CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Multi-Camera Visual Tracking Onboard UAVs

Vadim Mychko

Supervisor: Ing. Matouš Vrba
August 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Mychko Vadim**	Personal ID number:	**507455**
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:

Multi-Camera Visual Tracking Onboard UAVs

Bachelor’s thesis title in Czech:

Vizuální sledování cíle více kamerami na palubě UAV

Guidelines:

1. Research different methods and types of visual tracking including bounding box tracking and sparse feature tracking.
2. Develop a system for visual tracking of an object in images from multiple cameras placed onboard a moving autonomous UAV. The tracking should be initialized using a relative position of the target with respect to the UAV that is output by a detector using an onboard LiDAR [1].
3. Implement multiple visual tracking methods within the system including bounding box tracking and sparse feature tracking.
4. Evaluate the accuracy, robustness and general properties of the implemented methods in realistic simulations with drone-hunting scenarios.

Bibliography / sources:

[1] Matouš Vrba, Viktor Walter, Václav Pritzl, Michal Pliska, Tomáš Báča, Vojtěch Spurný, Daniel Heřt and Martin Saska, „On Onboard LiDAR-based Flying Object Detection,“ arXiv preprint cs.RO 2303.05404, 2023.
[2] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34(7):1409–1422, 2012.
[3] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In European Conference Computer Vision (ECCV), 2016.
[4] Jianbo Shi and Carlo Tomasi, „Good Features to Track,“ Conference on Computer Vision and Pattern Recognition, pages 593–600, 1994.

Name and workplace of bachelor’s thesis supervisor:

Ing. Matouš Vrba Multi-robot Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **12.01.2024** Deadline for bachelor thesis submission: **15.08.2024**

Assignment valid until: **21.09.2025**

Ing. Matouš Vrba	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

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

Acknowledgement / Declaration

I thank my supervisor, Ing. Matouš Vrba, for guiding me throughout the thesis topic.

I thank my family for their support throughout my studies at FEE CTU.

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

Prague, 15th August, 2024

. . . .

Abstrakt / Abstract

S rostoucím rozšířením bezpilotních letounů se zvyšuje důležitost zajištění bezpečnosti ve vzduchu. Tato práce se zabývá implementací systému vizuálního sledování pomocí více kamer na palubě bezpilotních letounů za účelem zachycení nespolupracujících letících cílů. Sledovací systém je navržen jako doplněk stávajícího detektoru využívající palubní LiDAR a řeší omezené zorné pole LiDARu. Různé metody vizuálního sledování jsou zkoumány a vyhodnocovány ve scénářích lovu dronů včetně sledování pomocí ohraničujících obdélníků a sledování významných bodů. Vizuální sledovače s jednou kamerou jsou rozšířeny pro nasazení s více kamerami s využitím modelu dírkové kamery. Implementovaný vícekamerový systém vykazuje značné problémy při vyhodnocování v simulaci, částečně kvůli předpokladům. Práce zdůrazňuje potřebu flexibilnějších přístupů ke scénářům zachycení a navrhuje směry budoucího výzkumu.

Klíčová slova: vizuální sledování více kamerami, počítačové vidění, bezpilotní letouny, sledování v reálném čase

Překlad názvu: Vizuální sledování cíle více kamerami na palubě UAV

As the utilization of Unmanned Aerial Vehicles (UAVs) continues to rise, ensuring aerial safety becomes increasingly critical. This thesis explores the implementation of a multi-camera visual tracking system onboard UAVs for capturing non-cooperating flying targets. The tracking system is designed to complement an existing LiDAR-based detection algorithm, addressing the limited field of view of the LiDAR sensor. Various visual tracking methods, including bounding box and sparse feature tracking, were researched and evaluated in drone-hunting scenarios. Single-camera visual trackers are extended into the multi-camera setting by utilizing the pinhole camera model. The implemented multi-camera system reveals significant challenges when evaluated in simulation, partially due to the made assumptions. The work highlights the need for more flexible approaches to interception scenarios and proposes directions for future research.

Keywords: multi-camera visual tracking, computer vision, unmanned aerial vehicles, real-time tracking

/ Contents

Tables / Figures

Chapter 1 Introduction

The widespread adoption of Unmanned Aerial Vehicles (UAVs), commonly known as drones, has significantly impacted various sectors, from aerial photography to surveillance. The global UAV market is experiencing rapid growth, with projections indicating an increase from \$33.7 billion in 2023 to \$54.6 billion by 2030 [1]. This rapid growth has raised security concerns regarding aerial safety, particularly in sensitive areas such as airports, government facilities, and critical infrastructure. The December 2018 Gatwick Airport incident involved multiple UAV sightings near the airport's runways, forcing the airport's closure for an extended period. This disruption affected approximately 140,000 passengers and resulted in the cancellation of 1,000 flights over a 36-hour period, demonstrating the severe consequences of unauthorized UAV activity in sensitive areas and highlighting the potential for malicious UAV use to cause significant economic and logistical damage.

Conventional counter-UAV systems, often ground-based and reliant on human operators, may face mobility and response time limitations. These systems frequently depend on jamming the navigation and control systems of intruding UAVs, which may not always be practical or desirable. Human-operated systems are prone to errors due to fatigue, stress, or limited visibility. Additionally, physical takedown methods can lead to uncontrolled crashes, posing risks to people and property on the ground and potentially destroying valuable evidence for subsequent investigation and forensic analysis. These limitations underline the need for more advanced, autonomous, and adaptable counter-UAV solutions to effectively address the growing challenges posed by modern UAV technologies.

Recent advancements in counter-UAV technology have led to the development of an innovative autonomous aerial interception system, as described in [17, 21]. This system employs an interceptor UAV equipped with a 3D Light Detection and Ranging (LiDAR) sensor and a high-performance onboard computer. The system utilizes the LiDAR sensor for detecting and tracking unauthorized UAVs and a novel fast-response proportional navigation method for interception guidance, allowing for quick response times and high accuracy when intercepting agile, maneuvering targets. The approach offers significant advantages over ground-based solutions, including non-destructive capture using a deployable net, enhanced situational awareness due to its aerial perspective, and fully autonomous operation that reduces reliance on human operators. Figure 1.1 illustrates a prototype of such an interceptor UAV.

The existing autonomous aerial interception system, while innovative, faces certain limitations that impact its effectiveness in some scenarios. A primary constraint lies in the LiDAR-based detection system, which, despite its accuracy, is restricted by a limited Field of View (FOV) and detection range. These limitations can hinder the system's ability to continuously track agile or distant targets, especially during complex interception maneuvers. This thesis proposes an extension to the current solution by incorporating a multi-camera visual tracking system to address these shortcomings.

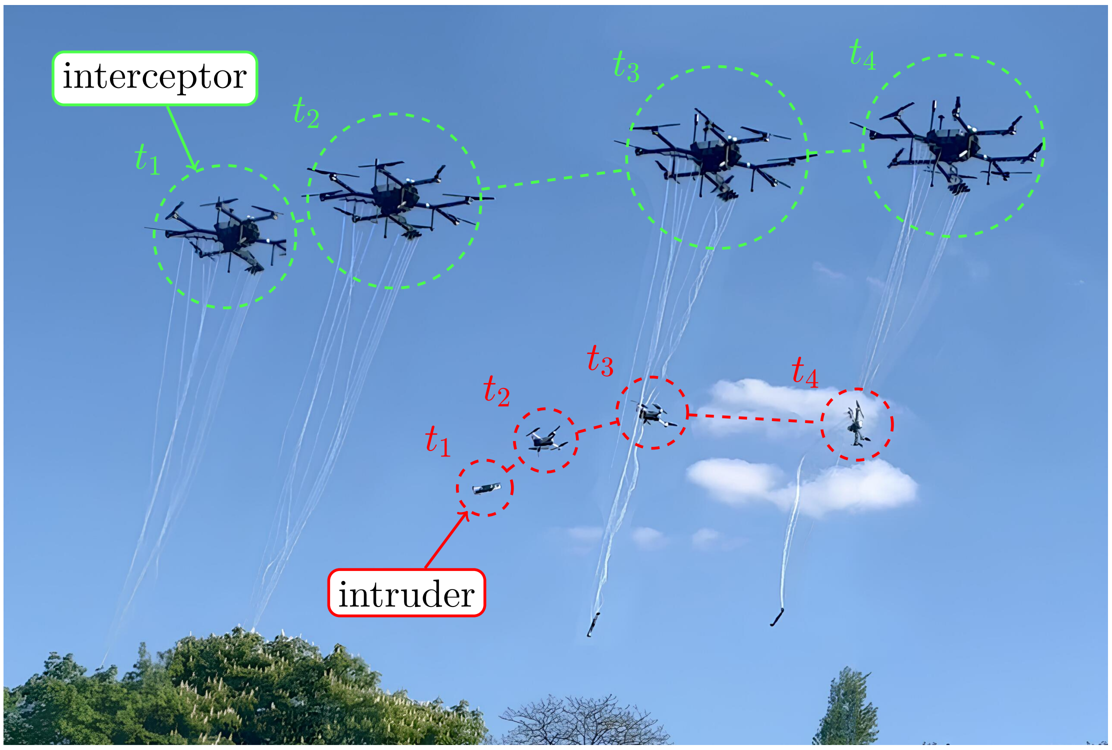

Figure 1.1. Collage of a successful autonomous interception of a moving target. The maneuver took approximately two seconds from t_1 to t_4. Taken from [17].

