Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems

# Relative Pose Estimation Using Event-Based Measurements of LED Signals

**Bachelor's Thesis**

**Jakub Pelc**

Prague, May 2025

Study programme: Open Informatics
Specialisation: Artificial Intelligence and Computer Science

**Supervisor: Ing. Vojtěch Vrba**

## Acknowledgments

I would like to express my sincere gratitude to all those who helped during the writing of this thesis. Specifically, I would like to thank my supervisor, Ing. Vojtěch Vrba, for his invaluable help and feedback during the many measurement sessions, experiments, and analyses, as well as the thesis discussions. I would also like to thank the entire Multi-robot Systems group for generously allowing me to use their equipment and broadening my knowledge of drones, cameras, swarming systems, and many more. Thank you to everyone who helped me gather data during the MRS Camp in Temešvár. Finally, I owe my deepest thanks to my parents for their unwavering support, encouragement, and patience over all those years of my studies.

# BACHELOR‘S THESIS ASSIGNMENT

## I. Personal and study details

| | | | |
|---|---|---|---|
| Student's name: | **Pelc Jakub** | Personal ID number: | **516090** |
| Faculty / Institute: | **Faculty of Electrical Engineering** | | |
| Department / Institute: | **Department of Cybernetics** | | |
| Study program: | **Open Informatics** | | |
| Specialisation: | **Artificial Intelligence and Computer Science** | | |

## II. Bachelor’s thesis details

Bachelor’s thesis title in English:

**Relative Pose Estimation Using Event-Based Measurements of LED Signals**

Bachelor’s thesis title in Czech:

**Odhad relativní polohy pomocí signálů LED měřených event kamerou**

Guidelines:

With their high dynamic range and temporal resolution, modern event-based cameras enable accurate measurements of attributes of modulated LED signals. The goal is to design, implement, and test a relative pose estimation method for UAV swarms that utilizes the response characteristics of these cameras to the LED signals.
1. Research the working principles of event-based cameras. Research the existing Visible Light Positioning (VLP) methods such as Received Signal Strength Ratio (RSSR).
2. Analyze the response of a static event-based camera to UV LEDs used by the UVDAR localization system in UAV swarms. Modulate the signals with varying frequency, relative distance, and angle.
3. Design an approach for relative pose estimation of UAV swarm members, utilizing the analysis results and a proper fisheye lens calibration method.
4. Implement the proposed solution for a Robot Operating System (ROS). Test the implementation on the data used for the response analysis and discuss the results.
5. Conduct a real-world UAV swarming experiment. Compare the estimation results with a GNSS ground truth.

Bibliography / sources:

[1] Gallego, Guillermo, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, et al. 2022. “Event-Based Vision: A Survey.” IEEE Transactions on Pattern Analysis and Machine Intelligence. Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/tpami.2020.3008413.
[2] Scaramuzza, D., A. Martinelli, and R. Siegwart. 2006. “A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion.” Fourth IEEE International Conference on Computer Vision Systems (ICVS’06). IEEE. https://doi.org/10.1109/icvs.2006.3.
[3] Jung, Soo-Yong, Seong Ro Lee, and Chang-Soo Park. 2014. “Indoor Location Awareness Based on Received Signal Strength Ratio and Time Division Multiplexing Using Light-Emitting Diode Light.” Optical Engineering. SPIE-Intl Soc Optical Eng. https://doi.org/10.1117/1.oe.53.1.016106.
[4] Walter, Viktor, Martin Saska, and Antonio Franchi. 2018. “Fast Mutual Relative Localization of UAVs Using Ultraviolet LED Markers.” 2018 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE. https://doi.org/10.1109/icuas.2018.8453331.

|  |
|---|
| Name and workplace of bachelor’s thesis supervisor: |
| **Ing. Vojtěch Vrba    Multi-robot Systems  FEE** |
| Name and workplace of second bachelor’s thesis supervisor or consultant: |
| **RNDr. Petr Štěpán, Ph.D.    Multi-robot Systems  FEE** |

Date of bachelor’s thesis assignment: **22.01.2025**   Deadline for bachelor thesis submission: __________

Assignment valid until: **20.09.2026**

| ___________________________ | ___________________________ |
|---|---|
| prof. Dr. Ing. Jan Kybic | prof. Mgr. Petr Páta, Ph.D. |
| Head of department’s signature | Vice-dean´s signature on behalf of the Dean |

## III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

| ____________________ | ________________________________ |
|---|---|
| Date of assignment receipt | Student’s signature |

**FAKULTA ELEKTROTECHNICKÁ**
**FACULTY OF ELECTRICAL ENGINEERING**
Technická 2
166 27 Praha 6

## DECLARATION

I, the undersigned

| | |
|---|---|
| Student's surname, given name(s): | Pelc Jakub |
| Personal number: | 516090 |
| Programme name: | Open Informatics |

declare that I have elaborated the bachelor's thesis entitled

Relative Pose Estimation Using Event-Based Measurements of LED Signals

independently, and have cited all information sources used in accordance with the Methodological Instruction on the Observance of Ethical Principles in the Preparation of University Theses and with the Framework Rules for the Use of Artificial Intelligence at CTU for Academic and Pedagogical Purposes in Bachelor's and Continuing Master's Programmes.

I declare that I used artificial intelligence tools during the preparation and writing of this thesis. I verified the generated content. I hereby confirm that I am aware of the fact that I am fully responsible for the contents of the thesis.

In Prague on 15.05.2025

Jakub Pelc
...............................................
student's signature

## Abstract

This thesis presents a way to estimate the relative pose of an object equipped with modulated light emitters using an event-based camera as the sensor. Our approach begins by detecting and clustering the asynchronous events into blobs of pixels, for which their centroids are computed. By associating these centroids with the known geometry of the light emitters, the rotation and translation of the object is solved for via the Perspective-n-Point algorithm. To ensure accurate measurements, a customized camera and lens calibration procedure is first performed, which consists of a target with equally spaced diodes that provide bright reference points during calibration. Experimental validation on static and dynamically moving quadrotors demonstrates an average localization accuracy of tens of centimeters in stationary scenarios and a meters-level precision at large distances during active motion.

**Keywords** Event-based Cameras, Pose Estimation, Camera Calibration, Unmanned Aerial Vehicles, Light Modulation

## Abstrakt

Tato práce představuje způsob odhadu relativní polohy objektu vybaveného modulovanými světelnými emitory za použití eventové kamery jako senzoru. Provedený přístup začíná detekcí a shlukováním asynchronních událostí do shluků, pro které jsou vypočteny jejich centroidy. Spojením těchto centroidů s již známou geometrií světelných emitorů je vypočtena rotace a translace objektu pomocí algoritmu Perspective-n-Point. Pro zajištění přesných měření je nejprve potřeba provést kalibraci kamery a objektivu, která zahrnuje snímání kalibrační mřížky s rovnoměrně rozmístěnými diodami, aby byly zajištěny jasné referenční body během kalibrace. Experimentální ověření na statických a pohybujících se kvadrokoptérách ukazuje průměrnou lokalizační přesnost v řádu desítek centimetrů ve stacionárních scénářích a přesnost na úrovni jednotek metrů na velkých vzdálenostech během aktivního pohybu.

**Klíčová slova** Eventové Kamery, Odhad Polohy, Kalibrace Kamer, Bezpilotní Prostředky, Modulované Světlo

## Abbreviations

**DOF** degree-of-freedom

**DVS** Dynamic Vision Sensors

**FOV** Field of View

**GNSS** Global Navigation Satellite System

**LED** Light Emitting Diode

**MRS** Multi-robot Systems Group

**P3P** Perspective-Three-Point

**PnP** Perspective-n-Point

**RANSAC** Random Sample Consensus

**ROI** Region of Interest

**ROS** Robot Operating System

**RSSR** Received Signal Strength Ratio

**RTK** Real-time Kinematics

**UAV** Unmanned Aerial Vehicle

**UV** Ultra Violet

**VLP** Visible Light Positioning

# Contents

# 1 Introduction

The rapid advancement of Unmanned Aerial Vehicle (UAV) swarms has intensified the demand for robust and scalable relative pose estimation methods. Traditional solutions relying on Global Navigation Satellite System (GNSS) suffer from limitations in indoor environments, signal occlusion, and interference that arises from multi-agent communication. Visible Light Positioning (VLP) systems, which leverage modulated Light Emitting Diode (LED) signals and optical sensors, offer a promising alternative due to their immunity to radio frequency interference and high precision.

However, conventional frame-based cameras used in VLP systems struggle with motion blur, latency, and dynamic range constraints. For example, under bright illumination, LEDs may not produce a sufficiently detectable signal, which may lead to localization failure. In contrast, event-based cameras overcome these limitations by asynchronously detecting pixel-level brightness changes, providing microsecond temporal resolution, high dynamic range, and minimal latency [1]. These attributes make them ideal for capturing high-frequency LED signals, even in challenging lighting conditions or during aggressive UAV maneuvers in agile swarming applications.

In this thesis a method for estimating the pose of an UAV equipped with Ultra Violet (UV) LED light sources - as used in the UVDAR system [2] is presented. These LEDs are observed by an event-based camera, with a fisheye lens. First, the LED modulation frequency, distance, and rotation influence on the event-based camera response is analyzed; this analysis informs the subsequent approach used. After the camera is properly calibrated and the LED source locations are identified, a Perspective-n-Point algorithm is used to estimate the location of the UAV in the 3D space. This estimation is then compared with the ground truth positions obtained from the GNSS.

## 1.1 Related work

Shiba et al. [3] released E-VLC dataset for visible light communication, a dataset combining an event camera, a frame camera, and synchronized ground-truth poses in various recording conditions for modulated visible-LED communication and localization tasks.

Ebmer et al. [4] proposed an event-based camera pipeline for real-time 6-degree-of-freedom (DOF) pose estimation using visible active LED markers. Their system achieved a latency below 0.5 ms, with a mean tracking accuracy of 34.5 mm (translation) and 0.738° (rotation). The detection mode showed higher errors, with mean values of 64.9 mm and 1.55° for translation and rotation, respectively. The standard deviations were 16.2 mm and 0.146° for tracking, but increased significantly to 121 mm and 5.12° for detection. The maximum observed errors reached 87.8 mm (tracking) and 1.233 m (detection) in translation, and up to 71.9° in rotation during detection.

Gou et al. [5] proposed a hybrid framework combining depth sensor data and event-based camera streams in a joint random optimization scheme. They achieved robust camera tracking and dense reconstruction under fast motion for agile robotics tasks. They introduce an innovative 3D-2D edge alignment method tailored for event-based camera usage. Their approach achieves robust performance even with high-speed camera motion exceeding 1 m/s.

Liu et al. [6] proposed a line-based object pose estimation method utilizing event-based cameras, by directly detecting object lines from event streams and performing pose optimization using robust estimation techniques. This approach overcomes the challenge of noise interference inherent in event-based sensors by assigning different weights to events based on their distance to the detected lines.

## 1.2 Mathematical notation

The following mathematical notation in Table 1.1 is used, unless otherwise specified.

| | |
|---|---|
| $\mathbf{x}$, $\vec{x}$ | vector, pseudo-vector, or tuple |
| $\mathbf{X}$ | matrix |
| $\mathbf{R}$ | rotation matrix |
| $\mathbf{t}$ | translation vector |
| $x^*$ | optimal solution for $x$ |
| *SO(3)* | 3D special orthogonal group of rotations |
| $\delta(t)$ | Dirac delta function |
| $\mathrm{Conv}(\cdot)$ | convex hull of points |

Table 1.1: Mathematical notation, nomenclature and notable symbols.

# 2 Methodology

## 2.1 Comparison of event-based and frame-based cameras

Traditional frame-based (either rolling or global shutter) cameras capture the scene as a sequence of still image frames at fixed intervals with fixed settings, providing a synchronous representation of the visual world. In terms of ease of use and the simplicity of the post-processing of data obtained from such cameras, they are wildly applicable across many fields. A single frame obtained from a frame-based camera may be described by the following equation (2.1) [7]

$$Y_j(\boldsymbol{p}) := \frac{1}{\epsilon}\int_{t_j-\epsilon}^{t_j} Y(\boldsymbol{p},\tau)\mathrm{d}\tau, \quad j \in 1,2,3... \tag{2.1}$$

where $Y(\boldsymbol{p}, t)$ denotes the irradiation intensity of a camera pixel at a specific time $t$, $t_j$ is the time-stamp of the image capture and $\epsilon$ is the exposure time. As can be seen, each frame is represented by a temporal average of irradiance over the exposure time $\epsilon$. Although this model simplifies image formation, it introduces artifacts such as motion blur, particularly when fast-moving objects are captured with an exposure time mismatched to their dynamics.

