Verification & Validation of Focus of Expansion estimation algorithm employing event-based optic flow

MSc Thesis

S.J.F. Knoops

Technische Universiteit Delft

Verification & Validation of Focus of Expansion estimation algorithm employing event-based optic flow

MSc Thesis

by

S.J.F. Knoops

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday November 15th, 2022 at 9:30 AM.

Student number:	4379756	
Project duration:	March 30th, 2021 – November 15th, 2022	
Thesis committee:	Prof. dr. ir. G.H.C.E. de Croon,	TU Delft, supervisor
	Dr.ir. C. de Wagter,	TU Delft
	Dr.ir. C.J.M. Verhoeven,	TU Delft

Preface

Dear reader,

Before you lies my thesis, written to obtain my MSc degree in Aerospace Engineering. Over the past year, I have researched Focus of Expansion estimation using an event camera on a drone. The final product consists of two parts: a scientific paper and a literature study. As you may notice, the subject of the literature study and of the paper differ, as unfortunately the different elements required for the original plan for my thesis were not as expected.

Admittedly, I have not always enjoyed the process of writing my thesis. At times, weeks have gone by without measurable progress, which have not done wonders for my motivation. Working on such a large project, working with software and programming languages barely used before, was overwhelming at first. This led me to approach the problems I had in a non-scientific way, but rather 'trying something and see what happens'. In hindsight, this has cost me a lot of time and I am glad my supervisor at some point reminded me that I need to look into *why* something is not working before trying to fix it. I think this is the most important thing I have learned during my graduation and something I will remember the rest of my career.

Via this way I would like to thank everyone that supported me in completing my studies. First of all, my supervisor Guido de Croon, for his guidance and help, especially in the last few months when my previous supervisor left the university. Next, all my peers and friends, whose mental support I could not have done without. Last, but certainly not least, my parents, who have supported me for my whole studies and tried to keep me as comfortable as possible, even when they had a lot going on themselves.

Looking back, I have been at the faculty of Aerospace Engineering for a bit more than 8 years. Whereas this is long, I think it was completely worth it. Next to my studies, I have held different side jobs, which have allowed me to really feel part of the university and get to know a lot of staff which I otherwise wouldn't have. Next to that, extracurricular activities at the VSV and FSC taught me a great set of skills, varying from public speaking and writing formal letters to graphic design and event organisation. I encourage everyone to spend time on such activities as the skills you pick up you will carry with you the rest of your life, while also you will meet some of your best friends.

Lastly, I would like to make a point regarding mental health. I have seen everyone from students and supporting staff to PhD's and fulltime teaching staff reach their limits. It seems to me that people only focus on results and not on how they got there. I believe no one will enjoy graduating cum laude or publishing their best paper if it has (permanently) damaged their mental health. Partly this is due to high expectations from work or studies, but I feel a large part is also due from people constantly themselves to others, with social media as the great catalyst. I hope, even though not a lot of people will read this, that I can encourage you to set your own goals and prioritise your well-being when necessary. Even though it may be hard to take a step back sometimes, I think in the long run, you will look back happier than you would otherwise.

For now, please enjoy reading my thesis. Even though I, as I wrote before, did not always like it, I am happy with the results and learned a lot in the process. I hope my work will contribute to the faculty's research and perhaps some day maybe make the world a slightly better place.

S.J.F. Knoops
Delft, October 2022

Contents

I

Scientific paper

Verification & Validation of Focus of Expansion estimation employing event-based optic flow

Author: Stefan Knoops; Supervisors: Guido de Croon, Julien Dupeyroux
Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands

***Abstract*—Event based vision has recently attracted a lot of attention. High data rates and robustness to lighting variations make it a valid option for indoor navigation. The previously developed FAITH algorithm calculates a possible Focus of Expansion area based on negative half-planes generated by optic flow and by employing a RANSAC search, a fast and reliable Focus of Expansion estimation can be performed. This paper builds upon this algorithm by verifying and validating the algorithm, improving the derotation capabilities and optimising for computational efficiency. Compared to earlier work, a higher accuracy and an increased robustness are realised by improving the data handling. Simulator results show accuracies in the range of 2 to 5 degrees. Online testing on a drone shows accuracies of up to 5 degrees while obtaining calculation times of only $2 \cdot 10^{-3} s$ and rates of $140 Hz$. Comparing the method to an alternative shows higher accuracy and better suitability to normal flow. Further research may contribute to more stable results and explore different hardware solutions.**

***Index Terms*—Event-based, optic flow, Focus of Expansion, heading estimation, planefitting,**

I. Introduction

In recent years, event cameras, with their high dynamic range, low weight and efficient and fast method of data collection, have shown much promise in several applications. One of which is in autonomous indoor navigation, where the high frequency of data collection enables quick decision-making in cluttered environments. In contrast to classic cameras, which are bound to a relatively low framerate and suffer from inefficient data collection as they transfer the same information several times.

Indoors, fast obstacle detection and avoidance are critical to autonomous flight. In outdoor environments, this often can be done while employing different types of sensors (e.g. sonar) and more freedom in the flight envelope. Indoors there is little margin to avoid objects and not all sensors are suited due to reflections and cluttering. This leads to a wish to develop an algorithm using input from an event camera to avoid objects.

The FAITH (FAst ITerative Half-plane focus of expansion estimation) algorithm [1] makes use of event-based optic flow to determine the focus of expansion in the field of view using a random search for computational efficiency. This is subsequently used to determine whether a collision is bound to happen by estimating the time to contact to an obstacle using the Focus of Expansion (FoE). Initial results showed an 80% success rate, however validation was lacking, derotation was not working and the computational performance needed improvement.

This paper describes the verification and validation of the FAITH algorithm. The algorithm is rewritten from scratch in native C++ for computational efficiency and has improved robustness and flexibility, as well as two-dimensional output to allow for 3D heading estimation. It is tested both in simulation and onboard a computer suitable for onboard flying and is compared to an alternative algorithm.

First, the backgrounds of event-cameras, optic flow and FoE estimation are explained in section II. Subsequently, the FAITH algorithm and this paper's contributions are elaborated upon in section III. Afterward, in section V and section VI, results of the focus of expansion estimation for simulated events as well as onboard estimations will be presented and discussed. Lastly, the algorithm's current limitations and recommendations for improving these are discussed in section VII.

II. Background

Over the past years, bio-inspired approaches have become more prominent in presence. Whereas classic research tends to formulate a mathematical solution for problems, more and more problems are solved by training neural networks. The bio-inspired approach is also used in other sectors, such as materials that try to mimic organic matter [2] or processors that are designed to mimic neurons [3]. Another bio-inspired hardware solution is an event camera, or neuromorphic camera, which mimics how the retina perceives images. The first electronic representations of the retina were already developed in the 70's [4]. Years later, an image of a cat captured by a silicon retina [5]. More recently, low latency and high-resolution cameras have been developed [6]–[9].

An event camera simulates the retina in that instead of capturing a frame, the change in intensity per pixel is provided. A pixel monitors the light intensity and checks whether the difference between the last stored intensity and the current intensity exceeds a threshold. When the difference exceeds the threshold, an event is created, including the timestamp t, the pixel's (x, y) position, and the polarity p. Events are created asynchronously and independently. This allows the data to be provided per pixel instead of per frame. This has two main advantages: the first is that pixel independence allows for efficient data transfer, as no irrelevant data is sent. This differs from frame-based cameras, which publish the information of all pixels at the same time, even when the information is identical to the previous. Secondly, the asynchronous nature of the camera allows for low latencies: readout rates can be up to 1200 MHz [10]. A typical rate of data transfer is 300

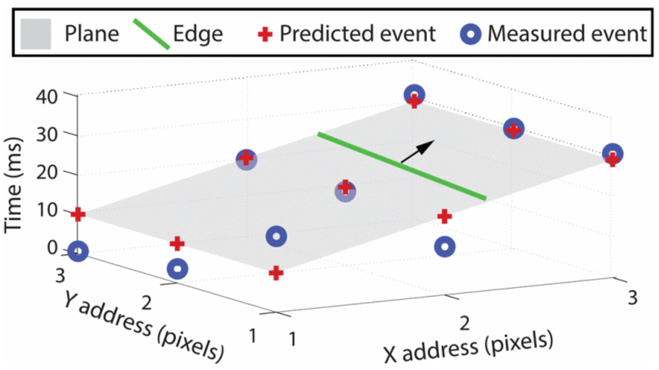


Figure 1: Illustration of the planefitting process. Over time, events at different times and pixel locations approximate a plane. This plane can then be used to describe a 2D motion in time. Illustration adopted from [14].

to 3000 Hz per pixel, depending on the brightness [6]. Other advantages of event-cameras include high dynamic ranges (due to pixel independence) and low power use.

The events generated can subsequently be used to determine the optic flow in the image. Optic flow is the movement of a feature over an image plane over time. This feature is often an edge or corner, as these are well-defined in an image and provide a contrast. In [11] and [12], overviews of event-based optical flow methods are presented. [12] compares several algorithms with large differences in processing time. Local plane fits are found among the faster performers, such as the one presented in [13], whereas Lukas-Kanade-inspired algorithms performed slower. This work utilizes a planefitting approach.

Local plane fits are designed to fit a plane to a stream of events, where the plane can be described as

$$ax + by + ct + d = 0 \tag{1}$$

Events along an edge can be modeled as a 3D plane with time as the third dimension, allowing one to model the movement over time. In the simplest form, the optic flow can be described as

$$\begin{bmatrix} u \\ v \end{bmatrix} = -c \begin{bmatrix} 1/a \\ 1/b \end{bmatrix} \tag{2}$$

In Figure 1, the planefit is illustrated. Over time, different events and locations are measured, indicating the movement of an edge. A plane can be fitted to these events in x, y, t space. Describing this plane by Equation 1 provides the parameters needed to find the normal flow.

It is important to note that by tracking the movement of a *plane* and not of a corner feature, only the movement normal to the plane can be determined. Therefore, it is vital to realize that all flow determined by the planefitting algorithm is normal to the edge causing the flow [14].

In the past, Focus of Expansion (FoE) estimation has shown to be an effective method of heading estimation in a wide range of applications [15]–[18], given that the actual rotational rates are known. Apart from [1], no applications include implementation on a real-time application using a neuromorphic camera.

The FoE can be described as the place in an image without translational flow. When a camera moves straight towards the center of its field of view, there is no flow in that exact location, illustrated in Figure 2B. The FoE changes position according to the observer's motion: if the straight motion in Figure 2 would go to the left, the FoE would also move to the left of the image. When the camera intrinsics are known, the FoE pixel coordinates can be translated to the heading of the camera.

In 6-dimensional flight, the FoE is not trivial to estimate, as illustrated in Figure 2A. The FoE appears to be on the right-hand side of the image. However, when decomposing the flow into translational and rotational flow in respectively Figure 2B and C, one can observe that the FoE is actually in the center. Decomposing the optic flow in its translational and rotational components is described in [19]. Given sufficient information on the rotational velocities, the flow can be 'derotated', leaving only the translational flow, which allows for more trivial heading estimation.

Heading estimation in indoor environments is challenging as a vehicle cannot rely on global navigation systems and integrating measurements of the Inertial Measurement Unit (IMU) is prone to drift. Heading estimation via a camera is not subject to drift and does not require outside connectivity. The use of event-cameras in particular also makes the estimation robust to light conditions. Using the event stream for optic flow estimation allows us to estimate the optic flow vectors sparsely. This allows us to quickly estimate optic flow in a cluttered environment, even when there are many events due to the abundance of textures.

The FAITH algorithm has been developed by Dinaux, Wessendorp, Dupeyroux, *et al.* [1] as part of the Comp4Drones project. FAITH uses a RANSAC (Random sample consensus) scheme to quickly find a possible FoE area based on limiting the area by the negative half-planes (see subsection III-C) defined by the normal flow vectors. The algorithm builds on earlier work [20] while optimizing for efficiency. [1] cites an accuracy of 10.06 ± 2.88 degrees with a computational time of $0.05 \pm 0.02s$. However, the trajectories are mostly straight flight, and accuracy is only measured on the x-axis of the frame. This led to the wish to evaluate the FoE estimation's performance in different flight envelopes and further develop its capabilities.

III. Method

This section elaborates on the end-to-end methodology of FoE estimation. Three main calculation steps take place from the event stream to the FoE estimation, each of which is discussed.

A. Planefitting

Events measured by the camera are first processed using a planefitting algorithm. The algorithm is based on [21], which is an adaptation of the algorithm presented in [13].

The planefitting algorithm is based on the theory that events along an edge can be considered a plane in space-time representation. Estimating this plane and, subsequently, its gradient allows for estimating the normal flow in this position.

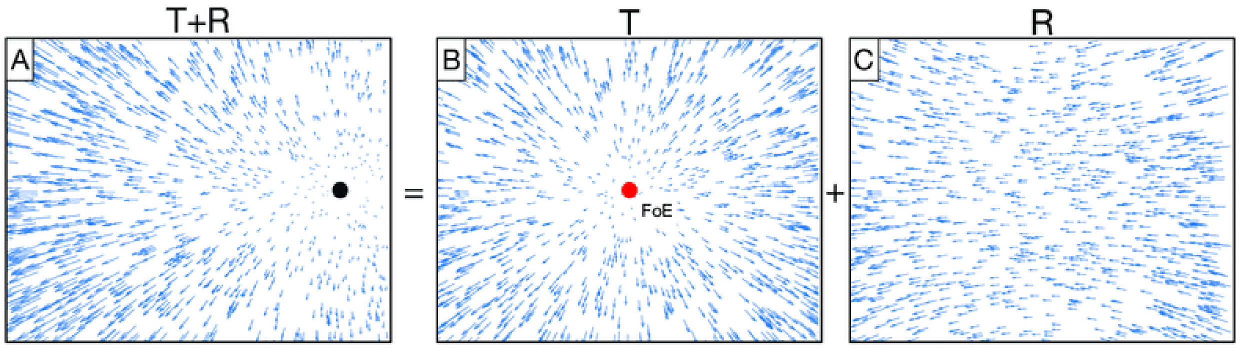


Figure 2: Illustration of optic flow including a rotational component. The FoE in illustration (A) seems to be on the right. However, after decomposing the flow in its translational and rotational components, it becomes clear that the FoE is actually in the center and the camera is under the influence of a yawing rotation.

Among the adaptions in [21], several improve performance on less powerful devices, such as onboard computers. These adaptations concern parameter reduction to decrease complexity and the implementation of a time buffer to maintain real-time performance.

Before calculating the optic flow, the event stream pixel coordinates need to be undistorted to consider camera intrinsics. The x and y pixel coordinates of the events are transformed into a calibrated coordinate, which is found in a lookup table. This allows for representing flow estimation in s^{-1} instead of pixels $\cdot\, s^{-1}$, favourable for derotating the flow as no transformation to pixels is necessary. The lookup table is generated performing Matlab's `undistortPoints` and `estimateCameraParameters` routines on event data, generated by flashing a checkerboard pattern. When using simulated events, a camera object can be generated in Matlab by hand-creating a `cameraIntrinsics` object and performing the `undistortPoints` subsequently.

B. Derotation

Before the FoE can be estimated, the flow needs to be derotated. In the case of regular optic flow, this can be done by subtracting the flow due to rotation in both components. In the case of normal flow, the component of the flow along the vector needs to be removed. Removing the optic flow generated by rotation will, in this case, not change the direction of the flow, as it remains normal to the edge. The flow may be canceled if only rotational flow is present or the flow direction may reverse.

As described in [19], optic flow can be expressed mathematically in terms of ego-motion. The rotational flow in x and y, (u_{rot}, v_{rot}), can be written as

$$u_{rot} = -q + r \cdot y_{nor} + p \cdot x_{nor} \cdot y_{nor} - q \cdot x_{nor} \cdot x_{nor} \tag{3}$$

and

$$v_{rot} = p - r \cdot x_{nor} - q \cdot x_{nor} \cdot y_{nor} + p \cdot y_{nor} \cdot y_{nor} \tag{4}$$

where p, q and r are respectively the rotational velocities in rad/s around the camera $x-, y-, z-$axes and x_{nor} and y_{nor} are the normalised pixel coordinates, taken from the same undistortion table as the planefitting algorithm.

In order to derotate the flow, the argument (the angle between positive x-axis and vector) of both the normal flow vector $\vec{\mathbf{v}}_n$ and the rotational flow vector $\vec{\mathbf{v}}_{rot}$ are taken and subtracted from each other, providing the angle α the vectors make:

$$\alpha = \arg(\vec{\mathbf{v}}_n) - \arg(\vec{\mathbf{v}}_{rot}) \tag{5}$$

which allows us to calculate the magnitude of the rotational flow along the normal flow axis using:

$$|\vec{\mathbf{v}}_{rot,proj}| = \cos\alpha \cdot |\vec{\mathbf{v}}_{rot}| \tag{6}$$

with $|\vec{\mathbf{v}}_{rot,proj}|$ being the magnitude of the projection of the flow due to rotation on the normal vector. This projection is then used to calculate the derotated normal flow according to

$$\begin{bmatrix} u_{derot} \\ v_{derot} \end{bmatrix} = \begin{bmatrix} \cos\arg(\vec{\mathbf{v}}) \\ \sin\arg(\vec{\mathbf{v}}) \end{bmatrix} \cdot (|\vec{\mathbf{v}}| - |\vec{\mathbf{v}}_{rot,proj}|) \tag{7}$$

which returns the magnitude decomposed in its u and v components.

A more elaborate proof of this method is presented in section A.

C. FoE estimation

The FoE estimation is based on the FAITH algorithm proposed in [1]. The FAITH algorithm estimates the most probable FoE as the center of an area bound by negative half-planes formed by the normal flow vectors. The algorithm is described as pseudocode in Algorithm 1 and in the next few paragraphs, the algorithm will be explained.

Initially, the FoE area is delimited by the edges of the frame. As described in section II, the flow vectors move away from the FoE, placing the FoE in the negative half-plane of each vector. A half-plane is the plane defined by the normal to the vector, away from the vector direction. Whereas [20] determines a probability map of all vectors and decides on the most probable location, a RANSAC search is employed in this work. While searching, a random vector gets selected, for which it is checked whether it contributes to making the FoE area smaller or not. A vector contributes when its normal provides intersections with the previous bounds of the possible FoE area and the vector lies within the area. In Figure 3, a

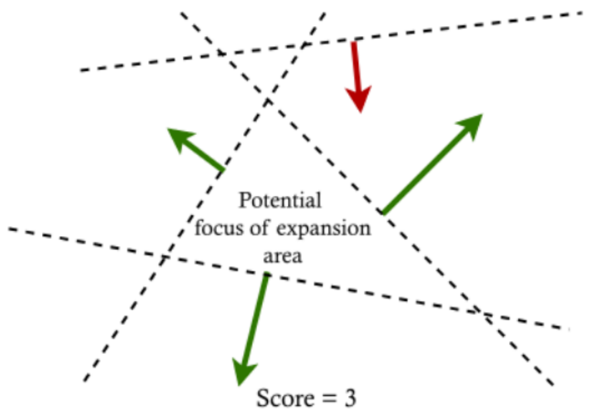


Figure 3: Illustration of the FoE estimation, adopted from [1]. The green vectors contribute to the current hull, whereas selecting the red vector would not futher limit the potential area. If the red vector would have been chosen first, no other vectors would have been able to contribute and the search would stop. The score would equal 2 as the FoE lies in the negative half plane of both the red vector as the bottom vector.