This enhancement aims to complement the existing LiDAR-based detector with visual input from digital cameras, expanding the system's overall FOV. By combining LiDAR and visual data, the proposed solution seeks to provide more robust and continuous tracking of the target, even when it momentarily moves outside the LiDAR's detection zone. Additionally, this multi-sensor approach is expected to improve the target's state estimation accuracy, potentially leading to more precise interception planning and execution. This thesis thus focuses on developing and integrating this visual tracking component to overcome the current limitations and enhance the overall performance of the existing system.

The implementation of a visual tracking system on the interceptor UAV introduces several challenges that need to be considered. The interceptor UAV must remain agile and lightweight to effectively pursue and capture targets, which introduces size, weight, and power consumption constraints. These constraints necessitate the development of highly efficient algorithms capable of concurrent execution on a low-power Central Processing Unit (CPU), including modules for detection, tracking, state estimation, and interception planning. The visual tracking component must demonstrate robustness in diverse environmental conditions, including varying lighting, occlusions, and rapid movements.

The primary objectives of this research are threefold. First, to investigate and evaluate sparse feature and template-based visual trackers in the context of drone-hunting scenarios. Second, to integrate the selected visual tracking methods into the existing system, combining them with the LiDAR-based detection algorithm. Finally, to extend single-camera visual tracking capabilities to a multi-camera configuration, enhancing the overall tracking performance and robustness of the interceptor UAV system.

The rest of the thesis is organized as follows: Chapter 2 reviews the related work. Chapter 3 discusses the implemented multi-camera system in more detail. Chapter 4 evaluates the integrated trackers in drone-hunting scenarios. Chapter 5 discusses the results and makes suggestions for future research.

Chapter 2
Related Work

This chapter reviews existing research on detection of UAVs, single-camera visual object tracking, state estimation techniques from a monocular camera, and navigation methods for aerial interception.

2.1 Detection of UAVs

A novel approach for detection and localization of flying objects using a 3D LiDAR sensor is presented in [21], which is specifically designed for autonomous aerial interception. The proposed system leverages a novel 3D occupancy voxel mapping method to spatially differentiate flying objects from the background. This system offers high localization accuracy, robustness to different environments and target appearance changes, and a great detection range.

2.2 Sparse Feature Tracking

This section presents an overview of the existing sparse feature trackers. Sparse feature tracking, also known as point tracking, is a computer vision technique that involves identifying and following distinct points or features across multiple frames of video or images.

2.2.1 Lucas-Kanade

One of the earliest works on tracking of features between images is the Lucas-Kanade (LK) method [14], which was originally proposed as a method for estimating optical flow. While this method does not explicitly track an object of interest, it may be employed to track its distinct features. The LK method assumes constant motion within the local neighbourhoods of the selected features, which implies the following:

- The LK method is unsuitable for estimating large motions across two images. This can be partially resolved by introducing image pyramids, where the resolution of each of the two images is essentially halved for each pyramid level [2]. Then, the LK method is used on the images with the smaller resolution; the obtained estimates are backpropagated to the images with the higher resolution, and the whole process is repeated. Such pyramidal implementation improves tracking of even large motions.
- The LK method is unsuitable for tracking dynamic objects within a static background. Unfortunately, this fact cannot be directly avoided or compensated for, as the assumption of constant flow within local neighbourhoods lies in the core design of the method.

The quality of the LK-based tracking depends on the *number of levels* in the image pyramid, the *size of the feature window* (the size of local neighbourhoods), the *texturedness* of the image within these windows, and the amount of *camera motion* between frames. Texturedness (or cornerness) is a measure of how much local intensity variation exists in a small neighborhood of pixels. Areas with high texturedness are rich in visual information and are more likely to be reliably tracked. The number of levels and the size of the feature window are hyperparameters, which are set before the algorithm is executed and are fine-tuned for the specific application. The camera motion cannot be calculated in advance. However, the choice of what features to track (the texturedness) can significantly boost the performance of the LK method. One method to choose the features is using the Shi-Tomasi corner detector [20], which finds suitable features for increasing the LK tracker's accuracy.

2.2.2 CoTracker

One of the most recent works on point tracking is the CoTracker [10], a deep learning-based model for joint pixel tracking across a video sequence. CoTracker demonstrates impressive qualitative results compared to other modern methods. However, it uses windowed inference, which implies that the video has to be obtained in advance to predict points in the following frames. In turn, it limits the ability to get predictions in the following frames in a rapid fashion. Moreover, the CoTracker is significantly more computationally demanding than the LK tracker, which makes it undesirable to use without hardware acceleration.

2.3 Bounding Box Tracking

This section provides an overview of the existing bounding box trackers. Bounding box tracking is a computer vision method that involves identifying and following an object of interest by enclosing it within a rectangular box across consecutive frames of video or images. This technique tracks the entire object as a unit, updating the position, size, and orientation of the bounding box to consistently represent the target object's location and dimensions throughout the sequence.

2.3.1 MedianFlow

Sparse feature trackers can be extended into bounding box tracking. One example is the MedianFlow tracker [8], which creates a grid of points inside the given initial bounding box. These points are tracked by the LK method, and for each point, the error is measured. After that, 50% of the worst predictions are filtered out, while the remaining predictions are used to calculate the displacement and scale change of the bounding box. The MedianFlow tracker explicitly assumes that the points are not entirely independent and are parts of bigger objects. However, the tracker tends to suffer from the same problems as the LK method because the LK method is in the core design of the MedianFlow tracker. This implies that it is not ideal for tracking dynamic objects, especially objects with non-rigid parts, such as rotary wings of UAVs.

2.3.2 TLD

Short-term trackers fail when the object of interest is absent in subsequent frames and are likely to never recover because such trackers usually do not implement any detection logic. However, any short-term tracker can be extended into a long-term tracker by utilizing the Tracking-Learning-Detection (TLD) framework [9]. This framework explicitly splits long-term tracking into three components: tracking, learning, and detection. The *tracker* estimates the object's motion frame-to-frame, the *detector* performs full scans of frames to localize all appearances observed and learned in the past, and *learning* monitors their performance to identify and correct errors by generating new training data.

The detector generates all possible scales and shifts of the initial bounding box, classifies generated patches, and accepts those with the object present. In the original version of the TLD tracker, MedianFlow, together with failure detection, is used as the tracking component. In turn, the TLD tracker suffers from the same problems as MedianFlow and LK trackers, rarely benefiting from the detection component in the application considered in this thesis.

The tracking component of the TLD tracker can be replaced with other single-target bounding box trackers. Doing so might offer advantages such as improved performance in challenging scenarios, adaptability to specific use cases, and the ability to incorporate newer tracking algorithms. A replacement tracker must provide similar functionality while maintaining TLD's real-time performance. Replacing the tracker would be a complex undertaking that could compromise the simplicity and efficiency of the original TLD framework.

2.3.3 GOTURN

With the rise of deep learning techniques and neural networks, new state-of-the-art has been achieved in many fields, including visual object tracking. One example is GOTURN [5], which is based on Convolutional Neural Networks (CNNs) and utilizes the architecture of Siamese neural networks. Siamese neural networks accept two different input vectors to compute the output while sharing the same weights of select layers [3]. In the case of the GOTURN tracker, the neural net accepts two inputs: the previous frame cropped, scaled, and centered on the object of interest ("what to track"), and the current frame cropped, scaled, and centered on the previous location of the object ("where to look"). The neural net outputs the coordinates of the bounding box in the current frame. Unfortunately, the tracker runs at approximately 21 Frame Per Second (FPS) on a multi-core CPU, making it impractical without the usage of hardware accelerators or a Graphics Processing Unit (GPU).

2.3.4 DaSiamRPN

Another tracker based on Siamese neural networks is DaSiamRPN [24]. The authors highlight the importance of quality training data and propose a novel training strategy. DaSiamRPN can be used for long-term tracking thanks to the proposed local-to-global region strategy. Like GOTURN, it can run at over 100 FPS on a GPU, but only approximately 13 FPS on a multi-core CPU. GOTURN and DaSiamRPN are used as baselines in the evaluation in terms of the chosen metrics and inference time but are unsuitable for deployment onboard resource-constrained UAVs.

2.3.5 Nano

Modern deep learning-based trackers heavily rely on CNNs, which typically consist of millions of parameters to be learned. Such trackers are not usable on embedded devices and onboard computers with limited computational power. This issue may be partially resolved with the MobileNet architectures [6] by introducing depthwise separable convolution, which is extremely efficient compared to the traditional convolutional layers. Models based on MobileNets achieve comparable (and sometimes better) accuracy than the models based on the traditional CNNs while performing up to 22 times less computations.

Another option for making neural networks more computationally efficient is quantization. Regarding the neural networks, the inputs and weights are restricted to integer-only arithmetic, allowing for a lower memory footprint and inference time with little to no reduction in performance [7].

Designing performant and lightweight models relies on human knowledge and expertise. LightTrack [23] uses a neural architecture search to automate the process of designing models suitable for embedded devices, allowing to find trackers that achieve better performance than hand-crafted trackers. Nano [4] is yet another tracker based on Siamese neural networks, which further builds on the LightTrack strategy and achieves better performance with fewer parameters. The Nano tracker is highly optimized, allowing it to run at over 100 FPS on a multi-core CPU.