Dynamic Vision Sensors (DVS) (or event-based cameras), are vision sensors that draw their inspiration from biological receptors in the retina of the eye, where each pixel reacts to change of the illumination in the scene. Each pixel individually recognizes a logarithm of the intensity and compares it to the previously recorded value. When a predefined threshold is crossed, this value is reset to the current one and a new event is generated. This event can be expressed as $e = \begin{bmatrix} x & y & \sigma & t \end{bmatrix}$, where $\begin{bmatrix} x & y \end{bmatrix}$ is the camera pixel coordinate, $\sigma$ is the polarity of change (where $\sigma = \pm 1$ for increasing or decreasing change, respectively) and $t$ is the timestamp of the event [1] [7]. The single event can be modeled as a Dirac delta function $\delta(t)$ and an event stream $e_i(\boldsymbol{p}, t)$ can be defined at a pixel $\boldsymbol{p}$ by (2.2) [7], with the threshold for generating an event described by the equation (2.3).

$$e_i(\boldsymbol{p}, t) := \sigma_i^{\boldsymbol{p}} c\, \delta(t - t_i^{\boldsymbol{p}}),\ i \in 1,2,3... \tag{2.2}$$

$$\sigma_i^{\boldsymbol{p}} c = \log(Y(\boldsymbol{p}, t_i)) - \log(Y(\boldsymbol{p}, t_{i-1})) \tag{2.3}$$

where $\sigma_i^{\boldsymbol{p}}$ is the polarity (sometimes referred simply as an ON or OFF event) and $t_i^{\boldsymbol{p}}$ is the timestamp of the $i$-th event at a pixel. The magnitude $c$ is the contrast threshold, a preset constant that defines a change in light intensity that is encoded by a singular event, at each pixel. Event-based cameras thus circumvent many common issues found in traditional frame-based cameras, such as the motion blur mentioned before. They offer significant advantages, including high dynamic range, low latency, and energy efficiency. This makes them perfect for the application of agile robotics, where the fast response time is crucial (especially in UAV swarming situations). With their submillisecond response time, event-based cameras can provide a significant advantage over traditional cameras in these applications. However, they also come with some drawbacks, such as the need for a different approach to data processing. While it is possible to reconstruct a logarithmic image from the event stream through temporal integration, enabling the use of conventional vision algorithms, this approach fundamentally undermines the core advantages of event-based cameras. The higher cost of the camera units themselves can also be the deciding factor. [1]

During recording, many camera settings can be changed, such as the Region of Interest (ROI) settings and the bias settings. The ROI setting during recording takes a rectangular region in pixel coordinates and discards events outside the specified region at the hardware level, while reducing computational load and noise from irrelevant areas. This setting is especially useful in cases where only a small static area needs to be observed or where many unrelated events may be generated, which would interfere with the measurement process.

The bias settings are a global camera setting parameters analogous to ISO or exposure time in traditional cameras, though they operate on fundamentally different principles. The configurable biases for the `EVK4` which directly influence event generation and noise filtering, are as follows [8]

- `bias_diff_on` adjusts the threshold on which events are generated; with higher setting, the more increasing change in pixel brightness needs to be present to trigger a generation of an event with positive $\sigma$
- `bias_diff_off` adjusts the threshold on which events are generated; with higher setting, the more decreasing change in pixel brightness needs to be present to trigger a generation of an event with negative $\sigma$
- `bias_fo` adjusts the low-pass filter which filters out the fast fluctuations in light intensity, effectively setting the maximum detectable oscillation frequency
- `bias_hpf` adjusts the high-pass filter which filters out the slow fluctuations in light intensity, setting the minimum detectable oscillation frequency
- `bias_refr` adjusts the pixel refractory period in which a pixel is inactive after generating an event

These settings were optimized during data acquisition to suppress noise and extraneous events (such as from wall reflections or scene illumination changes), which ensured that only events related to the experiment were captured.

## 2.2 Fisheye calibration model

To ensure accurate representation of distances and angles in the output data of the camera, a calibration needs to be performed with the specific lenses that are used. In this thesis a calibration method proposed by Scaramuzza et al. [9] for omnidirectional camera calibration is used to calibrate all the equipment used. After the calibration is performed, a mapping of the 3D world coordinates to the 2D image plane is able to be calculated. For this, the calibration method needs to obtain the extrinsic and intrinsic parameters of the camera, where the extrinsic parameters are the rotation and translation of the camera with respect to the world frame and can be expressed by equation 2.4,

$$\begin{bmatrix} X_{camera} \\ Y_{camera} \\ Z_{camera} \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} + \mathbf{t} \tag{2.4}$$

which is an affine transformation that uses a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$. The origin of the camera's coordinate system is at the optical center, that is, at the intersection of the optical axis from the center of the image with the image plane. This can be represented by Fig. 2.1.

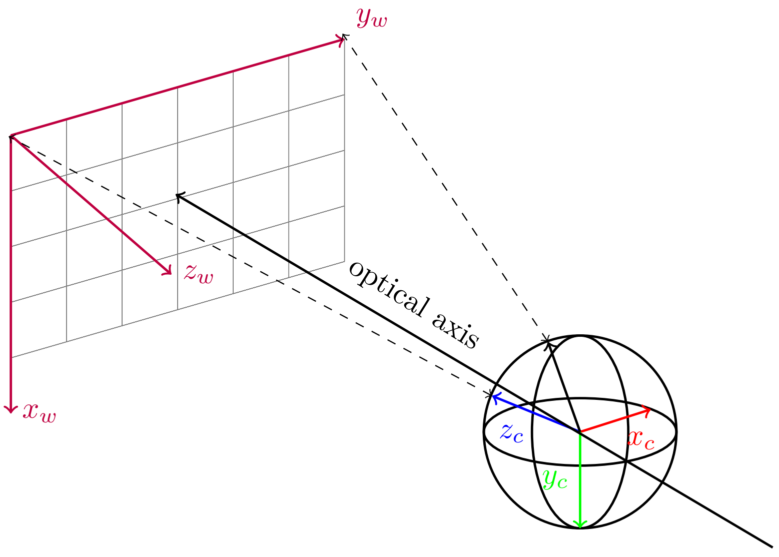


Figure 2.1: Extrinsic parameters of the camera [9].

The next step in the camera calibration is to fit an encompassing ellipse to the received data. The purpose of this is to see the center of distortion of the fisheye lens, as well as the stretching of the image on both the x and y axes. This process can be seen in Fig. 2.2. This ellipse is then used to calculate the point-mapping functions, which allows for transformation of points from the image plane to their respective 3D coordinates.

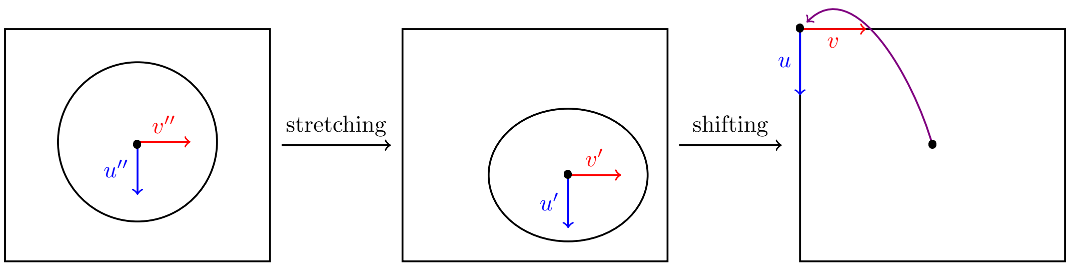


Figure 2.2: Encompassing ellipse fitting [9].

The intrinsic parameters of the camera specify the image format itself, which is influenced by the focal length, sensor size, and optical center. As there are many mapping functions that can be used while modeling the lens (their precision is limited by the manufacturing process), Scaramuzza et al. [9] proposed fitting a polynomial to find the optimal model for lens calibration. The mapping functions are shown in Fig. 2.3.

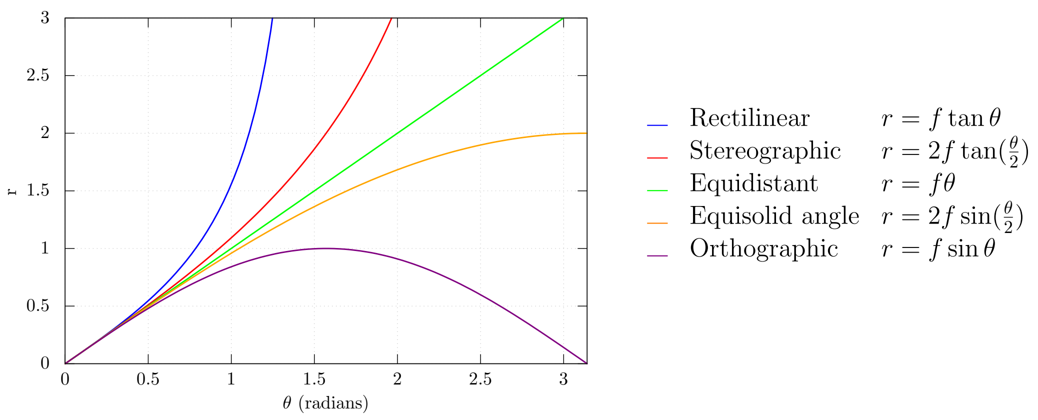


Figure 2.3: Fisheye mapping functions, $f$ is a parameter (focal length).

After fitting a polynomial, the image points can be mapped to their corresponding 3D vectors by an equation 2.5.

$$\lambda \cdot \alpha \cdot \begin{bmatrix} u' \\ v' \\ a_0 + a_1\rho + \cdots + a_N\rho^N \end{bmatrix} = \mathbf{P} \cdot \mathbf{X_w} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.5}$$

where:

- $\lambda$ is a scalar factor
- $\alpha$ is a scaling factor that is obtained from Fig. 2.2 by stretching the ellipse back to a circle and performing the affine transformation; $\alpha, \lambda > 0$
- $\rho$ is the Euclidean distance of a point from the center; $\rho = \sqrt{u^2 + v^2}$
- $a_0, \ldots a_N$ are the polynomial coefficients and $\mathbf{P}$ is the perspective projection matrix.

Another relation between a point $\mathbf{m'} = \begin{bmatrix} u' & v' \end{bmatrix}^T$ in the image plane and its corresponding point on the sensor plane $\mathbf{m} = \begin{bmatrix} u & v \end{bmatrix}^T$ can also be written. The coordinates of $\mathbf{m'}$ have the origin at the top left corner of the image, whereas the coordinates of $\mathbf{m}$ have the origin in the center of the image. This can be represented by an affine transformation $\mathbf{m'} = \mathbf{A}\mathbf{m} + \mathbf{O}_c$, on 2.6 [10].

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} o_u \\ o_v \end{bmatrix} \tag{2.6}$$

where matrix $\mathbf{A}$ accounts for lens-sensor misalignment, and the vector $\mathbf{O}_c$ captures the relation with the center of the distortion.

## 2.3 Pose estimation

Pose estimation can be approached in multiple ways using event-based vision. One way of solving the problem is to directly process the event stream, treating it as a signal that captures the average number of events that occur in certain areas of the sensor. Alternatively, events can be accumulated over time to produce a grayscale image (or a heatmap) that can be analyzed using conventional computer vision techniques for pose estimation.

### RSSR

The Received Signal Strength Ratio (RSSR) method provides an approach to estimate the position of an object by measuring the relative ratio of received optical powers from LED markers. This is done by making each of the LED markers radiate in their assigned time slot, measuring their optical powers at the time of the radiation. Jung et al. [11] demonstrated the RSSR method using a configuration where four LED lamps were mounted on the ceiling with a detector mounted on top of a moving object, parallel to the ceiling. They have arrived at the equation (2.7) which could be used to describe the power ratio $RSSR_{1,2}$

$$RSSR_{1,2} = \frac{P_{R1}}{P_{R2}} = \left(\frac{d_2}{d_1}\right)^{n+3} \tag{2.7}$$

where $P_{R1}$ and $P_{R2}$ are the received powers of the LEDs, $d_1$ and $d_2$ are the distances to the LEDs, and $n$ is the mode number of the radiation lobe (for a Lambertian model $n = 1$). The RSSR pose estimation approach typically consists of a combination of a photo diode, which records the intensity of visible light used for the ratio calculation, and a camera used to capture visual data [12]. The method also requires properly calibrated LED transmitters, with precisely known emission parameters at a known spatial configuration relative to the receiver. This then allows for a reliable estimation of the pose. The method is not used for pose estimation in this thesis due to the lack of time required and the inconclusive results during the analysis of the measured data (no special calibration of the LED transmitters was performed). It will be explored in future work, where it could be used to improve localization precision, if the method is found to be feasible for the application of localizing fast moving UAVs.

### Perspective-n-Point

The Perspective-n-Point (PnP) method has emerged as particularly robust for pose estimation that, unlike RSSR, does not rely on signal strength measurements. The method addresses the estimation of the position of an object relative to the camera, having six DOF

- Rotation - roll, pitch and yaw
- Translation - 3D vector representing the position of the object relative to the camera

This estimation is performed given a set of $n$ known 3D points $\{\mathbf{P}_i\}_{i=1}^n$ on the object and their corresponding 2D projections $\{\mathbf{p}_i\}_{i=1}^n$ in the image plane. In our application, the UAV has four LED-marked arms, where each of them is distinguishable by the camera as a point light source. However, due to physical limitations, only three LEDs may be visible at a single point in time, the fourth obstructed by the physical structure of the UAV when viewed from the front. For a case of only 3 image points, the problem becomes the minimal solvable case, called Perspective-Three-Point (P3P), which can be formulated as a set of 3D points $\mathbf{P}_i \in \mathbb{R}^3, i \in \{1, 2, 3\}$ in the world coordinate system, with their corresponding normalized image points $\mathbf{p}_i \in \mathbb{R}^2$, $|\mathbf{p}_i| = 1, i \in \{1, 2, 3\}$. These sets of points are related by a rigid transformation [13]

$$d_i \mathbf{p}_i = \mathbf{R}\mathbf{P}_i + \mathbf{t} \tag{2.8}$$

where $d_i \in \mathbb{R}^+$. Given the rigid transformation relationship shown in (2.8), the P3P problem reduces to solving for the rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$, translation vector $\mathbf{t} \in \mathbb{R}^3$, and depths $d_i > 0$. By exploiting geometric constraints between the 3D points and their 2D projections, the problem can be reformulated algebraically and solved using various techniques. In our approach, the method proposed by Kneip et al. [14] is used, which directly finds the rotation and translation with a novel parameterization of the model. The visualization of the P3P problem is shown on Fig. 2.4.