Algorithm 1 The FAITH algorithm with the amount of input vectors N and amount of RANSAC iterations A

```
Require: $N \geq 1$
  MaxScore $\leftarrow$ 0
  BestHull $\leftarrow$ Frame
  for $n \leq$ A do
      Hull $\leftarrow$ Frame
      while searching do
          select random vector $\tilde{\mathrm{v}}$
          determine $\bar{\mathrm{x}}$ where $\bar{\mathrm{x}} \perp \tilde{\mathrm{v}}$
          if $\bar{\mathrm{x}}$ contributes to Hull then
              Hull $\leftarrow \bar{\mathrm{x}}$
              remove redundant vectors from Hull
          else
              stop searching
          end if
      end while
      determine Score
      if Score $>$ MaxScore then
          BestHull = Hull
      end if
      $n++$
  end for
  return FoE
```

schematic example is given. In this example, the red vector is not able to contribute to a smaller area and the maximum score attainable is 3.

If the vector contributes, redundant intersections and hull lines are removed and a new vector is selected. This is repeated until a new vector does not contribute to making the hull smaller. Then, the FoE is calculated as the average x and y coordinate of all the hull intersections. The estimation is then scored according to the number of half-planes the estimated FoE lies in. This process is repeated N times, where N can be defined by the user, after which the highest-scoring FoE estimation is selected.

Some adaptations are made to improve efficiency and accuracy in this work. First of all, a moving average is implemented. As shown in section V, the estimations themselves tend to fluctuate around the actual value as there are $(n_{vectors}!)$ different combinations, with $n_{vectors}$ being the number of flow vectors available. A moving average allows for a smoother estimation curve that decreases the absolute error if the imposed time delay is sufficiently short.

In estimations where only a tiny amount of optic flow vectors is available, several solutions lead to the maximum amount of inliers. In order to find the most accurate estimation, the solution that used the most information as input is chosen, i.e. the solution that used the most vectors and therefore tends to be the smallest area.

Lastly, the publishing of FoE estimations is adapted only to publish when the RANSAC search found a new FoE, thus when it found at least one contributing vector. This has some influence on the publishing rate, however considered worthwhile, as empty estimations do not influence the moving average.

IV. Verification & Validation

An integral part of this research is the verification & validation procedure. As written before, the FAITH algorithm had yet to be adequately verified. Here, every aspect of the algorithm is evaluated to ensure later presented results are valid and do not depend on flukes or limited flight envelopes.

A. Simulator setup

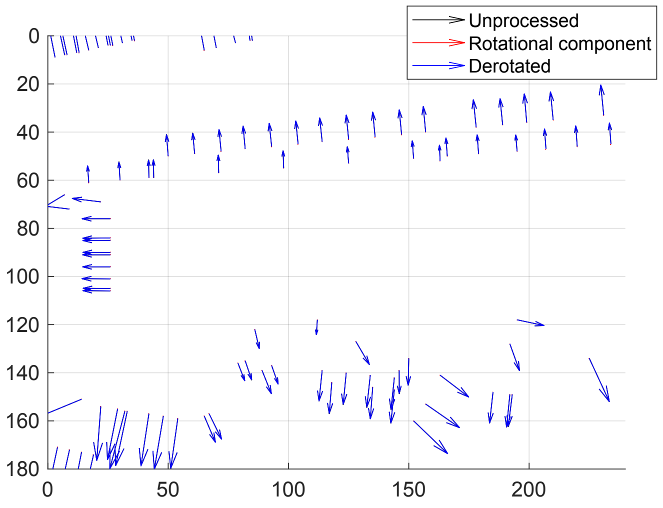
A simulator is used to validate the algorithm. eSim [22] is a simulator specifically developed for event cameras, which has as advantage that a ground truth can be established and experiments can be repeated, in contrast to actual flight, where each test is different from a previous run.

The simulator camera has a diagonal field of view of 73.48° and a 240 by 180 pixels resolution. IMU measurements are provided by eSim, containing information on accelerations and rotations at a rate of 1000 Hz, corrupted with noise and bias if required. Different trajectories are used to validate and test performance in varying circumstances. In particular, optic flow and its derotation are tested in simple trajectories such as straight flight or sole rotations.

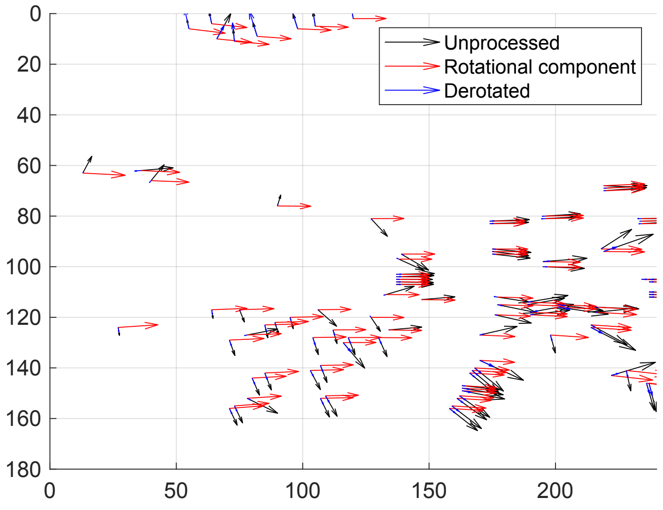
First, the optic flow and derotation are tested in subsection IV-B. Qualitatively, it is checked whether the optic flow generated by the planefitting algorithm confirms to the expectations. Derotation can be tested quantitatively by applying pure rotational motion to the camera, which after derotation should indicate zero translational flow. After that, the actual FoE estimation will be evaluated in subsection IV-C.

B. Optic Flow & Derotation

The optic flow is evaluated qualitatively. It is straightforward what type of flow to expect in certain trajectories, e.g., in a



(a) Derotated flow field in straight flight forward. One can see that there is no rotational flow and thus there is no derotation. Hence the unprocessed flow and derotated flow are identical.



(b) Derotated flow field in straight flight, including a yawing motion. One can see that there is rotational flow along the horizontal axis and the flow gets derotated along its own axis, leaving only the flow due to translation

Figure 4: Optic flow derotation in straight flight and straight flight with added yaw rotation. Figure 4a shows the optic flow in straight flight, Figure 4b shows the optic flow when a yawing motion is added.

straight trajectory, the flow is expected to point outwards, with the flow increasing in magnitude away from the FoE. We can fly at an angle to check whether the FoE moves and thus whether the axis system is implemented correctly.

Figure 4a and Figure 4b show optic flow fields of straight flight and a yawing motion, respectively. It is visible in the first figure that the optic flow moves away from the center of the image while still complying with the edges that are present in the scene. In the second figure, a large yawing motion is present. The optic flow shown in black is normal to the edge, whereas the rotational component is not. The optic flow shows a direction that complies with the rotation. Analogously, these observations can be made for other rotations, yielding similar results, and validating the axis system's correct orientation.

When performing the derotation procedure described in

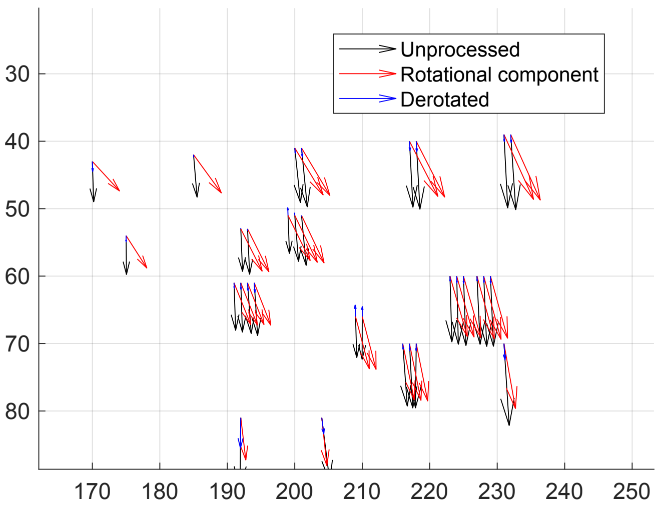


Figure 5: Section of derotated optic flow (not to scale) when the camera is subject to pure pitching motion. One can observe that the derotated flow, shown in blue, is not perfectly zero due to imperfections in the Optic Flow.

section III, one can observe that for pure rotational movements and no translations in space, the derotated flow should equal zero. One can observe small residuals by inspecting the flow as shown in Figure 5. These residuals are due to the planefitting algorithm, which does not determine perfect normal flow. Also, the included IMU noise leads to small differences in the *actual* and the *expected* optic flow generated by rotation. To evaluate the magnitude of this error ϵ, defined by $\epsilon = \vec{v}_{rot,perfect} - \vec{v}_{rot,estimated}$ is reviewed. In these four cases, pure rotational movements are exhibited by the camera. As there is pure rotation, the total flow equals the rotational flow and the derotated flow is equal to the error, which should be zero. Table I shows that the absolute errors approach zero and are relatively small in magnitude compared to the optic flow vector magnitude. Important to note that the mean errors for u and v components are in the order of 10^{-3} and are centered around $\epsilon = 0$, indicating that there is no significant bias.

This inaccuracy can cause small amounts of translational flow to wrongly switch signs, as the error might be larger than the translational flow. In the case of Figure 2, one can observe that such cases happen near the FoE itself. This implies that the possible FoE area may be larger. However, most likely still centered around the FoE as the flow further from the FoE does not suffer this issue. It is therefore assumed that this imprecision does not cause significant issues.

C. *Focus of Expansion estimation*

In order to verify the workings of the FAITH algorithm, we fly simple trajectories and subsequently investigate the output. In more detail, the following properties are verified:

1) Vectors are interpreted correctly.
2) The half-planes are used correctly; the FoE can only be placed on the negative half-plane.
3) The conditions for a vector to qualify as improving the search are implemented correctly.

Manoeuvre	$\mu_{\|\vec{v}\|}$	$\sigma_{\|\vec{v}\|}$	$\mu_{\|\epsilon\|}$	$\sigma_{\|\epsilon\|}$
Pure pitch	0.016077	0.30225	0.00023575	0.049364
Pure yaw	0.2208	0.1407	0.0237	0.0571
Pure roll	0.0060527	0.12227	0.00046476	0.023414
3D rotation	0.10145	0.7465	0.00065173	0.11086

Table I: Results of the derotation in one-dimensional and three-dimensional rotations using the living room environment portrayed in Figure 6b. However, a small error remains after derotation, deemed insignificant compared to the total flow magnitude. In flight, where translational flow is present, this offset will not likely produce significant results.

Parameter	**Value**
Iterations in RANSAC search	100
Minimal amount of vectors required	10
Length FoE buffer	5
Maximum estimation frequency [1]	100

Table II: Standard parameters used in FoE estimation

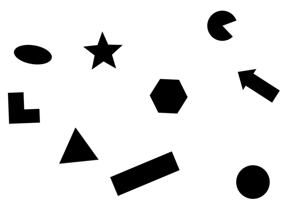
4) The RANSAC search and the scoring mechanism work as expected.

Items 1, 2 and 3 are verified by inspecting the logs of the algorithm and by visualising the vectors, hull and FoE per estimation. To check whether the RANSAC search performs as expected, the log output is studied using limiting the incoming vectors to trivially perform the estimation by hand and check whether the outcome is similar. Here it is found that, as required, vectors are rejected when they do not contribute to a smaller hull and the highest-scoring hull is kept. Equal-scoring hulls are selected when more vectors are contributing to the hull. When no FoE is found, no signal containing the FoE is published from the ROS node, so no other programs or algorithms will receive the information.

V. Results

This section presents the results of the algorithm. The first subsection elaborates upon the estimation accuracy using eSim. Next, the algorithm will be compared to another algorithm to evaluate the difference in performance. Afterward, estimation accuracy and runtime performance onboard a drone will be discussed, for both FAITH and the alternative algorithm.

Unless stated otherwise, the program runs with parameters described in Table II. These parameters are chosen as they give good results; however other combinations are also shown to work. The number of iterations is the amount of iterations n performed in the RANSAC search. Fewer iterations improve computational performance, but accuracy might decrease as there is a lower chance of finding an optimal solution. The minimal amount of vectors required is relevant for situations with little flow, at the risk of higher latencies. The FoE buffer length denotes the length of the moving average buffer of the FoE and the frequency is the desired frequency of the estimation. The frequency sets a minimum period after which a new estimation is started; the estimation will therefore run at max on the set frequency.


(a) Random shapes-filled plane used in planar renderer

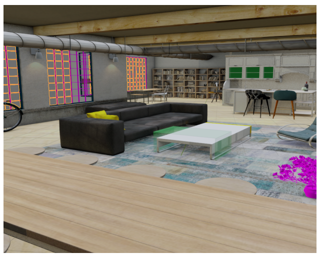

(b) Living room environment used in 3D renderer

Figure 6: Illustrations of testing environments

A. Simulator results

Performance is tested using a series of 3D trajectories, including non-corresponding rotations, both in a planar renderer, portraying a series of figures shown in Figure 6a, as well as in a 3D model of a living room, displayed in Figure 6b. The trajectories in the planar renderer take 5 seconds in $1ms^{-1}$ motion towards the plane, starting at $6m$. The living room environment allows for a $10s$ flight in which $20m$ is flown. Both trajectories are supplemented with lateral and vertical motions and rotations around all axes. Reference trajectories can be found in section B.

Table III demonstrates the results for five different trajectories in both renderers. In the planar renderer, the error is generally smaller and of similar value in both x and y direction. In the 3D renderer, the y component performs significantly better, with a lower average error and a lower standard deviation. This is believed to be because there is a larger amount of horizontal edges in the room environment, causing the vertical flow to be dominant compared to the horizontal flow. This is confirmed when looking at the distribution of $\arg(\vec{\mathbf{v}})$, shown in Figure 7. A lack of horizontal flow makes it more difficult to estimate the x-coordinate of the FoE, as the FoE area will be larger in the horizontal direction. In the planar renderer, the objects are evenly distributed over the plane, and the image is repeated at its edges.

In Figure 8, the results of a representative run in the 3D environment are presented over time. One can see that even though there is a significant amount of noise, the FoE is estimated well over time, especially in the vertical direction. The error distribution is displayed in Figure 10. Here it is visible that the error is close to normally distributed. Other trajectories show similar results.

Lengthening the averaging period of the FoE estimation will lead to more stable results at the cost of a slower response. In Table IV, one can observe that the results are not unanimously better when using 50 estimations in the averaging process. In the 3D environment, the results generally improve with higher averaging. Using the planar renderer, overall results are worse. This difference can be attributed to the introduced delay introducing a more significant error than the less stable results did. In the 3D renderer, the stable predictions outweigh the delay and overall provide a better estimation. To support this hypothesis, a visualization of the results in the 3D environment using 50 estimations as average is shown in Figure 11.

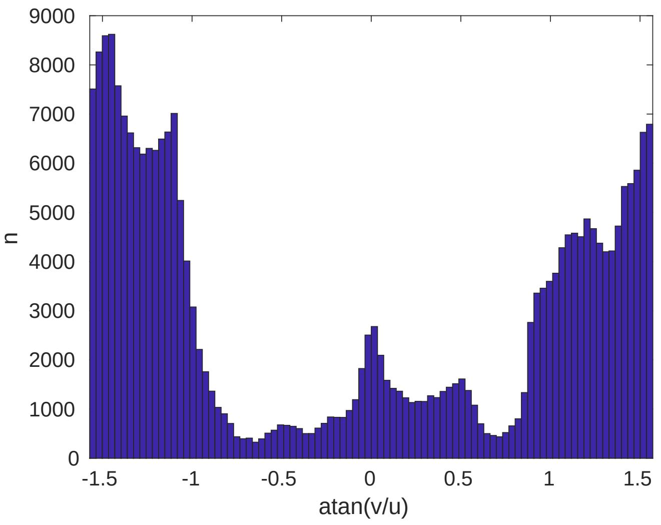


Figure 7: Distribution of flow vector argument in a flight through the living room environment in the 3D renderer. One can observe a significant representation of (near-)vertical vectors. This is considered the reason for the increased performance in the vertical axis of the FoE estimation.

B. Vector intersections

To put the performance into perspective, this section compares the algorithm to an algorithm based on the Vector Intersections method as described in [23]. First, the algorithm is explained and afterward, the performance is briefly elaborated upon.

The Vector Intersections method is designed to filter out outliers in an optical flow estimation. A RANSAC scheme finds the intersection between two vectors and subsequently checks how many vectors share that intersection, correcting for pixel discretization. In the end, the intersection with the most inliers is kept and vectors that do not share this FoE are classified as outliers.

It is important to note that the outlier classification is not applicable to normal flow. Normal flow vectors do not share a single FoE; subsequently, outlier classification does not apply to this. The RANSAC scheme still helps find the FoE with the most inliers, loosening the discretization constraint and counting all vectors that diverge from the estimated FoE as inliers.

Evaluating the accuracy, the results in Table III are found. Comparing this to the FAITH algorithm, one can see that the method is less accurate than FAITH, having both a higher error and a more significant standard deviation of this error. Again, the method is more accurate in the y-direction, performing consistently better.

C. Onboard results

Computational performance is tested on an UPboard. This small processing board consists of an Inter Atom processor with four $1.9GHz$ cores paired with 4 GB of memory. The UPboard is low-weight and has limited power use, making it suitable for use on a vehicle. The camera has a FOV of $44.2°$ horizontally and $33.8°$ vertically. The flying environment is displayed in Figure 12.

For the onboard experiments, the amount of iterations is changed to $n = 50$ to allow for higher rates. Calculation times of $\pm 2 \cdot 10^{-3}$ are found. Over nine runs, the average calculation times and standard deviations are found in Table VI, which are found by finding the time between the start of the estimation procedure and the final estimation. The estimations are published from the ROS node at $\pm 140Hz$. A discrepancy is found between the calculation time and the publishing rate, as $1/140 > 2 \cdot 10^{-3}$, which is attributed to the planefitting algorithm limiting the frequency at which information is gathered.

Looking at Figure 13, a number of observations can be made. First, the ground truth generated by Optitrack measurements is especially in the beginning noisy. However, one can observe that the general trend in the FoE is captured, especially in x direction. In y direction, the accuracy is less. This is expected to be due to that mostly vertical edges being present, which do not contribute to accuracy in the y direction. The accuracy improves in time due to a highly textured object at the end of the trajectory, which creates clearer edges when one gets near.

Table V demonstrates the accuracy of the algorithm in the CyberZoo (using the standard settings). One can clearly observe that FAITH performs worse than the simulator, as both accuracy and precision are worse.