2.4 State Estimation

The problem of estimating a flying target's 3D position and velocity from a monocular camera onboard a UAV is investigated in [15]. The author implements and compares multiple approaches, including geometric methods based on line intersection and Kalman Filter-based algorithms using bearing measurements. The performance of the state estimation techniques is evaluated through both simulated and real-world scenarios designed to resemble the interception maneuver. These experiments align with the primary application of the visual tracking system considered in this thesis – the interception of flying objects by autonomous aerial vehicles.

Several of the implemented methods demonstrated the capability to estimate the target's state under various conditions accurately. However, it is important to note that the quality of the state estimation was found to be highly dependent on the precision of the visual tracking system. The visual tracker, responsible for following the target throughout the video stream, plays a crucial role in providing accurate input data for the state estimation algorithms. Consequently, any imprecisions or errors in the visual tracking process can significantly impact the overall performance of the state estimation.

2.5 Navigation & Planning

A novel approach for autonomous aerial interception of non-cooperating UAVs by an interceptor UAV is presented in [17]. The authors propose a new guidance method called Fast Response Proportional Navigation, designed to intercept agile maneuvering targets while relying on onboard state estimation and tracking. The guidance method is compared to existing approaches in extensive simulations, demonstrating its superior performance in terms of response time and number of successful interceptions. The paper also addresses challenges in target state estimation, proposing an interacting multiple model filter and a new measurement model to improve accuracy when tracking targets following general trajectories.

The authors integrate their proposed methods into a complete autonomous interception system, which is evaluated in realistic simulations and real-world experiments. The system combines LiDAR-based detection and tracking of flying objects from [21] with the proposed state estimation and guidance techniques. The experiments show that the system can successfully intercept highly maneuvering targets in real-world conditions, outperforming existing solutions.

2.6 Datasets & Benchmarks

The Visual Object Tracking (VOT) challenge series [11–13] aims to advance single-camera visual tracking research. The benchmarks utilize curated datasets featuring diverse objects and scenarios, with recent editions transitioning from bounding boxes to segmentation mask annotations. The challenge was expanded to address various tracking scenarios, including short-term tracking, where trackers are reset after failure, and long-term tracking, where algorithms must handle target disappearance and re-detection.

UAV123 [16] is a dataset and benchmark for evaluating visual object tracking algorithms using video footage from cameras mounted on UAVs. UAV123 presents challenges that are common in aerial tracking scenarios but are underrepresented in existing ground-based datasets, such as significant scale and aspect ratio changes, fast motion, and low-resolution targets. By evaluating 14 popular trackers on this dataset, the authors demonstrate that many algorithms struggle with the specific challenges posed by UAV-based tracking.

Chapter 3
Methodology

This chapter describes the multi-camera system in more detail: how the tracking is initialized, how the object is tracked in one camera, and how the tracking information between cameras is exchanged.

3.1 System Overview

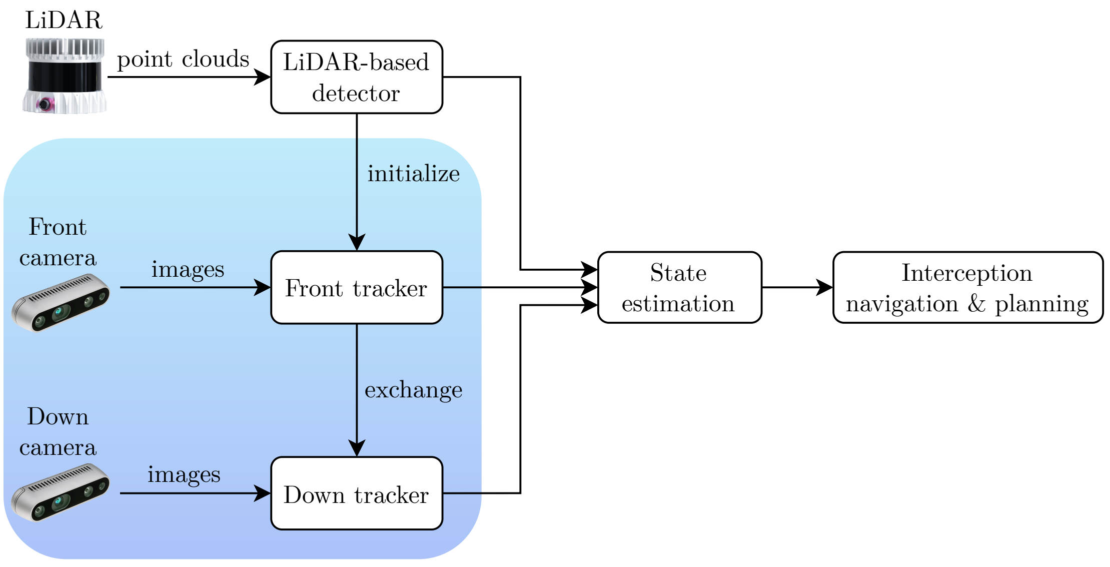

Figure 3.1. Diagram of a possible autonomous aerial interception system, complemented with multiple digital cameras. The multi-camera tracking system (highlighted in blue) is tackled within the thesis. The multi-camera system can be extended with more digital cameras by defining an "exchange" relationship between the cameras. The cameras must have overlapping FOVs and must be calibrated in order to exchange the tracking information.

Figure 3.1 illustrates the block diagram of the implemented multi-camera visual tracking system. The tracking is initialized using a relative position of the target with respect to the interceptor UAV, which is output from the detection algorithm [21]. In the diagram, each tracker ("Front tracker" and "Down tracker") is assumed to be a single-camera tracker. Exchange of tracking information between the trackers is established through a unidirectional "exchange" relationship. The intended behavior of this system is outlined as follows:

1. The front tracker, which has an overlapping FOV with the LiDAR sensor, receives a point cloud (a collection of 3D points) assumed to belong to the target, as identified by the detector.

2. The front tracker transforms the received point clouds into its coordinate system and projects them onto the image plane.

3. The front tracker independently tracks the target in the images and exchanges the tracking information with the down tracker whenever possible.

4. Upon successful tracking information exchange, the down tracker receives the initial bounding box and begins tracking the target independently from the front tracker.

5. The trackers predict the target's position in the image, even when the target moves outside the LiDAR's FOV. This can be used to estimate the target's position and velocity (e.g., by a Kalman Filter).

3.2 Tracking Initialization

A pinhole camera model is a purely geometric mathematical model for projecting points from the world onto the image plane and backprojecting points from the image plane into the world. This model is used for projecting the incoming detections (collections of 3D points) onto the image plane. This camera model requires *intrinsic* parameters (which characterize the camera itself) and *extrinsic* parameters (which define the camera's position and orientation in the world). The intrinsic parameters are typically obtained using calibration, while extrinsic parameters depend on the specific camera setup. This thesis does not cover the process of obtaining these parameters. Instead, the parameters are assumed to be known within a small margin of error. However, it should be noted that these parameter estimates may not be entirely accurate when applied in real-world experiments. This thesis assumes an ideal pinhole camera model and does not account for any distortion.

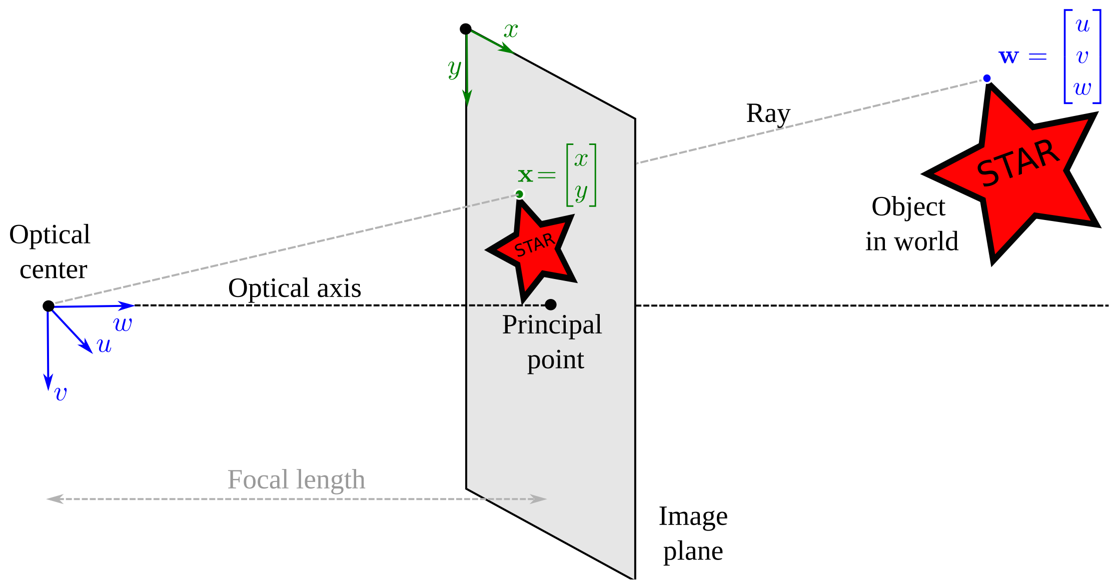
x
y
$\mathbf{w} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$
Ray
$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
STAR
STAR
Object in world
Optical center
Optical axis
w
u
v
Principal point
Focal length
Image plane

Figure 3.2. Pinhole camera model terminology. The optical center (pinhole) is placed at the origin of the 3D world coordinate system (u, v, w), and the image plane (where the virtual image is formed) is displaced along w-axis, which is also known as the optical axis. The position where the optical axis strikes the image plane is called the principal point. The distance between the image plane and the optical center is called the focal length. Taken from [18].