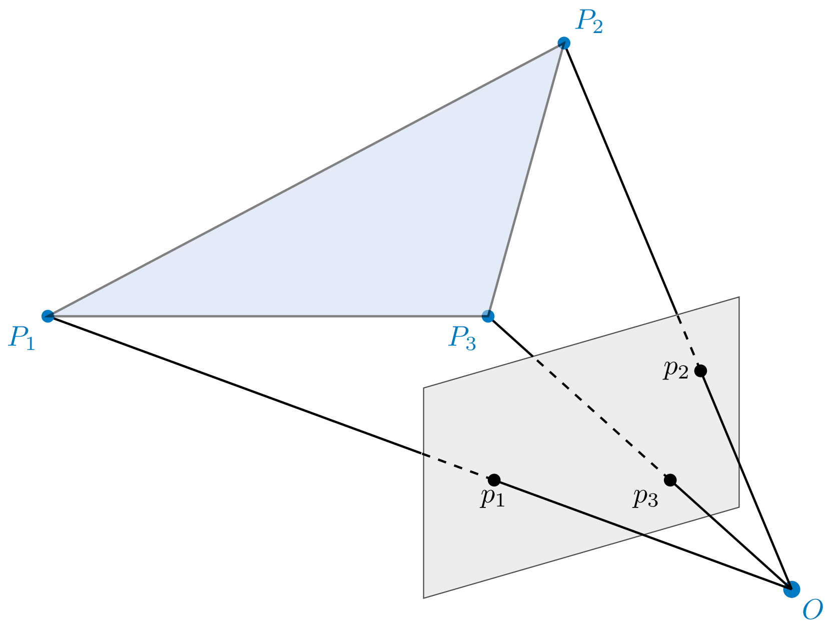


Figure 2.4: P3P problem visualization

In our implementation, the average of the estimated pose and distance is computed, thus simplifying the problem of identifying the correct solution from up to four provided candidates. If all four LEDs on the UAV are detected, a general PnP solution is calculated using the Random Sample Consensus (RANSAC) method. This yields only one solution, so the estimated distance is simply the distance from the camera to the geometric center of the UAV.

# 3 Experiments

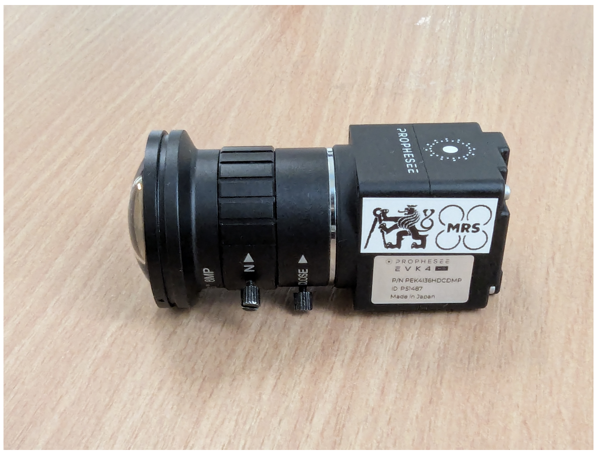
## 3.1 Equipment

### UAVs

The experimental platform for this work was the Multi-robot Systems Group (MRS) X500 [15] quadrotor UAV equipped with UV LEDs integrated to the UVDAR system [2]. Each of the UAV's arms was equipped with 2 UV LEDs at each end of the arm, placed at a right angle relative to each other. The LEDs on each arm could be modulated by using a binary sequence (for example, "$0, 1$" for simple blinking or "1" for a constant ON signal), with a common modulation frequency set for all the LEDs. In our approach, this functionality was used to differentiate between the arms by modulating each arm on a different frequency to be easily distinguishable. The UAV can be seen in Fig. 3.1b.

### Event-based camera

The event-based camera used in this thesis was the `Prophesee EVK4 HD`[1] [16], with `IMX636` sensor. The camera featured a resolution of $1280 \times 720$ pixels and is capable of generating 1.066 Gev/s, although limited by the USB3 interface to $\sim$ 150 Mev/s, and offered a dynamic range of 120 dB.


(a) The event-based camera EVK4 from Prophesee with a 2.5mm fisheye lens.

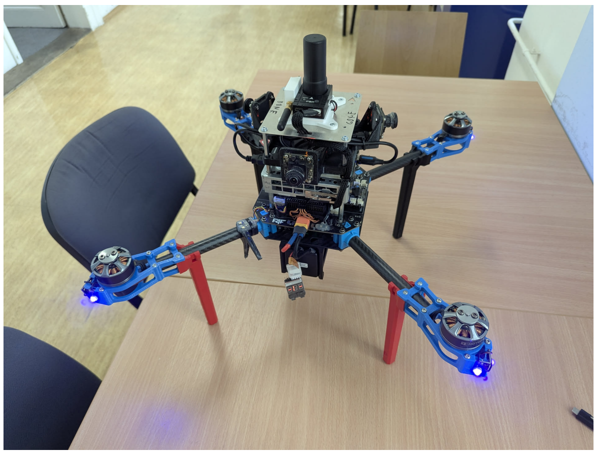

(b) X500 UAV unit equipped with UVDAR

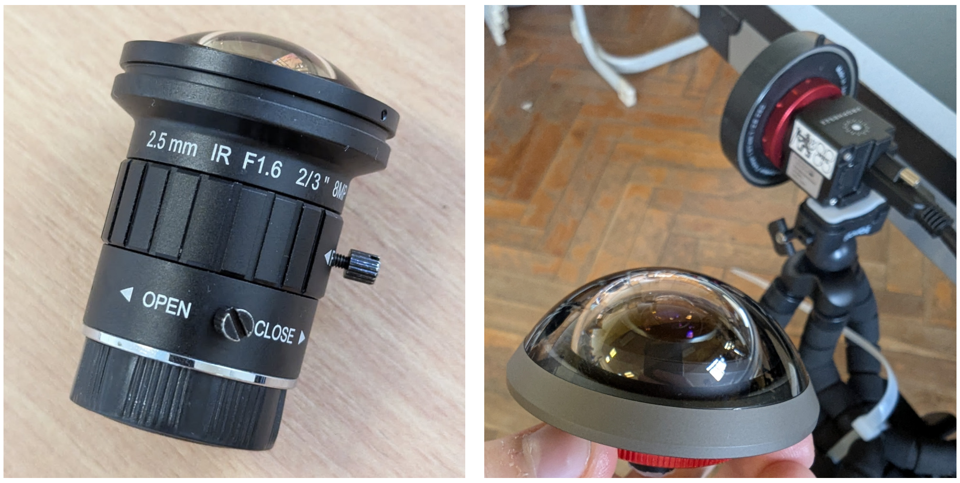
Figure 3.1: An event-based camera with a 2.5mm fisheye lens can be seen on Fig. 3.1a, which was used to measure the UV LEDs mounted on the X500 UAV unit modulated using UVDAR as seen on Fig. 3.1b.

### Lenses

During the measurements, a fisheye lens with a focal length of 2.5mm and the maximum aperture of f/1.6 with Field of View (FOV) of 187 degrees was used. Subsequently, during

[1]Prophesee EVK4 HD website: `https://www.prophesee.ai/event-camera-evk4/`.

the real life experiment, an ultra-wide 1.07mm f/2.8 fisheye lens from Entaniya with FOV of 280 degrees was used in combination with the first lens. Both lenses can be seen on Fig. 3.2. The lenses were equipped with narrowband UV filters which target the specific wavelength of the LEDs that are used in the UVDAR localization system. This ensured that the majority of generated events came from the LED sources mounted at the UAVs, and less events came from the surrounding area. Both lenses were also properly calibrated, which was required to provide correct results in the final pose estimation.



(a) 2.5mm f/1.6 fisheye lens (b) 1.07mm f/2.8 Entaniya fisheye lens

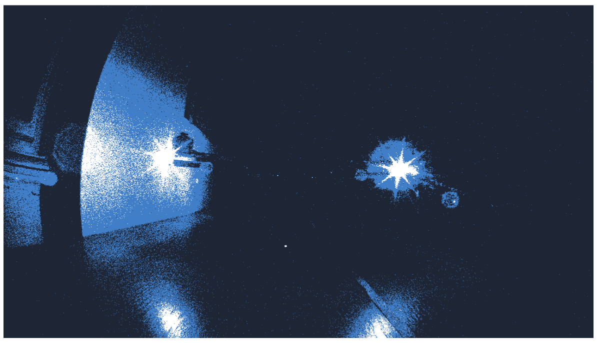
Figure 3.2: Lenses used during the calibration, a 187 degree lens in Fig. 3.2a and Entaniya 280 degree lens in Fig. 3.2b.

## 3.2 Response data collection

### Baseline

The baseline experiment involved a static event-based camera mounted on a tripod, observing a stationary UAV positioned at distances ranging from 0.5 m to 2.5 m under controlled indoor lighting. The UAV's LED markers were programmed to emit pulses of UV light with modulation frequencies ranging from 1 Hz to 30 kHz. No ROI constraints were applied during these recordings. This preliminary experiment revealed critical problems in the measuring technique:

- Multiple visible LEDs: The camera captured the scene as a whole, with no isolation of individual light sources. This means that the results would not have a correct representation of a single light source, which would be influenced by other light sources from the remaining UAV arms as well.
- Reflection artifacts: Reflections from walls and objects present in the scene acted as event-generating light sources, bouncing the light around, as seen on Fig. 3.3. This may confuse some blob detection algorithms for automatic LED source location detection, which would be used in the analysis.
- Capturing of the whole scene: As the whole scene was captured, more post-processing would have been needed to analyze the recorded data and to investigate the relations of measured effects, for example a local ROI filter could have been applied to LED centers detected by a blob detection algorithm.


(a) An event-based view of the UAV with UV LEDs.

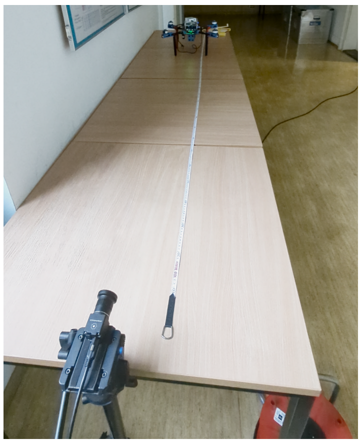

(b) View of the experiment setup.

Figure 3.3: The setup for measuring the event-camera response with a EVK4 camera. Visible reflections from a wall can be seen on Fig. 3.3a. The setup is shown on Fig. 3.3b.

#### Distance - frequency influence

With the critical problems revealed in the previous experiment, only one light source consisting of 2 UV LEDs at the end of the UAV arm was turned on and modulated at set frequencies. Measurements were made in areas isolated by ROI filter directly during recording on the hardware level, events were collected only in the selected area around the end of the UAV arm. This time, the position of the UAV was fixed relative to the camera on a blank background, with no wall reflections. The camera was placed on a tripod and moved in increments of 0.2 meters, starting from 1 meter and ending at 3 meters, with additional measurements made at 4 and 5 meters. The frequency range of the LED modulation was set in a range of 10 Hz to 30 kHz, with the blinking sequence set to "$0, 1$".

#### Rotation angle influence

In addition to distance and frequency influence, the rotation angle influence was also explored to verify the emitting characteristics of the light sources - if they can or cannot be considered Lambertian. The UAV was rotated at increments of 45 degrees relative to the event-based camera, at distances of 0.5, 1 and 2 meters, with frequencies ranging from 10 Hz to 10 kHz and the blinking sequence was set to "$0, 1$".

## 3.3 Calibration data collection

To facilitate calibration, a reference chessboard pattern with known square size is usually used, which offers high contrast between squares, making the square corners easily detectable. Multiple images are usually taken, at various rotations and distances, to obtain good calibration results. In our calibration procedure, a $5 \times 7$ lattice of UV LEDs is used, with the LEDs spaced 50 mm apart from each other. With this pattern, events are generated at the center of the LEDs and can be detected by any kind of blob detector. The LED lattice is used, so the event-based camera can easily detect events from bright LEDs, as opposed to the light being reflected from the chessboard pattern (which does not produce any light on its own). The calibration lattice can be seen in Fig. 3.4.

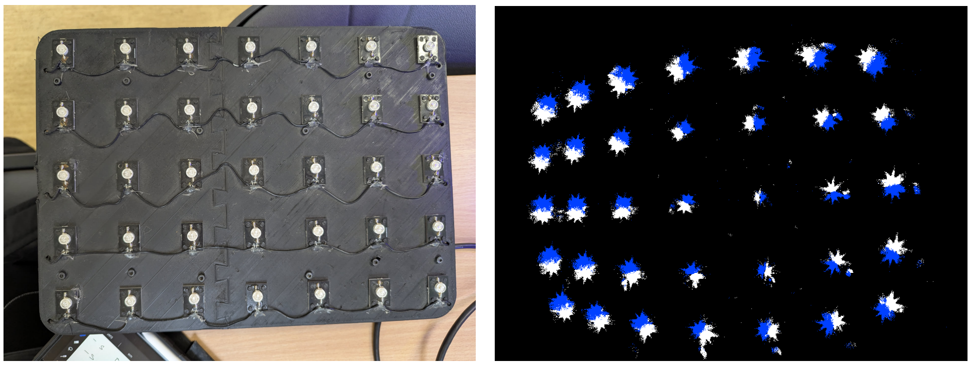

(a) Calibration lattice (b) Image from the event camera of the calibration lattice

Figure 3.4: Calibration lattice of $5 \times 7$ UV LEDs on Fig. 3.4a and the events being produced when placed in front of the camera at Fig. 3.4b, with typical fish-eye lens distortion.

The downside of using an LED lattice instead of a chessboard pattern is that at a very high FOV, the distortion of the fisheye lens sometimes does not allow reliable detection of the exact center of the blobs (or which events correspond to which LED), as can be seen at Fig. 3.5.

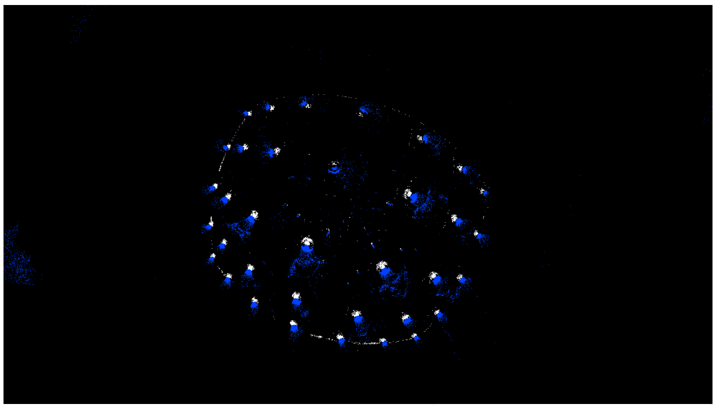


Figure 3.5: Calibration pattern with $5 \times 7$ lattice of UV LEDs, taken with a lens with Entaniya lens with FOV of 280 degrees.