Large errors can be found in the first and last seconds of flight, as seen in Figure 13, with the FoE being far outside of the FOV as one flies near vertically. This flight part is left

Environment	**Trajectory**	$\mu_{\lvert\epsilon_\mathbf{x}\rvert}$	$\mu_{\lvert\epsilon_\mathbf{x}\rvert}$	$\sigma_{\epsilon_\mathbf{x}}$	$\sigma_{\epsilon_\mathbf{x}}$	$\mu_{\lvert\epsilon_\mathbf{y}\rvert}$	$\mu_{\lvert\epsilon_\mathbf{y}\rvert}$	$\sigma_{\epsilon_\mathbf{y}}$	$\sigma_{\epsilon_\mathbf{y}}$
		FAITH	Vec.Inters.	FAITH	Vec.Inters.	FAITH	Vec.Inters.	FAITH	Vec.Inters.
Planar renderer	1	**1.9684**	3.0876	**3.6331**	4.9604	**1.6402**	2.5879	**2.4043**	3.5581
	2	**2.6653**	3.4137	**4.0175**	4.5675	**2.3276**	3.0337	**3.2119**	4.0301
	3	**2.6189**	3.7957	**5.1956**	6.3362	**2.1169**	2.5017	**2.5908**	3.1306
	4	**1.9106**	3.314	**2.4158**	4.1809	**1.9933**	3.5641	**2.6812**	4.128
	5	**3.7565**	4.9106	**6.2908**	7.8378	**2.5026**	4.0701	**3.47**	5.3207
3D renderer	1	**5.6484**	5.697	**7.2668**	7.6779	**2.3731**	2.3806	**2.9407**	3.0421
	2	**4.5979**	7.7479	**5.6583**	8.8797	**2.6898**	4.0232	**3.4849**	4.9641
	3	**6.2671**	8.8237	**8.2254**	11.4621	3.9173	**3.7585**	**5.0626**	5.0932
	4	**4.5301**	10.2845	**6.0509**	12.8287	**2.4734**	3.9137	**3.56**	5.5131
	5	**4.5314**	6.3417	**5.9836**	7.6693	**3.2533**	3.9988	**4.3897**	5.1596

Table III: Results of FoE estimation using FAITH and the adaptation of the Vector Intersections [23] method (see subsection V-B). The better value for each statistic is written in bold. A low mean absolute error implies that the predictions center around the actual value and a low standard deviation means that the spread is low.

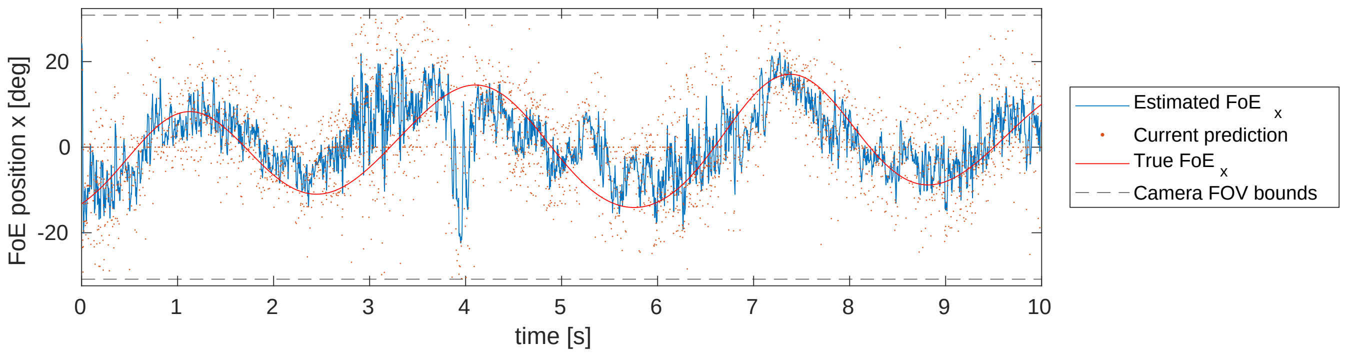


Figure 8: FoE estimation over time in the 3D renderer, in x and y positions. The y position performs better, exhibiting less noise and closely tracking the ground-truth FoE. Also, compared to Figure 9, a larger amount of noise is visible.

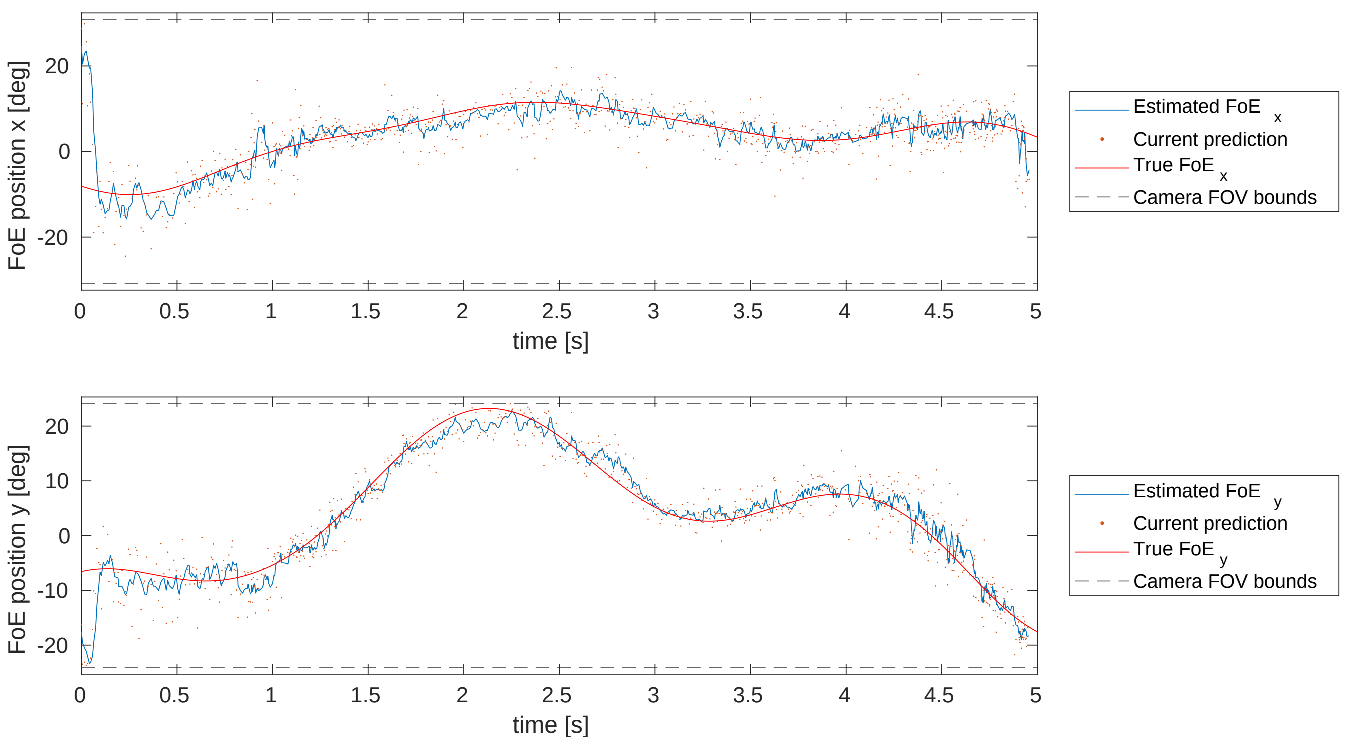


Figure 9: FoE estimation over time in the planar renderer, in x and y positions. After a bad initial estimate, assumed to be due to many events upon initialization, the FoE in both directions is estimated well. At $t = 2$, the true FoE is on the edge of the camera FOV in y direction, which results in a slight loss of accuracy.

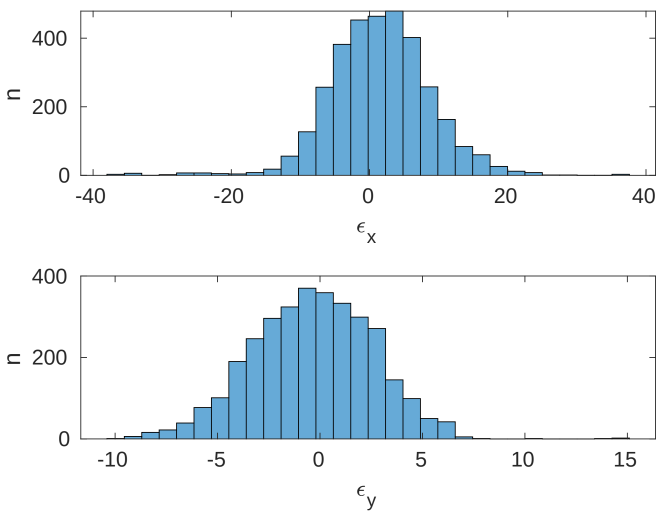


Figure 10: Error distribution of the FoE estimation shown in Figure 8. Both coordinates show approximately normally distributed errors. The x coordinate has a wider spread, as expected from the results in Table III

out of the average error as it gives a skewed representation of in-flight results. In section VII, recommendations are made to improve performance in such flight conditions.

VI. Discussion

As seen in the previous section, FAITH is able to predict the FoE with high accuracy. A significant difference is found between the performance of the planar renderer and the 3D renderer. This difference is attributed to the fact that the planar renderer has a more even spread of events with various orientations on which flow can be projected. In contrast, the living room render has more vertical and horizontal lines and flow is less evenly spread over the pixel area.

Environment	**Trajectory**	$\mu_{\|\epsilon_x\|}$	σ_{ϵ_x}	$\mu_{\|\epsilon_y\|}$	σ_{ϵ_y}
Planar renderer	1	2.3601	4.8934	2.9784	3.7422
	2	3.0022	3.9102	2.1725	3.1902
	3	3.7593	6.483	3.4948	4.6839
	4	2.9304	3.3568	2.6836	3.508
	5	5.7024	8.7231	2.3831	3.2925
3D renderer	1	4.6888	5.8278	1.7815	2.2127
	2	3.9185	4.5297	1.6923	2.359
	3	4.7751	6.3981	3.0016	3.9568
	4	4.5352	5.6992	2.0657	3.0745
	5	3.5862	4.4874	3.3329	4.3428

Table IV: Results of the FAITH algorithm in the simulator using 50 measurements as averaging time

Trajectory	$\mu_{\|\epsilon_x\|}$	σ_{ϵ_x}	$\mu_{\|\epsilon_y\|}$	σ_{ϵ_y}
1	5.1326	6.4252	9.2959	11.734
2	3.9573	5.3139	8.1744	9.5683
3	6.0057	5.9839	15.1602	19.1843
4	4.6097	5.4128	12.1011	13.0314
5	9.5908	11.7604	9.7102	11.4615
6	6.0898	8.2949	10.0329	12.4999
7	5.3549	9.7411	7.9964	10.6406
8	6.6294	6.4786	7.602	9.4992
9	12.4487	20.4562	14.191	23.8649

Table V: Results in the CyberZoo with standard settings. One can observe a large spread and a large absolute error.

Run	$\mu(T) \cdot 10^{-3}$	$\sigma^2(T) \cdot 10^{-4}$
1	2.0992	0.74945
2	2.0506	0.78337
3	2.1443	0.78013
4	2.0687	0.73057
5	2.1547	0.81207
6	2.0320	0.63452
7	2.1201	0.71303
8	2.3095	1.2308
9	2.1495	1.0607

Table VI: Calculation times for the two algorithms.

When only 5 measurements of averaging is applied, the residual in both renderers is in the order of ± 5 degrees. The amount of averaging should depend on the attainable frequency and the delay that is allowed; at high estimation frequencies, it is acceptable to use larger amounts of data points as the time delay will not cause more significant errors. Using too many data points at relatively low frequencies leads to fast behavior being filtered and significant time delays.

The number of iterations in the RANSAC search N has not been adapted in the research. There is currently no incentive to increase the computational performance, considering performance at this moment is dictated by the optic flow estimation. Furthermore, considering the number of unique iterations equals $n!$, with n being the number of vectors present per estimation, the amount of iterations quickly becomes insignificant, especially considering the requirement of at least 10 vectors per estimation.

In order to perform accurate estimations, it is essential to ensure that flow is evenly spread across the image. A limitation of this approach is that accuracy depends on the amount of flow close to the FoE. If flow is only present on the edges of the image plane, it is hard to estimate the FoE with high accuracy due to the estimated FoE area being extensive. Similarly, it was shown that the FoE estimation's accuracy depends on the flow's direction.

A comparison to the adapted Vector Intersections method indicates that our method is more suitable for FoE estimation than methods originally devised for regular optical flow. The use of half-planes recursively to accurately estimate the FoE works better than the Vector Intersections method as the flow does not share an FoE. This is however expected behavior and does not degrade the Vector Intersections method as a whole. However, it does affirm the need for new algorithms when dealing with event-based optic flow.

The onboard performance highlights the shortcomings regarding the flight envelope. Only headings within the FOV can be accurately estimated, which can be a limitation. Also, estimation accuracy in the y-axis is significantly lower compared to the x-axis. This is partly attributed to lower resolution on the y-axis, which limits the accuracy. The periodic behavior of the error is thought to be due to the pitching motion that stems from the forward velocity control; this could however not be verified.

Computational performance of $\pm 140 Hz$ suffices for online obstacle avoidance. A more lightweight planefitting process

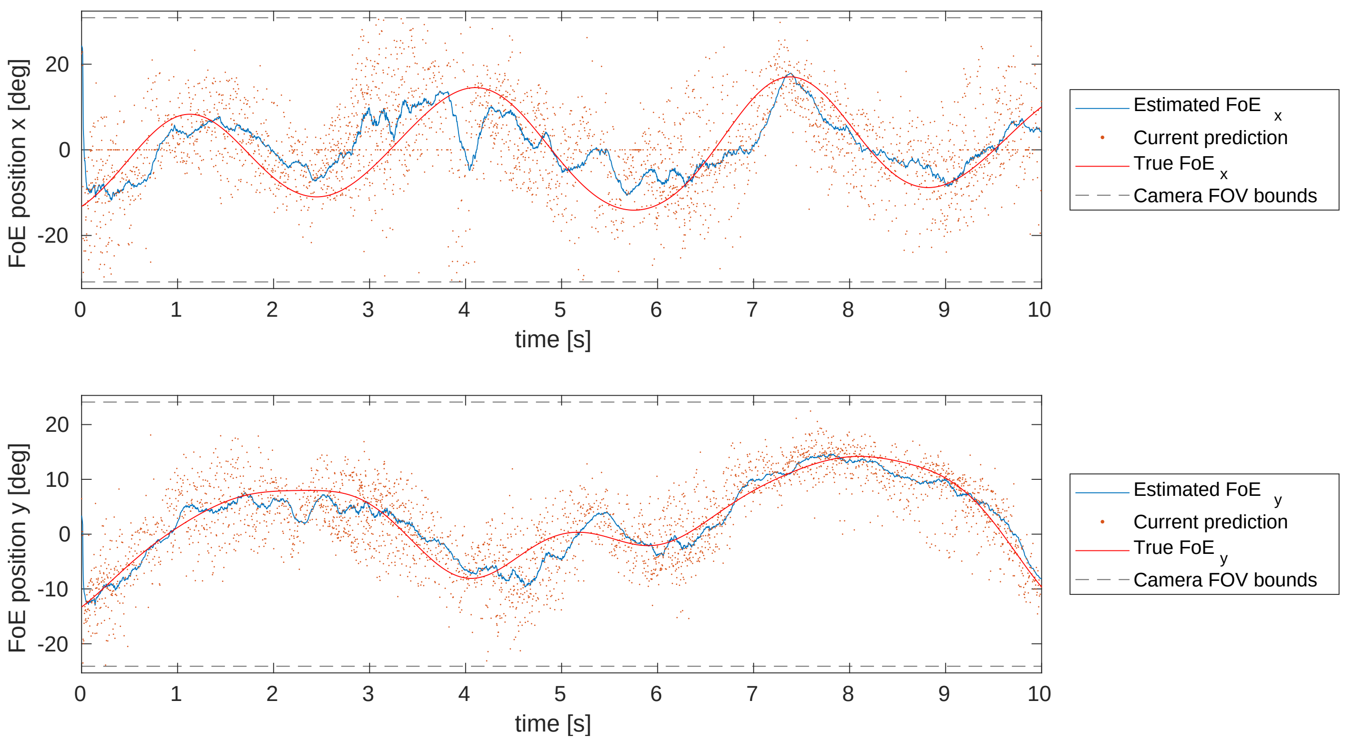


Figure 11: FoE estimation over time, in x and y positions, with increased averaging. One can see a decrease in noise, especially in the x-direction, compared to Figure 8. In the y-direction, one can observe a slight delay, especially near $t = 9$ seconds.

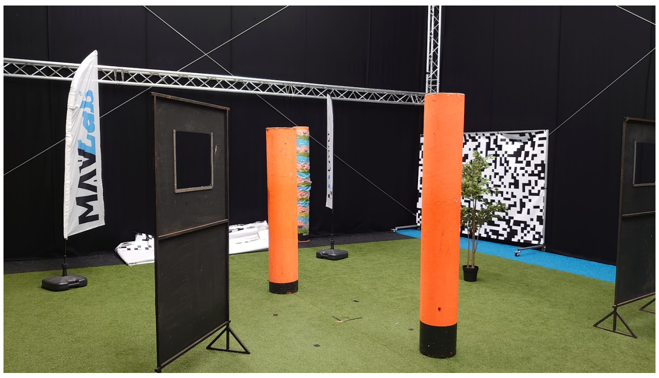


Figure 12: Image of the test setup in the CyberZoo. Notice that most objects consist of mostly vertical edges and the background has no texture.

may attain higher rates.

VII. Recommendations

This section will elaborate on the recommendations to improve this work or to use this work in other applications.

One of the main points of attention should be the noise removal from the estimations. One of the methods to achieve this is to treat the incoming flow differently by, for example, keeping a short temporal history in memory. Another method would be to employ a different filtering method than the simple moving average. A median filter, low-pass filter or an exponential weighted average might provide a different trade-off between latency and accuracy. Also, implementing a time buffer in the estimations could be more robust toward performance fluctuations.

Where in a controlled environment, a variety of textures can be guaranteed, environments such as a hallway, forest or street might prove not ideal. This behavior should be investigated and, if necessary, improved. A suggestion for applications in environments with low flow amounts might be to include the flow magnitude in the estimation.

The main problems to overcome regarding onboard performance are the unstable predictions and the small FOV. Getting more stable estimations, especially vertically, and the ability to cover a broader range of flight paths, increases performance significantly. Another recommendation regarding the onboard processing is to look into improvements of or alternatives to the planefitting algorithm, as that currently limits performance.

Lastly, as shown in the work of [1], FoE estimation can be used in obstacle avoidance. This research has focused on improving and validating the estimation itself, leaving obstacle avoidance out of scope. Object avoidance should be rewritten and re-tested considering the improved computational performance and derotation capabilities. Faster calculations increase speed and precision, allowing unmanned aerial vehicles to move through cluttered environments faster than previously possible autonomously.

