Universidad Miguel Hernández de Elche

MÁSTER UNIVERSITARIO EN ROBÓTICA

Event-based Detector of ArUco Markers

Trabajo de Fin de Máster

2022/2023

Autor: Fco Javier Luna Santa-María
Tutores: Oscar Reinoso García,
José Ramiro Martínez de Dios

A mi familia por el apoyo incondicional.

A los profesores del Máster por su labor,
particularmente a Oscar Reinoso por la atención prestada.

A Ramiro por su confianza y dedicación,
y a Raúl por sus constantes enseñanzas.

Agradecimientos

Este proyecto ha sido realizado junto con el Grupo de Robótica, Visión y Control de la Universidad de Sevilla. Agradecer también a la fundación valgrAI por su ayuda para mis estudios del Máster Universitario en Robótica.

Resumen

Esta proyecto presenta un nuevo enfoque para detectar marcadores ArUco utilizando cámaras de eventos. Las cámaras de eventos son sensores bioinspirados que capturan cambios de brillo a nivel de píxel de forma asíncrona, ofreciendo ventajas como alta resolución temporal, baja latencia y alto rango dinámico. Los marcadores fiduciales como ArUco son marcas de referencia visuales que se utilizan en tareas como localización mediante cámaras y realidad aumentada. Sin embargo, la detección de estos marcadores se basa en algoritmos tradicionales de visión por computador diseñados para cámaras estándar.

El detector de ArUco basado en eventos propuesto consiste en un proceso para detectar e identificar marcadores procesando los eventos de entrada. En primer lugar, se generan imágenes de bordes compensadas a partir del movimiento de los eventos mediante un método de maximización del contraste. A continuación, se extraen candidatos de marcadores mediante la detección de cuadriláteros en las imágenes compensadas. Por último, se presentan dos técnicas para la identificación de ArUco en los candidatos: un enfoque de codificación de cuadrículas de bordes adaptado a los eventos (eAruco-Grid) y una clasificación con una red neuronal convolucional (eAruco-CNN).

El método se evalúa con distintos diccionarios de ArUco. Los resultados demuestran una detección satisfactoria y una gran exactitud de identificación en diversas condiciones, como con desenfoque de movimiento y cambios de iluminación, en las que falla la detección de ArUco tradicional.

Abstract

This work presents a novel approach for detecting ArUco markers using event cameras. Event cameras are bio-inspired sensors that capture pixel-level brightness changes asynchronously, used in advantages like high speed, low latency, and high dynamic range. Fiducial markers like ArUco are visual landmarks enabling tasks such as camera localization and augmented reality. However, marker detection relies on traditional computer vision algorithms devised for standard cameras.

The proposed Event-based ArUco detector consists of a pipeline to detect and identify markers by processing input events. First, motion compensated edge images are generated from events using a contrast maximization method. Next, marker candidates are extracted by detecting quadrilaterals in the edge images. Finally, two techniques are presented for ArUco identification on the candidates - a novel grid encoding approach tailored to events (eAruco-Grid) and a convolutional neural network classifier (eAruco-CNN).

The method is evaluated on different ArUco dictionaries. Results demonstrate successful detection and high identification accuracy under various conditions like motion blur and lighting changes where traditional ArUco detection fails.

Abridged Contents

Contents

1 Introduction

In recent years, event cameras, also known as neuromorphic or dynamic vision sensors, have emerged as a revolutionary technology in the field of computer vision and robotics. These sensors depart from traditional frame-based cameras by capturing only changes in brightness at pixel-level, offering several advantages such as low latency, high dynamic range, and low power consumption. Research in the field of robotics using this type of sensor has increased due to its advantages. Among the possible applications, the detection and tracking of fiducial markers using event cameras is particularly a promising and challenging task.

Fiducial markers have been widely employed in different fields such as Augmented Reality (AR) or Robotics. This is mainly due to the ability to provide distinctive points of reference which are essential for camera pose and object localization.

In AR the identification of fiducial markers and their location allows to overlaid virtual objects markers with the real world. These types of systems with fiducial markers have been used in many areas such as education, entertainment, or industry.

One of the most common applications of fiducial markers is to help unmanned aerial systems (UAS) and other robots perform localization and mapping tasks. These tasks allow a vehicle to know its exact location, even when it does not receive GNSS signals, which is essential for high-precision missions. Other methods, such as Visual-Inertial Odometry (VIO) and Simultaneous Localization and Mapping (SLAM), can also provide accurate data, but they depend on good lighting conditions and clear visual features. Fiducial markers can benefit these methods to enhance the accuracy of position estimation in environments where these conditions are not available. Furthermore, fiducial marker technology is also widely used for UAS landing and object tracking in robotics.

ArUco markers are a type of fiducial markers particularly devised for camera pose estimation in applications like augmented reality and robot localization. The main feature of this system is the ability to create marker dictionaries with configurable sizes and bit numbers.

Most of the existing methods for fiducial marker detection are frame-based algorithms devised for standard cameras. Therefore, it is necessary to research new techniques to adapt

and design algorithms for this technology.

1.1 Objective

This master's thesis presents a novel approach for event-based detection of ArUco markers, a widely adopted type of fiducial marker. The primary objective of this research is to develop a robust method capable of detecting and identifying these markers using event cameras. Furthermore, the proposed method should be capable of performing effectively across different types of conditions, ensuring that is suitable for real applications.

1.2 Contribution

This research provides a solution for event-based ArUco marker detection, leveraging the advantages of event cameras. The method prioritizes robustness and adaptability, ensuring reliable performance in real-world conditions.

The proposed approach is based on using solely as input the generated events and adapting marker encoding to create an entire event-based vision algorithm. It bridges the gap between traditional marker detection algorithms and event cameras, enhancing the utilization of this sensor in fields such as robot navigation or AR, among others.

1.3 Structure

The documents consist of 7 chapters. Chapter 2 describes the state of art related to event cameras and fiducial markers. Chapter 3 presents the general operation of the proposed method. Chapters 4 and 5 detail the different modules of the presented pipeline. Chapter 6 presents the experiments and results obtained to evaluate the method. Finally, in Chapter 7 are presented the conclusions drawn during the development of the project, as well as possible future work.

2 State of Art

2.1 Introduction

This chapter presents the state of the art with the latest advancements in computer vision involving event cameras and fiducial markers. It begins with an introduction to event cameras, followed by an overview of different types of fiducial markers. Finally, it focuses on the ArUco marker system and details the primary existing detection methods used in both frame-based and event-based vision approaches.

2.2 Event Cameras

Event cameras are neuromorphic sensors whose working principle differs fundamentally from traditional cameras. Instead of capturing frames at a fixed rate, event cameras present a paradigm shift measuring per-pixel brightness changes asynchronously. This means event cameras do not produce entire frames like regular cameras - they only output the changes known as events. These are data-driven sensors, as their output depends on the amount of relative motion between the scene and the camera. If the scenario remains static the brightness changes can be generated only with camera motion. Their operation enables advantages including high-speed sensing, low latency, low power consumption, and high dynamic range. Despite the extensive research in this field and the commercial interest growing, new algorithms are required to unlock the full potential of this type of sensor.

2.2.1 Principle of operation

Event cameras respond to changes in brightness independently and asynchronously. Each pixel in the sensor array contains a circuit that continuously monitors changes in its logarithmic intensity. When the change exceeds a threshold $\pm C$, either positive or negative, the pixel asynchronously triggers an event $e = (x, y, t, p)$. Each event contains the location of the pixel (x, y), the timestamp t, and polarity p which represents the sign of the brightness change. Figure 2.1 represents the operation of a DVS (Dynamic Vision Sensor) pixel.

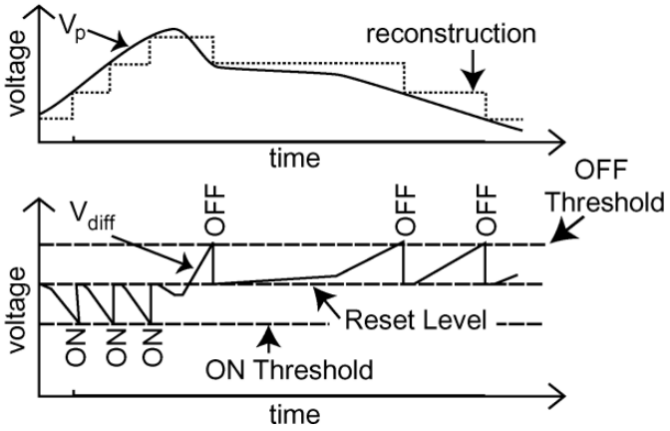

Figure 2.1 Diagram representing pixel principle of operation. Extracted from [1].

2.2.2 Advantages

Event cameras provide considerable benefits compared to traditional cameras for many computer vision applications. The following details the most remarkable advantages.

High Temporal Resolution: The continuous and fast monitoring of brightness changes by the independent pixels enables events to be detected and timestamped with microsecond resolution (1MHz clock). This allows event cameras to capture very fast motions without the motion blur usually suffered by standard cameras.

Low Latency: Since each pixel generates an event independently when a brightness change occurs, without waiting for global exposure time, events can be read out asynchronously with minimal delay. The latencies stand on the order of microseconds.

Low Power Consumption: Only asynchronous events are transmitted rather than providing complete images. This results in redundant data reduction, and consequently in very low power usage. Prototypes demonstrate less than 10mW consumption for the image sensor chip.

High Dynamic Range: The individual pixel circuits allow adaptation to extremely high dynamic range scenes, over 120dB against 60dB for standard cameras. Like biological retinas, each pixel can adjust to respond properly in very dark or very bright conditions.

2.3 Fiducial Markers

Fiducial markers are artificial landmarks with a specific pattern of known shape and size. They are used to provide reference points, a specific encoded ID or a different measurement. Depending on their application, simple patterns such as circles or rings can be used just to provide a point of reference. To encode a message or an identifier, more complex markers like well-known QR codes should be considered. Fiducial markers are employed in a wide variety of fields, ranging from medical imaging to robotics, augmented reality, and various computer vision applications.

There are various types of fiducial markers, which can be classified based on their shape, color (monochromatic or not), and type of information encoded. The majority of markers are monochromatic (particularly binary/BW), with circular or squared shapes. They use different geometric patterns to facilitate data encoding and identification procedures. The subsequent subsections will present a literature review of the most popular types of fiducial markers, which are grouped into three categories: squared, circular, and other shapes.

2.3.1 Square-based fiducial markers

These types of markers are very popular due to two main characteristics: their ability to provide four distinctive points, which are essential for pose estimation, and an inner identification region. The methods employed in the identification procedure are mainly based on binary encoding or template matching using arbitrary patterns.

ARToolKit [2] is a highly popular system in the arbitrary pattern category. These markers consist of a square black border surrounding an inner image, which can be any type of shape or pattern, which is identified using image correlation. These are used as IDs and, are stored in a database containing all reference patterns. Two examples of this type of marker can be seen in Figure 2.2. Despite they can be easily distinguishable by humans, this template matching approach with a non-defined structure of markers provides high false positive and inter-marker confusion rates. Square detection is conducted using a fixed global thresholding method, which is a drawback as it may not be robust enough to adapt to changes in lighting conditions.

Figure 2.2 Example of two ARToolKit markers with different patterns. Extracted from [2].

Regarding fiducial markers systems based on binary encoding, *Matrix* [3] was one of the first proposals. IT is based on the division of the the inner region into white and black square cells, which are converted into a binary code with redundant bits for error detection. Connected component analysis is used to select marker candidates once the image is binarized, then each cell bit is determined based on the amount of black and white pixels.

ARTag [4] stands as one of the most extensively utilized software packages. While it is based in ARToolkit, ARTag adopts the approach of binary encoding to construct the inner pattern of markers, see Figure 2.3. It involves designing the interior of the marker as 6x6 bit matrix, assigning a unique 36-bit code as the marker's identifier. This strategy ensures the distinctiveness of each marker's code. One limitation to note is that the marker dictionary is fixed to 36 bits, and only two erroneous bits can be corrected regardless of the distance between the markers used in any subset. In order to enhance the robustness of tag detection, the marker library is created to maintain a minimum Hamming distance between different codes. Instead of using fixed thresholds as ARToolKit does, an edge-based square detection method is employed to improve robustness to lighting conditions and partial occlusions. The method leverages a gradient-based technique to identify lines, which are subsequently organized into quadrilaterals representing potential markers.