The implementation used for the calibration was a Python library `py-OCamCalib` [2]. As a regular image of a chessboard pattern was not used (required by the library), rather a recording of accumulated events on LED grid, some data preprocessing had to be done beforehand. For the purpose of this, a simple Python app had been written. An event raw recording was loaded, and events were accumulated over a select period of time. The accumulated events were then saved to a grayscale image, where each pixel corresponded to the number of events that occurred in that pixel (positive and negative) with the image being normalized, as shown in Algorithm 1. The LED centers were detected by finding the contours on the binarized image obtained by thresholding, the center of the blobs was selected by finding the brightest spot in the detected blobs as shown in Algorithm 2.

[2] `py-OCamCalib` is available at https://github.com/jakarto3d/py-OCamCalib

**Algorithm 1** Event Accumulation and Blending

```
1: Input: Event stream $E$, accumulation duration $T$
2: Output: Normalized blended accumulation image $I_{\text{out}}$
3: Initialize accumulators:
4: $A_{\text{pos}} \leftarrow \text{zeros}(\text{Height}, \text{Width})$        ▷ Positive event accumulator
5: $A_{\text{neg}} \leftarrow \text{zeros}(\text{Height}, \text{Width})$        ▷ Negative event accumulator
6: Accumulate events for time $T$:
7: for each event $(x, y, p, \_)$ in $E$ within $T$ do
8:     if $p > 0$ then
9:         $A_{\text{pos}}[y, x] \leftarrow A_{\text{pos}}[y, x] + 1$
10:    else
11:        $A_{\text{neg}}[y, x] \leftarrow A_{\text{neg}}[y, x] + 1$
12:    end if
13: end for
14: Normalize and blend:
15: $N_{\text{pos}} \leftarrow \text{normalize}(A_{\text{pos}}, \text{min} = 0, \text{max} = 255)$
16: $N_{\text{neg}} \leftarrow \text{normalize}(A_{\text{neg}}, \text{min} = 0, \text{max} = 255)$
17: $I_{\text{out}} \leftarrow 0.5 \times N_{\text{pos}} + 0.5 \times N_{\text{neg}}$
18: return $I_{\text{out}}$
```

**Algorithm 2** Blob Detection in Grayscale Image

```
Require: Grayscale image $I$ of size $m \times n$
Ensure: List of blob centers $C = \{(x_1, y_1), ..., (x_k, y_k)\}$
1: Step 1: Image Smoothing
2: $I_{\text{blurred}} \leftarrow \text{GaussianBlur}(I, \text{kernel} = 5 \times 5, \sigma = 2)$
3: Step 2: Image Binarization
4: $I_{\text{binary}} \leftarrow \text{Threshold}(I_{\text{blurred}}, \text{threshold} = 50, \text{max} = 255)$
5: Step 3: Contour Detection
6: contours $\leftarrow \text{findContours}(I_{\text{binary}})$
7: Step 4: Blob Center Detection
8: Initialize empty list $C \leftarrow \emptyset$
9: for each contour $c$ in contours do
10:    Create mask $M$ filled with zeros of size $m \times n$
11:    Draw filled contour $c$ on mask $M$ with value 255
12:    $I_{\text{masked}} \leftarrow I$ AND $M$        ▷ Apply mask to original image
13:    $(minVal, maxVal, minLoc, maxLoc) \leftarrow \text{minMaxLoc}(I_{\text{masked}})$
14:    if $maxVal > 0$ then
15:        $C$.append($maxLoc$)    ▷ Add coordinates of brightest pixel in blob to the centers
16:    end if
17: end for
18: return $C$
```

The detected centers were then manually labeled row-wise, in the same order in every image, so the calibration process could identify how the grid of points changed across the images and thus calculate the lens distortion. After this, the regular calibration script from `py-OCamCalib` was used with the prelabeled grid points, the labeling of the LED centers can be seen in Fig. 3.6.

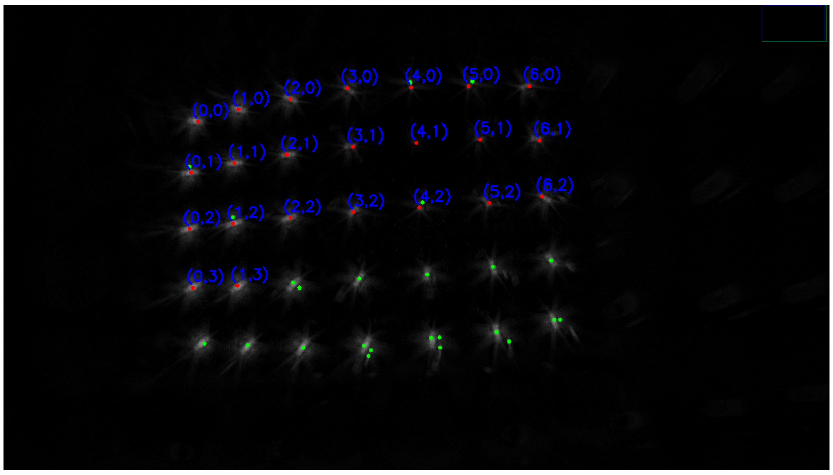


Figure 3.6: Calibration pattern with 5x7 lattice of UV LEDs, with the centers of the LEDs being labeled.

## 3.4 ROS implementation

To facilitate the deployment on real hardware, a Robot Operating System (ROS) `DistanceEstimator` node was implemented [3]. The functionality can be summarized with the flow chart in Fig. 3.7.

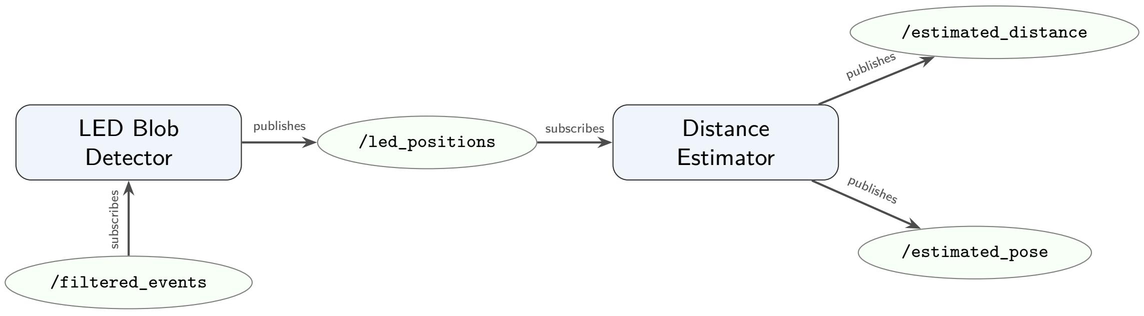


Figure 3.7: ROS distance estimation pipeline

On the input, a filtered event stream is present, with filtering based on the known blinking frequencies of the LED markers. On these events, an image is integrated and a blob detection is run on top of it. The resulting blobs are marked as the LED positions and passed on to the distance estimator, which uses a P3P (or PnP if all markers are visible) to estimate the pose of the UAV. The distance estimator publishes the estimated pose as a quaternion, but also an estimated distance as a floating-point number.

[3]The source code of the distance estimator ROS node is available at https://github.com/kubakubakuba/ros-event-distance

## 3.5 Stationary experiment

An experiment with a stationary UAV and event-based camera was conducted prior to the experiment with flying UAVs to ensure functionality. A UAV was placed on the floor several meters from the camera and rotated around to obstruct one LED in some measurements to test out P3P. The LED blinking frequencies were set to $\mathcal{F} = \{125, 250, 500, 1000\}$ Hz, and a blob detector was used to detect the centers of the visible blobs. The individual LEDs were identified by their visible brightness differences (which were influenced by the event-based camera response to the blinking frequencies) and by the known physical structure of the UAV. The view from the event-based camera can be seen on Fig. 3.8.

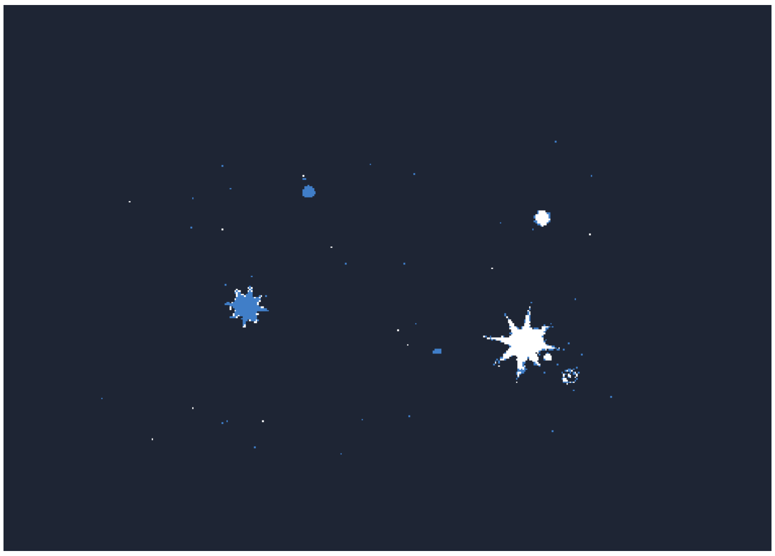


Figure 3.8: Data from the stationary experiment

## 3.6 Real-world experiment setup

The real-world experiment was performed with two X500 UAVs, each of them equipped with a Prophesee EVK4 event-based camera, one with a 2.5mm f/1.6 fisheye lens with an FOV of roughly 187 degrees and the second one with Entaniya 1.07mm f/2.8 fisheye lens with an FOV of 280 degrees. Each UAV was also equipped with a Basler camera with a fisheye lens, to provide normal video signal that was recorded alongside the event stream from the event-based camera. Both cameras were connected to the onboard Intel NUC computer running the ROS system, on which all the processing was done during the flight. Both UAVs were also equipped with a Real-time Kinematics (RTK) module, which was used to localize the UAV, and was used as ground truth data for the pose estimation. The UVDAR blinking frequencies $\mathcal{F} = \{4.0, 2.0, 1.\overline{3}, 1.0\}$ (in kHz) were defined, where each of the arms were blinking at its assigned frequency. The measurements were collected during the MRS Camp in Temešvár in August 2025, the UAVs can be seen on Fig. 3.9.

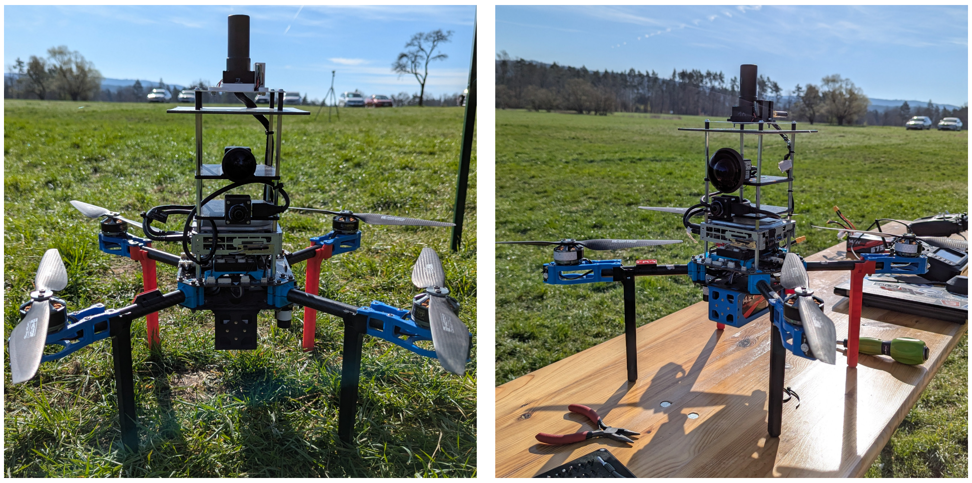

(a) UAV33 (b) UAV37

Figure 3.9: Two X500 UAVs, UAV33 on Fig. 3.9a and UAV37 on Fig. 3.9b.

Two pilots manually controlled the UAVs, systematically varying the distance and angles between them to generate diverse measurement data during the experiment. All in-flight data, including sensor measurements and camera streams, were recorded in a ROS bag file for subsequent analysis in a simulated environment. In addition, raw event stream data from the event-based camera was also recorded and saved. The camera view from the UAV33 can be seen on Fig. 3.10. During the recording about 17 million events were generated every second by each camera, which equates to a data throughput of about 45 MB per second per drone. Each raw recording is about 14 gigabytes in size.

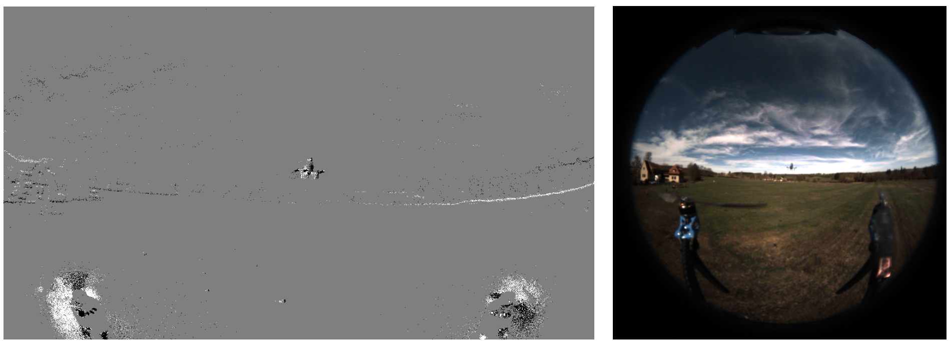

(a) Event-based camera output (b) Basler camera output

Figure 3.10: The view of the experiment from UAV33, with event data on Fig. 3.10a and Basler camera view on Fig. 3.10b.