VIII. Conclusion

It is shown that FoE estimation using event-based optic flow can be a reliable source of heading information. It is shown that using the FAITH algorithm, one can determine the FoE with high precision and accuracy. The FoE estimation is

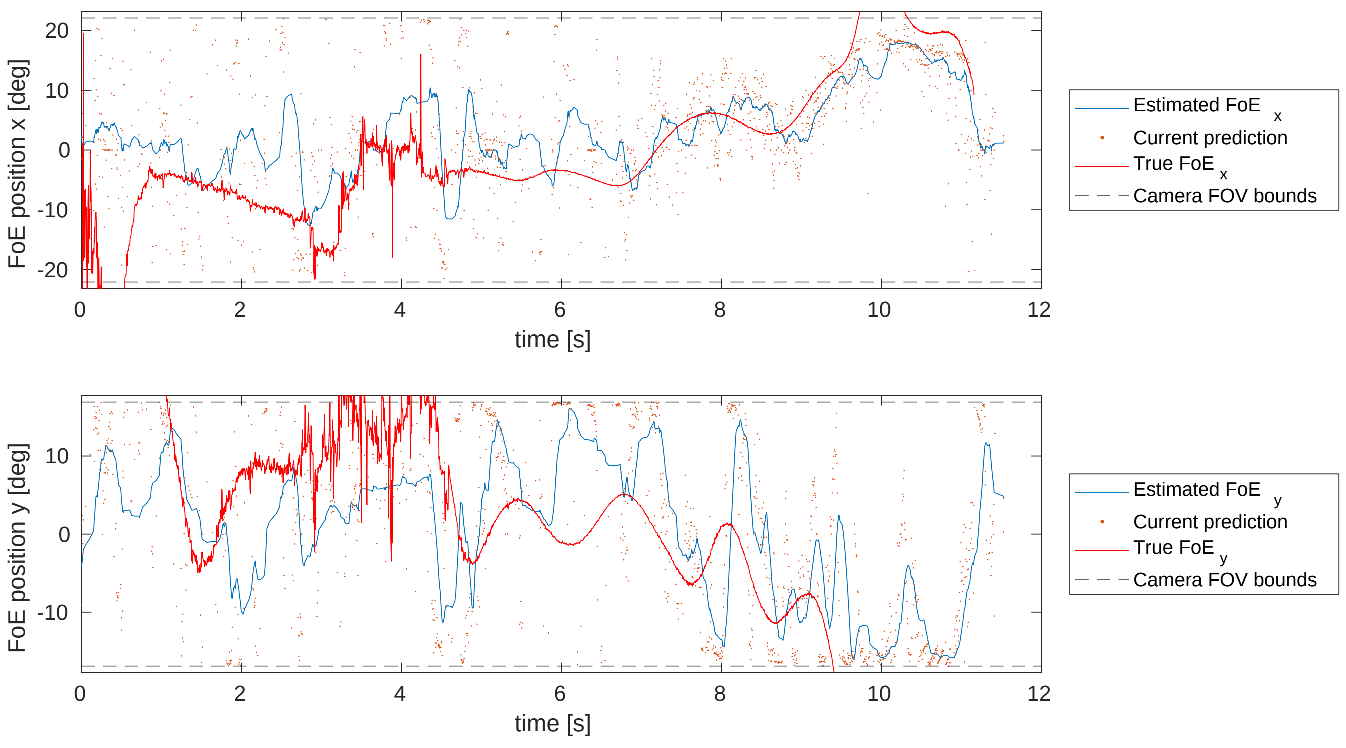


Figure 13: FoE estimation in the Cyberzoo using the UPSquared board on a drone.

however very dependent on the environment, which is required to have textures at different angles spread throughout the frame in order to provide accurate estimations. If textures are present in only a part of the frame, this may cause the estimation to include a bias. It is recommended to explore workarounds to increase robustness.

In flight, the computational load of the algorithm is low, attaining frequencies of $140Hz$. In simulation, higher accuracy is attained compared to a representable alternative due to its simple functions and excellent compatibility with normal flow. It is recommended to explore the limits of computational performance with a faster optic flow estimation algorithm and test different hardware, to further improve accuracy in flight.

Appendix A
Proof of Normal Flow Derotation

Set a flow vector $\vec{\mathbf{v}}$, rising from a 2D surface which indicated as a line on a 2D plane. Introduce a local axis system, of which the origin coincides with the vector's origin and the x-axis coincides with the 2D plane. This leads to the y-axis always being normal to the surface. The optic flow vector $\vec{\mathbf{v}}$ can be split into translational and rotational components [19], according to:

$$\vec{\mathbf{v}} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v_{x,T} \\ v_{y,T} \end{bmatrix} + \begin{bmatrix} v_{x,R} \\ v_{y,R} \end{bmatrix} \tag{8}$$

where the subscripts T and R denote the translational and rotational components of the flow and x and y denote the components in the local axis system, respectively. In the case of 'regular' optic flow which is not normal to the surface, the rotational component can be subtracted from the total, leaving

$$\begin{bmatrix} v_{x,derot} \\ v_{y,derot} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} - \begin{bmatrix} v_{x,R} \\ v_{y,R} \end{bmatrix} = \begin{bmatrix} v_{x,T} \\ v_{y,T} \end{bmatrix} \tag{9}$$

In the case of normal flow, the component of the flow perpendicular to the surface is estimated. Flow perpendicular to the surface coincides with the y-axis, due to both of them being orthogonal to the surface. Therefore, the normal part of the optic flow can be described as

$$\vec{\mathbf{v}_n} = \begin{bmatrix} 0 \\ v_{n,y} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \vec{\mathbf{v}} \tag{10}$$

where n denotes a normal flow vector. This can be expanded to

$$\vec{\mathbf{v}_n} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \left(\begin{bmatrix} v_{x,T} \\ v_{y,T} \end{bmatrix} + \begin{bmatrix} v_{x,R} \\ v_{y,R} \end{bmatrix} \right) \tag{11}$$

implying that the normal flow too is the sum of it's respective translational and rotational components:

$$\vec{\mathbf{v}_n} = \vec{\mathbf{v}}_{n,T} + \vec{\mathbf{v}}_{n,R} \tag{12}$$

If the derotated normal flow vector is then written as

$$\vec{\mathbf{v}}_{n,derot} = \vec{\mathbf{v}}_{n,T} = \vec{\mathbf{v}}_n - \vec{\mathbf{v}}_{n,R} \tag{13}$$

this can be expressed as

$$\vec{\mathbf{v}}_{n,derot} = \vec{\mathbf{v}}_n - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \vec{\mathbf{v}}_R \tag{14}$$

If we let α be the angle between the original flow vector and the local y-axis, this can be written as

$$\vec{\mathbf{v}}_{n,derot} = \vec{\mathbf{v}}_n - \cos\alpha \cdot \vec{\mathbf{v}}_R \tag{15}$$

which equals the normal flow minus the component of the rotational flow along this axis. In Figure 14, one can observe how the different components interact, visually. From this figure it becomes clear that the different components all correspond to their respective components along the normal.

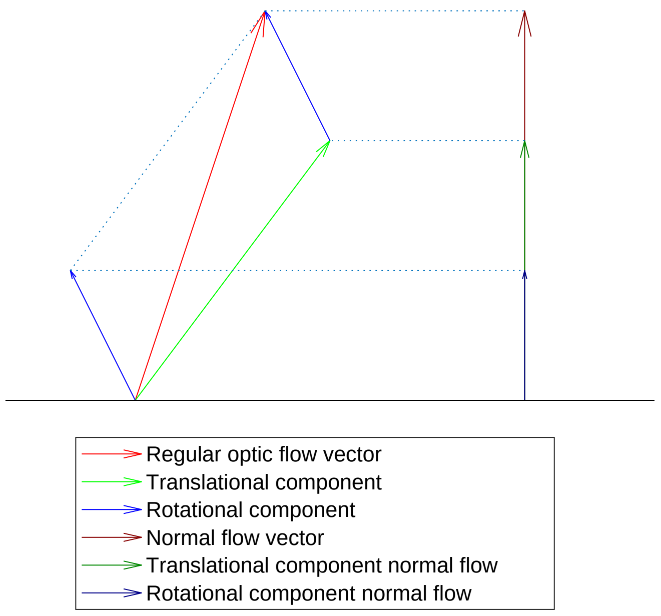

Regular optic flow vector
Translational component
Rotational component
Normal flow vector
Translational component normal flow
Rotational component normal flow

Figure 14: Visualization of both normal flow and regular optic flow, separated in its components. One can observe that the rotational and translational components correspond in the normal direction.

Appendix B
Example trajectories

This section provides examples (Figure 15, Figure 16, Figure 17) of the trajectories used in the different renderers and the CyberZoo. Other trajectories are the same in forward distance, however excursions to the side differ. Please note that for clarity all trajectories have been depicted with z as the forward direction, however in reality different settings have different axis systems.

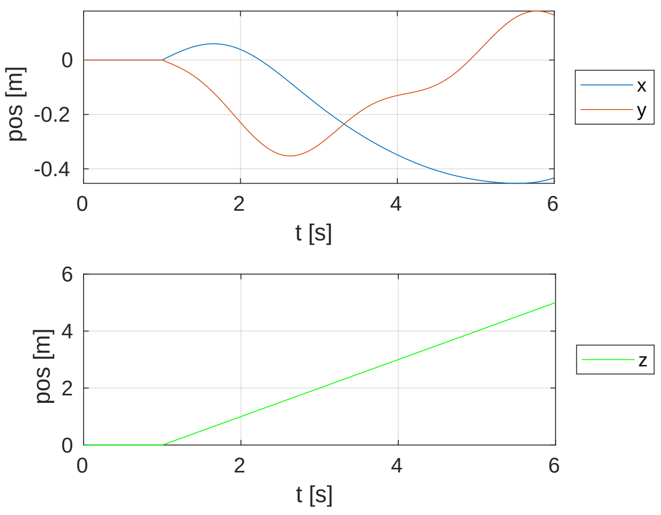


Figure 15: Example of a trajectory in the 2D planar renderer with the shapes background. The plane with shapes repeats itself laterally.

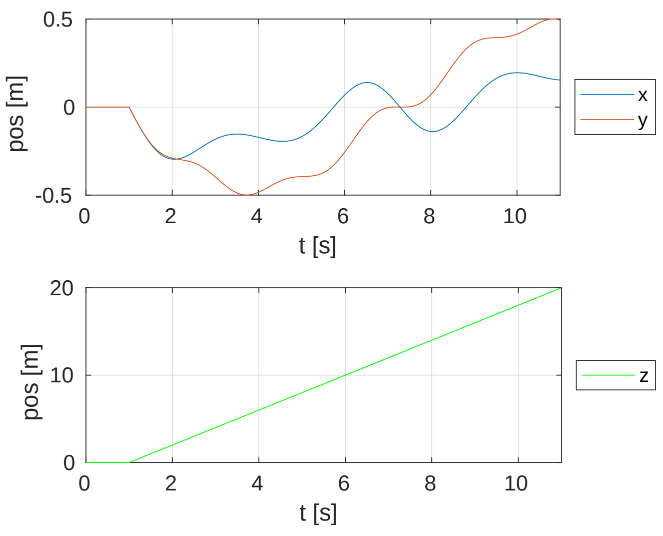


Figure 16: Example of a trajectory in the 3D renderer in the living room environment. During the trajectory, several furniture items are flown by.

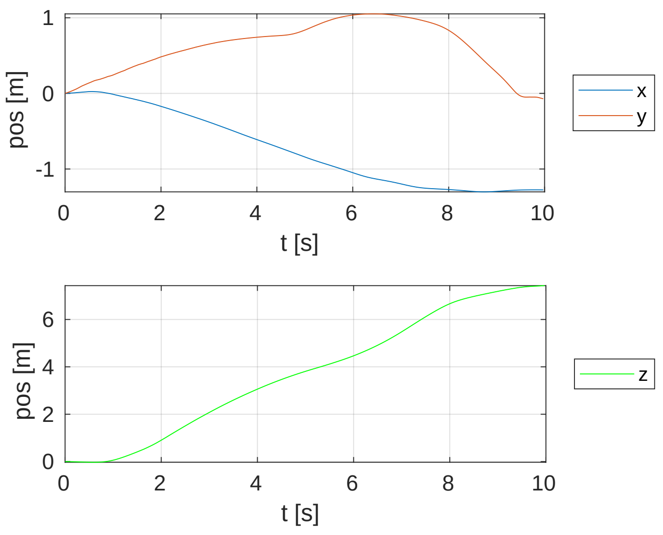


Figure 17: Example of a trajectory in the CyberZoo. In the CyberZoo, some poles and other obstacles were placed to provide more texture.

References

[1] R. Dinaux, N. Wessendorp, J. Dupeyroux, and G. de Croon, "FAITH: Fast iterative half-plane focus of expansion estimation using event-based optic flow," Feb. 2021. [Online]. Available: http://arxiv.org/abs/2102.12823.

[2] N. Zhao, Z. Wang, C. Cai, *et al.*, "Bioinspired Materials: from Low to High Dimensional Structure," *Advanced Materials*, vol. 26, no. 41, pp. 6994–7017, Nov. 2014, ISSN: 1521-4095. DOI: 10.1002/ADMA.201401718. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/adma.201401718%20https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201401718%20https://onlinelibrary.wiley.com/doi/10.1002/adma.201401718.

[3] M. Davies, N. Srinivasa, T. H. Lin, *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018, ISSN: 02721732. DOI: 10.1109/MM.2018.112130359.

[4] K. Fukushima, Y. Yamaguchi, M. Yasuda, and S. Nagata, "An Electronic Model of the Retina," *Proceedings of the IEEE*, vol. 58, no. 12, pp. 1950–1951, 1970, ISSN: 15582256. DOI: 10.1109/PROC.1970.8066.

[5] M. A. Mahowald and C. Mead, "The silicon retina," *Scientific American*, vol. 264, no. 5, pp. 76–82, 1991, ISSN: 0036-8733. DOI: 10.1038/SCIENTIFICAMERICAN0591-76. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/2052936/.

[6] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008, ISSN: 00189200. DOI: 10.1109/JSSC.2007.914337.

[7] C. Brandli, R. Berner, M. Yang, S. C. Liu, and T. Delbruck, "A 240 × 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014, ISSN: 00189200. DOI: 10.1109/JSSC.2014.2342715.

[8] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011, ISSN: 00189200. DOI: 10.1109/JSSC.2010.2085952.

[9] B. Son, Y. Suh, S. Kim, *et al.*, "A 640×480 dynamic vision sensor with a 9μm pixel and 300Meps address-event representation," *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, vol. 60, pp. 66–67, Mar. 2017, ISSN: 01936530. DOI: 10.1109/ISSCC.2017.7870263.

[10] Y. Suh, S. Choi, M. Ito, *et al.*, "A 1280x960 dynamic vision sensor with a 4.95-μm pixel pitch and motion artifact minimization," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2020-October, 2020, ISSN: 02714310. DOI: 10.1109/ISCAS45731.2020.9180436.

[11] E. M. Drakakis, A. Civit, J. Sitte, H. Neumann, T. Brosch, and S. Tschechne, "On event-based optical flow detection," *Frontiers in Neuroscience — www.frontiersin.org*, vol. 9, p. 137, 2015. DOI: 10.3389/fnins.2015.00137. [Online]. Available: www.frontiersin.org.

[12] J. C. Tapson, F. Stefanini, S. Hoi, T. Delbruck, and B. Rueckauer, "Evaluation of Event-Based Algorithms for Optical Flow with Ground-Truth from Inertial Measurement Sensor," *Front. Neurosci*, vol. 10, p. 176, 2016. DOI: 10.3389/fnins.2016.00176. [Online]. Available: www.frontiersin.org.

[13] R. Benosman, S. H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural Networks*, vol. 27, pp. 32–37, Mar. 2012, ISSN: 08936080. DOI: 10.1016/J.NEUNET.2011.11.001.

[14] M. T. Aung, R. Teo, and G. Orchard, "Event-based Plane-fitting Optical Flow for Dynamic Vision Sensors in FPGA," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2018-May, Apr. 2018, ISSN: 02714310. DOI: 10.1109/ISCAS.2018.8351588.

[15] G. Convertino, A. Branca, and A. Distante, "Focus of expansion estimation with a neural network," *IEEE International Conference on Neural Networks - Conference Proceedings*, vol. 3, pp. 1693–1697, 1996. DOI: 10.1109/ICNN.1996.549155.

[16] D. Sazbon, H. Rotstein, and E. Rivlin, "Finding the focus of expansion and estimating range using optical flow images and a matched filter," DOI: 10.1007/s00138-004-0152-7.

[17] N. Van Der Stap, R. Reilink, S. Misra, I. A. Broeders, and F. Van Der Heijden, "The use of the focus of expansion for automated steering of flexible endoscopes," *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 13–18, 2012, ISSN: 21551774. DOI: 10.1109/BIOROB.2012.6290804.

[18] S. Stabinger, A. Rodríguez-Sánchez, J. Piater, and S. Stabinger@uibk, "Monocular obstacle avoidance for blind people using probabilistic focus of expansion estimation; Monocular obstacle avoidance for blind people using probabilistic focus of expansion estimation," Tech. Rep., 2016. DOI: 10.1109/WACV.2016.7477608.

[19] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," in *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 208, The Royal Society, London, Jul. 1980, pp. 385–397. DOI: 10.1098/RSPB.1980.0057. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspb.1980.0057.

[20] X. Clady, C. Clercq, S. H. Ieng, *et al.*, "Asynchronous visual event-based time-to-contact," *Frontiers in Neuroscience*, 2014, ISSN: 1662453X. DOI: 10.3389/fnins.2014.00009.

[21] B. J. Pijnacker Hordijk, K. Y. Scheper, and G. C. de Croon, "Vertical landing for micro air vehicles using event-based optical flow," *Journal of Field Robotics*, vol. 35, no. 1, pp. 69–90, Jan. 2018, ISSN: 15564967. DOI: 10.1002/ROB.21764.

[22] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an Open Event Camera Simulator," 2018. [Online]. Available: https://www.blender.org/.

[23] M. Buczko and V. Willert, "Monocular Outlier Detection for Visual Odometry," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 739–745, Jul. 2017. DOI: 10.1109/IVS.2017.7995805.

II

Literature study

Sense and avoid in harsh environments

Literature Study

by

S.J.F. Knoops

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number:	4379756	
Thesis committee:	dr.ir. J.J.G. Dupeyroux	TU Delft, supervisor
	Prof.dr.ir. G.H.C.E. de Croon,	TU Delft

List of Figures

List of Tables

Executive summary

Recently, a lot of new developments have been taking place in the field of autonomous vehicles. Autonomous vehicles are more common these days, even being allowed on public roads. Also on small drones, autonomy is within reach due to the decrease in size and power use of components. One of the possible applications is to perform Search&Rescue missions, sending small drones to search for e.g. survivors of a building fire. This scenario however poses a problem, as navigation algorithms generally rely on clear, well-lit areas, which is often not the case. In order to be able to navigate in such harsh environments, a navigation solution using a radar and event-based camera will be developed.

There are many different ways to detect obstacles. Sensors can be divided into two main categories: vision-based and telematic methods. Vision based methods depend on the light that is incoming. A regular, frame-based camera is most common, however suffers from needing quite specific illumination circumstances. An event-camera is a novel type of camera with interesting properties such as high frequency, high dynamic range and being very energy efficient whilst performing better in low-light circumstances. An RGB-D camera is a frame-based camera extended with a system that emits infrared light, the reflection of which can be studied to perceive depth information.

Among telematic methods are LiDAR, radar and sonar. LiDAR works similar to RGB-D, however sending out laser beams instead of patterns, being more accurate but only able to register one point at a time. Radar sends out radio waves, penetrating smoke or dust and not requiring any illumination. A set of receivers is able to detect both bearing and distance to an obstruction. Sonar sends out audio waves that are reflected by objects. Sonar however is unable to determine bearing and performance rapidly degrades under influence of smoke and dust.

