Motion-based MAV Detection in GPS-denied Environments

MSc Thesis

Erik Vroon

Technische Universiteit Delft

TU Delft

Motion-based MAV Detection in GPS-denied Environments

MSc Thesis

by

Erik Vroon

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Friday July 16, 2021 at 9:00.

Student number:	5173264	
Project duration:	October 1, 2020 – July 16, 2021	
Thesis committee:	Prof. dr. G.C.H.E de Croon,	TU Delft, chair, supervisor
	Dr. ir. C. de Visser,	TU Delft, independent examiner
	Dr. ir. J. Guo,	TU Delft, external examiner
	Ir. J. Rojer,	TNO, supervisor, additional examiner

Preface

Programming a machine to interpret the world around us, similar to how humans perceive it, is a challenging task. By developing robots that are aware of their surroundings, as well as others of their kind, we can increase their safety and extend their capabilities. This project solves a very small part of this puzzle by enabling drones to detect other drones in the vicinity based on their apparent motion. The code developed for this thesis can be found publicly online [1].

I am thankful for TNO for letting me perform my graduation project in collaboration with a company. I express my gratitude to my supervisors Jim Rojer and Dr. Guido de Croon for their patience and support, as well as their flexible mindset in these exceptional times. Their sharp feedback has been crucial to create this thesis. Additionally, I would like to thank Nicolas Boehrer for sharing his expertise in the field of computer vision. Finally, I am thankful for my friends and family who provided me with support and motivation.

[1] `https://github.com/evroon/mav-detection`

Acronyms

BRIEF	Binary Robust Independent Elementary Features
CI	Covariance Intersection
CNN	Convolutional Neural Network
CoSLAM	Collaborative SLAM
EKF	Extended Kalman Filter
FAST	Micro Air Vehicle
FN	False Negative
FoE	Focus of Expansion
FoV	Field of View
FP	False Positive
FPR	False Positive Ratio
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
IR	Infrared
ISR	Intelligence, Surveillance and Reconnaissance
KF	Kalman Filter
LDOF	Large Displacement Optical Flow
LED	Light Emitting Diode
LIDAR	Light Detection And Ranging
LoS	Line of Sight
MAV	Micro Air Vehicle
MCS	Motion Capture Systems
MEMS	Micro Electrical Mechanical Sensors accelerometers
MOG	Mixture of Gaussians
OF	Optical Flow
ORB	Oriented FAST and Rotated BRIEF
RADAR	RAdio Detection And Ranging
RANSAC	Random sample consensus
RF	Radio Frequency
RMSE	Root Mean Squared Error
SfM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SLAMMOT	Simultaneous Localization, Mapping and Moving Object Tracking
SNR	Signal to Noise Ratio
SURF	Speeded Up Robust Features
TP	True Positive
TPR	True Positive Ratio
TTC	Time-To-Contact
UKF	Unscented Kalman Filter
UV	Ultraviolet
VIO	Visual Inertial Odometry
VO	Visual Odometry

Contents

List of Figures

List of Tables

Introduction

Nowadays, drones are becoming more and more common. To further extend the capabilities of drones, groups of drones, called swarms, are now investigated. These swarms can perform tasks that individual drones are unable to execute. For various tasks of the swarm, the relative locations of other drones in the swarm are needed. The standard method of obtaining the locations of MAVs, is by use of a Global Navigation Satellite System (GNSS). Unfortunately, these GNSS signals can be blocked, jammed, spoofed or interfered by multipath effects, rendering them unavailable. Computer vision is a suitable alternative to localize drones, because cameras are small and provide a wealth of information.

Typically, research papers investigate the use of appearance-based neural networks that detect drones in a single image. However, these methods depend on the shape and size of the drones. An alternative is to use the motion information in video feeds to detect drones. In this thesis, the apparent motion of objects in a video, called optical flow, is used to detect drones. Optical flow offers various advantages compared to other vision-based methods. Firstly, the method is potentially more robust against background clutter. Moreover, when using optical flow, the method can be independent on the appearance of the object to detect and therefore works for every moving object.

Simulations were made to acquire the data needed to validate the performance of the object detector. These simulations are made in AirSim and output a large amount of visual data, such as the camera image, depth image, a segmentation mask of the drone to detect and ground truth optical flow images. The main conclusion of this project is that the object detector has a high accuracy , but only if certain assumptions are met. Specifically, the motion of the object to detect must be large enough and different compared to the background motion.

This thesis consists of a scientific paper (part I), the preliminary literature study (part II) and finally an appendix containing recommendations and additional results of the object detector and the Focus of Expansion estimation.

I

Scientific Paper

Motion-based MAV Detection in GPS-denied Environments

Erik Vroon *[1], Jim Rojer †[2], and Guido C. H. E. de Croon ‡[1]

[1]MAVLab, Control and Operations, Faculty of Aerospace Engineering, TU Delft, The Netherlands
[2]Netherlands Organisation for Applied Scientific Research (TNO), The Hague, The Netherlands

Abstract

Drones need to be able to detect and localize each other if they are to collaborate in multi-robot teams or swarms. Typically, computer vision methods based on visual appearance are investigated to this end. In contrast, in this paper, a method based on dense optical flow (OF) is developed that detects dynamic objects. This is achieved by comparing the flow vectors of dense OF with the direction to the Focus of Expansion (FoE) in the image plane. A simulation in AirSim is developed to validate this approach and to create datasets for motion-based object detection of MAVs. This simulation includes ground-truth FoE, depth, OF and IMU data. The results show that this method performs well if the OF vector's magnitude is large enough and its angle is sufficiently different from those of static world points. We expect that the presented method will serve as a useful baseline for deep learning methods that use dense optical flow as input.

1 Introduction

Nowadays, Micro Air Vehicles (MAVs) are becoming more and more common. Reasons for their popularity include their high maneuverability, vertical take-off capabilities and ability to perform tasks that humans cannot endure [1]. To further enhance the capabilities of MAVs and overcome the individual limitations of MAVs, swarms of MAVs were introduced. To enable the proper functioning of the swarm, sensing of the environment and the other MAVs is paramount. In particular, the relative locations of MAVs inside the swarm are needed for collision avoidance and swarm coordination [2]. The most basic and robust method of obtaining the locations of other MAVs, is by exchanging positions obtained from Global Navigation Satellite System (GNSS) signals. However, GNSS signals are not always available, for example when the signals are blocked, spoofed, jammed or distorted by multipath effects.

Computer vision is a promising alternative, because cameras are small/lightweight and provide a vast amount of information [2]. There are two main types of approaches solving the relative localization problem. The first is to create a shared map of the environment and have the MAVs exchange their location in this map. Simultaneous Localization and Mapping (SLAM) is a wide field of research that targets the first type [3]. The second type of relative localization focuses on the detection of the MAVs themselves. This process is often simplified by the use of physical devices called markers. These can be either infrared (see the work of Walter et al. [4]) or ultraviolet (see the work of Roberts et al. [5]) LEDs, or colored objects. Markers require specific hardware changes to the device, which may not always be desirable. Markerless detection represents a more difficult problem. Some methods quite successfully rely on stereo vision [6, 7]. Of course, for resource minimization, methods using a single camera are of interest. Currently, the main approach with a single camera is to employ deep neural networks that detect other MAVs in a single image [8, 9]. These neural networks show promising results, but it is not yet clear how well the trained networks can deal with cluttered backgrounds. Moreover, if the drone appearance or environment changes substantially with respect to the training set, retraining may be necessary.

In order to obtain a solution that does not depend on stereo vision or markers and that is more generic compared to appearance-based methods, it may be useful to use optical flow. Optical flow has multiple advantages over its alternative vision-based methods. Firstly, MAVs will possibly be detected in situations where they are barely distinguishable for the human eye due to background clutter. Additionally, optical flow based methods are less dependent on shapes and appearances of MAVs compared to other methods. Finally, optical flow can offer a larger maximum detection range compared to active markers.

Some papers incorporate motion into their appearance-based neural network, the so-called hybrid methods, such as the work by Yoshihashi et al.

[10], where the temporal information improves the performance of the object detector in situations where there is little contrast between the background and foreground. Nonetheless, it is a ground-based method and uses static cameras, which is less challenging compared to the situation where the observer is moving.

To our current knowledge, the research done by Li et al. [11] is the only work using purely optical flow without artificial neural networks for the detection of MAVs from a moving observer in the air. With their method, they were able to detect other MAVs, even when they were barely visible because of their size and the cluttered background. It has approximately the same detection accuracy (87%) as appearance-based neural networks applied to MAV detection (maximum accuracy of approximately 90%) [8, 12]. However, it is based on some assumptions. The method of Li et al. is based on a combination of background subtraction and Lucas-Kanade optical flow. The background subtraction process assumes that the tracked objects have a very different motion compared to a distant background, of which the motion is modelled with a homography transformation.

This paper will focus on using motion-based object detection to detect MAVs from onboard a moving MAV in more general, 3D environments. Specifically, we present an optical-flow-based algorithm to detect dynamic objects in video feeds from a moving camera. This is done by comparing the flow vectors with the direction to the Focus of Expansion (FoE) in the image plane. This method is applied to simulations run in AirSim [13]. These simulations output ground-truth FoE, depth, optical flow and IMU data, which are valuable for the development and validation of motion-based object detection techniques. The algorithm involves a non-learning image processing pipeline that is based on knowledge of the properties of the (derotated) optical flow field. We believe that eventually deep-learning-based methods could achieve higher performance if they also exploit optical flow, and expect that the presented, completely comprehensible pipeline will be a useful benchmark method.

In this paper, all moving objects (except moving clouds) are assumed to be MAVs. To output only MAVs in an environment with other types of dynamic objects, the pipeline has to be extended with a step that differentiates the MAVs from other moving objects.

2 Detection method

The object detection method is illustrated in figure 2. First, the optical flow (OF) field is derotated using the rotation rates of the IMU. The location of the FoE is calculated using the derotated flow. FoE is the point where the translational flow is 0. This is the motion direction of the camera. All static points in the environment move away from the FoE. Points that are closer to the camera in terms of depth, have larger flow. Points that are further away from the FoE have larger flow as well. Dynamic objects may move in other directions. Then the associated flow vectors do not point away from the FoE. Unfortunately, they may move away from the FoE leading to flow that is similar to static objects. The angle κ between the vector pointing towards the FoE and the flow vector is calculated, as illustrated in figure 1. The larger κ, the more likely a pixel belongs to an object moving relative to the camera. In the following subsections, the individual steps of the method are explained in detail. All code used to reproduce this method and its results can be found publicly online[1].

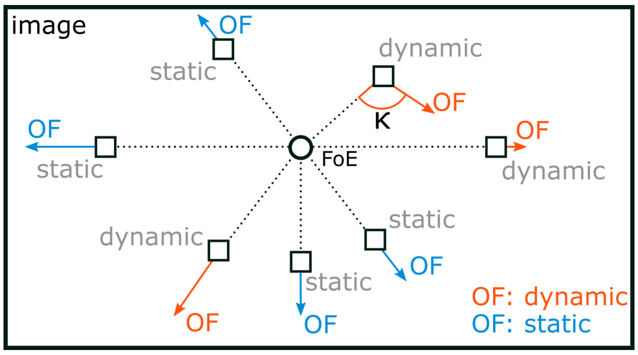


Figure 1: Illustration of κ for a camera moving forward. The κ angle denotes the difference between angle of the vector pointing at the FoE and the angle of the flow vector. For pixels of static objects, κ is approximately zero. For dynamic pixels, κ is non-zero, except when the object moves away from the FoE.

2.1 *Calculating optical flow*

As the object detection method relies on optical flow, an accurate dense optical flow estimator must be used. In figure 3, four neural networks estimating optical flow are compared. They illustrate that on the MIDGARD [14] dataset, LiteFlowNet [15] and Maskflownet [16] perform worse compared to RAFT [17] and FlowNet2 [18]. For all networks, the default weights were used. By visual inspection, FlowNet2 appears to perform best for small moving objects. Therefore, FlowNet2 is used for the results in the rest of this paper.

2.2 *Derotation*

Derotation has to be applied to the optical flow field to estimate an accurate FoE. The derotation technique in this paper is based on the work of Dinaux et al. [19]. The derotation vector per pixel coordinate can be calculated from equation 1 describing optical flow (u, v). See

[1]`https://github.com/evroon/mav-detection`

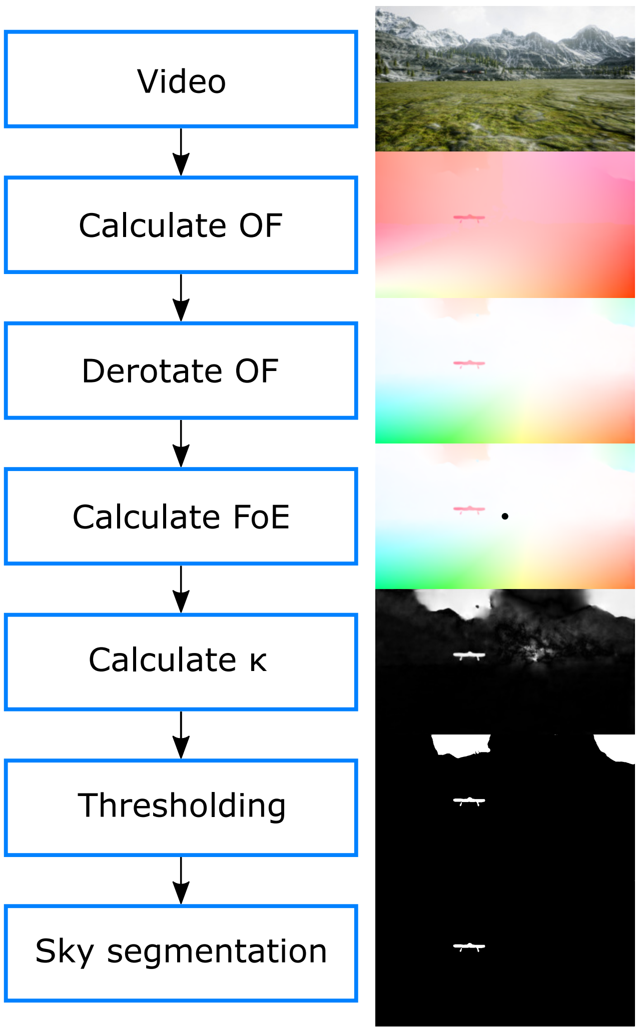


Figure 2: Pipeline of detecting moving objects with visualizations per step.

Longuet-Higgins et al. [20] for the derivation.

$$
\begin{aligned}
u &= -\frac{U}{Z} + x\frac{W}{Z} + Axy - Bx^2 - B + Cy = u_T + u_R \\
v &= -\frac{V}{Z} + y\frac{W}{Z} - Cx + A + Ay^2 - Bxy = v_T + v_R
\end{aligned}
\tag{1}
$$

The optical flow can be split into two factors: the rotational (u_R, v_R) and translational (u_T, v_T) parts. The rotational part is only dependent on the pixel coordinate and rotational rates (A, B, C) of the camera. Therefore, the structure of the scene (in particular, depth) has no influence on the rotational part of optical flow. The Inertial Measurement Unit (IMU) of an MAV can be used to measure the rotational rate.

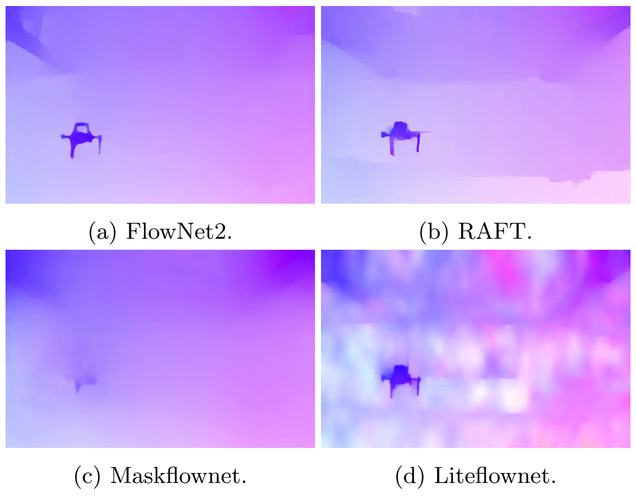


Figure 3: Different neural networks estimating optical flow compared using the MIDGARD [14] dataset.

2.3 Calculation of the FoE

The Focus of Expansion (FoE) is the point where all flow vectors point towards or originate from when an observer moves through an environment. This point can lie outside the camera's Field of View, but in this paper it is assumed to lie in the image plane. Nonetheless, the method does work for FoEs outside the Field of View.

The FoE is calculated as presented in figure 4. First, two optical flow vectors are randomly sampled. The intersection of the two vectors is calculated. This process is repeated N times, where N equals 1000. A RANSAC scheme [21] is applied to the set of intersections to make it more robust against outliers. The RANSAC method calculates a location in the image where most intersections have a distance to this point that is lower than a certain threshold. The resulting location is taken as the location of the FoE.

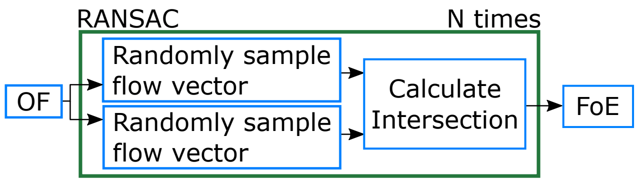


Figure 4: FoE method flowchart.

2.4 Sky segmentation

In outdoor environments, clouds in the sky can also move independently from the camera and generate substantial flow. Therefore, we segment clouds and sky by appearance and mask them out from the result. To this end, we use HRNet-OCR [22] with the default weights trained on the Cityscapes dataset [23]. By comparing the depth buffer from AirSim with the segmentation mask for the sky, one can validate the performance of the segmentation. Because of the visual simplicity of the environment in AirSim, the TPR of the sky segmentation is at least 99.5% and the FPR is less than 0.1%. The sky segmentation is performed at half the resolution of the captured images from AirSim, to reduce memory and computational effort of the GPU.

2.5 Thresholding and detection output

The output of the algorithm is based on the angle κ as illustrated in figure 1. The larger κ, the more likely it is that that pixel belongs to an object moving relative to the observer. Pixels with a κ angle larger than 15° are marked as moving objects. Out of these marked pixels, flow vectors with a magnitude smaller than 1 pixel/frame are discarded, because the angle of such vectors is sensitive to noise. Similarly, all pixels belonging to the sky are assumed to be stationary.

However, the threshold on κ can be more substantiated by analyzing how the error in the angle of the flow vectors behaves for various magnitudes of flow. One would expect that the error of the estimated OF direction increases for decreasing OF magnitude. This is the case, as shown in figure 5. For 100 FlowNet2 images, the radial error with respect to the ground truth OF data is plotted for all pixels (except the sky) versus the magnitude of the OF. The white line of $0.25 \pm (0.5 + \frac{8}{|\mathrm{OF}|})$ is fitted manually. The flow magnitude and value of κ that lie in the area between the upper and lower parts of this function, are discarded. Additionally, flow vectors with a magnitude lower than 0.5 pixels/frame are removed. The performance difference when using this 'dynamic' method of thresholding depending on the flow's magnitude is presented in the results section (see figure 10).

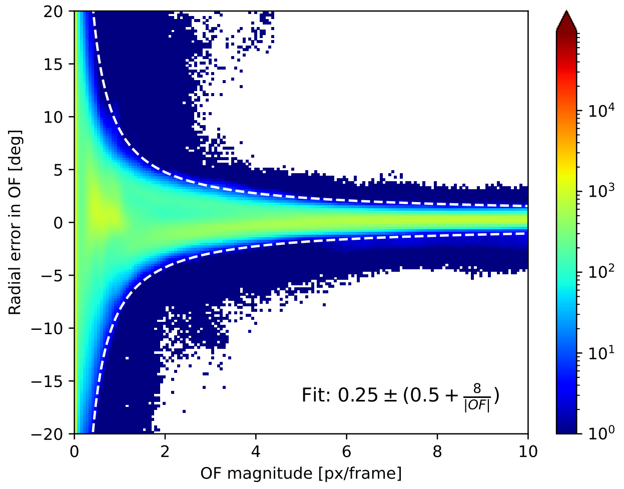


Figure 5: Histogram of the radial error in FlowNet2 (compared to the ground truth OF) versus the magnitude of the OF. Averaged over 100 OF fields.

3 AirSim

Simulations in AirSim [13] are carried out for various reasons. Most importantly, simulations can provide ground truth optical flow and FoE data that cannot be retrieved in real life. The ground truth optical flow makes it easier to develop a motion-based object detector, because the ground truth optical flow has no noise or artifacts. Simulations also enable validation of the algorithm on low level, by for example comparing the FoE estimation with the ground truth FoE. Specifically, AirSim is chosen because of its realistic rendering and support for MAVs, including various simulated sensors.