# 4 Results

## 4.1 Response analysis results

The event-based camera response data was analyzed with the help of the Metavision SDK[1] using its Python API. Each recording can be loaded as a raw file, producing a structured Numpy array of events, where each event is structured as an array of values $(t, x, y, p)$. Specifically, $t$ represents the time stamp from the start of the recording, $x$ and $y$ the spatial location of the event on the camera sensor, and $p$ the polarity of the change in the detected brightness (compared to the previously recorded one).

### Distance - frequency influence

The distance frequency data set has recordings of the UAV placed in front of the camera at distances $\mathcal{D}$ with one LED being modulated at frequencies $\mathcal{F}$ [2]. The tested ranges were:

$$\mathcal{F} = \{10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000, 20000, 30000\}\,\text{Hz},$$

$$\mathcal{D} = \{1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 4.0, 5.0\}\,\text{m}.$$

The obtained data set can be loaded into a matrix representing the distances and frequencies, then a select number of events from each recording can be collected. The data is then resampled into a signal, represented by a 1D array obtained from summing polarities $p_i$ over a selected bin width $\Delta t$ [3] (4.1).

$$S[k] = \sum_{i \in \mathcal{B}_k} p_i, \quad \mathcal{B}_k = \{i | t_k \leq t_i < t_k + \Delta t\}, \quad k = 0, 1, 2, \ldots \tag{4.1}$$

Peaks in this signal are then analyzed by SciPy's `findpeaks` function, and the average number of events with the standard deviation is calculated for each frequency and distance. The influence of distance and frequency on the average number of events can be seen in Fig. 4.1 and Fig. 4.2, respectively. The data show a decreasing trend of the average number of events with the increase of distance or frequency. The drop related to the distance can be explained by the perceived decrease in the intensity of the light source with increasing distance. The decrease in the average number of events at higher frequencies is caused by the Lambertian emission pattern of the LED. As the frequency increases, the light pulses become shorter. For pixels located further from the LED's center, where the light intensity is lower, a longer time is needed to accumulate enough change to trigger an event. At high frequencies, this required integration time often exceeds the short duration of the light pulses, preventing these pixels from generating events. This leads to fewer total events and a perceived drop in brightness. On very high frequencies and distances, the camera is not able to detect any real events at all, as there is more noise generated by the camera itself at this point. This can be observed at Fig. 4.1 with a frequency of 30 kHz at 3 meters.

[1] Metavision SDK Docs: https://docs.prophesee.ai/stable/index.html

[2] The frequencies represented in this list are the actual frequencies sent to the UVDAR unit. The preserved frequencies are half of the values in this list - UVDAR interprets the frequency with a reference to the length of the sequence (here the sequence being "$0, 1$").

[3] The bin width should be adjusted appropriately, as the farther the event-based camera is from the source, the fewer events are generated.

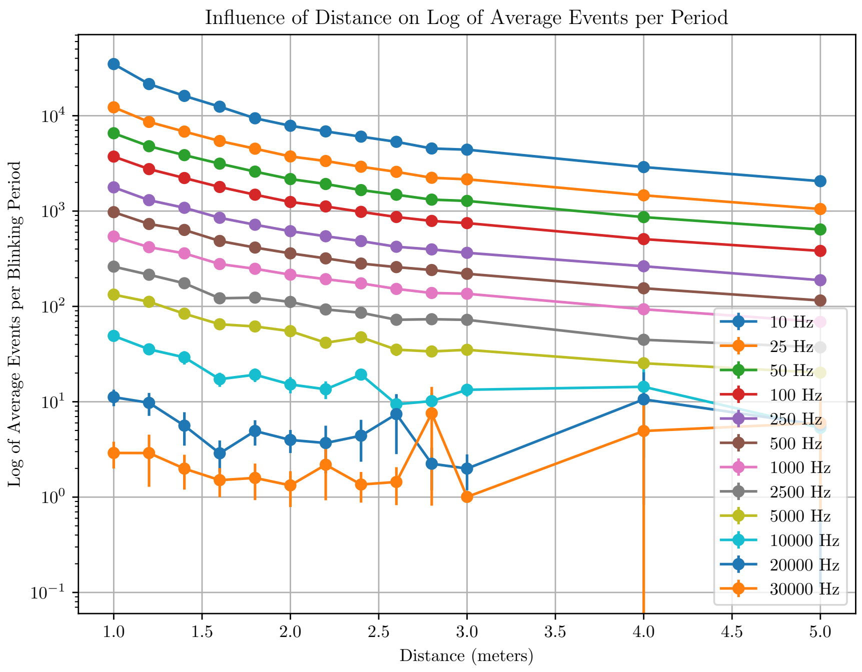


Figure 4.1: Influence of distance on the log of the average number of events, with the UAV rotated 0 degrees relative to the event-based camera.

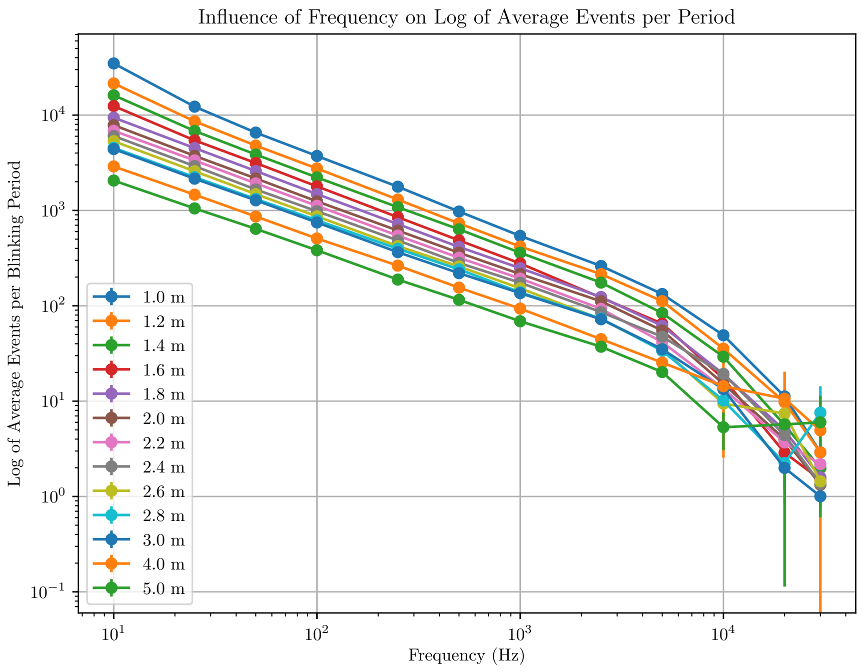
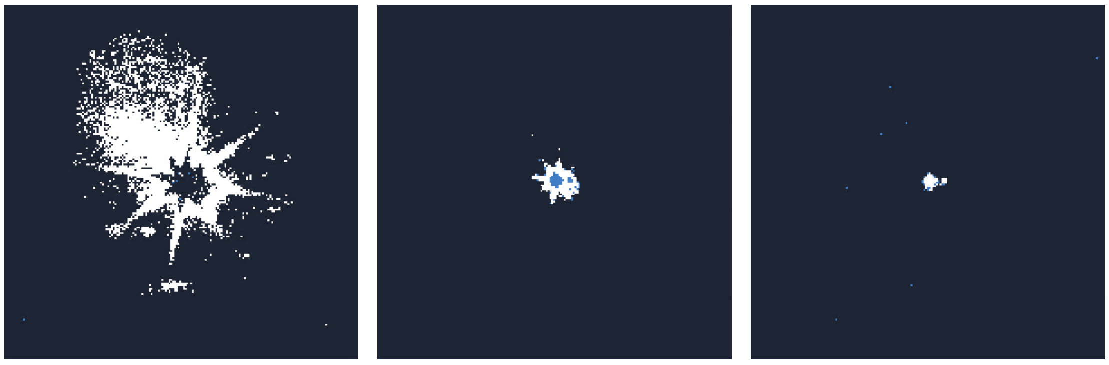


Figure 4.2: Influence of frequency on the log of the average number of events, with the UAV rotated 0 degrees relative to the event-based camera.



(a) 10 Hz blinking frequency (b) 250 Hz blinking frequency (c) 2500 Hz blinking frequency

Figure 4.3: The influence of frequency on the perceived brightness of the LEDs, with blinking frequencies of 10 Hz on Fig. 4.3a, 250 Hz on Fig. 4.3b and 2500 Hz on Fig. 4.3c.

For a selected frequency, the relationship between the average number of events and distance closely mirrors the inverse square law (4.2), a principle well known to describe how phenomena such as brightness, sound intensity, and electromagnetic field diminish with the square of increasing distance. As depicted in Fig. 4.4, when the number of events is normalized with respect to the number of events at 1 meter, and then fitted with the inverse square law, it can be observed that the data copies the theoretical drop in the number of events with increasing distance. This suggests that the number of events generated by the event-based camera is directly related to the brightness of the light source.

$$\text{intensity} \propto \frac{1}{\text{distance}^2} \tag{4.2}$$

While more complex functions could be used to fit the data, they would likely lead to overfitting rather than capturing the underlying trend in a generalizable way, thus the inverse square law provides a good approximation of the data.

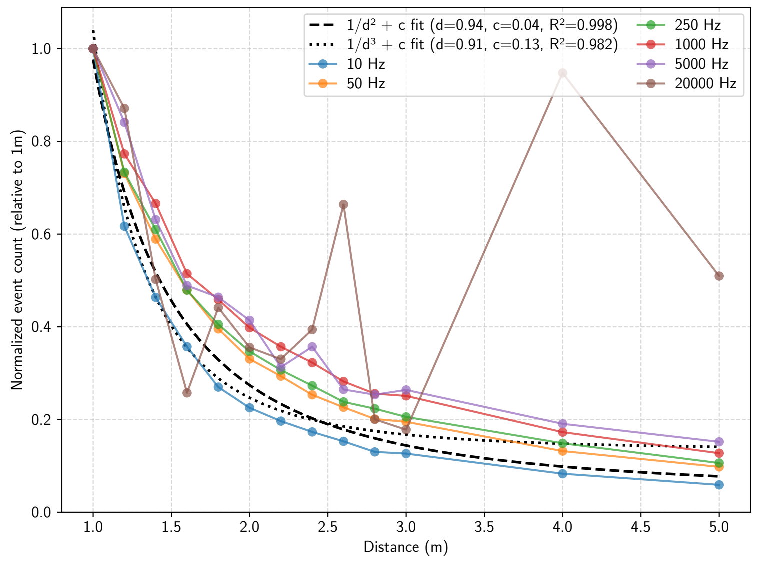


Figure 4.4: Experimental validation of the inverse square law

### Rotation angle influence

From the manufacturer's datasheet of the used ProLight PM2B-1LLE 1W UV Power LED [4] used in the UVDAR system, it can be learned, that the LEDs have a Lambertian radiation pattern, which can be seen on Fig. 4.5.

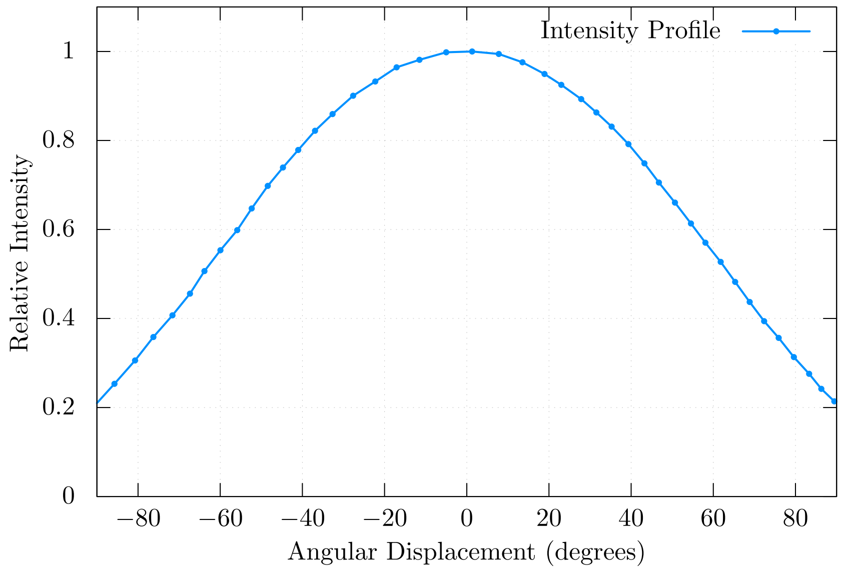


Figure 4.5: Lambertian radiation pattern of the PM2B-1LLE UV LED.

This means that the intensity of the light emitted from the LED decreases with the cosine of the angle between the normal of the LED and the direction of the light (4.3).

$$I(\theta) = I_0 \cos(\theta) \tag{4.3}$$

To represent the whole end of the UAV arm, two LED sources need to be considered, orthogonal to each other. This can be represented by shifting the previous distributions by ±45 degrees and adding them together. The theoretical distribution pattern of the light source is visible in Fig. 4.6. The results extracted from the dataset of the UAV rotations relative to the camera are shown in Fig. 4.7.