Figure 2.3 Example of ARTag marker. Extracted from [4].

AprilTag [5] method introduces several enhancements upon the ARTag framework. It utilizes a graph-based image segmentation algorithm to analyze gradient patterns and accurately estimate lines. The quad extraction method allows the identification of edges that may not precisely intersect. It incorporates a new specific coding system to mitigate problems of the 2D barcode system like rotation incompatibility and false positives in outdoor environments. These improvements reduced the number of misdetections and enhanced robustness against occlusions and warping.

ARToolKit Plus [6] is an improved version of ARToolKit. The global threshold used for candidate detection is updated considering values from previous markers detected. Moreover, the inner matrix is designed for the employment of binary codes, which include error correction methods. The markers are coded using 36 bits, being two the minimum hamming distance.

CALTag [7] is another square and monochromatic marker that consists of black and

white squares, which have inside more individual identification bits. It is developed for camera calibration tasks due to the advantages that this amount of elements provides to obtain higher precision. An example is shown in Figure 2.4.

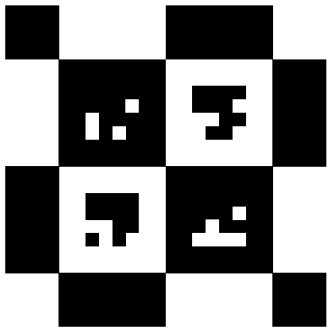

Figure 2.4 Example of a CALTag pattern. Extracted from [7].

ArUco [8] markers introduced a great contribution by allowing users to create custom dictionaries. The number of markers can be chosen depending on the application requirements instead of using all possible markers in a fixed dictionary. Therefore, the distance between the particular number of markers selected will be the highest possible. This not only leads to improving the inter-marker separation for identification, but also decreases the computing cost since a smaller dictionary is being used. Moreover, it includes a method for error detection and correction of a greater number of erroneous bits compared to others. Finally, a technique using color-coded markers and image segmentation is introduced to precisely handle occlusion in augmented reality applications.

Figure 2.5 Example of ArUco markers generated with different grid sizes. Extracted from [8].

2.3.2 Circular fiducial markers

Contrasting Concentric Circle (CCC) [9] is one of the most popular and the first designed in this category. It is a simple pattern with no encoding, which consists of a black ring placed over a white background.

TRIP [10] is another circular monochromatic marker. The centre of the marker displays a bull's eye, with two concentric rings split into 16 sectors that encode the ID. With this approach, the message is encoded using ternary numbers in the range of $(1, 3^9 - 1)$. The first sector or synchronisation sector indicates where the code begins and defines one of the axes of the marker, which is read in an anti-clockwise fashion. The next two sections are designated for performing a parity check on the extracted identifier. The four sections that follow encode the central bull's eye radius in millimetres. The last nine sections encode a ternary identifier, as shown in Figure 2.6.

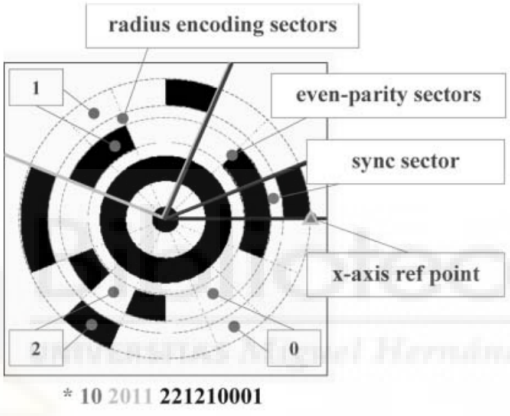

Figure 2.6 TRIP tag. Extracted from [10].

Another similar example is the *InterSense* [11] marker, which was designed to be robust to various real-world lighting in an AR application for visual-inertial tracking. It is structured with a black outer ring, a 2D barcoded interior based on two data rings, and an inner black ring. The inner ring always contains a white eye in the centre of the marker. The structure of both data rings consists of eight equally sized sectors. Three of them remain constant to define the orientation of the marker, and the rest of the sectors with three data cells each, are used to encode the markers. This design allows to encode 2^{15} different markers. An example of this marker can be seen in Figure 2.7.

RuneTag [12] (Figure 2.8) is a circular marker with one or three concentric rings divided into equal sectors. The rings contain black dots, with outer dots being the largest and inner dots being the smallest. These dot patterns define the tag's IDs. Its design provides high precision for camera calibration and object measurement tasks, being also robust against partial occlusions. Nonetheless, due to the rapid detection required in robotics applications, it is not suitable for such tasks. High robustness is prioritized against speed in marker detection.

Figure 2.7 InterSense tag. Extracted from [11].

Figure 2.8 Example of RuneTag marker, RUNE-129. Extracted from [12].

CCTag [13] is a round marker with black and white concentric rings, see Figure 2.9. It is specifically designed to detect and identify features in blurry images, particularly in cases of unidirectional motion blur. Unlike other markers, CCTag does not have an ID encoding. It outperforms RuneTag in terms of detection rates, especially at a distance, even when there are occlusions or blur.

Figure 2.9 CCTag marker example. Extracted from [13].

The FourierTag [14] employs a unique approach by representing binary data as a radially symmetric grayscale structure, see Figure 2.10, departing from traditional topological methods. This method is named after its utilization of the Fourier transform's amplitude spectrum to explicitly encode a bit pattern. In contrast, markers like ARTag and similar fiducial systems rely on geometric patterns to encode data, rendering them susceptible

to degradation in viewing conditions such as distance or sensor noise. This degradation can lead to pattern ambiguity and the inability to extract information, even when the landmark remains observable. Therefore, the Fourier tag was designed to tackle the issue of insufficient resolution, as well as rapid detection and decoding. When the Fourier tag is observed with less accuracy, the encoded information degrades smoothly. Thus, if only partially recognized, the high-order bits representing the pattern's identity are retained, while the low-order bits progressively deteriorate. This approach is achieved by encoding information in the amplitude spectrum of the frequency domain, where lower-order bits correspond to higher frequencies. As the tag's image loses resolution with distance due to the imaging process behaving like a low-pass filter, the lower-order bits are the ones that are primarily lost. Its main advantages are the slow degradation of accuracy, the improvement of the detection range and, the optimization of data extraction under adverse images.

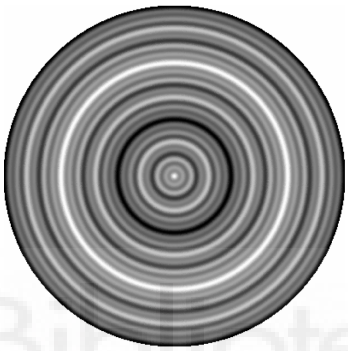

Figure 2.10 Fourier tag. Extracted from [14] .

2.3.3 Other fiducial markers

Topological fiducial markers, as previously presented, commonly maintain a consistent shape, such as square or circular. However, certain marker systems, like *TopoTag* [15], allow for adaptable shapes for both internal and external components. This flexibility in shape choice serves to reduce uncertainties related to marker rotation during decoding. Additionally, it includes a unique baseline node, designed to stand apart from other nodes, which aids in identifying the surrounding normal nodes. Different examples of TopoTag can be seen in Figure 2.11.

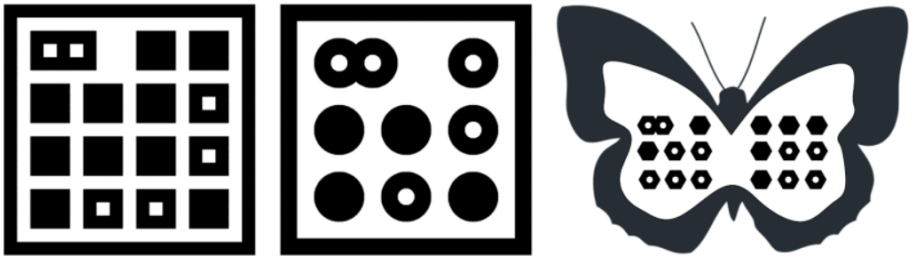

Figure 2.11 Examples of TopoTag markers. Extracted from [15] .

Pi-Tag [16] is a fiducial marker designed with a square border and circular internal bits for identification, see Figure 2.12. Its innovation consists of simplifying the usual two-step process of initial homography estimation to remove the perspective projection and identification by combining both steps directly in the image space. The method utilizes collinearity and cross-ratios, resulting in a quick, precise, and robust algorithm for applications like camera calibration and augmented reality.

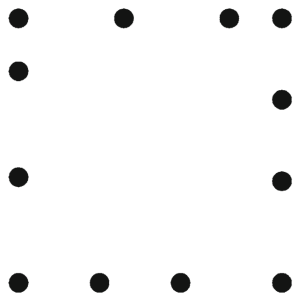

Figure 2.12 Pi-Tag. Extracted from [16] .

2.4 ArUco Markers Detection

This section provides an extended explanation of ArUco markers along with the process of detecting them. Additionally, it will cover the few methods existing in event-based vision for utilizing these markers.

ArUco markers were proposed in [8] as a fiducial marker system to address mainly the problem of predefined dictionaries of markers. Depending on the number of markers required by the applications, more markers should be available if needed, or smaller dictionaries should be used to achieve the highest possible inter-marker distances. In AR applications, occlusion is a critical problem to face when using fiducial markers. If a real object occludes a marker, where the virtual scene is overlaid, the detection procedure should not fail. Moreover, the real objects should be kept visible to enhance the interactive experience. Therefore, the presented method provides solutions for this problem. Initially, it provides an algorithm to create configurable dictionaries with the number of markers required. It maximises the inter-marker distance and the number of bit transitions. Furthermore, a method for detecting and correcting errors using the generated dictionaries is proposed. It has a higher capability of correcting erroneous bits compared to other state-of-the-art systems. To handle occlusion problems, the solution adopted is the use of multiple markers and a color map to detect the visible pixels to render the virtual scene on them with higher accuracy.

2.4.1 Frame-based ArUco detection

The proposed ArUco marker detector identifies rectangles in the images, extracts their binary code, and finds a match in the dictionary. This method is composed of different stages:

1. Image segmentation: Taking as input a grayscale image, a local adaptive thresholding method is used for edge detection.
2. Contour extraction and filtering: Using the thresholded image with the edges, the contours of the image are extracted and approximated to polygons. Considering the rectangular shape of the markers, the external contour polygons with 4 vertexes are selected.
3. Code extraction: Once the rectangular candidates are identified, the code is extracted by analyzing the inner region. For each candidate, the homography matrix is computed to remove perspective projection and the image is thresholded. This image is divided into a grid, and depending on the pixel values of each cell, a value of 0 or 1 is assigned.
4. Identification and error correction: For each marker candidate, there are four different codes, one for each rotation. These codes are looked up in the dictionary, and if any of them is found, it is considered a valid marker. A correction method is applied if no match is found. Defining the minimum distance between any two markers as τ, it is possible to detect and correct errors of up to N bits. The method consists in computing the distance of the incorrect candidate to all the markers in the dictionary. If any of the distances is equal to or smaller than N, the nearest one is considered the

valid marker ID.

Another method for efficient detection of squared fiducial markers in video sequences is proposed in [17]. It is based on a multi-scale strategy which selects the most appropriate scale for detection, identification and corner estimation. The key enhancements compared to other techniques like ArUco aim to improve processing speed, especially for high-resolution images, while maintaining accuracy and robustness. The pipeline operates on a resized version of the input image, determined by a minimum expected marker size τ_i and a canonical marker image size τ_c. Performing initial segmentation and contour extraction on the resized image provides significant speedup. A global thresholding method handles illumination changes based on the previous frame. The contour candidates are filtered based on perimeter and shape constraints. A multi-scale image pyramid enables constant-time canonical image extraction by selecting the optimal scale where the marker contour length is most similar to the canonical image length. This avoids costly warping and interpolation. To localize accurately the corners of the detected markers in the original image, the corners are upsampled from the resized image to the original. Using the images of the pyramid the corners are incrementally upsampled until the original image is reached. The minimum marker size τ_i is estimated automatically for each frame based on the smallest detected marker in the previous frame, adapting the approach for maximum performance. The pipeline dynamically optimizes the processing steps for the specific analyzed frame.

