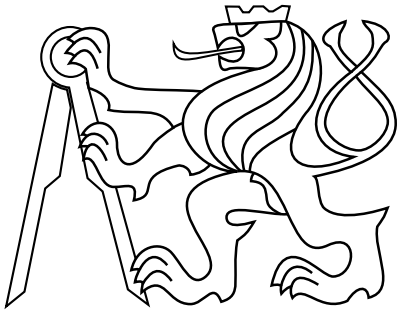
Czech Technical University in Prague

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
Branch of study: Artificial Intelligence and Computer Science

Supervisor: Ing. Vojtěch Vrba

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor.

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.

Declaration

I declare that presented work was developed independently, and that I have listed all sources of information used within, in accordance with the Methodical instructions for observing ethical principles in preparation of university theses.

Date ....

Abstract

The study of autonomous Unmanned Aerial Vehicles (UAVs) has become a prominent sub-field of mobile robotics.

Keywords Unmanned Aerial Vehicles, Event Cameras

Abstrakt

Výzkum na poli autonomních bezpilotních prostředků (UAV) se stal významným oborem mobilní robotiky.

Klíčová slova Bezpilotní Prostředky, Eventové Kamery

Abbreviations

DOF degree-of-freedom

DVS Dynamic Vision Sensors

FOV Field of View

FPS Frames Per Second

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 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. 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, we present a method for estimating the pose of a UAV equipped with Ultra Violet (UV) LED light sources as in the UVDAR system. [11] These LEDs are modulated at select frequencies, which aids in their identification and also helps with their prevalence on the scene observed by the camera. 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 works

Recent advances in relative pose estimation for UAV swarming applications have focused on GNSS-denied environments and the overcoming of limitations the navigation faces in these environments.

Shiba et al. [2] 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. Standard deviations were 16.2 mm and 0.146° for tracking, but increased significantly to 121 mm and 5.12° for detection. 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. [1] proposed a hybrid framework fusing depth-sensor data and event-based camera streams in a joint random-optimization scheme to achieve robust camera tracking and dense reconstruction under fast motion for agile robotics tasks.

1.2 Contributions

TODO: write this if necessary

This section should describe the author's contributions to the field of research.

1.3 Mathematical notation

TODO: write this if necessary

It is a good practice to define basic mathematical notation in the introduction. See Table 1.1 for an example.

$\mathbf{x}$, $\boldsymbol{\alpha}$	vector, pseudo-vector, or tuple
$\hat{\mathbf{x}}$, $\hat{\boldsymbol{\omega}}$	unit vector or unit pseudo-vector
$\mathbf{X}, \boldsymbol{\Omega}$	matrix
$\mathbf{I}$	identity matrix
$\mathbf{R}$	rotation matrix
$\mathbf{t}$	translation vector
$SO(3)$	3D special orthogonal group of rotations
$\delta(t)$	Dirac delta function

Table 1.1: Mathematical notation, nomenclature and notable symbols.

2 Response of an event-based camera

2.1 Event-based cameras compared to frame-based cameras

Traditional frame-based (sometimes called 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) [10]

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

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 ϵ is the exposure time. As we can see, each frame is represented by a temporal average of irradiance over the exposure time ϵ. 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 nature bio-receptors, where each pixel reacts to the change of illumination in the scene. Each pixel individually recognizes the log 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, σ is the polarity of change (where $\sigma = \pm 1$ for increasing or decreasing change, respectively) and t is the timestamp of the event. [6] [10] We can model the single event as a Dirac-delta function $\delta(t)$ and can define an event stream $e_i(\boldsymbol{p},t)$ at a pixel $\boldsymbol{p}$ by (2.2) [10]

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

where $\sigma_i^{\boldsymbol{p}}$ is the polarity (sometimes referred simply as an ON or OFF event) and $t_i^{\boldsymbol{p}}$ is the time-stamp of the i-th event at a pixel. The magnitude c is the contrast threshold, a preset constant (similar to exposure time in frame-based cameras), which 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 (images can be reconstructed from the event stream by simply integrating the events over time, making the usage of normal vision algorithms possible, but it also goes against the main advantage of event-based cameras) and the higher cost of the camera units themselves. [6]

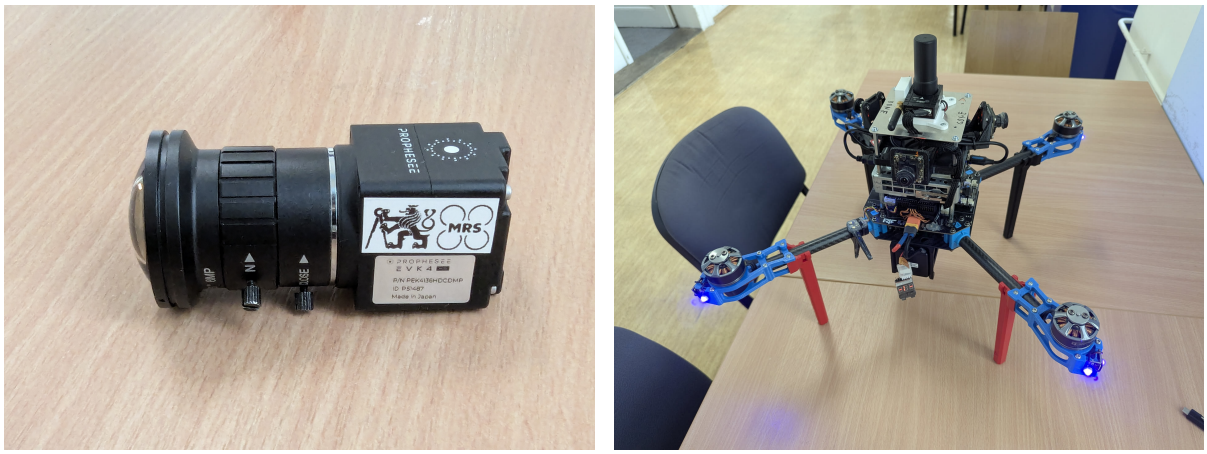
2.2 Equipment used

UAVs

The experimental platform for this work is the MRS X500[7] quadrotor UAV equipped with UV LEDs integrated to the UVDAR[11] system. Each of the UAV's arms is 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 can 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, we use this functionality to differentiate between the arms by modulating each arm on a different frequency to be easily distinguishable. The UAV can be seen in Fig. 2.1b.

Event-based camera

The event-based camera used in this thesis is the `Prophesee EVK4 HD`[1][8], with `IMX636` sensor. The camera features a resolution of 1280×720 pixels and is capable of generating 1.066×10^9 events per second (equivalent to 10.000 Frames Per Second (FPS) in conventional frame-based terms) and offers a dynamic range of 120 dB. During recording, many camera settings can be changed, such as the Region of Interest (ROI) settings and the bias settings.



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

(b) X500 UAV unit equipped with UVDAR

Figure 2.1: An event-based camera with a 2.5mm fish eye lens can be seen on Fig. 2.1a, which was used to measure the UV LEDs mounted on the X500 UAV unit modulated using UVDAR as seen on Fig. 2.1b.

ROI

The ROI setting during recording takes a rectangular region in pixel coordinates and discards events outside the specified region at the hardware level, 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, that would interfere with the measurement process.

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

Bias settings

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[5]

- `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 σ
- `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 σ
- `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 ensures that only events related to the experiment are captured.

Lenses

During the measurements, a 2.5mm f/1.6 fish eye lens with a Field of View (FOV) of 93.5 degrees was used. Subsequently, during the final experiment, an ultra-wide 1.07mm f/2.8 fish eye lens from Entaniya with FOV of 280 degrees was used in combination with the first lens. Both lenses were equipped with narrowband UV filters which target the specific wavelength of the LEDs that are used in the UVDAR localization system. This ensures, that the majority of generated events come from the LED sources mounted at the UAVs, and less events come from the surrounding area. All lenses were properly calibrated (see chapter 3), which is required to provide correct results in the final pose estimation.