3.1 Environments

One environment is used in AirSim: LandscapeMountains[2]. LandscapeMountains is a freely available project from Epic Games, the publisher of Unreal Engine. It was chosen because of its realism, while at the same time being not too demanding. To diminish the influence of visual effects on the estimation of optical flow and the performance of the object detector, most of these influences were removed from the simulation. All moving actors (gates to fly through, birds) are made invisible. The clouds are translated vertically by 500m such that they appear above the terrain. Additionally, to avoid reflections, the ice is replaced by a grass material and the fog is disabled. This limits the method to a set of real-world environments, but in a large range of applications these assumptions can still be considered valid. The only visible visual effect is the shadow of the terrain and MAVs.

3.2 MAV control

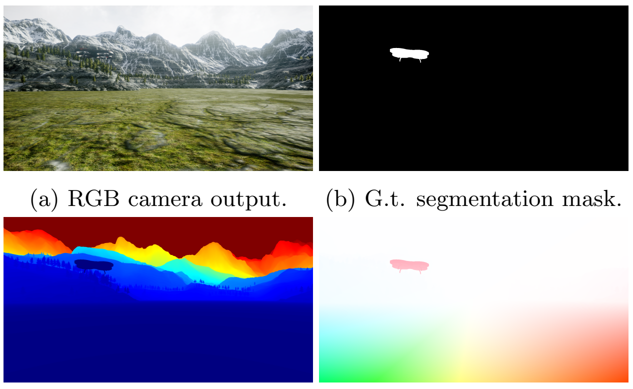
The MAVs are controlled using Python scripts. A loop is run for each simulation configuration, in which the MAVs are controlled and the data from AirSim is captured. First, the control inputs are calculated for the MAV to detect and the observing MAV. The time is advanced for 43ms (23Hz) and lastly, the data from AirSim is collected. The simulation is paused while obtaining the data of AirSim, such that the IMU data and camera frames are taken at the same timestep. The MAVs follow their flight path with a maximum deviation of 0.14m.

Two types of sequences are recorded. Firstly, collision courses, where the MAVs fly towards the same point at the same time at 4m/s. Secondly, sideways trajectories in which one MAV moves sideways in front of the observing MAV, which moves forwards at 4m/s.

[2]`https://www.unrealengine.com/marketplace/en-US/product/landscape-mountains`

3.3 Data acquisition

There are three visual outputs of the simulations: the RGB camera image, the depth in the camera image and the segmentation mask of the MAV inside the images. These three outputs are taken from the same camera, so all use the same projections. These outputs are shown in figures 6a to 6c. The camera image and segmentation mask are saved as PNG files, while the depth image is saved in AirSim's pfm format, enabling the use of floats. Additionally, sensor data is stored of both MAVs. This includes IMU and GPS data, but also contains collision data, the control inputs, FoE coordinates and camera properties. The ground truth FoE is calculated using the view projection matrix of the observer's camera and the observer's velocity vector. The images are collected at a resolution of 1920x1024 pixels with a framerate of approximately 23Hz. The field of view of the camera is 90° and there is no distortion or noise in the image.


(a) RGB camera output. (b) G.t. segmentation mask.

(c) Depth output. (d) Ground truth optical flow.

Figure 6: The different ground truth (g.t.) output frames captured in AirSim (a-c) and the g.t. optical flow (d) calculated from the depth output.

3.4 Ground truth optical flow

AirSim has no built-in method of calculating dense ground truth optical flow. However, it can be calculated from the depth image and the viewprojection matrix of the camera. This method is based on the work of Mayer et al. [24]. A visualization of the ground truth optical flow is shown in figure 6d and the steps of the method are shown in figure 7. Using the depth image, one can deduce the 3D world positions of all projected pixels by multiplying the inverse of the viewprojection matrix with the homogeneous pixel coordinates. This will result in a point cloud. From these 3D points, one can calculate their 3D positions one timestep ago. Finally, by applying the viewprojection matrix of the previous frame to the 3D points, one obtains the 2D coordinates of the original pixels one timestep ago. The difference between the original and the reprojected coordinates yields the ground truth OF.

The optical flow calculation has some limitations. For example, the flow of visual effects is not taken into account. This includes shadows, animations of vegetation, reflections/refractions etc. This limitation can be avoided by using environments without reflecting surfaces and removing dynamic objects.

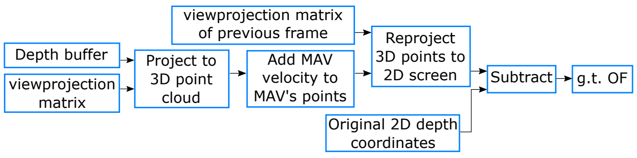

Depth buffer
viewprojection matrix
Project to 3D point cloud
viewprojection matrix of previous frame
Add MAV velocity to MAV's points
Reproject 3D points to 2D screen
Original 2D depth coordinates
Subtract
g.t. OF

Figure 7: Flowchart for calculating the ground truth OF.

3.5 IMU

The IMU is modeled using the default IMU in AirLib, the library implementing MAV dynamics and sensors inside AirSim. The biases and random walks of the gyroscope and accelerometer are set to zero. The IMU data is used for derotation of the optical flow field, as explained in subsection 2.2.

3.6 Overview of parameters

An overview of all parameters for the simulations and the object detection method is shown in table 1.

Table 1: Parameters of the simulation and method.

Parameter	Value
Resolution	1920x1024 px
Framerate	23 Hz
Field of View	90°
Observing MAV speed	4m/s
Fixed OF magnitude threshold	1 px/frame
Fixed OF radial threshold	15°
Number of collision course sequences	6
Number of sideways sequences	9
Number of FoE validation sequences	3

4 Results

This section will present the results in terms of performance on the FoE estimation and object detection for the simulations in AirSim.

4.1 AirSim

Because the accuracy of the object detection depends on the quality of the FoE estimation, the error between the estimated and ground-truth FoE is analyzed for different situations. A histogram of the FoE errors for one sequence is shown in figure 8. This is recorded for an MAV moving (without rotation) at 4 m/s with an FoE

20 pixels from the left and right edges of the image and an FoE in the center. Two characteristics are notable. For a forward moving MAV, the estimated FoE is on average slightly offset upwards (by 7.2 pixels) and to the right (by 2.8 pixels), but this is small compared to the total resolution of the image and therefore negligible for the majority of all pixels. Moreover, the location of the FoE affects the mean of the x distribution slightly, as the estimated FoE tends towards the center.

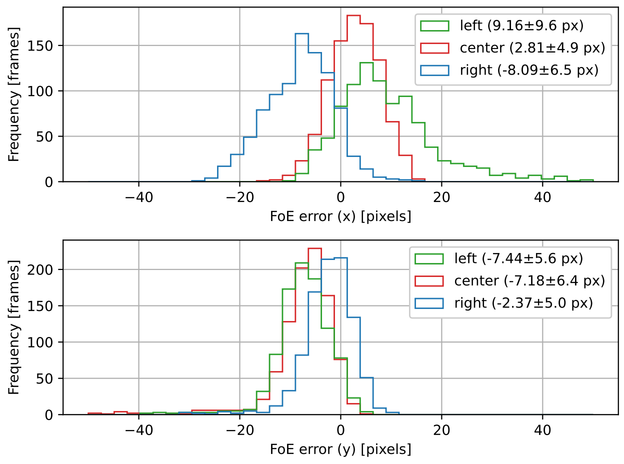


Figure 8: Histograms showing the error (in x and y direction) between the g.t. FoE and estimated FoE, for a FoE in the far left (at $x = 20$ px), center (at $x = 960$ px) and far right (at $x = 1900$ px) part of the image. The legend includes the means and standard deviations of the distributions.

The performance of the object detection method is determined by the True Positive Rate (TPR) and the False Positive Rate (FPR). TPR is the percentage of pixels from dynamic objects that are identified as dynamic object pixels. FPR is the percentage of pixels from static objects that are identified as dynamic object pixels. Ideally, one would have a large TPR for a very small FPR. In this case, the FPR is always relatively small, but the TPR varies considerably. This is shown in figure 9, where the TPR is plotted against κ for various speeds of the MAV to detect. As can be seen, the object detector is less accurate for slower moving objects.

The relation between the TPR/FPR and the magnitude of the OF of the detected object is presented in figure 10). The average TPR for κ between 180° and 90° is taken as measure of performance. It is clear that lower OF magnitudes decrease the TPR, but the FPR is unaffected. As hypothesized in section 2.5, a threshold that is dependent on the magnitude of the OF vector (a dynamic threshold) indeed results in a higher TPR for slower moving objects. However, this also increases the

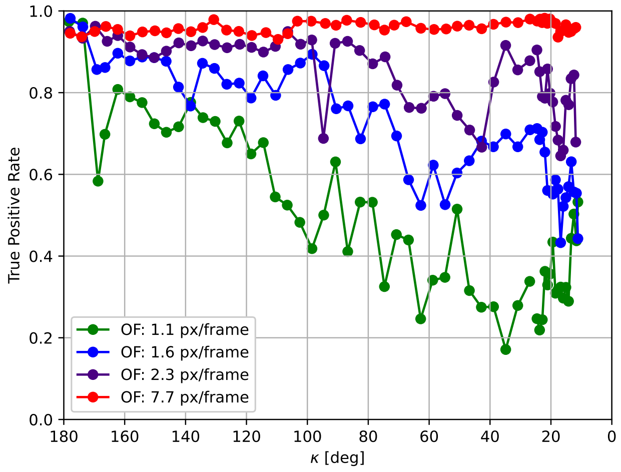


Figure 9: TPR vs κ, where MAV to detect moves from left to right with four different speeds (thus four magnitudes of OF) at a relative distance of 5m, decreasing κ from 180° to 0°. A dynamic threshold is applied.

FPR to 0.5% - 2.0%, which could be considered acceptable depending on the application. In situations where the object to detect has a large OF vector, a fixed threshold would be more suitable.

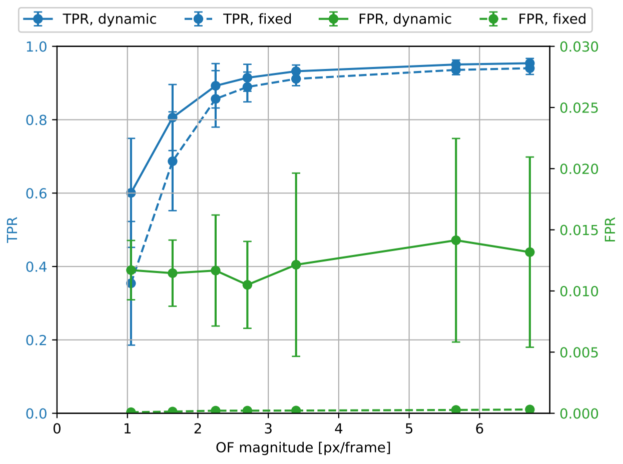
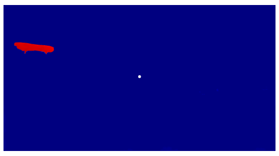
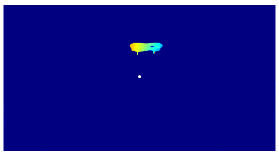


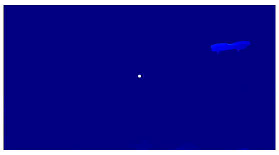
Figure 10: TPR and FPR vs the magnitude of the OF of the MAV to detect for $\kappa > 90°$ using a fixed and dynamic threshold.

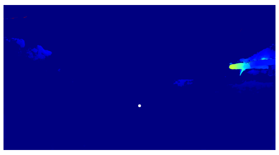
Additionally, lower values of κ degrade the performance of the object detector. This is illustrated in figure 11, in which the angle κ is visualized. A higher intensity in the image indicates a higher value of κ, meaning that the flow vector is not pointing towards the FoE. Thus, such a flow vector does not only correspond to the flow created by the translation of the camera, but also to the motion of the object belonging to that pixel. In figure 11a, κ is large and therefore the MAV is easy to detect. In figure 11b, the MAV is more challenging to detect and in figure 11c, the method is unable to detect the MAV as κ is close to zero.

To test the method in more complex circumstances, data was recorded for a collision course where the flight paths of the MAVs cross at an angle of 75°, shown in figure 11d. In this case, the MAV to detect remains at the same location in the image during the sequence, but becomes closer and therefore larger in the image. It can be seen that the right part has a κ angle close to zero. However, using a dynamic threshold, the TPR is still high (0.98) at the cost of a relatively high FPR ($2.8 \cdot 10^{-2}$). Unfortunately, this is only the case for short distances. For larger distances, the magnitude of the flow is too small to properly estimate κ.


(a) $\kappa \approx 180°$. TPR: 0.97, FPR: $6.2 \cdot 10^{-3}$.


(b) $\kappa \approx 90°$. TPR: 0.95, FPR: $4.3 \cdot 10^{-3}$.


(c) $\kappa \approx 0°$. TPR: 0.93, FPR: $1.5 \cdot 10^{-2}$.


(d) CC. TPR: 0.98, FPR: $2.8 \cdot 10^{-2}$.

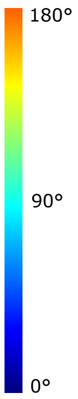


Figure 11: κ displayed for various situations. In (a) to (c), the MAV moves sideways from left to right. In (d), the observer and target are on a collision course (CC) of 75°. The white dot represents the FoE. A dynamic threshold is applied to calculate the TPR and FPR.

5 Discussion and Conclusion

We have introduced an optical-flow-based algorithm for detecting other moving objects, where our interest lies in the detection of other drones. The object detection method in this paper proves to work successfully if the angle of the optical flow vector of the object to detect is sufficiently different from the background flow, as illustrated in figure 11. This means that objects moving towards the FoE, which are crossing the flight path of the observer and are thus considered dangerous, can be successfully detected. Although the method is based on assumptions of the OF, it does not assume a specific appearance of the moving object, which makes it suitable for a wide range of applications.

The method in this paper has the following limitations. Most importantly, the method is based on motion, so if the observer is stationary or the object to detect has no optical flow as observed by the observer, detection by means of flow direction will not succeed. Therefore, MAVs on head-on collision courses can not be detected in this way because they have the same flow field as the surroundings, but with a larger divergence. One could utilize the divergence of the OF field to detect head-on colliding objects (just as for static objects).

Another limitation is the computational effort of our current implementation. FlowNet2 runs on approximately 1.7 Hz on an RTX 2070 for 1920x1024 images. This would be too slow to use in real-time on MAVs themselves. Therefore, the resolution has to be reduced and/or a smaller neural network must be used. The efficiency of the algorithm itself can be improved by implementing it in C++ instead of Python.

In this paper, a classical implementation is presented to demonstrate its performance without relying on neural networks for the detection task. We expect that deep neural networks could achieve a higher performance when provided with multiple images, the optical flow field, or even the κ angle per pixel. However, we expect that the currently presented pipeline can provide a valuable benchmark performance.

References

[1] Shweta Gupte, Paul Infant Teenu Mohandas, and James M. Conrad. A survey of quadrotor unmanned aerial vehicles. *Conference Proceedings - IEEE SOUTHEASTCON*, 2012.

[2] Mario Coppola, Kimberly N. McGuire, Christophe De Wagter, and Guido C. H. E. de Croon. A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints. *Frontiers in Robotics and AI*, 7, feb 2020.

[3] Danping Zou, Ping Tan, and Wenxian Yu. Collaborative visual SLAM for multiple agents: A brief survey. *Virtual Reality & Intelligent Hardware*, 1(5):461–482, oct 2019.

[4] Viktor Walter, Nicolas Staub, Antonio Franchi, and Martin Saska. UVDAR System for Visual Relative Localization with Application to Leader-Follower Formations of Multirotor UAVs. *IEEE Robotics and Automation Letters*, 4(3):2637–2644, jul 2019.

[5] James F. Roberts, Timothy Stirling, Jean Christophe Zufferey, and Dario Floreano. 3-D relative positioning sensor for indoor flying robots. *Autonomous Robots*, 33(1-2):5–20, 2012.

[6] Matous Vrba, Daniel Hert, and Martin Saska. Onboard Marker-Less Detection and Localization of Non-Cooperating Drones for Their Safe Interception by an Autonomous Aerial System. *IEEE Robotics and Automation Letters*, 4(4):3402–3409, 2019.

[7] Changhong Fu, Adrian Carrio, Miguel A. Olivares-Mendez, Ramon Suarez-Fernandez, and Pascual Campoy. Robust real-time vision-based aircraft Tracking from Unmanned Aerial Vehicles. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5441–5446. IEEE, may 2014.

[8] Bilal Taha and Abdulhadi Shoufan. Machine Learning-Based Drone Detection and Classification: State-of-the-Art in Research. *IEEE Access*, 7:138669–138682, 2019.

[9] Fabian Schilling, Julien Lecoeur, Fabrizio Schiano, and Dario Floreano. Learning Vision-Based Flight in Drone Swarms by Imitation. *IEEE Robotics and Automation Letters*, 4(4):4523–4530, oct 2019.

[10] Ryota Yoshihashi, Tu Tuan Trinh, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Differentiating objects by motion: Joint detection and tracking of small flying objects. *arXiv*, 2017.

[11] Jing Li, Dong Hye Ye, Timothy Chung, Mathias Kolsch, Juan Wachs, and Charles Bouman. Multi-target detection and tracking from a single camera in Unmanned Aerial Vehicles (UAVs). *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4992–4997, oct 2016.

[12] Cemal Aker and Sinan Kalkan. Using Deep Networks for Drone Detection. *arXiv*, jun 2017.

[13] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *arXiv*, pages 1–14, may 2017.

[14] Viktor Walter, Matous Vrba, and Martin Saska. On training datasets for machine learning-based visual relative localization of micro-scale UAVs. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 10674–10680, 2020.

[15] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[16] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I.Chao Chang, and Yan Xu. Maskflownet: Asymmetric feature matching with learnable occlusion mask. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, c:6277–6286, 2020.

[17] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12347 LNCS:402–419, 2020.

[18] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1647–1655, 2017.

[19] Raoul Dinaux, Nikhil Wessendorp, Julien Dupeyroux, and Guido de Croon. FAITH: Fast iterative half-plane focus of expansion estimation using event-based optic flow. *arXiv*, feb 2021.

[20] H.C. Longuet and K. Prazdny. The Interpretation of a Moving Retinal Image. *Proceding of the royal society of london*, 208(1173):385–397, 1980.

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus. *Communications of the ACM*, 24(6):381–395, jun 1981.

[22] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical Multi-Scale Attention for Semantic Segmentation. *arXiv*, pages 1–11, may 2020.

[23] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:3213–3223, 2016.

[24] N Mayer, E Ilg, P Häusser, P Fischer, D Cremers, A Dosovitskiy, and T Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

II

Literature Study

1 Introduction

Nowadays, Micro Air Vehicles (MAVs), also known as drones, are becoming more and more common. Reasons for their popularity include their high maneuverability and agility, simple design and vertical take-off capabilities. However, the relatively small size of MAVs also introduces the drawbacks of MAVs. MAVs have typically a short flight time and limited sensing capabilities. These disadvantages introduced a strong motivation to study the concept of swarms of MAVs, where the idea is to create larger groups of MAVs to reduce the individual limitations of the MAVs. Moreover, swarms of MAVs can perform tasks that a single MAV is unable to do, such as lifting a heavy object.

Three additional reasons on the global level of the swarm exist in favor of swarms compared to individual MAVs. Firstly, swarms are more robust to the loss of individual MAVs. Secondly, swarms can reconfigure to execute different tasks, making them flexible. Additionally, the swarm can adjust its size according to the global task that needs to be performed. [14] Moreover, a swarm of drones can cover larger areas in reconnaissance and surveillance tasks, as well as gather more information compared to individual MAVs. [66]

To enable the proper functioning of the swarm, as well as the MAVs on an individual level, sensing of the environment and the other MAVs is paramount. In particular, the relative locations of MAVs inside the swarm are needed for a large number of applications. The most basic and optimal methods of obtaining the locations of MAVs, is by using Global Navigation Satellite System (GNSS) (specifically, GPS) or Motion Capture System (MCS) signals. However, MCS requires an extensive setup of calibrated cameras and markers mounted on the MAV. [14]

Unfortunately, similar to MCS, GPS is not always available. This is where the motivation of this project starts to become clear. In warfare situations, GPS signals are actively jammed or spoofed. Jamming is the emission of arbitrary radio waves to make the original signal unusable and spoofing attempts to mislead the receiver by emitting incorrect signals on purpose. In these cases, the MAV is located inside a GPS-denied environment. A temporary workaround is to change the frequency bands on which to send and receive GPS signals. However, this is not a very effective method as the enemy will be able to notice this and jam or spoof signals on the new frequencies. In other cases of GPS denied environments, the GPS signal is entirely unusable or unreliable, for instance indoors, in cases where multipath effects[1] play a role or at high latitudes, where no signals from GPS satellites can be received.

This project will focus on using vision to determine the relative positions of drones within a swarm.

[1]Multipath effects correspond to interference of multiple reflected signals from satellites, decreasing the accuracy of GPS signals

To provide navigation even when GNSS is unavailable, additional research is needed. Multiple methods can be found in literature to solve this problem, such as the use of radar, sound, infra-red and vision. Specifically, optical flow will be used to detect MAVs in video feeds. Using the location of an MAV in an image, the direction of the MAV relative to the observer can be determined. Using depth information, obtained by for instance radio communication or stereo vision, one can determine the relative localization of other MAVs inside the swarm. Using communication, the drones will share their estimates to other drones in order to improve each other's estimates, provided that the drones are within the camera's FoV. State estimation will be applied to the position estimates to obtain a better estimate of the relative localization.