The paper presented in [18] focuses on improving the identification of square fiducial markers under difficult real-world conditions like motion blur, variable lighting, small scale, and defocus. The fiducial marker identification problem is reformulated as a classification task based on popular machine learning techniques, rather than relying on traditional methods based on image processing and bit pattern detection. An example of an ArUco marker in an image suffering from motion blur is shown in Figure 2.13.

Figure 2.13 Example of a blurred ArUco marker using a standard camera.

Three types of classifiers are evaluated on this task: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Support Vector Machine (SVM). The key idea is that these classifiers can be trained on synthetically generated images that apply transformations to emulate real conditions like blurring, lighting changes, noise, etc. For training data, the authors utilize both synthetically transformed marker images as well as real

background images containing no markers. The resulting classifiers are compared against the current state of the art ArUco and AprilTags methods on real test videos containing challenging situations including motion blur, defocus, overexposure, distance, and lighting variation. The results demonstrate that all three classifiers significantly outperform ArUco and AprilTags in identification accuracy on the real test sequences, especially for small marker sizes. The differences in performance between the MLP, CNN, and SVM classifiers are minor. In contrast, the ArUco and AprilTags approaches, while having high precision, suffer from very low recall rates. Regarding computing cost, the proposed classifiers are valid for real-time performance despite being slower than the traditional techniques evaluated.

2.4.2 Event-based ArUco detection

At present, there are very few event-based methods to detect and decode fiducial markers. In the following, the existing attempts on the literature are reviewed.

The method proposed in [19] presents a novel method for detecting and decoding planar fiducial markers using an event camera. The approach first detects line segments representing marker edges in images created from event packets. It matches "on" and "off" event line segments to form marker candidates. Candidates are unwarped via perspective transformation to a standard square shape. The black and white grid cell values are decoded by convolving Gaussians at expected event locations along cell borders for "on" and "off" event images. The binary code is extracted from the reconstructed marker and validated in a dictionary, checking the four possible rotations. For ArUco markers, this involves concatenating cell code rows and looking up the binary code in the dictionary. Valid codes yield the marker ID, otherwise, the candidate is rejected. Experiments demonstrate successful decoding where traditional methods fail due to motion blur, even with a high frame rate camera. Key limitations are the requirements for movement parallel to marker edges, without rotation, zooming, or diagonal motions, which lead to errors in marker decoding.

The work presented in [20] introduces an approach for event-based detection of ArUco markers. Due to event camera operation, conventional ArUco markers cannot be directly captured. To overcome this limitation, an ArUco marker is simulated using an LED dot matrix, combining the benefits of the ArUco pattern with the event camera's sensitivity to brightness changes. The emulated Aruco marker is captured by the event camera due to the events triggered by adjusting the LED dot matrix brightness. Besides the generation of the LED marker, the detection method proposed applies mean shift filtering, corner point refinement, error detection, and weighted statistics. An example of an event image containing an LED marker captured is shown in Figure 2.14.

Mean shift filtering algorithm is proposed for noise removal. It provides improved noise robustness in dynamic environments by iteratively shifting the kernel center point toward the direction of maximum increase in local density. By tracing the local density increases, mean shift retains significant edges while filtering noise. The initial corner points detected undergo refinement to improve overall robustness. The three most likely actual corner points are filtered based on the cosine of the angles formed between point pairs.

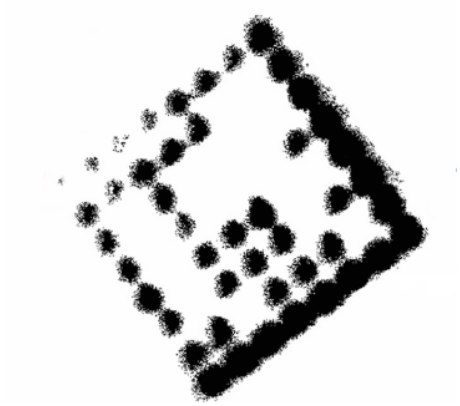

Figure 2.14 LED marker captured with an event camera. Extracted from [20].

The position of the fourth point is then estimated from these filtered points, reducing the impact of any single inaccurate point. Error detection is incorporated by leveraging the high Hamming distance between ArUco marker codes. The encoded marker is matched against the dictionary codes, selecting the code with less Hamming distance. A successful detection occurs only if this distance falls within a set accepted range. Weighted statistics are applied in the encoding grid regions. Different weights are assigned with higher values near the center and lower weights at the periphery. This minimizes inter-pixel effects arising from motion-induced interference.

In [21], a method for ChAruco board detection based on image reconstruction with a deep learning approach to estimate 6DoF pose of an event camera is proposed. A convolutional recurrent neural network, E2VID [22], is used to reconstruct the grayscale images from the event streams. The results obtained with this method are considerable, but usually contain a lot of noise and tend to have low dynamic ranges. Thus, the reconstructed images are pre-processed to improve their quality. To reduce noise smoothing the image and preserving edges, a bilateral filter is applied. Moreover, a contrast adjustment method is used to increase the brightness. Despite being an approach in which frame-based algorithms can be employed, it is limited by the failures in image reconstruction. The C-RNN does not always have the capability to reconstruct images with the required quality, due to noisy events hampering reconstruction, bad results which affect the later frames, or failures in initial frames reconstruction. Figure 2.15 shows two examples of images reconstructed from events. The image on the left is an accurate reconstruction, whereas the image on the right is a deficient result.

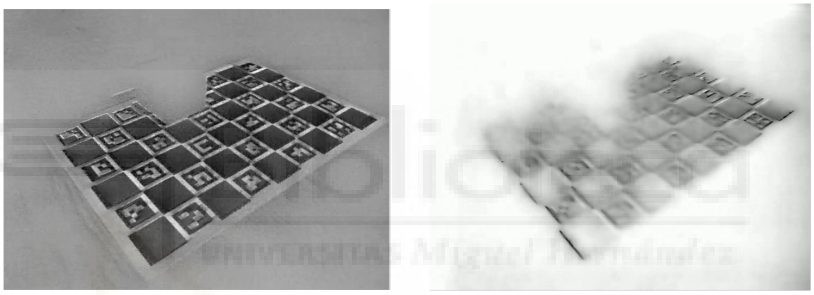

Figure 2.15 Examples of accurate and bad results of image reconstruction. Extracted from [21].

3 Event-based ArUco Detector

3.1 Introduction

The method proposed aims to be a solution for ArUco markers detection using event cameras. This chapter presents a general description of the presented Event-based ArUco detector, with the operation scheme of the method, see Figure 3.1, and an introduction to each module, which will be further explained in the following chapters.

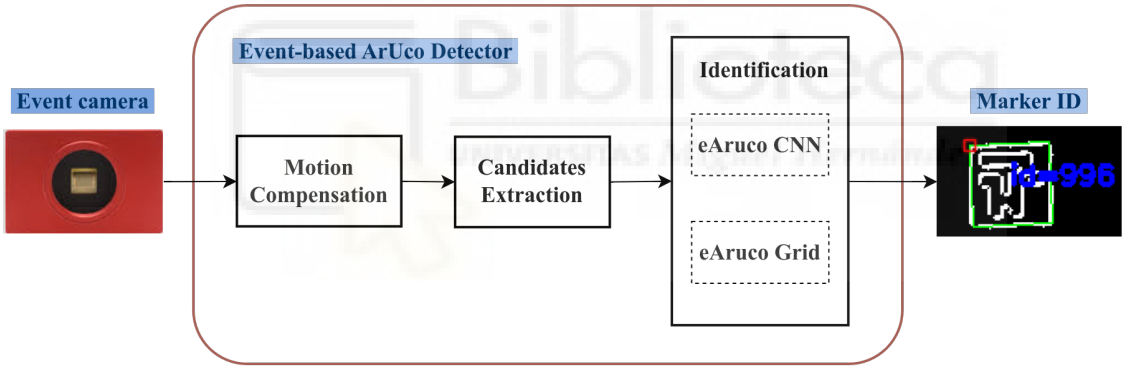

Figure 3.1 Scheme of the Event-based ArUco Detector.

The first module is responsible for generating motion compensated images from the input events. Afterwards, the candidate extraction module identifies in the compensated images those edges that could correspond to markers. Finally, the last module performs the identification of the valid candidates obtaining the IDs. In the identification module, two different methods are proposed: a Deep Learning based method using a CNN, and another based on binary encoding to describe markers.

3.2 Motion Compensated Images

The raw events reported by the event camera should be converted into a representation that allows for the markers identification. In this case, a method based on operating with groups of events and exploiting their spatio-temporal information is used to create images

of motion-corrected edges.

In [23] a unifying contrast maximization framework to estimate motion, depth and optical flow is proposed. Moreover, it is able to produce motion compensated images of the edges. The method is based on establishing correspondences between events to extract information. Particularly, since event cameras respond to the apparent motion of edges, it is focused on determining which events were triggered by the same edges. It proposes to find the point trajectories on the image plane that best fit the generated events by maximizing the contrast of an Image of Warped Events (IWE). Figure 3.2 shows an example of a motion compensated image obtained for event-based ArUco detection.

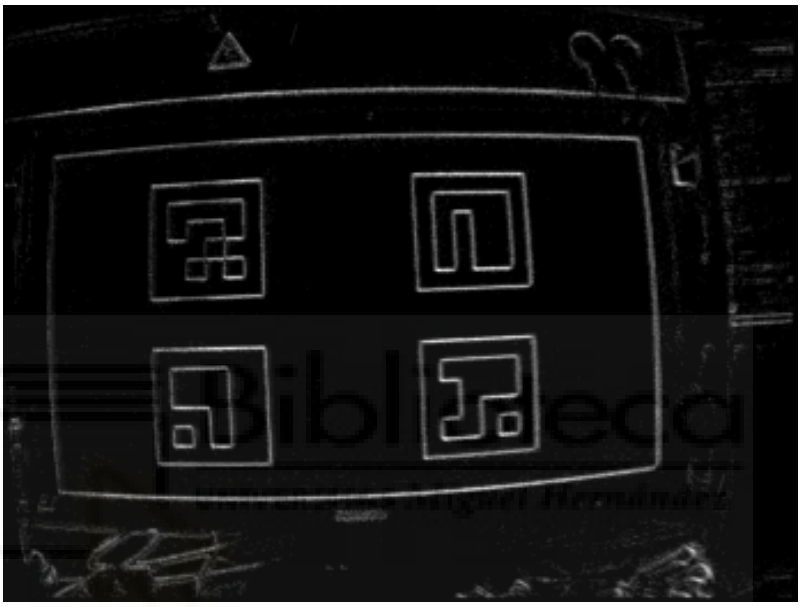

Figure 3.2 Example of a resulting image of warped events (i.e. motion compensated image).

The images shown in this document are not undistorted because it is intended to design a general method that is capable of working with non-calibrated cameras or cameras with unknown calibration.

3.3 Candidates Extraction

Markers candidate extraction is an essential step for the subsequent identification. The input of this module is the motion compensated image, and the objective is to find quadrangular shapes almost similar to a square.

If the relative motion between the camera and markers is parallel to marker edges, events will be generated mostly on the two opposite edges. Hence, in the compensated images the quadrangular shape containing the marker will be defined just with a couple of parallel edges. Due to that fact, candidates can not be identified by directly finding contours that

can be approximated by the desired shape.

The approach adopted is based on finding segments in the images and checking with geometric constraints whether pairs of segments can correspond to the sides of a square or not. Once all possible candidates are found, a bounding box fusion is performed, if necessary. This avoids having multiple candidates for a single marker and also helps in refining the estimation. For instance, in case the quadrangular shape is fully defined in the image with 4 sides, there can exist different candidates for the same marker, as two different pairs of segments can fulfil the constraints. Figure 3.3 shows examples of candidates extracted in the image.

