels that are not of interest.

## Agents-based Object Detection (*AOD*)

Agents-based Object Detection [21], developed by Landing AI, is an advanced computer vision technique that allows objects to be identified based on their unique attributes (color, shape, texture) without the need for prior training. Instead of relying on large labeled data sets, AOD uses an agent-based approach to reason about the specific characteristics of objects, achieving more flexible and accurate recognition.

Its technology is distinguished by the ability to interpret natural language descriptions, allowing users to specify what they are looking for without the need to program complex rules.

### Segmentation process

The segmentation process using this model is very simple. All that is needed is to enter a prompt that clearly identifies the element to be labeled (for example, "*box on top of wagon*"

[16] In these cases, a frame has been chosen on purpose. Not all the frames where the bounding boxes are generated directly are erroneous.

or "*human*"). Once this has been entered and executed, a `.json` file is obtained with all the necessary segmentation information.

The specific piece of information used in this case is the bounding box coordinates. These coordinates are represented in a simple vector with only 4 values that represent the maximum and minimum vertical and horizontal coordinates of the bounding box.

#### Experiments

As in the case of the use of SAM2 (previus section) the model has been tested using the same two types of images: RGB and the reconstruction of events in grayscale.

### A. Segmentation from RGB image

The Agentic Object Detection (AOD) algorithm has shown mixed results in image segmentation. On the one hand, it allows for acceptable detections, but they are not always completely accurate. To improve the quality of the generated labels, a correction factor (larger than in SAM2) is applied that adjusts their coordinates and aligns them better with the real objects. However, this adjustment is not infallible and can affect accuracy in some cases.

An important aspect of AOD is that it depends to a large extent on the prompt that is entered. This means that, if the description of the object is not clear or is too complex, the model may not correctly interpret the instruction and fail in the segmentation. For example, if the term "`human`" is entered, the system understands it without problem and generates the segmentation. But if the prompt is something more elaborate, such as "`human pulling a wagon`", the model may not understand it and simply not generate any segmentation. This makes the success of the method largely dependent on the way in which requests are formulated.


(a) Prompt = "box" (b) Prompt = "box in a wagon"

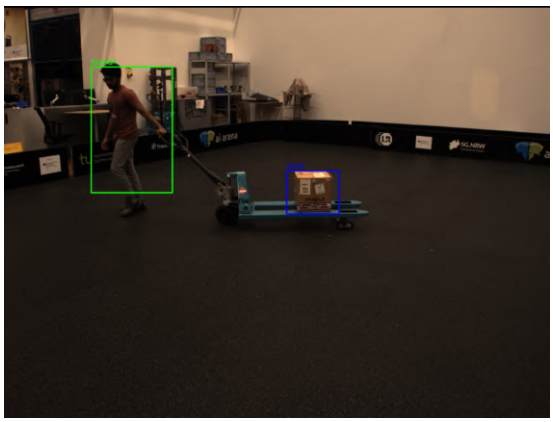
Figure 5.7: Need to write a corrent prompt.

Another problem with this approach is its high computational cost. AOD needs to be executed in each of the frames of a scene, which consumes a lot of resources and processing time. Compared to other methods, this aspect makes it less efficient when working with large volumes of images or videos.

On the other hand, one advantage of AOD is that it simplifies the process of obtaining bounding boxes. Instead of manually searching for the coordinates of the object within the image, it is enough to type its name and the model directly returns the parameters of the bounding box. This avoids having to calculate the maximum and minimum values of the segmented mask, as happens in SAM2, facilitating the process in some cases.

Despite its strengths, AOD also has problems with coordinate conversion. Like SAM2, it needs to transform the data from a *2048x1536* pixel RGB image to a *304x240* pixel event format. This change in resolution generates errors in the location of the labels, which can affect the final quality of the segmentation.


(a) Segmentation labels using AOD ("*human*" and "*box in a wagen*" respectively)



(b) Directly event segmentation

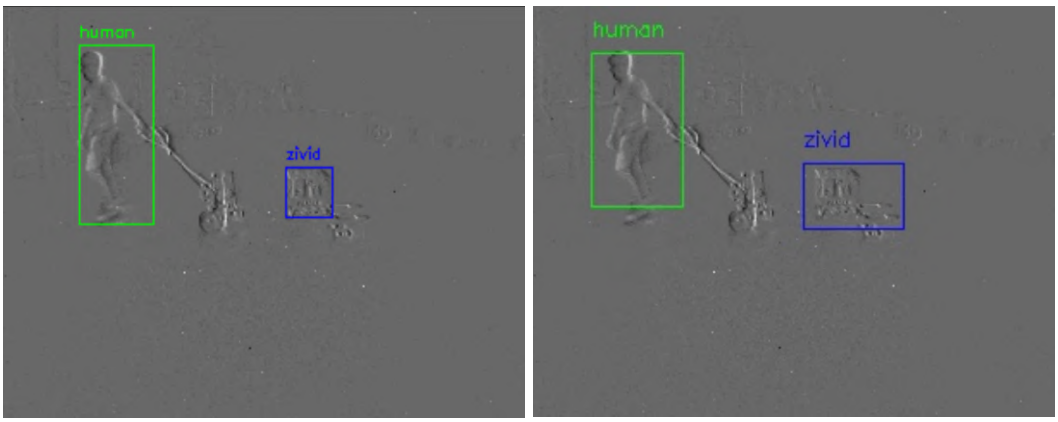
Figure 5.9: AOD outputs by propmt and resulting bounding boxes.

### B. Segmentation based on reconstruction of events in grayscale

In the case of using images generated from the reconstruction of grayscale events, a correction factor in the labeling is also required, since it does not always obtain accurate results. In some frames, the model may not be able to identify the indicated object, which causes some labels to not be generated or to be incorrect. This problem is similar to the one presented in SAM2, where

in certain cases the segmentation fails and correct labels are not obtained for some elements.

This means that, although AOD offers a flexible form of text-based segmentation, it does not always guarantee reliable detection in all frames. The need to apply a correction factor makes the process more complex because, although it helps to improve incorrect labels, it does not always completely solve the problem and can introduce other errors in the segmentation.




(a) (b)

Figure 5.10: RGB to event by fitting bounding boxes (a) vs directly event segmentation without correction (b) for the same timestamp.
[17]

**AOD + SAM2**

Another option is to mix both algorithms to create labels. This would greatly reduce the amount of time spent searching for and indicating the coordinates of each of the first frames of each scene.

Because in many cases the same elements are used for each scene, it would be enough to indicate the elements to be labeled and the respective bounding boxes of each of them would be obtained. This would be used only for the first frames of each scene and would be the data that would be fed to the SAM2 algorithm, which would act normally.

This could be treated as a possible expansion of the SAM2 algorithm, in which the algorithm could be used directly from a prompt powered by an LLM.

[17]This result was obtained after changing the prompt of the zivid box from "*box*" to "*square box*", otherwise the "zivid" label was not obtained.

- **Adjustment, filtering and transformation of labels:**

Since the segmented images are those from the RGB camera and have different timestamps than the events, a way has been sought to solve this incompatibility.

### Synchronization

The events will be the ones that dictate the timing, as they are the data that give meaning to the purpose of the project (working with events instead of conventional images). Furthermore, when it comes to structuring and giving meaning to the combination of tags matched with the events, these are the ones that provide the timing in the representation of events.

What is proposed in this section is a matching between the timestamps of the RGB images and the timestamps of the events. Therefore, the first thing is to associate, for each RGB image, the timestamp of the closest event.

### Transformation

Once we have the same timestamps in both the events and the labeled RGB images, the next step is to transform the label coordinates of the RGB images to the dimensions of the event images.

In the case of having the positions of the element, human and cameras, a transformation matrix could be applied to transform the label points from the RGB image to the event image.[18] However, this process is not so straightforward as the scenes we have been working with do not obtain the position of the human.

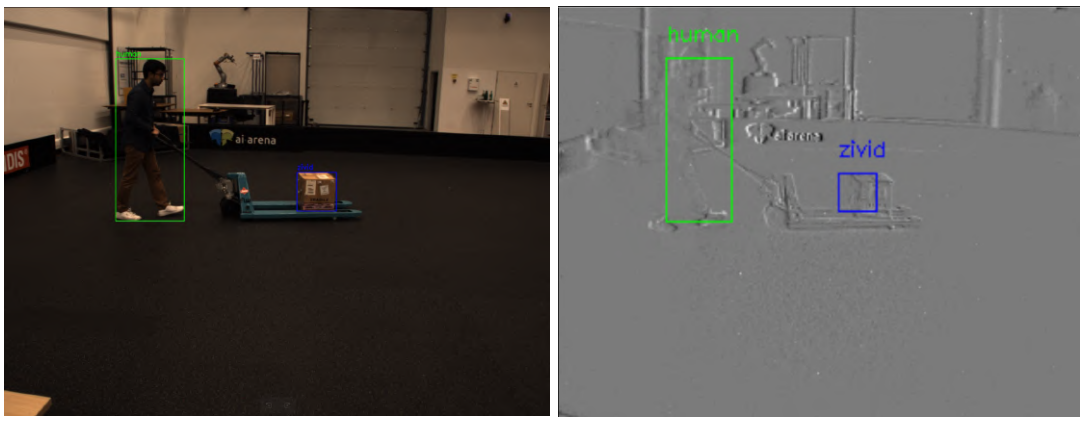



Figure 5.11: Mismatch in the transformation of bounding boxes from RGB image to event reconstruction

In Figure 5.11, the image on the left shows the labels generated in the RGB image, and the image on the right represents the labels generated by direct transformation in the image provided

[18]We would need to have the matrices: RGB element-camera, human-RGB camera, RGB camera-event camera, event camera-generated image (These transformations can be obtained from "`Eye of Hand Calibration`" – calibrationHandEye() to calibrate the Vicon and the camera; and using the "`Kalibr`" repository to calibrate the three cameras)

by the event camera. As can be seen, the event camera captures an image that is somewhat more cropped than the RGB camera and of lower quality. This can be seen in the dimensions of each image, where the RGB camera provides images of *2048x1536* pixels compared to *304x240* pixels from the event camera.

If we calculate their ratios, we obtain a ratio of 0.1484 (304/2048) on the vertical axis, compared to a ratio of 0.1563 (240/1536) on the horizontal axis. This means that a linear transformation cannot be applied to the image coordinates, as they do not have a direct relationship and therefore the elements drawn on them would be distorted.

In addition to this, another problem is that it is difficult to know exactly the cropped dimensions of the image, since both cameras are not synchronous, therefore, it is not possible to compare the same timestamp and align the images. Therefore, to study these cases more specifically, a script was created to obtain the closest timestamps between the RGB images and the events.

Once the two closest images in time had been obtained, their main differences were analyzed in search of greater precision when evaluating the cropping of the image. On the other hand, after aligning them, phenomena related to irregularities in the event images were observed.

- **Problems and irregularities:**

After the comparative analysis of the images, the following irregularities were observed:

Figure 5.12: Irregularity: central alignment

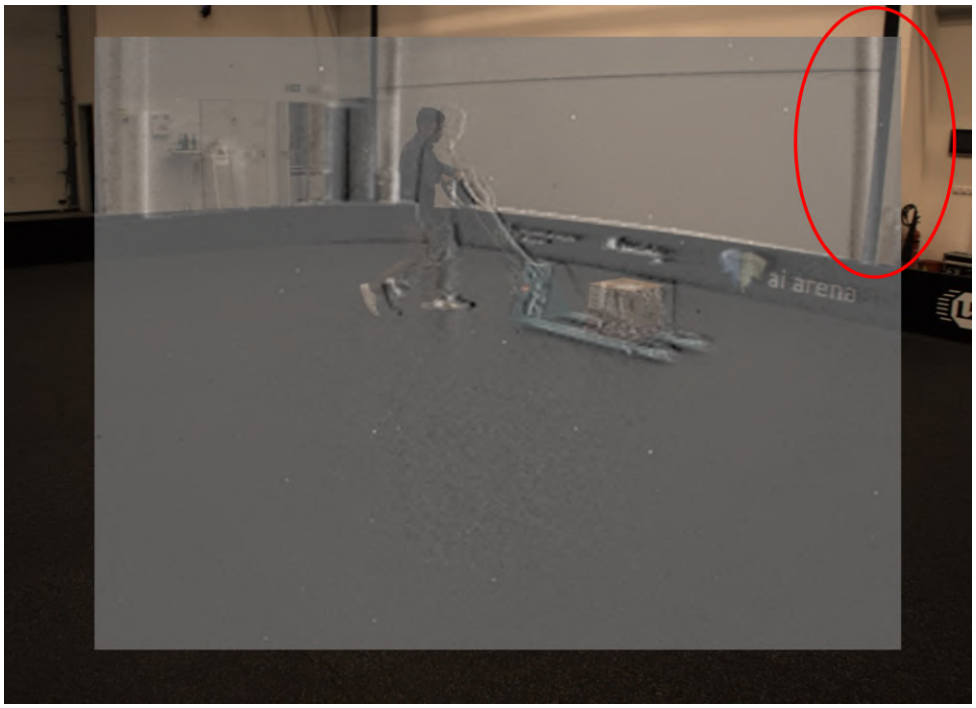



Figure 5.13: Irregularity: distortion


Figure 5.14: Irregularity: lateral alignment and distortion

As can be seen in the three previous figures, the objective is to align the image of the reconstruction of events with the RGB to detect irregularities. In the case of Figure 5.12, the central part is aligned, but the left edges and the "ai arena" sign are not aligned in turn. On the other hand, in Figure 5.13, it can be seen that the black edge of the wall on the right does not have the same morphology in both images, with the event reconstruction altering the morphology of the objects.