1.1. Research questions

The main research question is posed as follows:

> "How can optical flow be applied to detect MAVs in order to perform relative localization of MAVs inside a swarm as objective?"

The sub research questions are:

1. What kind of camera(s) are needed to be able to detect and identify MAVs at the relevant maximum range in its operational context?
2. How is the problem of relative localization typically solved for swarms of MAVs?
3. Which kind of optical flow is the most suitable and best performing in the domain of moving object detection?
4. How is moving object detection typically handled using computer vision?
5. What is state estimation and how will it possibly improve the relative localization of other MAVs inside a swarm?

1.2. Structure of the report

This report represents an literature study of the current research progress in the domains of relative localization. Each chapter handles one or two (sub) research question(s). In chapter 2, the sensors typically used for relative localization will be discussed (see research question 1). Moreover, different aspects of relative localization will be elaborated (see research question 2). Both SLAM, vision-based and the non-vision based MAV detection methods are explained. Finally, the types of communication between MAVs are compared. In chapter 3, optical flow is explained and the most important methods of optical flow are discussed (see research question 3). Based on chapter 3, chapter 4 explains how optical flow and appearance based neural networks can be used to detect moving objects (see research question 4). In chapter 5, state estimation using different types of Kalman Filters is pointed out (see research question 5). The preliminary results, in chapter 6, illustrate how various optical flow methods can be applied to perform moving object detection. The method for the thesis work is proposed in chapter 7. The conclusion and planning can be found in chapter 8 and 9.

2 Relative Localization

In swarms of MAVs, one of the most important types of information is knowing what the relative position is of one MAV to the other MAVs. Sensing in a swarm is more complex than an individual MAV attempting to sense its environment. An MAV in a swarm has to take the other MAVs into account while perceiving the environment.

In the second half of the previous century, a lot of research was put into the development of Kalman Filters as means of navigation. GPS did not exist yet at that time and Kalman Filters were the most useful method of determining accurate position estimates. When GPS was introduced in the 1980's, navigation systems became much simpler and the research into the Kalman Filter for navigation purposes became less relevant because of the high accuracy and absence of drift in GPS position estimates.

However, GPS is not always available. In so-called GPS-denied environments, where no reliable GPS signal is available, either because it is being jammed or spoofed, blocked by obstacles (for instance, indoors), has been degraded due to multipath effects or is not available in areas with high latitude, e.g. the north pole and south pole. An alternative is the use of Motion Capture Systems. However, this requires a setup of various cameras placed in the environment and markers mounted on the MAV. Obviously, the use of an MCS system heavily restricts the area in which MAVs can operate to specific indoor scenarios. Therefore, relative localization methods that rely solely on on-board sensing must be considered in this context.

In this chapter, different aspects of relative localization will be elaborated. Firstly, the sensors typically used for relative localization will be discussed. Subsequently, visual odometry (VO) and (visual) SLAM will be explained, along with their characteristics. The most relevant SLAM techniques are presented as well. As an alternative to SLAM, both the vision-based and the non-vision based MAV detection methods are presented. Finally, the types of communication between MAVs are compared.

2.1. Sensors

2.1.1. IMU

An Inertial Measurement Unit (IMU) is a principal sensor in any form of aircraft. It is one of the most important sensors onboard an MAV. Followed by the smartphone industry, Micro Electrical Mechanical Sensors accelerometers (MEMS) nowadays form a crucial component of MAVs. An IMU consists of an accelerometer and a gyroscope. A MEMS accelerometer determines the specific force of an object with respect to the inertial reference frame in rest. A MEMS gyroscope measures the rotational rate of the object. The main advantages of MEMS include their very low cost, very small volume and their lightweightness. Associated with these advantages, however, the main disadvantage is that their output includes significant noise. More expensive and heavier gyroscopes, such as the Ring laser gyroscope, offer better accuracy and Signal to Noise Ratios (SNR).

Next to noise, MEMS gyroscopes also suffer from drift. This drift depends on the temperature of the sensor and the type of sensor. The drift and noise of the MEMS accelerometer, as well as measurements from other sensors can be corrected for using sensor fusion. This is described in chapter 5.

Next to the IMU, several other sensors are present on most MAVs. A magnetometer measures the magnetic field in the vicinity of the MAV. If there are not too much disturbances in this magnetic field, the field lines of the earth's magnetic fields can be measured, by which the direction of the north pole can be estimated. A barometer measures the air pressure, yielding a reasonably accurate altitude estimate if the air pressure on ground is known.

2.1.2. Vision

An obvious concept to solve the relative localization problem is to rely on vision. A single image contains a rich amount of data, a video even more so. A main advantage of vision is its high directional accuracy, due to the relative high density of pixels in modern cameras.

However, vision suffers from a high amount of disadvantages as well. Firstly, vision-based detection relies on a Line of Sight (LoS) between the observer and the target. This Line of Sight assumption can be broken by obstacles or in situations with low visibility, such as introduced by fog or (low altitude) clouds. Another cause of non-LoS situations is the limited FoV of cameras. Omni-directional vision can only be achieved using multiple cameras. Mounting multiple cameras on an MAV increases the mass of the MAV and reduces the space available etc. Moreover, vision-based approaches are generally computationally expensive due to the large amount of unstructured data that has to be processed. This constraint drives a large amount of current research that is driven by the objective of reducing the computational effort of navigational algorithms and therefore allowing these algorithms to be used on smaller MAVs. Algorithms requiring lots of computations also require a large amount of power, which in turn reduces the amount of energy available for the MAV's flying task and thus reduces the action radius of an MAV.

In figure 2.1, the different methods of vision-based relative localization are presented. The two main types of solving the localization problem are based on either using the environment or the detection of MAVs directly. Simultaneous Localization and Mapping (SLAM) is wide field of research that targets the first type, where simultaneously a map is build of the environment, as well as the pose of the observer is estimated. Collaborative SLAM (CoSLAM) extends SLAM to multiple cameras with different poses. CoSLAM uses communication of keyframes between different entities to build a global map of the surroundings.

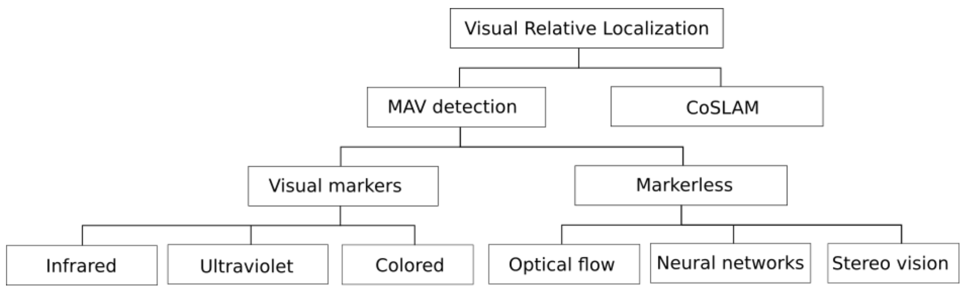


Figure 2.1: Overview of the different methods of vision-based relative localization.

The second type of relative localization focuses on the detection of the MAVs themselves. This can either be done using markers or without markers. Markers are physical devices that make the MAVs easier to detect. These can be either infrared (IR), ultraviolet (UV) or colored objects. Markerless detections are harder to perform and rely on methods such as optical flow, neural networks and stereo vision. The advantages and disadvantages of the various methods are summarized in table 2.1.

Table 2.1: Qualitative characteristics of relative localization methods.

Method	Advantages	Disadvantages
Colored markers [62]	• Can detect IDs of different MAVs • Can detect other MAVs (relatively) easily	• Not feasible on smaller MAVs or flapping wing MAVs • Can only detect other MAVs with markers • Relatively limited range (15m)
Infrared [49]	• Accurate • Computationally simple • Can perform in visually cluttered situations • Lower dependence on lighting conditions	• Heavy (approx. 400g) • Needs visual line of sight • High energy expense if many sensors are used (10W) • Relatively low range (12m)
Optical flow [36]	• Robust against cluttered backgrounds • Less dependent on the perceived appearance of the target • Can provide data individual frames cannot yield	• No real-time method yet that can be run onboard. • Accuracy of 86% • Only works for small differences between frames
Neural networks [58]	• Low-cost depending on cameras	• Works for clearly visible and distinguishable shapes of MAVs • Accuracy of around 85% to 90%
Stereo vision [60]	• MAVs without markers can be detected • (Slightly) less weight, power consumption etc	• MAVs can be recognized at different angles, positions, speeds and sizes • Other MAVs have to be detected against cluttered, dynamic backgrounds
SLAM [11]	• It can mitigate the integration drift of VO. • Can provide reliable navigation in GPS denied environments. • Includes IDs. • Does not need LoS.	• Provides a 3D map of the surroundings • Computationally expensive (except Navion [56]) • Requires communication • Depends on lighting conditions • Requires sufficient visual overlap

In table 2.2, the different methods of relative localization for MAVs are presented. Note that there exist clear differences between the markerless methods and the other marker-based methods. The methods based on the detection of active markers, either using color, IR or UV, have a relatively short maximum range of at most 15m in the case of UV markers. This is because a small marker is hard to perceive at long distances, where the marker only spans a few (or less) pixels in the image. For larger distances, the results became unreliable. The maximum range could theoretically be increased by increasing the resolution of the camera, as well as using more powerful and larger LEDs. However, in practice this could be unpractical. A method based on markerless detection, such as the stereo vision approach of [60] can reach larger distances. Unfortunately, maximum ranges for neural network and optical flow based approaches are not given in literature. [58]

The main disadvantage of the markerless methods is that they require a high amount of computations. As shown in table 2.2, the markerless methods only work in decentralized setups, where the computations are performed on a desktop, except for the stereo-vision based method of Navion [56].

Table 2.2: Quantitative characteristics of relative localization methods. 'D' means that the computation is deployed on a desktop computer, 'O' corresponds to computation performed onboard an MAV. Other symbols: 'E'=Error, 'R'=Recall. Accuracies in percentages represent the ratio of positive detections divided by true positives.

Method	Range	MAV/marker size	Sensor mass, power, size	Computational effort	Accuracy
Colored markers [19]	5.5m[1]	9x9cm	<2.6W[2], 23g, 39x26x25 mm	O, 600MHz ARM, 7-27 Hz	6.5cm
IR [49]	6.0m[3]	11x11cm	10W, 400g, 220mm diameter	O, 1kHz	5-35cm[4]
UV [61–63]	15m[5]	55cm	50g	O, 72Hz	10-20%
Stereo vision [60]	20m[6]	55cm	<3.8W, Karmin2 (450g, baseline 25cm)	O, 40Hz	R>75%, E<1m
Optic flow [36]	NS	NS	NS, GoPro 3 (700g, 105x248x103mm)	D, 8.9Hz	87%
Neural networks [58]	NS	NS	NS	D, High	85%-90%
ORB-SLAM [42]	N/A	N/A	Regular handheld camera	D, High	2 cm indoor
Navion [56]	N/A	N/A	24mW, 5x4mm	O, 171Hz	E=0.28%

2.2. Visual Odometry and Visual SLAM

Visual Odometry (VO) can be used by robots to estimate its motion based on measurements derived from vision. It has an inertial counterpart, Visual Inertial Odometry (VIO), which combines vision data with the data from an Inertial Measurement Unit (IMU). VIO can be used by MAVs to navigate precisely.

SLAM is an extension of VIO, where, next to the localization, simultaneously a map of the environment is build. SLAM is the combination of VO and loop detection and closure and an application of the Structure from Motion (SfM) field. SLAM guarantees global consistency, which VO does not, meaning that the estimated trajectory is correct from start to end. SLAM fails easily if not used in combination with IMU. SLAM has the advantage that it can diminish the integration drift that VIO methods suffer from. In general, SLAM methods are computationally intensive. However, Navion [56] counters this and introduces a lightweight alternative which uses stereo vision, only uses 25mW and can still run at 170Hz on a very small chip.

SLAM can also be used with other sensors than cameras. LIDAR-based SLAM depends less on lighting and is computationally less requiring. Contrarily, it is more expensive, has more mass and a LIDAR sensor requires more power [14, 43].

SLAM can be used very effectively in static environments. However, in dynamic environments, for example an MAV that is part of a swarm, this assumption is no longer valid. One solution to this problem is to discard these dynamic features as outliers. However, at the same time, these features contain useful data that can be used to estimate the relative trajectories between moving objects, such as MAVs. To correctly achieve this, one has to separate the tracking errors (the real outliers) from the dynamics features. By combining the methods of collaborative visual SLAM and optical flow, one can solve the problems of relative localization and mapping for MAVs in a swarm. The IMU can be used for improving the Visual Inertial Odometry estimate and obtaining the absolute scale of the environment, as well as Communication between the MAVs is needed for collaborative visual SLAM. The computational power that is required will be relatively high, as SLAM and optical flow require a large amount of calculations.

[1] For larger distances, the structure of circle is lost.
[2] 2.6W is the power usage of the complete system.
[3] Could have been more, as the sensor was not calibrated for >6m.
[4] The accuracy depends on distance.
[5] For larger distances, spurious drop-offs happen as the spot area becomes too small
[6] Determined by the camera's resolution and baseline. Between 20m and 50m, results were not reliable anymore.

2.3. Characteristics of vSLAM methods

Visual SLAM methods are typically divided into two categories: The direct approach and indirect approach. The direct vSLAM methods do not use features, but use pixels directly. An example of such an approach is LSD-SLAM. [18] These direct approaches can give higher accuracy, if a precise calibration of the camera is available. [73]

Another difference between SLAM methods is the differentiation between filter based approaches and key-frame based approaches. Filter based approaches use an Extended Kalman Filter (EKF), where the state vector contains the camera pose. as well as the locations of the tracked landmarks. This means that filter based approaches become inefficient for high numbers of landmarks, as each landmark has to be updated each iteration. [73] A solution to this problem is the Multi-State Constraint Kalman filter (MSCKF). [41] In this model, the landmarks are removed from the state vector and instead, constraints on the motion of the camera are put into place. Now, the iterations are performed efficiently despite the large number of landmarks.

The key-frame approaches split the two main tasks into separate threads. The first thread tracks the pose of the camera, while the other thread generates a map of observed features. [33] This makes the method very efficient. The camera pose and mapping tasks are dependent on each other. The map is built up by triangulating the mapped features. [73]

The application determines which of the two types is best suited. Filter-based approaches are generally easier to implement and have more performance if the state vector is not too large or if MSCKF is used. Key-frame approaches have typically higher accuracy but require more computational effort. Additionally, Key-frame approaches are less complicated to extend to collaborative situations. The agents can send their key frames to a central server, where the global map is generated constantly. Filter based approaches have the challenge of applying a single filter on different agents in order to enable collaboration. [73]

2.4. vSLAM methods highlighted

Several open-source SLAM implementations exist, most notably ORB-SLAM2 [42]. Other SLAM methods have been developed that focus on dynamic scenes, such as SLAMMOT [13] and DynaSLAM 2 [7].

2.4.1. ORB-SLAM2

ORB-SLAM (and its successor ORB-SLAM2) is considered the most robust type of SLAM implementation. Its name derives from the fact that it uses the ORB feature detector. [53] ORB-SLAM2 features a loop closing, map reuse and relocalization techniques. Bundle adjustment is used to refine the generated 3D maps. Bundle adjustment uses nonlinear least squares solving to optimize the SfM results. [73]

2.4.2. SLAMMOT

Most SLAM methods ignore the information of moving objects. However, some this information can also be extracted and used for the tracking of objects. This type of SLAM is called Simultaneous Localization, Mapping and Moving Object Tracking (SLAMMOT) [13]. SLAMMOT decomposes the information of the gained features into two separate maps, the static and dynamic map.

2.4.3. DynaSLAM 2

Interestingly, SLAM methods suited for general dynamic environments have also been pursued. An example is DynaSLAM 2 [7]. DynaSLAM 2 targets stereo and RGB-D platforms and uses the ORB feature detector to detect and track features.

2.5. Vision-based MAV detection methods

As presented in figure 2.1, the alternative to SLAM is using 'direct' detection techniques that focus on detecting other MAVs instead of using information of features in the surroundings. This section will elaborate on these techniques, first discussing the methods based on the use of markers. Subsequently, the more advanced marker-less techniques are discussed.

2.5.1. Infrared markers

Infrared sensors were researched by Roberts [49] as a way for flying robots to collectively navigate indoors. It operates using LEDs, called the marker, emitting infrared light at a given wavelength and phototransistors receiving the signal on the same wavelength. [17] RGB-D sensors use this infrared technique to calculate a depth map. An RGB-D sensor combines an RGB camera with an IR projector and an IR camera, resulting in a depth map.

Similar to infrared sensors, Light Detection And Ranging (LIDAR) work similarly to infrared sensors by using Time of Flight. It features a laser rotating along one axis at 30 Hz to 1kHz which simultaneously receives the emitted laser pulses. The time difference between departure and arrival, along with phase information can be used for depth estimation. The advantage of LIDAR compared to regular cameras is its reduced sensitivity to environmental factors and the ability of estimating accurate depth information. On the other side, LIDAR requires more power, are heavy and large. Additionally, extra care has to be taken for operating with lasers safely in urban environments. Miniature LIDARs have been developed to address these issues. However, currently miniature LIDARs suffer from solar interference. [17]

Infrared sensors have the advantage of not requiring algorithms with a high computational load, sensors or modifications to the environment. Additionally, varying indoor illumination does not affect the operation of the infrared sensor. Unfortunately, the maximum range at which the infrared markers can be detected is very limited (around 12m) [49]. The Field of View (FoV) of the sensors are confined to a few degrees. Another major disadvantage of using infrared-based methods such as the RGB-D sensor is interference with external sources of infrared. The RGB-D sensor is less robust to natural light than standard stereo cameras and is therefore less suitable for outdoor use. Finally, the total setup for infrared sensors used by Roberts [49] is heavy. The total system of sensors has a mass of approximately 450g.

2.5.2. Ultraviolet markers

Walter et al. [61–64] researched an alternative to infrared markers, namely ultraviolet markers. Ultraviolet is harder to find in nature, which makes the detection of such active markers easier. It features a higher accuracy compared to infrared sensors. By mounting the LEDs on the ends of the rotor arms, one can use the relative orientation of the two LED spots in the image to estimate the orientation of the MAV.

The number of pixels covered by the LED in the image determines the distance between the marker and the observer. The range is however still limited. The LED spots in the image cover a single pixel for large distances. This maximum range is determined by the resolution of the camera, radiation intensity of the LED, sensor type and exposure rate.

2.5.3. Colored markers

Roelofsen et al. [50] has considered the use of non-active (passive) markers in the form of colored balls attached to an MAV. Color segmentation is applied to detect the marker in the image. This method as well reduces the complexity of the problem and makes the detection task much easier compared to markerless solutions. By registering the amount of pixels of the sphere in the camera image, a distance measurement can be estimated.

2.5.4. Stereo vision

Stereo vision is the method of combining data from multiple cameras (typically two) to determine per-pixel depth in an image. By matching every pixel in the left image to the same feature in the right image, one can calculate the depth (the euclidian distance between the camera and the object) using the offset of the pixel between the two images, called the disparity.

Stereo vision can be used as a method of detecting MAVs, as MAVs seem to be “floating” in the air [12, 60]. In their work, the YOLOv2 network [46] is used to detect MAVs from stereo disparity images. The YOLOv2 network has one convolutional neural network which puts bounding boxes around objects. YOLOv2 can detect objects from more than 9000 object categories, which explains its nickname YOLO9000, in real-time. It has been used for MAV detection in RGB images as well by Aker and Kalkan [2]. Returning to the stereo vision method of Carrio et al., a synthetic dataset was developed using Microsoft’s AirSim [55], which is a computer graphics tool based on Unreal Engine 4 that is focused on generating realistic graphics for robot simulations, as well as generating sensor measurements.

2.6. Non-vision based MAV detection methods

Next to vision, measurements from various other sensors can be analyzed to detect MAVs. In this section, sound and RADAR are elaborated.

2.6.1. Sound

Alternatively, acoustic signals can be used for detection of MAVs. Acoustic patterns of MAV sounds should be analyzed in order to distinguish MAV sounds from sounds of other objects. Acoustic analyses require information of current wind conditions. Also, this method may fail in environments with high background noise, such as cities. The advantages of this method are that it relies on low-cost components and it can work outside line-of-sight conditions. [24] Microphones can pick up sounds from all directions, whereas cameras have a limited FoV. Therefore, the aperture of this method is larger than vision-based methods.

2.6.2. RADAR

Opposed to vision-based techniques and in line with acoustics methods, RAdio Detection And Ranging (RADAR) is not affected by low visibility situations, such as fog, clouds etc. In general, it does not require a LoS to other MAVs. Also, it does not need the MAV to transmit signals and higher frequencies allow the use of Doppler effect. On the adversary, small MAVs are difficult to detect using RADAR. Until now, not enough research has been put into the detection of multiple types of MAVs, different RADAR geometries to achieve robust detection of these scenarios. [24]

2.7. Analysis of Relative Localization Methods

When considering non-SLAM based relative localization methods, the main differentiation is that of using markers or not. Developing methods without markers, saves mass and volume [60].