Figure 3.2 introduces common pinhole camera model terminology. To establish the position $\mathbf{x} = [x, y]^T$ in the image plane where the 3D point $\mathbf{w} = [u, v, w]^T$ is projected, the following equation in homogeneous coordinates is solved:

$$\lambda \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & 0 & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\tau} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix}, \tag{1}$$

where λ is an arbitrary scaling factor, ϕ_x and ϕ_y are focal lengths in the x- and y-directions, $[\delta_x,\ \delta_y]^T$ is the position of the principal point (in pixels), $\boldsymbol{\Omega} \in \mathbb{R}^{3\times3}$ is a rotation matrix, and $\boldsymbol{\tau} \in \mathbb{R}^{3\times1}$ is a translation vector. $\{\phi_x, \phi_y, \delta_x, \delta_y\}$ are intrinsic parameters, and $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$ are extrinsic parameters.

The conversion of incoming detections (collections of 3D points) to the camera's 2D image plane is achieved by solving Equation (1) for each 3D point. These projected points are immediately usable for sparse feature trackers without additional processing. However, an additional step is necessary when employing these projections to initialize bounding box trackers. This involves transforming the projected points into a bounding box configuration by identifying the minimum and maximum coordinates along each axis, thereby defining the four corners of the required initial bounding box.

3.3 Single-Camera Tracking

Once initialized, trackers with an overlapping FOV with the LiDAR sensor start tracking the object of interest, operating independently from the LiDAR-based detector. Sparse feature trackers track the distinct features of the object while bounding box trackers assign bounding boxes in each consequent frame. The choice of the tracker is not restricted and may include short-term trackers without built-in re-detection capabilities. It is assumed that during crucial interception maneuvers, the LiDAR-based detector, rather than the tracker itself, will perform any necessary re-detection.

3.4 Information Exchange

Two cameras must be positioned with overlapping FOVs to exchange tracking information. Both cameras are assumed to be modeled using the pinhole camera model, and the intrinsic and extrinsic parameters to be known. When a point is backprojected from one camera into the world, a ray represents the possible positions of the 3D point corresponding to the image point, as shown in Figure 3.3. When this ray is projected onto the image plane of the second camera, it forms an epipolar line, making the exact position of the projected point ambiguous. The ray in the camera's coordinate system knowing its 2D projection $\mathbf{x} = [x, y]^T$ in the image plane is obtained as:

$$\begin{aligned} u &= \left(\frac{x - \delta_x}{\phi_x} \right) w, \\ v &= \left(\frac{y - \delta_y}{\phi_y} \right) w, \end{aligned} \tag{2}$$

where $w \in \mathbb{R}$ is an unknown parameter and $\mathbf{w} = [u, v, w]^T$ is a point lying on the ray.

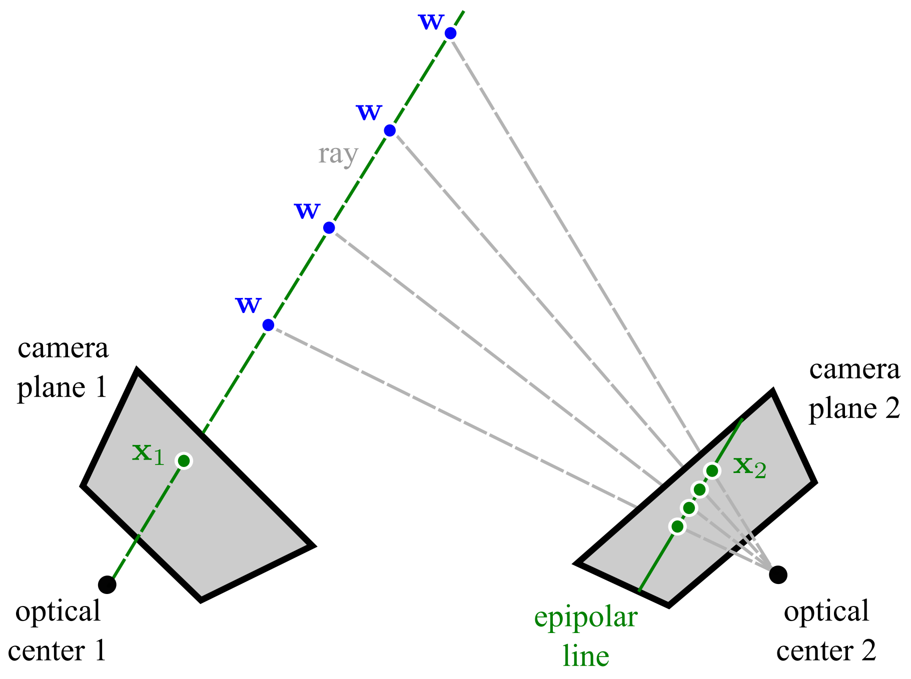

Figure 3.3. A 3D point $\mathbf{w}$ corresponding to $\mathbf{x}_1$ on the image plane lies somewhere on a ray, which passes through the optical center of the first camera, and the position of $\mathbf{x}_1$. However, it is unknown where the 3D point $\mathbf{w}$ lies on the ray. Projections of all possible positions of $\mathbf{w}$ onto the image plane of the second camera form a line, which is referred to as an epipolar line. Taken from [19].

The exact position of a 3D point backprojected from an image plane can be determined when complemented with depth information. This thesis assumes that the camera, which has an overlapping FOV with the LiDAR sensor, provides depth information aligned with its RGB visual image (e.g., by a stereo camera). Then, for the pixel $\mathbf{x} = [x, y]^T$ in the image plane, the depth $d \in \mathbb{R}$ (in meters) is known, and substituting the w variable in Equation (2) with $w = d$, the complete solution is obtained as:

$$\begin{aligned} u &= \left(\frac{x - \delta_x}{\phi_x}\right) d, \\ v &= \left(\frac{y - \delta_y}{\phi_y}\right) d, \\ w &= d. \end{aligned}$$

After establishing the 3D point, it is transformed and projected onto the second camera's image plane by solving Equation (1). For sparse feature trackers, the procedure is repeated for each point tracked by the first camera, and the second camera's tracker becomes initialized.

For bounding box trackers, a more complex approach is employed. First, the tracked object's center depth is estimated by analyzing the depth distribution within the bounding box. This estimation calculates the average of the middle 50% of sorted depth values within the bounding box. Subsequently, for each pixel within the bounding box, the absolute difference between its specific depth and the estimated center depth is calculated. If this difference falls within a predetermined threshold (specified as a hyperparameter), the pixel is accepted for the previously described routine. After obtaining multiple

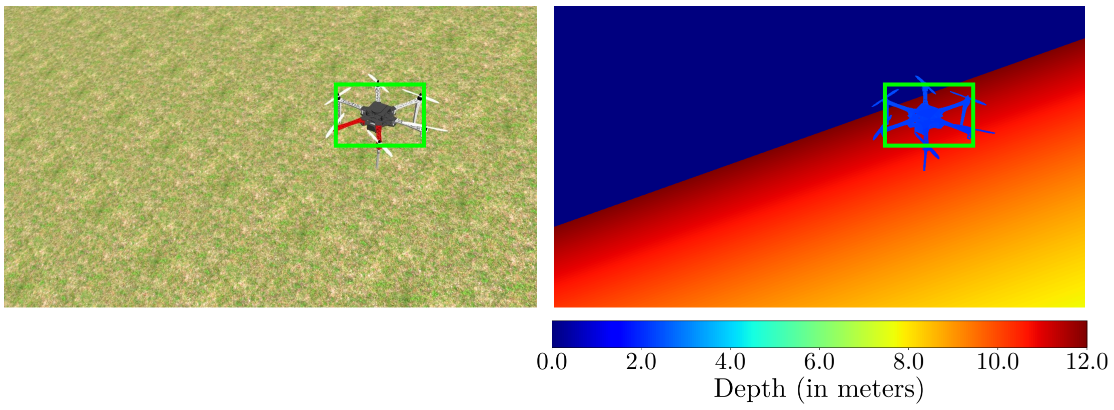

Figure 3.4. RGB image of the target with the aligned depth estimations taken from simulation using the Intel RealSense RGB-D camera. The depth information is represented as a 16-bit grayscale image, with the depth equal to zero in the areas where depth estimation is unavailable (notice the dark region at the top of the depth image). The green bounding box denotes what a single-camera tracker could predict.

projected points in the second camera's image plane, the points are transformed into the bounding box, similarly to how the detections are transformed.

Figure 3.4 shows an example of an RGB image of the target with the aligned depth estimations. Inside the green bounding box, there is not only the body of the tracked target but also the background. Only the points belonging to the tracked object are desirable when exchanging tracking information. Therefore, the previously described procedure filters the background points inside the bounding box.

Chapter 4
Evaluation

This chapter evaluates bounding box and sparse feature trackers in drone-hunting scenarios. This chapter also examines how the implemented multi-camera tracking system behaves during interception maneuvers in simulation.