Finally, in Figure 5.14, which is a mixture of the previous two, we can see how the straight lines are not so straight in the event camera (they do not respect the morphology as in the previous case) and, in addition, we can see how, when trying to align the left part, a great misalignment is produced in the rest of the image.

These irregularities suggest that the transformation between images is somewhat more complex than expected. Therefore, it is necessary to obtain a transformation matrix that represents this transformation coherently.

To obtain the values of the proposed transformation matrix, RGB labels have been generated and labels have been created manually directly in the event reconstruction image. The availability of both types of labels allows us to know the initial and the sought points, so we have all the necessary information to obtain the transformation matrix that will resolve the matrix between the two.

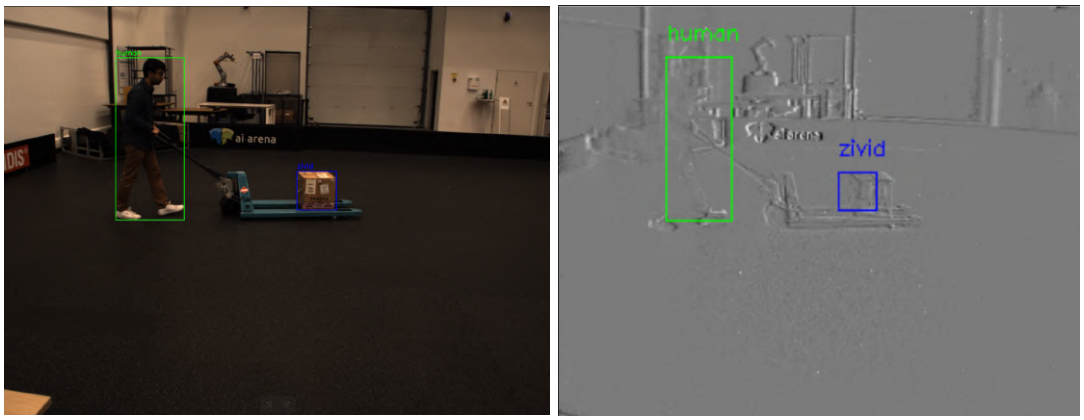



Figure 5.15: Labels transformation with only one transformationmatrix.

The Figure 5.15 really show how, despite aligning the image in terms of its vertical center, both the left and right sides vary. And if either of the two parts is aligned, the opposite part also looks misaligned (this is also observed in the detection flow, where the resulting joining boxes vary differently and in opposite directions).

However, after a process of trial and error, observing the previous figure, it has been concluded that the best way to transform the points between the two images is to divide the image vertically into two equal parts and obtain a transformation matrix for each half.

$$H_1 = \begin{bmatrix} 0.1591 & -4.51 \times 10^{-15} & -11.4773 \\ 1.47 \times 10^{-14} & 0.1757 & -12.3579 \\ 1.17 \times 10^{-16} & -7.02 \times 10^{-17} & 1 \end{bmatrix}$$

*Transformation matrix for the left side*

$$H_2 = \begin{bmatrix} 0.1647 & -1.23 \times 10^{-13} & -11.7647 \\ 1.64 \times 10^{-13} & 0.1619 & -3.0211 \\ 1.81 \times 10^{-15} & -1.69 \times 10^{-15} & 1 \end{bmatrix}$$

*Transformation matrix for the right side*

After this update, the resulting bounding boxes are more effective. Despite this, given that the objective of this project is not to maximize the precision of the labels or to generate the most perfect preprocessing possible, the matrices used produce perfect precision. To do this, it would be necessary to generate these matrices from a greater number of data and iterate the process for each one.


Figure 5.16: Process of transforming points between RGB image (original labels) and gray-scale event reconstruction image (desired labels).

#### 5.1.2.3 Event representation

Before starting to create the event representation, it must be taken into account that their asynchronous nature makes them difficult to use in conventional learning models, which operate with fixed matrix representations. To address this limitation, the events are grouped into temporal histograms, allowing the activity of events to be represented in an organized structure comparable to traditional images. These histograms capture the density of events in discrete time intervals and provide a robust representation.

The storage and processing of these histograms is a critical step in the event-based vision pipeline. A structured conversion of the event data is required, ensuring that each histogram adequately reflects the activity in a defined time window.

- **Histograms Creation:**

The representation of events using histograms is an essential technique for organizing the information captured by event cameras. Its objective is to transform discrete events into a three-dimensional structure that allows its use in automatic learning tasks. This process follows a well-defined flow that guarantees the spatial and temporal coherence of the data.

### Definition of Parameters and Storage Structures

Before processing the events, key parameters are established that will define the way in which the histograms will be constructed. The spatial resolution of the histogram is determined, which can be different to the original resolution of the sensor, allowing a reduction in dimensionality if necessary. The number of temporal bins within each histogram is also defined, which will segment the events into different time windows, and the time interval of each bin (which will be discussed in Section 5.1.2.3), which delimits the duration that each bin will cover. With these parameters, a storage structure is initialized in the form of a three-dimensional matrix H with dimensions (*bins, height, width*), where initially all the values are zero.

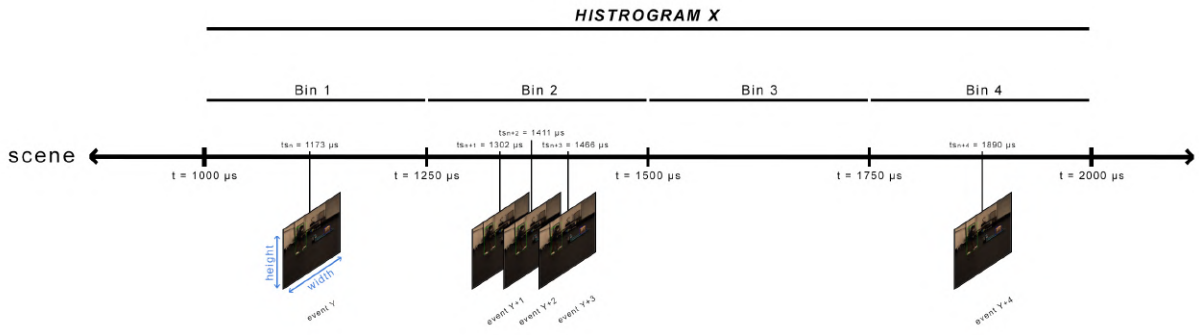



Figure 5.17: Recreation of a histogram representing events. `Histogram [1, 3, 0, 1]` with `bin_max = 4`.

19

### Event Processing and Normalization

The events are extracted sequentially from the ROSBag file, each with information about its spatial coordinate `(x,y)`, its timestamp `t` and its polarity `p` (raw event data). Before assigning them to a histogram, it is necessary to define a temporal reference point. To do this, the first valid event is taken and its start time is adjusted to the nearest multiple of the temporal bin interval:

$$start\ time\ fitting = start\ time - (start\ time \mod time\ interval)$$

This adjustment guarantees that the histograms start at aligned times and avoids mismatches between time bins.

### Assigning Events to Time Bins

Each event must be placed in the correct time bin within its histogram. To do this, its relative time is calculated with respect to the adjusted start:

[19] If this histogram is repeated more than once, the algorithm can interpret it as a possible pattern. This, together with the labels, allows the algorithm to infer a possible solution in a faster and more structured way.

$$t_{relativo} = t - t_{start}^{fitting}$$

Then, it is determined in which time bin the event falls:

$$bin\ index = \frac{relative\ time - current\ bin\ start\ time}{time\ interval}$$

If the calculated index is greater than or equal to the number of bins available in the current histogram, it means that the event belongs to a new histogram. In this case, the current histogram is stored and a new empty histogram is started, adjusting the start time to the next time interval.

#### Spatial Mapping and Event Registration in the Histogram

Once the corresponding temporal bin has been identified, the event must be located in the correct spatial position within the histogram. To do this, its position is rescaled according to the resolution of the histogram. In this way, the event is assigned to the corresponding cell of the three-dimensional matrix, increasing its value by one.

#### HDF5 File Storage and Creation

Once all the histograms have been generated, they are stored in an HDF5 file for subsequent processing in neural networks. This format allows for efficient data compression and fast access during model training. However, when comparing the storage size of our data in the generated "*event_representations.h5*" file with the data in the same file in the Gen1 dataset [22], we obtained the following results:

| | **Flw dataset** | **Gen1 dataset** |
|---|---|---|
| Número de datos | 475 | 1198 |
| **event_representations.h5** | **693.1 MB** | **85.1 MB** |
| timestamps_us.npy (repr.) | 7.7 kB | 18.4 kB |
| objframe_idx2_repr_idx.npy | 4.9 kB | 512 bytes |
| **labels.pnz** | **48.2 kB** | **4.4 kB** |
| timestamps_us.npy (labels) | 4.9 kB | 512 bytes |

Table 5.1: Data from only one element of each dataset with no compression

As can be seen, the difference in size between our dataset files and those of Gen1 is almost 10 times greater despite having less data (more than half the data compared to Gen1). This means that LEOD requires more resources to train, which could lead to the need for fewer workers to work simultaneously. In turn, due to the large amount of data injected, it can result in a bottleneck at the time of execution. As a result, the training time can be much longer than normal or it may not even be possible to complete the training correctly.

To solve this, data compression is proposed. The type of compression chosen was **BLOSC compression** [23], an algorithm specifically designed to handle large volumes of data in multi-dimensional matrices. BLOSC reduces the size of the file without significantly compromising read and write speed, which is essential in machine learning applications where computational efficiency is key. This resulted in the following:

| | **Flw dataset** | **Flw dataset (BLOSC)** |
|---|---|---|
| Número de datos | 475 | 475 |
| **event_representations.h5** | **693.1 MB** | **14.6 MB** |
| timestamps_us.npy (repr.) | 7.7 kB | 7.7 kB |
| objframe_idx2_repr_idx.npy | 4.9 kB | 4.9 kB |

Table 5.2: Data from only one element of each dataset with BLOSC compression

As can be seen, the size of the data has been reduced to approximately 2% of the total. This is quite a significant achievement, as it saves up to approximately 50 times disk space (an important factor when working with datasets and machine learning).

On the other hand, the `timestamps_us.npy` and `objframe_idx_2_repr_idx.npy` files are also stored. This data does not need to be compressed, as its disk size is not too relevant.

- **Selecting the correct histogram size:**

The choice between using histograms with more or fewer images per histogram depends on a balance between temporal resolution, computational load and alignment with annotations.

If **fewer images per histogram** are chosen, the total number of histograms increases, which implies a greater volume of data to be stored and processed. However, this configuration allows for better temporal alignment with the annotated images, since each histogram represents a smaller and more specific temporal window. This is especially useful in applications where temporal precision is critical, such as in the analysis of rapid movements or in the training of learning models that require precise correlation between events and annotations.

On the other hand, if **more images per histogram** are chosen, fewer histograms are generated, which reduces storage and computational load. By grouping more events in each temporal bin, a more stable representation is achieved, which can be beneficial for models that require greater robustness against data variability. However, this strategy has disadvantages such as the loss of temporal resolution, which could make it difficult to capture fast movements or short-duration events. Furthermore, if the histograms group too many events, they can become saturated and lose detailed information, especially in scenes with high activity. Another potential problem is the mismatch between the histogram and the image labels, which can affect accuracy in classification or detection tasks.

Ultimately, the choice depends on the trade-off between temporal accuracy and computational efficiency. In the case of our dataset, we do not have a large amount of data, so we can dispense with computational efficiency and ensure that the annotations are correctly matched with the

events to try to guarantee maximum prediction accuracy.[20]

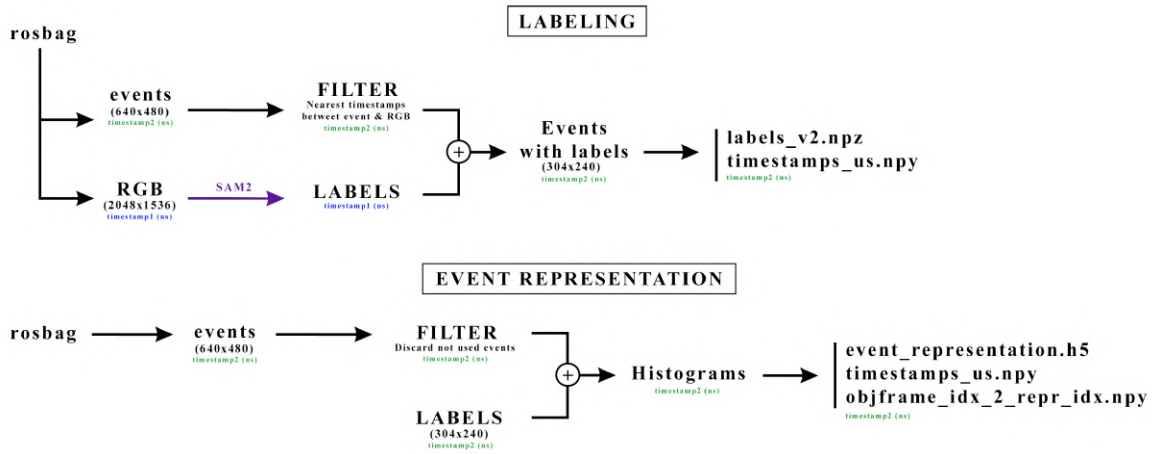



Figure 5.18: Step-by-step labeling scheme and event representation
21

[20] As a reference, for the algorithms used, it is recommended to use intervals of 50 us.