2.8. Communication

Crucial in collaborative relative localization techniques is the communication between agents. This is typically done using radio waves. MAVs can communicate using different technologies. The major forms of communication include the IEEE 802.11 standard and the XBee-Pro platforms. The latency of IEEE 802.11 varies significantly during operation from 2ms to 270ms. The latency of XBee-Pro is more robust at around 10ms. However, IEEE 802.11 has a much larger bandwidth of 11 to 54 Mb/s compared to 250kb/s in the case of XBee-PRO. [3] In the optimal case, a combination of XBee-Pro and IEEE 802.11 would be used, to combine the advantages of both platforms, namely the low latency of XBee-Pro and the high throughput of IEEE 802.11.

3 Optical Flow

Optical flow is a widely used approach for the tracking of features. The optical flow often represents the physical velocity of an image, projected onto the camera image. Optical flow can be implemented in many different forms. The methods are mostly differentiated by one characteristic: It is either based on the tracking of a set of features (sparse optical flow) or determine optical flow for every pixel, or blocks of pixels, in the frame (dense optical flow. [34] The first person to describe optical flow was Gibson [23]. One needs a distance measurement to be able to calculate velocity. [14] In the next section will be explained why optical flow was chosen as the method to detect MAVs.

3.1. Why optical flow?

Optical flow has multiple advantages over its alternative vision-based methods, such as appearance-based neural networks. Firstly, according to Li et al. [36] MAVs could be detected in situations where they were barely distinguishable for the human eye due to background clutter and their size. Additionally, optical flow based methods are less dependent on shapes and appearances of MAVs compared to other methods. [36] Moreover, neural networks typically use an input format with a certain (relatively low) image resolution, such as 800 x 600. Higher resolutions require higher computational effort and retraining of the network. Optical flow methods do not suffer from this constraint. Finally, optical flow has hypothetically relatively large maximum range compared to active markers. But this points to one of the research questions of this project.

A major drawback of optical flow is the computational effort that is required. Especially dense optical flow methods face a large challenge when attempted to run in real-time. Additionally, optical flow-based detection assumes the target has different motion compared to the background. Otherwise, the output is zero and other methods such as appearance-based methods have to be applied.

3.2. Interpretation of Optical Flow

The interpretation of optical flow has been studied by Longuet and Prazdny. [38] The most important notion in this paper is the fact that flow can be split into a translational and a rotational part. In equation 3.1, the equations describing the optic flow vectors in the image plane (u and v) are described, where U, V and W represent the translational velocities, A, B and C represent the rotational rates and the inertial coordinates in the physical world are X, Y and Z. As can be seen, the translational part depending on U, V and W can be split into u_T and the rotational part depending on A, B and C can be substituted by u_R.

$$
\begin{aligned}
u &= -\frac{U}{z} + x\frac{W}{Z} + Axy - Bx^2 - B + Cy = u_T + u_R \\
u &= -\frac{V}{Z} + y\frac{W}{Z} - Cx + A + Ay^2 - Bxy = v_T + v_R
\end{aligned}
\tag{3.1}
$$

The Focus of Expansion (FoE) is the point inside or outside the image where the optical flow has zero magnitude. Using the FoE, one can estimate the direction of movement of the camera. Setting $u_T = u_R = 0$ in equation 3.1, one obtains:

$$
\frac{x_{FoE}}{y_{FoE}} = \frac{U}{V} \tag{3.2}
$$

Another observation can be made: The Time-To-Contact (TTC), which is given by $\frac{Z}{W}$. This yields a measure of time that can be used to determine whether a collision is about to happen. An example of TTC calculation for MAVs was presented by de Croon et al. [16]. In their work, the FoE is not calculated, as errors in the location of the FoE lead to large errors in the results. Instead, the parameters of the optical flow field are determined directly.

3.3. Implementations of Optical Flow

Currently, various methods of calculating optical flow exist. Most important measures of optical flow approaches are their accuracy and computational effort. In this section, the approaches will be discussed in order of rising accuracy. Different approaches are

3.3.1. Traditional approaches

Lucas-Kanade

Lucas-Kanade optical flow [39] is a very popular type of optical flow and is typically used as a sparse form of optical flow. It matches a certain sub-region, for instance a block of three by three pixels, numerically from the current frame in the next frame. This template is warped using a transformation back into the original image. This transformation can be simply a rotation or an affine transformation. The objective of Lucas-Kanade is to minimize the sum of the squared errors between the template and the warped image. The warp is calculating using interpolation between the pixels. [5]

Lucas-Kanade must rely on a feature detector. The most popular feature detectors include Harris [25], FAST [59] and Shi-Tomasi [29]. Scale Invariant Feature Transform (SIFT) is a method which robustly handles arbitrary transformations of features between frames. It has a lightweight variant called Speeded Up Robust Features (SURF) [6]. A method that combines the methods of FAST and BRIEF is Oriented FAST and Rotated BRIEF (ORB) [53]. ORB has a better performance than SIFT and SURF and is more than 100 times faster than SIFT. [53]

Farnebäck

Farnebäck [20] introduces a dense optical flow method. In various research papers, Farnebäck was proven to outperform Lucas-Kanade. [8, 15, 22] The method of Farnebäck consists of two steps. In the first step, all block of pixels are fitted by quadratic polynomials. The second step involves the estimation of displacement fields from polynomial approximations.

LDOF

Brox [8, 9] optical flow has a higher accuracy compared to Farnebäck, as proven in the work from Brox et al. Brox et al.'s Large Displacement Optical Flow (LDOF) method

LLUVIA
A very high performance open source optical flow implementation is LLUVIA [1]. It takes approximately 1ms to process an image of 1016x544 pixels on a GTX-1080 GPU. It uses the Vulkan API, enabling support for a wide range of hardware configurations, in contrast to many approaches that rely on the proprietary CUDA software, which needs an Nvidia GPU to operate. An alternative is OpenCL, which is the open-source counterpart of CUDA. OpenCL is, however, slowly adapted by the industry and it is difficult to develop for a wide range of hardware. Vulkan gives more control to the developer than its predecessor OpenGL, meaning at the same time that the developer has to write more fundamental code. Most importantly, it is rapidly adopted by the major GPU manufacturers, as the industry has a large need for very efficient GPU drivers. The method of LLUVIA uses a filter-based approach.

3.3.2. Neural networks

Neural networks are becoming increasingly popular over the years in numerous research fields. Computer vision is one of the main fields. Neural networks consist of layers connected by nodes in which each node has a certain weight. These weights form the information stored in the network. The layers connect the input to the output of the network. The weights are unknown and have to be determined by a process called training. For each node, the activation function determines the output of the node given the input. The weights determine the shape of the activation functions. The performance of a neural network is defined by a squared cost function. Using gradient descent, the back propagation method attempts to minimize the cost function by slowly improving each weight in the network. Gradient descent is an optimization method that tries to find the (preferably global) minimum of a function by analyzing the gradient at its current input of the function. Other optimization methods include branch and bound, single step solvers, dual ascent etc.

An important disadvantage of neural networks is that they are certainly not a black box model. The weights and nodes have no physical meaning and the training process is arbitrary. There is no real connection between the model and its physical representation. Additionally, neural networks can require immense amounts of computational power. On the contrary, a benefit of using neural networks is that they can fit any complex dataset, however scattered or nonlinear it may be.

An upcoming technique for calculating dense flow is by the use of neural networks. The first neural network that could learn to apply optical flow using an artifical dataset was FlowNet. [28] FlowNet makes use of a Convolutional Neural Network (CNN) for estimating a dense optical flow field. The disadvantage of this method, is that it cannot accurately calculate small flows. This is interesting, as conventional optical flow methods have little difficulty with small motions compared to large motions. FlowNet has a successor called FlowNet 2.0, which stacks the FlowNetS and FlowNetC architectures as a deeper network. Additionally, it adds a network that works in parallel and handles small flow vectors. Alternatively, DeepFlow [67] uses Deep Matching to match patches in images instead of learning. The most relevant types of neural-network based optical flow methods are presented in table 3.1, benchmarked with the MPI-Sintel dataset [28].

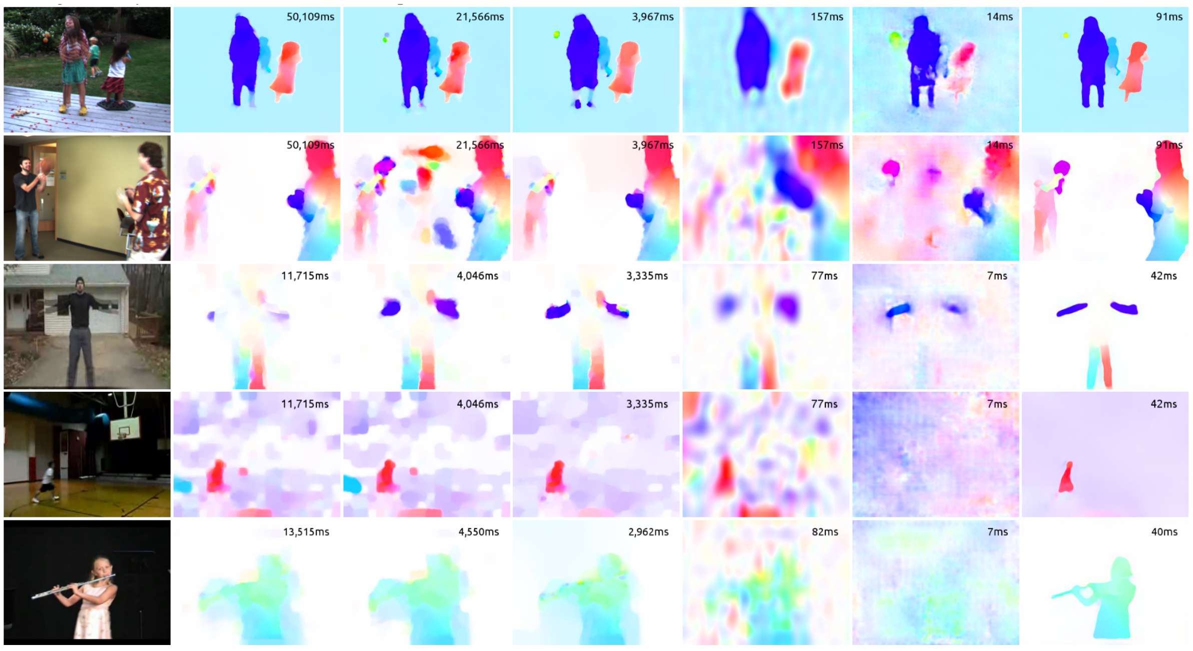
Note however, that neural network-based approaches for calculating optical flow, especially FlowNet2, can be very sensitive to attacks, where a certain pattern spanning less than 1% can disrupt a large part of the image. [45] A patch of 0.1% can disrupt a FlowNetC network by increasing the error by 100%. A patch of 4.5% of the image size can increase the flow error by 555%.

FlowNet2 has a more lightweight version with a higher runtime of around 2.2 times faster, but at the same time also a higher accuracy, called LiteFlowNet3. Additionally, LiteFlowNet3 has only 5.2 M parameters compared to the 160M parameters of FlowNet2. [26]

An open-source optical flow framework created by Microsoft called MaskFlownet [71] achieves excellent results in the MPI Sintel and KITTI benchmarks, respectively third and fifth. These results are achieved by taking occlusion into account.

Table 3.1: Characteristics of a relevant optical flow methods. Taken from the KITTI benchmark. [22] The density of the methods is always 100.00% except for Lucas-Kanade, where it is 99.90%. The programming languages are: C = C/C++, P = Python, M = Matlab.

Method	**Out-Noc**	**Out-All**	**Avg-Noc**	**Avg-All**	**Runtime**	**Environment**
MaskFlownet [71]	2.07 %	4.82 %	0.6 px	1.1 px	0.06 s	TITAN Xp
LiteFlowNet3-S [26]	2.49 %	5.91 %	0.7 px	1.3 px	0.07s	GTX 1080
LiteFlowNet3 [26]	2.51 %	5.90 %	0.7 px	1.3 px	0.07s	GTX 1080
LiteFlowNet2 [27]	2.63 %	6.16 %	0.7 px	1.4 px	0.0486 s	GTX 1080
FlowNet2 [28]	4.82 %	8.80 %	1.0 px	1.8 px	0.1 s	GPU @ 2.5 Ghz (C)
DeepFlow2 [48]	6.61 %	17.35 %	1.4 px	5.3 px	22 s	1 core @ >3.5 Ghz (C)
DeepFlow [67]	7.22 %	17.79 %	1.5 px	5.8 px	17 s	1 core @ 3.6Ghz (P)
EpicFlow [47]	7.88 %	17.08 %	1.5 px	3.8 px	15 s	1 core @ 3.6 Ghz (C)
LDOF [9]	21.93 %	31.39 %	5.6 px	12.4 px	1 min	1 core @ 2.5 Ghz (C)
PolyExpand [20]	47.59 %	54.00 %	17.3 px	25.3 px	1 s	1 core @ 2.5 Ghz (C)
Pyramid-LK [39]	65.81 %	70.16 %	21.8 px	33.2 px	1.5 min	1 core @ 2.5 Ghz (M)



(a) Source (b) FlowFields [4] (c) DeepFlow [67] (d) LDOF [9] (e) PCA-Flow [68] (f) FlowNetS [28] (g) FlowNet2 [28]

Figure 3.1: Results of various open-source optical flow methods. The runtime is displayed in the upper right corner. 'Noc' means no occlusion. Figure taken from Ilg et al. [28].

4
Moving object detection

The vision-based detection of MAVs in a swarm is still a challenge. [14] The algorithm has to be able to detect MAVs at various orientations, velocities and shapes. Additionally, imperfect lighting conditions pose extra challenges on the detection task. In existing literature, two main methods exist of detecting moving objects from a moving camera. The first one is based on the appearance of the MAV. Typically, neural networks are used to detect MAVs based on their appearance. [51] Haar models can be used as well, but give a lower accuracy. In return, Haar cascade models have do not require the availability of a GPU, require less computational effort and have a higher easy of use compared to neural networks. [44] In this chapter, both optical flow and machine learning methods will be discussed, along with their advantages and disadvantages.

An advantage of using vision for the detection of MAVs, is that drones can be detected that do not have Radio Frequency (RF) transmission. Moreover, optical sensors are typically cheap and have very high angular resolution. [24] A major disadvantage of vision-based detection of MAVs and other objects, is that it needs a line-of-sight (LoS) between the observer and the object that is to be detected. This LoS assumption can be broken in situations where visibility is limited due to fog or clouds, when occlusion takes place and the objects is (partially) hidden behind another object, when the MAV is outside the FoV of the camera or during night-time. Radar-based methods can greatly overcome this problem. However, radar-based methods suffer from the challenge of detecting small objects, because their radar cross-section is very small. [24]

4.1. Optical Flow

To our current knowledge, the research done by Li et al. [36] is the only work using purely optical flow without artificial neural networks for the detection of MAVs. With their method, they were able to detect other MAVs, even when they were barely visible because of their size and the cluttered background. It has approximately the same detection accuracy (87%) as appearance-based neural networks applied to MAV detection (maximum accuracy of approximately 90%). [2, 58] The method of Li is based on a combination of background subtraction and Lucas-Kanade optical flow. The background subtraction process assumes that the tracked objects have very different motion compared to the background. If the motion difference is larger than a certain threshold, the object is pruned and these points are clustered.

4.2. Appearance-based MAV detection using neural networks

In line with its rise in a wide range of research fields, machine learning is now applied to the problem of detecting MAVs in an image based on their appearance. As explained by Taha and Shoufan [58], openly accessible datasets in this research area are hard to find. The papers written in this research domain typically fail to provide the maximum range of their method at which MAVs could still be detected. Typically, the device used and the type of drone are also neglected in their papers.

YOLO9000 [35] is a commonly used neural network model for detecting objects in images. The YOLO9000 network has one convolutional neural network which puts bounding boxes around objects. YOLOv2 can in real-time detect objects from more than 9000 object categories, which explains its nickname YOLO9000. It has been used for MAV detection in RGB images as well by Aker and Kalkan [2].

4.3. Hybrid methods

Advanced methods combine the advantages of both neural networks segmentation models and optical flow methods, effectively combining motion and appearance information. The authors of [51] apply this strategy. The main advantage of hybrid method is that they do not suffer from the situation in which targets become smaller, resulting in unreliable optical flow estimates. ClusterNet [35] also combines appearance and motion information. Yoshihashi et al. [69] introduce a Recurrent Correlation Network

4.4. Background subtraction

Next to optical flow, a commonly used method of detecting moving objects is the method of background subtraction. By calculating a geometric model of the pose difference between frames, one can exclude the features or pixels that do not conform this model. One technique for removing outliers is the use of RANdom SAmple Consensus (RANSAC) [21]. Alternatively, the camera ego-motion can be estimated using an inertial measurement unit (IMU). [54]

The Mixture of Gaussians 2 (MOG2) [72] method is an efficient background subtraction method that revolves around a Gaussian Mixture Model (GMM). MOG2 models each background pixel using a set of Gaussian distributions. The weights of this set correspond to the durations for which these pixels remain inside the scene. The background pixels are those that stay longer at the same location. MOG2 slightly improves its accuracy compared to the original MOG, but greatly improves processing time and is more robust to illumination changes.

4.5. Datasets

To avoid spending large amounts of time on creating a dataset and to be able to compare the method with other methods in literature, existing datasets were sought. As explained by Taha and Shoufan [58], openly accessible datasets in this research area are hard to find. Most datasets focus on the usage of neural networks and provide separate images that are uncorrelated with each other and can therefore not be used. Instead, video feeds are necessary, where each frame follows the previous frame by a few milliseconds, such that optical flow can be enabled.

A dataset with videos taken from an MAV of other MAVs is the one from Rozantsev [52]. This dataset provides greyscale videos of drones and planes. It is a challenging dataset, because of the relatively low resolution of around 752x480 pixels, as well as high brightness changes and motion blur in the drones category of videos.

A dataset that focuses on the detection of MAVs in order to improve security and defense in the public domain has been created by Zhao[1]. The dataset provides RGB, as well as (thermal) IR videos. However, the dataset provides videos in which the target MAV is already centered in the image frames and the background is very uniform, making the MAVs easy to detect. Thus, the dataset

[1]See: https://github.com/ZhaoJ9014/Anti-UAV

is not very representative for real-world situations relevant to this project. Li et al. [37] created a dataset to estimate the positions of MAVs from multiple cameras. The cameras are positioned statically, which makes the detection task less challenging. Therefore, it is suitable for early stage testing of detection algorithms, but less realistic for real-world situations.

Pawelczyk et al. [44] created a dataset for which they collected hundreds of YouTube videos containing MAVs and videos without MAVs. Unfortunately, the dataset cannot be downloaded, there is a corresponding issue on the GitHub repository. Similarly, an MSc thesis focusing on drone detection using machine learning [57] provides a dataset with RGB and IR real-world videos of airplanes, birds, MAVs and helicopters. Except for the helicopter videos, all videos are obtained from a camera that is not moving. The videos are 640×512 pixels in resolution and have associated masks.

A popular dataset with video feeds taken from a camera mounted on various MAVs is created by Zhu et al. called VisDrone2018 [70, 71]. Unfortunately, the dataset has no videos containing other MAVs in the video feeds. However, it features a large set of videos taken from different situations, with (relatively) small objects such as cars, pedestrians present in the video feeds. It can therefore still based as a more general dataset for object detection or for background subtraction.

Walter et al. [64] automatically created a dataset called MIDGARD in which other MAVs in the video feeds are annotated by the use of UV sensors. By marking the other drone with two UV markers and filtering this data using a Kalman Filter, the estimated position of the MAV can be reprojected into the camera image and subsequently, the MAV can be annotated. The project is focused towards machine learning applications, but is also very useful for optical flow methods.

A typically used dataset for optical flow is the MPI Sintel dataset [10]. In this dataset, different scenes are rendered using different rendering settings. Each higher render setting contains more render passes, therefore increasing the realism of the scene. The albedo pass only represents meshes without shading. The clean pass includes basic shading and specular reflections (reflection where the strength is based on the direction of light and the normal of the surface). The final pass includes motion blur, depth of field, atmospheric effects etc. An alternative to the MPI Sintel dataset is the FlyingThings3D dataset [40]. This dataset is a popular artificial dataset for training neural networks in computer vision. Another alternative to the two aforementioned datasets is KITTI [22]. KITTI is a more diverse dataset that can be used for optical flow and visual odometry is KITTI. KITTI is widely used in these fields.

5 State estimation

State estimation is needed because system and measurement noise introduce biased estimates for the states and therefore cause biased estimators and thus reduce the accuracy of the models. Additionally, not all states can be measured but have to be calculated based on the data of other sensors. Finally, sensor fusion improves the accuracy of the estimates by combining the data of multiple sensors.