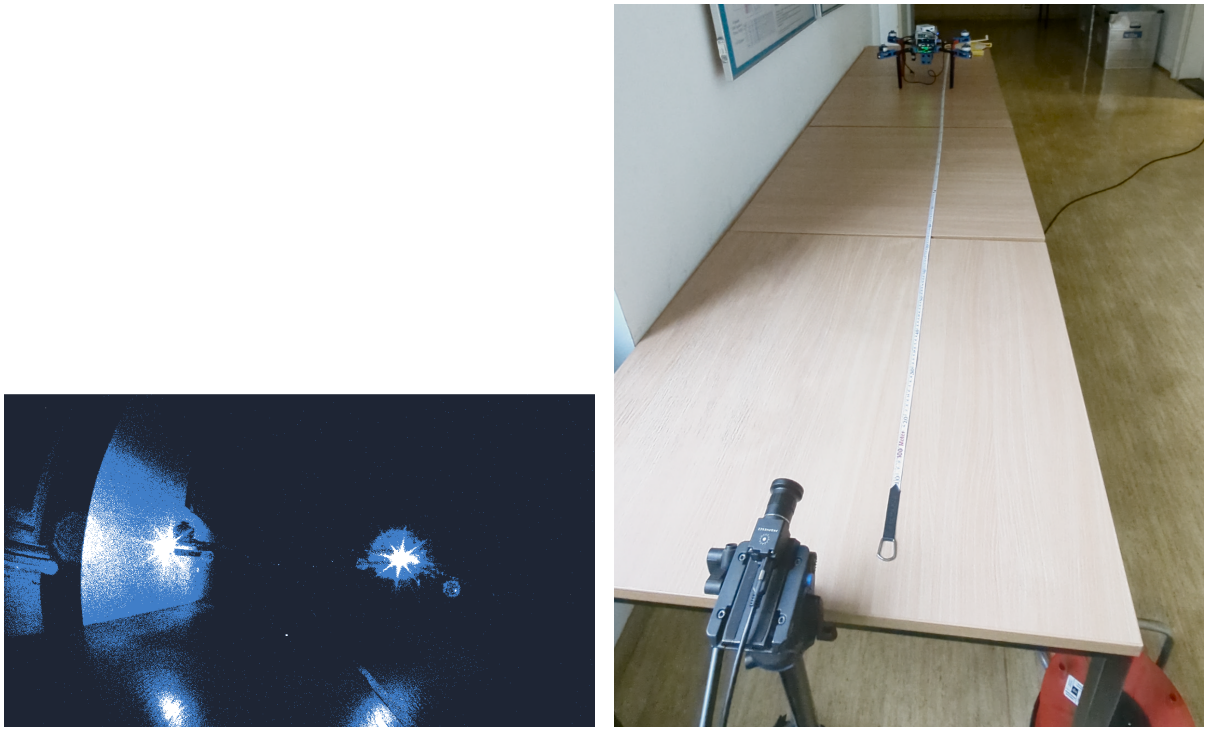
2.3 Data collection

Initial measurements

The baseline experiment involved a static event 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 captures 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, but 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 are working as event-generating light sources, bouncing the light around, as seen on Fig. 2.2. 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 need to be done to analyze the recorded data and investigate the relations of measured effects, for example a local ROI filter could be applied to LED centers detected by a blob detection algorithm.



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

(b) View of the experiment setup.

Figure 2.2: The setup for measuring the event-camera response with a EVK4 camera. Visible reflections from a wall can be seen on Fig. 2.2. The setup is shown on Fig. 2.2b.

- **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 and 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 also needs to be considered, 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`.

2.4 Event response data processing

The event-based camera response data was analyzed using the Metavision SDK[2] 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 $\mathcal{F}$ [3]. The tested ranges were:

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

$$\mathcal{D} = \{1.0 + 0.2k \mid k \in \{0, \ldots, 10\}\} \cup \{4.0, 5.0\}\,\text{m}.$$

We can load the obtained data set into a matrix representing the distances and frequencies, then load a select number of events from each recording. The data is then resampled into a signal, represented by a 1D array obtained from summing polarities p_i over a selected bin width Δt [4] (2.3).

$$S[k] = \sum_{i \in \mathcal{B}_k} p_i, \quad \mathcal{B}_k = \{i | t_k \leq t_i < t_k + \Delta t\} \tag{2.3}$$

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. We can see the influence of distance and frequency on the average number of events in Fig. 2.3 and Fig. 2.4, 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. With an increasing frequency, the camera cannot capture all the changes that are generated by the light source, leading to 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. 2.3 with a frequency of 30 kHz at 3 meters.

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

[3] 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]`).

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

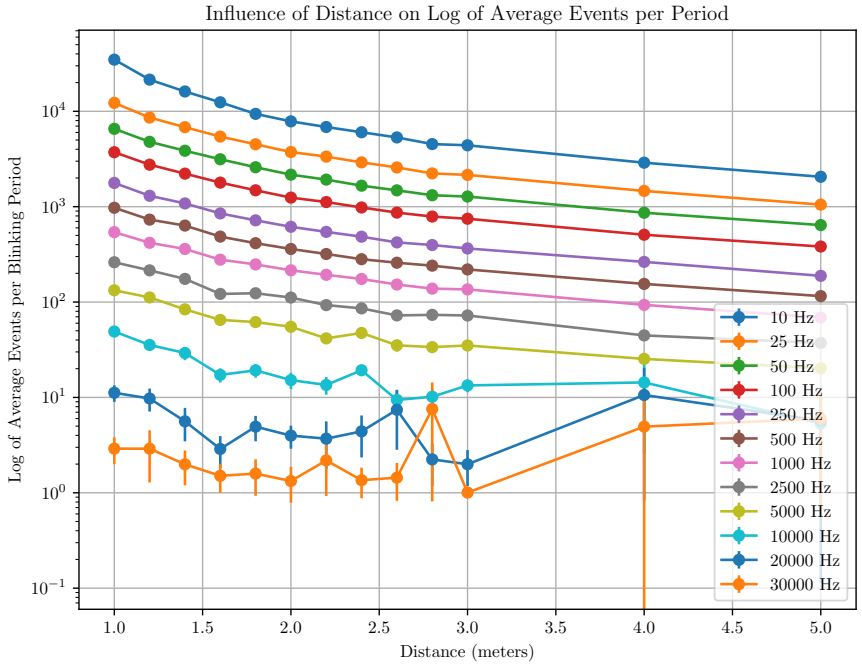


Figure 2.3: Influence of distance on the log of the average number of events, with the UAV rotated 0 degrees relative to the event-based camera.

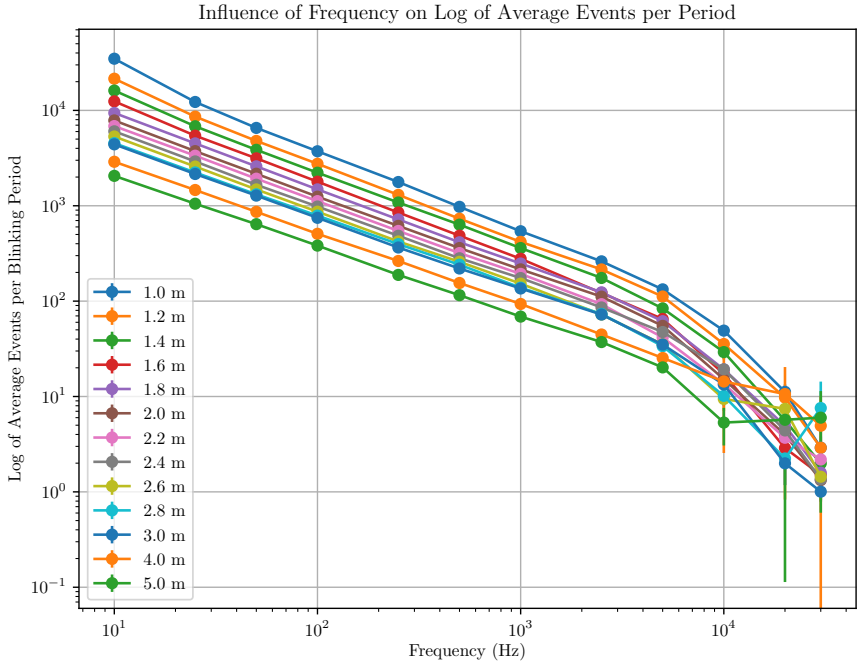


Figure 2.4: Influence of frequency on the log of the average number of events, with the UAV rotated 0 degrees relative to the event-based camera.

If we now select one frequency and try to fit it with a curve, we can observe that the data can be approximated with a rational or an exponential function, as shown in Fig. 2.5. The best fit without being too complex is the inverse square law, which can be expressed as

$$\text{intensity} \propto \frac{1}{\text{distance}^2} \tag{2.4}$$

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.