[21] Despite the fact that some files are identified as "*us*" which represents "*microseconds*", nanoseconds have been used during the project to guarantee fewer errors between the association of labels and events (it has not been changed to comply with the file names stipulated by the evaluated models).

## 5.2 Three-dimensional Dataset

To create a dataset composed of three-dimensional information (point cloud) of elements in a simple way and without the need for additional hardware, the NeRF technique has been used. NeRFs (Neural Radiance Fields) are an advanced technique in computer vision that allows three-dimensional reconstruction from 2D images (from images obtained by an RGB camera). They use neural networks to estimate the geometry and lighting of a scene and learn to map spatial coordinates and viewing directions to generate realistic views from any angle, using a rendering process to generate highly realistic 3D models.

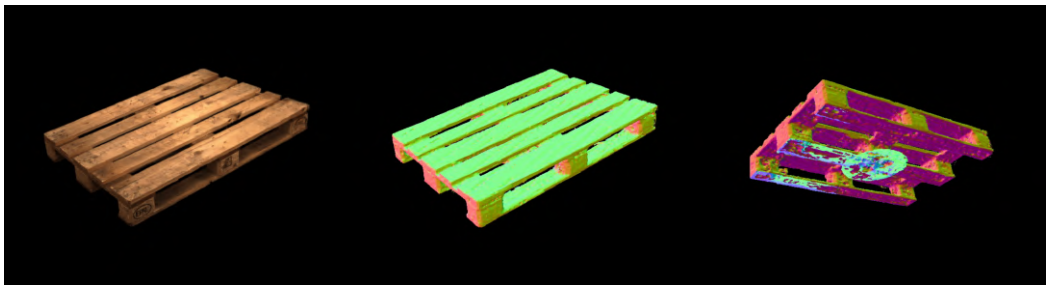
For their implementation, there are tools such as Instant-NGP (from NVIDIA) [24] and Nerfstudio [25], which allow NeRF models to be trained and rendered in an optimized way. Next, we explore their characteristics, the process of NeRF generation and the challenges overcome to achieve efficient reconstruction.

### Purpose

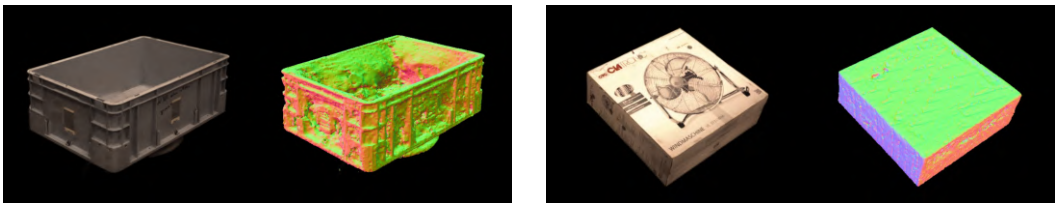
The purpose of using NeRFs in this study is to improve the representation of objects present in 2D datasets, generating three-dimensional versions of them. This will enrich prediction models by incorporating depth and geometry information, increasing the reliability of the inferences made.

Currently, object identification is based on 2D data and events, which introduces limitations in terms of accuracy and spatial understanding. By adding 3D data, it will be possible to obtain a more detailed and robust representation, optimizing the detection and classification of elements in future applications.

This is also a preliminary version to approach the creation of NeRFs with the event camera, as mentioned in Section 3.2. In the same way that in the case of the 2D dataset we rely on the RGB camera to create event annotations, the RGB camera could also be used to annotate events in three dimensions.



(a) Pallet



(b) Big KLT (c) Clatronic fan box

Figure 5.19: Examples of the 3D dataset created (Insulated model and mesh)

### 5.2.1 Process: Instant-NGP

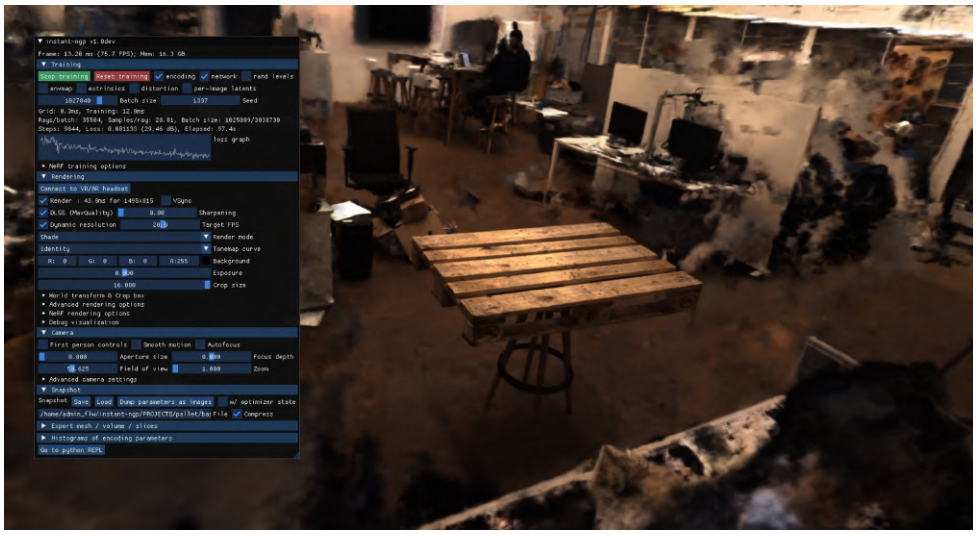
For this project, the Instant-NGP (Instant Neural Graphics Primitives) library, developed by NVIDIA, was finally used. This is a library designed to accelerate NeRFs training through the use of compressed data structures and advanced neural representation techniques. Its main advantage is computational efficiency, allowing models to be trained in a matter of minutes instead of the hours or days required by other methods.

The process of using Instant-NGP involves the following steps:

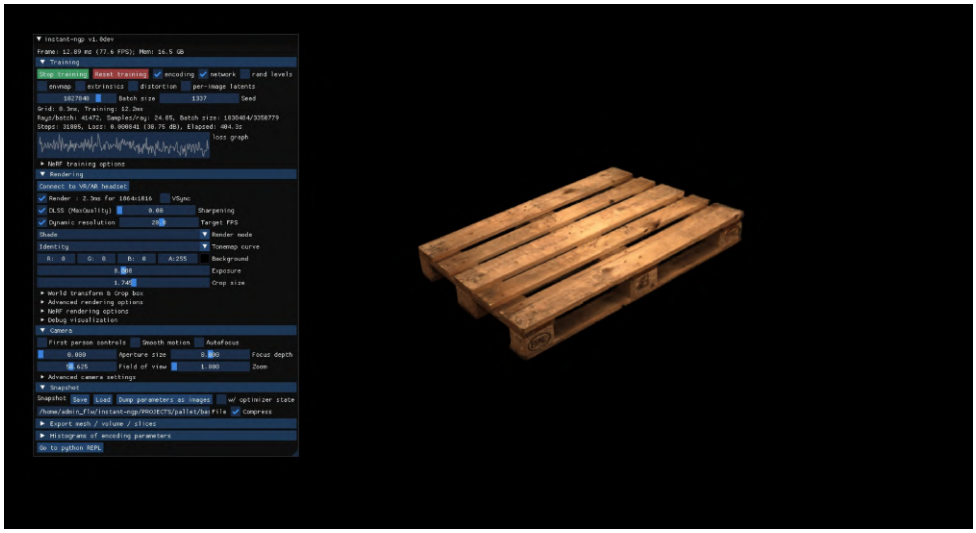
- **Preparing the dataset:** Images of the scene are collected from multiple angles and organized into folders following the recommended structure.
- **Preprocessing with `COLMAP`:** `COLMAP` is used to estimate camera positions and generate an initial point cloud.
- **Data conversion:** The `colmap2nerf.py` script is executed to transform the data into a format compatible with Instant-NGP.
- **Model training.**
- **Rendering and parameter adjustment:** Options such as batch size, dynamic resolution and shading modes are explored to optimize the final result.

As can be seen, the process of creating NeRFs is simple and requires some manual attention when adjusting and refining the results obtained. This process produces a 3D representation of the entire captured scene, which means that if you want to isolate a specific element, you should adjust the viewing depth as much as possible.[22]

[22]In addition, to isolate as much as possible the element you are trying to render, it is best to place it on the smallest possible surface or to hang it. In this way you can see more of the surface of the analyzed element.



(a) Complete scene



(b) Isolated element

Figure 5.20: 3D NeRF generation.

### 5.2.2 Possible problems and solutions

During the generation of NeRFs, different challenges arose, mainly in terms of computational consumption and quality of input data.

One of the main problems is the high computational requirement. When processing long videos, the load on the GPU becomes excessive, causing errors or long waiting times. To mitigate this problem, it is necessary to reduce the length of the videos or decrease the number of frames processed per second, thus reducing the computing requirements of the reconstruction process.

This is a very common problem, as when creating videos of the scenes to be input, there is a tendency to expand their duration in order to obtain every detail of the filmed object with precision. However, when the video is divided into frames, many very similar images are obtained. This means that more irrelevant information is being fed to the model, which saturates it and could even introduce noise at the output.

Another major obstacle is the inconsistent quality of the images. Photographs taken with blurred or poor lighting conditions affect the accuracy of the model. To solve this problem, images can be filtered, either manually eliminating low quality ones or by using automatic tools (such as "*Blurry Frame Remover*"[26] to optimize the filtering of noisy data.

On the other hand, the adjustment of hyperparameters is a critical aspect in achieving efficient reconstruction. Parameters such as "`aabb_scale`" directly influence the accuracy and efficiency of the model. This parameter is one of the most important when configuring the model. A high value expands the evaluated area, capturing more details at the expense of greater memory and processing consumption. On the other hand, reducing "`aabb_scale`" decreases the computational load but can cut out essential details, affecting the quality of the model.

# 6 DISCUSSION

Before analyzing the results, it is important to remember and bear in mind that LEOD is not a new detection model, but rather an optimized training framework for learning with fewer labels based on RVT. This optimization allows the model to learn not only from manual labels but also from unannotated data, reducing the dependence on exhaustive labeling. Furthermore, unlike conventional supervised training, LEOD filters and refines self-generated labels to minimize noise, thus improving the detection of moving objects and its generalization.

## 6.1 Preliminary study

Before training the LEOD model, it is essential to analyze the results of the paper in depth. This allows us to understand its behavior and handling, validate its effectiveness, and detect possible problems before investing computational resources in the implementation.

Regarding the configuration of the model when validating and training it, the only parameters specified are the *steps*, *batch time*, *learning rate* and *pseudo-labeling thresholds*. Despite being a very small part of the parameters that can actually be configured, they represent the most relevant modifications for a Machine Learning model. The following explains what each of them indicates:

- **Steps:** Indicates how many times the model weights are updated before training is complete. The number of steps is adjusted to avoid overfitting on limited data and to ensure that the model learns well.
- **Batch size:** The batch size is adjusted according to the GPU capacity and the stability of the training. In LEOD, `8` is an optimal value to work with event data without compromising memory or stability (however, **for our dataset, it is not possible to reach that amount due to the low amount of data compared to *Gen1***). In addition, for the same number of steps, a greater or lesser number of epochs will be achieved depending on the indicated batch size value.
- **Learning rate:** Defines how much the model weights are adjusted in each update. This depends on the dataset time used (for *Gen1* a `learning rate` of `0.0005` is used). A high `learning rate (>0.001)` indicates rapid convergence, although more noise can be created. On the other hand, a low `learning rate (<0.001)` stabilizes the training, although it can take a long time to converge or remain at local minima.
- **Pseudo-labeling threshold:** It tries to adjust itself according to the class of objects to maximize precision without losing important detections.

The other general parameters and those involved in the dataset are not mentioned in the paper.

On the other hand, different variants of the RVT (Recurrent Vision Transformer) model have been used, specifically **RVT-s (Small)** and **RVT-b (Base)**. However, in the case of RVT-b it is only used to make comparisons in the state of the art. It is RVT-s with which all the main experiments have been carried out (WSOD and SSOD).

### 6.1.1 Supervised Baseline

The Supervised Baseline is the reference point that will be used in the different experiments to evaluate the evolution of LEOD and RVT with pseudolabels. This refers to the training of the model with the labels generated in the preprocessing of the information (model trained without pseudolabels). As this labels are considered as Ground Truth (GT) in the evaluation of the other models, the Supervised Baseline can be considered as the Ground Truth between training outputs.