4.1 Bounding Box Tracking

The considered bounding box trackers were evaluated on chosen sequences from the UAV123 dataset [16]. The chosen sequences (specifically "uav[1-8]") consist of tracking other UAVs from the camera mounted on a chasing UAV. The sequences offer a challenging dataset that closely mirrors the real-world complexity of the expected application of the implemented system. Moreover, the sequence "uav1" is included in the VOTS2023 benchmark [13] on segmentation tracking, indicating the high-quality and challenging nature of the sequences. All used implementations of the bounding box trackers are from the OpenCV library[1], including the weights for the deep learning-based trackers. Figure 4.1 shows a few frames from the selected UAV123 sequences.

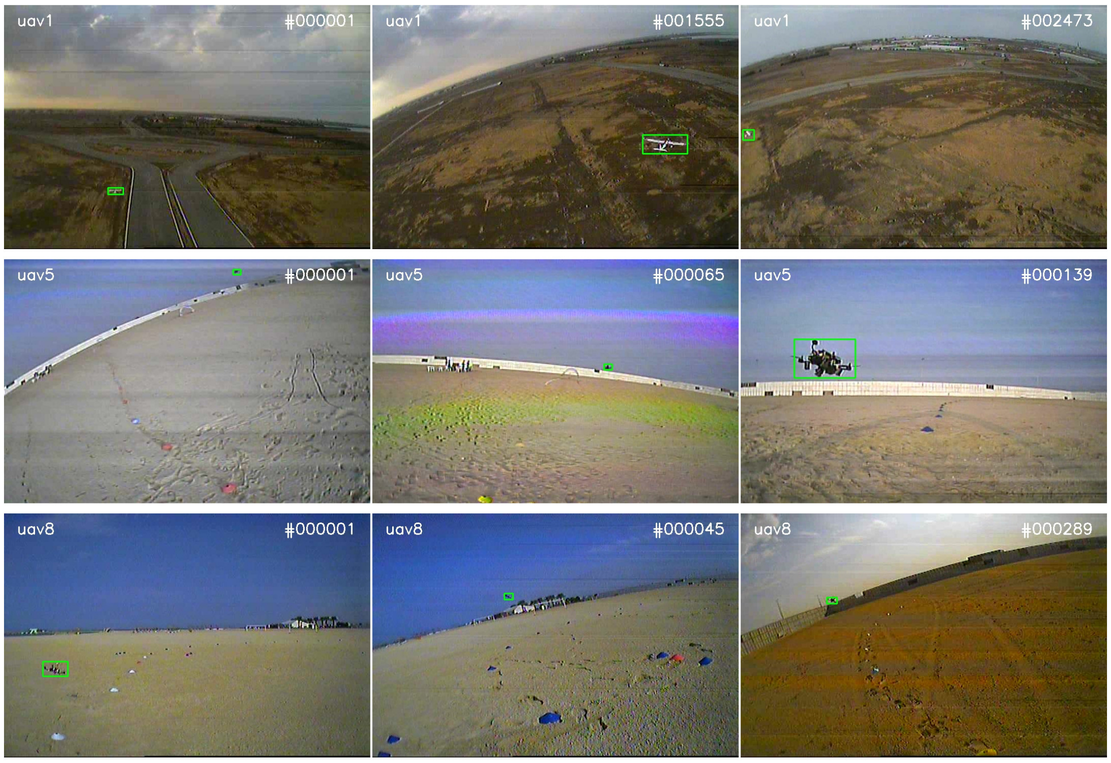

Figure 4.1. Frames of the selected sequences from the UAV123 dataset. The green bounding box indicates the ground truth annotation.

[1] https://opencv.org

The anchor-based evaluation methodology from the VOT2020 benchmark [11] is used for the evaluation. Each tracker is initialized at a specific frame (anchor) and run without further re-initialization. Anchors are placed approximately 50 frames apart (depending on whether the object is visible or not). If an anchor is placed in the first half of a sequence, the tracker is evaluated in the forward direction. Otherwise, the tracker is evaluated in the backward direction. Such an evaluation technique allows an objective comparison of all trackers because the trackers cannot show improved performance based on a random favorable re-initialization. On a sub-sequence starting from an anchor a of a sequence s, the considered metrics taken from [12, 22] are defined as:

- **Accuracy** A is the average overlap (or Intersection over Union, IoU) between the ground truth and the prediction before a *failure*, averaged over all sequences. Failure is defined as the frame where the overlap between the ground truth and prediction dropped below 0.1 and did not increase above this threshold during the subsequent 10 frames. This is expressed as:

  $$A = \frac{1}{\sum_s \sum_a N^F_{s,a}} \sum_s \sum_a \sum_{i=1}^{N^F_{s,a}} \Omega_{s,a}(i),$$

  where $N^F_{s,a}$ is the number of frames before the failure starting from the anchor a in the sequence s, and $\Omega_{s,a}(i)$ is the overlap between the prediction and the ground truth bounding boxes at the frame i.

- **Robustness** R is the percentage of successfully tracked frames averaged over all sequences. Successfully tracked frames are all frames before a failure. This is expressed as:

  $$R = \frac{\sum_s \sum_a N^F_{s,a}}{\sum_s \sum_a N_{s,a}},$$

  where $N_{s,a}$ is the number of frames starting from the anchor a in the sequence s.

- **Precision** P is the percentage of frames whose Euclidean distance (in pixels) between centers of the ground truth and the prediction is lower than a given threshold, averaged over all sequences. A precision plot then depicts the precision metric as a function of different thresholds (typically 0-50 pixels). A threshold of 20 pixels was chosen as the representative precision included in Table 4.1. This is expressed as:

  $$P(\alpha) = \frac{1}{\sum_s \sum_a N_{s,a}} \sum_s \sum_a \sum_{i=1}^{N_{s,a}} [d_{s,a}(i) \le \alpha],$$

  where $d_{s,a}(i)$ is the Euclidean distance between centers of the prediction and the ground truth bounding boxes starting from the anchor a in the sequence s at the frame i.

- **Success rate** S is the percentage of frames whose overlap between the ground truth and the prediction is higher than a given threshold, averaged over all sequences. A success plot then depicts the success rate as a function of different thresholds. A threshold of 0.5 was chosen as the representative success rate included in Table 4.1. This is expressed as:

  $$S(\alpha) = \frac{1}{\sum_s \sum_a N_{s,a}} \sum_s \sum_a \sum_{i=1}^{N_{s,a}} [\Omega_{s,a}(i) \ge \alpha].$$

- **Area Under Curve (AUC)** is the area under the curve of the success plot. This metric ranks the trackers in Table 4.1. This is expressed as:

$$\mathrm{AUC} = \int_0^1 S(\alpha)\, d\alpha.$$

Figure 4.2 shows a plot of each tracker's accuracy/robustness metrics. Figure 4.3 depicts the precision and success plots. Table 4.1 summarizes all the metrics; for precision/success scores, the representative thresholds described previously are used. Table 4.1 also lists the average computation speed of the trackers, which is crucial to consider for resource-constrained environments.

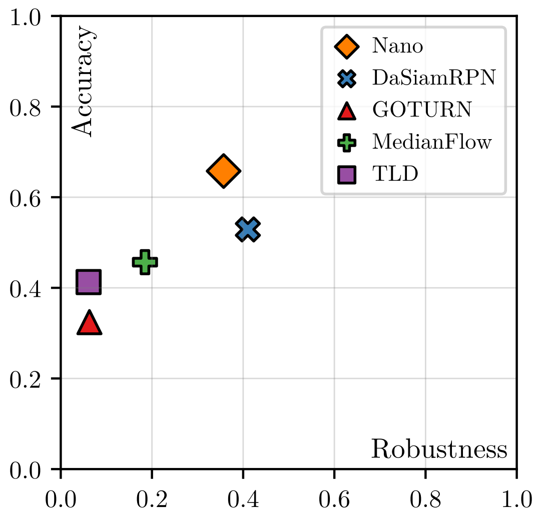

Figure 4.2. Accuracy/robustness plot of the bounding box trackers evaluated on chosen sequences from the UAV123 dataset.

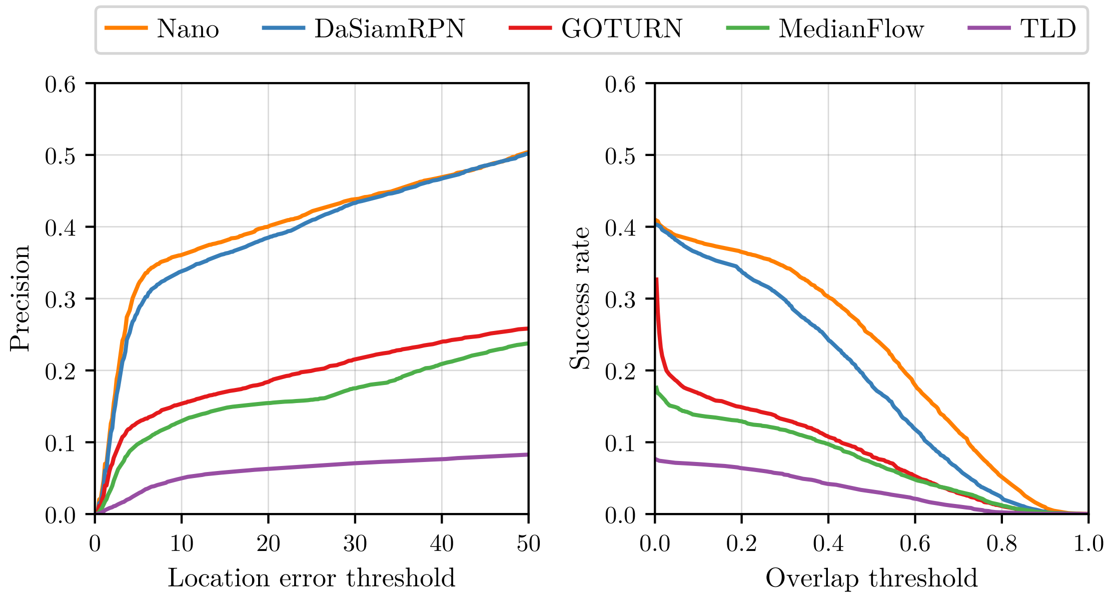

Figure 4.3. Plots of the precision and success metrics for the bounding box trackers evaluated on chosen sequences from the UAV123 dataset.