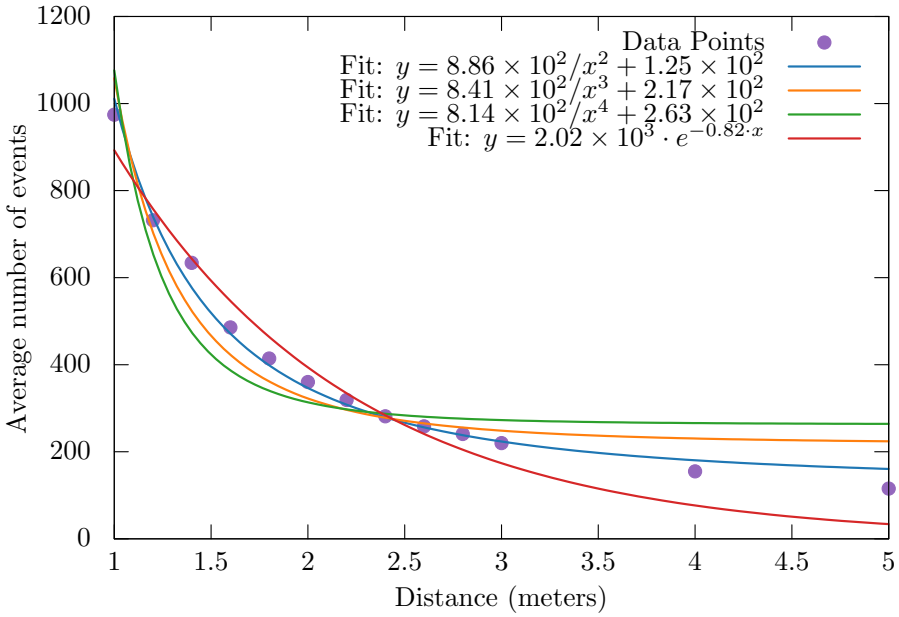


Figure 2.5: Influence of distance data fitted with various curves.

Rotation angle influence

From the manufacturers datasheet of the used ProLight PM2B-1LLE 1W UV Power LED [5] used in the UVDAR system, we can learn that the LEDs have a Lambertian radiation pattern, which can be seen on Fig. 2.6.

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

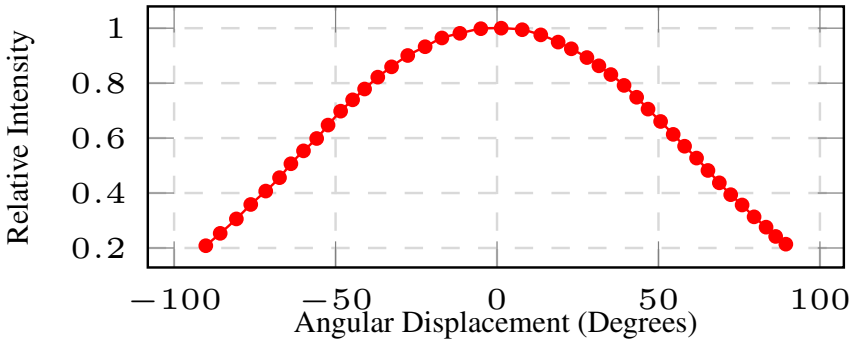


Figure 2.6: Lambertian radiation pattern of the PM2B-1LLE UV LED.

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 (2.5).

$$I(\theta) = I_0 \cos(\theta) \tag{2.5}$$

To represent the whole end of the UAV arm, we need to consider two LED sources, 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. 2.7. The results extracted from the dataset of the UAV rotations relative to the camera are shown in Fig. 2.8.

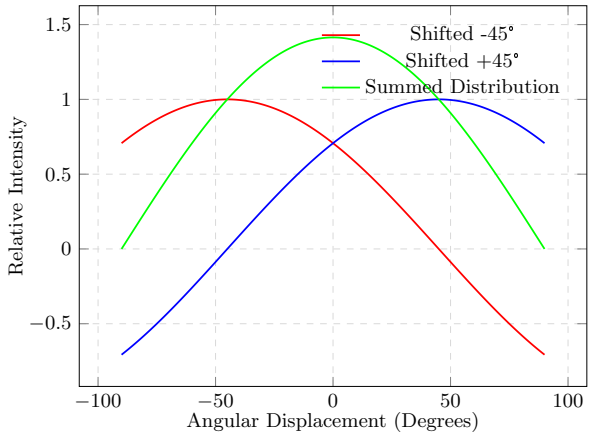
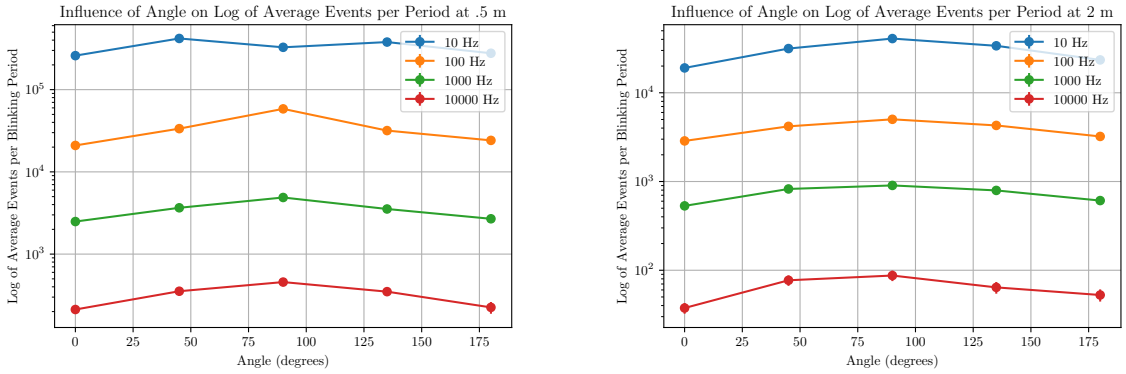


Figure 2.7: Radiation pattern of two lambertian light sources shifted by ±45 degrees.

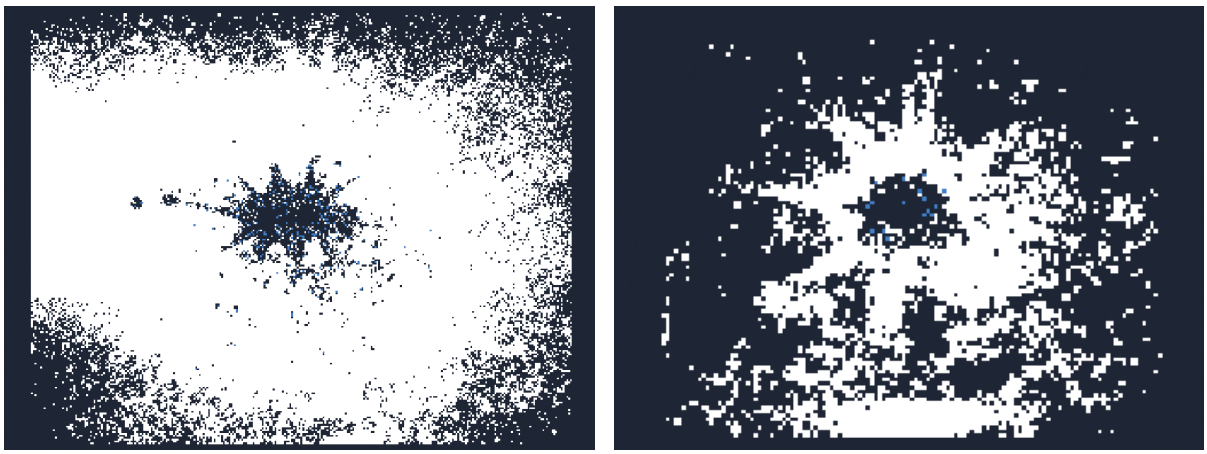


(a) Influence of rotation of the UAV on the log of average number of events at 0.5 m.

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

Figure 2.8: The influence of rotation angle on the log of average number of events at 0.5 m on Fig. 2.8a and at 2 m on Fig. 2.8b.

The data show a rough approximation of the theoretical distribution on Fig. 2.7, 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. 2.9.