### 6.1.2 Semi-Supervised Object Detection (SSOD)

Semi-Supervised Object Detection is a learning approach in which a model is trained with a combination of labeled and unlabeled data. Its main objective is to improve the ability to detect objects without the need for a large volume of manually annotated data, which makes the training process more efficient and less costly. The way SSOD works is based on the idea that even if only part of the data has real labels, the model can extract useful information from the unlabeled data through strategies such as:

- **Pseudo-labeling:** The model generates labels for unannotated data and uses them in its own training.
- **Self-training:** The model is first trained with labeled data and then uses its own predictions on unannotated data to improve its learning.
- **Consistency Training:** This method imposes restrictions on the model to make consistent predictions on modified versions of the same data.

In the case of event-based vision, it is particularly useful because event sensors generate continuous data streams at very high frequencies (thousands of times per second). Manually annotating all these events is extremely costly and impractical, so SSOD allows the model to learn with a limited amount of real labels, taking advantage of unannotated data through pseudo-labels and prediction refinement.

As can be seen in Figure 6.1, it shows how LEOD manages to significantly improve performance with very little annotated data. In experiments, when training a standard supervised model (RVT-s) with only 1% labels, around 15.8% of **mAP!** is achieved. This is to be expected, since, in an event-based detection environment, where the data is highly dynamic and difficult to label accurately, a model that depends exclusively on the few available labels simply does not have enough information to generalize well.

Furthermore, it can be seen that as the percentage of labeled data increases, the difference between LEOD and the Supervised Baseline line remains constant, showing significant improvements. With only 5% of data annotated, LEOD achieves 38.0% of mAP, surpassing the traditional supervised model by more than 8 percentage points.

The most interesting thing is that, when 10% of labeled data is reached, LEOD almost matches the performance of the fully supervised model (100% of the labels). While the supervised baseline with 10% of data reaches 35.1% of mAP, LEOD achieves 40.7%, approaching the performance that could previously only be obtained with the entire annotated dataset.

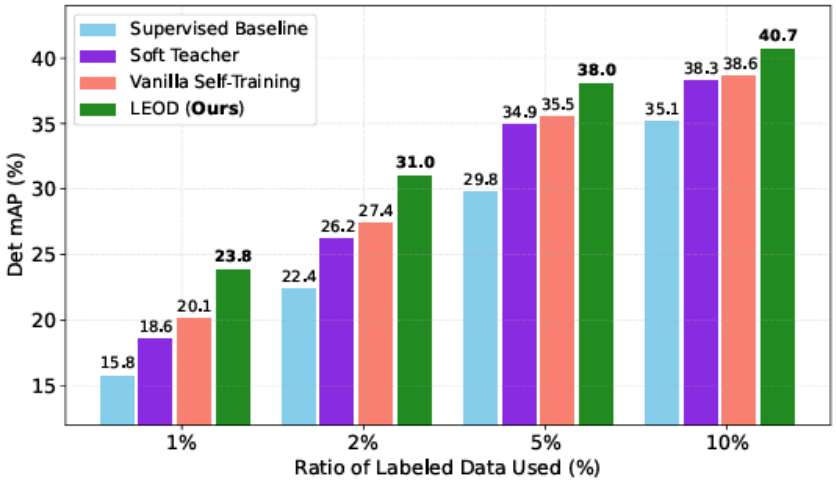
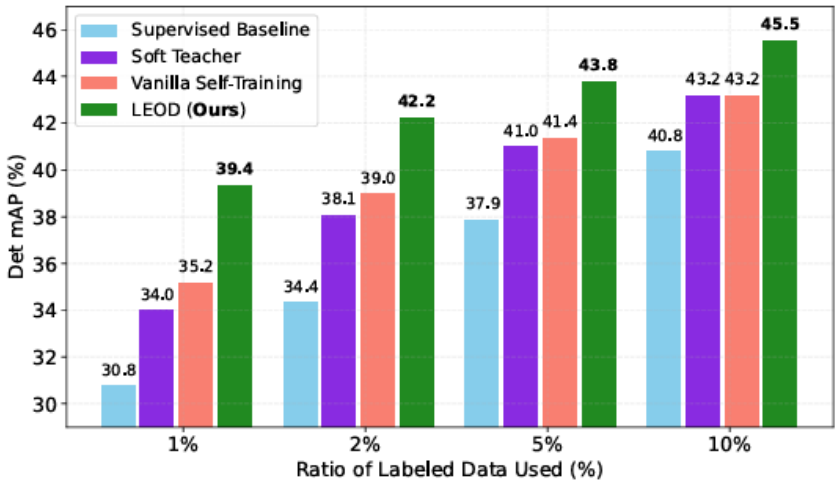



Figure 6.1: Gen1 Semi-Supervised Object Detection(SSOD) results [8]

### 6.1.3 Weakly-Supervised Object Detection (WSOD)

While in SSOD some sequences are fully annotated and others have no labels at all, in WSOD all sequences contain labels, but they are scattered over time. This means that SSOD learns from complete segments of annotated data and extrapolates its knowledge to the unlabeled ones through pseudo-labeling, while WSOD forces the model to infer information between separate labels within each sequence.

Inferring information is beneficial between sparse labels because it forces the model to learn general patterns and not rely exclusively on explicit examples, thus improving its generalization ability. Instead of memorizing only the labeled cases, it develops the ability to reconstruct the behavior of objects over time, detecting their continuity and evolution even in frames without annotations. This is especially useful in event-based vision, where the data is continuous and object detection should not depend solely on specific instances, but on an overall understanding of the movement and structure of the scene.




(a) Gen1 weakly-supervised object detection (WSOD) results

Figure 6.2: Gen1 Weakly-Supervised Object Detection (WSOD) results [8]

Figure 6.2 of the paper presents the results of the WSOD approach on the *Gen1* dataset, comparing the performance of LEOD with the Supervised Baseline and other self-training methods.

The data show that when the base model is trained with 1% of the labeled data, its performance is 30.8% of mAP. This indicates an expected limitation in a sparse labeling environment, where the information provided to the model is minimal and may not be sufficient for accurate detection. In comparison, LEOD achieves 39.4% of mAP, representing an improvement of +8.6 points.

This behavior is maintained as the percentage of labeled data increases. With 5% of annotated data, LEOD obtains 43.8% of mAP, and with 10% of data, it reaches 45.5%, approaching the performance of a fully supervised model, which achieves 46.5%.

## 6.2 Performance study of the submodels

Given that LEOD is based on RVT, it is relevant to compare its performance under different configurations to evaluate under what conditions LEOD can outperform RVT or if, in contrast, it presents a decrease in its performance.

For this analysis, the LEOD and RVT models have been evaluated in their *base*, *small* and *tiny* versions, running inference and validation tests on a fully labeled dataset. Average Precision (AP) has been measured at different IoU thresholds and on different object scales (*small*, *medium* and *large*) to analyze its sensitivity to changes in the size of the detected objects.

The results obtained in the validation phase are as follows:

| **Modelo** | **AP** | **AP@50** | **AP@75** | **AP_L**[23] | **AP_M**[24] | **AP_S**[25] |
|---|---|---|---|---|---|---|
| **RVT-base** | 0.4949 | 0.7479 | 0.5326 | 0.5451 | 0.5605 | 0.4209 |
| **RVT-small** | 0.4868 | 0.7399 | 0.5222 | 0.5580 | 0.5540 | 0.4122 |
| **RVT-tiny** | 0.4577 | 0.7105 | 0.4878 | 0.5497 | 0.5224 | 0.3837 |

Table 6.1: Comparación de rendimiento de modelos RVT en diferentes métricas AP.

The analysis of the overall AP values shows that, within the RVT series, the `RVT-base` model obtains the best precision with a value of 0.4949. The precision decreases slightly when moving on to the smaller models, with `RVT-small` reaching 0.4868 and `RVT-tiny` obtaining 0.4577. The difference between `RVT-base` and `RVT-small` is relatively small (approximately 0.8%), while the difference widens considerably when comparing `RVT-small` with `RVT-tiny` (around 3%).

These results suggest that, if a balance between precision and computational efficiency is sought, the `RVT-small` model may be a suitable option, offering a good balance by maintaining a precision close to the base model with a significant reduction in size and complexity. For this reason, it is the one chosen for the study in Section 6.3. The `RVT-tiny` model, however, although it performs less well in terms of precision, may be preferred in situations where computing resources are highly limited and processing speed takes priority over maximum precision.

## 6.3 Study with FLW Dataset

This study will not perform an exhaustive analysis of how each of the configurable parameters of the model affects the result. Only the parameters identified as relevant in the original article have been modified, in order to evaluate the performance of the algorithm in the FLW dataset and establish the main differences between the RVT and LEOD methods for that dataset. The adopted configuration corresponds to the best identified in the previous sections, taking into account both the similarities and the differences between our dataset and the *Gen1* dataset.

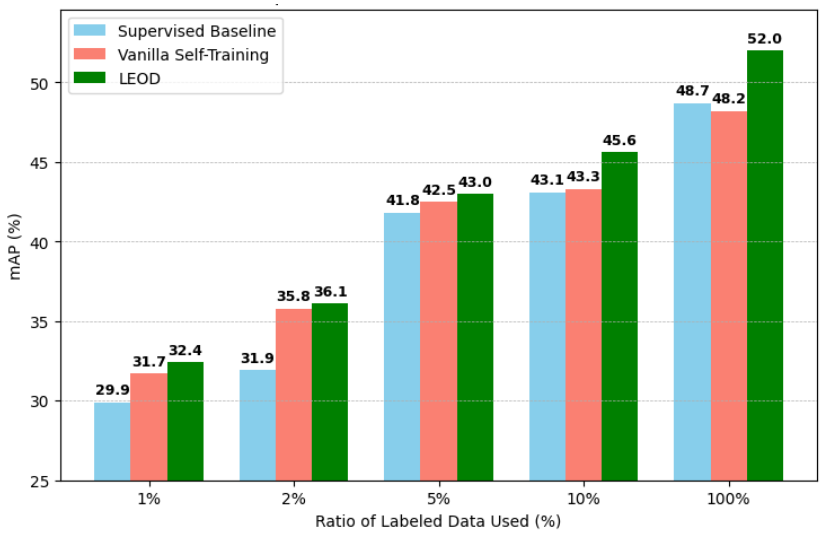



Figure 6.3: Weakly-Supervised Object Detection (WSOD) for FLW Dataset

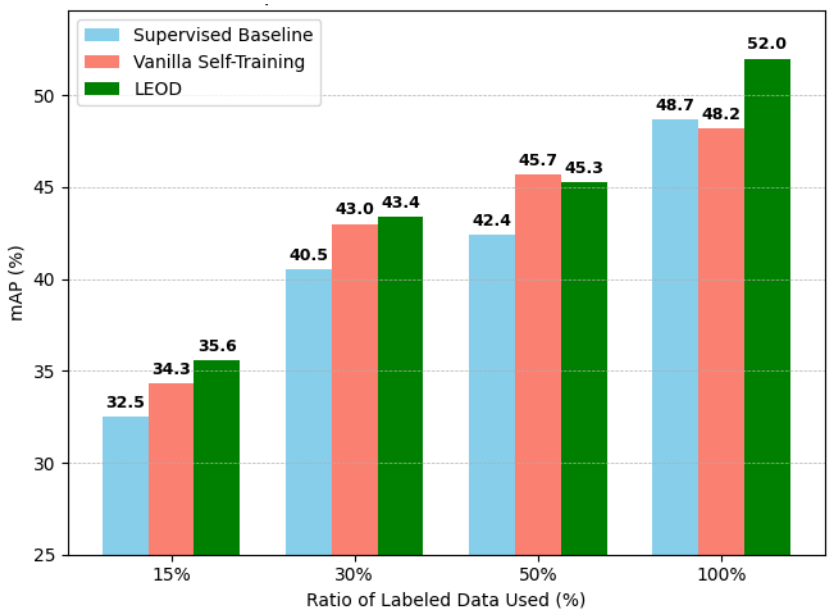



Figure 6.4: Semi-Supervised Object Detection (SSOD) for FLW Dataset

It can be seen that, for small amounts of training data, the WSOD method provides greater precision compared to SSOD. This result was predictable according to the analysis previously carried out in Section 6.1.

At first sight, the results obtained are in line with expectations and, particularly in the case of the WSOD method, they even exceed the results reported in the original article for the *Gen1*

and *1Mpx* datasets. This improvement could be attributed to the difference in the amount of data available in each dataset. In the context of the present project, the dataset is small and the scenes are limited to the same environment with variations only in the movements of the objects to be predicted. This means that the model does not need to generalize excessively, which favors precision. It is likely that, as the size of the dataset increases with more information, these results could even improve, although the magnitude of the improvement would be expected to be smaller.

The only case in which no improvements are observed is when the model is trained with only 1% of the data. This phenomenon can be explained by the fact that these experiments were carried out using a single round of self-training. As the original researchers indicate, increasing the number of pseudo-training rounds significantly improves the precision achieved, with two being the optimal number of rounds suggested. In fact, the evidence reviewed indicates that, for lower ratios (that is, reduced percentages of initial training data), there is considerably more room for improvement in each additional round of self-training. Therefore, to optimize these results, it would be advisable to run between two and three rounds with the generated pseudolabels.

| **Rounds** | 1% | 2% | 5% | 10% | 100% | P. (1%) | P. (2%) | P. (5%) | P. (10%) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 38.1 | 41.1 | 43.1 | 45.3 | 48.5 | 0.65 | 0.69 | 0.74 | 0.79 |
| 2 | 39.4 | 42.2 | **43.8** | **45.6** | **48.7** | 0.72 | **0.75** | **0.77** | **0.81** |
| 3 | **39.5** | 42.2 | 43.6 | 45.4 | 48.6 | 0.72 | 0.74 | 0.74 | 0.76 |

Figure 6.5: Number of self-training rounds used in LEOD paper. Model AP in left side and pseudolabels AP in right side (P.).