3.4 Event-based ArUco Identification

The identification of the ArUco markers candidates IDs is the most relevant module of the detector. The input of this module consists of the marker candidates found. The output is the ID obtained for each candidate. As previously explained, due to the generation of the events, the candidates can have all the edges from the ArUcos cells, only the vertical ones or only the horizontal ones. This has also been considered when designing the identification method. Therefore, the two proposed methods are able to work correctly under any of these three cases.

The first proposed method, **eAruco-Grid**, is based on analyzing the candidate images using a grid with the borders of the marker's cells. Each of these borders in the grid is encoded with a bit. Thus, by concatenating all these bits, a binary code is obtained when the grid checking is completed. Next, the binary code describing the candidate is looked up in a dictionary. The ID and the hamming distance to the nearest reference code in the dictionary are provided. In case, the code is found in the dictionary this hamming distance is zero.

Specific dictionaries for this new type of encoding are required. These dictionaries encoding the borders are generated using original synthetic ArUco markers. For that purpose, each marker image in the original dictionaries is processed by first applying edges detection. Then, the corresponding binary code is extracted as explained above with the grid of borders.

The other approach presented, **eAruco-CNN**, is based on Deep Learning. It relies on a Convolutional Neural Network to perform the identification step with the candidate images. This method requires to train the CNN for each ArUco dictionary. However, this procedure is simple, it just has to be done once for two reasons. First, the marker candidate detection is completed in the previous step. Hence, the images that will be evaluated are canonical images, where the perspective projection has already been removed. Second and most important, the training and validation step of each dictionary model is performed using synthetic data. The original ArUco markers images from the dictionary are converted into edge images, and the models are trained solely relying on these images and data augmentation techniques. Finally, the evaluation of the model is performed using real data of canonical images with marker candidates extracted from motion compensated images.

An example of ArUco markers candidates extracted and identified is shown in Figure 3.3.

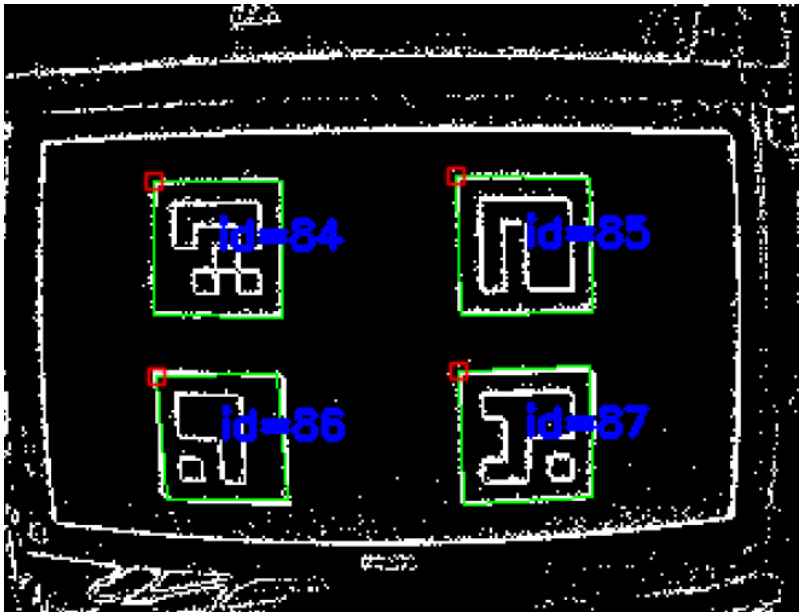

Figure 3.3 Example of Event-based ArUco marker detection results after candidates extraction and identification.

4 Event-based Candidates Detection

This chapter details the process of creating motion compensated images and extracting marker candidates. These are the initial steps of the Event-based ArUco Detector pipeline and are crucial for identifying candidates in the later stage.

4.1 Motion Compensated Images

Based on Contrast Maximization framework the proposed solution to represent the input events consists of creating motion compensated images by estimating the global optical flow at the image pixels with the information contained in the events.

The method consists of warping the events into the IWE given a candidate of optical flow and computing the variance of all the pixels of the IWE, which measures how well the events are aligned in the different candidate trajectories. The polarity is not used, thus, the IWE only counts the number of warped events per pixel, otherwise, polarities should be summed.

Due to the local-flow-constancy hypothesis, it can be assumed that the flow is constant in the space-time region in which events are warped. Events $e_k = (\mathbf{x}_k, t_k)$ are transformed according to a common time t_{ref} to $e'_k = (\mathbf{x}'_k, t_{ref})$, see Equation 4.1, by moving them along a straight line trajectory defined by the motion parameter $\mathbf{v}$, representing $(\mathbf{x}'_k, t_k)$ their new location in space-time.

$$\mathbf{x}'_k = \mathbf{x}_k - (t_k - t_{ref})\mathbf{v} \tag{4.1}$$

The IWE I is created according to Equation 4.2 by summing the contribution per pixel of each warped event.

$$I(\mathbf{x}) = \sum_k ker(\mathbf{x} - \mathbf{x}'_k), \tag{4.2}$$

where $ker(x)$ is a kernel using a combination of bilinear voting and Gaussian smoothing to spread the contribution of each warped event to the closest pixels.

The strategy consists of finding the optical flow value that maximizes the contrast function, since images with the sharpest edges corresponds to those with highest contrast. In this case, it is defined as the variance of all pixels in the IWE. The warped events best aligned will result in the IWE with the highest variance. Hence, sharpness edge images can be obtained using contrast maximization.

The estimation of the global optical flow and generation of the motion-corrected edge images operate with groups of N_e events. The input events are packed into groups of size N_e to be processed. Figure 4.1 illustrates the operation of the method, showing the trajectories of events caused by the marker in a space-time region, and the resulting image with sharp edges.

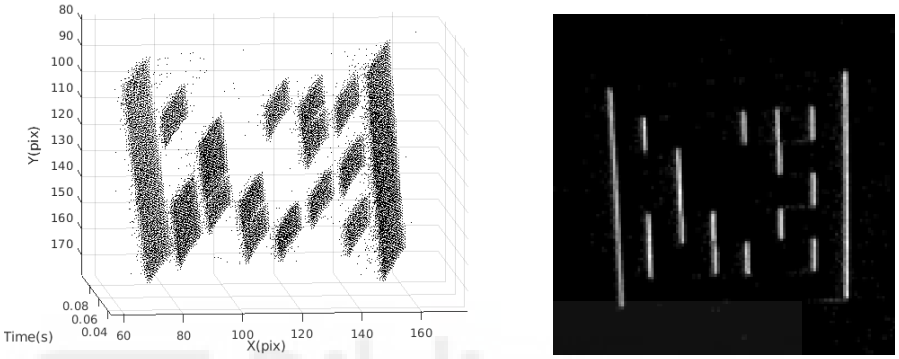

Figure 4.1 Left) Events caused by the marker in a spatio-temporal representation of the image plane. Right) Motion compensated images obtained from the events.

4.2 Cantidates Extraction

4.2.1 Introduction

The input of this module is the motion compensated image, and the objective is extracting the corners of marker candidates. The method is based on the assumption that the markers must be contained in a quadrangular or squared contour. Depending on the perspective, the appearance of the marker's edges can be perceived as either squares or shapes that closely resemble squares. Thus, the method is based on detecting quadrangular shapes similar to squares in the motion compensated images. For this purpose, various geometric constraints are used, which can be adjusted based on the degree of similarity to a square desired for the candidates.

The input images can be classified into three main types based on the relative motion between the camera and markers. If there is motion in the direction of the vertical and horizontal edges of the marker, images will include all its edges. Otherwise, if the motion is only parallel to vertical edges, the images will contain just the horizontal edges of the markers and vice versa. This behaviour, which is a consequence of events generation, is shown in Figure 4.2 with the three different cases.

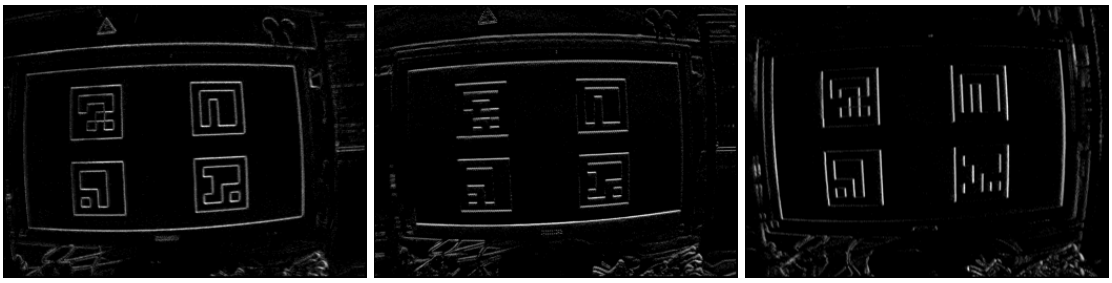

Figure 4.2 Examples of motion compensated images: Left) All markers edges, center) horizontal edges, right) vertical edges.

To address the various appearances of markers in images, the quadrangular shape detection method needs to be adapted to this issue. Therefore, it is not possible to search for contours defined by four vertex in the images as is done in the frame-based methods for standard cameras. The approach adopted to detect quadrangular shapes when having just the edges of horizontal or vertical sides, is based on using line segment detection. By detecting segments in the image, the available sides of the squares can be identified by matching pairs of segments that satisfy certain constraints. Parallelism and segment lengths within a range are the conditions imposed to create pairs of segments of interest. Afterwards, these are analysed to verify whether the desired quadrangular shape can be reconstructed from them or not.

Finally, there can exist multiple candidates associated with the same marker, for instance, when having all the edges of the marker in the image. In order to have a single candidate for each marker, the bounding boxes of the candidates are merged using the Intersection Over Union (IOU) metric as the requirement.

4.2.2 Method description

This section details the proposed algorithm operation for extracting marker candidates. The objective consists of detecting quadrangular shapes similar to squares under certain constraints. In the following, the different stages of the algorithm are described.

Image pre-processing

The input image is a grayscale image with values from the warped events normalized between 0 and 255. The most defined the edges are, the most near to 255 will be its pixel values. Depending on the movement, some edges of the quadrangular shape are not completely defined, containing just some spurious pixels or low values. Moreover, there is also noise in the image due to noisy events, or as a result of optical flow estimation not being accurate enough in motion compensation stage.

Regarding the characteristics of input images, the first step of the algorithm performs a binarization of the image. A global thresholding operation is applied using the Otsu method to find the optimal threshold value. Thus, the resulting image is a binary image in which most of the spurious pixels from undefined edges or noise are removed. An example

of the binarization step operation is shown in Figure 4.3.

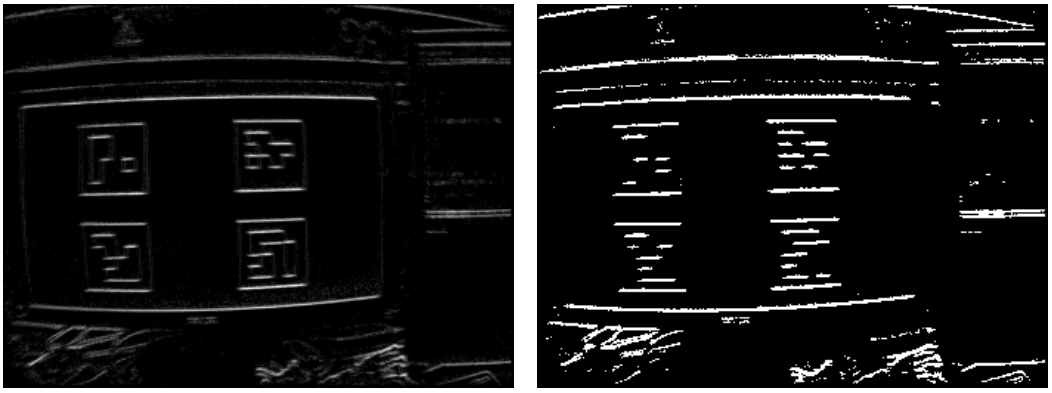

Figure 4.3 Image pre-processing step. Left) motion compensated input image, right) resulting binarized image .