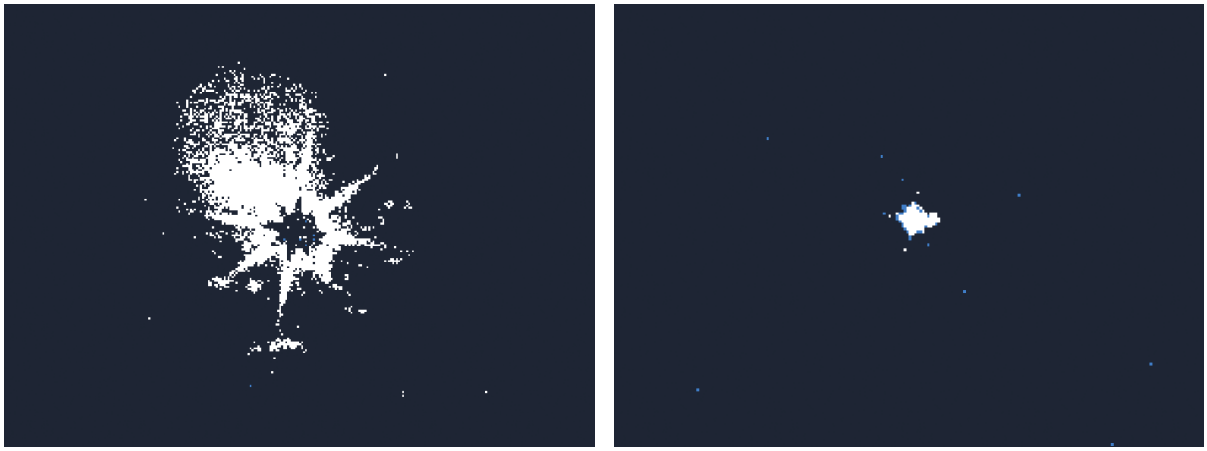


(a) 2 LEDs with blinking frequency of 10 Hz at 0.5 m.

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

Figure 2.9: 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. 2.9a and 2 m at Fig. 2.9b.

What we can also observe from Fig. 2.9 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 depend on the number of blades, the set aperture and the light source intensity then causes stars of different levels of profoundness.[9] We can observe this by comparing how profound the star shapes are on different frequencies, as shown on Fig. 2.10.



(a) LED blinking at 10 Hz at 1.0 m

(b) LED blinking at 1 kHz at 1.0 m

Figure 2.10: Two same LED light sources at 1.0 meters, blinking at 10 Hz and 1 kHz. Fig. 2.10a shows a visible diffraction star (while being much brighter), while Fig. 2.10b shows a much more cicular source of light that is not as bright.

3 Camera calibration

3.1 Method used

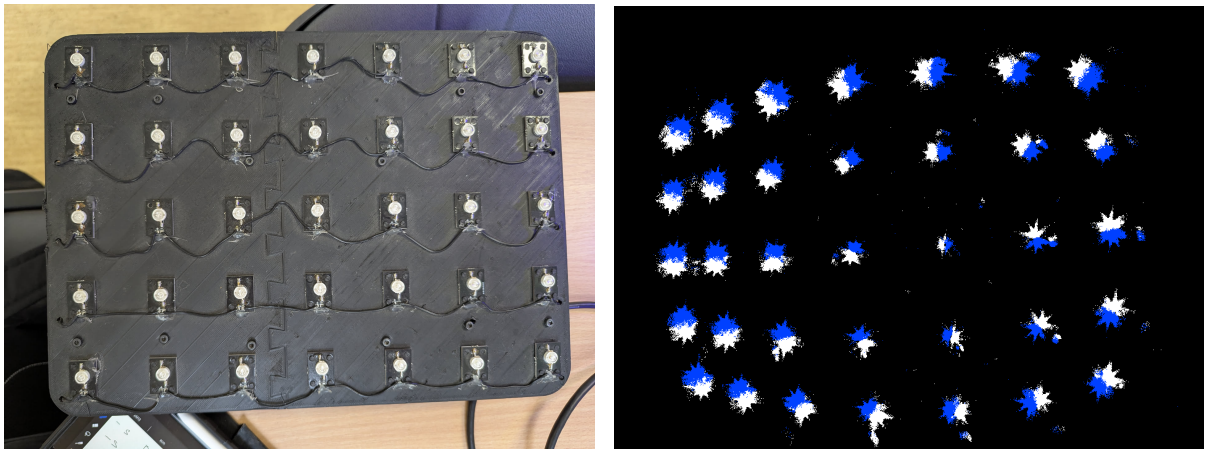
For the correct representation of distances and angles in the received data from the camera, a camera calibration with the used lenses needs to be performed. The lenses used to obtain the data were a 2.5mm f/1.6 fisheye lens, with a 93.5 degrees FOV and Entaniya 1.07mm f/2.8 fish eye lens with a FOV of 280 degrees, both can be seen in Fig. 3.1.

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

Figure 3.1: Lenses used during the calibration, a 93.5 degree lens in Fig. 3.1a and Entaniya 280 degree lens in Fig. 3.1b.

In this work a calibration method proposed by Scaramuzza et al. [15] is used to calibrate all used equipment. The implementation used for calibration is a Python library py-OCamCalib [1].For calibration purposes, a calibration chessboard pattern is usually used, which offers high contrast between squares (and thus easy detections of square corners) and a known square size. Multiple images are usually taken, at various rotations and distances, to obtain good calibration results. In our calibration procedure, we use a 5×7 lattice of UV LEDs, which are 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 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. 3.2.

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



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

Figure 3.2: Calibration lattice of 5×7 UV LEDs on Fig. 3.2a and the events being produced when placed in front of the camera at Fig. 3.2b, 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 fish eye lens sometimes does not allow reliable detection of the exact blob centers (or which events correspond to which LED), as can be seen at Fig. 3.3.

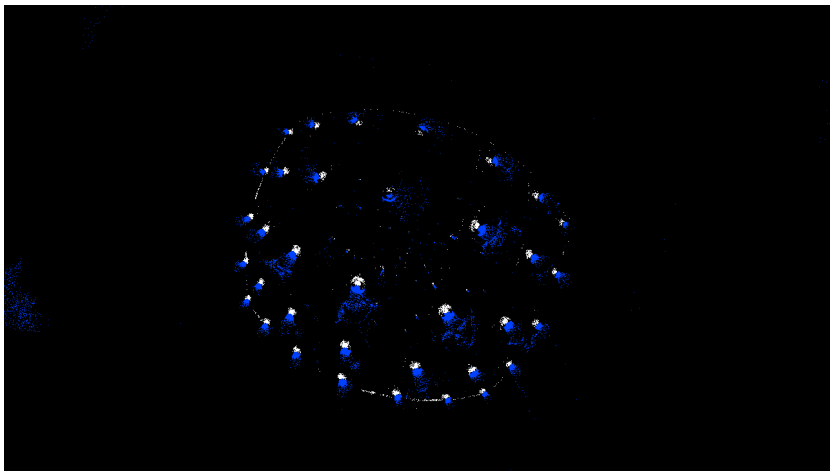


Figure 3.3: Calibration pattern with 5×7 lattice of UV LEDs, taken with a lens with Entaniya lens with FOV of 280 degrees.

As we are not using a regular image of a chessboard pattern, rather a recording of accumulated events, some data preprocessing has to be done beforehand. For the purpose of this, a simple Python app has been written. An event raw recording is loaded, and events are accumulated over a select period of time. The accumulated events are then saved to a grayscale image, where each pixel corresponds to the number of events that occurred in that pixel [2]. The image is then normalized, and the LED centers are detected using the `findContours` function of OpenCV [3]. The detected centers can then be manually labeled row-wise, in the same order

[2] https://github.com/kubakubakuba/metavision-pyocamcalib/blob/main/generate_frames.py
[3] https://github.com/kubakubakuba/metavision-pyocamcalib/blob/main/detect_blob_centers.py

in every image, so the calibration process can identify how the grid of points changes across the images, and thus calculate the lens distortion. After this, the regular calibration script from py-OCamCalib can be used. The labeling of the LED centers can be seen in Fig. 3.4.