[4]The datasheet of ProLight PM2B-1LLE 1W UV Power LED can be obtained from `https://www.tme.eu/Document/9dfb498784ffdd07892a42f4f17c6f37/PM2B-1LLE-DTE.pdf`

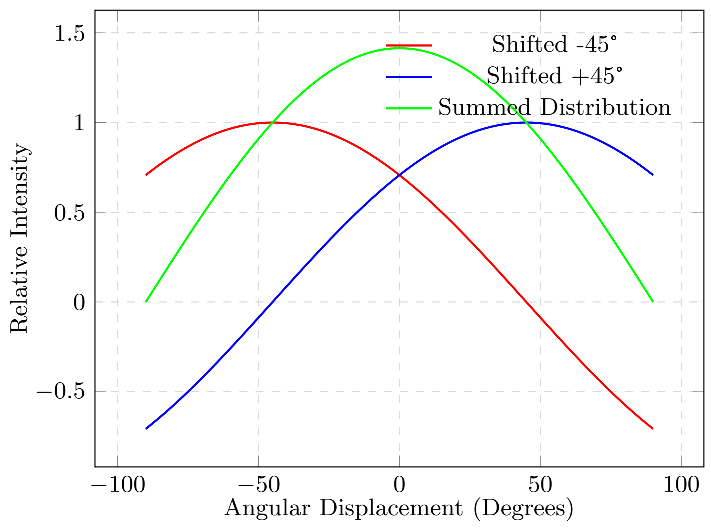
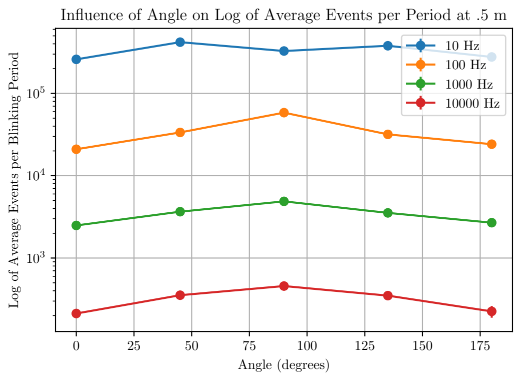
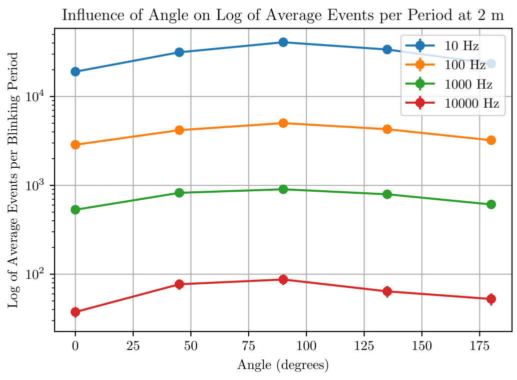


Figure 4.6: Radiation pattern of two lambertian light sources shifted by ±45 degrees.



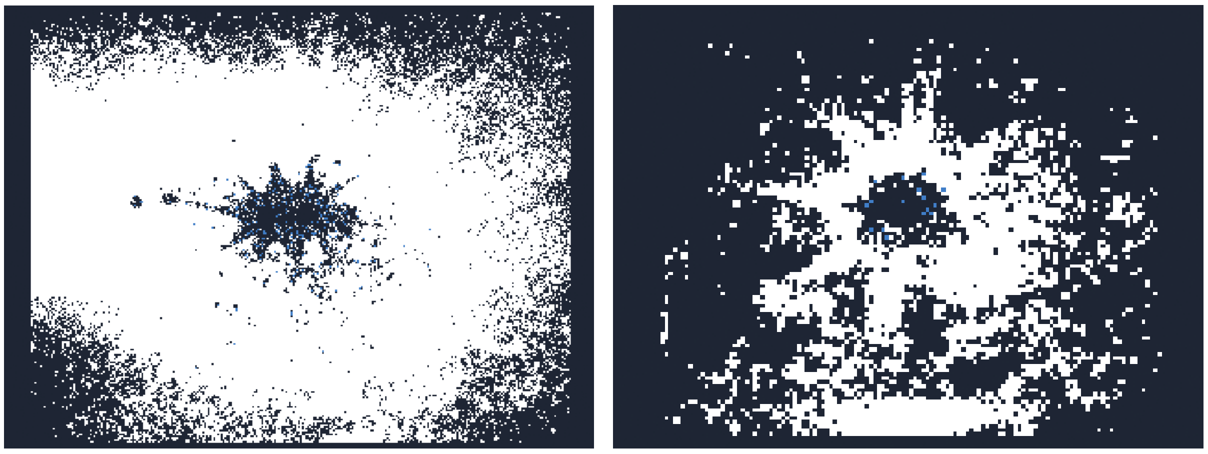
(a) Influence of rotation of the UAV on the log of average number of events at 0.5 m.



(b) Influence of rotation of the UAV on the log of average number of events at 2 m.

Figure 4.7: The influence of rotation angle on the log of average number of events at 0.5 m on Fig. 4.7a and at 2 m on Fig. 4.7b.

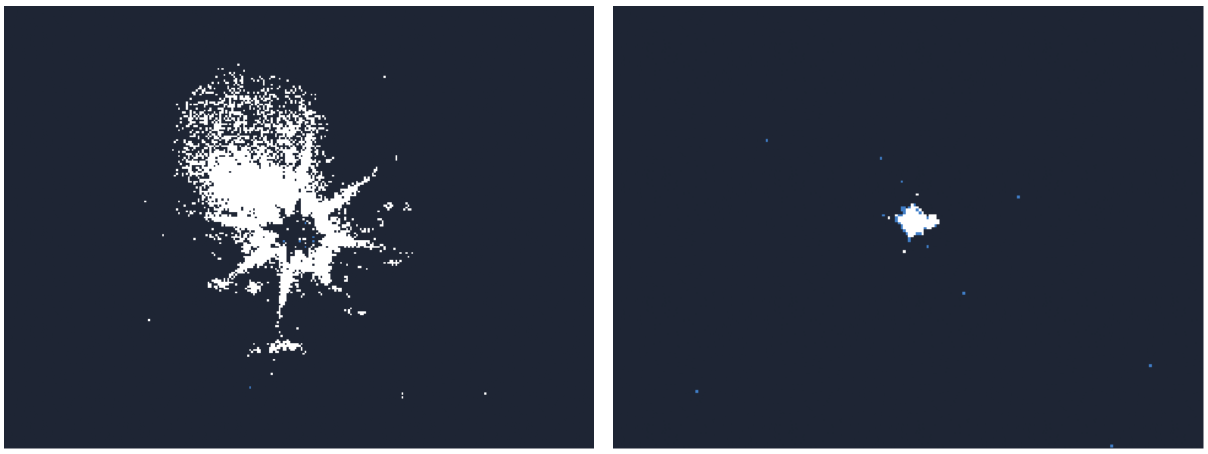
The data show a rough approximation of the theoretical distribution on Fig. 4.6, but with a drop of intensity at the middle of the distribution. This could be caused by the fact that LEDs, when close to the camera, can be perceived as multiple light sources, but when moved further away, they merge into one source as shown on Fig. 4.8.



(a) 2 LEDs with blinking frequency of 10 Hz at 0.5 m. (b) 2 LEDs with blinking frequency of 10 Hz at 2 m.

Figure 4.8: The light source on one arm of the UAV, consisting of two UV LEDs, blinking at a frequency of 10 Hz, placed at 0.5 m on Fig. 4.8a and 2 m at Fig. 4.8b.

What can also be observed from Fig. 4.8 are the star-like shapes of the LEDs, which are supposed to be circular. Those shapes are caused by light diffraction (and are named diffraction spikes), which are, in turn, caused by the aperture blades in the lens of the camera. The number of star spikes depends on the number of blades, the set aperture and the light source intensity then causes stars of different levels of profoundness [17]. This can be observed by comparing how profound the star shapes are on different frequencies, as shown on Fig. 4.9.

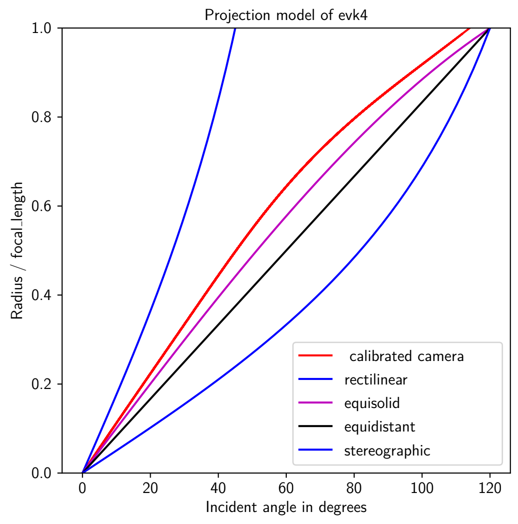


(a) LED blinking at 10 Hz at 1.0 m (b) LED blinking at 1 kHz at 1.0 m

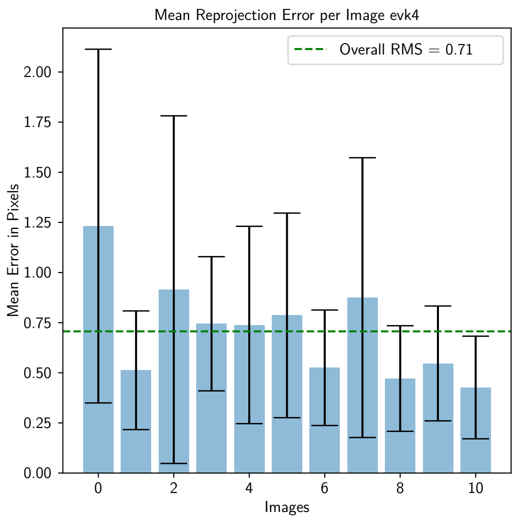
Figure 4.9: Two same LED light sources at 1.0 meters, blinking at 10 Hz and 1 kHz. Fig. 4.9a shows a visible diffraction star (while being much brighter), while Fig. 4.9b shows a much more cicular source of light that is not as bright.

## 4.2 Calibration results

The camera calibration was performed on a series of images, where the calibration lattice was placed as various angles and distances from the camera. For ideal calibration results, the lattice should be placed in all visible parts of the image, as the distortion of the fisheye is more pronounced at the edges of the visible area. The calibration was performed on a polynomial of degree 4, more would lead to overfitting and is also not necessary. The calibration results can be seen in Fig. 4.10.



(a) Calibrated lens model function



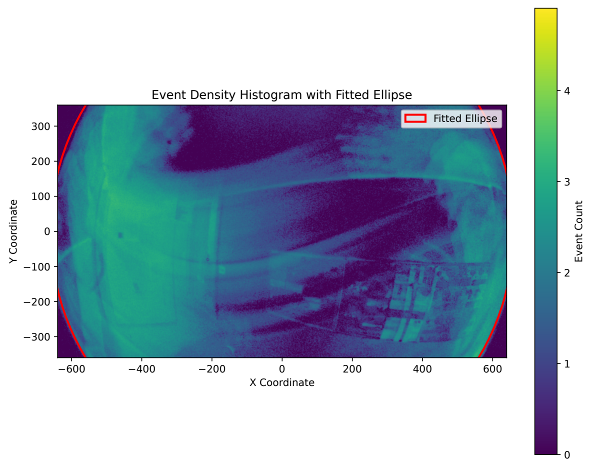
(b) Calibration images mean reprojection errors

Figure 4.10: Calibration results with the calibrated lens model function highlighted in red in Fig. 4.10a and the calibration images mean reprojection errors on Fig. 4.10b.

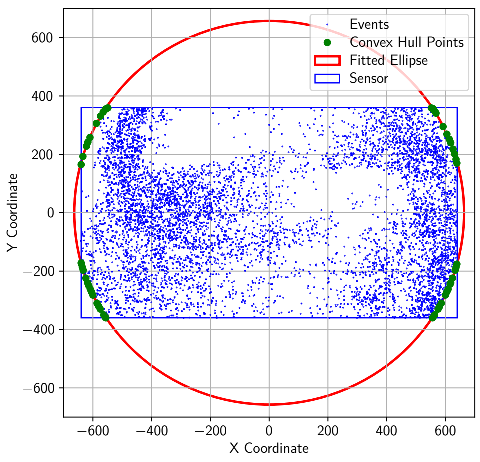
The problem of fitting a minimal encompassing ellipse to event data, to obtain the visible area of the camera sensor, can be formulated as an unconstrained optimization problem 4.4, where the squared distances of the convex hull points to the ellipse boundary are minimized. The optimized ellipse can be seen in Fig. 4.11.

$$(a^*, b^*) = \underset{a,b>0}{\operatorname{argmin}} \sum_{(x_i,y_i)\in\mathcal{H}} \left(\frac{x_i^2}{a^2} + \frac{y_i^2}{b^2} - 1\right)^2 \tag{4.4}$$

where $\mathcal{H} = \text{Conv}\{(x_i, y_i)\}_{i=1}^n$ is the set of points in the convex hull of generated events.



(a) Fitted ellipse.



(b) Events with convex hull points

Figure 4.11: Fitted ellipse to the calibration data, with semi-major axis $a^* = 663$, and semi-minor axis $b^* = 657$ on Fig. 4.11a with separated convex hull points in green on Fig. 4.11b.

Finally, to visualize the calibration results, every point from the image plane is mapped using the `cam2world`[5] function from `py-OCamCalib`, which takes a 2D image point, and returns the corresponding 3D optical ray on the camera's unit sphere. For each point, its angle from the optical axis (a vector $\mathbf{v} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$) is calculated and the visible area is masked out with the ellipse fitted in Fig. 4.11. The results can be seen in Fig. 4.12.