An event-camera and a radar together make for an accurate and robust solution that works well in both illuminated and pitch black scenario's. The radar is able to penetrate smoke or dust while in clear scenario's, the event-camera provides very accurate and fast information about the environment.

After information is received, some form of coupling is required in order to combine the information. This can be done either fusion-based or optimisation based. Fusion based is less computationally expensive and examples are the Kalman Filter or Particle Filter. Optimisation based generally has higher accuracy, however at cost of computational power required to minimise a cost function.

Sensor fusion approaches are tightly bound to the context of their development. No general solutions exist and solutions are highly tailored to their purpose and hardware.

Before flight is enabled, an avoidance scheme needs to be implemented. Two main methods are discussed: the Artificial Potential Field and Velocity Obstacles. The Artificial Potential Field generates virtual potential fields around the goal and obstacles. Summing these fields and taking the gradient finds a route around obstacles towards the objective. Velocity Obstacles defines no-fly zones in terms of velocity and bearing vectors by analysing ego-motion and motion of obstacles. A velocity vector should be chosen outside the set that Velocity Obstacles generates in order to avoid collision. The fact that Velocity Obstacles works with velocities and also has the capability to handle dynamic obstacles, makes it favourite for this research topic.

Next step in this research will be development of the system, implement all aforementioned elements. The research question

> ”How can a radar system and and eventcamera system be fused adequately to run on a small autonomous drone in order to achieve a robust object avoidance solution?”

will be answered. A drone able to autonomously avoid obstacles will be the goal of the thesis.

Contents

1 Introduction

The field of unmanned vehicles is one that is rapidly evolving. Whereas fully autonomous vehicles were perceived as a danger by the public not too long ago, nowadays a (supervised) self-driving car is generally accepted and can be found on the road in most countries. This shift comes hand in hand with technologies that push the limit on autonomy every day. New and improved types of sensors, faster and more efficient processors and bio-inspired algorithms have, among other factors, a large impact on the capabilities of these vehicles.

One of the main consequences of the development of autonomous technology is the fact that it allows humans to access places that where otherwise impossible, while mitigating human error or faulty communications. A swarm of small robots is able to quickly explore large areas without requiring an operator for each drone. Taking the human (partly) out of the loop may also help to prevent e.g. fatigue or error which could lead to reduced accidents. Whereas autonomous drones are not yet perfect and their applications currently are limited, they do show great potential.

Among the earlier mentioned new and improved sensors are radar chips and neuromorphic cameras. Radar systems, commonly known by the grand public as large apparatus that span several meters, has been around for decades but recently shrunk in size to chips only weighing a few grams. This technology can now also be applied in smaller applications, sensing an indoor environment instead of an aircraft and mounted on a MAV instead of a battleship.

Neuromorphic cameras are cameras that more closely mimic the biological eye, allowing for very fast and accurate gathering of information. First developed in the '80s, these sensors behave completely different than regular cameras and outperform the latter on several fields.

This research is done in the context of Search&Rescue (S&R) missions. In Search&Rescue operations often view is worsened as for example smoke hangs in buildings and/or electricity is down. This proves for a difficult environment as human eyes are not able see through smoke or even in the dark. Also, S&R operations tend to take place in dangerous environments where it may not always be desirable to blindly send in more people. In those cases, where humans either perform badly or might risk their lives, autonomous drones might be the answer.

Where a regular drone would already answer some of those problems with a solution, an autonomous drone has as advantage that it has the potential to significantly reduce manpower required. Where an operator can only fly (or drive, for that matter) one or maybe a few drones at the time, coordinating a larger workforce of autonomous drones is relatively simple. Also, an autonomous drone does not require to always be in contact with the operator and thus can more easily penetrate dense areas.

A major difficulty of an autonomous system however, perhaps especially in the case of flying robots due to the large impact a collision may have, is navigation. Autopilots have been quite thoroughly researched and developed and are usually not the issue. Sensing and avoiding obstacles however is a major challenge still, as the detection and tracking of objects depends on a lot of (noisy) measurements and requires quite some computational power.

The innovations listed above and the potential of autonomous drones being able to reach where humans can not, makes it interesting for S&R operations. However, the problems that navigation can bring yet need to be solved in order to be able to rely on such systems for S&R operations.

This research will address the difficulties as described in the above. A sense-and-avoid system will be designed that is robust regardless the illumination circumstances.

The study builds on research done previously by two peers, Nikhil Wessendorp and Raoul Dinaux. They have investigated the application of a radar and a neuromorphic camera on a small UAV. Both systems have pros and cons, but this research will show that ideally, the two systems would work in parallel and complement each other. As will later be shown, this has the potential to provide a solution to navigation problems in a great variety of circumstances, including low illumination or obscured flight.

The end goal of this literature study is to provide the reader with arguments on the relevance of these sensors and their combined qualities. Different types of sensors will be described and traded off. Next to that, the methods that one can use to couple sensors are outlined. Concluding will be a review of avoidance strategies. This will provide a complete overview of a sense and avoid algorithm in harsh environments and serve as base for the thesis that follows this literature study.

First, in chapter 2, different sensing methods will be elaborated upon and some recent work using those sensors will be shortly presented. Also, the previous work will be further detailed. In the end of the chapter, the choice of a neuromorphic camera and radar will be validated at hand of the findings of the chapter. In chapter 3 we will discuss how these sensors can possible be fused by reviewing the principle of sensor fusion. Lastly, in chapter 4, the avoidance strategy will be discussed at hand of two main methods of avoiding objects. Lastly, in chapter 5, the literature study will be concluded and the next steps of this research will shortly be discussed.

2 Object detection

This chapter will outline the current state of object detection and odometry and describe the pros and cons of different approaches. First, different types of vision based methods will be discussed. Vision based methods are very common in odometry and avoidance applications. There are many different approaches, but all consist of three main choices one can make: the approach to extract key information, the camera type and the camera pose. In this section, three different camera types, a regular frame-based camera(section 2.1), a neuromorphic camera (section 2.2) and a RGB-D camera (section 2.3), will be discussed. Other types may include e.g. fish-eye lens cameras or thermal cameras. Next to that, the approach to extracting key information will be discussed.

After that, methods that rely on telematics will be discussed in section 2.4, section 2.5 and section 2.6. These methods do not rely on images as we might know them. As will be discussed, LiDAR has a lot in common with the RGB-D cameras discussed in section 2.3 and operates in the grey area between vision-based and telematic methods. After that, radar and sonar will be discussed, both of which do not operate with (near-) visible light but with radio and audio waves, respectfully.

Finally, section 2.7 will briefly summarise the most important findings and motivate the decision to continue the research of peers.

2.1. Frame-based camera

A frame-based camera is a camera how most of the population knows it. 'Frame-based' refers to the fact that the camera provides the information per frame. Initially, one may think that there is no other way, however, the next section will describe a different, not frame-based approach. Frame-based cameras have been around for decades, with the first mechanical cameras being developed in the early 1900's. Since the 1980's, digital video cameras are common and since the beginning of this century, almost everyone walks around with one or several video camera options in their pockets. The fact that they are used everyday is vital in the decrease in size, which makes it interesting for small drones as they can only carry limited mass.

2.1.1. Extracting information

In order to extract key information from an array of images, several methods exist. Firstly, the image itself can be compared to a previous one to estimate the optical flow. Also known as the direct method, pixel intensity in the frames is used to determine the change between them. This results in a 2D displacement vector per pixel that is used to estimate motion. This can be done either densely using all available pixels, using for example the Horn-Shunck algorithm [19], or sparsely, where only some pixels on the image are used, using for example Lucas-Kanade [28].

An illustration of optic flow estimation on a moving vehicle is seen in Figure 2.1.

Another approach is to perform feature detection. Here, instead of determining the flow at pixel level, interesting points are determined to subsequently be tracked. These points, often edges or corners, are easy to track as they contain well-noticeable changes in brightness due to their 3D-geometry. This makes this approach more robust, as the direct approach described above is sensitive to inconsistent

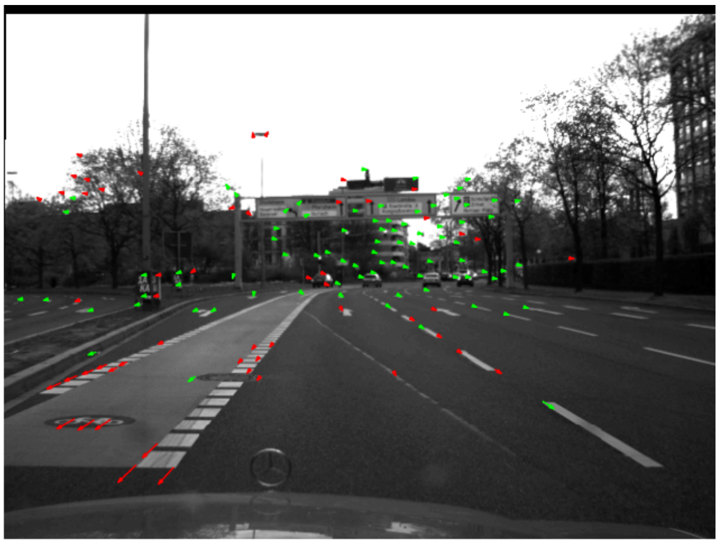


Figure 2.1: Illustration of optic flow estimation on a moving vehicle. By tracking the edges over multiple frames, one can use the displacement to estimate ego-motion. In this example, taken from [37], the green arrows are used.

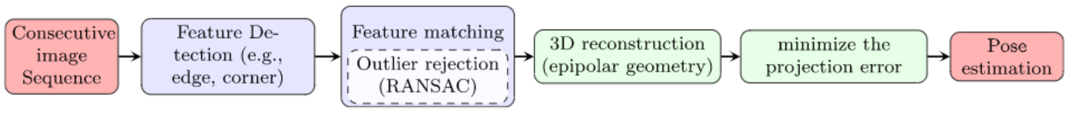

Consecutive image Sequence
Feature Detection (e.g., edge, corner)
Feature matching
Outlier rejection (RANSAC)
3D reconstruction (epipolar geometry)
minimize the projection error
Pose estimation

Figure 2.2: General pipeline of a feature detector. Adopted from [34]

brightness. This comes however at a computational cost that is proportional to the amount of features being extracted. Therefore, only a small amount of features can be maintained on systems with low computational capacities, discarding large amounts of information. The general pipeline of a feature detector is illustrated in Figure 2.2.

Typical feature extractors are the Harris detector [18], SIFT [26], SURF [2], FAST [39] and ORB [40]. These are all based on corner detection, as corners provide two dimensional intensity changes compared to only one for edges. An example of an edge detector is the Canny edgy detector [4].

Lastly, hybrid approaches are possible. These combine the traits of both direct and feature based approaches. In hybrid approaches, one can use the pros of feature detection based algorithms, but with the added option to fall back on direct approaches when for example no features can be detected.

2.1.2. Camera setups

Different setups exist to solve odometry-related problems using a frame-based camera. Most notable for this application is the decision whether to use monocular or stereo vision. Using a single camera, the velocity can only be calculated on a relative scale. This problem can be partly overcome by use of other sensors such as an IMU for ego-motion estimation, however still monocular vision suffers from issues in not knowing the scale of objects around and under/overestimate the distance to these objects.

Using two cameras in a stereo setup with a fixed and known distance between them, allows for the difference in measurements by the two cameras to provide information about the actual absolute environment. Stereo vision however suffers from regular need of recalibration due to vibrations in the vehicle and requires having a fixed baseline distance. If this baseline distance is short, objects far away will have almost identical optical flow estimates, whereas when the baseline distance is large, objects close by cannot be detected. Depending on research context, stereo vision might therefore not always be the answer.

The absolute scale is however not always a requirement. Depending on the end goal, one may consider whether it is required to know if something is close (which can be derived from optic flow on a monocular camera) or how far something exactly is. In a cluttered environment where precise navigation is required, or when a secondary sensor provides absolute measurements, one may opt for absolute measurements from the camera.

2.1.3. Performance under bad illumination

No matter the exact approach taken, frame-based camera related approaches all depend on the availability of information in the images that is inherently connected with illumination as this information is encoded in the incoming light. In dim or even dark conditions, incoming information is very low, rapidly degrading the accuracy of the vision system. More noise is introduced and the amount of trackable features degrades. In environments with obscured eyesight by i.e. fog or smoke particles, cameras are not able to detect anything beyond the obscuration. Visible light is hindered by particles and not penetrate.

Starr and Lattimer [48] evaluate the performance of several sensors in a fire smoke environment. By counting the edges that they were able to detect, they quantified the performance when smoke entered. They demonstrate that a frame-based camera performs very badly in fire smoke environment, only being able to detect edges when visibility is higher than $1m$ when tested in controlled environments. When visibility drops to 4 meters, performance starts degrading heavily. Also in a second, large scale experiment which represents real-life conditions where smoke is less uniform, targets at only 3 or 4.9 meters were completely invisible to the cameras.

2.1.4. Previous work

Arguably most work on avoidance or mapping systems using small drones is done using cameras as they are lightweight and not computationally expensive.

Zingg et al. [59] propose a wall collision avoidance algorithm based on a depth map generated by optical flow from a fisheye lens camera. Important to note is that computations are done externally, however hardware has made significant progression in the last decade. The algorithm is implemented to navigate corridors by estimating the distance based on the optical flow and the velocity as derived from IMU measurements. Optic flow is a product of the distance and velocity of an object at a certain angle and from there, the distance can be derived.

On the other side of the computational performance spectrum, heavy processors are used in an indoor navigation case[32]. A stereo-vision system is used to generate a local map of the surroundings in order to aid in navigation. A local map in this case is a map that only contains that what is in sight at that moment. This decreases computational cost without decreasing the performance. The experiments have been conducted in a large warehouse, providing a relatively large but cluttered environment.

Cigla et al. have implemented stereo-vision in leader/follower tracking tasks [7]. Here they search the received image for typical patterns that resemble drones. These are then tracked in order for the drone to follow its leader. The developed algorithm is more robust than others with high accuracy, low offsets and the ability to re-acquire the target in case of loss. Especially the last item may be significant for the current research as in cluttered environments, objects may move in front of each other which would result in the loss of tracking targets.

Another approach is seen in [25] where a static algorithm is extended to create a sense-and-avoid system with moving objects in 3D. From the stereo cameras, a depth map is constructed defining object bounding boxes in 3D. Based on several frames the velocity of the object is estimated and the pose in the next frames is predicted. This is then used to determine a safe route to travel. The implemented model is able to navigate a confined lab space with two walking human beings without problems.

[33] demonstrates an efficient optical flow algorithm for velocity and depth estimation on an extremely small drone. A novel optical flow algorithm is developed that is able to run on very low computational power. Using a simple finite state machine, autonomous navigation on a tiny drone is made possible.

2.2. Neuromorphic cameras

A neuromorphic camera, also called an event camera, is a bio-inspired camera system resembling the retina, first designed by Caltech graduate student Misha Mahowald and Caltech professor Carver Mead. A neuromorphic camera mimics the neural architecture from a real eye which utilises different methods to observe and subsequently analyse the surroundings.

Table 2.1 outlines some of the main differences between frame-based cameras and neuromorphic cameras. The first two differences, dense/sparse and synchronous/asynchronous, are a result of third

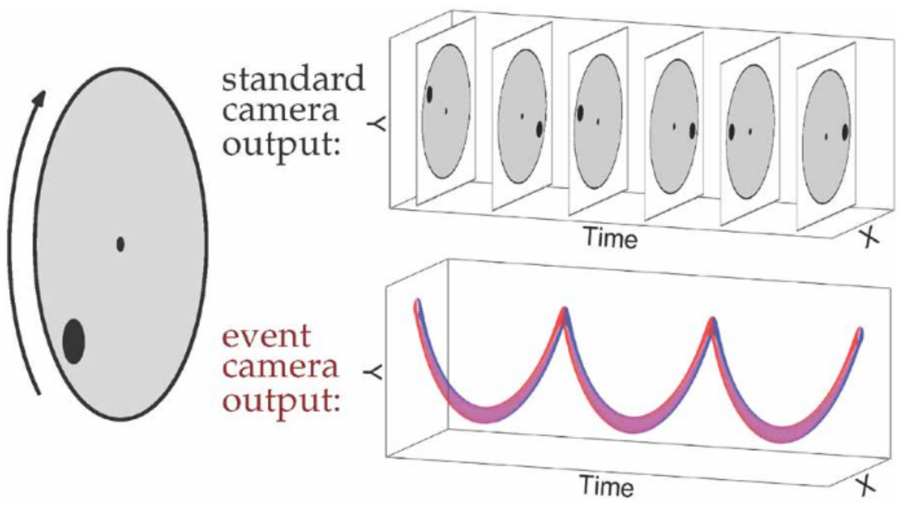


Figure 2.3: Illustration of the continuous property of an event-camera, adopted from [12]. The event camera shows a continuous change in brightness per pixel, red and blue denoting either increase or decrease. The standard camera only shows intermittently the current position of the dot.

Frame-based camera	Neuromorphic camera
Dense information	Sparse information
Synchronous	Asynchronous
Responds to incoming light	Responds to changes in incoming light
Near unlimited framerate	Framerate usually limited to $30 - 60Hz$

Table 2.1: Table outlining the main differences between frame-based and neuromorphic cameras

main difference, being that a signal is sent when the light intensity changes. This means that:

- A signal will only be sent when a change in intensity changes in that pixel (asynchronous)
- A signal will only be sent in the pixels that experience a change in intensity, others will be silent (sparse)

In more detail, the workflow of a pixel is as follows: a pixel continuously monitors the perceived brightness. When the brightness deviates by a predetermined amount from a set value, an event is sent out. The brightness at time of the event is then saved and will define the threshold for future events. The event contains the x and y location of the pixel, the event time t and the polarity p of the brightness change (increase or decrease). These events are sent to the periphery of the sensor and subsequently out of the camera, where they are read.

An illustration of the continuous property of a neuromorphic camera is seen in Figure 2.3. The standard camera shows the current position of the rotating black dot at a fixed rate, whereas the event-camera shows the change of position continuously.

As stated above, a neuromorphic camera responds to changes in light intensity, not to light directly. At constant lighting, changes in intensity are the result of ego-motion or of moving objects.

Important to note is also the inherent framerate of a neuromorphic camera, which is very high. A frame-based camera usually has framerates between $30 - 60Hz$. In sense and avoid applications, a high framerate increases accuracy and improves the sensitivity towards fast-moving objects.

2.2.1. Performance under bad illumination

To investigate the application in low-light conditions, one may look at the dynamic range of the sensor. The dynamic range is defined as the ratio between the largest and smallest observable values of light in one frame. This can artificially be increase by using High Dynamic Range (HDR) imaging. An illustration of HDR imaging is seen in Figure 2.4. This figure also illustrates the effect that tunnels or other sudden light changes can have on cameras: when a part of the image suddenly becomes bright, the camera can not display this part properly as its current lighting range is set to darker environments.