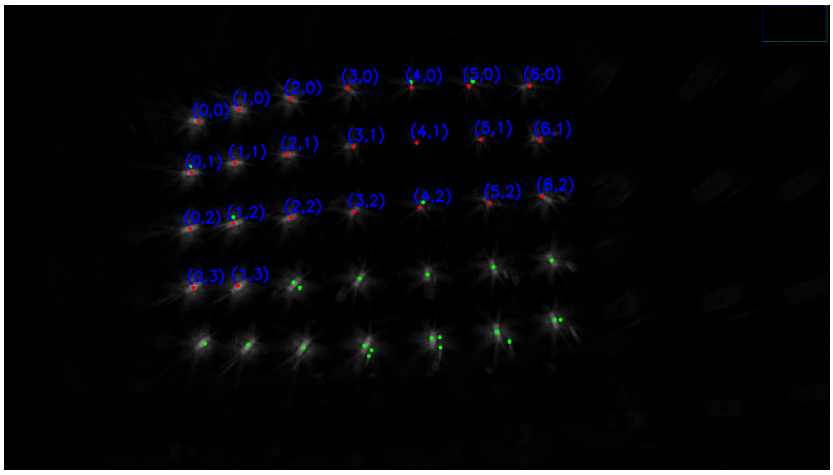


Figure 3.4: Calibration pattern with 5x7 lattice of UV LEDs, with the centers of the LEDs being labeled.

3.2 Principles of the calibration

After the calibration is performed, we are able to map the 3D world coordinates to the 2D image plane. 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 3.1,

$$\begin{bmatrix} X_{camera} \\ Y_{camera} \\ Z_{camera} \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} + \mathbf{t} \tag{3.1}$$

which is an affine transformation that uses a rotation matrix R and a translation vector t. 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. 3.5.

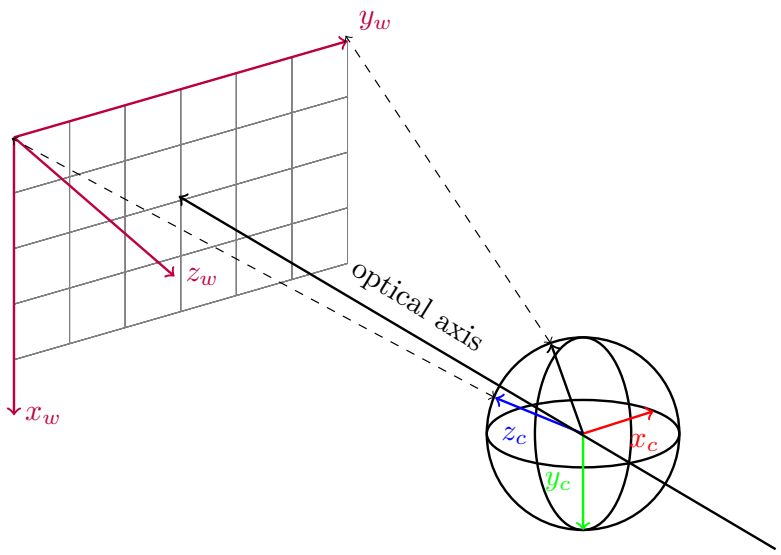


Figure 3.5: Extrinsic parameters of the camera.

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 fish eye lens, as well as the stretching of the image on both the x and y axes. This process can be seen in Fig. 3.6. This ellipse is then used to calculate the point-mapping functions, which allows for tranformation of points from the image plane to their respective 3D coordinates.

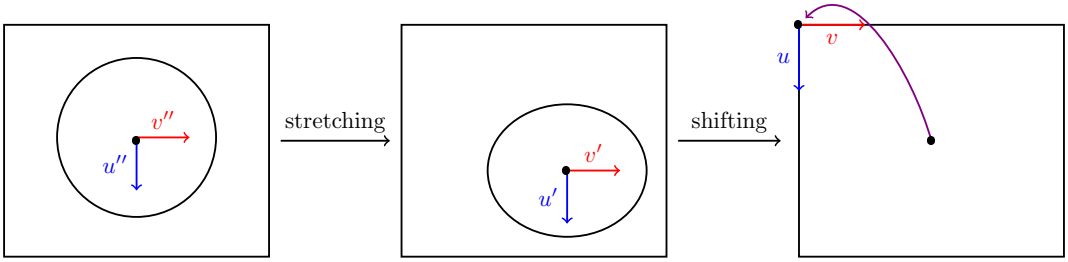


Figure 3.6

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. [15] proposed fitting a polynomial to find the optimal model for lens calibration. The mapping functions are shown in Fig. 3.7.

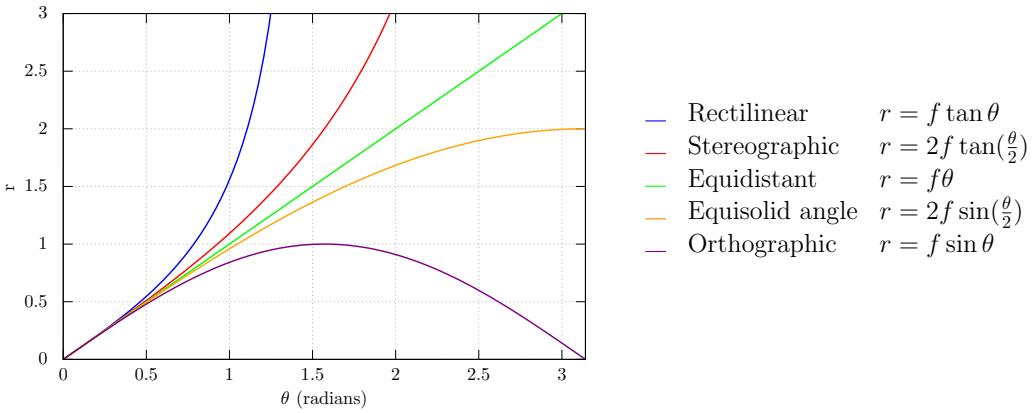


Figure 3.7: Fisheye mapping functions, f is a parameter (focal length).

After we fit a polynomial, we can map the image points to their corresponding 3D vectors by an equation 3.2.

$$\lambda \cdot \alpha \cdot \begin{bmatrix} u' \\ v' \\ a_0 + a_1\rho + \cdots + a_N\rho^N \end{bmatrix} = \mathbf{P} \cdot \mathbf{X_w} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.2}$$

where:

λ is a scalar factor

α is a scaling factor that we obtain from Fig. 3.6 by stretching the ellipse back to a circle and performing the affine transformation; $\alpha, \lambda > 0$

ρ 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.

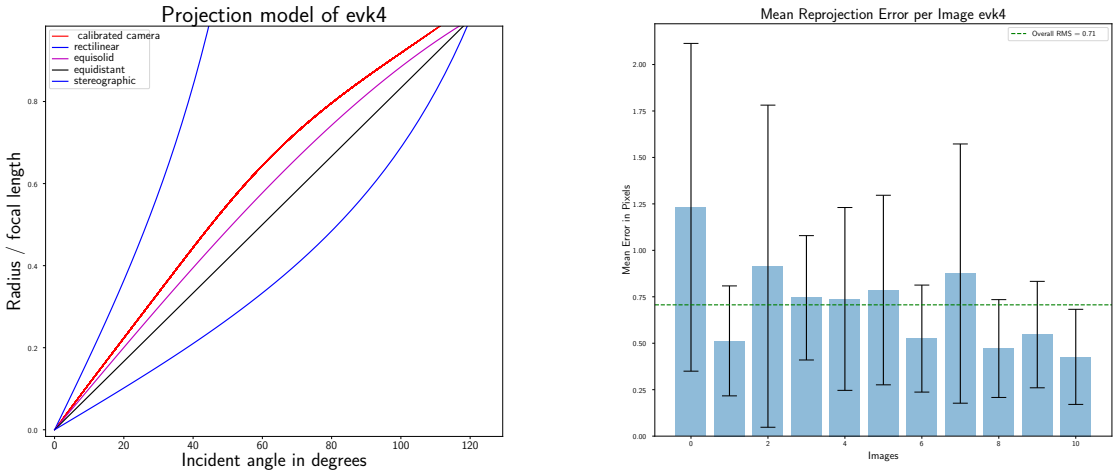
We can also write another relation between a point $\mathbf{m'} = \begin{bmatrix} u' & v' \end{bmatrix}^T$ in the image plane and its corresponding point on the sensor plane $\mathbf{m} = \begin{bmatrix} u & v \end{bmatrix}^T$. The coordinates of $\mathbf{m'}$ have the origin at the top left corner of 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 $\mathbf{m'} = \mathbf{Am} + \mathbf{O}_c$, on 3.3 [12].

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} o_u \\ o_v \end{bmatrix} \tag{3.3}$$

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

3.3 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 fish eye 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. 3.8.



(a) Calibrated lens model function

(b) Calibration images mean reprojection errors

Figure 3.8: Calibration results with the calibrated lens model function highlighted in red in Fig. 3.8a and the calibration images mean reprojection errors on Fig. 3.8b.