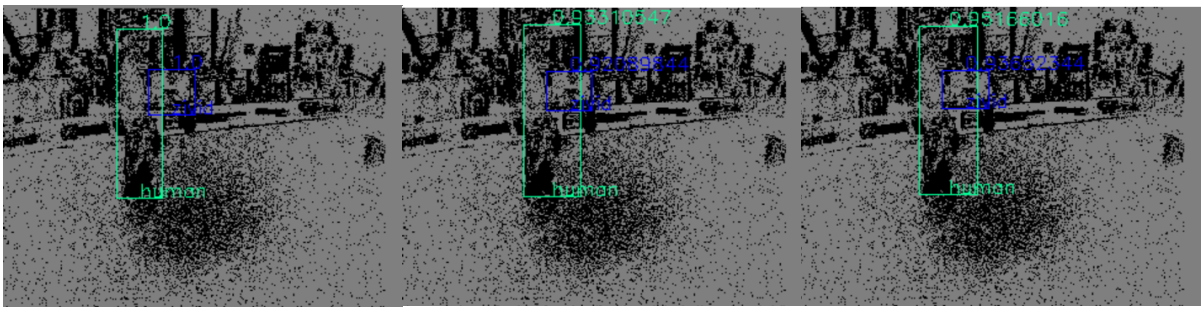
However, the results obtained with SSOD are not so remarkable and do not manage to exceed those reported in the original article for the *Gen1* and *1Mpx* datasets. As in the previous case, this difference could be mainly due to the limited amount of data available. For example, when 15% of the total available scenes are used for training, the effective number of scenes is reduced to just 2 ($17\times0.15 = 2.55 \rightarrow 2$), in contrast to the 47 scenes used in the *Gen1* dataset with a 10% selection ($470\times0.1 = 47$). This limitation makes it considerably more difficult for the model to obtain comparable results with only two available scenes.

Despite this, the use of the LEOD method for the prediction of logistic objects, such as those used in this study, is considered to be successful. This method not only reduces the future need for labeling, but also achieves better prediction precision without significantly compromising precision, as shown in Section 6.3.

### Precision vs accuracy

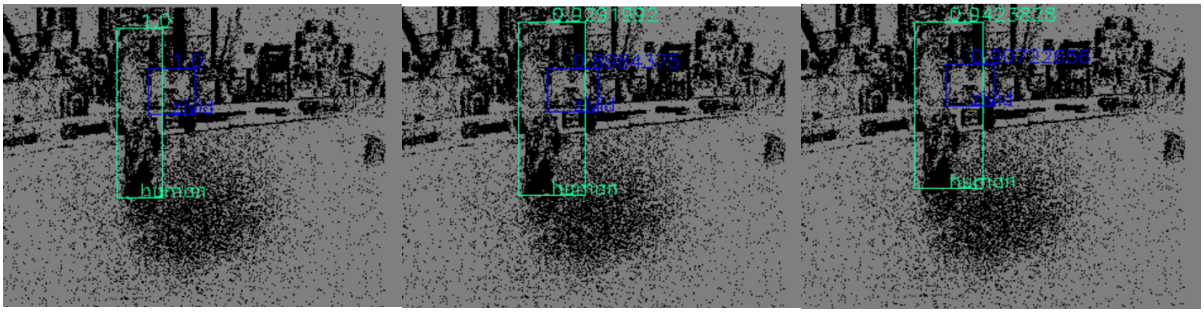
The generation of pseudolabels implies that the original training labels (if any) are modified based on the first prediction obtained in the baseline training. This means that if initial labels are incorrectly adjusted due to poor initial training, they could become trapped in local minima, thus limiting their progress. Therefore, it is essential to carry out a solid baseline training that allows for a more robust subsequent performance and minimizes adverse situations.

In the context of the models trained in this project, it is evident how the quality of the results obtained in the pseudolabels depends directly on the quality of the initial data used.



**a1.** Ground Truth (GT) **a2.** Supervised Baseline **a3.** SWOD

**(a)** Bounding boxes visualization for 10% training data

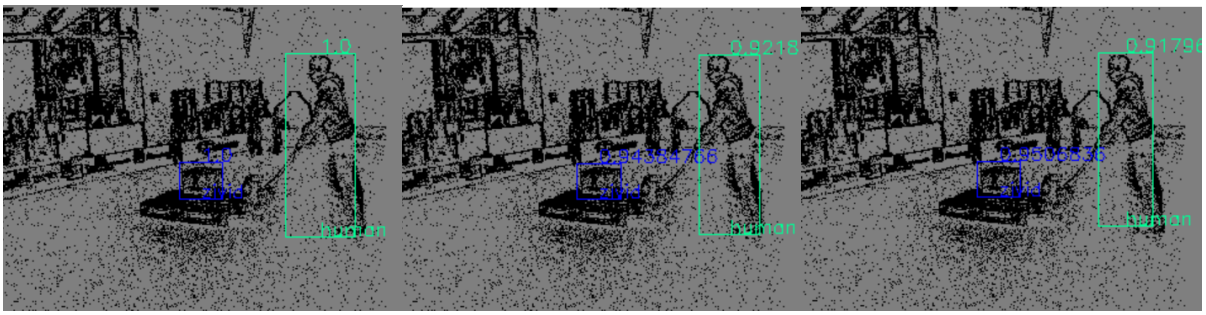


**b1.** Ground Truth (GT) **b2.** Supervised Baseline **b3.** SWOD

**(b)** Bounding boxes visualization for 1% training data

Figure 6.6: Visual evaluation in same frame for generated output bounding boxes with different models

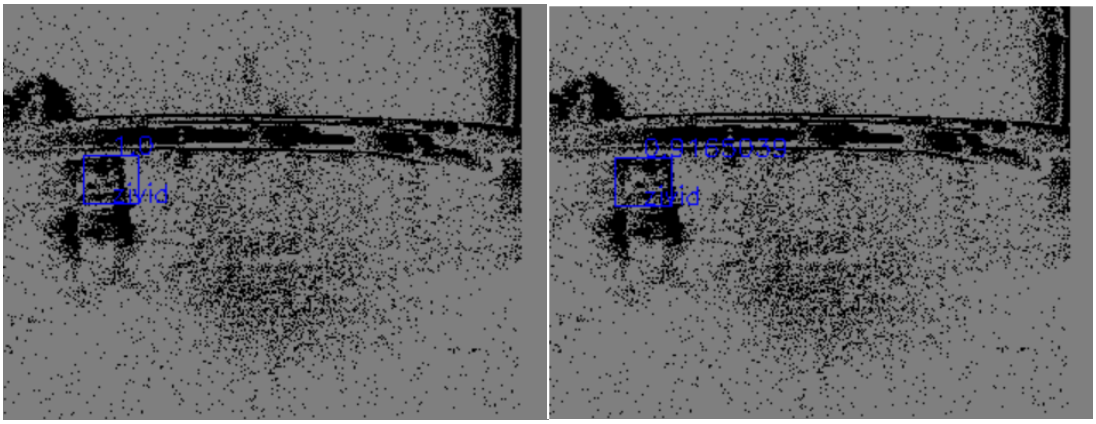
As can be seen in Figure 6.6, the pseudolabels generated in each round of training have different percentages of precision. In addition, this precision depends directly on the volume of data used in the initial training; thus, in the same frame, the model trained with 10% of the data shows better precision in both rounds than the one trained only with 1%. In the specific case of the human object in *(a)*, it can also be seen how in the Supervised Baseline round a precision of 93% is obtained, while in the first round of self-training this precision is 95%. However, there are also cases where the precision between the baseline and self-training rounds decreases, as illustrated in Figure 6.8.



**a.** Ground Truth (GT) **b.** Supervised Baseline Figure 6.7: **c.** SWOD

Figure 6.8: Visual evaluation in same frame with visual precision improvement

However, it is important to note that, although a decrease in precision measured as AP is observed in some cases, a real improvement could be visually appreciated due to the fact that the Ground Truth label used could lack sufficient precision. Similarly, in Figure 6.9 a reduction in AP can be seen, although what is really happening is an increase in accuracy due to the more adequate displacement of the bounding box, evidencing the algorithm's capacity to correct initial labeling errors. This underlines the importance of accurate and precise labeling to improve critical capacity and confidence in analysis.



**a.** Ground Truth (GT) **b.** WSOD

Figure 6.9: Visual evaluation in same frame with visual accuracy improvement

# 7 CONCLUSIONS

## 7.1 Conclusions

In conclusion, the four main objectives of the project are evaluated: automation of preprocessing and annotation correction, prediction accuracy improvement, analysis of the LEOD model in comparison with previous models and creation of a 3D dataset.

- **Automation of preprocessing and annotation correction.**
  The generation of annotations has been significantly optimized by automating preprocessing and its subprocesses, reducing manual intervention and computation time, and saving disk space compared with previous annotations. The use of *SAM2* and the subsequent transformation of bounding boxes make it possible to obtain annotations that are sufficiently reliable for model evaluation.

- **Improved prediction accuracy after pseudo-labelling.**
  The pseudo-labelling process employed by the model not only achieves good precision in the resulting predictions, but has also been shown to improve the accuracy of labels that are of poor quality or loosely adjusted. It can therefore serve as a solid starting point for subsequent rounds of self-training and allows greater flexibility in the manual generation of ground-truth labels.

- **Evaluation of the LEOD model with FLW laboratory data.**
  Trained with scenes from the FLW laboratory, LEOD outperforms its previous results in *Weakly-Supervised Object Detection* when compared with the Gen1 and 1Mpx datasets, whereas in *Semi-Supervised Object Detection* it does not reach the same level of success. These differences are mainly related to the unequal volume of data and annotations between datasets and to the nature of the scenes. The FLW dataset represents a structured, controlled environment with very similar backgrounds, unlike the Gen1 and 1Mpx datasets, which were recorded outdoors on automobile traffic routes.

- **Creation of a 3D dataset for object prediction.**
  The new three-dimensional dataset, in line with advances such as *EventNeRF*, lays the foundations for dimensional comparison with future 3D models, boosts precise object prediction from an additional dimension in laboratory environments with event cameras, and opens the door to further lines of research in this direction.

All of these overall findings derived from the in-depth analysis of the model's behaviour represent a major advance in predicting logistic objects with event-based cameras in a controlled scenario, and they provide a solid foundation for future related projects.

## 7.2 Outlook

### 7.2.1 Multiple elements prediction

In the current version of the LEOD model, the algorithm operates with the Gen1 and 1Mpx data sets. These data sets are commonly used to validate models with similar characteristics and objectives. However, the current implementation has a limitation in terms of the maximum number of object types it can predict: two for Gen1 ("*ped*" and "*car*") and three for 1Mpx ("*ped*", "*car*" and "*cycle*"). This limitation restricts the model's ability to generalize in more complex or diversified scenarios.

In the specific context of the present project, we have initially worked only with two categories of objects ("*hum*" and "*zivid*"), following a methodology analogous to that applied in the Gen1 set. This approach constitutes a first contact aimed at evaluating the possibilities of the model in broader logistic applications. However, the new scenes captured to create a logistics dataset of our own contain more than fifteen different types of logistics objects, together with other relevant elements such as transport units (*wagons*) and people. In view of this reality, it is clear that the predictive capacity of the model needs to be expanded, incorporating a greater number of object types for their effective integration into more complete data sets that are more representative of real situations.

To address this challenge, significant modifications to the model's original code will be essential. Likewise, the automations developed during the initial phase of the project will have to be adjusted to achieve a greater degree of generalization, thus facilitating the annotation of a broader spectrum of elements. This process not only involves technical adjustments, but also raises the need to explore new strategies and innovative methods to guarantee accuracy, efficiency and consistency in the generation of annotations. Therefore, this expansion represents an important area of future research that could lead to significant improvements in the overall performance of the LEOD model.