5.1. Kalman Filter

Kalman filtering [32] is a recursive state estimation technique in which, for each timestep, the state is estimated using a weighted average between a prediction of the state in the next timestep and a set of measurements. The Kalman gain determines whether the prediction has more effect on the estimation or whether the measurements determine the new estimation. The Kalman Filter is an optimal filter. Unfortunately, it only applies to linear systems. However, in line with most natural processes, the dynamics of an MAV are highly nonlinear. Therefore, the standard Kalman Filter cannot be used to estimate the states of this problem.

5.1.1. Extended Kalman Filter

As an extension of the original Kalman Filter, the Extended Kalman Filter [30] enables the Kalman Filter for use in non-linear systems. The EKF achieves this by linearizing the system locally in the vicinity of the current state estimation. This is done by calculating the jacobians of the transition and output matrices. The EKF is, however, not an optimal filter and is therefore not guaranteed to converge.

$\vec{x}(t)$	state vector of dimension n	$\mathbf{F}(t)$	System matrix (n×n)
$\vec{x}_0$	initial state	$\mathbf{B}(t)$	Input matrix (n×m)
$\vec{u}(t)$	control input vector of dimension m	$\mathbf{G}(t)$	System noise input matrix (n×m)
$\vec{w}(t)$	system noise vector of dimension m	$\mathbf{H}(t)$	Observation matrix (m×n)
$\vec{v}(t)$	measurement noise vector of dimension p	$\mathbf{D}(t)$	Feedforward matrix (p×m)

Table 5.1: Vectors and matrices used in the Kalman Filter.

Nnote that the D matrix has to be zero for Kalman Filtering. Using the $\vec{x}$, $\vec{u}$, $\vec{z}$ vectors, along with the matrices F, B, G, H and D, and the noise vectors $\vec{v}$, $\vec{w}$, the complete continuous system can be described as follows:

$$\begin{aligned} \dot{\vec{x}} &= F(t)\vec{x}(t) + B(t)\vec{u}(t) + G(t)\vec{w}(t) \\ \vec{z} &= H(t)\vec{x}(t) + D(t)\vec{u}(t) + \vec{v}(t) \end{aligned} \tag{5.1}$$

This system can be discretized, which is needed for computing the system on a computer, yielding:

$$\begin{aligned}\vec{x}_{k+1} &= \Phi_{k+1,k}\,\vec{x}_k + \Psi_{k+1,k}\,\vec{u}_k + \Gamma_{k+1,k}\,\vec{w}_{d,k}\\ \vec{z} &= H_{k+1}\,\vec{x}_{k+1} + D_{k+1}\,\vec{u}_{k+1} + \vec{v}_{k+1}\end{aligned} \tag{5.2}$$

The measured data is calculated as follows:

$$\vec{z}(t) = \vec{h}(\vec{x}(t), \vec{u}(t), t) + \vec{v}(t) \tag{5.3}$$

The Extended Kalman Filter executes mainly seven steps for each timestep. The first step is to calculate the one-step ahead prediction. This is done by integrating the function $\vec{f}(\vec{x}, \vec{u})$ from time k to k+1, as such:

$$\vec{x}_{k+1,k} = \vec{x}_{k+1,k} + \int_{t_k}^{t_{k+1}} \vec{f}(\vec{x}_{k,k}, \vec{u}_k, t)dt \tag{5.4}$$

When using this equation numerically on a computer, the integral becomes a summation.

The collection of these equations describe the vector function $f(\vec{x}(t), \vec{u}(t), t)$. The jacobian of $f(\vec{x}(t), \vec{u}(t), t)$ is called $F(\vec{x}(t), \vec{u}(t), t)$. Similarly, the jacobian of $h(\vec{x}(t), \vec{u}(t), t)$ is called $H(\vec{x}(t), \vec{u}(t), t)$. So:

$$\begin{aligned}F(\vec{x}(t), \vec{u}(t), t) &= \partial_{\vec{x}}\, f(\vec{x}(t), \vec{u}(t), t)\\ H(\vec{x}(t), \vec{u}(t), t) &= \partial_{\vec{x}}\, h(\vec{x}(t), \vec{u}(t), t)\end{aligned} \tag{5.5}$$

The second step calculates the jacobians F and H as stated in equation 5.5. The third step discretizes the system into the Φ and Γ matrices. The fourth step calculates the covariance matrix of state prediction error ($P_{k+1,k}$), using:

$$P_{k+1,k} = \Phi_{k+1,k} P_{k,k} \Phi_{k+1,k}^T + \Gamma_{k+1,k} Q \Gamma_{k+1,k} \tag{5.6}$$

The third step determines the Kalman Gain:

$$K_{k+1} = P_{k+1,k} H^T (H P_{k+1,k} H^T + R_{k+1})^{-1} \tag{5.7}$$

The fourth step updates the measurement according to the calculated Kalman gain:

$$\vec{x}_{k+1,k+1} = \vec{x}_{k+1,k} + K_{k+1}(z_{k+1} - \vec{h}(\vec{x}_{k+1,k}, \vec{u}_{k+1})) \tag{5.8}$$

Finally, the fifth step calculates the covariance matrix of state estimation error ($P_{k+1,k+1}$):

$$P_{k+1,k+1} = (I_n - K_{k+1} H) P_{k+1,k} \tag{5.9}$$

This process is repeated for all timesteps.

5.1.2. Unscented Kalman Filter

An alternative to the Extended Kalman Filter is the Unscented Kalman Filter (UKF) as introduced by Julier and Uhlmann [30]. The difference between the UKF and EKF is that the UKF addresses approximation problems of the EKF. In the EKF, the state distribution is modeled by a Gaussian Random Variable (GRV) to which a first order linearization is applied. This first order approximation can produce large errors in the mean and covariance of the calculated output. This results in performance and divergence issues. The UKF solves this problem by sampling the state distribution using multiple ($2N + 1$, with N the number of dimensions) sample points. This estimates the mean and covariance up to the third order of a Taylor expansion. The computational effort required by the UKF is comparable to that of the EKF. Another advantage of the UKF over the EKF is that no calculation of the Jacobians or Hessians are required. [65]

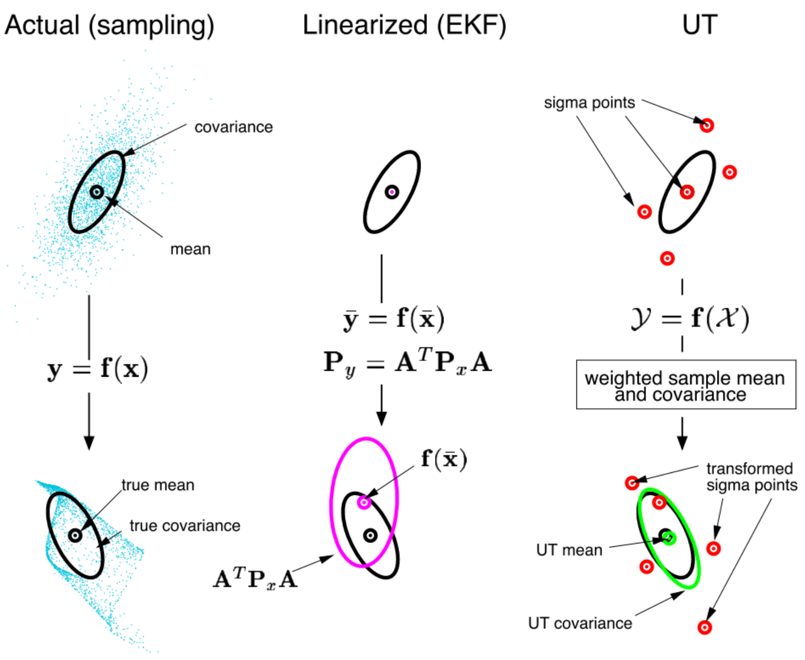

Actual (sampling)
Linearized (EKF)
UT
covariance
mean
sigma points
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T\mathbf{P}_x\mathbf{A}$
$\mathcal{Y} = \mathbf{f}(\mathcal{X})$
weighted sample mean and covariance
$\mathbf{f}(\bar{\mathbf{x}})$
true mean
true covariance
$\mathbf{A}^T\mathbf{P}_x\mathbf{A}$
transformed sigma points
UT mean
UT covariance

Figure 5.1: The Unscented Transform. Source: Wan et al. [65]

5.2. Covariance Intersection

The Kalman Filter and Extended Kalman Filter assume that the correlations between and among the state estimates and measurements are uncorrelated. However, in a large number of applications, this assumption does not hold. The covariance intersection (CI) technique proposed by Julier and Uhlmann [31] solves this problem.

Taking two variables, A and B, for example measurements and assuming they are to be fused together to form C. The means are denoted $\hat{\mathbf{a}}$, $\hat{\mathbf{b}}$ and $\hat{\mathbf{c}}$. The covariances are denoted **A**, **B** and **C**. The properties $\hat{\mathbf{c}}$ and **C** can be calculated via equation 5.10. [31]

$$\begin{aligned} \mathbf{C}^{-1} &= \omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1} \\ \hat{\mathbf{c}} &= \mathbf{C}\left(\omega\mathbf{A}^{-1}\hat{a} + (1-\omega)\mathbf{B}^{-1}\hat{b}\right) \end{aligned} \tag{5.10}$$

Now, the equation $x^T P^{-1} x = 1$ gives two covariance ellipses for A and B. Using the equation 5.10, the updated covariance ellipse is located inside the region of intersection of the covariance ellipses A and B. ω chooses which ellipse is selected as update. ω has to be chosen by optimizing a criterion of uncertainty, such as $|\mathbf{C}|$. The difference between the ellipses corresponding to Covariance Intersection and the original Kalman covariance ellipses are shown in figure 5.2.

Using CI for SLAM, one can save 100 times less storage and computational costs, while having a less than three times larger standard deviation. Therefore, the performance gain of utilizing the cross covariance information between A and B is much smaller compared to the quantity of computational resources that is required.

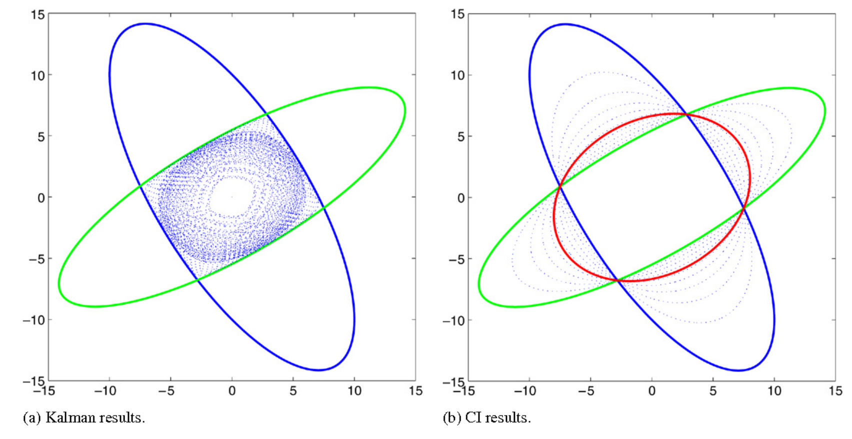


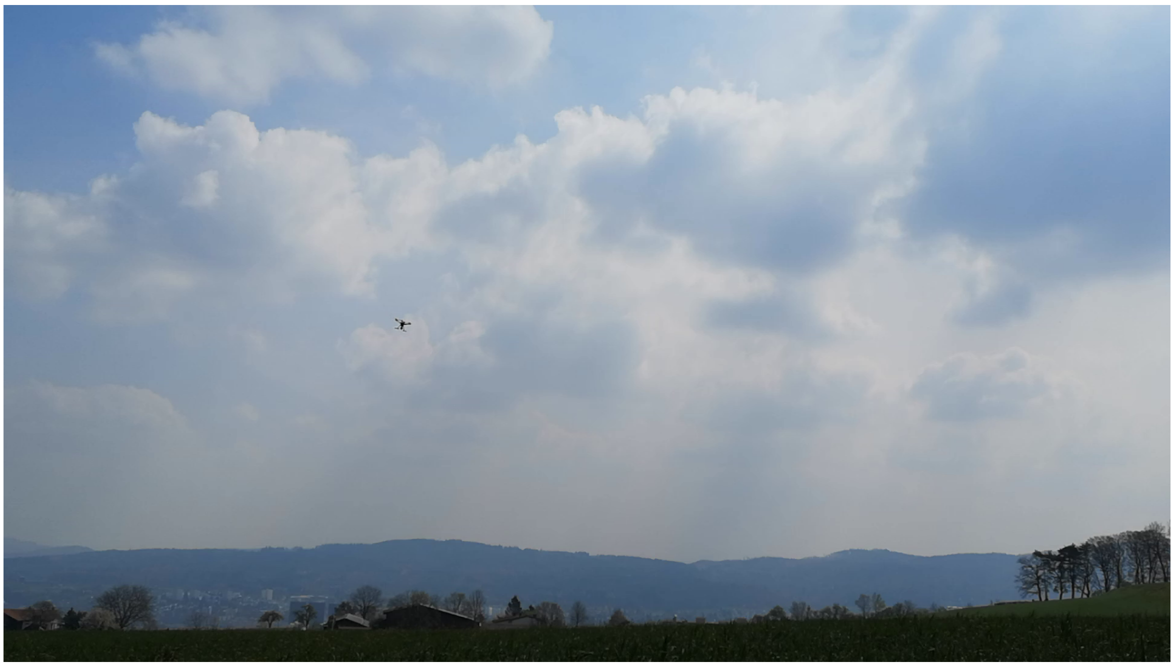
Figure 5.2: Covariance Intersection. Source: Wan et al. [65]

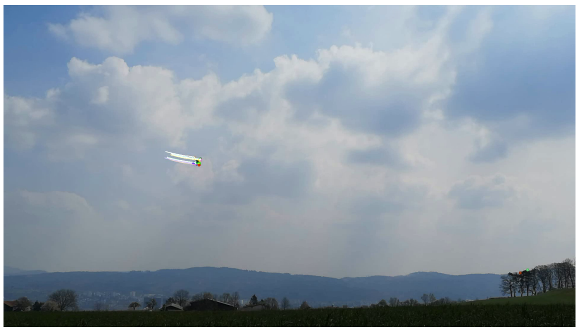
6 Preliminary Results

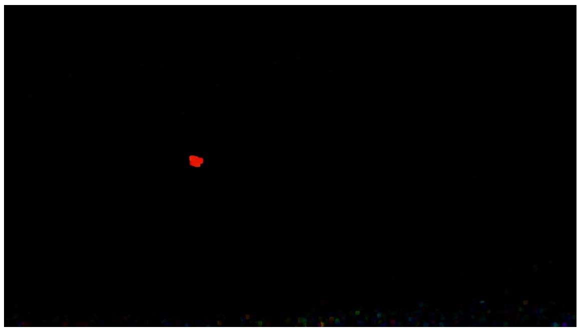
This chapter presents the preliminary results obtained during the literature study. These results act as a first feasibility check for optical flow based moving object detection. In order to find a proper MAV detection algorithm based on optical flow, first the optimal OF method has to be found. Therefore, Lucas-Kanade and Farnebäck were analyzed.

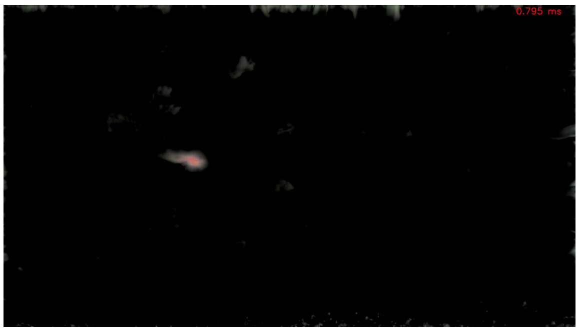
6.1. Comparison between various optical flow methods

In figure 6.1, different optical flow methods are applied to the dataset of Li et al. [37]. Specifically, camera 1 (cam1) from dataset 2 is used. The videos are taken from a static camera and focus on an MAV that is moving. Qualitatively speaking, FlowNet performs the best as area of flow matches the real flow more accurately than Farnebäck and LLUVIA. Farnebäck and LLUVIA indicate a greater area of flow around the MAV than is physically there. Moreover, LLUVIA generates flow in parts of the image where there is no optical flow which results in noise. Lucas-Kanade only tracks a low number of features on the MAV and therefore is less accurate. FlowNet2 performs best of these four methods in KITTI [22] and MPI-Sintel [68] benchmarks as well, so it is not surprising that applies here as well.


(a) Original frame.


(b) Lucas-Kanade.


(c) Farnebäck.



(d) LLUVIA.

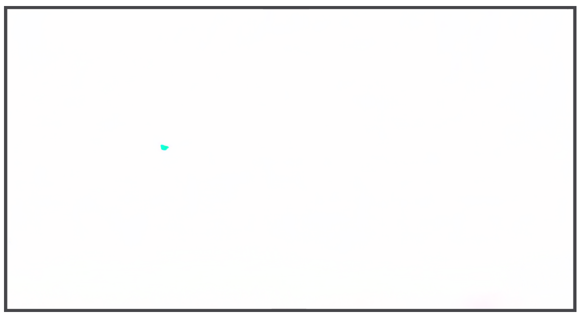
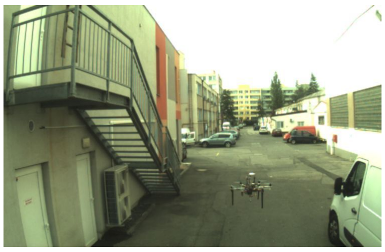
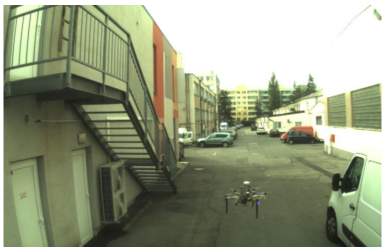

(e) FlowNet 2.0.

Figure 6.1: Different types of optical flow used to detect MAVs. Dataset: [37].

6.2. FlowNet2 applied to the MIDGARD dataset

As FlowNet2 showed most potential, this method will be investigated in more detail. In figure 6.2, FlowNet2.0 is applied to the MIDGARD dataset [64]. Specifically, the warehouse-transition scenario of the indoor-modern folder is used. In this case, the images are taken from an MAV and have one or more MAVs in their FoV. Both the observing and the observed MAVs are flying and in motion. Using the input images shown in a and b, one obtains a optical flow image in which the MAV is clearly visible and easy to detect using a proper moving object detection algorithm focused on detecting MAVs. In figure 6.3, the optical flow image clearly makes the MAV more distinguishable. In the original two input frames, the MAV is hard to recognize due to the low illumination and low contrast relative to the background. However, the optical flow output successfully yields the different flow of the MAV compared to the background pixels. Unfortunately, in a high number of input frames, the MAV has little motion in the image and therefore the output of FlowNet2 does not clearly visualize the MAV, as can be seen in figure 6.4. This could be solved by combing the optical flow detection method with appearance based object detectors or by taking information of the previous frames into account and assuming that if there is no flow, the MAV remains at the same position in the image.


(a) Input image 1.


(b) Input image 2.

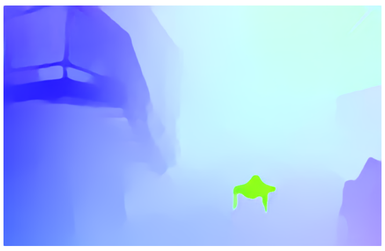
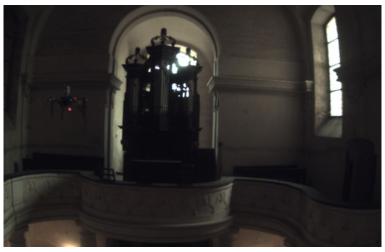
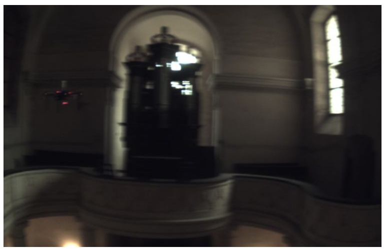

(c) FlowNet2 output image.

Figure 6.2: Situation in which works succesfully in terms of highlighting the MAV. Dataset: MIDGARD [64].


(a) Input image 1.


(b) Input image 2.

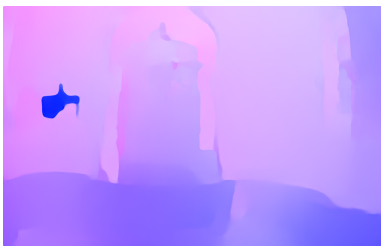
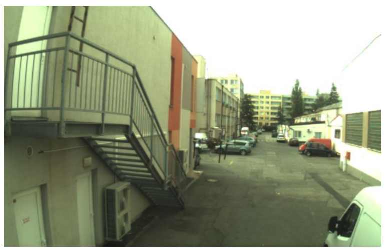
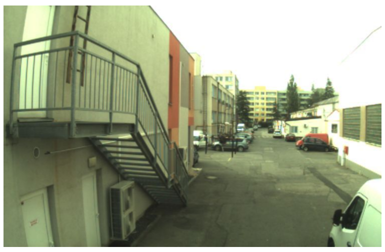

(c) FlowNet2 output image.