We can now fit the encompassing ellipse, which can be formulated as an optimization problem (3.4) of minimizing the sum of point distances, that lie outside of the ellipse, while minimizing the ellipse parameters a, b. The optimized ellipse can be seen in Fig. 3.9.

$$(a^*, b^*) = \underset{a,b>0}{\operatorname{argmin}} \sum_{i=1}^{n} \max\left(\frac{x_i^2}{a^2} + \frac{y_i^2}{b^2} - 1, 0\right). \tag{3.4}$$

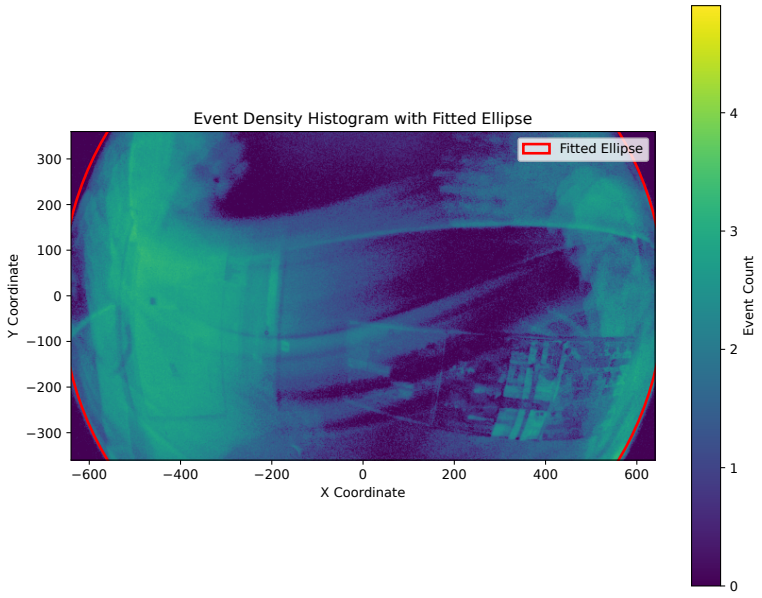


Figure 3.9: Fitted ellipse to the calibration data, with semi-major axis $a^* = 652$, and semi-minor axis $b^* = 650$.

Finally, to visualize the calibration results, we map every point from the image plane using the `cam2world`[4] 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, we calculate its angle from the optical axis (a vector $\mathbf{v} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$), and mask out the visible area with the ellipse fitted in Fig. 3.9. We can see the results in Fig. 3.10.

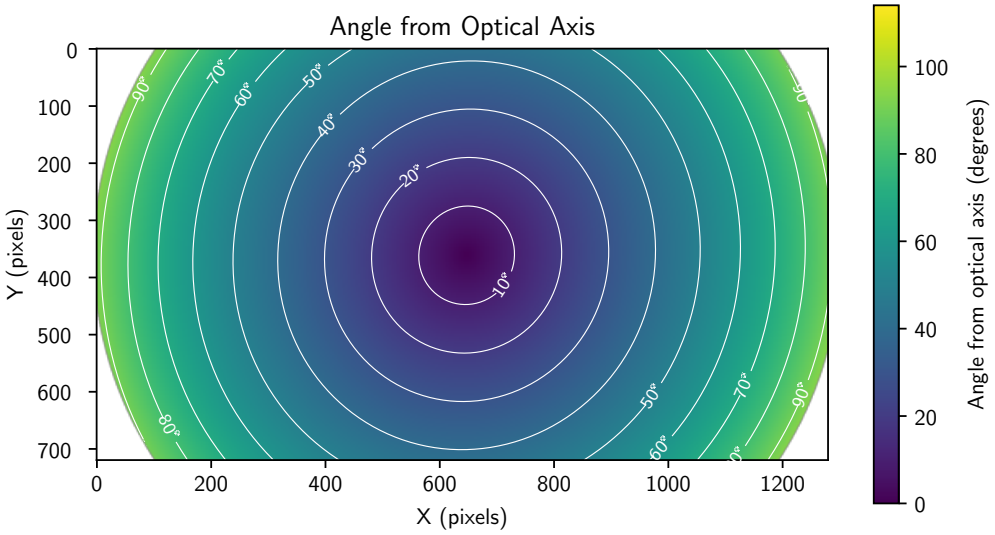
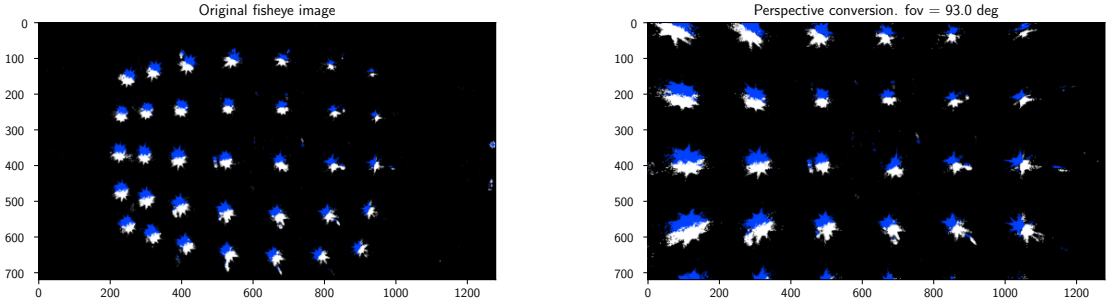


Figure 3.10: Angle from optical axis visualization, with the maximum angle of 93.47 degrees.

We can also apply a perspective conversion to the whole image, which now correctly represents distances and angles. We can notice this by looking at the calibration lattice at Fig. 3.11, which now looks like a grid of points.



(a) Uncalibrated image.

(b) Calibrated image.

Figure 3.11: Two photos of the calibration lattice, one uncalibrated on Fig. 3.11a and the one calibrated at Fig. 3.11b, which does not exhibit any distortion.

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

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. 3.12. The calibration results can be seen on Fig. 3.13, with the visualization on Fig. 3.14 and the perspective projection on Fig. 3.15.

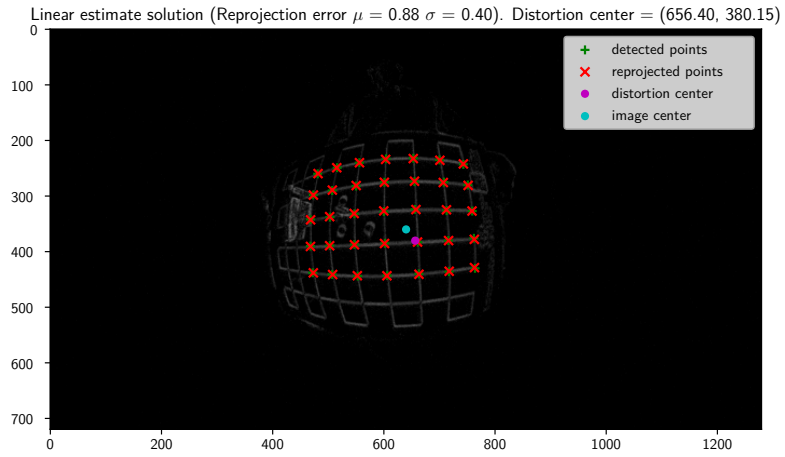
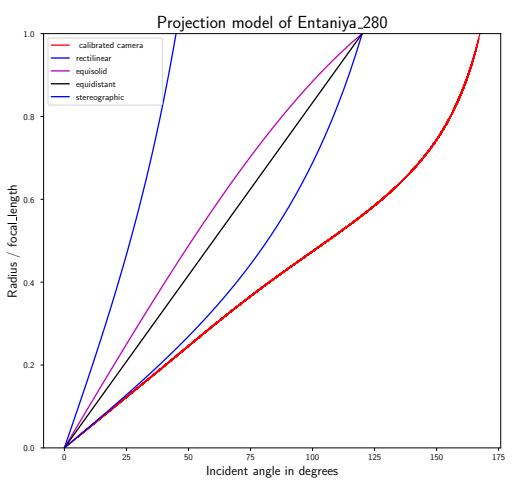
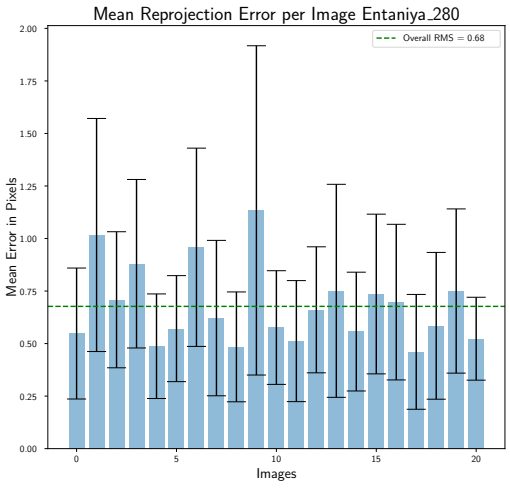


Figure 3.12: Chessboard calibration target visible in 280 degree lens with marked chessboard edge points



(a) Calibrated lens model function



(b) Calibration images mean reprojection errors

Figure 3.13: Calibration results for the Entaniya 280 degree lens with the calibrated lens model function highlighted in red in Fig. 3.13a and the calibration images mean reprojection errors on Fig. 3.13b.