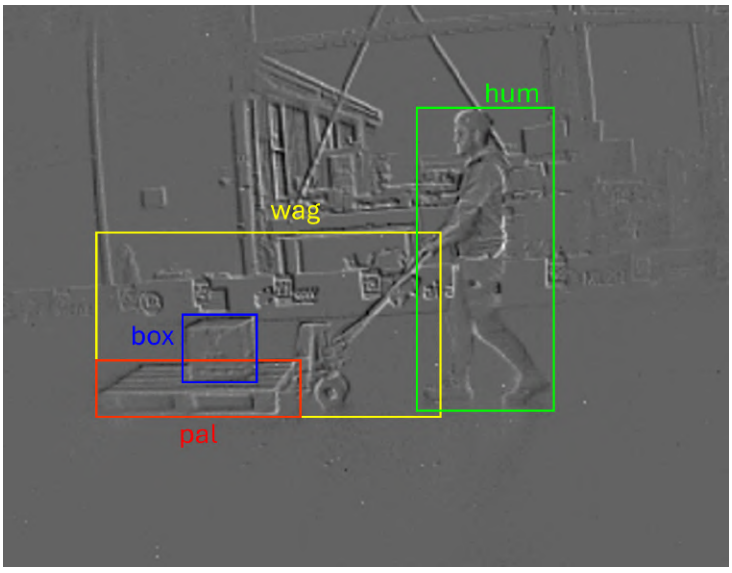



Figure 7.1: Multiple object detection in reconstructed event-based image

### 7.2.2 Extension of the LEOD Algorithm for 3D

The modifications made to the LEOD algorithm code (extension of the RVT model) in the present project have allowed a deeper understanding of its internal functioning. A possible future line of research would be the incorporation of a new variable in the model by replacing the RGB camera with a camera with a depth sensor (RGB-D) or implementing *EventNeRFs* as in Section 3.2. The implementation of this technology would make it easier to capture an additional dimension in the data collected, providing more complete information about the morphology of objects through depth detection. This would make it possible to significantly improve the prediction and segmentation of logistical objects in complex three-dimensional scenarios, especially in combination with the previously generated 3D dataset.

To achieve this integration, it would be necessary to adapt the current preprocessing and incorporate new three-dimensional data that modifies the representation of events. Additionally, it would be advisable to explore the synchronization and correlation of information obtained from multiple event cameras (two or more), aligning it with the incorporated depth values. This strategy would offer diverse complementary perspectives on the captured scenes, increasing the robustness of the model and optimizing its capacity to make accurate predictions in complex three-dimensional environments.

Although the integration of these technologies would imply greater complexity at the hardware and implementation level, the impact on the accuracy and reliability of the model could be considerable, consolidating the LEOD algorithm as an effective tool for prediction and analysis in advanced logistics applications.

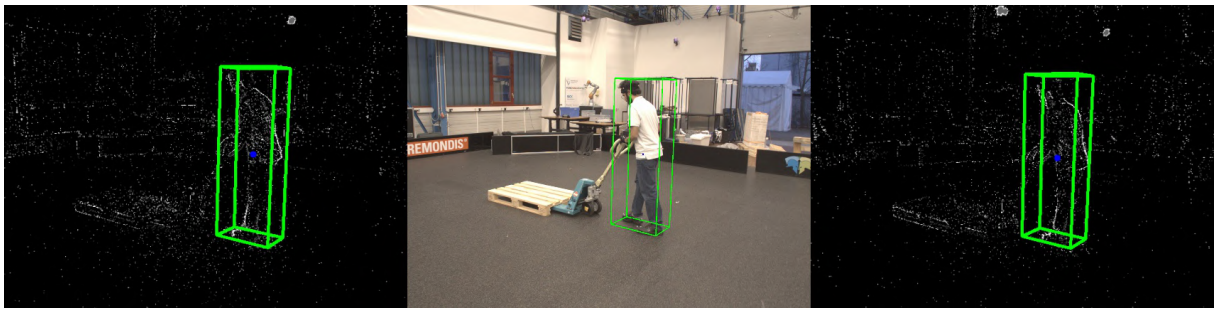



Figure 7.2: 3D annotations for the algorithm extension

### 7.2.3 Collecting a Greater Amount of Data

One possible future implementation would be to replace the current RGB camera of 58 fps (compared to the 165 million events per second that the event camera can capture) with a new one with a higher frame rate. This change could be key to increasing the amount of information obtained per scene, which would have a significant impact on the system's ability to capture details of events in real time. Given that the events generated by the event camera are asynchronous, the amount of annotations available per scene is directly limited by the number of frames captured by the RGB camera. Currently, the annotation generation process is based on the reconstruction of events in time intervals corresponding to the frames of the RGB camera. The comparison and transformation of points between these two data sources allows the labels for each moment in time to be generated. Therefore, an increase in the rate of captured frames

would imply an increase in the number of processable timestamps, which, in turn, would result in a greater number of annotations throughout the temporal flow of each recorded sequence. This would not only improve the accuracy of the models, but would also allow for more efficient training and better tracking of the movements in the scene.

### 7.2.4 Replacing markers for human segmentation

As a proposed procedure aimed at saving time and even money, a change in the method used to detect humans is suggested. In the first videos inherited from the project, only the information relating to the position of the logistic object to be segmented was available. However, in the new videos recorded to complement the data set used in this work, additional information on the position of various parts of the human body (hands, feet, torso and head) has been incorporated through the use of physical markers. Thanks to Optitrack technology [27] and the arrangement of cameras in the recording area, it is possible to visualize these markers and, by triangulation, obtain their exact position in a three-dimensional space.

Although it has not been necessary to exploit this positional information within the framework of this project, since all annotations have been generated using 2D segmentation models, the inclusion of this data in the dataset provides a richer basis for future research focusing on three-dimensional predictions.

The use of these markers during recording involves certain operational difficulties, such as the need to continuously verify that the Optitrack cameras correctly detect the markers, to ensure their selection within the tracking system, in addition to the manual placement of the markers on the subject's clothing, with the risk of them coming off or moving during recording.

To address these drawbacks and streamline the data capture process, it is proposed to replace the hardware-based method with a purely software-based approach. Alternatives such as *MmPose* [28] or *Human Pose Detection*[29] represent viable options for the detection and segmentation of the human body in three dimensions. Although the exact precision of these models (Figure 7.3) requires a more detailed study (beyond the scope of the present project), preliminary evaluations suggest that they offer robust detection and even provide new virtual markers that can more effectively represent human movement.

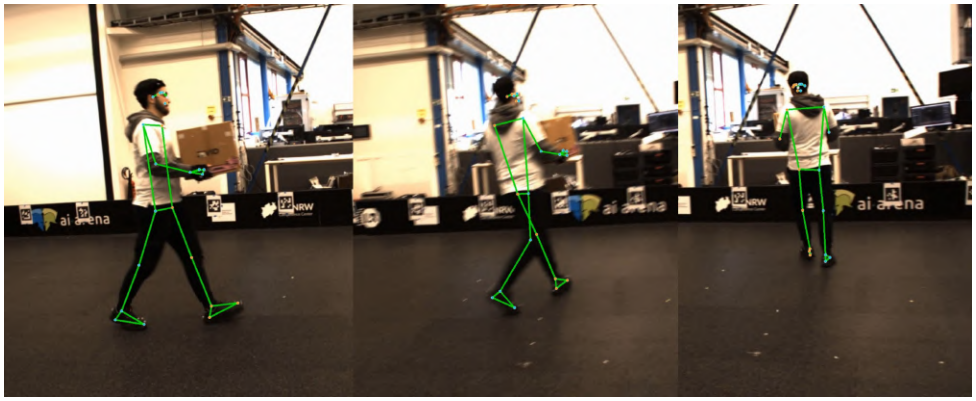


Figure 7.3: Human Pose Detection model performance. [29]

Although in the current phase of the project only 2D bounding boxes have been required and in the future it would be possible to move towards 3D bounding boxes generated by this method,

the wealth of additional information could be used to predict human behavior. Furthermore, the transition to a software-based approach would make it possible to record a greater number of scenes in less time, optimize storage space, and avoid the need to repeat recordings due to synchronization failures between hardware and software. It would also open up new possibilities for the extraction of advanced information on human behavior in the context of captured scenes.

# REFERENCES


[1] Mathias Gehrig and Davide Scaramuzza. "Recurrent Vision Transformers for Object Detection with Event Cameras". In: Proceedings IEEE/CVF Conference Computer Vision and Pattern Recognition (CVPR). 2023. URL: https://arxiv.org/abs/2212.05598.

[2] Viktor Rudnev et al. "EventNeRF: Neural Radiance Fields from a Single Colour Event Camera". In: Computer Vision and Pattern Recognition (CVPR). 2023.

[3] Wikimedia Commons Contributors. DVS Sensor Image. 2024. URL: https://commons.wikimedia.org/w/index.php?curid=97727937.

[4] Wikimedia Commons Contributors. Event-Based Vision System Diagram. 2024. URL: https://commons.wikimedia.org/w/index.php?curid=149386066.

[5] Wikimedia Commons Contributors. Neuromorphic Camera Concept. 2024. URL: https://commons.wikimedia.org/w/index.php?curid=142141576.

[6] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. "Reconstructing video from event-based cameras with sparse latent representations". In: arXiv preprint arXiv:1812.07605 (2018). URL: https://github.com/cedric-scheerlinck/dvs_image_reconstruction/blob/master/images/thumbnail.png.

[7] Federico Paredes-Vallés, Jesse Hagenaars, and Guido de Croon. "Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks". In: arXiv preprint arXiv:2106.01249 (2021). URL: https://arxiv.org/abs/2106.01249.

[8] Ziyi Wu et al. "LEOD: Label-Efficient Object Detection for Event Cameras". In: arXiv preprint arXiv:2311.17286 (2024). URL: https://arxiv.org/abs/2311.17286.

[9] Clarivate Analytics. Web of Science. Accessed: 2024-02-26. 2024. URL: https://www.webofscience.com/.

[10] Guillermo Gallego et al. "Event-based vision: A survey". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 43.6 (2020), pp. 154–180. DOI: 10.1109/TPAMI.2020.3008413. URL: https://arxiv.org/abs/1904.08405.

[11] Clarivate Analytics. Web of Science Record: WOS:000728561300013. 2024. URL: https://www.webofscience.com/wos/alldb/full-record/WOS:000728561300013.

[12] Elias Mueggler et al. "The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM". In: The International Journal of Robotics Research 36.2 (2017), pp. 142–149. DOI: 10.1177/0278364917691115. URL: https://arxiv.org/abs/1610.08336.

[13] Clarivate Analytics. Event-Camera Dataset: Pose Estim, Visual Odom and SLAM. 2024. URL: https://www.webofscience.com/wos/alldb/full-record/WOS:000399558300002.

[14] Zili Zhang et al. “Intelligent Machinery Fault Diagnosis With Event-Based Camera”. In: IEEE Transactions on Industrial Informatics (2023). DOI: 10.1109/TII.2023.3242271. URL: https://ieeexplore.ieee.org/document/10086648.

[15] Clarivate Analytics. Intelligent Machinery Fault Diagnosis With Event-Based Camera. 2024. URL: https://www.webofscience.com/wos/alldb/full-record/WOS:001142900000058.

[16] First Author and Second Author. “Title of the Paper for arXiv:2311.01881”. In: arXiv preprint arXiv:2311.01881 (2023). URL: https://arxiv.org/abs/2311.01881.

[17] Ziyi Wu et al. LEOD nstalation file. 2024. URL: https://github.com/Wuziyi616/LEOD/blob/main/docs/install.mdD.

[18] Liyuan Pan et al. “High Frame Rate Video Reconstruction Based on an Event Camera”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 42.11 (2020), pp. 2780–2794. DOI: 10.1109/TPAMI.2019.2963386. URL: https://arxiv.org/abs/1903.06531.

[19] Cedric Scheerlinck et al. “Fast Image Reconstruction with an Event Camera”. In: Proceedings IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). 2020, pp. 156–163. DOI: 10.1109/WACV45572.2020.9093435. URL: https://rpg.ifi.uzh.ch/docs/WACV20_Scheerlinck.pdf.

[20] First Author and Second Author. “Title of the Paper for arXiv:2408.00714”. In: arXiv preprint arXiv:2408.00714 (2024). URL: https://arxiv.org/abs/2408.00714.

[21] Landing AI. Agentic OD Demo. https://va.landing.ai/demo/agentic-od. 2023.

[22] Robotics and Perception Group, University of Zurich. GEN1 Preprocessed Dataset. Dataset available at https://download.ifi.uzh.ch/rpg/RVT/datasets/preprocessed/gen1.tar. 2024.

[23] Blosc Developers. HDF5 Blosc: A High-Performance Compression Filter for HDF5. https://github.com/blosc/hdf5-blosc. 2024.

[24] NVlabs. instant-ngp. https://github.com/NVlabs/instant-ngp. 2021.

[25] NeRF Studio. NeRF Studio Documentation. https://docs.nerf.studio. 2023.

[26] DeepShakes Developers. BlurryFrameRemover: Removing Blurry Frames from Videos. https://github.com/deepshakes/BlurryFrameRemover. 2024.

[27] OptiTrack. OptiTrack: Motion Capture Systems. https://www.optitrack.com/. 2024.

[28] OpenMMLab. MMPose: OpenMMLab Toolbox for Pose Estimation. https://github.com/open-mmlab/mmpose. 2024.

[29] Nitin Gour. Human Pose Detection using Deep Learning. https://github.com/nitingour1203/human_pose_detection. 2024.

[30] Facebook Research. SAM2. https://github.com/facebookresearch/sam2. 2023.

[31] Landing AI. Vision Agent. https://github.com/landing-ai/vision-agent. 2023.

# APPENDIX

## A Setup

Requirements:

- Integrated PC with:
  - **Graphics memory:** `NVIDIA GeForce RTX 4090`
  - **Random Access Memory (RAM) memory:** `64 GB (32 GB + 32 GB)`[26]
  - **Read-Only Memory (ROM) memory:** `2 TB`[27]
- **Operating System:** `Ubuntu 22.04 Jammy Jellyfish`[28]
- `Compute Unified Device Architecture (CUDA) 12.6`
- Image segmentation libraries such as `Segment Anything Model 2 (SAM2)` [20][30] and `Agentic Object Detection`[21][31]
- Computer vision and machine learning libraries such as `OpenCV, PyTorch` and other related libraries.
- **NeRFs libraries** such as `NerfStudio` [25] and `Instant-NGP` [24]
- **Two event-based cameras** DVXplorer Lite and **one RGB camera** U3-3270CP Rev.2.2.