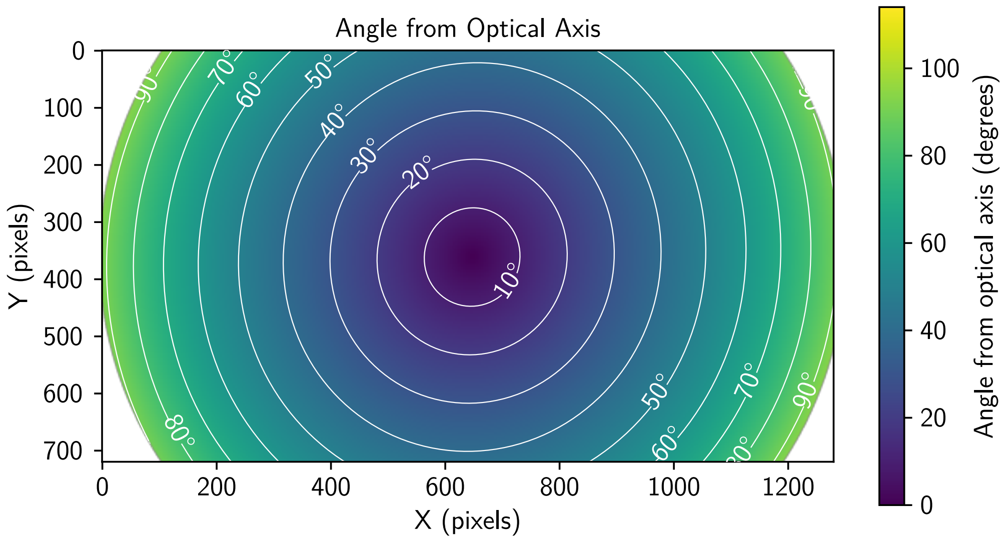
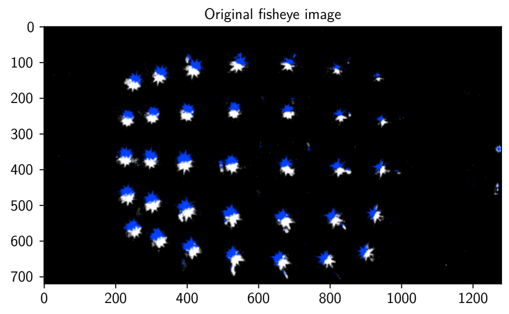


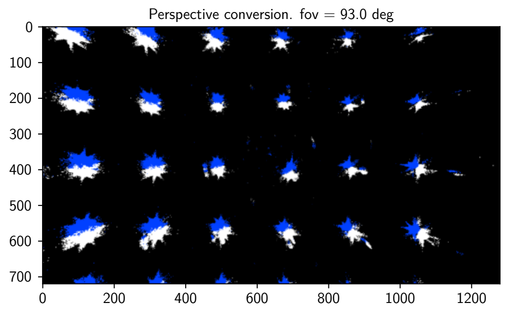
Figure 4.12: Angle from optical axis visualization, with the maximum angle of 93.47 degrees.

[5] https://github.com/jakarto3d/py-OCamCalib/blob/main/src/pyocamcalib/modelling/camera.py

A perspective conversion can also be applied to the whole image, which now correctly represents distances and angles. This can be noticed by looking at the calibration lattice at Fig. 4.13, which now looks like a grid of points.



(a) Uncalibrated image.



(b) Calibrated image.

Figure 4.13: Two photos of the calibration lattice, one uncalibrated on Fig. 4.13a and the other calibrated at Fig. 4.13b, which does not exhibit any distortion.

For the calibration of the Entaniya 280 degree lens a classical chessboard target was used, as the precise localization of the blob centers proved to be nearly impossible while using the LED lattice calibration target. The corners of the chessboard pattern provide better contrast and allow for precise localization of the center; unfortunately, they need to be labeled manually in this case as seen in Fig. 4.14. The calibration results can be seen on Fig. 4.15, with the visualization on Fig. 4.16 and the perspective projection on Fig. 4.17.

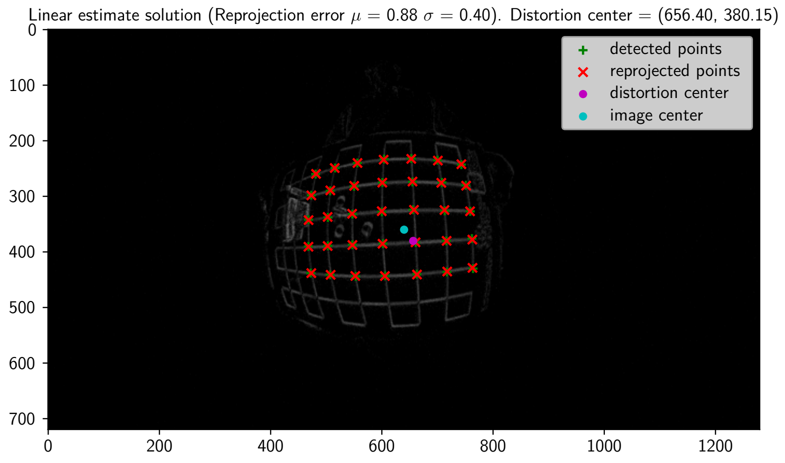
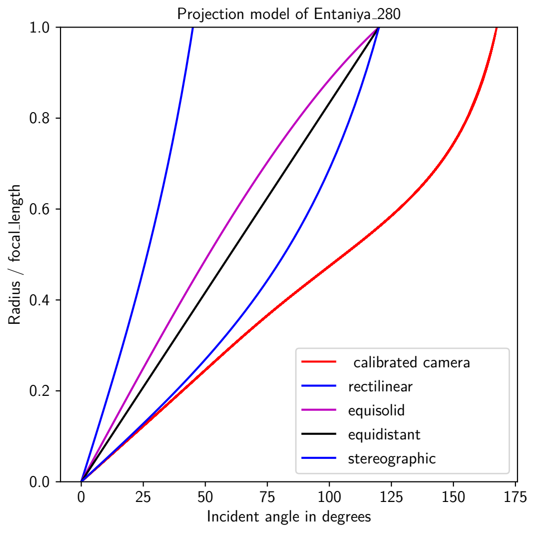
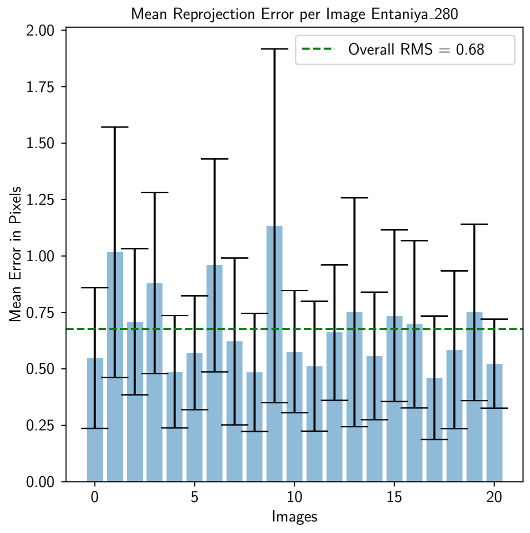


Figure 4.14: Chessboard calibration target visible in 280 degree lens with marked chessboard corner points



(a) Calibrated lens model function



(b) Calibration images mean reprojection errors

Figure 4.15: Calibration results for the Entaniya 280 degree lens with the calibrated lens model function highlighted in red in Fig. 4.15a and the calibration images mean reprojection errors on Fig. 4.15b.

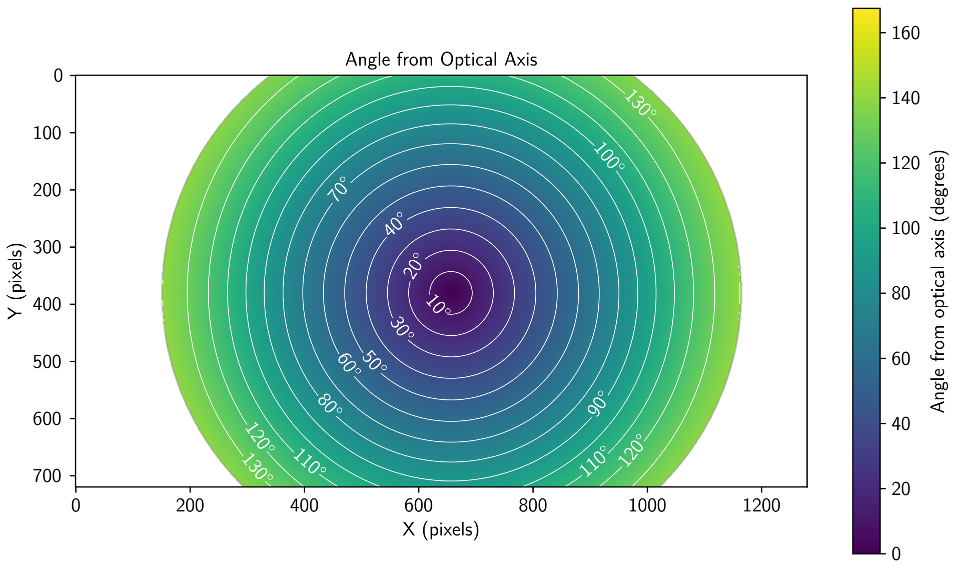
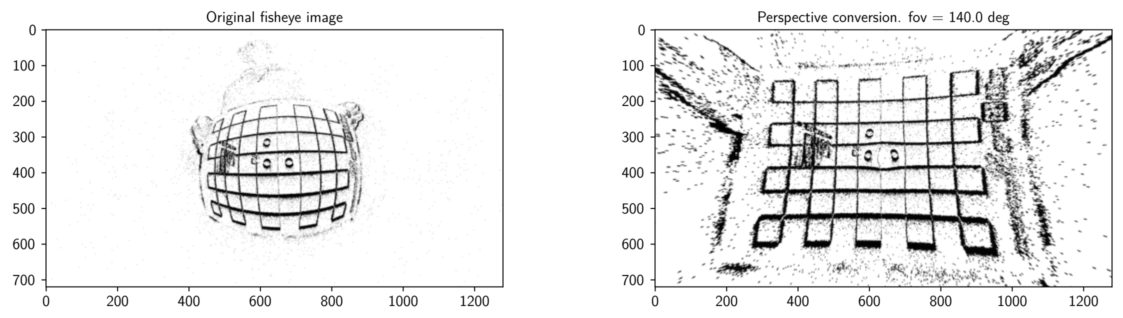


Figure 4.16: Angle from optical axis visualization, with the maximum angle of 140 degrees on each side, making its FOV 280 degrees.



(a) Uncalibrated image. (b) Calibrated image.

Figure 4.17: Two photos of the calibration chessboard from the Entaniya 280 degree lens, one uncalibrated on Fig. 4.17a and the one calibrated at Fig. 4.17a.

## 4.3 Stationary experiments results

For the stationary experiment a simple PnP (P3P) estimation was performed with a calibrated camera, the ground truth data being the measured distance from the camera to the stationary UAV placed on the ground. The results can be seen on Fig. 4.18, with several recordings present for each distance. The mean distance estimation error was 0.34 meters with a standard deviation of 0.16 meters.

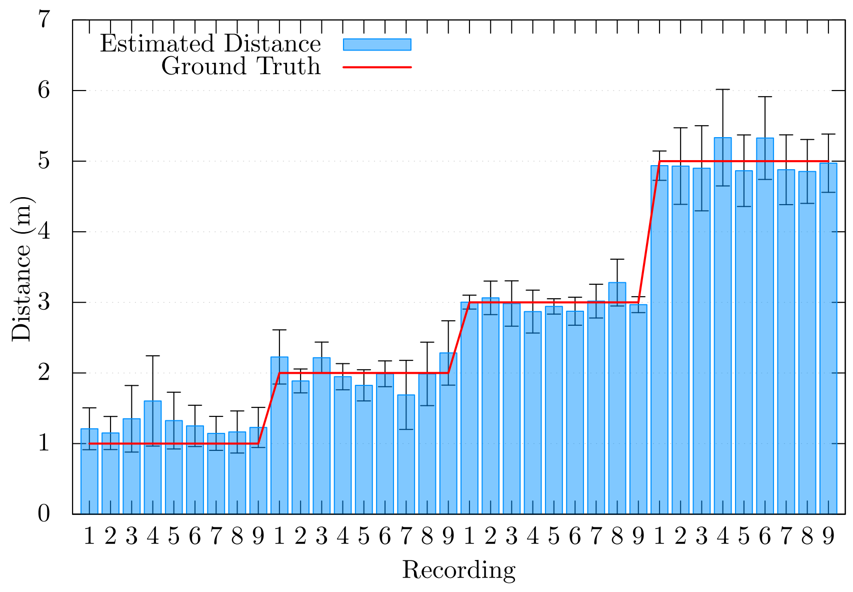


Figure 4.18: PnP estimation results

## 4.4 Real-world experiment results

The data was analyzed after the recording, by generating image frames from the raw event recordings, which were synchronized to the recorded data in the ROS bags. The generated images, as seen on Fig. 4.19, have then been manually labeled and with the use of a blob detector the LED sources were selected. For each of the selected blobs, a center was then selected by calculating the closest pixel to the blob's centroid.

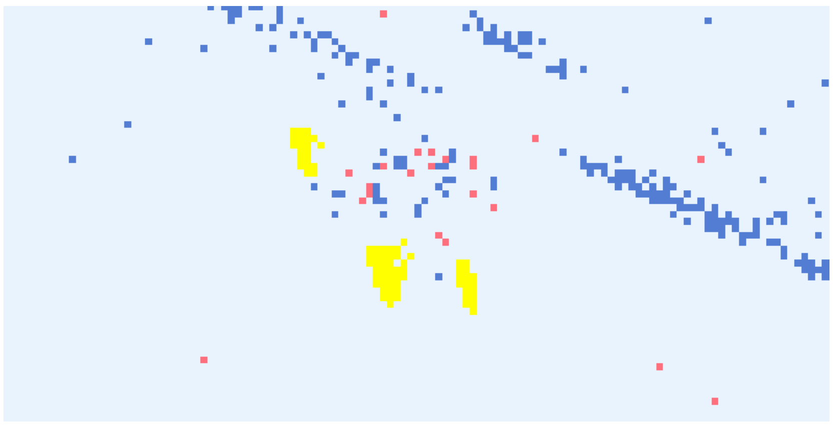


Figure 4.19: Labeled blob centers

For each group of labeled points, a distance estimation was performed with the `DistanceEstimator` ROS node using the P3P algorithm. For each estimated pose, a distance from the camera was calculated, which was then compared to the ground truth distance obtained from the GNSS. The distance is calculated as a norm of the difference of the UAV positions. The results demonstrate that our approach achieves a mean absolute error of 2.47 meters, with a standard deviation of 1.75 meters, indicating moderate precision under the experiment conditions. The distance estimation results can be seen on Fig. 4.20 with the estimated pose results visible on Fig. 4.21.