An event-based camera usually has a very high dynamic range of over $120dB$, being on-par with

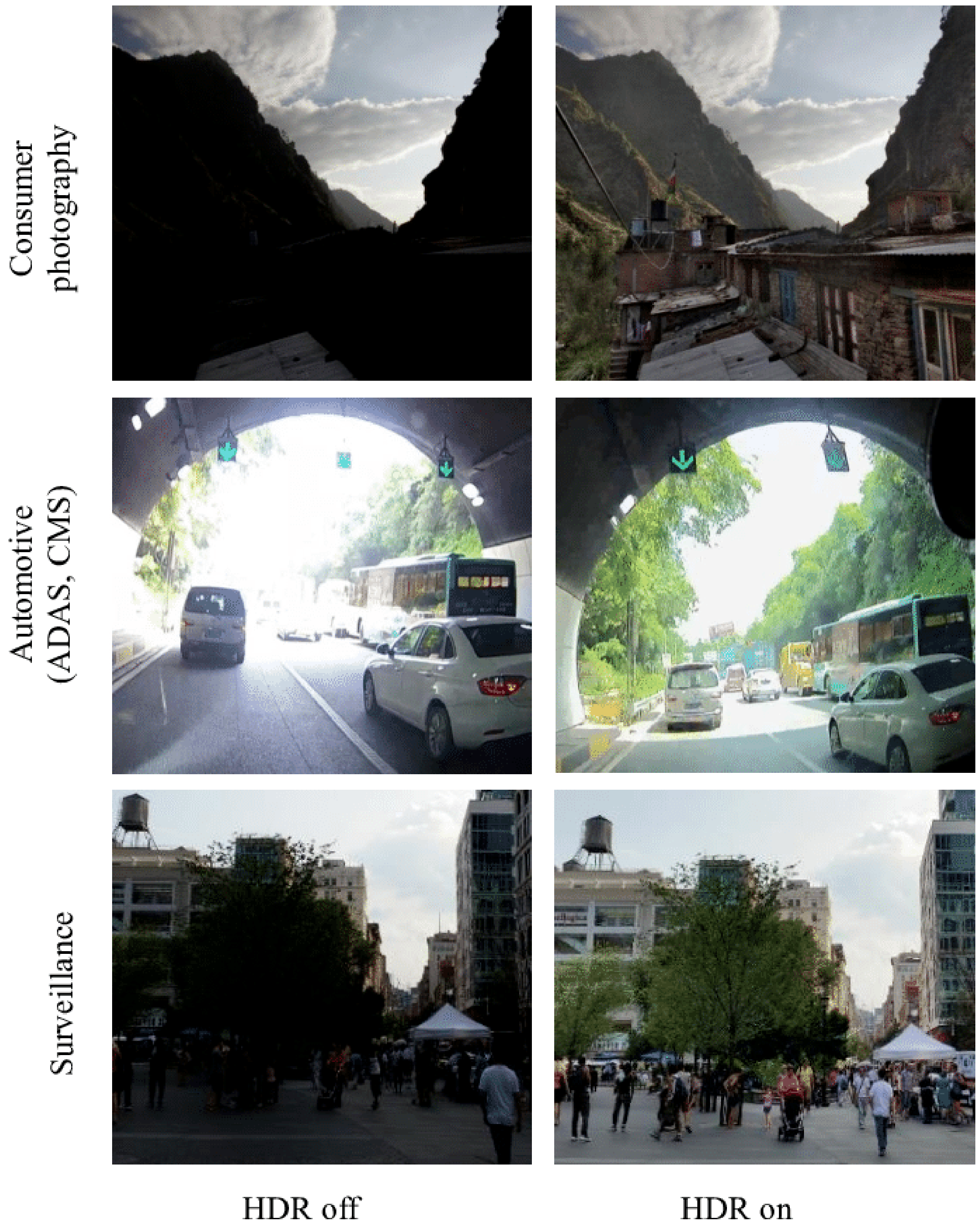


Figure 2.4: Illustration of HDR imaging. One can see that when the dynamic range is increased (pictures on the right), objects or scenes that where too bright or dark before, become clearly visible. Image adopted from [11]

human eyes, whereas high-quality frame-based cameras have a dynamic range of $60dB$. This makes the cameras excellent for both low light conditions as very bright conditions, simultaneously. Some light will always be required though, as in completely dark situations, the camera will not pick up any changes in lighting either.

Dust or smoke will prevent an event-camera from working well, as light can not penetrate well through those mediums and they actually scatter light in a way that will feed a lot of useless information to the camera.

2.2.2. Extracting key information

Just as with the frame-based camera, key information needs to be extracted from the incoming signals. However, due to the key differences between a frame-based camera and a neuromorphic camera, classic algorithms do not work.

In [17] the main challenges of feature detection and tracking are listed as the fact that the brightness change is dependent on the motion direction. Next to that, sensor noise may disrupt accurate estimations. In the same work, the issues that come up with optical flow estimation are described. Here, the same issue occurs: change in intensity on itself does not tell anything about the optical flow. In practice, this is solved by treating the event-data similar to regular frames.

2.2.3. Previous work

Falanga et al. [12] use event-based vision in both a mono- and stereo-setup to detect and avoid incoming projectiles. The motivation behind this work is that a large amount of drone crashes is due to e.g. birds or objects thrown at the vehicle. Therefore, there exists a wish for a very low-latency avoidance scheme. The authors describe that a state-of-the-art system using frame-based cameras has too high of a latency to accurately track fast moving objects. Therefore, an approach is taken where IMU data is used for ego-motion compensation. They do not optimise the ego-motion estimation, which drastically lowers computational cost. Detected events are then clustered and tracked. The avoidance part of the research will be elaborately discussed in chapter 4. Their algorithm is able to achieve latencies of only $3.5ms$, close to 5 times faster as would be theoretically possible using a $60Hz$ camera. This is largely due to the lack of optimisation in the ego-motion estimation. This does however result in a loss of accuracy. The authors argue however that the very fast response to a perceived object does make up for the reduced accuracy.

A completely different approach compared to the previous mentioned is that of [42]. Here, Sanket et al. use a series of neural networks to detect and avoid obstacles. The nets are divided in two: first, a net for deblurring, followed by a net for segmentation of objects. This is then followed by a control policy for avoidance. The nets are trained in a simulation and subsequently transferred to a real-world scenario without alterations. Unfortunately, the authors do not compare performance to non-neural approaches. They do however mention a 70% success ratio in low-light experiments.

2.2.4. Prior work by peer

In Dinaux et al.(2021)[9], a novel method for obstacle detection and avoidance using a neuromorphic camera is developed. The novel method introduced is called FAITH: an algorithm determining the course of the MAV by finding the focus of expansion (FOE). The FOE is a property of the optic flow, being a singular point from which the optic flow expands in a static scene where the observer only translates and rotation is zero.

Figure 2.5 illustrates the FOE estimation of the FAITH method. Edges produce optic flow vectors, normal to the edge. The FOE is bound by three half-planes that lead to the smallest area. In this case, vector (4) does not reduce the area and is therefore rejected.

Normally, the FOE estimation is challenging and computationally expensive. The newly proposed method however determines the FOE based on optic flow half-planes. When compared to other schemes, it outperforms on both computational time and accuracy. This is done by improving the work of [8] by applying a RANSAC-based scheme in order to decrease the amount of calculations required. Dinaux et al. describes no loss of accuracy while drastically decreasing computational power. Next to that, the computational power in the improved algorithm is at user’s choice by defining thresholds. When required, accuracy and power can that way be traded off against one another.

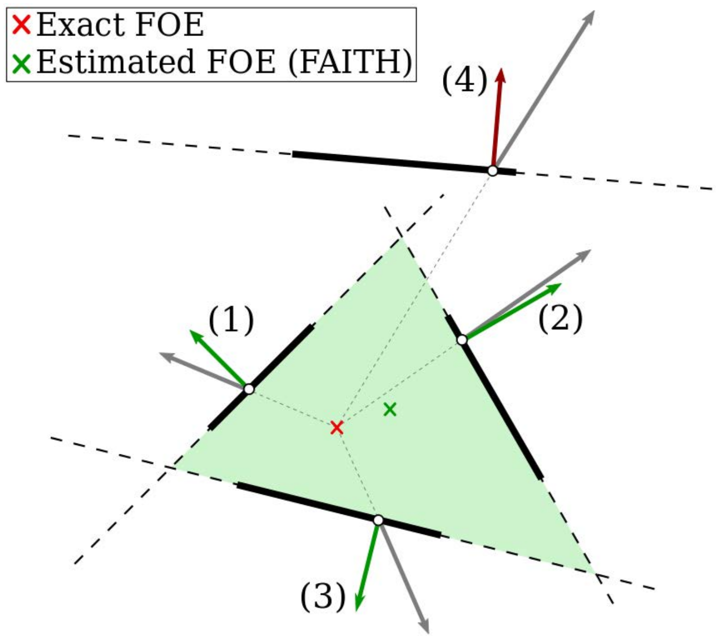


Figure 2.5: Illustration of the FAITH method from Dinaux et al. [9]. Coloured arrows represent the estimated optic flow vectors whereas the grey arrows represent the actual optic flow.

2.3. RGB-D cameras

RGB-D cameras are extended camera systems that next to the regular image as in section 2.1, obtain a depth map, a frame that contains information on the distance to the sensor for every pixel. A depth map can be generated using several methods, all including some form of light emission, often infrared, the reflection of which is then analysed.

One of those methods is to perceive depth by emitting structured light. Structured light is light that is emitted in a known pattern, often in infrared frequencies so that the RGB image remains undisturbed. Upon reflection, the patterns are disturbed and subsequently reflected towards the sensor, that can then determine the distance to the object.

Another way is to use a so called time-of-flight camera. This camera sends out the light and measures the time it takes to get back to the lens. Later LiDAR will be discussed which works according to the same principle. However, this type of system is able to perceive a whole frame at once.

RGB-D cameras are a recent development and made popular and readily available by the Kinect™sensor. Since the Kinect™sensor became available, a lot of research has been done using that sensor.

2.3.1. Performance under bad illumination

As RGB-D cameras send out their own illumination in the form of infrared light, they are able to perform well in dark environment. In [55] it is shown that using a Kinect™sensor, it is possible to construct a 3D image of a dark room using the IR sensor. They do however state that the self provided illumination is not enough to allow distant objects to be detected accurately. One can intuitively realise that light scattered over a larger area will reflect less intense. The RGB camera is not functional in those circumstances, only depth can be perceived.

In [48], the Kinect™sensor is tested. Both the regular and the depth sensor are tested separately. The depth sensor however was not able to perceive anything beyond the smoke and was found not useful. Especially as the smoke tested originated from a flame, the depth image was disturbed.

Interestingly, when operated outdoors, sunlight interferes with the RGB-D sensor as it provides a large amount of infrared light. This makes it one of the few sensors that perform better with bad illumination compared to very bright outdoor illumination. The RGB sensor however will still perform well in outdoor conditions, whereas that sensor will perform badly with a lack of illumination.

2.3.2. Previous work

Francis et al. [15] uses an RGB-D camera for flow estimation and obstacle avoidance under dim conditions. The RGB-D provides the depth information that is then used to reconstruct a 3D flow field. Results show that the robot is able to detect a tripod effectively in order to avoid it. Unfortunately, no direct comparison with a state-of-the-art frame-based camera is made.

Hua et al. [20] has developed an avoidance algorithm on a road robot, specifically for smaller obstacles. According to the authors, smaller obstacles tend to be overlooked by existing solutions that focus on larger objects. A two stage network is used to segment objects and mark them as obstacles on the path. Next to that, the ground plane is segmented from large obstacles and e.g. buildings next to the road. A method called Artificial Potential Field navigation, which will be described later, is then used for path planning.

2.4. LiDAR

As mentioned before, LiDAR is based on the same principles as a RGB-D camera, with a few key differences.

A LiDAR system emits laser beams that are reflected after which the time of flight can be calculated. Only one point can be determined at the same time, in contrast to the full images of the RGB-D sensors. This requires the LiDAR to rotate in a plane or in 3D in order to perceive a full image. Modern sensors can routinely scan larger areas or multiple LiDAR sensors can be employed to get a more broad view of the environment.

Next to the position and bearing, the Doppler shift can be used to estimate the velocity of an obstacle. In order to accurately estimate this on a moving object, one needs to know its own velocity. However, the same principle can also be used to estimate self-motion by measuring the Doppler shift towards e.g. a wall.

2.4.1. Performance under bad illumination

Under bad illumination, LiDAR works very similar to RGB-D cameras. The sensor sends out its own pulse of light and measures the returning signal. A main difference however is that due to the beam nature, LiDAR is less sensitive to sunlight. Still though, attenuation of signals may be problematic in large environments. Also, dust particles may scatter the light or reflect a significant amount, disturbing measurements.

2.4.2. Previous work

Wang et al. [52] uses 2D LiDAR to navigate a complex environment. The authors focused on a low computational impact in order to be able to detect obstacles online. The algorithm approximates everything as a circular shape, decreasing complexity while being able to avoid the 'circle'. The objects are tracked using a Kalman filter and GNN fitting. In [51], the same method is used to avoid obstacles in piloted flight. Here, the pilot commands a path and when the LiDAR detects an obstacle in the way, deviates from this path.

In [31] research is done on autonomous navigation in cave corridors. These present narrow, dark and highly irregular wall patterns, with the occasional rock sticking out. A 2D LiDAR system is implemented in a Nonlinear Model Predictive Controller to detect and navigate around obstacles. The paper focuses on the controller, however do show that at no point the LiDAR failed and demonstrate the ability to safely navigate a subterranean environment.

Sakthivel [41] uses a LiDAR sensor only to estimate the distance to an object. When the object is at 1.5 meters, they use the pinhole projection principle to estimate the size of the object from a camera image. Subsequently, the object can be avoided. This gets rid of a lot of the complications of tracking objects in space in order to avoid in 3D, but rather perform some more extensive measurements only once and react accordingly.

2.5. Radar

Even though radar has been around for decades, only recently the shift towards small vehicles has taken place, due to the rapid decline in size.

Generally, two types of radar exist: pulse radar and continuous wave (CW) radar. Pulse radar oper-

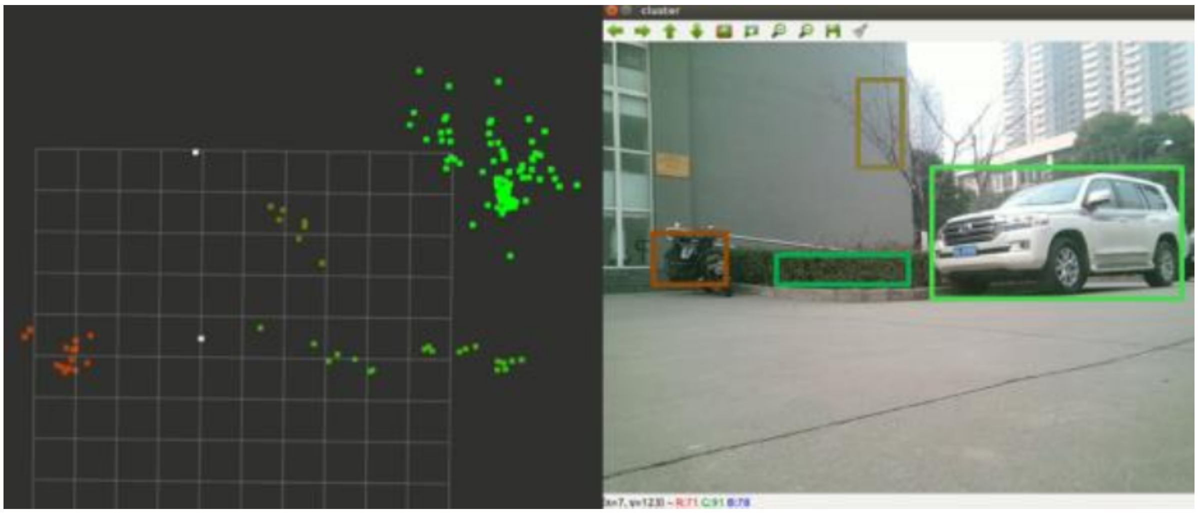


Figure 2.6: Illustration of radar scan information compared to the same scene perceived by a monocular camera. The colours correspond for different objects. Adopted from [56].

ates by sending out pulses and using the received information to extract features. CW radar operates according to the same principle, however, with as main difference the fact that it sends out radar waves all the time, as the name suggests. This allows CW radar to achieve much higher resolutions.

What makes radar especially interesting is that it is not dependent on the illumination and atmospheric conditions around it. Radar waves penetrate dust, fog or smoke and also work with low textured-areas as it is not dependent on reflected brightness.

Radar works by sending out an electromagnetic signal and studying its reflection. Using the Doppler effect, the range of the reflection can be determined. Combining this with the measured angle of incidence using multiple receivers, one can define a 2D range map, determining the range of objects at a certain angle. This then allows for feature extraction and subsequent processing which could include tracking or in this case, determining whether this object needs to be avoided or not. Figure 2.6 demonstrates an example of radar scan information. The image also contains the same scene as perceived by a monocular camera.

2.5.1. Performance under bad illumination

As stated above, radar does not require any illumination, as it does not contain a camera. It provides its own 'illumination' in the form of radar waves, penetrating any particles in the way. This makes it a great alternative to cameras in situations with bad or even no lighting.

[27] studies the application of radar in smoke environments. A self-developed algorithm `milliMap` is used for mapping in a smoke environment. Initially, without smoke, the LiDAR that accompanies the radar outperforms the radar. However, when smoke is added, the radar performance remains constant whereas the LiDAR quickly degrades. This is illustrated in Figure 2.7. One of the challenges described by the author is 'ghost points' generated by radar. These are points of reflection, measured by the radar, that are the result of multi-path and imperfect beams. These points then have the potential to define walls that are not there in reality. Another challenge is the sparsity of radar measurements. The LiDAR that is used has over 100 times more scan points. Sparsity might pose a problem in cluttered environments as smaller object may fall between scanning points.

2.5.2. Previous work

Only recently radar systems have evolved to the small scale required for UAV navigation. Object avoidance is scarce as most research is carried out regarding odometry in GPS-denied environments. Some of the most notable work is done by Scannapieco et al., exploring the limits of UAV navigation. In [44] and [45], the authors investigate path tracking using radar and compare it to the GPS track that

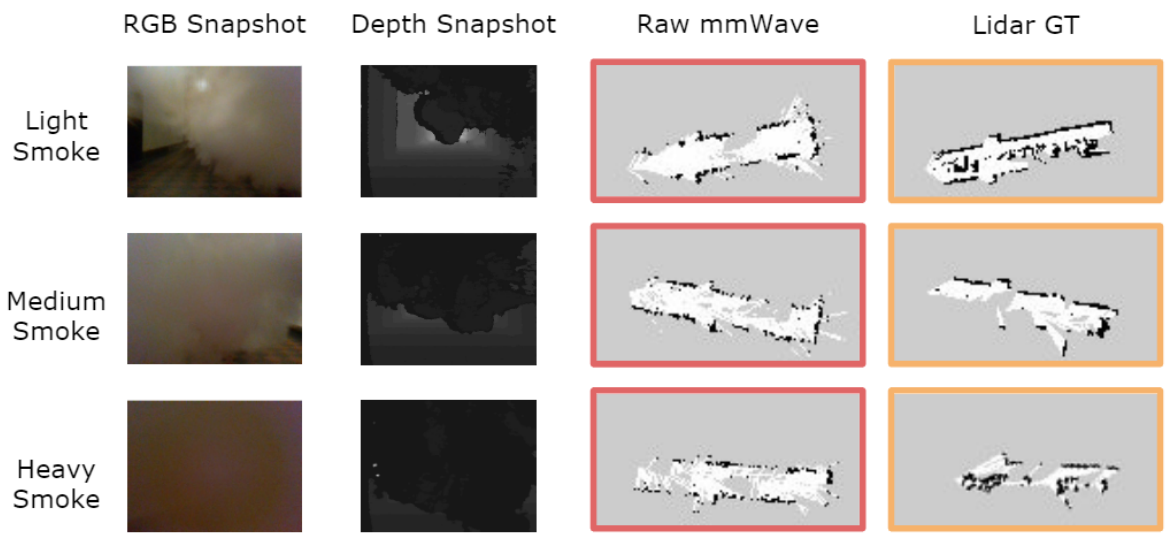


Figure 2.7: Comparison of LiDAR and radar data in smoke environments. Adopted from [27]

was considered truth. In [46], outlier rejection is added which leads to loosened constraints and results in more accuracy.