[26]It is interesting to have a high computational load capacity since the use of Neural Radiance Fields (NeRF)s as required in the project needed it

[27]It is important to have a large memory capacity as the Gen1 and 1Mpx datasets occupy 198 GB and 381 GB respectively when uncompressed. On the other hand, it must be taken into account that each element of the dataset obtained from NeRFs occupies from 15 GB to 20 GB. In addition, each of the raw data obtained directly from the recordings occupies approximately 6 to 12 GB, not counting the final preprocessed data.

[28]Please note that ROS Noetic is not compatible with this version of Ubuntu, so for some processes it will be necessary to operate through an Ubuntu container.

# B More Implementation Details

### Dataset Configuration

When creating the dataset used in cases where pseudolabels are used, it is useful to introduce the scenes where the best labels exist in the */test* and */val* folder. This is because they are the only two folders in the dataset that will not be modified by the pseudo-labeling and they are the ones that will dictate the accuracy at all times, as they act as Ground Truth.

### Train Configuration

**General:**

- learning_rate = 0.0005
- batch_size = 2
- num_workers = 2
- confidence_threshold = 0.01
- tta = disable

**For WSOD:**
In baseline training, 200k, 300k, 400k and 400k steps have been used in the cases of radio 0.01, 0.02, 0.05 and 0.1 respectively.

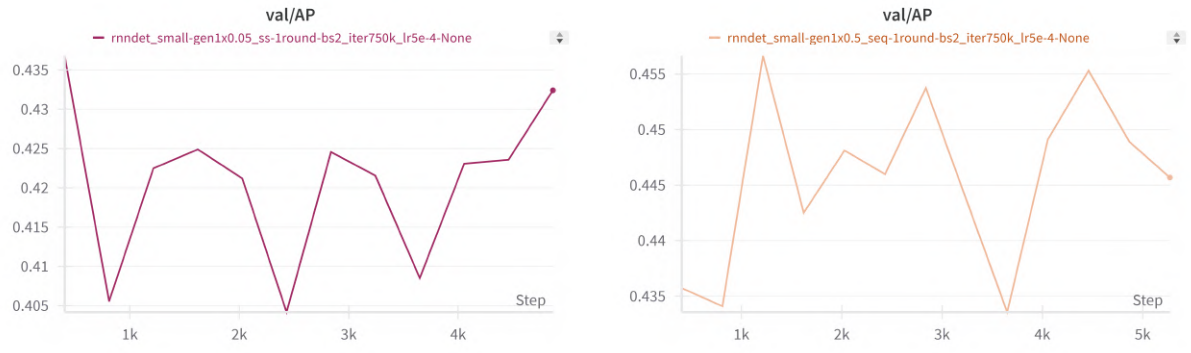
In training sessions with RVT (`rnndet`) and LEOD (`rnndet-soft`) 100k, 100k, 200k, 200k steps have been used, respectively. This has been the case because after several tests it has been considered that good results are achieved for the amount of time available. However, it is interesting to carry out a new study for training with the same number of steps as those used in the baseline.

**For SSOD:**
In the baseline training, 200k, 400k and 400k steps have been used in the cases of radius 0.15, 0.3 and 0.5, respectively.

In training with RVT (`rnndet`) and LEOD (`rnndet-soft`) 100k, 200k, 200k steps have been used, respectively. The reason is the same as that indicated for WSOD.

### Results obtained

From the results obtained and after a subsequent analysis of the precision graphs obtained during the training flow of each of them, it can be seen that for the dataset used in this project there is no mAP value that is generally later than 200k steps. Reducing to around 150k for radius values of 1% and 2%. However, as can be seen in Figure B.1, the accuracy of the model training fluctuates considerably, so it does not mean that higher values cannot be achieved in some cases as the training progresses.




WSOD 5% training

SSOD 50% training

Figure B.1: AP evolution by number of steps. Units of X axis: *Steps x 100*

A more exhaustive study of the number of steps used and the different pseudolabel training methods could reveal a clearer pattern in the evolution of accuracy versus model training time.

# C Evaluation of Pseudolabels

## Weakly-Supervised Object Detection

Table C.1: Pseudolabels Validation - WSOD 0.01

| Metric | Value |
|---|---|
| ssod/teacher_AP@25_ziv | 0.8463 |
| ssod/teacher_AP@50_ziv | 0.8385 |
| ssod/teacher_AP@75_ziv | 0.8169 |
| ssod/gt_num_ziv | 1.0232 |
| ssod/pred_num_ziv | 0.8977 |
| ssod/teacher_AP@25_hum | 0.9130 |
| ssod/teacher_AP@50_hum | 0.8808 |
| ssod/teacher_AP@75_hum | 0.8491 |
| ssod/gt_num_hum | 1.0204 |
| ssod/pred_num_hum | 0.9975 |

Table C.2: Pseudolabels Validation - WSOD 0.02

| Metric | Value |
|---|---|
| ssod/teacher_AP@25_ziv | 0.8680 |
| ssod/teacher_AP@50_ziv | 0.8609 |
| ssod/teacher_AP@75_ziv | 0.8431 |
| ssod/gt_num_ziv | 1.0231 |
| ssod/pred_num_ziv | 0.8982 |
| ssod/teacher_AP@25_hum | 0.9288 |
| ssod/teacher_AP@50_hum | 0.9197 |
| ssod/teacher_AP@75_hum | 0.8956 |
| ssod/gt_num_hum | 1.0203 |
| ssod/pred_num_hum | 1.0111 |

Table C.3: Pseudolabels Validation - WSOD 0.05

| Metric | Value |
|---|---|
| ssod/teacher_AP@25_ziv | 0.9529 |
| ssod/teacher_AP@50_ziv | 0.9503 |
| ssod/teacher_AP@75_ziv | 0.9346 |
| ssod/gt_num_ziv | 1.0235 |
| ssod/pred_num_ziv | 0.9827 |
| ssod/teacher_AP@25_hum | 0.9626 |
| ssod/teacher_AP@50_hum | 0.9602 |
| ssod/teacher_AP@75_hum | 0.9387 |
| ssod/gt_num_hum | 1.0208 |
| ssod/pred_num_hum | 0.9955 |

Table C.4: Pseudolabels Validation - WSOD 0.1

| Metric | Value |
|---|---|
| ssod/teacher_AP@25_ziv | 0.9784 |
| ssod/teacher_AP@50_ziv | 0.9706 |
| ssod/teacher_AP@75_ziv | 0.9556 |
| ssod/gt_num_ziv | 1.0240 |
| ssod/pred_num_ziv | 1.0018 |
| ssod/teacher_AP@25_hum | 0.9801 |
| ssod/teacher_AP@50_hum | 0.9757 |
| ssod/teacher_AP@75_hum | 0.9608 |
| ssod/gt_num_hum | 1.0211 |
| ssod/pred_num_hum | 1.0115 |

### Semi-Supervised Object Detection

Table C.5: Pseudolabels Validation - SSOD 0.15

| Metric | Value |
|---|---|
| ssod/teacher_AP@25_ziv | 0.7199 |
| ssod/teacher_AP@50_ziv | 0.7024 |
| ssod/teacher_AP@75_ziv | 0.4295 |
| ssod/gt_num_ziv | 1.0179 |
| ssod/pred_num_ziv | 0.7432 |
| ssod/teacher_AP@25_hum | 0.8839 |
| ssod/teacher_AP@50_hum | 0.8008 |
| ssod/teacher_AP@75_hum | 0.8359 |
| ssod/gt_num_hum | 1.0121 |
| ssod/pred_num_hum | 0.9661 |

Table C.6: Pseudolabels Validation - SSOD 0.3

| Metric | Value |
|---|---|
| ssod/teacher_AP@25_ziv | 0.8542 |
| ssod/teacher_AP@50_ziv | 0.8536 |
| ssod/teacher_AP@75_ziv | 0.8408 |
| ssod/gt_num_ziv | 1.0168 |
| ssod/pred_num_ziv | 0.8776 |
| ssod/teacher_AP@25_hum | 0.8829 |
| ssod/teacher_AP@50_hum | 0.8826 |
| ssod/teacher_AP@75_hum | 0.8407 |
| ssod/gt_num_hum | 1.0099 |
| ssod/pred_num_hum | 0.9575 |

Table C.7: Pseudolabels Validation - SSOD 0.5

| Metric | Value |
|---|---|
| ssod/teacher_AP@25_ziv | 0.9286 |
| ssod/teacher_AP@50_ziv | 0.9276 |
| ssod/teacher_AP@75_ziv | 0.9107 |
| ssod/gt_num_ziv | 1.0250 |
| ssod/pred_num_ziv | 0.9578 |
| ssod/teacher_AP@25_hum | 0.9567 |
| ssod/teacher_AP@50_hum | 0.9342 |
| ssod/teacher_AP@75_hum | 0.9021 |
| ssod/gt_num_hum | 1.0171 |
| ssod/pred_num_hum | 1.0581 |

### Analysis

It can be clearly seen that, as the amount of data used in the baseline training increases, the precision of the generated pseudo-labels improves. It is particularly relevant to highlight that a maximum precision of 0.97 is reached for 50% of the labels in both the human prediction and the zivid prediction, using only 10% of the labels (WSOD 0.1).

Furthermore, it can be seen that the relationship between the amount of data and the accuracy achieved is not linear, as anticipated. Specifically, the increase in accuracy between the WSOD 0.02 and WSOD 0.05 models is not proportional to the increase between WSOD 0.05 and WSOD 0.1, despite the fact that the percentage increase in data is greater in the second case.

On the other hand, when the number of labels used is increased, it can be seen that the SSOD model generates less accurate pseudo-labels in comparison with WSOD. This is because SSOD only learns from complete scenes, allowing it to capture the complete flow of elements per scene, but limiting its capacity for generalization due to the smaller number of cases it analyzes. In contrast, WSOD learns from a greater number of scenes, even if it does not retain the complete flow of elements per scene, which favors a greater capacity for generalization and learning over a wider range of cases.

# D Timeline and Budget

A breakdown of the time and resources used in this project is provided to facilitate its future reproduction.

## D.1 Planning

This project began in mid-October 2024 and was completed in March 2025, lasting approximately 5 months. During this period, the dedication was 20 hours per week, which multiplied by the 20 weeks of duration gives a total of approximately 400 hours.

The development took place within the research group *FLW (The Chair of Material Handling and Warehousing)* of the the *Technical University of Dortmund (TUD)*.

The WBS not only facilitates project planning and execution but also serves as a base tool for organizing the workflow, assigning responsibilities, and monitoring progress. The following diagram in Figure D.2, presents the Work Breakdown Structure (WBS) designed for this thesis, structured around four main areas: foundational research, data preprocessing, algorithm adaptation, and dataset creation.

As shown in WBS diagram, the project has four main groups. This groups are represented in the following Gantt Chart (Figure D.1) where each row represents a logical phase of work: dark bars mark each major block, while light bars are the sub-phases into which every block is split. In the next diagram you can see what was done in each phase and how much weight it carried in the overall timeline.

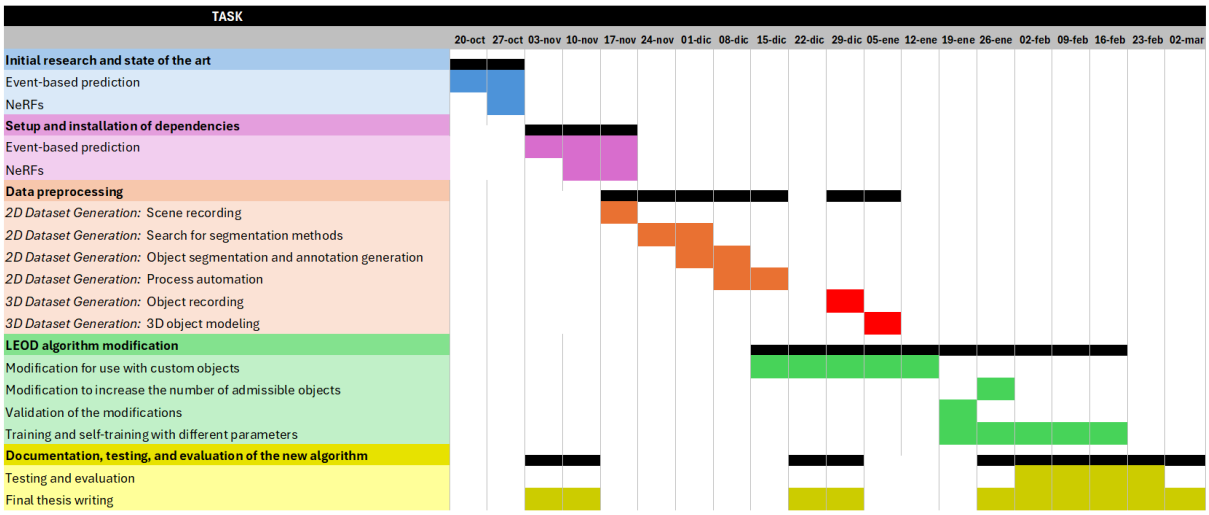



Figure D.1: Gantt chart

**Initial Research and State of the Art:** This was a short phase aimed at starting the project with solid technical context. We reviewed relevant literature on event-based vision (RVT and LEOD), as well as the latest developments in Neural Radiance Fields (NeRF).