Tracker	Accuracy	Robustness	Precision	Success	AUC	FPS
Nano	0.658	0.357	0.400	0.250	0.220	108.5
DaSiamRPN	0.529	0.410	0.385	0.181	0.183	13.1
GOTURN	0.324	0.063	0.184	0.082	0.085	21.8
MedianFlow	0.457	0.185	0.155	0.072	0.074	623.7
TLD	0.413	0.061	0.063	0.032	0.035	42.7

Table 4.1. Summary of the different metrics of the bounding box trackers evaluated on chosen sequences of the UAV123 dataset. For each metric, red, green, and blue colors denote first, second, and third place, respectively. FPS was measured on the Intel Core i7-12700H processor with 14 cores and 20 physical threads. Note that FPS heavily depends on the platform.

Tracker	Window	Levels
MedianFlow	21×21	4
TLD	10×10	2

Table 4.2. Differences between hyperparameters of the MedianFlow and TLD trackers. “Levels” denote the number of levels in the image pyramid, which may significantly impact the tracker’s ability to capture large motions. “Window” denotes the size of the feature window in pixels, which may notably affect the tracker’s accuracy.

The results show that the best performant tracker is Nano, which only lacks robustness. DaSiamRPN achieving better robustness than Nano may be explained by the local-to-global search strategy for re-detecting the object of interest. Long-term tracking is not essential for the intended application, as the needed re-detection is executed by the LiDAR-based detection algorithm. Regarding computation speed, Nano runs at over 100 FPS on a multi-core CPU, which is suitable for real-time tracking onboard UAVs. The only tracker executing at a much higher FPS is MedianFlow, which, on the other hand, performs much worse than Nano in the other metrics.

Surprisingly, MedianFlow shows much better results than TLD at a lower computational cost. This may be explained by the fact that the hyperparameters of the MedianFlow tracker have been set to capture large movements, while the TLD tracker uses the default parameters of MedianFlow. Table 4.2 shows used hyperparameters. Hyperparameters of the trackers were not set the same due to implementation limitations, as the available Application Programming Interface (API) does not provide a way to change the parameters.

Notably, GOTURN performs the worst according to the VOT2020 metrics. However, GOTURN is the third best regarding precision/success metrics. This may be explained by the fact that the VOT2020 metrics are calculated before a failure, and any further recoveries are not considered in the calculation once failed. GOTURN might recover past 10 frames used in the failure definition, which might provide better results, reflected in the precision/success results. Even though GOTURN does not explicitly possess a re-detection module, the previous template (“what to track”) together with new locations (“where to look”) give a chance for recovery.

4.2 Sparse Feature Tracking

The Lucas-Kanade and CoTracker sparse feature trackers were evaluated on a manually annotated video sequence of 312 frames. The sequence consists of tracking a UAV from a camera placed onboard another chasing UAV. Figure 4.4 shows the first and last frames of the sequence together with the ground truth. This sequence, similar to those in the UAV123 dataset, presents significant challenges for tracking. The target UAV moves continuously, has rotating wings, and is filmed by a camera that also moves and rotates. These factors combine to create a complex scenario to thoroughly test the capabilities of sparse feature trackers in drone-hunting applications.

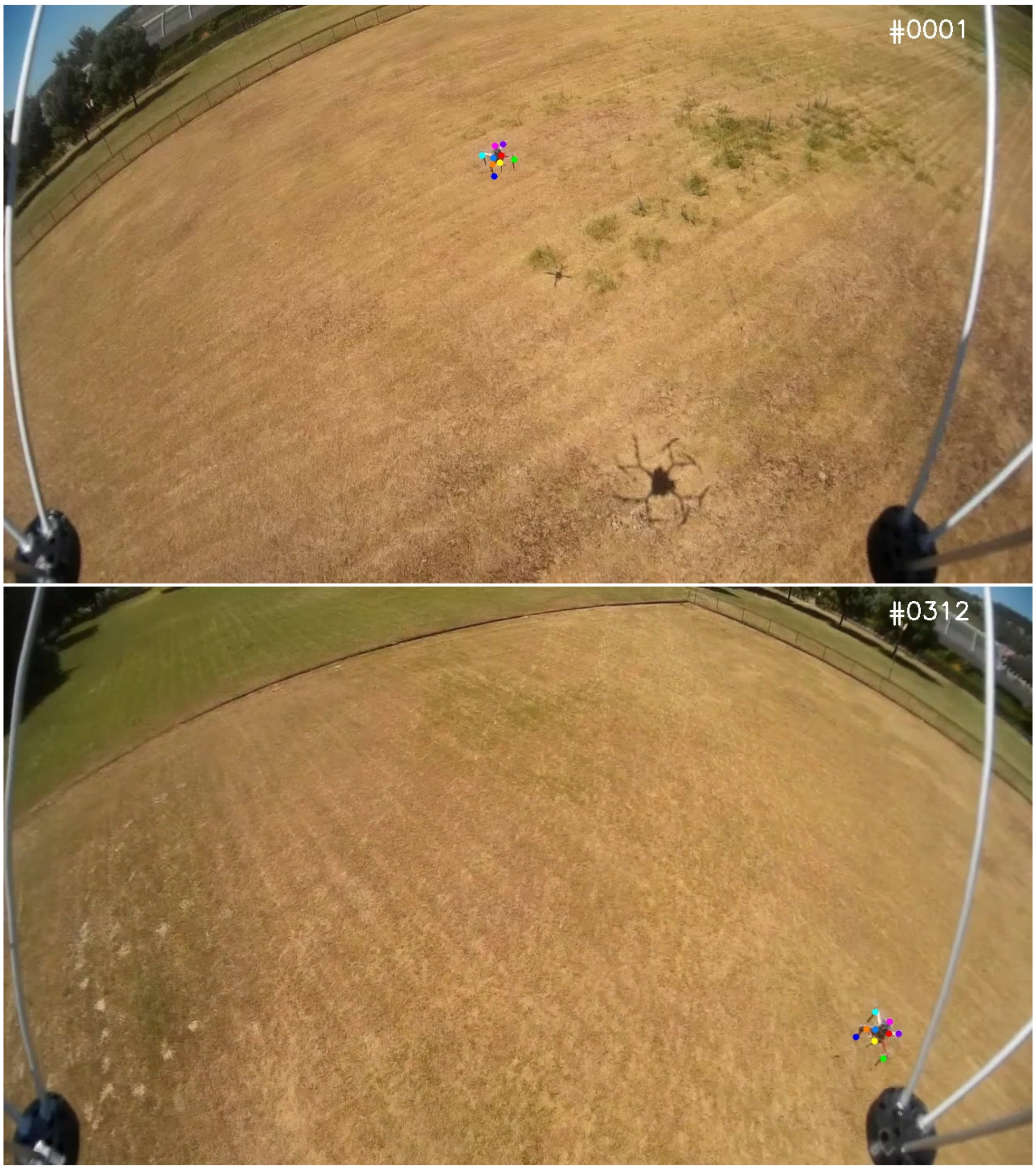

Figure 4.4. First and last frames of the manually annotated sequence for evaluating sparse feature trackers. Distinct color circles denote different ground truth point annotations.

The implementation of the Lucas-Kanade tracker utilizes a pyramidal approach from the OpenCV library, employing a four-level image pyramid and a 21×21 pixel feature window. These hyperparameters are similar to those used in the previously discussed MedianFlow tracker. For the CoTracker, the weights from the project's official GitHub repository[2] are used.

The calculated statistics for evaluating sparse feature trackers include accuracy and FPS. *Accuracy* A is the percentage of points whose Euclidean distance (in pixels) between the ground truth and the prediction is lower than a given threshold, averaged over all frames of the video sequence. This is expressed as:

$$A(\alpha) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{M_t}\sum_{i=1}^{M_t}[d_t(i) \le \alpha],$$

where N is the number of frames in the sequence, M_t is the number of non-occluded points at the frame t, and $d_t(i)$ is the Euclidean distance between the i-th non-occluded ground truth and predicted point at the frame t. A_{avg} is the average of $A(\alpha)$ across five thresholds: 1, 2, 4, 8, and 16 pixels.