In [23] a radar avoidance system for large UAV's is developed using radar. Using a state-machine with three different modes, 'search', 'awareness' and 'avoidance', the radar system is able to move the UAV out the way of objects even when the approach speed is 1000 km/h. This is some of the earliest work on radar application on UAV's and is very concise. The state-machine is an interesting trait which might save computational power and thus battery power over long flights, however in cluttered environment this might not be ideal as the 'search' mode will be largely redundant.

Yu et al. has researched radar as extension of vision-based object detection in [56]. A monocular camera uses ORB-feature detection to detect points of interest. Then an EKF algorithm is implemented to match the features to the radar output. In this case, the radar processing is not elaborated on further than stating the output is the centre point of an object. Matching allows the true 3D coordinates of an object to be known instead of the relative position in the FOV of the camera as described in section 2.1. This allows very efficient 3D path planning as the exact position is stable over time due to filtering. While not primarily relying on radar, this does show a good application of the non-relative detection that radar provides.

2.5.3. Previous work by peer

Wessendorp et al. (2021) [53] has created an algorithm that uses radar measurements to detect objects in the way. A 24 GHz radar is used that is able to detect objects at 75 cm apart. Objects are detected, filtered and track for as long as they remain in the FOV.

As the data is quite noisy, data association and tracking are employed. This is done in two steps, the first being a a Global Nearest Neighbourhood optimisation in order to associate the data to an object. Then, a Kalman filter is used for tracking and subsequently validating object detections.

2.6. Sonar

Sonar is similar to radar and LiDAR, however instead of invisible radio waves, ultrasonic audio waves are used. The phenomenon is widely known as the method of navigation of e.g. bats and dolphins. The sonic waves bounce back on, in this case, walls, objects or beings. Then again, the time of flight and Doppler shift can be used in order to detect the position and perhaps velocity of obstacles.

Sonar works by emitting ultrasound waves that are reflected on surfaces. Much like radar and LiDAR, both the time of flight and the Doppler shift can be measured in order to acquire more information on an object. Similar as to LiDAR, sonar is unidirectional. Sonar is unable to receive the bearing of an object by only 1 measurement.

2.6.1. Performance under bad illumination

Sonar does not suffer under illumination conditions as light is not required for operation. However, under conditions with highly varying atmospheric conditions, such as a fire, accuracy is lost. This is due to the speed of sound being very sensitive to the density and temperature of the gasses it passes through. Next to that, dust and smoke particles may scatter and attenuate sound waves at a faster rate.

2.6.2. Previous work

As far as this research could find, no research has been done on using the sonar on drones to avoid objects. Instead, sonar is often used for altimeter and landing purposes. A ground plane gives the sonar a sturdy, flat object that reflects sound waves nicely at a relatively close range.

2.7. Summary

This chapter has discussed a wide variety of sensors and showcased some research that has been done using these. Every sensor has been concisely presented and its performance evaluated.

As one can read, there is significantly more work done on frame-based vision compared to event-based vision. The novel RGB-D sensor has a relative large amount of research done, compared to its short lifetime. Research does however quite evidently show that in low-light conditions, frame-based cameras are the definite underperformer. Table 2.2 summarises the performance of the different camera types.

Considering the system needs mounting on a drone, mass and size are relevant. Current RGB-D systems are quite heavy and bulky, exceeding the mass budget currently envisioned.

More research is required to develop the possibilities with neuromorphic and RGB-D cameras to the standards that nowadays exist in frame-based vision. For example the ultralight solution described in [33] is something that has not been matched using such cameras. Complete autonomous navigation in general is still open to a lot of research.

When objects are incoming instead of when the robot is navigating itself, it has been shown in [12] and in [42] that event-based cameras are very well suited to avoid incoming objects. The main challenge in using event cameras is to be able to computationally efficient separate ego-motion and external motion cue in order to effectively navigate dynamic environment. Application of neuromorphic cameras in e.g. [25] would certainly spike interest.

Sonar is not up to standards for our research. Sonar does not have the ability to detect bearing of objects without an extensive setup that a drone does not support. This makes navigating indoor environments impossible and therefore the sonar will not be considered, as this is crucial.

LiDAR and radar have similar features in their method of detection. However, radar in this analysis quite clearly comes out the better option:

- Using two receivers, both bearing and distance can be determined in a wide range
- Radar is able to penetrate dust and smoke
- Radar is not influenced by external light sources and is able to fly in pitch-black conditions

This validates the decision to continue the work as presented by Wessendorp et al.. A radar sensor is an excellent addition to a system that is meant for flight in difficult environments. The sensor will be paired with a neuromorphic camera, implemented with the FAITH as in the work of Dinaux et al., in order to provide all-round, robust object detection.

	Frame-based camera	RGB-D camera	Neuromorphic camera
Performance in low-light conditions	Little to no input	Only depth sensor is functional	Very good
Performance in conditions with obscured vision by e.g. dust, fog or smoke	Very bad, visibility in smoke degrades rapidly [48]	Very bad	Very bad
Computational power required			Requires novel methods of processing but is computationally more efficient
Framerate	Maximum several $100Hz$	Similar to frame-rate	Infinite/up to hundreds of MHz when processed
Mass, cost, etc.	Mass produced at low cost. Can be very small	Novel, heavy and bulky	Novelty and relatively expensive compared to frame-based cameras. Can be very small

Table 2.2: Overview of different camera options

3 Sensor fusion

With the preferred method of object detection installed, next we will address what to do with the information that is found.

The information needs to be processed in order to output potential objects that might be hazardous. The raw data contains noise and considering the sensors in question, of completely different format. In section 3.1 different methods of processing will be discussed.

After the data is made useful and in a workable format, in section 3.2 the method of coupling the data will be studied. Here, the data will be processed into a format that actually outputs (the lack of) obstacles in our way.

3.1. Processing data

This section will elaborate upon the data that is provided by the sensors and how this can be used subsequently.

3.1.1. Neuromorphic camera data

As described in chapter 2, the neuromorphic camera is asynchronous and delivers information sparsely. As CPU's work at a fixed rate, this already poses the first problem.

The asynchronous delivery of information can be solved by transforming the stream into a regular computer vision representation, in a frame at a fixed rate. This however comes at the cost of one of the key features of a neuromorphic camera, being that there are no frames to look at and that data in between frames might be lost. However, the fact that is measures a change in brightness instead of a direct pixel intensity, remains.

Alternatively, one might decide to implement a neuromorphic processor. This allows for asynchronous processing using Spiking Neural Networks (SNN), to which neuromorphic processors are better suited. However, neuromorphic processors are very novel and the implementation of an event-camera and a neuromorphic processor on a drone is a research topic in itself.

As mentioned earlier, the data gathered by the camera has information on the change in brightness, if present. Upon registering such an event, they send out data packs containing the time, 1 bit polarity (increase or decrease) and the x and y location. The camera does thus not elaborate on the amount of change it registers. The event can be described as

$$e_k \doteq (\mathbf{x}_k, p_k, t_k)$$

with e_k the event k at time t at location $\mathbf{x} = (x, y)$.

Depending on the system used, a range of different methods of representation in the backend is possible. A broad division in two main groups is possible:

- Methods that operate on an *event-by-event* basis: these methods allow the system to change upon arrival of a single event. These are often used in SNN's as those are well-able to use single events.

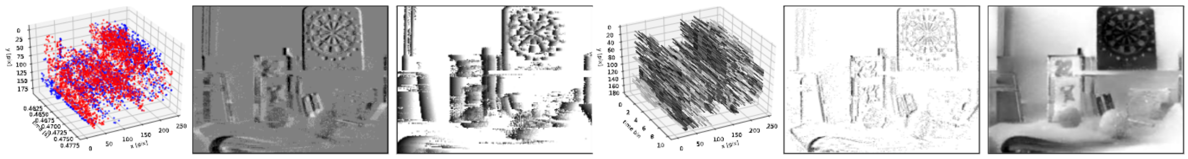


Figure 3.1: Overview of event-based visualisations, adopted from [17].

- Methods that operate on groups of events: these methods introduce some latency as the system gains information over a period of time. After grouping events, latency is the same.

Events over time can be visualised in a range of ways. Visualisations often are implemented in order to have a frame with the information gathered from the camera in a similar format as a frame-based camera in order to make it easier to process in a more regular setting. Some of the methods applicable are listed below, adopted from [17]:

- **Events in space time:** this method shows all events per timestamp. A 3D plot can be generated in order to visualised the events over time.
- **2D histogram:** this method is able to count events or accumulate polarity in order to generate an image. By doing this, one may observe e.g. depth or relative velocity as intensity can be observed.
- **Time surface:** Here, all events get plotted on a grayscale map, which fades over time. This way some sort of path can be observed.
- **Interpolated voxel grid:** This is a different type of histogram that better preserves the temporal information by keeping track of movements per predefined time 'bin'.
- **Motion compensated event-image:** This method uses both events as an hypothesis on the motion to attempt to align edges of motion in order to produce a sharp edge. This representation looks familiar when observed and is often used for feature tracking purposes.
- **Reconstructed images:** This uses the brightness information to reconstruct the image in grayscale.

The methods described below are visualised in order in Figure 3.1. The 2D methods are compatible with classical computer vision methods as those are designed for 2D inputs as well. For the 3D-based methods such as the 2D-histogram, novel approaches are required.

After the image information is collected and processed in a way of our liking, some sort of calculations can be done. In this research, of interest are optical flow estimations, odometry and/or feature tracking. First, the next section will discuss how radar data is processed. After that, sensor fusion will be studied to investigate how the data of both sensors can be used to achieve the final processing steps.

3.1.2. Radar data

Radar measures the bearing and distance to reflective surface. In his work, Scannapieco et al. [44] describes radar aided navigation on a UAV. A transmitting antenna sends out radar waves that are subsequently received by receiving antennas. When a signal is reflected on a surface, the distance of that object determines the reflected frequency. Then, upon receiving the reflection, the range to an object can be determined.

Using multiple radar receivers, the difference in range can be used to also find the bearing θ using the following formula from Wessendorp et al. [53]:

$$\theta(f_R, N) = arcsin\left(\frac{\Delta\omega}{\pi}\right) \tag{3.1}$$

with $\Delta\omega$ denoting the phase difference.

A reader may observe that the workings of radar are described much more concise than those of the event-camera. However, it is important to note that radar depends on a fairly simple principle that

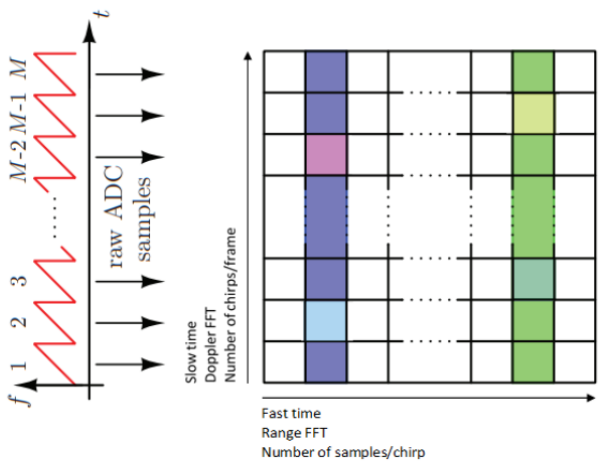


Figure 3.2: Illustration of 2D Fast Fourier Transform of radar data. Horizontal axis in grid denotes single frame, vertical axis denotes frames over time.

works and where raw measurements and final calculations do not differ a lot. An event-camera is not only fairly new, but also requires much more processing in order to provide meaningful information. The raw measurements on itself are worth less than those of the radar.

Next to range and bearing, a radar system is able to determine the velocity of objects. By determining the FFT over multiple frames, the frequency shift over multiple timestamps can be determined which tells the observer about the velocity of an object. By FFT'ing over more than two frames, both range and bearing velocity can be determined [53]. This is illustrated in Figure 3.2.

3.2. Sensor fusion

The previous section has dealt with some of the preprocessing steps required to make sense of the incoming data. To recap: the following inputs are gathered:

- Event-data gathered by the camera. This contains location, timestamp and polarity of events.
- Radar data in the form of distance, bearing and optionally velocity
- IMU data on accelerations and rotations

The incoming data needs to be filtered and fused in order to remove noise and combine measurements to increase accuracy. In this section, some sensor fusion methods will be elaborated upon. Not only will we look at the specific combination of sensors that we use, but also other combinations will be reviewed if their coupling is interesting.

3.2.1. The principles of sensor fusion

In their literature review [1], Alatise and Hancke review the state of the art of sensor fusion on autonomous robots. They discuss various sensors in applications such as odometry and object recognition, followed by sensor fusion algorithms and a classification thereof. Three fundamental ways of combining sensor data are listed:

1. **Competitive:** Here different sensors measure the same quantity in order to allow fusion or to switch between sensors if required. This method works well with heterogenous data sources.
2. **Complementary:** Here sensors do not depend on each other but rather complement each other with measurements in order to resolve incompleteness of sensor data. An example is complementing a monocular camera with an IMU to derotate the images or using multiple cameras to achieve a broader field of view.

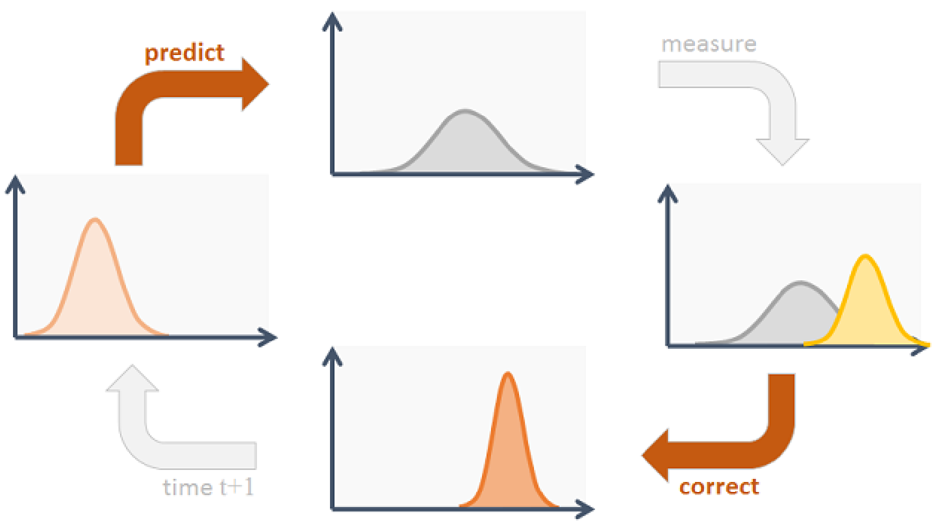


Figure 3.3: The Kalman Cycle. In the top, predicted state is displayed. This is then combined with measurements to achieve a next best prediction in the bottom.[1]

3. **Cooperative:** This method uses the information of two sensors to obtain data that would not be available from a single sensor. An example is using stereoscopic vision to achieve a 3D image of the environment. These type of sensor fusion methods are arguably the most difficult as high accuracy is required in all participating sensors.

Sensor fusion approaches are then categorised in two main categories: State Estimation Methods and Decision Fusion Methods. Here we will focus on State Estimation methods.

Two major methods are the Kalman Filter and the Particle Filter. The Kalman filter is perhaps the most popular sensor fusion method in existence. In short it calculates the current state using the previous state and current measurements. Being very easy to implement, the equations for the basic Kalman Filter are as follows:

$$\hat{x_k} = F_k \hat{x_{k-1}} + B_k u_k \tag{3.2}$$

$$P_k = F_k P_{k-1} {F_k}^T + Q_k \tag{3.3}$$

Where $\hat{x_k}$ is the estimated state of system x_k. P_k is the predicted covariance matrix, F is the matrix that denotes system dynamics, B is the control matrix and Q is the noise covariance.

The estimates are updated using another stage of equations, given as:

$$\hat{x_k}' = \hat{x_k} + K'(z_k - H_k \hat{x_k}) \tag{3.4}$$

and

$$P_k' = P_k - k' H_k P_k \tag{3.5}$$

with z_k the measurements, H the transformation matrix, R the covariance matrix of the measurement noise and k the time interval. K is the Kalman gain which denotes the amount of update needed. An illustration of the Kalman filter cycle is given in Figure 3.3.

It is important to note that the Kalman filter is linear, where a lot of real-life problems are not. For that purpose, the Extended Kalman Filter is designed. This is a Kalman filter that includes linearisation for estimations. Another general approach called the Unscented Kalman Filter exists which is able to better predict highly nonlinear systems by sampling close to the mean.

Another approach to filtering is the Particle Filter. This is a filtering approach that is able to approximate PDFs of any form. In this method, a probability density function is build using random samples called particles. These are propagated over time and rejected when required in order to gain confidence in the PDF. PF's are used in for example tracking applications, allowing an accurate and stable method of tracking targets.

[1]Source: `https://www.codeproject.com/Articles/865935/Object-Tracking-Kalman-Filter-with-Ease`

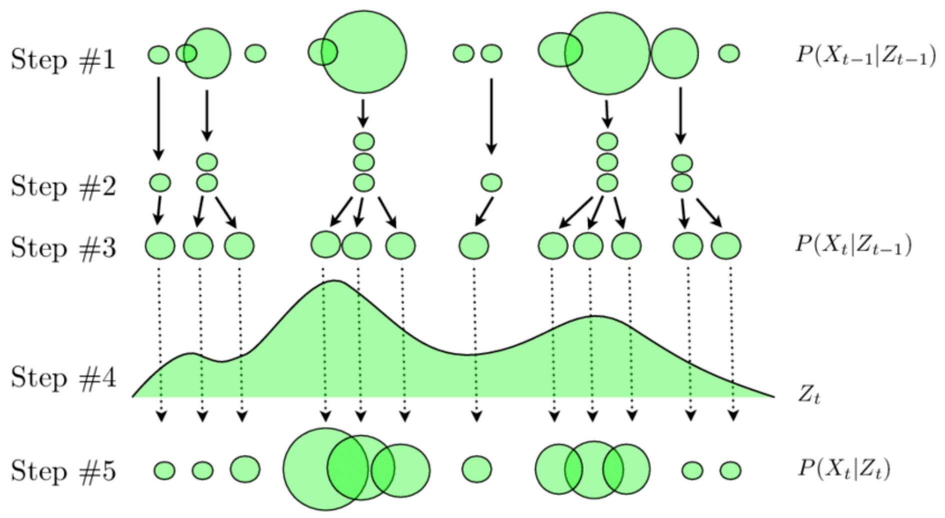


Figure 3.4: Illustration of the 5 steps of a particle filter. Adopted from [35]

Figure 3.4 illustrates the five steps of the particle filter. First, the initial weights of the particles get defined. From there, the particles get re-sampled according to their weights. A new PDF is determined according to the updated samples and finally, the weights are determined again.