Otsu method [24] is a global thresholding technique, which works well when the image histogram has two distinct peaks, corresponding to the foreground and background regions. To find the optimal thresholding value, the histogram of the image is normalized and analysed as a probability distribution. The operation consists of finding the threshold value that minimizes the weighted intra-class variance or inter-class variance. It iterates through all possible threshold values dividing the pixels into two classes and calculating their class variances. Then, the one with the highest inter-class variance or the lowest intra-class variance is selected.

Line Segment Detection

Line segment detection in the binarized image relies on the Probabilistic Hough Transform [25] method implemented in OpenCV Library. This technique is an efficient implementation of Hough Line Transform [26] and provides as output the extremes of the detected lines.

Segments Pairing

Matching pairs of segments under certain constraints is crucial for detecting candidates with only two sides of the quadrangular polygon. This stage consists of two main steps:

1. **Segments sorting and rejection**.

   First, segments are sorted by their length, then, those under a threshold value, l_{segm}, are discarded. Thus, it is possible to avoid analyzing small segments without any relevance.

2. **Segments comparison and pairing**.

   The similarity between pairs of segments is analysed. In descending order of length, a reference segment, l_{ref}, is chosen and compared with the rest, l_i. If the similarity constraints are satisfied, a pair of segments is created to be validated in the next step

as a marker candidate. The constraints imposed to consider a pair of segments valid are described in Equations 4.3 and 4.4.

First, the segment length ratio is checked to verify that both lengths are close by using:

$$\frac{length(l_i)}{length(l_{ref})} > l_{pair}, \tag{4.3}$$

where l_{ref} is the reference segment selected, l_i the other segment being studied, and l_{pair} the ratio threshold to satisfy length similarity constraint.

Second, the angle formed by the segments is verified to check the parallelism constraint as follows:

$$\theta(l_i, l_{ref}) < \theta_{segm}, \tag{4.4}$$

where θ, see Equation 4.5, is the angle between both line segments vectors, v_i and, v_{ref}:

$$\theta = \arccos\left(\frac{v_i \cdot v_{ref}}{\|v_i\|\|v_{ref}\|}\right) \tag{4.5}$$

Candidates Validation

Pairs of segment matched are required to be analysed in order to validate whether they actually belong to the target marker candidates or not. This stage is also divided into two main steps:

1. **Quadrangular polygon creation**

   Using the segments paired, a polygon is created for later candidate validation. Segment pairs are defined by four points, two per segment. A convex hull is calculated with the four points of each segment pair. Hence, the polygon enclosed between the two segments is defined.

2. **Validation constraints**

   Candidates are validated by checking constraints of interest that polygons should satisfy to enclose the markers. These constraints allow to customize the similarity of the candidate polygon to a square, whereas the previous explanation for segments is a first filtering step.

   First, it is required to verify that the geometry of the polygon is similar to a square. For this purpose, as described in Equation 4.7, the length proportion of the shortest and largest sides of the polygons must be greater than a defined threshold.

$$\frac{l_{min}}{l_{max}} > l_{squares}, \tag{4.6}$$

where l_{max} is the length of the greatest side of the polygon, l_{min} is the length of the shortest, and $l_{squares}$ is the length threshold ratio.

Second, the sides of a polygon similar to a square must form angles around $\frac{\pi}{2}$. Equation 4.7 defines this constraint.

$$|\theta(l_1, l_2) - \frac{\pi}{2}| < \theta_{squares}, \tag{4.7}$$

where l_1 and l_2 are two of the intersecting sides of the polygon, and $\theta_{squares}$ is the angle difference threshold.

Finally, if the constraints above are satisfied, the last verification is the ratio between the polygon diagonals length, Equation 4.8. Values closer to 1 will mean more similarity to a square.

$$\frac{d_1}{d_2} > d_{squares}, \tag{4.8}$$

where d_1 and d_2 are the lengths of the two diagonals of the polygon, and $d_{squares}$ is the diagonal threshold ratio.

If all these constraints are satisfied, the candidate will be considered valid to enclose a marker for later identification.

Markers Candidates Merge

Multiple candidates for the same marker can be detected. Therefore, this final stage aims to merge similar candidates to extract one per marker.

The identification of candidates to merge is performed using the Intersection over Union (IoU) metric. IoU provides an overlapping ratio between two bounding boxes. The approach adopted consists of computing the candidates' IoUs with regard to a reference candidate, and those that exceed a threshold value are stored in a set of similar candidates. The resulting candidate merged for each set is the union of all its candidates. Equation 4.9 defines the IoU metric and the condition to find candidates of the same marker.

$$IoU = \frac{A1 \cap A2}{A1 \cup A2} > IoU_{th}, \tag{4.9}$$

where IoU_{th} is the metric threshold to consider similar candidates. Being A_1 and A_2 the areas of the two analysed regions of interest of the markers, recall that $A1 \cup A2 = A1 + A2 - A1 \cap A2$.

5 Event-based ArUco Identification

This chapter describes the two proposed methods for Event-based ArUco marker identification. This stage provides the output of the pipeline, the markers IDs, receiving as input the marker candidates detected in the image.

5.1 Introduction

When utilizing event cameras for ArUco markers identification, the unique information obtained is the edges due to the contrast between the white and black cells. Figure 5.1 shows an example of an ArUco marker using a standard camera, and the corresponding image using an event camera and motion compensation method. Therefore, there is a paradigm shift as the information to encode are edges instead of pixel values in the cells.

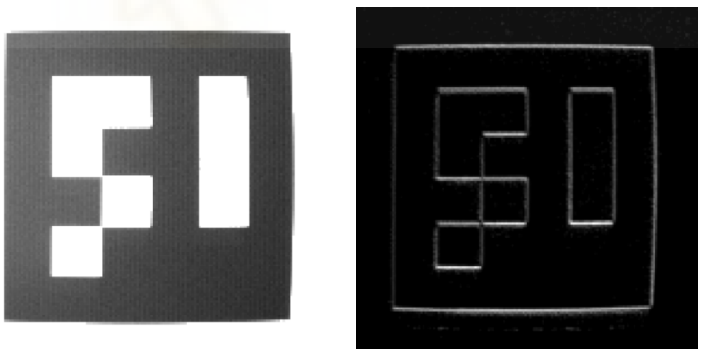

Figure 5.1 Left) Example of ArUco marker using a standard camera. Right) Example using an event camera and motion-compensated images of events.

Two methods are proposed for event-based identification of ArUco markers. First, eAruco-Grid is presented in Section 2, which performs binary encoding of the cell edges using a Gaussian weighted grid. Second, Section 3 describes eAruco-CNN, which is a Deep Learning approach based on a Convolutional Neural Network.

Regardless of the method employed to analyse the input candidates, an initial step is required in this stage. For each input candidate, perspective projection is removed calculat-

ing the homography matrix that maps the candidate corners to a canonical image with a standard size. An example is shown in Figure 5.2.

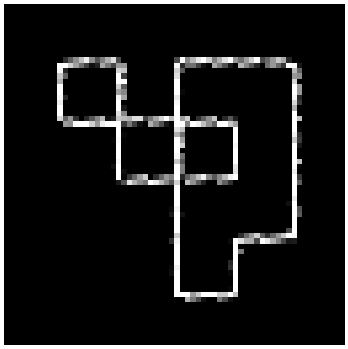

Figure 5.2 Example of a canonical image of a candidate with perspective removed.

5.2 eAruco-Grid

This section details the first method proposed. First, it describes the weighted grid employed for encoding events information. Second, the creation of a new type of dictionary. Finally, the complete algorithm operation for marker identification.

5.2.1 Weighted Grid Binary Encoding

The approach is based on dividing the inner region of the canonical image into a grid with the cell edges positions. A binary value is assigned to each grid region depending on its occupancy. Thus, a binary code is obtained to describe the markers by concatenating all the values.

Figure 5.3 illustrates the grid employed to binary encode cell border information obtained from events. It consists of regions containing cell borders with a margin. The values in these regions are weighted using a Gaussian function centred in the midpoint of the border. Thus, the pixels nearest to the center of the border will be weighted with the highest values, and those nearest to the extremes of border with the lowest values. The gradient in the grid image illustrates the Gaussian weight, with darker green in the center of the segments representing the highest values, and white values in the corners the lowest values.

The weighted grid proposed allows for increasing robustness. First, it avoids the impact of noisy pixels in the information encoding. Moreover, it prevents from border extremes influencing near regions by weighting them with lower values. Finally, the margin given to the region to contain the border enhances robustness against shifting in candidates detection.

To obtain the binary code that describes a marker, the canonical image is analysed using the grid. The result is a value for each region. The values are normalized based on the

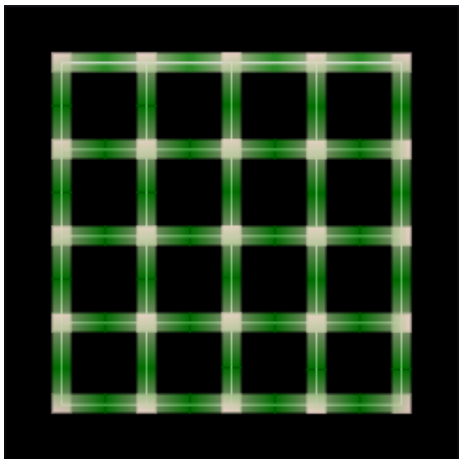

Figure 5.3 Diagram representative of the grid used to encode cell border information.

maximum and then subjected to a threshold. Finally, regions with values over the threshold will be assigned with a value of 1, otherwise with a value of 0.

For different types of markers (4x4, 5x5, 6x6), different grids are defined and adapted to the number of cells. The number of bits required to encode a marker depends on the original marker size of N x N bits. In this case, the grid is of size N_e x N_e, where $N_e = N + 1$ because the grid analyzes the borders instead of cells. Each row and column of the grid consists of N bits. Hence, the number of bits used to encode a marker is determined as: $N_{bits} = 2 \cdot N_e \cdot N$.

Figure 5.4 provides an example in which the detected borders of the marker using the grid are drawn in green over the canonical image.

Figure 5.4 Examples of border detection of 6X6 ArUco marker in cases with all edges, only horizontals, and only verticals.

5.2.2 Dictionary Generation

A dictionary is a data structure containing the binary codes and the identifiers of a set of markers. These are required in the identification stage to find the corresponding ID once the code is extracted.

Because of the new way of encoding the markers using events, original ArUco marker dictionaries can not be used. Therefore, it is required to create a new type of dictionary based on the cell borders.

The generation of the new dictionaries is based on using the ArUco marker images of the original dictionaries. For the desired dictionary, the reference codes are extracted by applying the grid border encoding method on the detected edges in the ideal images of the markers. Thus, for each marker in a dictionary, only two steps are required: First, edge detection on the synthetic marker. Second, code extraction using the proposed grid, and storing the code and marker id.

Furthermore, these types of dictionaries address the issue stated before in which exists images with all borders, only with verticals, or only with horizontals. The approach adopted includes the three cases per marker in the dictionaries. For each original marker, the edges detected are divided into verticals and horizontals, and their code is also extracted. Consequently, the size of a dictionary of N markers is of $3 \cdot N$ codes and identifiers.

Dictionary generation also includes a procedure to achieve rotational invariance in marker identification. When extracting the reference code with the ideal borders, four possible codes are extracted, one for each rotation. Then, the code with the minimum value is selected as the reference one and stored in the dictionary. This procedure requires to be repeated in the identification stage with the marker being analyzed, selecting also the minimum value of the four possible. Hence, the binary codes used as marker descriptors are rotationally invariant.

5.2.3 Identification method operation

The identification algorithm, see Algorithm 1, is based on the two features detailed above: The weighted grid for binary encoding, and the eAruco dictionaries. The input is the canonical image of a candidate. The output is the resulting marker ID, and a measure of how accurate is the identification. This measure is the Hamming distance, d, between the extracted code and the nearest in the dictionary. A distance $d = 0$ indicates that the code is contained exactly in the dictionary. The operation of the algorithm consists of the following stages.

First, for the 4 possible rotations, three codes are extracted and stored, one for each type of border case: full, horizontal and vertical borders.