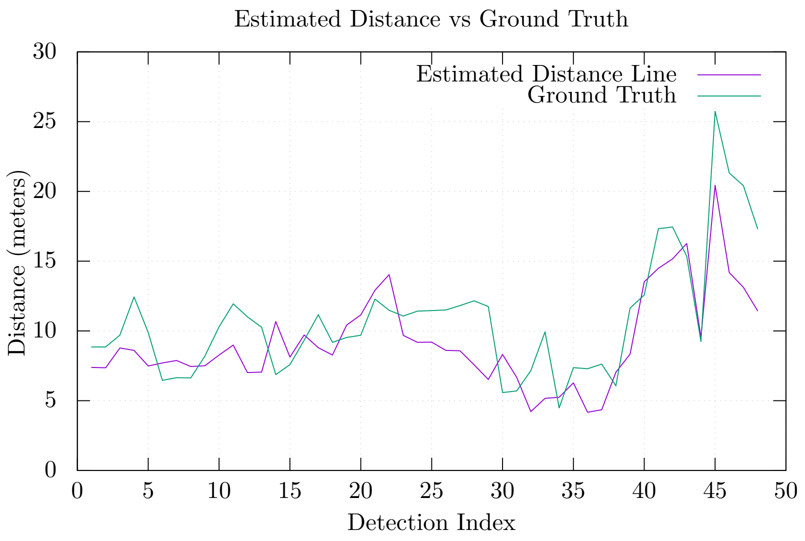
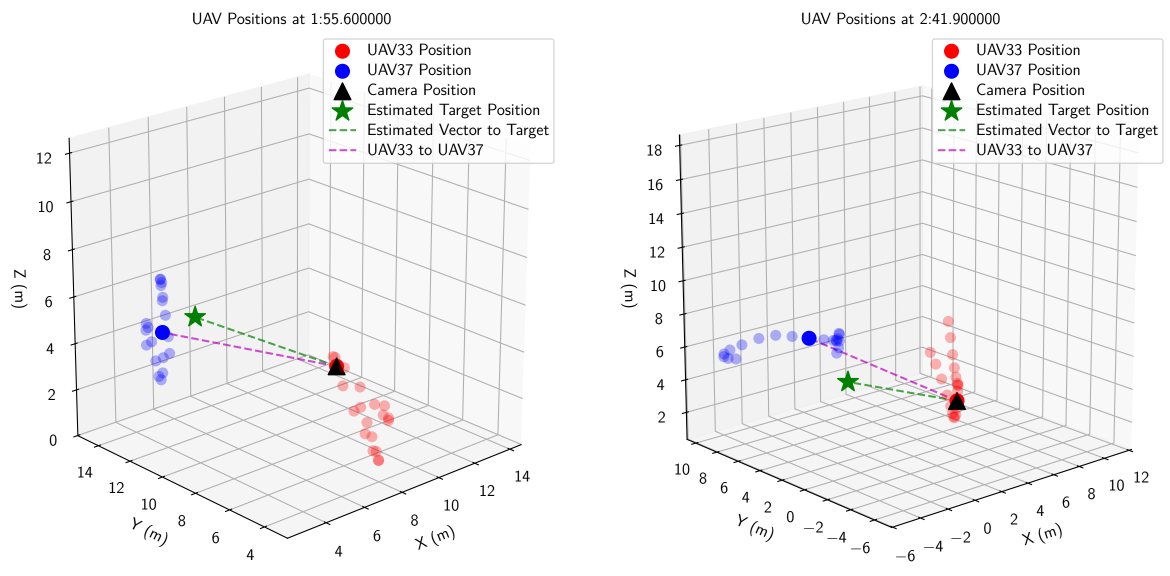


Figure 4.20: The distance estimation data from the experiment, compared with the recorded GNSS data used as ground truth



(a) Fist pose estimation

(b) Second pose estimation

Figure 4.21: The estimated poses of UAV37 at two different timestamps at Fig. 4.21a and Fig. 4.21b, with the UAV trajectories for UAV33 highlighted in red and for UAV37 highlighted in blue.

## 4.5 Real-world deployment - estimation challenges

The analysis presents several challenges, one of them being the identification of the moving UAV in the recorded data. The ideal solution is to identify which pixels are being generated with their specific frequeny, which the LED markers are set to blink on the. Sadly, during the measurement of our experiment, the UAV produced a lot of vibration in the data, which interfered with our measurements, and thus the detection of blobs has become a large obsacle. When images are generated from the recording, the problem persists; How to identify the correct blobs automatically, when the UAV can be at an arbitrary distance and orientation? Manual labeling is not the definitive answer to our problem either because even the blob center selection itself can greatly influence the estimation precision. In Fig. 4.22, a single pixel is selected from each blob and the distance estimation is run for each combination of pixels. As can observed, even a difference in the range of 3 pixels can change the estimated distance by 1.5 meters. The center of a blob may be calculated as a geometrical center of the blob's pixels, but this approach fails when the blob contains pixels which do not belong to them.

A fundamental limitation of any vision-based positioning system is the finite spatial resolution of the camera sensor. As the distance to the LEDs increases, beyond roughly 20 meters in our setup, their angular size on the image plane decreases to just a few pixels. Once a marker spans only a few sensor pixels, adjacent light sources can begin to merge: individual LED spots can no longer be differentiated and their point-spread functions overlap. This blending of pixel responses prevents the reliable separation of each LED signal and ultimately makes accurate detection and decoding impossible.

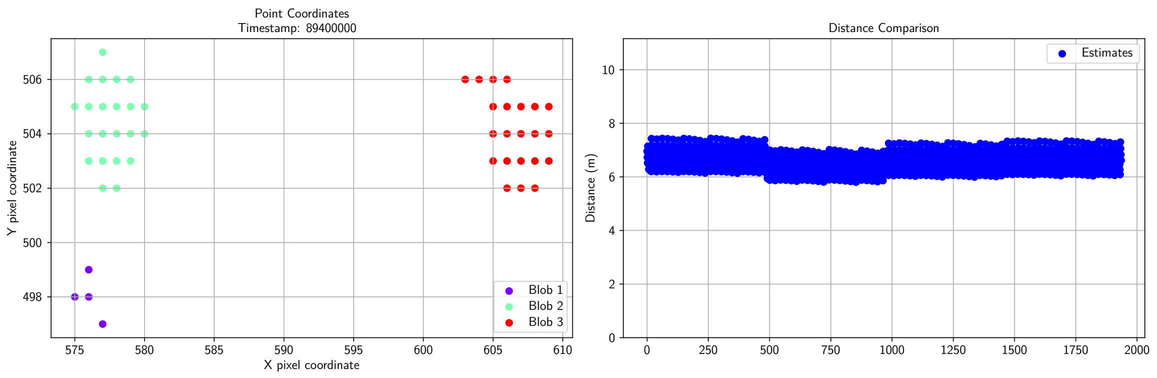


Figure 4.22: Blob centers with the resulting distance estimations for any combination of center selection

# 5 Conclusion

In this thesis, the response of an event-based camera to a modulated source of light (a UV LED) was discussed. It was shown how the changing frequency, distance, and incidence angle influenced the average number of events generated by the camera. The calibration method for fisheye lenses by Scaramuzza et al. [9] was then discussed, which, in our case, utilized an LED lattice target instead of the normally used chessboard image pattern.

Distance estimation was performed using the P3P algorithm, which estimated the rotation and translation of a known arrangement of 3D marker locations using their 2D image coordinates. A stationary data set was collected and analyzed with this approach, which has shown a median distance estimation error of 0.34 meters with a standard deviation of 0.16 meters. A `DistanceEstimator` node was implemented in ROS, which eased the distance estimation by subscribing to detected LED locations and publishing the estimated pose and distance.

This approach was used in our real-life experiment, where data was collected from two flying UAVs, which were manually controlled by two pilots. The data was then analyzed afterward, by manually marking the LED locations. The resulting estimate has shown higher errors (a mean absolute error of 2.47 meters with a standard deviation of 1.75 meters), compared to the statically measured data set, due to the increased relative distance during flight and physical limitations of the methods and equipment used.

## 5.1 Future work

In the future, the detection of the LEDs is planned to be automated, enabling real-time distance and pose estimation. This could be achieved by incorporating a robust feature detection algorithm or a machine learning-based object detection algorithm into the pipeline. A method using RSSR for the improvement of the pose estimation could also be used to further improve pose estimation precision, which was not further explored in this thesis, as for the inconclusive results obtained during the trial analysis of the data measured for this method and the lack of time for further research.

## 6 References

[1] G. Gallego, T. Delbrück, G. Orchard, *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022. DOI: 10.1109/TPAMI.2020.3008413.

[2] V. Walter, M. Saska, and A. Franchi, "Fast mutual relative localization of uavs using ultraviolet led markers," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 1217–1226. DOI: 10.1109/ICUAS.2018.8453331.

[3] S. Shiba, Q. Kong, and N. Kobori, *E-vlc: A real-world dataset for event-based visible light communication and localization*, 2025. arXiv: 2504.18521 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2504.18521.

[4] G. Ebmer, A. Loch, M. N. Vu, *et al.*, *Real-time 6-dof pose estimation by an event-based camera using active led markers*, 2023. arXiv: 2310.16618 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2310.16618.

[5] G. Gou, X. Wang, Y. Ye, *et al.*, "Hybrid depth-event pose estimation for online dense reconstruction in challenging conditions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 223, pp. 328–343, 2025, ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2025.03.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092427162500111X.

[6] Z. Liu, B. Guan, Y. Shang, Q. Yu, and L. Kneip, *Line-based 6-dof object pose estimation and tracking with an event camera*, 2024. arXiv: 2408.03225 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2408.03225.

[7] C. Scheerlinck, N. Barnes, and R. E. Mahony, "Continuous-time intensity estimation using event cameras," *CoRR*, vol. abs/1811.00386, 2018. arXiv: 1811.00386. [Online]. Available: http://arxiv.org/abs/1811.00386.

[8] M. S. Dilmaghani, W. Shariff, C. Ryan, J. Lemley, and P. Corcoran, *Control and evaluation of event cameras output sharpness via bias*, 2022. arXiv: 2210.13929 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2210.13929.

[9] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, 2006, pp. 45–45. DOI: 10.1109/ICVS.2006.3.

[10] S. Urban, J. Leitloff, and S. Hinz, "Improved wide-angle, fisheye and omnidirectional camera calibration," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 108, pp. 72–79, 2015, ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2015.06.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271615001616.

[11] S.-Y. Jung, S. R. Lee, and C.-S. Park, "Indoor location awareness based on received signal strength ratio and time division multiplexing using light-emitting diode light," *Optical Engineering*, vol. 53, no. 1, p. 016 106, 2014. DOI: 10.1117/1.OE.53.1.016106. [Online]. Available: https://doi.org/10.1117/1.OE.53.1.016106.

[12] L. Bai, Y. Yang, C. Feng, C. Guo, and J. Cheng, *A high coverage camera assisted received signal strength ratio algorithm for indoor visible light positioning*, 2020. arXiv: 2004.06294 [eess.SP]. [Online]. Available: https://arxiv.org/abs/2004.06294.

[13] Y. Ding, J. Yang, V. Larsson, C. Olsson, and K. Åström, "Revisiting the p3p problem," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 4872–4880.

[14] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *CVPR 2011*, 2011, pp. 2969–2976. DOI: 10.1109/CVPR.2011.5995464.

[15] D. Hert, T. Baca, P. Petracek, *et al.*, "Mrs modular uav hardware platforms for supporting research in real-world outdoor and indoor environments," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 1264–1273. DOI: 10.1109/ICUAS54217.2022.9836083.

[16] T. Finateu, A. Niwa, D. Matolin, *et al.*, "5.10 a 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86µm pixels, 1.066geps readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 112–114. DOI: 10.1109/ISSCC19947.2020.9063149.

[17] M. Lendermann, J. S. Q. Tan, J. M. Koh, and K. H. Cheong, "Computational Imaging Prediction of Starburst-Effect Diffraction Spikes," *Scientific Reports*, vol. 8, no. 1, p. 16 919, 2018. DOI: 10.1038/s41598-018-34400-z.

# A Attachments

Below is a list of attachments to this thesis.

| | |
|---|---|
| `mrs-uvdar-distance-estimator.zip` | code used in the event-based camera response section of this thesis |
| `metavision-pyocamcalib.zip` | code used for the calibration of lenses, and the generation of calibration frames |
| `ros-event-distance.zip` | ROS implementation of a pose estimator |

All source code related to this thesis is also publicly available in the repositories listed below.

| | |
|---|---|
| **Thesis LaTeX Source** | github.com/kubakubakuba/Bachelor-Thesis |
| **Response Analysis** | github.com/kubakubakuba/mrs-uvdar-distance-estimator |
| **Calibration Scripts** | github.com/kubakubakuba/metavision-pyocamcalib |
| **ROS DistanceEstimator** | github.com/kubakubakuba/ros-event-distance |

# B Used AI Software

In accordance with the `Methodological guideline No. 5/2023` [1] the following software was used during the writing of this thesis:

- Writefull[2] for rewording in the online Overleaf[3] editor
- GitHub Copilot[4] for code completion
- ChatGPT[5] for wording suggestions and feedback, for plotting help

[1] https://www.cvut.cz/sites/default/files/content/d1dc93cd-5894-4521-b799-c7e715d3c59e/en/20231003-methodological-guideline-no-52023.pdf

[2] https://www.writefull.com/

[3] https://www.overleaf.com/

[4] https://github.com/features/copilot

[5] https://chat.openai.com/