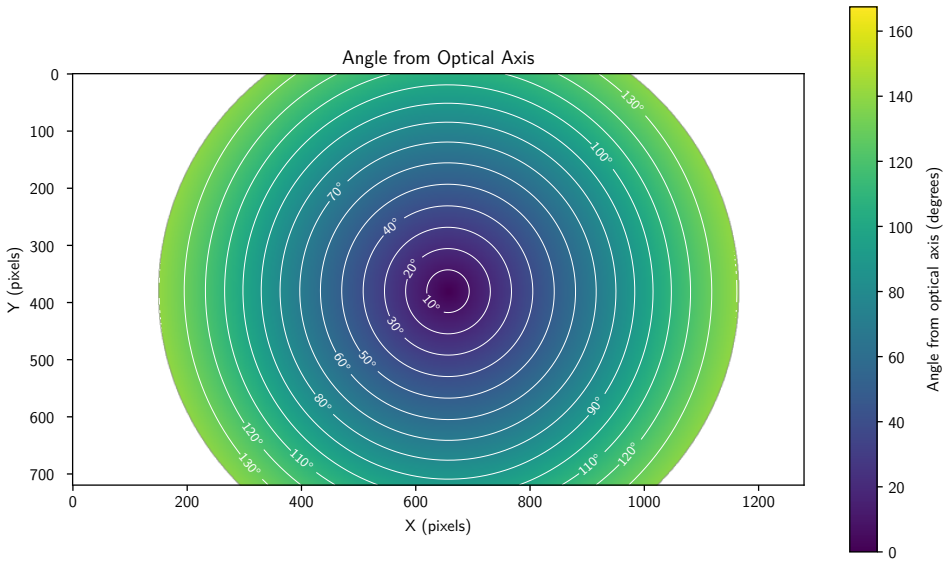
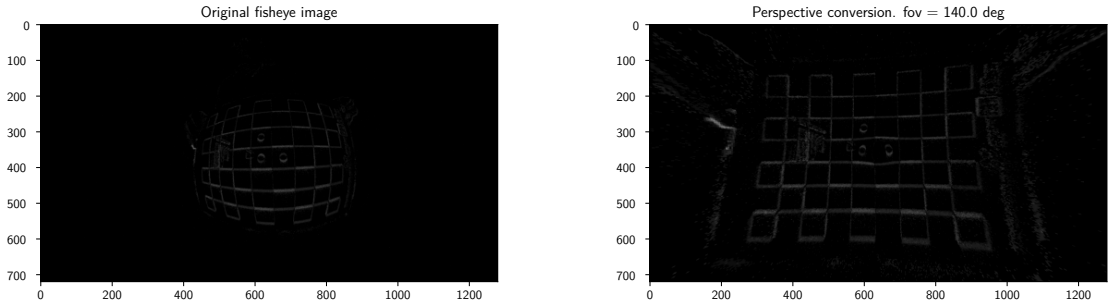


Figure 3.14: Angle from optical axis visualization, with the maximum angle of 140 degrees on each side, making its FOV 280 degrees.



(a) Uncalibrated image.

(b) Calibrated image.

Figure 3.15: Two photos of the calibration chessboard from the Entaniya 280 degree lens, one uncalibrated on Fig. 3.15a and the one calibrated at Fig. 3.15a.

4 Distance estimation

For distance estimation, one can leverage many different approaches, either directly relying on the event data stream itself, or integrating the events over time (thus producing a grayscale image or a heatmap), and then estimating the position from the produced image. As we have shown in chapter 2, the number of events generated by LED sources decreases monotonically with the square of the distance and also decreases with increasing modulation frequency. We could then assume that a simple distance predictor could be made by simply fitting a curve to the training dataset consisting of an average number of events per blinking period (thus training this predictor for average number of events w. r. t. to distance) and then making the prediction of distance by calculating the average in real time.

This is possible to do in very specific cases where the camera settings and the scene lighting conditions do not change between training measurements and the deployment. For example, changing the `bias-diff-on` or `bias-diff-off` settings of the camera changes the brightness change threshold on which the camera generates events. This changes the number of events that are generated without any information on the distance from the camera, this is similar to changing the eposure time on a global shutter camera, which can then give an under- or over-exposed frame. It also does not generalize the problem of estimating the position of an arbitrary UAV marked with LED lights and a camera with previously unspecified settings. For a more robust way of estimating the pose of the UAV from the camera, we can leverage the following methods.

4.1 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. (2014) [13] 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 (4.1) which could be used to describe the power ratio $RSSR_{1,2}$

$$RSSR_{1,2} = \frac{P_{R1}}{P_{R2}} = \left(\frac{d_2}{d_1}\right)^{n+3} \tag{4.1}$$

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 $n = 1$).

4.2 Perspective-n-Point

The Perspective-n-Point (PnP) problem 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 can be distingushed by the camera as a point

light source. However, due to physical limits, only three LEDs will be visible at a single point in time, the fourth one behind 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 $\mathbf{p}_i \in \mathbb{R}^3$, $|\mathbf{p}_i| = 1, i \in \{1,2,3\}$. These sets of points are related by a rigid transformation [3]

$$d_i \mathbf{p}_i = \mathbf{R}\mathbf{P}_i + \mathbf{t} \tag{4.2}$$

where $d_i \in \mathbb{R}^+$. Given the rigid transformation relationship shown in (4.2), 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, we use the method proposed by Kneip et al. [14], which directly finds the rotation and translation with a novel parameterization of the model. The visualization of the P3P problem is shown on Fig. 4.1.

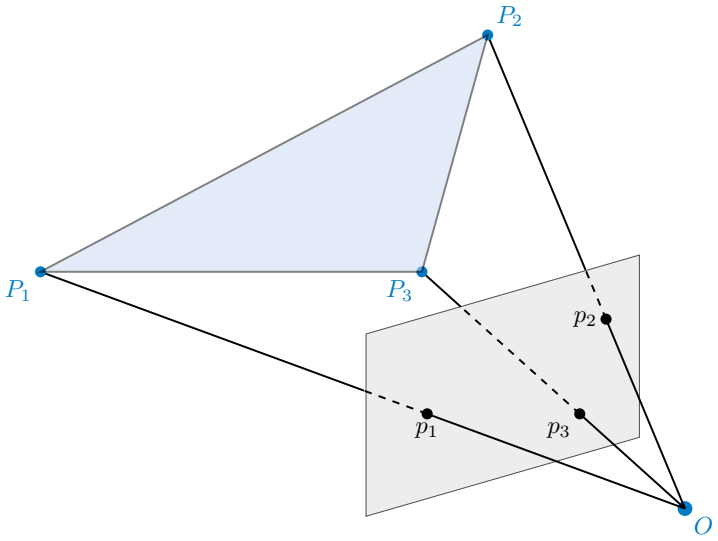


Figure 4.1: P3P problem visualization

In our implementation, we compute the average of the estimated pose and distance, thus simplifying the problem of identifying the correct solution. 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 geometrical center of the UAV.

4.3 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 at 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 that was influenced by their blinking frequency and by the known physical structure of the UAV. The view from the event-based camera can be seen on Fig. 4.2.