Second, for each type of code, the minimum value is obtained. This step is performed to achieve rotational invariance.

Next, the codes describing horizontal and vertical borders are looked up in the dictionary D. If the extracted code does not contain any erroneous bits, the marker can be described just with the vertical or horizontal border codes. That is why they are the first checked in the dictionary.

Finally, depending on the results obtained in the dictionary lookup, it is decided which is the identifier. Figure 5.5 illustrates the possible cases in the identification procedure. Case A, if both IDs obtained are the same, this is the selected ID. In cases B1 and B2, if the horizontal or vertical code is found in the dictionary and not the other, the corresponding ID is selected directly.

Cases C1 and C2 are devised as an error correction stage, allowing the identification of the nearest marker when exist errors in code extraction. Case C1 occurs if the IDs associated with codes found in the dictionary are different. Case C2 occurs if neither horizontal nor vertical codes are in the dictionary. In both cases C1 and C2, the ID is searched in the dictionary D using the four codes of full borders case, one per rotation. This relies on hamming distance to find the nearest code. This step is required for error correction because a change of just one bit can negatively impact in the minimum selected for rotation invariance.

As stated above, D is a dictionary containing three rotationally invariant codes per marker: full, horizontal, and vertical borders code. Hence, the size of D is $3 \cdot N$, where N is the number of markers in the dictionary.

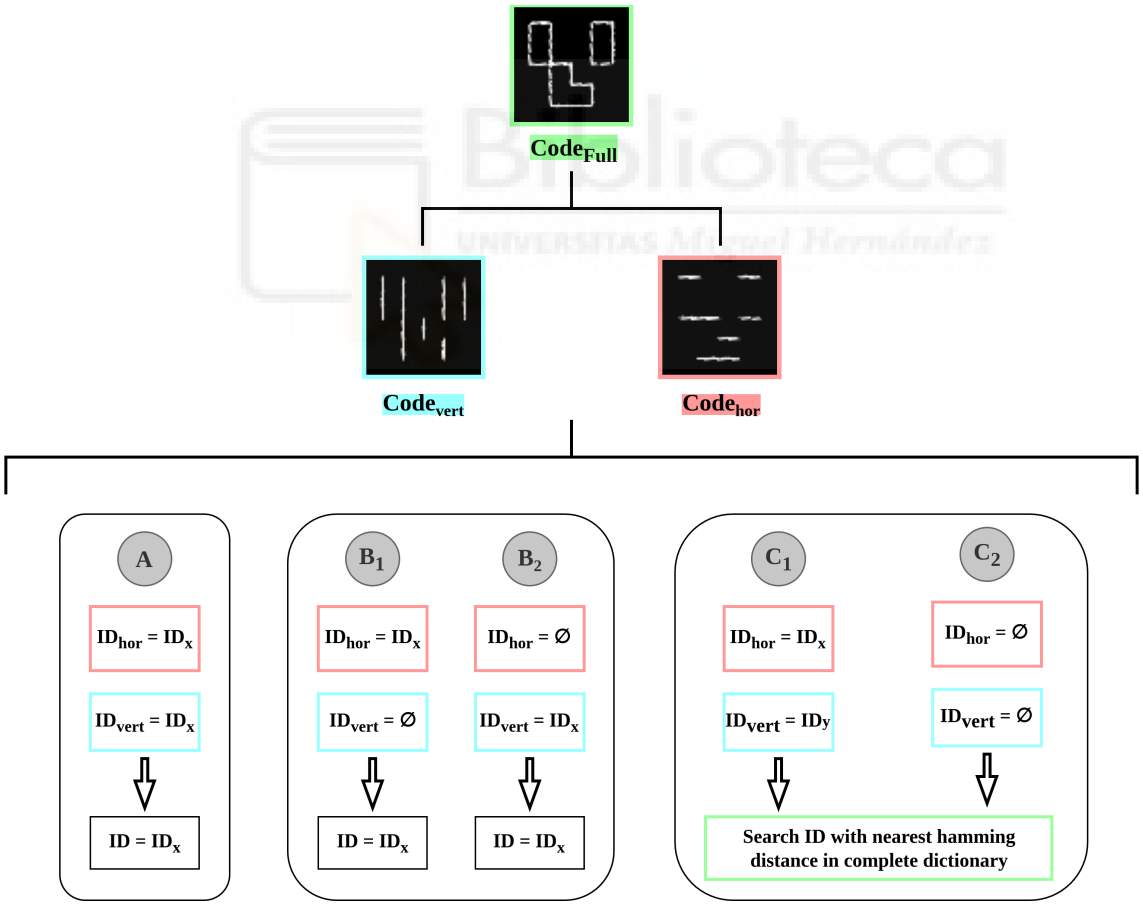

Figure 5.5 Diagram of ArUco Identification possible cases in dictionary lookup step.

Algorithm 1 eArUco Identification Algorithm

Input: I_c ▷ Candidate Canonical Image
Output: ID, d ▷ Marker ID, Distance
$S_f, S_h, S_v \leftarrow$ emptyArray() ▷ Initialization of arrays to store codes of 4 rotations
for $i \leftarrow 1$ **to** 4 **do**
    $code_f, code_v, code_h \leftarrow$ checkGrid(I_c)
    $S_f \leftarrow$ insert($S_f, code_f$)
    $S_h \leftarrow$ insert($S_h, code_h$)
    $S_v \leftarrow$ insert($S_v, code_v$)
end for
$code_{fmin} \leftarrow$ minCode(S_f)
$code_{hmin} \leftarrow$ minCode(S_h)
$code_{vmin} \leftarrow$ minCode(S_v)
$ID_h \leftarrow$ findCode($code_{hmin}$, D)
$ID_v \leftarrow$ findCode($code_{vmin}$, D)
if $ID_h = ID_v$ **and** $ID_h \neq \varnothing$ **then**
    $ID \leftarrow ID_h$
    $d \leftarrow 0$
else if $ID_h \neq \varnothing$ **and** $ID_v = \varnothing$ **then**
    $ID \leftarrow ID_h$
    $d \leftarrow 0$
else if $ID_h = \varnothing$ **and** $ID_v \neq \varnothing$ **then**
    $ID \leftarrow ID_v$
    $d \leftarrow 0$
else
    ID, $d \leftarrow$ findCodeHamming(S_f, D)
end if

5.2.4 Preliminary Results

Dictionaries generation ensures the discrimination between markers. All binary codes used to describe the markers are different. The minimum hamming distance between markers has been obtained for each dictionary of grid sizes 4x4, 5x5, and 6x6. These results are shown in Table 5.1.

Table 5.1 Minimum Hamming distance for each dictionary depending on the number of markers.

	50	**100**	**250**	**1000**
4x4	2	2	2	2
5x5	6	6	4	4
6x6	10	10	8	8

Therefore, if code extraction and previous steps are performed successfully, the marker identification is robust due to the ensured discrimination between codes.

5.3 eAruco-CNN

In this section is detailed the Deep Learning approach for identifying ArUco Markers using event cameras. It relies on a trained Convolutional Neural Network to identify the marker IDs. The input required is the canonical image of a marker candidate, and the output provided is the ID predicted. It has been fully implemented using PyTorch [27] framework, which is an open-source machine learning library for building deep learning models.

This method allows for training and validating the model using only synthetic data. Hence, eliminating the need for gathering a large amount of real data to train the CNN. This is a remarkable feature, considering that in the case of event-based vision, there are no public datasets for this task.

For each ArUco dictionary, a different model is trained. Thus, instead of different dictionaries for code look-up, there are pre-trained models for ID prediction.

5.3.1 Model Training Description

The training procedure of the models can be achieved using synthetic data, which is more practical than using real data. A synthetic dataset emulating real data is created for each model trained.

To create the synthetic event-based ArUco datasets, all the original ArUco markers contained in the specific dictionary are used. Applying an edge detector to each marker image, three new images are created. One with all the borders detected of the marker, another only with the horizontal edges, and another only with the vertical edges. Therefore, for each marker in a dictionary, a dataset of $3 \cdot N$ images is created. An example of the three synthetic images generated for a marker is shown in Figure 5.6.

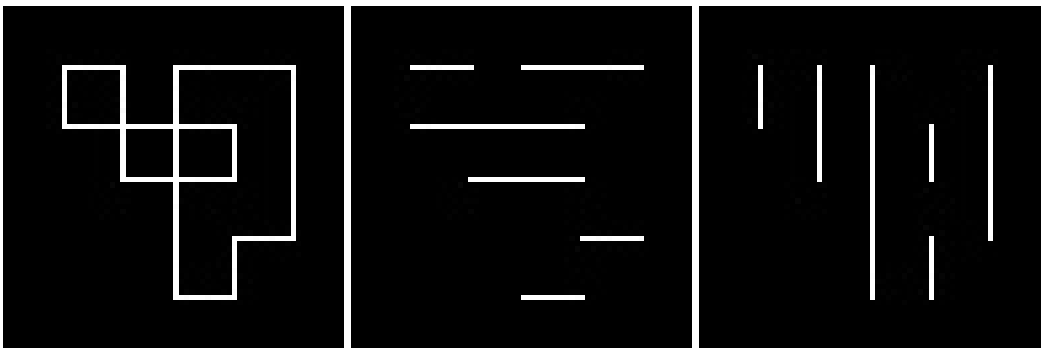

Figure 5.6 Example of the three edges images generated from original marker image.

In order to emulate real conditions in the dataset, data augmentation is applied during the training step. New copies of the images are generated by applying transformation operations. It allows to increase the amount of training data and its variability, and hence, it can reduces overfitting. Therefore, when the model is being trained, transformations

are applied to the synthetic border images generated for a dictionary. The transformations proposed to replicate real data are the following:

- **Rotation**: To achieve rotation invariance, one of the four possible rotations is applied with the same probability.
- **Dilation**: It simulates the effect of thick edges in cases where the motion compensated images are not enough accurate. A $3x3$ kernel is utilized for dilation morphological operation.
- **Shift**. The error in candidate corners estimation is simulated by translating them with an offset. This value has been defined as a proportion of the image and sampled from a uniform distribution in the interval $(-0.05, 0.05)$. Thus, a maximum of ± 0.05 of the canonical image width or height can be shifted in each axis.
- **Noise**. Salt and pepper noise have been added to the images to simulate noisy events, or spurious pixels consequence of compensated images.

The implementation to perform the data augmentation using these transform operations relies on the library *Albumentations* [28]. Figure 5.7 shows some examples of the transformed images generated for training.

Figure 5.7 Example of different images generated by applying transformations.

About the parameters used in the training step of the model, Adam optimizer has been selected to update the weights of the network. This method computes adaptive learning rates for each parameter from estimations of first and second-order moments of the gradients in gradient descent. This combination helps the Adam optimizer to achieve faster convergence and better generalization in many cases. The unique parameter provided to the optimizer has been the initial learning rate. It varies depending on the number of markers N in the dictionary, from 1^{-4} for $N = 50$, to 1^{-6} for $N = 1000$. The batch size employed also depends on N, ranging from 10 to 50. Finally, the number of epochs required to obtain a cross-validation training accuracy over 95% is of 50 epochs for $N = 50$, and 200 epochs for $N = 1000$.

5.3.2 Model Architecture

The architecture of the proposed CNN is shown in Figure 5.8. The input of the network is the canonical image to analyse with size 64x64. The approach consists of a classification problem with N classes, where N is the number of markers in the dictionary. The outputs of the network are the probabilities of each marker ID to correspond to the input marker.