Figure 6.3: Situation in which the MAV is poorly recognizable in the original frames, but much easier to detect in the output of Flownet2. Dataset: MIDGARD [64].


(a) Input image 1.


(b) Input image 2.

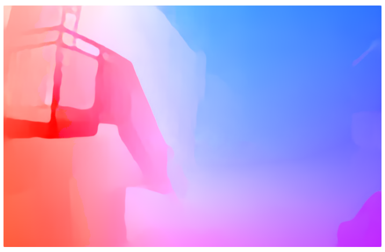

(c) FlowNet2 output image.

Figure 6.4: Situation in which the MAV is not moving compared to the camera and the MAV is unrecognizable in the output of Flownet2. Dataset: MIDGARD [64].

6.3. Background subtraction using Focus of Expansion

A method of subtracting the background from an image was explored by use of Lucas-Kanade optical flow. First, approximately 2000 features are detected and tracked. If there are less than 1000 features, new features will be detected and tracked. The total flow per features for around ten frames is considered to increase robustness. Now, if the camera moves, the intersection of all tracks of the static features is a single point called the Focus of Expansion (FoE). The features that do move in physical space will, generally speaking, not point towards the FoE. These features can be extracted as the dynamic features. The FoE is calculated by taking for each track a random other track and calculating the point of intersection. The FoE is then the median of all intersection x and y values. RANSAC could be used as a more sophisticated method of removing outliers. In figure 6.5, the method is presented where the green lines are static feature tracks and the red lines are dynamic feature tracks. The FoE is the red dot in the center of the image. Therefore, in this case, the camera is moving forward.

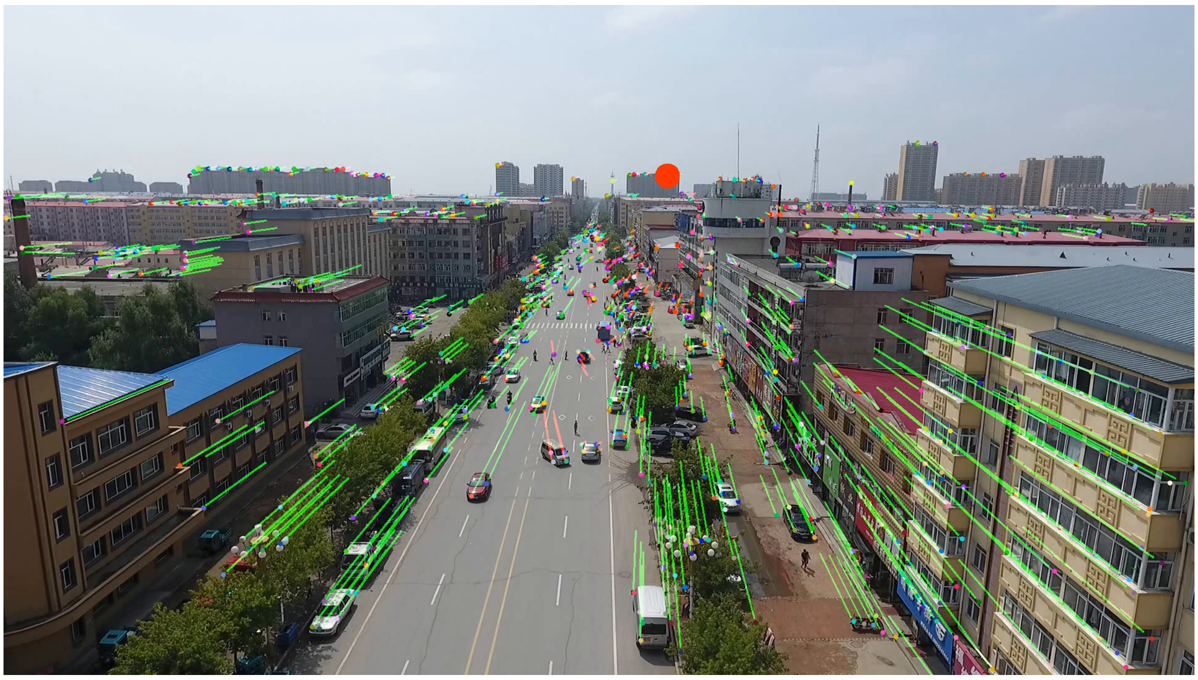


Figure 6.5: Dynamic feature detection using Lucas-Kanade optical flow. The red dot is the calculated FoE. Green lines represent the flow of static features, red lines those of dynamic features. Dataset: VisDrone [70, 71].

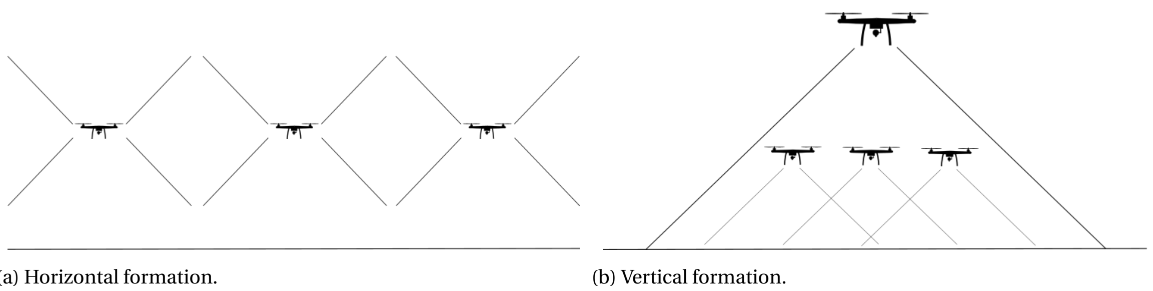
7 Proposed method

This chapter describes the method that is selected for the thesis work after the literature study. In the next section, the use case is described. Furthermore, the research objective and research questions are explained.

7.1. Use case

This project focuses mainly on navigation for swarms in GPS-denied environments. In this context, an MAV will be considered to be a quadcopter, as it is the most popular type of MAV and most datasets are focused on this type of MAV. Different types of formations can be explored. in which MAVs have to be detected. The formation of MAVs can either be: one MAV (MAV A) on top of other MAVs, where MAV A has a camera with a field of view that spans the other MAVs (see figure 7.1). Alternatively, all MAVs can fly side-by-side, with cameras pointing horizontally towards each other or only downward facing cameras that partially overlap per MAV.

The use case of this research is that swarms can be investigated as a faster, more persistent and possibly less expensive way to conduct ISR missions. ISR missions aim to search areas for possible hostile or neutral objects relevant before entering the area. The operational scope will be land-based operations, where areas from 1x1km up to 10x10km are searched.



(a) Horizontal formation.

(b) Vertical formation.

Figure 7.1: Different formations for MAV swarms. The diagonal lines represent the FoV of the cameras mounted on the MAVs.

7.2. Research objective

The main research objective of this thesis is to detect MAVs using optical flow, in order to perform relative localization of MAVs in a swarm.

This objective is aimed at improving the current existing methods of vision-based MAV detection, specifically those that involve optical flow. First, the most suitable and best performing optical flow method will be determined using simple scripts. This optical flow method will be analyzed in the context of MAV detection. This includes analyzing the cases where the method works well, but also analyzing its weaknesses, for instance in cases with little motion. Now, the method can be optimized for MAV detection instead of moving object detection in general. Using the datasets that are openly available, the method will be tested for its effectiveness and accuracy. If proven useful, the optical flow-based method can be combined with neural networks aimed at appearance-based detection, which hopefully increases the accuracy of the method with a trade-off in computational complexity. Experiments will be conducted to assess the accuracy of the detection method in real-life situations.

7.3. Research questions

The main research question for the research, which is based on the literature, is posed as follows:

> "How effective is the use of optical flow, obtained with a monocular camera, when used to detect other MAVs?"

The sub research questions are:

1. What is the maximum range at which optical flow can be used to detect MAVs of a given size?
2. How can one use background subtraction, combined with other sensor data, to improve the MAV positions?
3. What is the optimum in the trade-off of accuracy vs. total computational effort per MAV?
4. Which kind of (lateral) formation is optimal for the optical flow detection technique for MAVs?

7.4. Measures of performance

As this research is mainly focused around the detection of MAVs, the main objective is to develop an algorithm that has very few false positives (FP) and a relatively high number of true positives (TP), as well as few false negatives (FN). By plotting the TP / FP ratio for different parameters, such as a threshold, one hopes to find a very high curvature in this figure, meaning that for a low number of FP, a large number of TP is found. Furthermore, recall and precision are often used as evaluation metrics, where the precision is $\frac{TP}{TP+FP}$ and the recall is $\frac{TP}{TP+FN}$. Ideally, both precision and recall should be one. For the state estimation part, the goal is to have a very small Root Mean Squared Error (RMSE) in the position estimation.

8 Conclusion

This report has given an overview of the existing literature in the domain of vision-based relative localization. The different methods of relative localization are discussed, including marker-based and markerless detections of MAVs, as well as SLAM. Various classical and modern state-of-the-art optical flow approaches are explained, along with their respective advantages and disadvantages.

This project attempts to improve relative localization of MAVs in GPS-denied situations. In literature, remarkably little research has been put into detection of MAVs using optical flow. Therefore, this project aims to apply optical flow to the problem of MAV detection in video feeds. After all, existing literature has shown that combining motion data with appearance information can improve the detection of small moving objects.

Other methods of relative localization involve the use of markers. These markers can make the detection of MAVs easier, as the target has a much more distinct appearance compared to the background. These markers can operate in the visual range of the spectrum, but can also be based on infrared or ultraviolet LEDs. Especially ultraviolet is not very common in nature and therefore is suited for the task of detection. However, markers form an extra requirement on MAVs and require the MAV to carry extra weight and reduce the space available on the MAV. Additionally, and more importantly in the case of large MAVs, LEDs are hard to detect on large distances (>15m).

Because of this relatively small maximum distance, markerless approaches were considered in this project. Markerless approaches include neural networks that detect MAVs based on appearance, optical flow based methods and stereo vision methods. Although stereo vision can perform detections of larger distances and yields a wide range of information, it requires more sensors and has (yet) a relatively large amount of false negatives.

To reduce the time spent on generating a dataset, various datasets were explored and analyzed. However, as pointed out in literature, a low amount of high-quality datasets currently exist that can be used for vision-based MAV detection. The datasets that are best suited for this project are MIDGARD [64], the work by Li et al. [37] and the dataset used for the MSc thesis of Svanstrom [57].

State estimation can be used for drones to calculate the states of an MAV given accelerometer inputs. Additional sensors, such as gyroscopes, barometers, magnetometers and cameras are also often available on MAVs. Kalman Filtering is a form of state estimation. The most suitable Kalman Filter would be the Unscented Kalman Filter, because of its non-linear nature, accuracy and because it does not need the calculation of Jacobian or Hessian matrices. Covariance Intersection can be applied to avoid the problems of correlated measurements.

Based on benchmarks in literature and preliminary results, neural networks such as FlowNet2 prove to be the most promising method of obtaining optical flow. However, it is common that the flow in an image is of low magnitude and therefore it is hard to detect an MAV using optical flow. This could be solved by combing the optical flow detection method with appearance based object detectors or by taking information of the previous frames into account and assuming that if there is no flow, the MAV remains at the same position in the image.

The proposed method for the thesis work follows from the preliminary results. The main research objective of the thesis is to detect MAVs using optical flow, in order to perform relative localization of MAVs in a swarm. This objective is aimed at improving the current existing methods of vision-based MAV detection, specifically those that involve optical flow. In the next chapter, a more detailed description of the future work and planning of the thesis will be given.

9 Planning

This chapter describes the planning of the literature study and thesis. In figure 9.1, the planning is presented using a Gantt diagram. The planning is based on the research objectives introduced in section 7.2.

9.1. Developing and optimizing the MAV detection method

First, the MAV detection algorithm as introduced in the preliminary results will be further developed. The neural network will be used and trained specifically for the detection of MAVs, in particular quadrotors. Furthermore, this detection method will be validated using the datasets given in chapter 4.

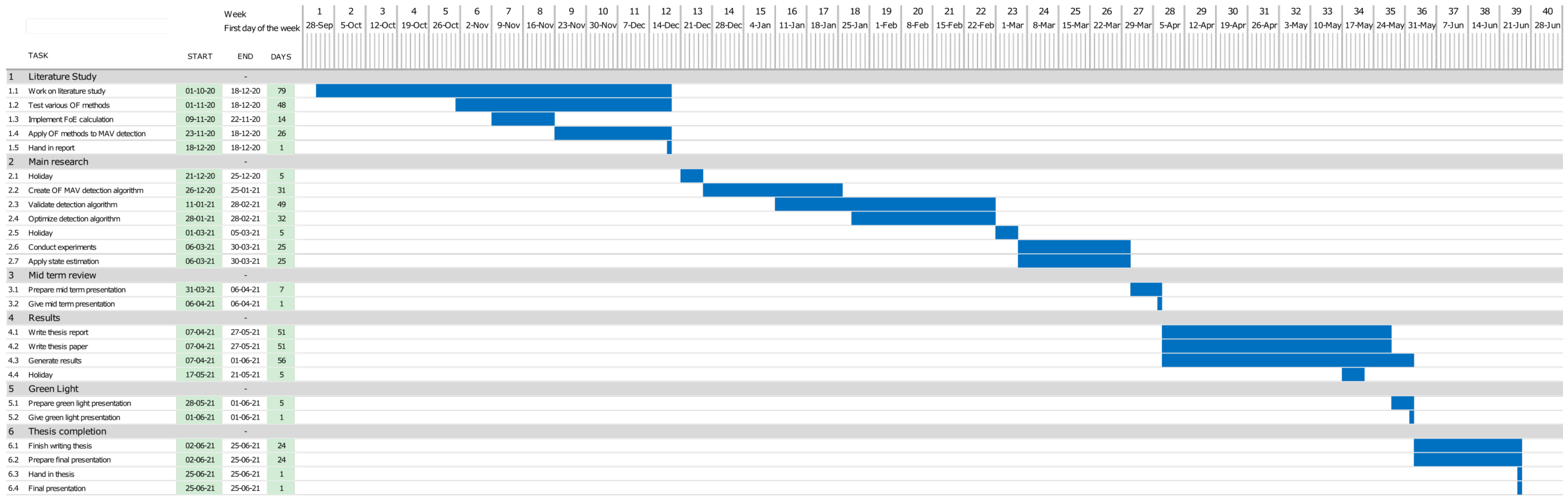
The optical flow detection will be optimized to achieve better performances (computationally-wise) and accuracies. One example of optimization is the combination of optical flow with appearance-based object detection, which can hopefully increase the accuracy of the detections. Additionally, background subtraction is expected to yield better results.

9.2. Experiments

After the development of the detection method, experiments will be conducted in order to prove the robustness in real-life situations. Using cameras mounted on octocopters, other MAVs inside the FoV of these cameras will be attempted to detect using the developed method. Whether the algorithm can be run onboard the MAV during the experiment or not depends on the practical limitations of the setup of the experiments and whether the method is computationally fast enough to run onboard the MAVs.

9.3. State estimation

If there is enough time, state estimation will be applied to perform full relative localization, but in principle this project focuses on MAV detection. Around the same time as the conduction of the experiments, state estimation will be applied to the detection approach to provide a relative localization method. This will complete the relative localization aspect of the method, where state estimation is expected to yield the relative position of other MAVs given the position of the observing MAV. The state estimation step essentially transforms the 2D image coordinates of the detected MAV (in the video feed) to 3D inertial coordinates in the physical world. By combining IMU data using sensor fusion, one can (in theory) create a robust relative localization method.



	TASK	START	END	DAYS
1	Literature Study		-	
1.1	Work on literature study	01-10-20	18-12-20	79
1.2	Test various OF methods	01-11-20	18-12-20	48
1.3	Implement FoE calculation	09-11-20	22-11-20	14
1.4	Apply OF methods to MAV detection	23-11-20	18-12-20	26
1.5	Hand in report	18-12-20	18-12-20	1
2	Main research		-	
2.1	Holiday	21-12-20	25-12-20	5
2.2	Create OF MAV detection algorithm	26-12-20	25-01-21	31
2.3	Validate detection algorithm	11-01-21	28-02-21	49
2.4	Optimize detection algorithm	28-01-21	28-02-21	32
2.5	Holiday	01-03-21	05-03-21	5
2.6	Conduct experiments	06-03-21	30-03-21	25
2.7	Apply state estimation	06-03-21	30-03-21	25
3	Mid term review		-	
3.1	Prepare mid term presentation	31-03-21	06-04-21	7
3.2	Give mid term presentation	06-04-21	06-04-21	1
4	Results		-	
4.1	Write thesis report	07-04-21	27-05-21	51
4.2	Write thesis paper	07-04-21	27-05-21	51
4.3	Generate results	07-04-21	01-06-21	56
4.4	Holiday	17-05-21	21-05-21	5
5	Green Light		-	
5.1	Prepare green light presentation	28-05-21	01-06-21	5
5.2	Give green light presentation	01-06-21	01-06-21	1
6	Thesis completion		-	
6.1	Finish writing thesis	02-06-21	25-06-21	24
6.2	Prepare final presentation	02-06-21	25-06-21	24
6.3	Hand in thesis	25-06-21	25-06-21	1
6.4	Final presentation	25-06-21	25-06-21	1

Figure 9.1: Gantt diagram representing the planning of the thesis.

Bibliography

[1] Juan David Adarve and Robert Mahony. A Filter Formulation for Computing Real Time Optical Flow. *IEEE Robotics and Automation Letters*, 1(2):1192–1199, 2016. ISSN 23773766. doi: 10.1109/LRA.2016.2532928.

[2] Cemal Aker and Sinan Kalkan. Using Deep Networks for Drone Detection. *arXiv*, jun 2017. URL `http://arxiv.org/abs/1706.05726`.

[3] Torsten Andre, Karin Anna Hummel, Angela P. Schoellig, Evsen Yanmaz, Mahdi Asadpour, Christian Bettstetter, Pasquale Grippa, Hermann Hellwagner, Stephan Sand, and Siwei Zhang. Application-driven design of aerial communication networks. *IEEE Communications Magazine*, 52(5):129–137, 2014. ISSN 01636804. doi: 10.1109/MCOM.2014.6815903.

[4] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1879–1892, 2019. ISSN 19393539. doi: 10.1109/TPAMI.2018.2859970.

[5] Simon Baker, Raju Patil, German Cheung, and Iain Matthews. Lucas-Kanade 20 Years On: An Unifying Framework: Part 5. *CMU-RI Report*, 56(3):14, 2004.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. ISSN 10773142. doi: 10.1016/j.cviu.2007.09.014.

[7] Berta Bescos, Carlos Campos, Juan D. Tardós, and José Neira. DynaSLAM II: Tightly-Coupled Multi-Object Tracking and SLAM. *ArXiv*, abs/2010.0, oct 2020. URL `http://arxiv.org/abs/2010.07820`.

[8] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3024 (May):25–36, 2004. ISSN 16113349. doi: 10.1007/978-3-540-24673-2_3.

[9] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, 2009 IEEE:41–48, 2009. doi: 10.1109/CVPRW.2009.5206697.

[10] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7577 LNCS(PART 6): 611–625, 2012. ISSN 03029743. doi: 10.1007/978-3-642-33783-3_44.

[11] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. ISSN 15523098. doi: 10.1109/TRO.2016.2624754.

[12] Adrian Carrio, Hriday Bavle, and Pascual Campoy. Attitude estimation using horizon detection in thermal images. *International Journal of Micro Air Vehicles*, 10(4):352–361, 2018. ISSN 17568307. doi: 10.1177/1756829318804761.

[13] Shu Yun Chung and Han Pang Huang. SLAMMOT-SP: Simultaneous SLAMMOT and scene prediction. *Advanced Robotics*, 24(7):979–1002, 2010. ISSN 01691864. doi: 10.1163/016918610X496946.

[14] Mario Coppola, Kimberly N. McGuire, Christophe De Wagter, and Guido C. H. E. de Croon. A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints. *Frontiers in Robotics and AI*, 7, feb 2020. ISSN 2296-9144. doi: 10.3389/frobt.2020.00018. URL `https://www.frontiersin.org/article/10.3389/frobt.2020.00018/full`.

[15] Jasper De Boer and Mathieu Kalksma. Choosing between optical flow algorithms for UAV position change measurement. *Sc@ Rug*, 2015.

[16] G.C.H.E. de Croon, H.W. Ho, C De Wagter, E van Kampen, B Remes, and Q.P. Chu. Optic-Flow Based Slope Estimation for Autonomous Landing. *International Journal of Micro Air Vehicles*, 5(4):287–297, dec 2013. ISSN 1756-8293. doi: 10.1260/1756-8293.5.4.287. URL `http://journals.sagepub.com/doi/10.1260/1756-8293.5.4.287`.

[17] M. Elbanhawi, A. Mohamed, R. Clothier, J. L. Palmer, M. Simic, and S. Watkins. Enabling technologies for autonomous MAV operations. *Progress in Aerospace Sciences*, 91(March):27–52, 2017. ISSN 03760421. doi: 10.1016/j.paerosci.2017.03.002. URL `http://dx.doi.org/10.1016/j.paerosci.2017.03.002`.