An additional metric worth considering is occlusion prediction, where each tracker estimates whether a tracked point becomes occluded. While occlusion prediction is a valuable metric for evaluating tracker performance, it was not calculated in this thesis. This omission is primarily due to the Lucas-Kanade method's inherent limitation in providing such predictions. Although the CoTracker offers this capability, the absence of comparable data from the Lucas-Kanade tracker makes a fair comparison impossible. Therefore, while occlusion prediction remains a noteworthy consideration for future evaluations, it is not applicable in the current analysis.

Figure 4.5 shows the accuracy plot of the evaluated sparse feature trackers. Table 4.3 lists each tracker's average accuracy and computation speed. The results show that the best sparse feature tracker is the Lucas-Kanade, which demonstrates superior accuracy and excellent computation speed on multi-core CPUs. In contrast, the CoTracker is less accurate, which may be explained by several factors: the high-speed rotary wings of the tracked UAV, the non-stationary camera movement, and the changing appearance of the tracked UAV due to variations in illumination and viewpoint throughout the sequence.

The CoTracker's windowed inference approach, while sophisticated, poses challenges for rapid frame-by-frame prediction, particularly when the number of incoming frames is less than the inference window. This limitation and slow computation speed on CPUs make the CoTracker unsuitable for embedded applications but still valuable as a baseline for evaluating other point trackers.

Tracking points on highly dynamic objects with the LK method can be challenging, especially on UAVs with rotary wings, as it directly violates the assumptions of the LK algorithm. However, despite the disadvantages of the LK method, it is highly computationally efficient, especially for a low number of points. Moreover, it can be effectively parallelized by performing the same operation for each point, resulting in a linear speedup.

It is important to note that both trackers' accuracy metric heavily depends on the quality of the provided annotations. The short sequence was manually annotated to

[2] `https://github.com/facebookresearch/co-tracker`

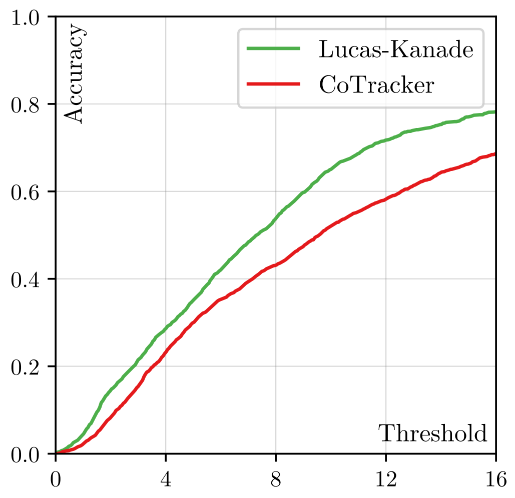

Figure 4.5. Accuracy plot of the sparse feature trackers evaluated on the manually annotated sequence.

Tracker	A_{avg}	FPS
Lucas-Kanade	0.359	2135.38
CoTracker	0.291	0.24

Table 4.3. Summary of the different metrics of the sparse feature trackers evaluated on the manually annotated sequence. FPS was measured on the Intel Core i7-12700H processor with 14 cores and 20 physical threads.

ensure reliability. However, to validate these results more comprehensively, developing a realistic simulation environment capable of generating accurate annotations would be beneficial. Such an approach would offer a controlled setting for evaluating trackers' performance and verifying findings from real-world data.

4.3 Multi-Camera Tracking

The implemented multi-camera system was evaluated in simulation, with the configuration shown in Figure 4.6. The MedianFlow tracker was used for each camera's independent tracker, with the hyperparameters described previously. The interception scenarios were generated using the guidance technique introduced in [17]. The evaluation of the multi-camera system revealed several challenges inherent to high-speed interception maneuvers. The critical problems identified include:

- **Rarity of Ideal Scenarios**: Many generated maneuvers are unusable because the ideal scenario — when an object is detected within the front camera's FOV and the front camera successfully exchanges information with the down camera — occurs infrequently.
- **Limited Frame Availability**: Generated scenarios typically contain only 20-30 frames in total, often with just 2-3 frames per camera during the interception maneuver. This scarcity of camera frames makes continuous tracking of the object extremely challenging.

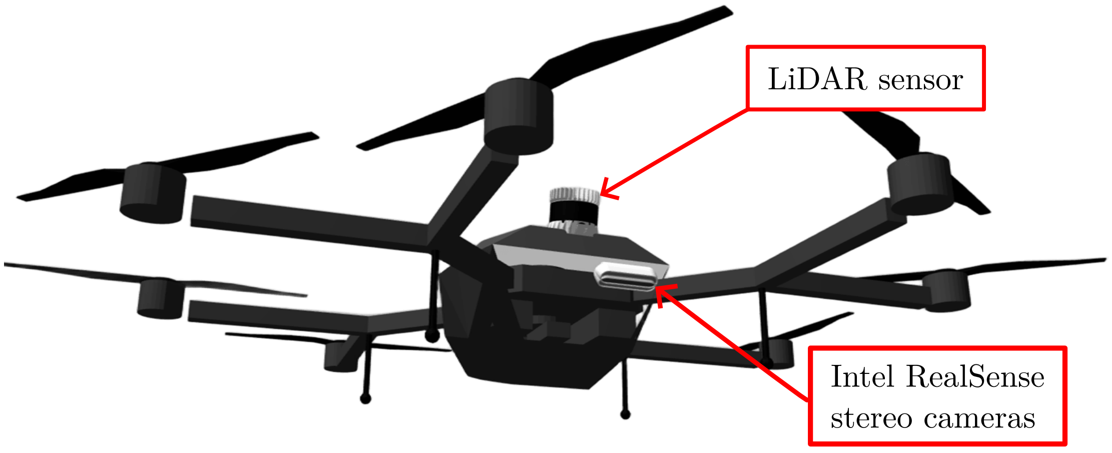

Figure 4.6. UAV configuration for evaluating the implemented multi-camera system in the simulation. The UAV is equipped with a 3D LiDAR sensor and a pair of Intel RealSense stereo cameras. In the simulation, the stereo cameras are placed at the same coordinate and orientation, differing only in the pitch angles: 45° for the front camera and 65° for the down camera.

- **Tracker Assumptions vs. Reality**: Standard trackers assume relatively smooth object motion between subsequent frames. However, this assumption breaks down during high-speed interception maneuvers, even in simulation environments.
- **Asynchronous Sensor Operation**: The system's sensors operate at different frequencies — LiDAR sensor at 10 Hz and cameras at 30 Hz — introducing temporal misalignment between incoming detections and camera images. While software synchronization is implemented, mismatches between incoming detections and the tracked object still occur, complicating the initialization of the front camera's tracker.
- **Camera Synchronization Issues**: The cameras in the system are not hardware-synchronized. To ensure proper system functionality, additional software-based synchronization is necessary. This software synchronization introduces its own complexities and potential for timing misalignments.

Figure 4.7 illustrates the behavior of the implemented multi-camera tracking system during a critical interception maneuver in one cherry-picked scenario. The results reveal that the front camera's initial bounding box demonstrates high accuracy, and the MedianFlow tracker successfully maintains object tracking despite the high-speed nature of the maneuver. After several frames, the front camera effectively shares information with the down camera, initializing its tracker. However, the down camera's tracker initially struggles, losing the object after just one frame and only resuming tracking upon receiving another exchange from the front camera. The second information exchange provides a more precise bounding box, enabling the down camera to track the object consistently until it leaves its FOV. Throughout the entire sequence, the front camera maintains successful object tracking. These observations highlight the system's capabilities and potential areas for improvement in multi-camera tracking during high-speed interception scenarios.