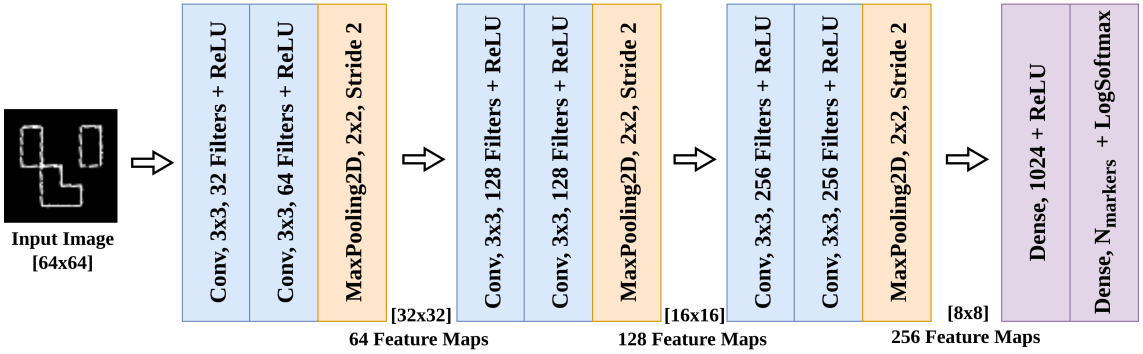

Figure 5.8 Architecture of the CNN proposed for event-based ArUco identification.

The model is composed of three blocks in which features from the input images are extracted. At the end, two fully connected or dense layers are employed for the classification problem.

The blocks for feature extraction contain two convolutional layers, followed by a MaxPool layer to reduce the number of parameters of the model while maintaining the most relevant features. The number of filters used in the convolutional layers is increased by a factor of two. After each layer, ReLU activation function is applied, see Equation 5.1.

$$f(x) = max(0,x) \tag{5.1}$$

The output of the last dense layer must have the same size N as the dictionary being trained. To obtain the probability of each marker ID corresponding to the input marker, the LogSoftmax function is applied at the end of the model after the last dense layer. The reasons for using this function are the following.

Softmax function, see Equation 5.2, is a function typically used at the end of the networks to convert the raw output or logits into a probability distribution that sums one. LogSoftmax is a variant that improves numerical stability when computing the Softmax function. The process involves computing Softmax and then taking the logarithm of the resulting probabilities, see Equation 5.3.

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)} \tag{5.2}$$

$$LogSoftmax(x_i) = \log(\frac{\exp(x_i)}{\sum_{j=1}^{N}\exp(x_j)}) \tag{5.3}$$

Pytorch implementation of *CrossEntropy Loss* [29] expects receiving the output logits of the network instead of softmax output. Thus, the softmax function can not be included in the model during training. Instead, the combination of the LogSoftmax function and *Negative Logarithmic Likelihood Loss* [30] can be used to train the model with this function at the end. This is because the mentioned loss function requires the log-probabilities of each class as input. Moreover, recovering the probabilities from model predictions is as simple as taking the exponential value of the output. This configuration with LogSoftmax function and *Negative Logarithmic Likelihood Loss* has provided during training the best results in terms of accuracy and convergence, which is more difficult to achieve with the highest number of markers.

6 Experimental results

This chapter presents the experiments in order to validate the performance of the proposed Event-based Aruco Detector. Despite several experiments conducted during the development of the method, three particular experiments are presented for evaluation and results analysis.

First, an accuracy test to evaluate the identification methods operation is presented. It consists of evaluating the eAruco-Grid and eAruco-CNN methods for three different ArUco dictionaries.

Next, two experiments to compare the proposed method against standard ArUco detection under challenging conditions are proposed. First, an experiment with high motion of the camera to evaluate their performance against motion blur. Second, an experiment in a scenario with different illuminations to verify if the detectors are able to identify the markers.

For all these experiments, the number of events, N_e, grouped to generate motion compensated images is $N_e = 20000$ events. The event camera used is the DAVIS346.

6.1 General evaluation

The objective is to validate the operation of the method with different types of dictionaries and evaluate the accuracy of both marker identification methods. Three different dictionaries with N markers are evaluated:

- *DICT 4x4_1000*: Dictionary with grid size 4x4 and 1000 markers.
- *DICT 5x5_1000*: Dictionary with grid size 5x5 and 1000 markers.
- *DICT 6x6_1000*: Dictionary with grid size 6x6 and 1000 markers.

Evaluating these dictionaries, different grid sizes are covered with the maximum number of markers, $N = 1000$. This number of markers is selected to test the methods under worst-case scenarios. Using all possible markers increases difficulty in discrimination between markers.

As stated in Chapter 4, it is verified that the dictionaries generated for eAruco-Grid method can distinguish all markers because they have unique reference codes. Hence, identification errors only can appear from the code extraction step or bad results in motion compensated images. However, the eAruco-CNN method despite obtaining high accuracies in the training step of the CNN models, can not guarantee that errors will not occur during the ID inference step.

Detection accuracy evaluation for all the markers in these dictionaries would require collecting a great amount of data. Instead, the approach adopted consists of selecting ten random markers in each dictionary to be evaluated. The experiments are conducted with the camera pointing to a screen with each selected marker individually placed on a white background. The camera is moved in different directions in front of the marker. First, with a circular trajectory. Second, almost parallel to the horizontal edges of the marker, and finally, almost parallel to the vertical edges.

Accuracy results obtained for this experiment are shown in Tables 6.1, 6.2, and 6.3. The results obtained are remarkable and also prove the behaviour predicted for each method.

Table 6.1 Detection accuracy results obtained for *DICT 4x4_1000* with both methods .

	13	**18**	**315**	**362**	**391**	**428**	**536**	**569**	**953**	**955**
Grid	97.11	97.17	95.50	97.40	94.53	95.64	97.15	98.09	94.48	96.79
CNN	95.67	83.28	93.49	97.49	95.15	95.07	96.67	98.10	94.75	97.04

Table 6.2 Detection accuracy results obtained for *DICT 5x5_1000* with both methods .

	78	**180**	**198**	**401**	**488**	**610**	**836**	**900**	**970**	**985**
Grid	96.98	97.01	96.47	97.04	97.50	96.43	95.24	96.56	97.26	97.07
CNN	96.50	96.90	96.34	96.88	97.40	96.15	95.65	95.71	99.01	97.42

Table 6.3 Detection accuracy results obtained for *DICT 6x6_1000* with both methods .

	106	**235**	**241**	**252**	**265**	**277**	**329**	**518**	**709**	**710**
Grid	97.69	99.24	95.08	97.73	96.93	94.15	98.12	93.77	98.35	95.10
CNN	97.01	97.72	93.91	96.81	96.07	92.23	77.62	91.50	96.71	91.38

When using eAruco-CNN, remarkable results are obtained, but a drop in the accuracy detection of one marker can be observed in cases of *DICT_4x4_1000* and *DICT_6x6_1000*. Despite achieving over 90% of accuracy during model training with validation data, certain markers are not accurately predicted. This can be caused by the overfitting problem, in which the models are excessively tailored to the training data and struggle to effectively generalize with other data. However, these isolated values remain acceptable.

For eAruco-Grid method, high and almost constant accuracy values are achieved. The errors can be originated from inaccurate results in motion compensated images. Despite being a valid and useful method for this application, it is simple because it just estimates

global optical flow, instead of computing the optical flow for different patches in the image. This can also be a source of error in occasional cases before model inference in the eAruco-CNN method.

Several conclusions can be extracted after analysing these results. First, the performance of the method is robust regardless of the different dictionary grid sizes, as results remain similar in all of them. Second, the accuracy values achieved are good enough to validate these methods for detecting ArUco markers using event cameras. Finally, it is verified that the eAruco-Grid method can provide more robustness for identification in these largest dictionaries.

6.2 Comparison with standard camera under challenging conditions

Two experiments are conducted to compare the performance of the ArUco marker detector using a standard camera and the proposed method under some challenging conditions.

6.2.1 High-speed Motion

The standard camera used for conventional ArUco detection is the Intel RealSense D435. In this experiment, both standard and event cameras are set in front of an ArUco marker with a white background. The camera movements are the same as in previous experiments but at a higher speed during all the experiment.

The results obtained in this case are 55.02% accuracy detection for standard ArUco detection, 92.39% for eAruco-CNN, and 94.26% for eAruco-Grid.

The original ArUco detector is not capable of detecting the marker in many cases. This is a consequence of motion blur usually suffered from standard cameras in these conditions. However, the ability of event cameras to capture very fast motions allows marker detection with the proposed method.

Figure 6.1 shows the evolution of the marker detection with both cameras during the high-speed motion experiment.

Figure 6.2 shows two images extracted from the detection procedure with the tested data with high-speed motion. On the left, the blurred frame with no marker detected, and on the right, the motion compensated image with the detected marker by the proposed method.

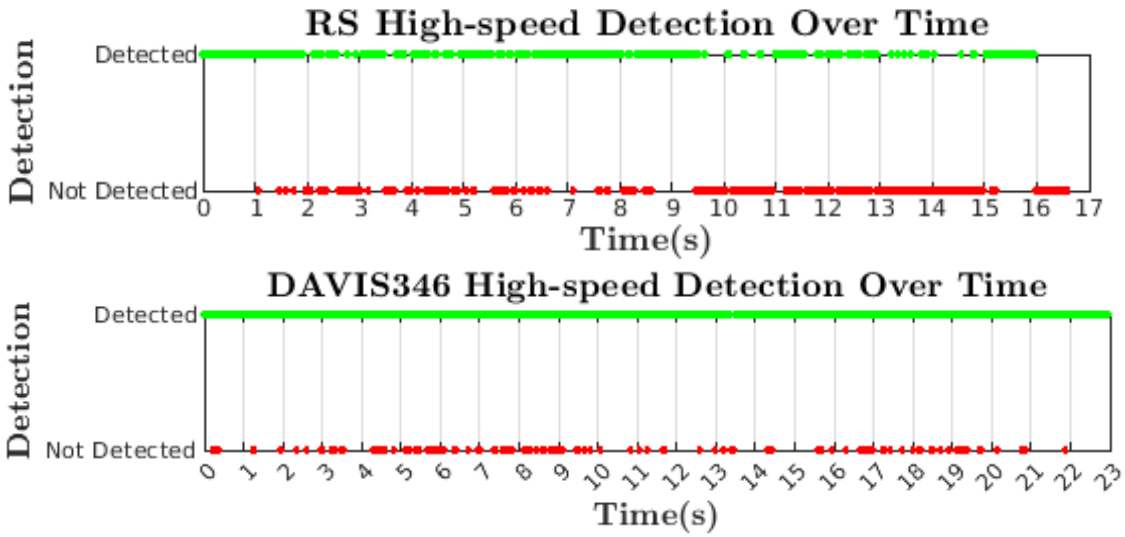

Figure 6.1 Marker detection success evolution during high-speed motion experiment.

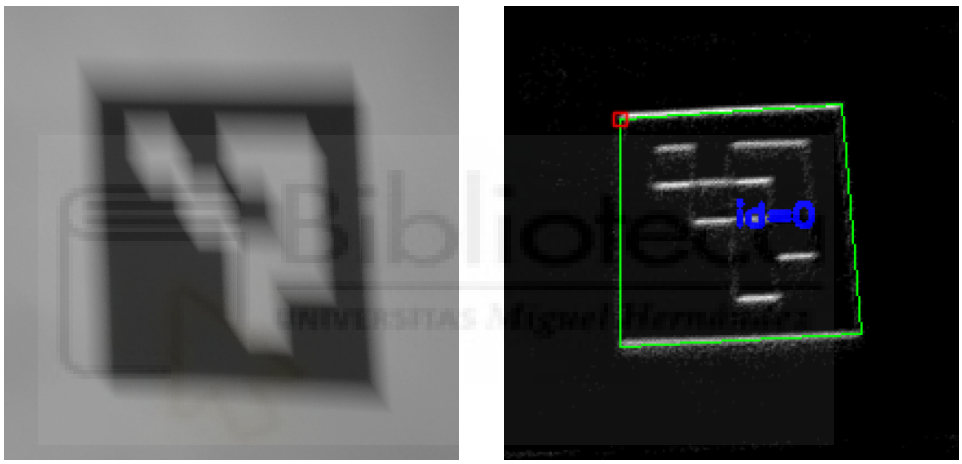

Figure 6.2 Images extracted from high-speed experiments. Left) Blurred image using standard ArUco detection. Right) Motion compensated imaged with the marker detected.

6.2.2 High Dynamic Range

Dynamic range measures the ability of a camera to capture different brightness level in a scene. This experiment aims to compare the capacity of standard ArUco detection against the proposed method under HDR conditions. The cameras are pointing to a marker with normal illumination conditions, and in the middle of the experiment, the illumination of only one half of the marker is increased to high luminance values. The two scenarios are shown in Figure 6.3.