[18] Jakob Engel, Jurgen Sturm, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1456, 2013. ISSN 00201693.

[19] Jan Faigl, Tomas Krajnik, Jan Chudoba, Libor Preucil, and Martin Saska. Low-cost embedded system for relative localization in robotic swarms. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 993–998, 2013. ISBN 9781467356411. doi: 10.1109/ICRA.2013.6630694.

[20] Gunnar Farneback. Two-Frame Motion Estimation Based on Polynomial Expansion. *Lecture Notes in Computer Science*, 2749(1):363–370, 2003.

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus. *Communications of the ACM*, 24(6):381–395, jun 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `https://dl.acm.org/doi/10.1145/358669.358692`.

[22] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. ISSN 10636919. doi: 10.1109/CVPR.2012.6248074.

[23] James J. Gibson. The Perception of the Visual World. *The Philosophical Review*, 60(4):594, 1950. ISSN 00318108. doi: 10.2307/2181436.

[24] Ismail Guvenc, Farshad Koohifar, Simran Singh, Mihail L. Sichitiu, and David Matolak. Detection, Tracking, and Interdiction for Amateur Drones. *IEEE Communications Magazine*, 56(4):75–81, 2018. ISSN 01636804. doi: 10.1109/MCOM.2018.1700455.

[25] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6. Alvey Vision Club, 1988. doi: 10.5244/C.2.23. URL `http://www.bmva.org/bmvc/1988/avc-88-023.html`.

[26] Tak-Wai Hui and Chen Change Loy. LiteFlowNet3: Resolving Correspondence Ambiguity for More Accurate Optical Flow Estimation. *Springer International Publishing*, pages 169—-184, 2020. URL `http://arxiv.org/abs/2007.09319`.

[27] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2020.2976928. URL `https://ieeexplore.ieee.org/document/9018073/`.

[28] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1647–1655, 2017. doi: 10.1109/CVPR.2017.179.

[29] Jianbo Shi and Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, pages 593–600. IEEE Comput. Soc. Press, 1994. ISBN 0-8186-5825-8. doi: 10.1109/CVPR.1994.323794. URL `http://ieeexplore.ieee.org/document/323794/`.

[30] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the Kalman filter to nonlinear systems. In Ivan Kadar, editor, *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, page 182, jul 1997. ISBN 0819424838. doi: 10.1117/12.280797. URL `http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.280797`.

[31] Simon J. Julier and Jeffrey K. Uhlmann. Using covariance intersection for SLAM. *Robotics and Autonomous Systems*, 55(1):3–20, 2007. ISSN 09218890. doi: 10.1016/j.robot.2006.06.011.

[32] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*, 82(1):35–45, 1960. ISSN 1528901X. doi: 10.1115/1.3662552.

[33] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, pages 225–234, 2007. doi: 10.1109/ISMAR.2007.4538852.

[34] Steven Krukowski and Adrien Perkins. Tracking of Small Unmanned Aerial Vehicles. *10TH EUROPEAN CONFERENCE ON RADAR IN METEOROLOGY & HYDROLOGY*, 2018.

[35] Rodney Lalonde, Dong Zhang, and Mubarak Shah. ClusterNet: Detecting Small Objects in Large Scenes by Exploiting Spatio-Temporal Information. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4003–4012, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00421.

[36] Jing Li, Dong Hye Ye, Timothy Chung, Mathias Kolsch, Juan Wachs, and Charles Bouman. Multi-target detection and tracking from a single camera in Unmanned Aerial Vehicles (UAVs). *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4992–4997, oct 2016. ISSN 21530866. doi: 10.1109/IROS.2016.7759733. URL `http://ieeexplore.ieee.org/document/7759733/`.

[37] Jingtong Li, Jesse Murray, Dorina Ismaili, Konrad Schindler, and Cenek Albl. Reconstruction of 3D flight trajectories from ad-hoc camera networks. *arXiv*, 2020. URL `http://arxiv.org/abs/2003.04784`.

[38] H.C. Longuet and K. Prazdny. The Interpretation of a Moving Retinal Image. *Proceding of the royal society of london*, 208(1173):385–397, 1980.

[39] Bruce D. Lucas and Takeo Kanade. Iterative Image Registration Technique With an Application To Stereo Vision. *Proceedings of the International Joint Conference on Artificial Intelligence*, 2: 674–679, 1981.

[40] N Mayer, E Ilg, P Häusser, P Fischer, D Cremers, A Dosovitskiy, and T Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109. URL `http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16`.

[41] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007. ISSN 10504729. doi: 10.1109/ROBOT.2007.364024.

[42] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. ISSN 15523098. doi: 10.1109/TRO.2017.2705103.

[43] Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, Michele Grassi, and Al Savvaris. LIDAR-inertial integration for UAV localization and mapping in complex environments. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 649–656. IEEE, jun 2016. ISBN 978-1-4673-9334-8. doi: 10.1109/ICUAS.2016.7502580. URL `http://ieeexplore.ieee.org/document/7502580/`.

[44] Maciej L. Pawelczyk and Marek Wojtyra. Real World Object Detection Dataset for Quadcopter Unmanned Aerial Vehicle Detection. *IEEE Access*, 8:174394–174409, 2020. doi: 10.1109/access.2020.3026192.

[45] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael Black. Attacking optical flow. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 2404–2413, oct 2019. ISBN 9781728148038. doi: 10.1109/ICCV.2019.00249. URL `http://arxiv.org/abs/1910.10053`.

[46] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, pages 6517–6525. IEEE, jul 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.690. URL `http://ieeexplore.ieee.org/document/8100173/`.

[47] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*, 2015.

[48] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. DeepMatching: Hierarchical Deformable Dense Matching. Technical report, 2015. URL `https://hal.inria.fr/hal-01148432`.

[49] James F. Roberts, Timothy Stirling, Jean Christophe Zufferey, and Dario Floreano. 3-D relative positioning sensor for indoor flying robots. *Autonomous Robots*, 33(1-2):5–20, 2012. ISSN 09295593. doi: 10.1007/s10514-012-9277-0.

[50] Steven Roelofsen, Denis Gillet, and Alcherio Martinoli. Reciprocal collision avoidance for quadrotors using on-board visual detection. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:4810–4817, 2015. ISSN 21530866. doi: 10.1109/IROS.2015.7354053.

[51] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Flying objects detection from a single moving camera. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 07-12-June, pages 4128–4136. IEEE, jun 2015. ISBN 978-1-4673-6964-0. doi: 10.1109/CVPR.2015.7299040. URL `http://ieeexplore.ieee.org/document/7299040/`.

[52] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Detecting Flying Objects Using a Single Moving Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5):879–892, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2564408.

[53] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.

[54] Muhamad Risqi U. Saputra, Andrew Markham, and Niki Trigoni. Visual SLAM and Structure from Motion in Dynamic Environments. *ACM Computing Surveys*, 51(2):1–36, jun 2018. ISSN 0360-0300. doi: 10.1145/3177853. URL `https://dl.acm.org/doi/10.1145/3177853`.

[55] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *arXiv*, pages 1–14, may 2017. URL `http://arxiv.org/abs/1705.05065`.

[56] Amr Suleiman, Zhengdong Zhang, Luca Carlone, Sertac Karaman, and Vivienne Sze. Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones. *IEEE Journal of Solid-State Circuits*, 54(4):1106–1119, apr 2019. ISSN 0018-9200. doi: 10.1109/JSSC.2018.2886342. URL `https://ieeexplore.ieee.org/document/8600375/`.

[57] Fredrik Svanström. *Drone Detection and Classification using Machine Learning and Sensor Fusion*. PhD thesis, Halmstad University, 2020.

[58] Bilal Taha and Abdulhadi Shoufan. Machine Learning-Based Drone Detection and Classification: State-of-the-Art in Research. *IEEE Access*, 7:138669–138682, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2942944.

[59] Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and Vision Computing*, 16 (2):75–87, 1998. ISSN 02628856. doi: 10.1016/s0262-8856(97)00056-5.

[60] Matous Vrba, Daniel Hert, and Martin Saska. Onboard Marker-Less Detection and Localization of Non-Cooperating Drones for Their Safe Interception by an Autonomous Aerial System. *IEEE Robotics and Automation Letters*, 4(4):3402–3409, 2019. ISSN 23773766. doi: 10.1109/LRA.2019.2927130.

[61] Viktor Walter, Martin Saska, and Antonio Franchi. Fast Mutual Relative Localization of UAVs using Ultraviolet LED Markers. *2018 International Conference on Unmanned Aircraft Systems, ICUAS 2018*, pages 1217–1226, 2018. doi: 10.1109/ICUAS.2018.8453331.

[62] Viktor Walter, Nicolas Staub, Martin Saska, and Antonio Franchi. Mutual Localization of UAVs based on Blinking Ultraviolet Markers and 3D Time-Position Hough Transform. *IEEE International Conference on Automation Science and Engineering*, 2018-Augus:298–303, 2018. ISSN 21618089. doi: 10.1109/COASE.2018.8560384.

[63] Viktor Walter, Nicolas Staub, Antonio Franchi, and Martin Saska. UVDAR System for Visual Relative Localization with Application to Leader-Follower Formations of Multirotor UAVs. *IEEE Robotics and Automation Letters*, 4(3):2637–2644, jul 2019. ISSN 23773766. doi: 10.1109/LRA.2019.2901683.

[64] Viktor Walter, Matous Vrba, and Martin Saska. On training datasets for machine learning-based visual relative localization of micro-scale UAVs. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 10674–10680, 2020. ISSN 10504729. doi: 10.1109/ICRA40945.2020.9196947.

[65] E.A. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158. IEEE, 2000. ISBN 0-7803-5800-7. doi: 10.1109/ASSPCC.2000.882463. URL `http://ieeexplore.ieee.org/document/882463/`.

[66] Aaron Weinstein, Adam Cho, Giuseppe Loianno, and Vijay Kumar. Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors. *IEEE Robotics and Automation Letters*, 3(3):1801–1807, 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2800119.

[67] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large Displacement Optical Flow with Deep Matching. In *2013 IEEE International Conference on Computer Vision*, pages 1385–1392. IEEE, dec 2013. ISBN 978-1-4799-2840-8. doi: 10.1109/ICCV.2013.175. URL `https://hal.inria.fr/hal-00873592/documenthttp://ieeexplore.ieee.org/document/6751282/`.

[68] Jonas Wulff and Michael J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June(c):120–130, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298607.

[69] Ryota Yoshihashi, Tu Tuan Trinh, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Differentiating objects by motion: Joint detection and tracking of small flying objects. *arXiv*, 2017.

[70] Pengfei Zhu, Longyin Wen, Xiao Bian, Haibin Ling, and Qinghua Hu. Vision meets drones: A challenge. *arXiv*, pages 1–11, 2018.

[71] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Qinghua Hu, and Haibin Ling. Vision meets drones: Past, present and future. *arXiv*, pages 1–20, 2020.

[72] Zoran Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. *Proceedings - International Conference on Pattern Recognition*, 2(May):28–31, 2004. ISSN 10514651. doi: 10.1109/icpr.2004.1333992.

[73] Danping Zou, Ping Tan, and Wenxian Yu. Collaborative visual SLAM for multiple agents: A brief survey. *Virtual Reality & Intelligent Hardware*, 1(5):461–482, oct 2019. ISSN 20965796. doi: 10.1016/j.vrih.2019.09.002. URL `http://dx.doi.org/10.1016/j.vrih.2019.09.002https://linkinghub.elsevier.com/retrieve/pii/S2096579619300634`.

III

Recommendations and Additional Results

A
Recommendations

In the paper, some recommendations were suggested. In this section, these recommendations are explained in more detail. Additionally, some recommendations are added that are not in the paper.

An aspect that is not covered by the paper is how to perform relative localization. To perform relative localization, two steps have to be added to the pipeline. Firstly, the drones have to be distinguished from the other moving objects. This would have to be performed by the use of an appearance-based method, such as template matching or an appearance-based neural network, depending on the use case. Secondly, to estimate the 3D position of the other drones, not only the direction towards the other drones is needed, but also a measure of distance, for instance by the use of range-based methods. However, these signals can be jammed or spoofed, similar to GPS signals in a GPS-denied environment. An alternative could be to estimate the depth using a monocular depth estimating neural network.

To enable real-time onboard execution of the object detection method on drones, the algorithm has to be made more efficient. FlowNet2 runs on approximately 1.7 Hz on an RTX 2070 for 1920x1024 images. This would be too slow to use in real-time on MAVs themselves. Therefore, the resolution has to be reduced and/or a smaller neural network must be used. The efficiency of the algorithm itself can be improved by porting the code from Python to C++ and making use of multithreading.

To improve the performance of the object detector, a neural network can be trained on images of κ. This neural network will possibly be more accurate at distinguishing moving objects from static objects. Additionally, one could add the magnitude of the (derotated) optical flow field as input for this network as well. This may improve the accuracy further.

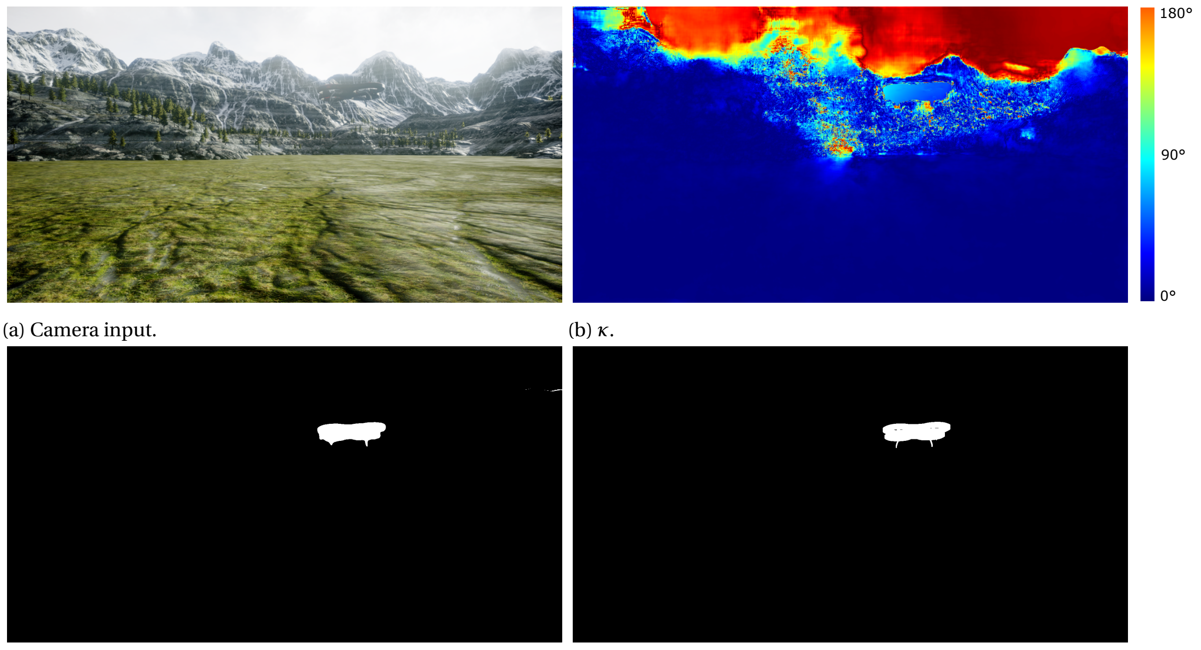
A remarkable result would be to prove this object detection method works on real-world experimental data. In Appendix B, an attempt is made to apply the object detector to experimental data. However, the results are not comparable to the results based on simulation data. The dense optical flow estimated by FlowNet2 does not show a diverging flow field around an FoE as expected for flow generated by a translating camera. Further investigation would be needed to assert the validity of this method in real-world circumstances.

B

Additional results

B.1. Qualitative object detection results

In this section, additional results of the object detector are presented for various situations and environments. The object detector either performs well or it is affected by artifacts, such as shadows, a slightly off sky segmentation or other moving objects in the scene. Figures B.1 to B.4 are demonstrations using the same parameters as figure 11 (a-c) in the paper. The MAV to detect moves sideways towards the right at a constant speed of 0.75 m/s relative to the observer at a distance of 5 meters. The observing MAV moves with a speed of 4 m/s forwards. Figure B.5 is an example where the MAV is on a collision course but too far away to be detected, as FlowNet2 cannot estimate flows of such small objects. All examples use a static threshold of 15°.



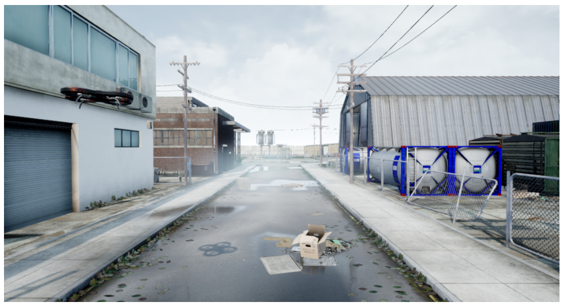
(a) Camera input.

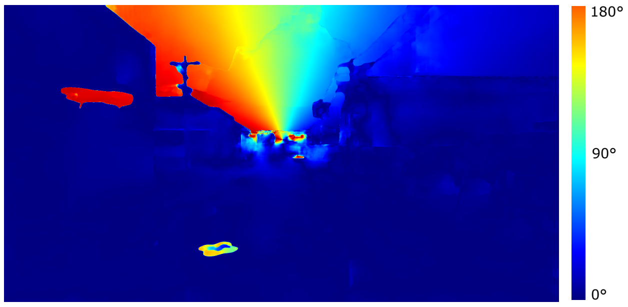
(b) κ.

(c) Result.

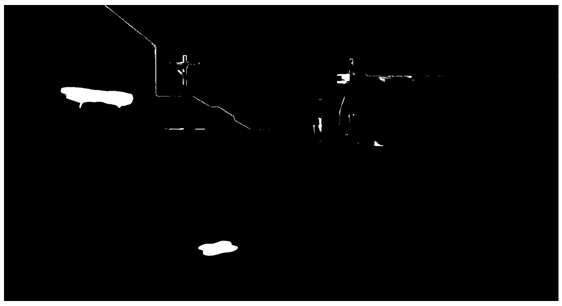
(d) G.t. segmentation mask.

Figure B.1: An example where the motion-based object detector works accurately. TPR: 0.96, FPR: $6.3 \cdot 10^{-3}$.


(a) Camera input.



(b) κ.


(c) Result.

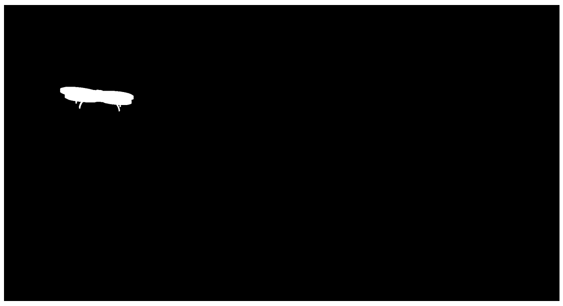
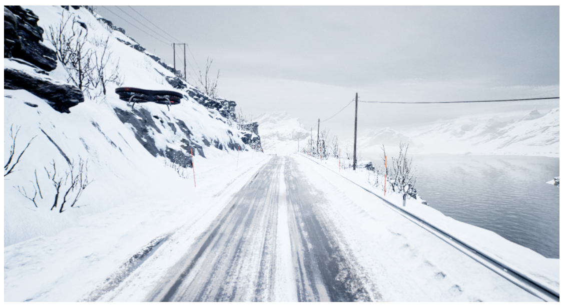
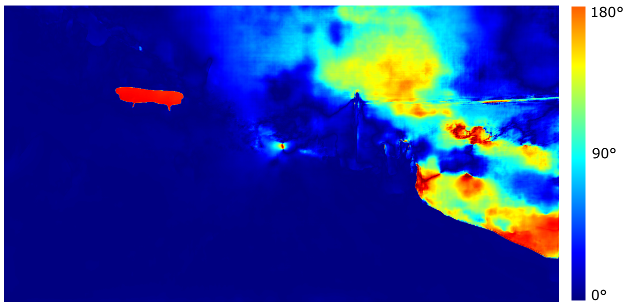
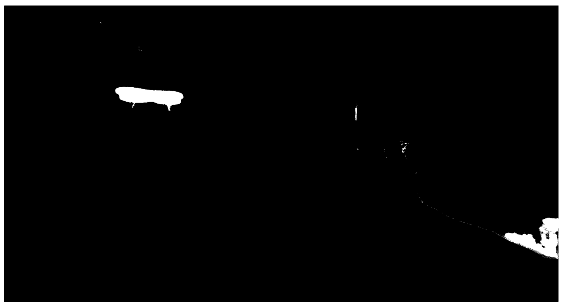

(d) G.t. segmentation mask.

Figure B.2: Hangar scene[1]. Note the edges of buildings and poles where the sky segmentation is slightly off. Also, the shadow of the MAV is clearly detected as a moving "object". TPR: 0.98, FPR: 0.14.