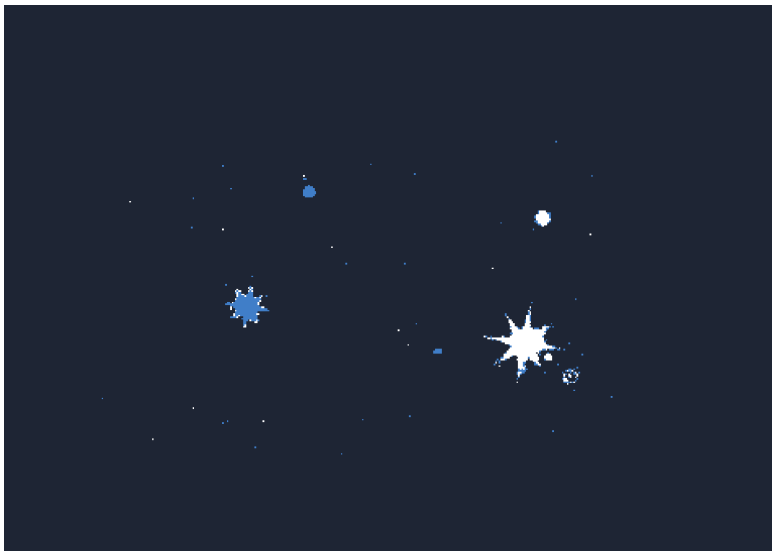


Figure 4.2: Data from the stationary experiment

Then a 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. 4.3, with several recordings present for each distance.

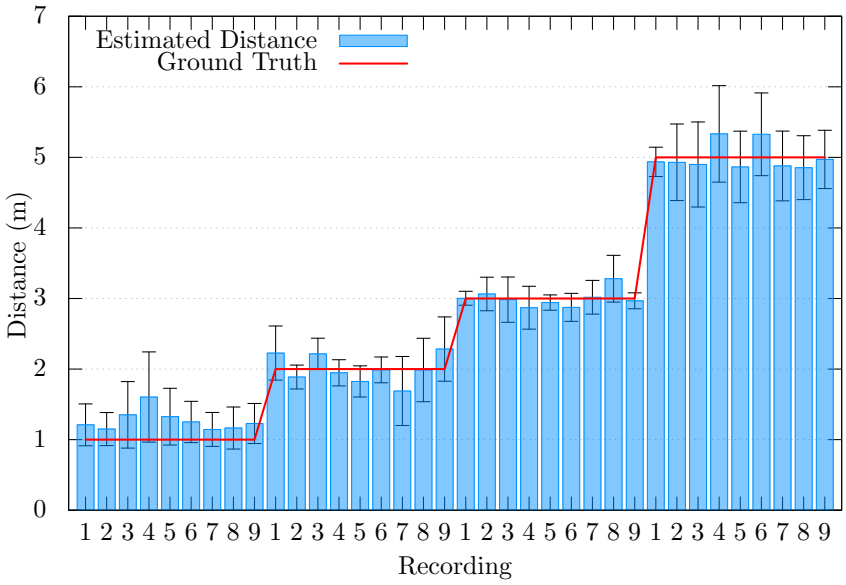


Figure 4.3: PnP estimation results

4.4 ROS implementation

To facilitate the deployment on real hardware, a Robot Operating System (ROS) distance estimator node was implemented [1]. The functionality can be summarized with the flow chart in Fig. 4.4.

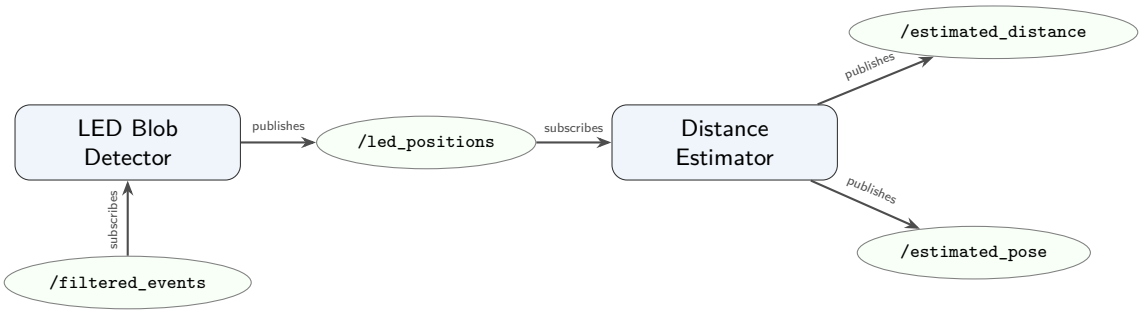


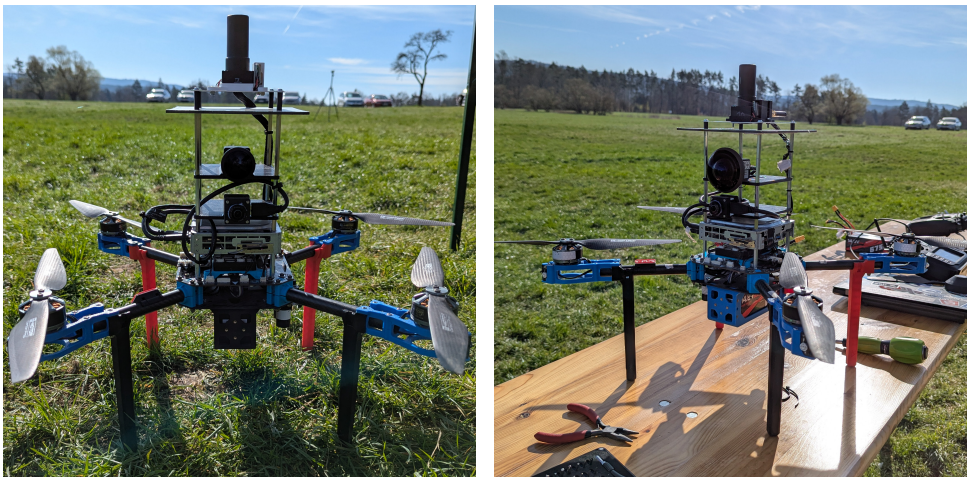
Figure 4.4: ROS distance estimation pipeline

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.

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

5 Experiment

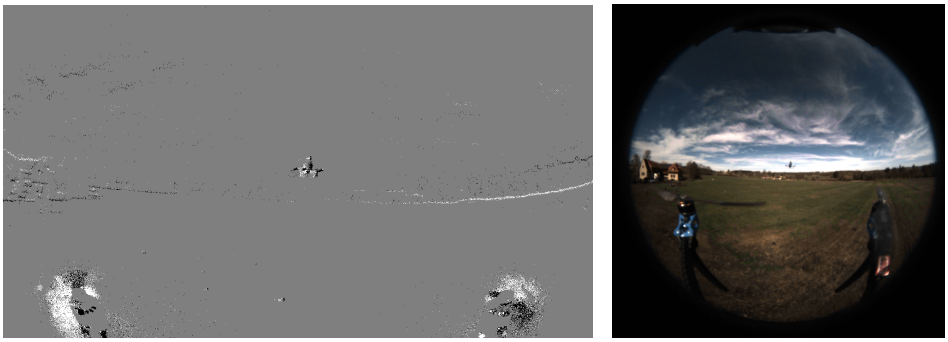
The experiment was performed with two X500 UAVs, each of them equipped with a Prophesee EVK4 event-based camera, one with a 2.5mm f/1.6 fish eye lens with an FOV of 93.5 degrees and the second one with Entaniya 1.07mm f/2.8 fish eye lens with an FOV of 280 degrees. Each UAV is also equipped with a Basler camera with a fish eye lens, to provide normal video signal that is recorded alongside the event stream from the event-based camera. Both cameras are connected to the onboard Intel NUC computer running the ROS system, on which all the processing is done during the flight. Both UAVs are also equipped with a Real-time Kinematics (RTK) module, which is used to localize the UAV, and is 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 Multi-robot Systems Group (MRS) Camp in Temešvár in August 2025, the UAVs can be seen on Fig. 5.1.



(a) UAV33 (b) UAV37

Figure 5.1: Two X500 UAVs, UAV33 on Fig. 5.1a and UAV37 on Fig. 5.1b.

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, we recorden 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. 5.2



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

Figure 5.2: The view of the experiment from UAV33, with event data on Fig. 5.2a and Basler camera view on Fig. 5.2b.

TODO: SHOW MEASURED DATA FROM RQT

TODO: SHOW GNSS/ESTIMATION DIFFERENCES

TODO: SHOW THE RVIZ/RQTPLOT VISUALIZATION PIPELINE

6 Conclusion

Summarize the achieved results. Can be similar as an abstract or an introduction, however, it should be written in past tense.

6.1 Future work

7 References

[1] 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.

[2] 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.

[3] 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.

[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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[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. 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.

A Appendix A

Used AI ...