Results obtained for the experiments are the following: 50.73% accuracy detection for standard ArUco marker detector, 83.85% for eAruco-CNN, and 87.44% for eAruco-Grid. These values confirm that in HDR scenarios standard ArUco detection fails, it is just able to detect markers in the first half of the experiments. On the other side, the proposed method is able to maintain acceptable accuracy values, leveraging the HDR of event cameras. Most of

the errors in the event-based detector are caused by motion compensated images, especially during the abrupt change in lighting conditions.

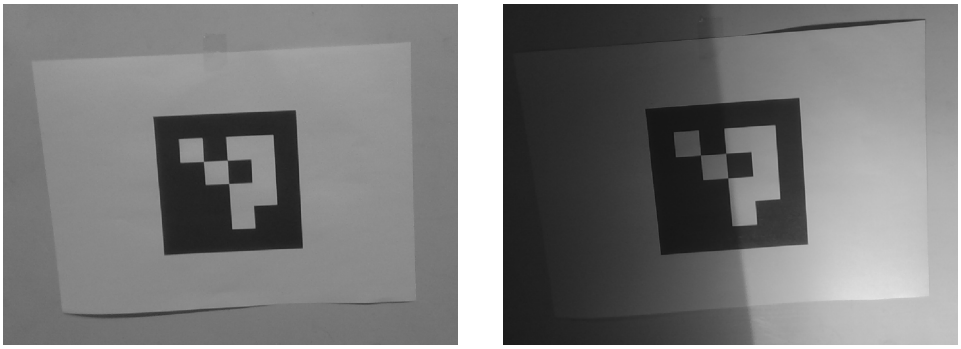

Figure 6.3 Images extracted from HDR experiments. Left) Scenario of the first half of the experiment. Right) Scenario of the second half of the experiment.

Figure 6.4 represents the evolution of the marker detection with both cameras during this experiment. It can be observed that in the second half of the experiment with the HDR scenario, the standard camera is not able to detect the marker, while the DVS346 detects it with the proposed method.

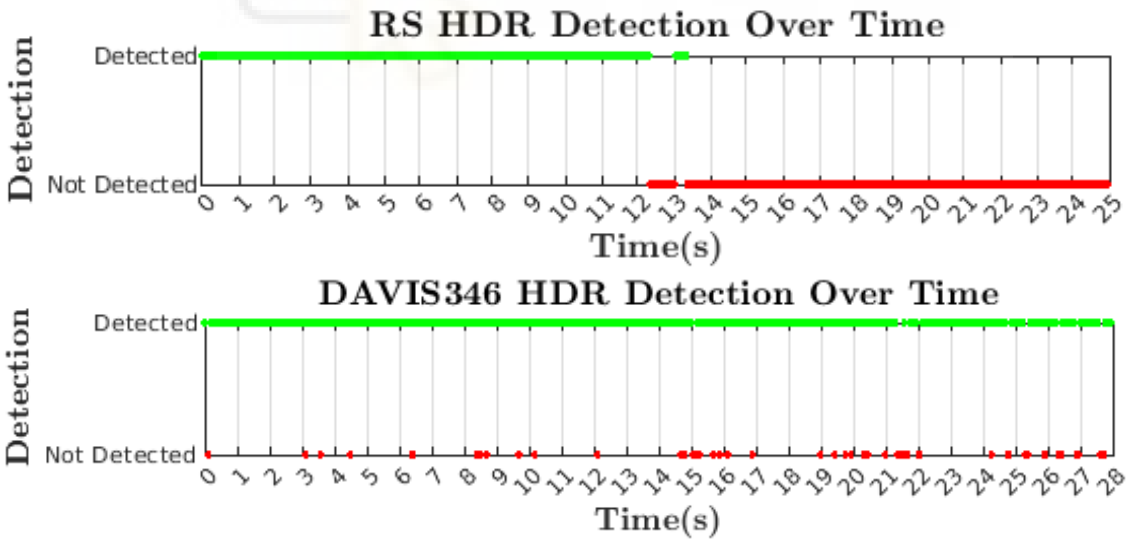

Figure 6.4 Marker detection success evolution during HDR experiment.

Figure 6.5 shows the resulting detection of the proposed method in HDR conditions, in which the standard detector does not detect the marker.

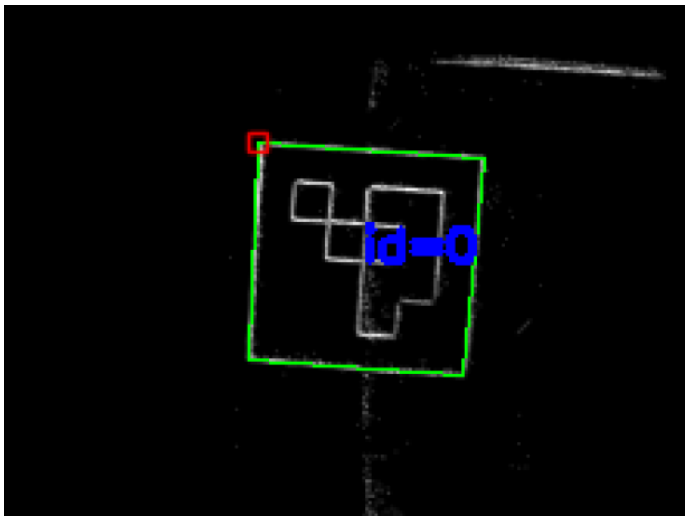

Figure 6.5 Image of a detected marker in HDR scenario.

6.3 Conclusions

The experimental results obtained verify the performance of the proposed method. Detection accuracies achieved in both identification modules validate them to be used for detecting ArUco markers using event cameras.

7 Conclusions and Future Work

This chapter summarizes the conclusions of the proposed project and some potential future work to improve the Event-based ArUco detector.

7.1 Conclusions

Event-based cameras involve a paradigm shift compared to traditional computer vision. Due to the recent breakthrough of these sensors, there is a lack of event-based algorithms compared to traditional computer vision algorithms. Thus, there is a need to further develop specific algorithms for event-based vision. Particularly, this project makes a contribution by developing an event-based algorithm for ArUco marker detection. Moreover, two different methods for the identification of the markers are proposed.

The results obtained in the previous chapter prove the successful performance of the method. Robustness has been demonstrated by testing the method under different challenging conditions such as high-speed camera motion and HDR scenarios. Accuracies of over 90% have been achieved, which is considered satisfactory for concluding the project's objective.

7.2 Future Work

This section outlines some potential works in each module of the pipeline for improving the performance of the proposed method.

7.2.1 Motion compensated images

The current method estimates global optical flow in the edge image motion-correction creation. Thus, is assumed that the optical flow of all the warped events is the same. Despite providing results good enough for this application in most cases, an improvement could be extending the method for estimating optical flow for different patches in the image.

This could lead to achieving highly precise motion compensated images and thus enhancing the accuracy detection of the pipeline.

7.2.2 Candidates Extraction

One of the biggest challenges in this module is the extraction of the candidates when having only horizontal or vertical borders.

Two possible lines of work for improving this module are possible. First, more extensive validation and parameter tuning for particular cases in which the perspective makes candidates differ much from squares. Second, a rejection step before identification could be included to avoid evaluating candidates with similar shapes that could be easily rejected. For instance, candidates which almost all pixel values in the inner region are zero.

7.2.3 eAruco-Grid

The main relevant aspect to improve in this module is the efficiency of marker encoding. Due to the border encoding approach, the number of bits required is higher than in standard markers procedure with cells.

The focus should be placed on reducing the dimension of the binary codes obtained with the grid. For that purpose, discrete principal component analysis or similar techniques could be explored for dimensionality reduction of binary descriptors to a fixed number of bits.

7.2.4 eAruco-CNN

The performance of the models trained with synthetic data is remarkable. However, to improve cases in the largest dictionaries in which the accuracy drops, some regularization techniques such as dropout can be used. A set of neurons is randomly deactivated during training, preventing from learning statistical noise in the dataset and improving generalization.

Additionally, research in testing different network architectures can be performed to find if the same results can be achieved with less parameters.

List of Figures

List of Tables

Bibliography

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

[2] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, 1999, pp. 85–94.

[3] J. Rekimoto, "Matrix: A realtime object identification and registration method for augmented reality," in *Proceedings. 3rd Asia Pacific Computer Human Interaction (Cat. No. 98EX110)*. IEEE, 1998, pp. 63–68.

[4] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 2005, pp. 590–596 vol. 2.

[5] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.

[6] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," 2007.

[7] B. Atcheson, F. Heide, and W. Heidrich, "Caltag: High precision fiducial markers for camera calibration." in *VMV*, vol. 10, 2010, pp. 41–48.

[8] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235

[9] L. B. Gatrell, W. A. Hoff, and C. W. Sklair, "Robust image features: concentric contrasting circles and their image extraction," in *Cooperative Intelligent Robotics in Space II*, W. E. Stoney, Ed., vol. 1612, International Society for Optics and Photonics. SPIE, 1992, pp. 235 – 244. [Online]. Available: https://doi.org/10.1117/12.56761

[10] D. Lopez de Ipiña, P. R. Mendonça, A. Hopper, and A. Hopper, "Trip: A low-cost vision-based location system for ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 6, pp. 206–219, 2002.

[11] L. Naimark and E. Foxlin, "Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker," in *Proceedings. International Symposium on Mixed and Augmented Reality*, 2002, pp. 27–36.

[12] F. Bergamasco, A. Albarelli, E. Rodolà, and A. Torsello, "Rune-tag: A high accuracy fiducial marker with strong occlusion resilience," in *CVPR 2011*, 2011, pp. 113–120.

[13] L. Calvet, P. Gurdjos, C. Griwodz, and S. Gasparini, "Detection and accurate localization of circular fiducials under highly challenging conditions," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 562–570.

[14] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," in *Fourth Canadian Conference on Computer and Robot Vision (CRV '07)*, 2007, pp. 165–174.

[15] G. Yu, Y. Hu, and J. Dai, "Topotag: A robust and scalable topological fiducial marker system," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 9, pp. 3769–3780, 2021.

[16] F. Bergamasco, A. Albarelli, and A. Torsello, "Pi-tag: a fast image-space marker design based on projective invariants," *Machine vision and applications*, vol. 24, pp. 1295–1310, 2013.

[17] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and vision Computing*, vol. 76, pp. 38–47, 2018.

[18] V. Mondéjar-Guerra, S. Garrido-Jurado, R. Muñoz-Salinas, M. J. Marín-Jiménez, and R. Medina-Carnicer, "Robust identification of fiducial markers in challenging conditions," *Expert Systems with Applications*, vol. 93, pp. 336–345, 2018.

[19] H. Sarmadi, R. Muñoz-Salinas, M. A. Olivares-Mendez, and R. Medina-Carnicer, "Detection of binary square fiducial markers using an event camera," *IEEE Access*, vol. 9, pp. 27 813–27 826, 2021.

[20] S. Zhang, Y. Huang, X. Pei, H. Lin, W. Zheng, W. Wang, and T. Hou, "An improved led aruco-marker detection method for event camera," in *International Congress on Communications, Networking, and Information Systems*. Springer, 2023, pp. 47–57.

[21] N. T. Mai, R. Komatsu, H. Asama, and A. Yamashita, "Pose estimation for event camera using charuco board based on image reconstruction," in *2023 IEEE/SICE International Symposium on System Integration (SII)*, 2023, pp. 1–6.

[22] P. R. G. Cadena, Y. Qian, C. Wang, and M. Yang, "Spade-e2vid: Spatially-adaptive denormalization for event-based video reconstruction," *IEEE Transactions on Image Processing*, vol. 30, pp. 2488–2500, 2021.

[23] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3867–3876.

[24] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[25] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer vision and image understanding*, vol. 78, no. 1, pp. 119–137, 2000.

[26] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[27] "PyTorch," https://pytorch.org/, August 2023.

[28] "Albumentations," https://albumentations.ai, August 2023.

[29] "CrossEntropyLoss," https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, August 2023.

[30] "NLLLoss," https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html, August 2023.