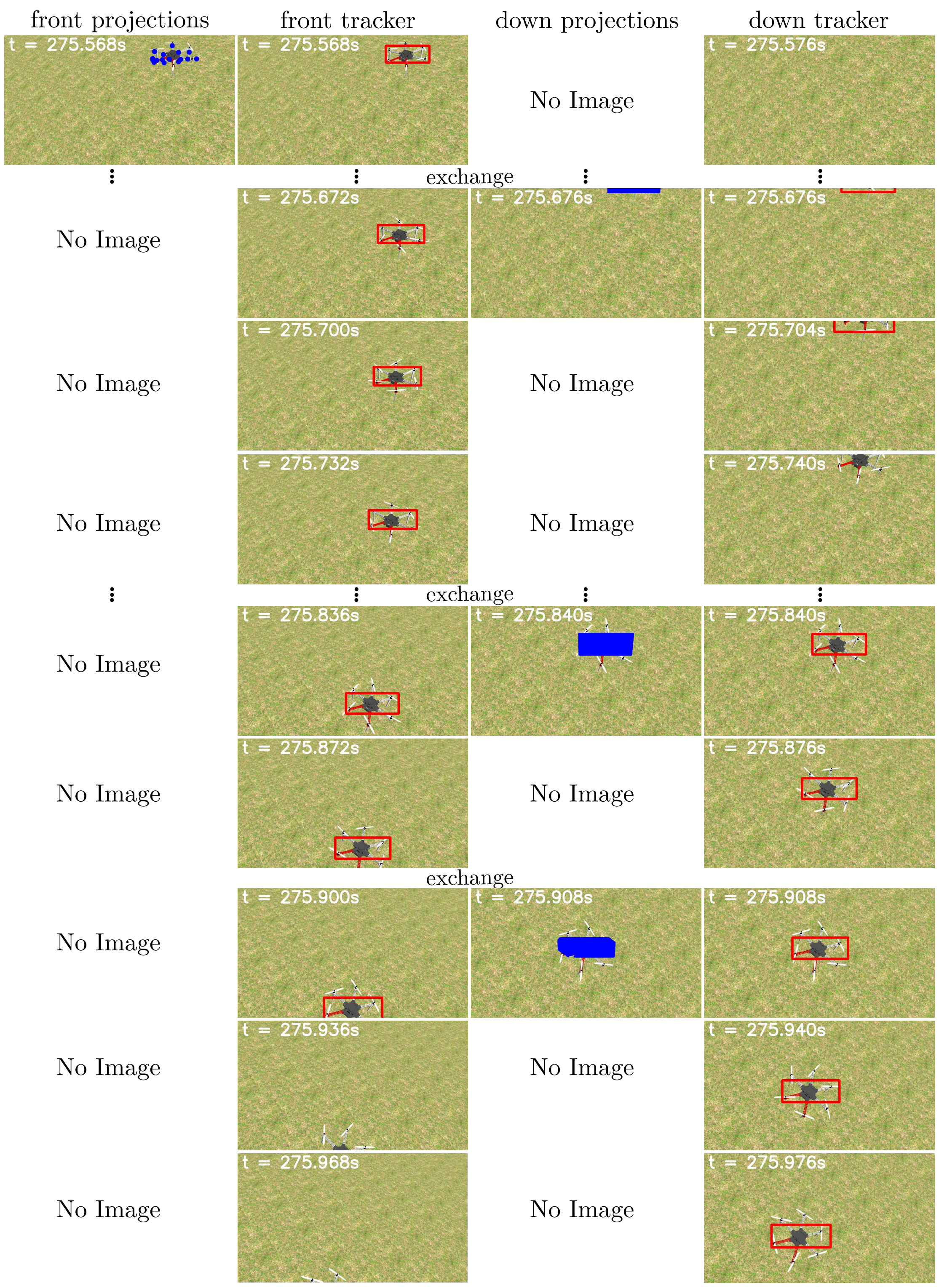

Figure 4.7. Behavior of the implemented system during the interception maneuver. "Front projections" denote the projected points from the LiDAR-based detector. "Down projections" denote the projected points by exchanging tracking information. "Front tracker" and "Down tracker" are independent trackers for each camera.

Chapter 5
Conclusion

This thesis explored several approaches to multi-camera visual tracking onboard UAVs to complement a LiDAR-based detection algorithm [21] in the context of drone-hunting. Various visual tracking methods were investigated, including sparse feature and bounding box tracking, highlighting each method's strengths and weaknesses. The evaluation of the bounding box trackers revealed trade-offs between computational cost and tracking performance. Among the considered bounding box trackers, the Nano tracker emerged as highly efficient and accurate for real-time deployment onboard UAVs. Furthermore, the feasibility of extending single-camera trackers into the multi-camera setting was examined using the projective geometry and the pinhole camera model.

The simulation-based evaluation of the implemented multi-camera system revealed significant challenges. The majority of the simulated trajectories were found to be impractical for immediate use, primarily due to the made assumptions about expected system behavior. The hypothetical ideal scenario for the interception maneuver proved to be excessively constraining, revealing the need for a more flexible approach in real-world applications.

Future research should prioritize the development of a more flexible approach to interception scenarios, moving beyond the current restrictive assumptions. Emphasis should be placed on creating a unified sensor integration framework, thoroughly exploring the fusion of LiDAR-derived spatial information with visual clues from cameras. Leveraging pre-calculated apriori maps to enhance security in specific areas presents an intriguing research opportunity. Moreover, the current reliance on projective geometry may be overly limiting, suggesting the need to explore alternative approaches for combining data from multiple cameras.

References

[1] ALVARADO, Ed. The commercial drone market in 2023: Insights and growth projections. In: *droneii.com* [online]. 2023. Available from `https://droneii.com/commercial-drone-market-2023`.

[2] BOUGUET, Jean-Yves. Pyramidal implementation of the Lucas Kanade feature tracker. *Intel Microprocessor Research Labs* [online]. 1999. Available from `http://robots.stanford.edu/cs223b04/algo_tracking.pdf`.

[3] BROMLEY, Jane, Isabelle GUYON, Yann LECUN et al. Signature verification using a "Siamese" time delay neural network. *Advances in Neural Information Processing Systems*. 1993, Vol. 6.

[4] CHU, Honglin. NanoTrack. In: *github.com* [online]. 2022. Available from `https://github.com/HonglinChu/NanoTrack`.

[5] HELD, David, Sebastian THRUN, and Silvio SAVARESE. Learning to track at 100 FPS with deep regression networks. In: *European Conference on Computer Vision*. 2016. pp. 749–765.

[6] HOWARD, Andrew, Menglong ZHU, Bo CHEN et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861v1*. 2017.

[7] JACOB, Benoit, Skirmantas KLIGYS, Bo CHEN et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018. pp. 2704–2713.

[8] KÁLAL, Zdeněk, Krystian MIKOLAJCZYK, and Jiří MATAS. Forward-backward error: Automatic detection of tracking failures. In: *20th International Conference on Pattern Recognition*. 2010. pp. 2756–2759.

[9] KÁLAL, Zdeněk, Krystian MIKOLAJCZYK, and Jiří MATAS. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2011, Vol. 34, No. 7, pp. 1409–1422.

[10] KARAEV, Nikita, Ignacio ROCCO, Benjamin GRAHAM et al. CoTracker: It is better to track together. *arXiv preprint arXiv:2307.07635v2*. 2023.

[11] KRISTAN, Matej, Aleš LEONARDIS, Jiří MATAS et al. The eighth visual object tracking VOT2020 challenge results. In: *Proceedings of the European Conference on Computer Vision Workshops*. 2020. pp. 547–601.

[12] KRISTAN, Matej, Aleš LEONARDIS, Jiří MATAS et al. The tenth visual object tracking VOT2022 challenge results. In: *Proceedings of the European Conference on Computer Vision Workshops*. 2022. pp. 431–460.

[13] KRISTAN, Matej, Jiří MATAS, Martin DANELLJAN et al. The first visual object tracking segmentation VOTS2023 challenge results. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023. pp. 1796–1818.

[14] LUCAS, Bruce, and Takeo KANADE. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence.* 1981. pp. 674–679.

[15] MORHUNENKO, Mykola. *Position estimation of a flying target from a camera onboard a UAV using visual tracking* [online]. 2024. Master's Thesis. CTU in Prague, Faculty of Electrical Engineering, Department of Cybernetics. Available from `https://dspace.cvut.cz/handle/10467/115332`.

[16] MUELLER, Matthias, Neil SMITH, and Bernard GHANEM. A benchmark and simulator for UAV tracking. In: *European Conference on Computer Vision.* 2016. pp. 445–461.

[17] PLISKA, Michal, Matouš VRBA, Tomáš BÁČA et al. Towards safe mid-air drone interception: Strategies for tracking & capture. *arXiv preprint arXiv:2405.13542v1.* 2024.

[18] PRINCE, Simon. The pinhole camera. In: *Computer vision: Models, learning and inference.* Cambridge University Press, 2012. pp. 359–388. ISBN 978-1107011793.

[19] PRINCE, Simon. Multiple cameras. In: *Computer vision: Models, learning and inference.* Cambridge University Press, 2012. pp. 423–456. ISBN 978-1107011793.

[20] SHI, Jianbo, and Carlo TOMASI. Good features to track. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* 1994. pp. 593–600.

[21] VRBA, Matouš, Viktor WALTER, Václav PRITZL et al. On onboard LiDAR-based flying object detection. *arXiv preprint arXiv:2303.05404v2.* 2023.

[22] WU, Yi, Jongwoo LIM, and Ming-Hsuan YANG. Online object tracking: A benchmark. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2013. pp. 2411–2418.

[23] YAN, Bin, Houwen PENG, Kan WU et al. LightTrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021. pp. 15180–15189.

[24] ZHU, Zheng, Qiang WANG, Bo LI et al. Distractor-aware Siamese networks for visual object tracking. In: *European Conference on Computer Vision.* 2018. pp. 101–117.

Appendix A
AI Tools

This appendix provides a brief overview of the Artificial Intelligence (AI) tools used in the research and writing process of this bachelor's thesis. The use of these AI tools aligns with the guidelines and permitted extent outlined in the "Framework Rules for the Use of Artificial Intelligence at CTU for Study and Teaching Purposes in Bachelor and Follow-up Master Studies" document (issued on 29th January 2024).

ChatGPT (OpenAI)[1], and **Gemini (Google DeepMind)**[2] have been used as general-purpose tools, facilitating self-study, text reformulation, literature search, and data visualization. **Elicit**[3], **Consensus**[4], and **Semantic Scholar**[5] have been used for the initial literature search. **SciSpace**[6] has been used for literature review with the AI copilot.

[1] https://chat.openai.com
[2] https://gemini.google.com
[3] https://elicit.com
[4] https://consensus.app
[5] https://semanticscholar.org
[6] https://typeset.io

Appendix B
Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under Curve
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FOV	Field of View
FPS	Frames Per Second
GPU	Graphics Processing Unit
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
LK	Lucas-Kanade
TLD	Tracking-Learning-Detection
UAV	Unmanned Aerial Vehicle
VOT	Visual Object Tracking