(a) Camera input.



(b) κ.


(c) Result.

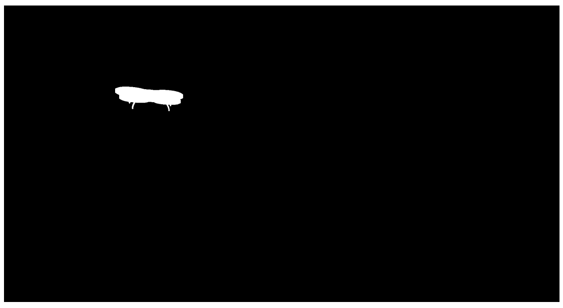
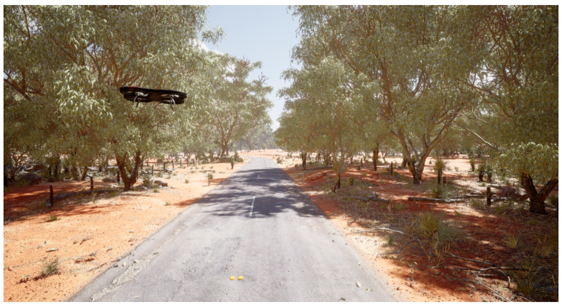
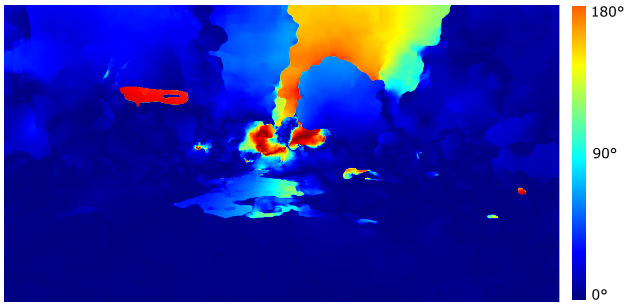

(d) G.t. segmentation mask.

Figure B.3: Automotive Winter Scene[2]. Note the water in the right part of the image introducing a region of false positives. TPR: 0.98, FPR: $5.0 \cdot 10^{-2}$.

[1]`https://www.unrealengine.com/marketplace/en-US/product/industrial-area-hangar`
[2]`https://www.unrealengine.com/marketplace/en-US/product/automotive-winter-scene`


(a) Camera input.



(b) κ.

(c) Result.

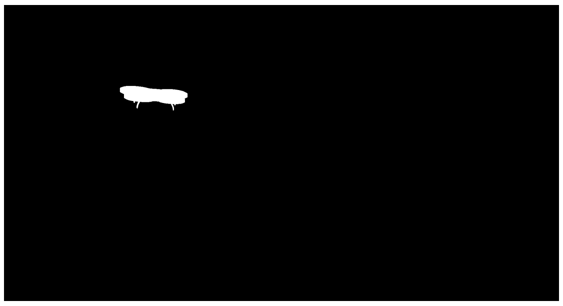
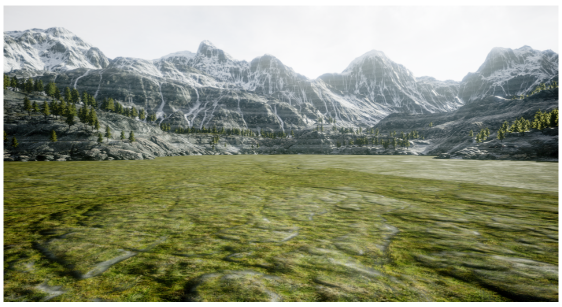
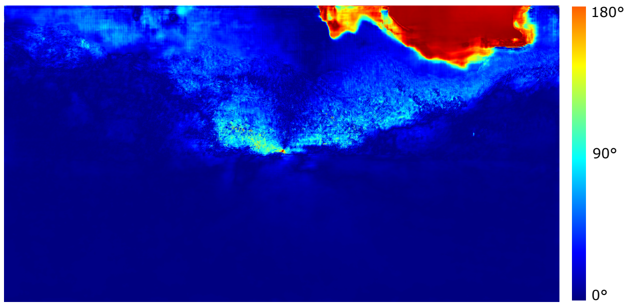
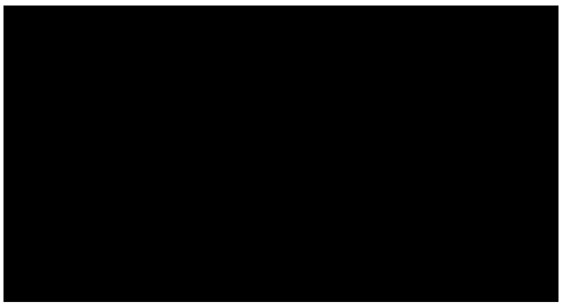

(d) G.t. segmentation mask.

Figure B.4: Rural Australia scene[3]. There is a high amount of false positives due to moving tree leaves and corresponding shadows. TPR: 0.98, FPR: 0.52.


(a) Camera input.



(b) κ.


(c) Result.

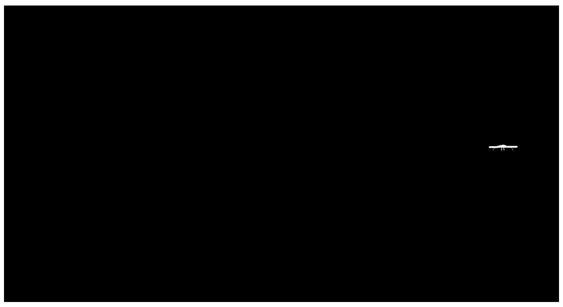

(d) G.t. segmentation mask.

Figure B.5: Collision course of 75°. The MAV to detect is not detected because it is too far away. TPR: 0.0, FPR: $3.6 \cdot 10^{-3}$.

[3]`https://www.unrealengine.com/marketplace/en-US/product/rural-australia`

B.2. ROC curve

In figure B.6, the Receiver Operating Characteristic (ROC) curve is presented for a typical sequence with the LandscapeMountains environment. Note that the false positive rate is very small compared to the true positive rate.

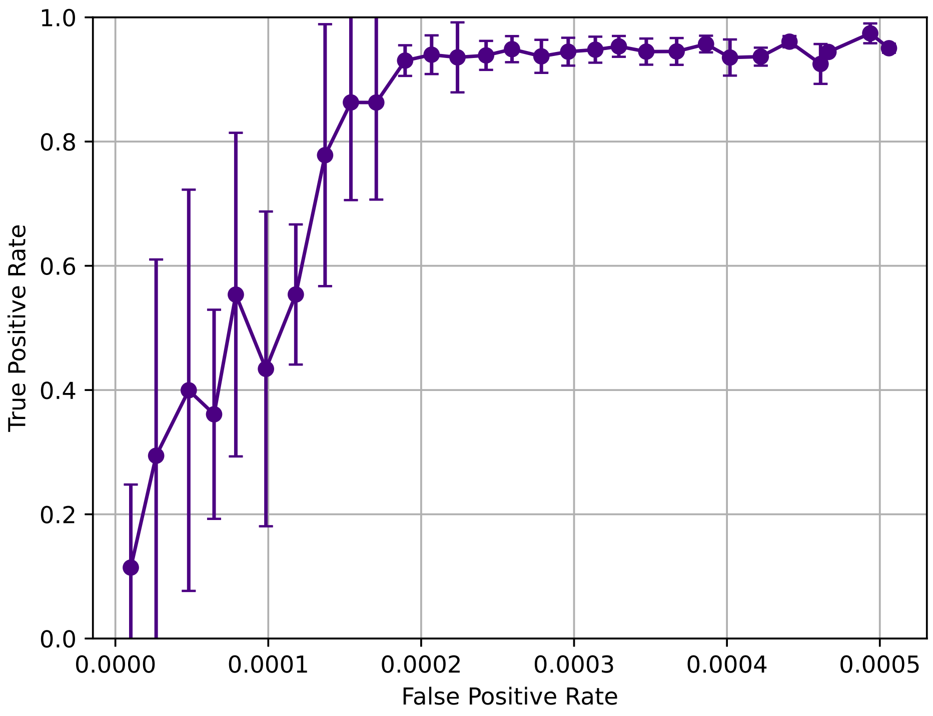
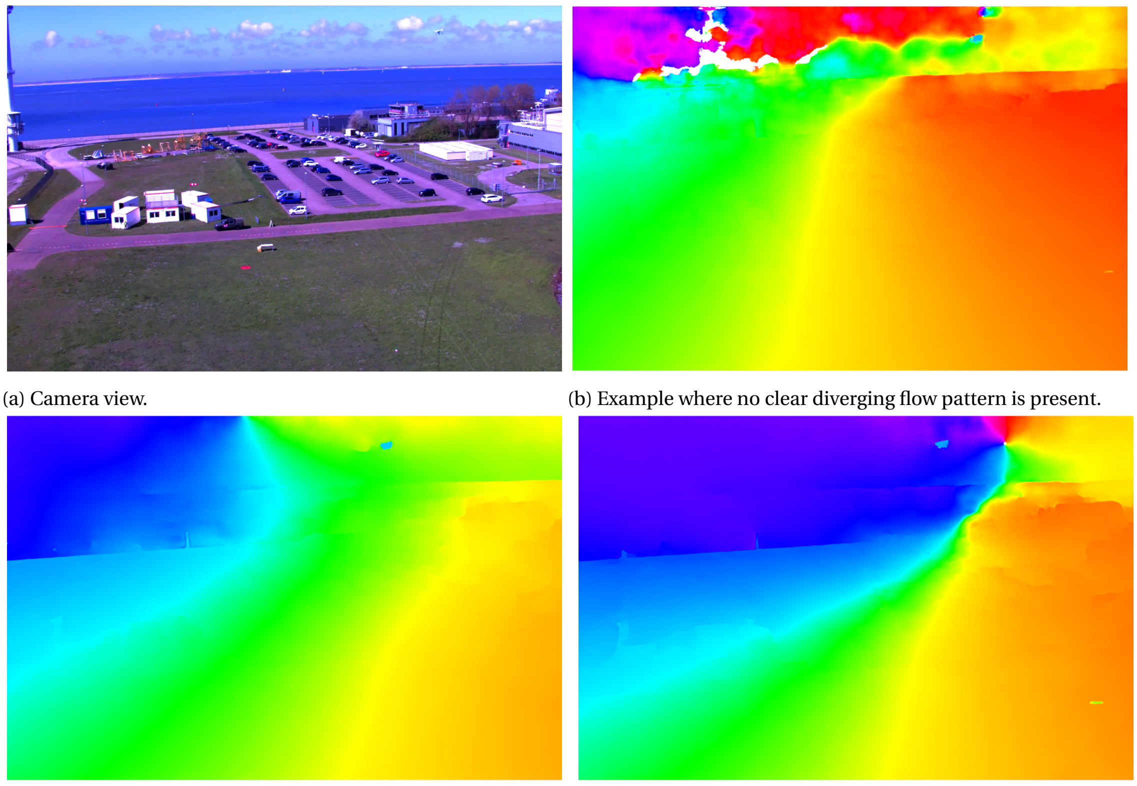


Figure B.6: ROC curve for an MAV to detect moving sideways at 7.7 pixels per frame using a fixed threshold.

B.3. Experimental data

Additionally, the object detector was tested on a real dataset. However, the results are largely affected by poor estimations of FlowNet2 for the angle of the optical flow vectors. In figure B.7, the normalized optical flow fields are presented. The optical flow is, similar to the simulations, derotated using IMU data. The camera and IMU were placed on a gimbal to reduce unwanted vibrations and rotations. The original footage was recorded with a field of view of 47° at 38Hz, but sped up 16 times (by skipping all frames except each 16th frame) to increase the overall magnitude of the flow, which improves the flow estimation. There are almost no moving objects in the scene, except for the drone's shadow (visible in figure B.7) and moving clouds. A second drone is flying in the field of view of the observing drone, from the right side of the image to the left. Both drones fly at a height of approximately 20 meters (because of safety reasons), which is much higher than the drones in the simulations that fly at 2.5 meters above the ground. However, the magnitude of the diverging flow due to the motion of the camera is similar to the simulated data, due to the accelerated speed of the footage.

The three examples demonstrate that the optical flow fields have no clear diverging optical flow vectors pointing away from an FoE, which should be expected. A possible explanation could be that FlowNet2 is not trained for this kind of an environment or that there is too much vibration of the camera, which is not present in the AirSim simulations. These results are included to indicate the challenge that is needed to apply the object detection method to real-world situations.


(a) Camera view.

(b) Example where no clear diverging flow pattern is present.

(c) Example where no clear diverging flow pattern is present.

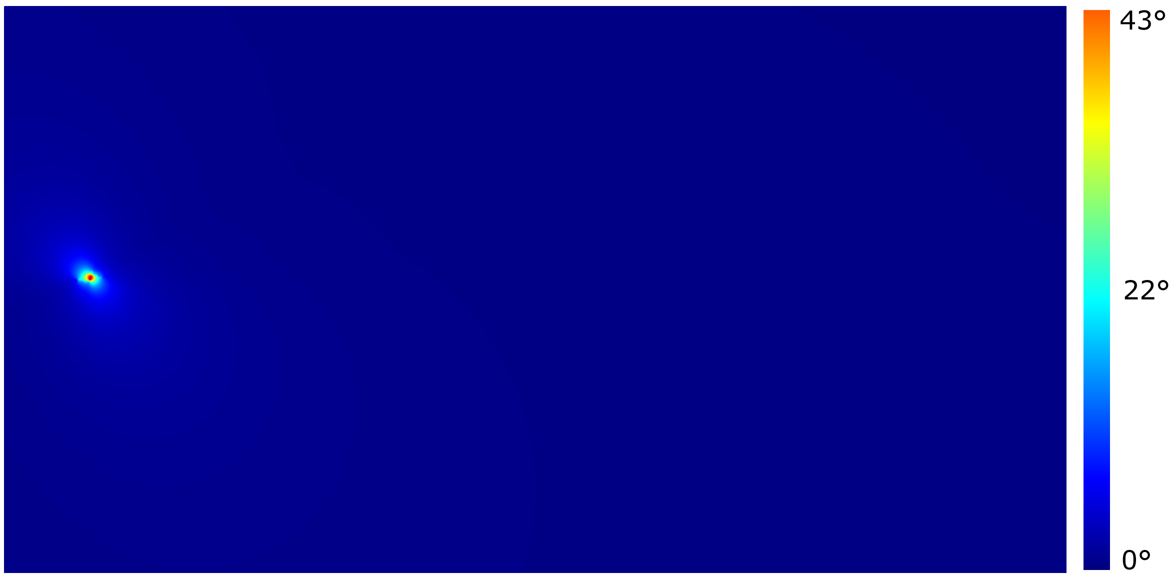
(d) Example where there is a diverging flow pattern visible, but the FoE lies too far to the right and the flow does not diverge from the FoE in straight lines.

Figure B.7: The optical flow fields estimated by FlowNet2 for experimental data, where the magnitude of the flow is normalized to one to visualize the direction of the flow. White pixels have a magnitude of flow smaller than 0.01 px/frame.

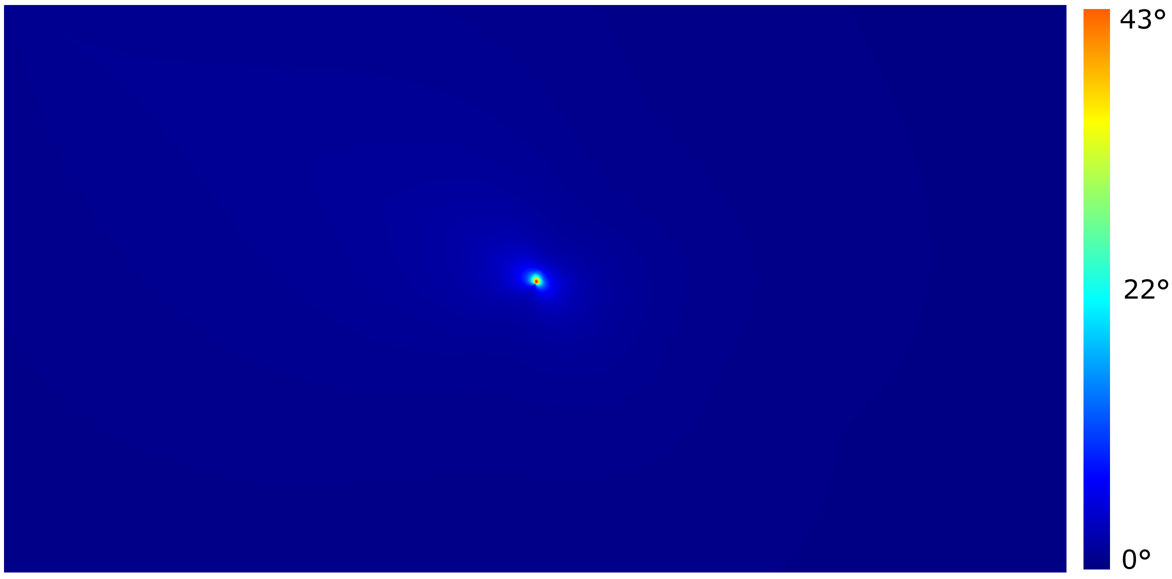
B.4. FoE error analysis

To determine how much the error in the FoE estimation affects κ, the angle between the g.t. FoE and estimated FoE is presented in figure B.8 for three sequences. The three sequences are the same as in figure 8 of the paper. In one sequence, the MAV moves towards the left and therefore the FoE is located in the left part of the image. In the other two, the FoE lies in the center and right side of the image. The results are averaged over 100 images per sequence.

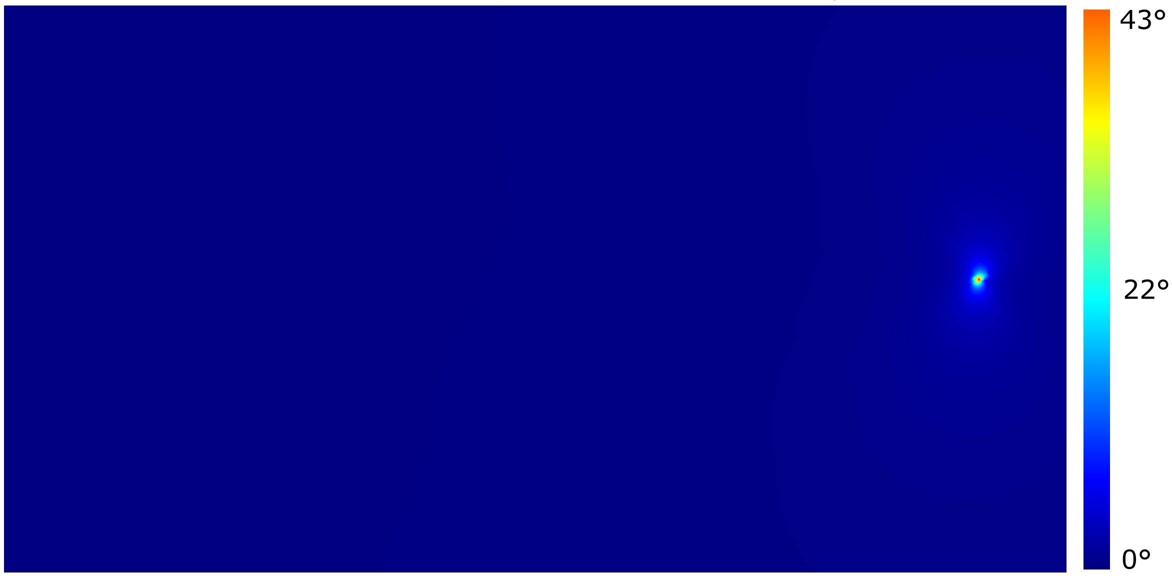
It can be seen that the introduced angular error only affects a relatively small portion of the image. At least 91.4% of the pixels have an angular error of 1° or less and 99.6% of the pixels have an angular error of at most 5°, which is small when compared to a (fixed) angular threshold of 15°. By assuming the error to be only dependent on the distance to the FoE and not the location relative to the FoE, the distance to the FoE can be calculated where the angular error is smaller than 1° or 5°. In the worst case scenario (where the FoE is in the left part of the image), a distance of 52 pixels from the FoE is needed for the error to be less than 5°. At a radius of 232 pixels, the error is 1° or smaller.



(a) FoE positioned in the left part of the image. 91.4% of the pixels have an error less than 1°, and 99.6% of the pixels have an error less than 5°. At a distance of 52 pixels from the FoE, the error is than less 5°. 232 pixels away, the error is less than 1°.



(b) FoE positioned in the center of the image. 94.3% of the pixels have an error less than 1°, and 99.9% of the pixels have an error less than 5°. At a distance of 26 pixels from the FoE, the error is than less 5°. 188 pixels away, the error is less than 1°.



(c) FoE positioned in the right part of the image. 97.0% of the pixels have an error less than 1°, and 99.9% of the pixels have an error less than 5°. At a distance of 28 pixels from the FoE, the error is than less 5°. 137 pixels away, the error is less than 1°.

Figure B.8: Error between the g.t. FoE and estimated FoE (in degrees) for three different locations of the FoE.