**Setup and Installation of Dependencies:** The focus here was on configuring the working machine, aligning CUDA / PyTorch versions, container environments (for ROS 2 Noetic), and NeRF-related libraries. This phase took longer than expected due to hardware incompatibilities

(limited disk and GPU memory), requiring multiple setup attempts across several devices until a functional configuration was achieved.

**Data Preprocessing:** This was the longest phase of the project.

- New logistics scenes were recorded using both RGB and event-based cameras.
- An automated pipeline for extraction, segmentation, and compression was developed, reducing event histogram sizes by approximately 90%.
- In parallel, a 3D dataset was built using NeRF to generate detailed meshes of key objects.

The outcome was a coherent and well-annotated repository that served as the foundation for training.

**LEOD Algorithm Modification:** This phase involved the most execution time (though not the most active working time), as every modification had to be validated through simulations with associated time costs. The LEOD algorithm was adapted to accommodate the new classes in the "FLW dataset", and confidence thresholds were adjusted accordingly.

**Documentation, Testing and Evaluation:** Although not as lengthy as the previous phase, this stage continued steadily until the project's conclusion. Precision metrics were measured, qualitative analyses were performed, and conclusions were drawn—sometimes leading to model adjustments that enriched the study. Simultaneously, the project report was drafted, and explanatory sections were written to clarify all findings.

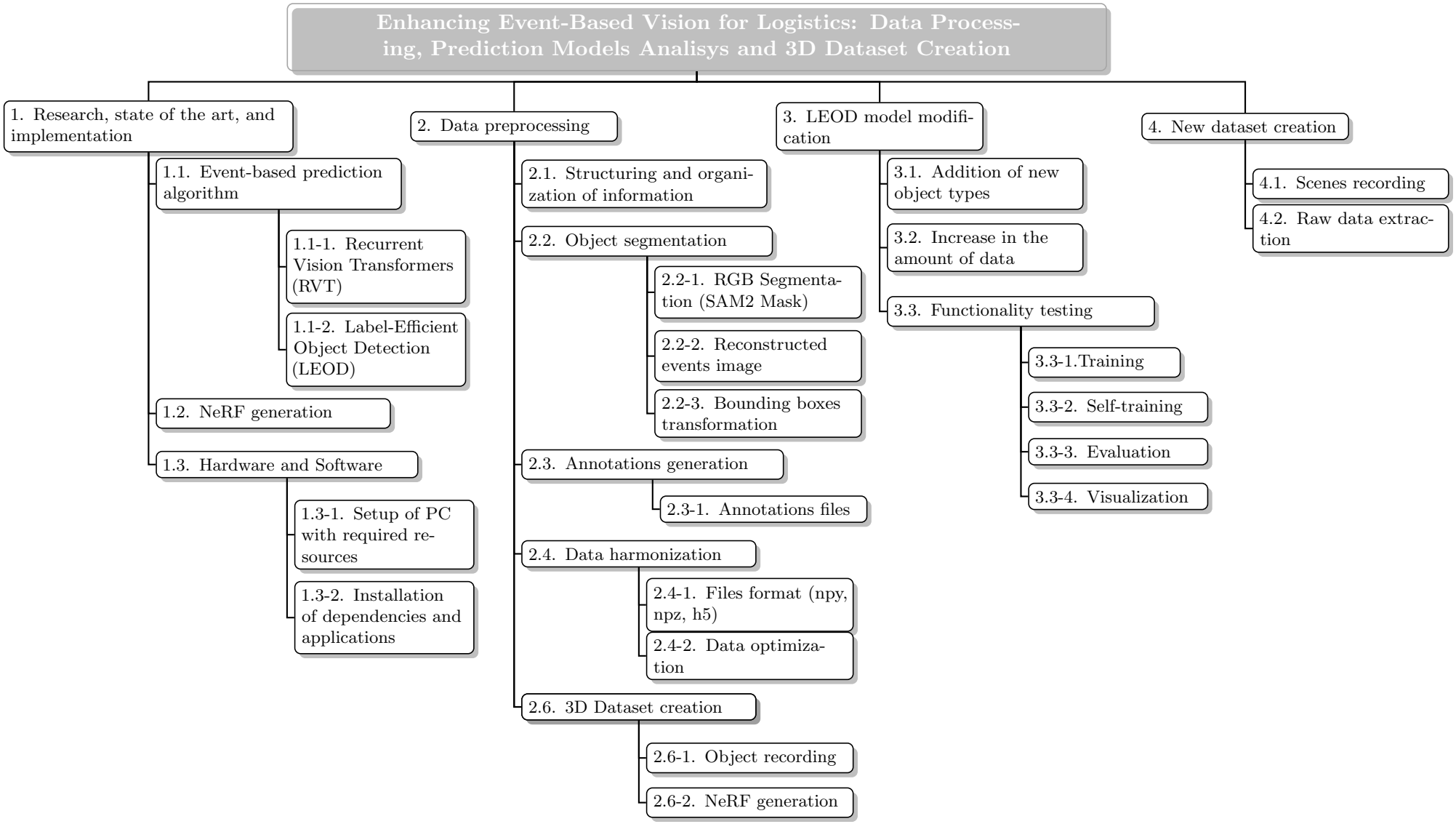



Figure D.2: Work Breakdown Structure (WBS)

## D.2 Budget

The budget for this project is divided into the cost of human resources and the cost of equipment used.

The 400 hours of work dedicated by the student are multiplied by the hourly wage of a full-time research assistant at the university, which is approximately 8.33 euros per hour (1250 euros monthly). According to UPM's transparency portal, the tutor's hourly wage is 33.72 euros. Table D.8 shows the total cost of human resources.

| **Role** | **Hours** | **Hourly rate (EUR)** | **Total cost (EUR)** |
|---|---|---|---|
| Student | 400 | 8.33 | 3332 |
| Supervisor | 20 | 33.72 | 671.4 |
| **Total** | | | **4006.4** |

Table D.8: Human resource costs involved in the project

Among the equipment required for this project are the drone with all its components and the laptop used during the development. Table D.9 summarizes the cost of all these items.

| **Component** | **Units** | **Unit cost (EUR)** | **Total cost (EUR)** |
|---|---|---|---|
| PC [App. A] | 1 | 3500.00 | 3500.00 |
| DVXplorer Lite | 2 | 1600.00 | 3200.00 |
| U3-3270CP-HQ Rev.2.2 | 1 | 626.00 | 626.00 |
| Bosch BT 150 tripod | 1 | 56.00 | 56.00 |
| USB 3.0 Micro-B | 3 | 21.00 | 63.00 |
| USB 3.1 Tipo-C | 3 | 26.60 | 79.80 |
| 6 pins JST-GH | 2 | 34.00 | 68.00 |
| Assembly Components | - | - | 30.00 |
| Reflective markers | - | - | 42.00 |
| **Total** | | | **7664.80** |

Table D.9: Cost of equipment used in the project

The total cost of the project is:

| **Item** | **Total cost (EUR)** |
|---|---|
| Human Resources | 4006.4 |
| Equipment and Components | 7664.80 |
| **Total** | **11671.2** |

Table D.10: Total development cost of the project

# E Social Impact, Environmental Impact and the 2030 Agenda

An analysis of the project's impact beyond purely technical aspects is carried out to ensure compliance with current regulations, the safeguarding of all rights, and its contribution to sustainable development.

## E.1 Social Impact, Ethical and Legal Aspects

The rapid deployment of eventbased vision systems in logistics environments has brought this technology within the reach of an evergrowing number of warehouse operators and visitors. It is therefore essential to analyse its social implications and the regulatory framework that governs its rollout.

The main concern centres on protecting workers' privacy and dignity. Although dynamic vision sensors (DVS) do not capture full frames, their event streams allow silhouettes and even identifiable movements to be reconstructed. To guarantee the fundamental right to privacy (*Art. 18.1 Spanish Constitution*), the company must comply with *Regulation (EU) 2016/679 (GDPR)* and its transposition into Spanish law via *Organic Law 3/2018 (LOPDGDD)*. Among other measures, this requires informing employees (*Art. 13 GDPR*), applying the principle of data minimisation, and carrying out a Data Protection Impact Assessment (DPIA) before commissioning (*Art. 35 GDPR*).

From the standpoint of corporate control, *Art. 20.3 of the Workers' Statute* empowers the employer to introduce monitoring measures provided they are proportionate and respect the worker's dignity. Constitutional Court case law (e.g., *STC 39/2016*) reinforces the need to give explicit, prior notice of the purpose of video surveillance, even when only aggregated data are used for process analytics.

In matters of occupational health and safety, the installation of cameras and servers must comply with *Law 31/1995 on the Prevention of Occupational Risks*, *Royal Decree 614/2001* on electrical hazards, and Royal Decree 1215/1997 on the use of work equipment. *UNE EN ISO 10218 2* (robots and robotic devices in industrial settings) provides guidance for defining safe zones and emergency stop plans when sensors are integrated with AGVs or collaborative robots.

The draft European Artificial Intelligence Regulation (*AI Act, COM 2021/206*) classifies workplace surveillance systems as "high risk" (*Annex III*), which entails the obligation to:

1. Set up a risk management system throughout the life cycle;
2. Ensure event logging and model traceability;
3. Provide understandable explanations to affected users;
4. Subject the system to a conformity assessment and, where applicable, register it in the European database before it is placed on the market.

Finally, the distribution of the software and the dataset generated is covered by *Patent Law 24/2015 and Directive (EU) 2019/790* on copyright in the Digital Single Market. To encourage academic reproducibility while protecting commercial exploitation, a dual licence is recommended: CC BY NC 4.0 for the scientific component and a proprietary or reasoned copyleft licence for the business modules.

In short, the introduction of asynchronous vision in logistics offers undeniable improvements in safety and productivity, but it calls for transparent, proportionate data governance and periodic audits in accordance with current legislation and the forthcoming European regulations.

### E.2 Environmental Impact

Unlike conventional RGB cameras, event based vision sensors **consume on the order of 10–20 mW compared with the usual 2–3 W**. In a typical logistics centre with 150 inspection points, complete replacement would **save roughly 1.2 MWh per year**—equivalent to the annual energy consumption of eleven average Spanish households.

This lower consumption also reduces internal thermal loads, easing the demand on air conditioning during warm months and therefore indirectly cutting the $CO_2$ emissions associated with the HVAC system (in line with *Directive (EU) 2018/2002* on energy efficiency and the objectives of *Law 7/2021* on Climate Change and Energy Transition).

Moreover, DVS sensors dispense with a mechanical shutter and use fewer moving parts, which **extends their service life** and **reduces the generation of waste** electrical and electronic equipment (WEEE) regulated by *Royal Decree 110/2015*. When their life cycle ends, their minimal size and weight simplify management as "professional waste", fulfilling the principle of producer responsibility.

In conclusion, adopting event based vision not only enhances operational efficiency and worker safety but also minimises the environmental impact of both the surveillance system and the broader logistics flow, consolidating a more responsible and competitive model.

### E.3 Sustainable Development and the 2030 Agenda

The United Nations 2030 Agenda sets out 17 Sustainable Development Goals (SDGs) that guide any project committed to responsible progress. This master's thesis on event based vision for logistics has a particular impact on SDGs 7, 13 and 17.

First, ***SDG 7 -Affordable and Clean Energy-*** is strengthened because DVS sensors consume only a few milliwatts compared with the watts drawn by industrial cameras. This leap in efficiency—about 1.2 MWh less per year in an average warehouse—reduces electricity demand and cooling requirements, lowering the energy cost per handled package and facilitating the integration of the facility into renewable self consumption schemes.

***SDG 13 -Climate Action-*** is addressed indirectly but tangibly: the lower electricity consumption reduces Scope 2 emissions, while real time inventory information enables shorter delivery routes and avoids unnecessary trips, cutting Scope 3 $CO_2$. In addition, the energy data generated by the platform itself can be integrated into ISO 50001 management plans, fostering measurable decarbonisation strategies.

Finally, ***SDG 17 -Partnerships for the Goals-*** is realised through the open publication of the dataset and base code. By offering resources that any university or start up can reuse, the project drives scientific collaboration, North–South shared innovation and technology transfer, broadening social impact and accelerating the adoption of sustainable logistics in diverse contexts.

## F Source Code Repository

All the source code developed as part of this Master's Thesis is publicly available in the following GitHub repository:

**GitHub Repository – LEOD_FLW**
https://github.com/tsvillaluenga/LEOD_FLW.git

This repository contains the full implementation of the algorithms, preprocessing tools, and experimental workflows discussed throughout the thesis. In particular, the docs/ directory provides detailed documentation and step-by-step instructions on how to compile, configure, and use the project. This includes information on required dependencies, dataset setup, and execution commands to ensure full reproducibility of the results.

The repository is intended as both a technical reference and a practical resource for researchers and practitioners interested in event-based object detection and the LEOD framework.