But why do we implement sensor fusion? Alatise and Hancke cite the following benefits of sensor fusion techniques:

- **Reduction in uncertainty:** Multi-sensor data reduces the uncertainty by combining measurements and gaining confidence in the combination of those.
- **Increase in accuracy and reliability:** In the case of partial failure, the system is able to provide information as it does not rely on one sensor only.
- **Extended Spatial and temporal coverage:** In for example the combination of an accelerometer and a camera, the camera is able to gather acceleration information from a large area whereas the accelerometer only gather information from the route it takes.
- **Improved resolution:** Multiple independent measurements increase the resolution of the value
- **Reduce system complexity:** Sensor fusion can standardise output, therefore making later computation steps less complex and improving ease of operation.

A different type of classification is shortly described in [34]. In his work, Mohamed et al. specifically discusses Visual-Inertial Odometry approaches. Combining an IMU and a camera can be done in different ways and fall in two categories:

- Tightly coupled approaches
- Loosely coupled approaches

and

- Fusion-based methods
- Optimisation-based methods

The different coupling approaches denote the moment in time that the information is coupled, being either delayed or non-delayed. Loosely coupled methods combine the information in a later stage, allowing for lower computational complexity. Tightly coupled approaches instead fuse the information in early stages, allowing for generally better performance at the cost of higher computational complexity.

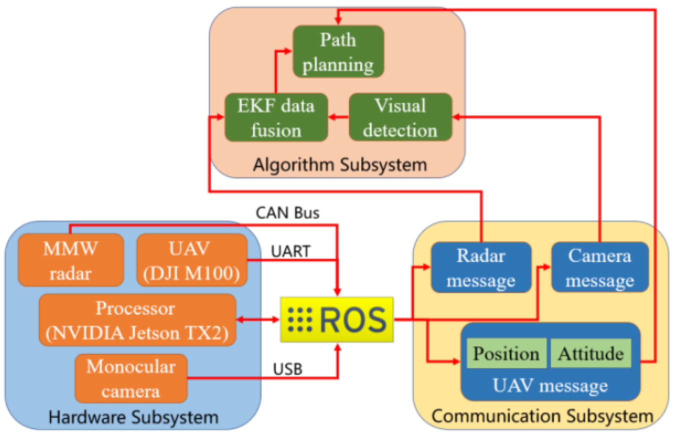


Figure 3.5: Illustration of the system as presented in Yu et al.[56].

Loosely coupled approaches are filter based and use for example the Kalman filter for coupling. Tightly coupled approaches can be either fusion-based or optimisation based.

Fusion-based approaches generally consist of a prediction step and an update step, as the Kalman filter explained earlier. Optimisation-based approaches estimate the pose by optimising the information extracted from sensors. They generally outperform filter-based approaches. However, the optimisation requires minimising an error function iteratively, requiring more computational resources.

3.2.2. Applications of sensor fusion

In [10], a particle filter is implemented to cope with GPS-denied areas on an airfield. In this case, the aircraft is provided with a map of the environment. The particle filter is used to determine the probability density function of the Markov localisation process.

In [56], a frame-based camera is coupled with a millimetre wave radar. Here, the coupling exists of the radar data with ORB features points that are already extracted from the images. The radar already grouped the measurements and outputs the centre points and corresponding velocities of obstacles. The system is illustrated in Figure 3.5.

The radar and camera data come in two different formats, respectively in range-bearing and in pixel coordinate systems. First, the radar measurements are projected on the pixel plane. Then, an EKF is implemented to combine the outline information of the camera with the range, bearing and velocity data of the radar. This method allows for obtaining the true 3D locations of obstacles instead of the relative positions only.

In [54], a set of relatively cheap sensors is fused in order to achieve object avoidance. An IMU, camera and LiDAR are fused using an error-state Kalman Filter (ESKF). The ESKF has as advantages that the error variables usually are small and near-zero, decreasing the chance of e.g. gimbal lock compared to regular Kalman filters. Next to that, numerical stability is guaranteed. In an ESKF the true state is considered a sum of the nominal state and the error state.

Papachristos et al. [36] use an EKF for fusion of a thermal camera, a stereo frame-camera and an IMU for navigation through obscurants. The EKF is used in conjunction with the open source Robust Visual Inertial Odometry (ROVIO) [3] that couples the tracking of image patches with the EKF to constrain the Kalman filter computational requirements.

Scannapieco et al. [47] uses the radar as a support system for a monocular camera in a odometry estimation algorithm. The solution he proposes, relies mainly on the camera. However, the radar is used for estimating the true scale. Whereas normally visual odometry would suffer from the lack of a true scale, the radar is able to provide real life measurements without the drift that e.g. an IMU would carry.

A similar approach is taken in [41]. Where again not completely fusing the sensors, in this case LiDAR is used to provide true scale for a camera. Here use is made of the pinhole projection principle that cameras have. In order to get a sense of the true height, the distance to an object has to be known, something a camera normally can not provide. However, the LiDAR sensor that the MAV is also equipped with is able to provide exactly that information.

In [29], the authors argue that in literature there is an abundance of sensor fusion methods, however all tightly bound to a specific purpose. In their paper, they propose a generic framework for multi-sensor fusion, allowing a theoretically unlimited amount of sensors to be included, regardless of scale, absolute/relative measurements or delay. Their proposed architecture is called Multi-Sensor Fusion EKF (MSF-EKF). Next to fusing, their proposal also includes calibration, re-linearization of constraints and efficient tracking of covariance terms. A `C++` based algorithm allows everyone to implement this system to their sensor fusion approach.

3.2.3. Sensor fusion of an event-camera

Event-cameras have been fused with several sensors, including RGB-cameras and IMU's, but also with audio sensors and physiological sensors. A lot of these works are done in other field than the autonomous drone field. However, these novel methods may be used to draw inspiration from, as the interesting properties of a neuromorphic camera make it an interesting option for several solutions that may not spring to mind very quickly.

One of the more out of context works considering the topic at hand is [49]. This work fuses an event camera with a electromyography (EMG) sensor, used for 'listening' to human body signals, in order to get a very low latency solution that can be implemented in e.g. prosthetic limbs. The low latency in that case provides a safety factor. The EMG signals are transformed to spikes that, together with the spikes from the camera, are fed into a neural network that serves as fusion layer.

In [43], event cameras are used as the first step in a Voice Activity Detection system. Then, a neural network is sparsely activated through a neural network in order to achieve a low-computational cost system as the cameras allow for sparse activation and work as a gate in the sense that they can very accurately and efficiently determine when lips are moving, as opposed to either listening the whole time or using a higher power consuming regular camera.

Another way to fuse an event camera is to fuse it in the form of stereo vision. This is done for example in [57]. Here, two neuromorphic cameras are used in conjunction and fused through an optimisation function in order to get depth estimation. By introducing and subsequently minimising an energy function that denotes consistency as function of depth, a 3D reconstruction of a scene can be made.

A work that describes fusion of an event camera with other sensors on a drone is [50], where an event-camera, frame-based camera and IMU are fused in order to achieve a robust odometry solution. [50] uses the three sensors in order to find an optimal estimate on the current speed and orientation. The sensors each deliver their own estimates on the current state of the UAV. Then an optimiser is used to optimise the state estimation.

IMU's are generally used in the fusion with event-cameras to derotate the flow. For example in [12], the IMU is used to estimate the angular velocity of the UAV in order to compensate for that in the motion estimation done by the event camera. Without this compensation, an event-camera can not determine whether a motion is due to ego-motion or external factors.

Here a clear research gap is identified: sensor fusion with a event-based camera, not having odometry as the end goal. This research gap fits well within the research thus far.

3.3. Summary

This section has discussed the principle of sensor fusion and discussed some examples of sensor fusion in similar research. A wide variety of methods is available and one can tailor the exact solution to the needs of the solution.

The solution that fits best is dependent on the way information is put out and on computational power available. High accuracy is possible through optimisations, however, this comes at a huge expense regarding computational power. Filtering has less accurate results, however does allow for faster computations.

A research gap was identified regarding the fusion of the event-camera and radar. Event-cameras are often used by itself fused only with an IMU, and sometimes with a regular camera. This research

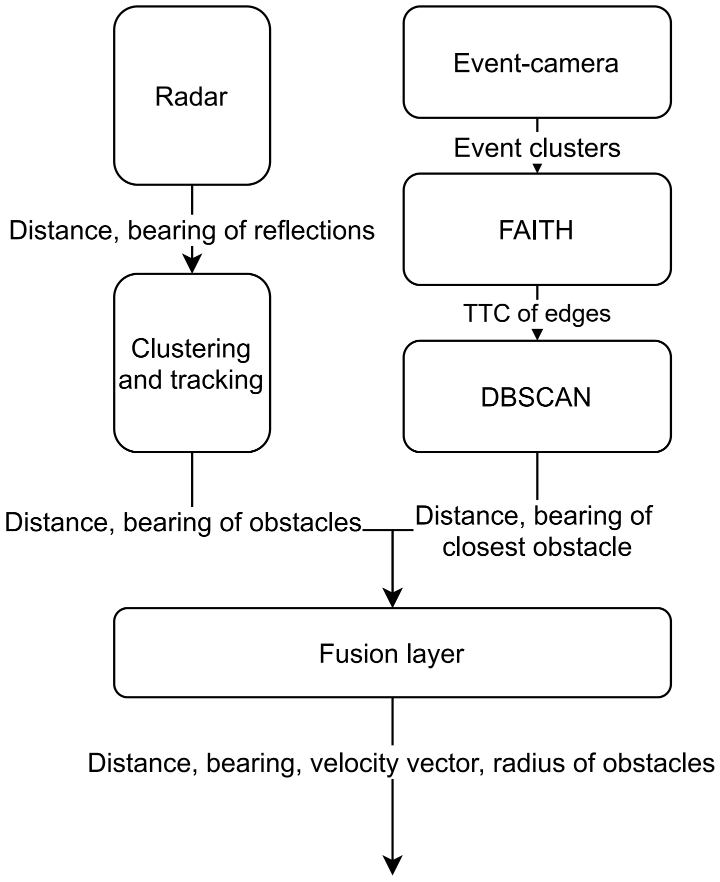

Radar
Event-camera
Event clusters
FAITH
Distance, bearing of reflections
Clustering and tracking
TTC of edges
DBSCAN
Distance, bearing of obstacles
Distance, bearing of closest obstacle
Fusion layer
Distance, bearing, velocity vector, radius of obstacles

Figure 3.6: Proposed high-level system to be developed during thesis phase

has found no evidence of fusion with different types of sensor such as radar or LiDAR.

The main challenge of the remainder of this research will therefore be to develop an efficient fusion algorithm in order to best cope with the data that is gathered by the sensors and that fits the to be chosen avoidance strategy well.

An illustration of the to be designed fusion layer is projected in Figure 3.6. Here, a rough pipeline of the system is portrayed and indicated what information will be fed into the fusion layer and what the ideal output will look like.

4 Avoidance

In the previous chapter, object detection was discussed. Already it was found that a lot of research unifies the detection and avoidance. This makes sense, as the way something is detected - early or late, accurate or inaccurate - will largely influence how one will avoid something. A slow avoidance system will not work on a system that only detects objects when they are very close.

Where possible and relevant, the avoidance strategies mentioned in chapter 2 will be discussed. Also, new strategies will be introduced. Concluding this chapter will be a discussion on the approach this research will take.

This research will focus on short-term avoidance and replanning. This implies that no extensive algorithms that reconsider the complete navigation will be discussed. The algorithms should have as end goal to avoid an object and continue on the path that was initially planned.

More specifically, this study will discuss avoidance of objects that pop up suddenly. These can be either static or dynamic. Avoidance will be local and no global navigation will be discussed.

4.1. Overview of algorithms

This section will discuss two families of algorithms that are highly applicable in our situation. The drone in question is required to navigate in a straight line where possible and avoid obstacles if required. In order to do that, we only want to consider a very short term planning, reacting on the environment.

4.1.1. Artificial Potential Field methods

A common algorithm for object avoidance is the Artificial Potential Field (APF). First described in 1985 by Khatib [22], it has been well tested in a variety of environments. Recently, as quadrotors have become popular and their computational capabilities increased, they've also acquired the ability to use APF for navigation purposes.

APFs, as the name might suggest, create a potential field that sums attractive and repulsive forces in the field. Repulsive forces are exerted by objects whereas attractive forces are generated by the robot itself and the goal. It is therefore possible to choose a path with the highest potential in order to move towards the goal. Moving past objectives will automatically allow the robot to move towards the objective again as the potential is higher in a direct path towards it.

The potential field can be described as follows: let U_A and U_R be respectively the attractive and repulsive potential field, then the Artificial Potential Field U_{APF} can be described as

$$U_{APF} = U_A + U_R$$

which allows us to determine the gradient F by taking

$$F = -grad(U_A) - grad(U_R)$$

which tells us the most ideal way to travel.

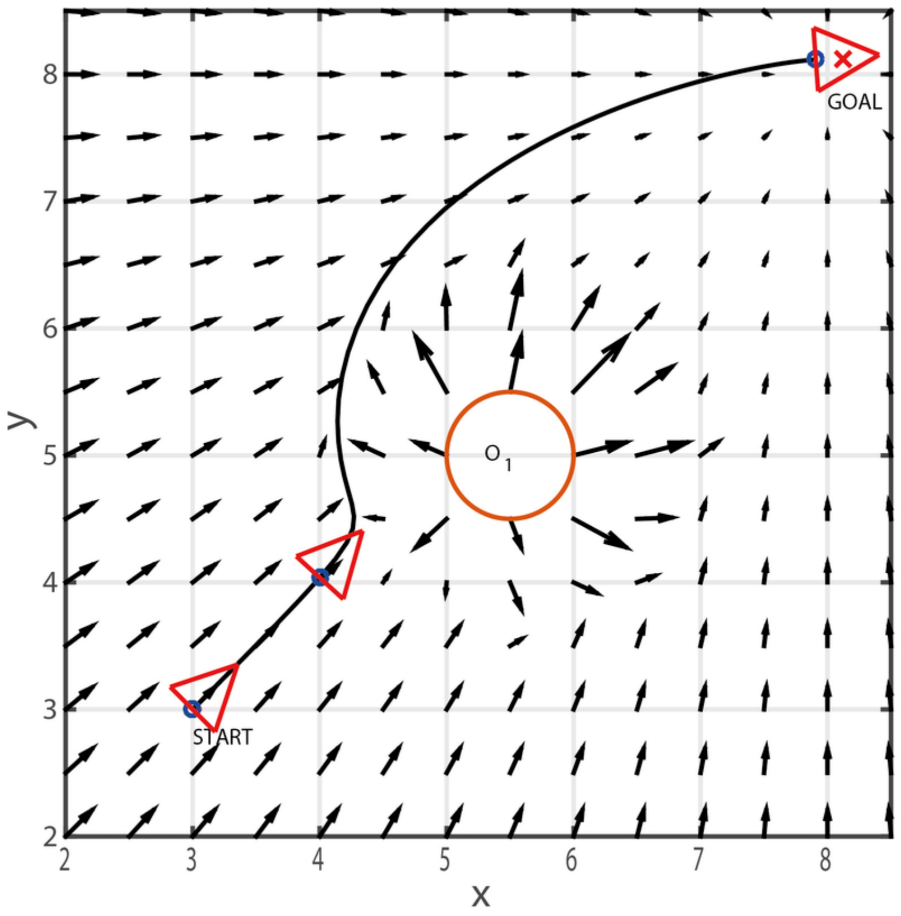


Figure 4.1: Illustration of a potential field adopted from [13]. Object O_1 repels the robot, making the potential gradient go around the object.

In Figure 4.1, a schematic 2D representation [13] is presented. One can see that when the attractive and the repulsive fields are added, the gradient goes around the obstacle. It is important to note that the arrows in this image are not the potential field, but the gradient.

Recently, [30] has implemented a modified algorithm on a quadrotor that overcomes two flaws in the original algorithm. The so-called Local Minima and GNRON problems are solved by implementation of a virtual force. This allows the APF to overcome the extremities caused by local minima and GNRON.

Another proposal on a novel method based on the APF algorithm is done in [5]. Here, Chen et al. add the control of the UAV in finding the optimal route. This is done by adding the control force and slightly editing the APF principle into an optimisation problem where the most optimal route is found.

In [58] an adaption of the APF algorithm is proposed that includes avoidance of random and dynamic obstacles. It includes physical constraints of the MAV to ensure the avoidance solution lies within the physical limitations that the craft has.

4.1.2. Velocity Obstacles

Another method that has survived the test of time is Velocity Obstacles by Fiorini and Shiller [14]. Also implemented on Wessendorp et al. [53], Velocity Obstacles is a is fairly simple but effective algorithm that finds collision free paths when an obstacle, static or dynamic, is detected.

The principle of Velocity Obstacles is to use vector calculations to avoid collisions. An illustration of the method is seen in Figure 4.2. A collision between object A and B is predicted when the velocity vector

$$\mathbf{V}_{AB} = \mathbf{V}_A - \mathbf{V}_B$$

lies within the cone originating in object A and limited by the circle defined by $r_A + r_B$ that has its origin in B.

In order to avoid an object, the velocity vector of either A or B can be changed. Assuming A is the controlled object, the velocity or bearing can be changed. In case of the illustration in Figure 4.2, the following options exist:

- Increase velocity, A goes in front of B
- Decrease velocity, B goes in front of A
- Adjust bearing to the left, A goes in front of B
- Adjust bearing to the right, B goes in front of A

Depending on the control strategy employed, one can consider not all options possible. In the case of Figure 4.2 one might observe that an incredible acceleration is required from A to achieve the velocity required. This brings us to the solution space that is bound by the control strategy of the drone and further limited by the cone(s) formed by the drone and any obstacles near.

Fiorini and Shiller also propose both an on- and offline search for path planning. Here only the online approach will be discussed. The authors propose an incremental trajectory generation that chooses the best next step based on the perceived obstacles and avoidance strategy set, which can be to follow a line as straight as possible, a maximum velocity strategy or a very safe strategy.

Interesting about this approach that it focuses on the velocity vectors of objects and not on the position. It plans velocity changes for avoidance instead of static waypoints. This makes it excellent for avoiding dynamic obstacles and also well suited to incorporate in autonomous robots.

What makes it suitable for our research is that it can be implemented very well with our global path planning. The global path planning gives a direction of motion required and the Velocity Obstacles can determine whether that is possible or give a next best alternative.

Fuad et al. uses an adaptation called Hybrid Velocity Obstacles to avoid objects and also takes into account nonlinear trajectories. This is done by merging different versions of Velocity Obstacles, being for static and dynamic obstacles and accumulating the sets of 'free' velocities that are put out. From this accumulated set, the velocity that takes into account all objects is chosen.

In [6], Velocity Obstacles is implemented on a micro-UAV with very limited sensing capabilities. The only measurement that the UAV makes is the distance to an object. Then, using some assumptions and

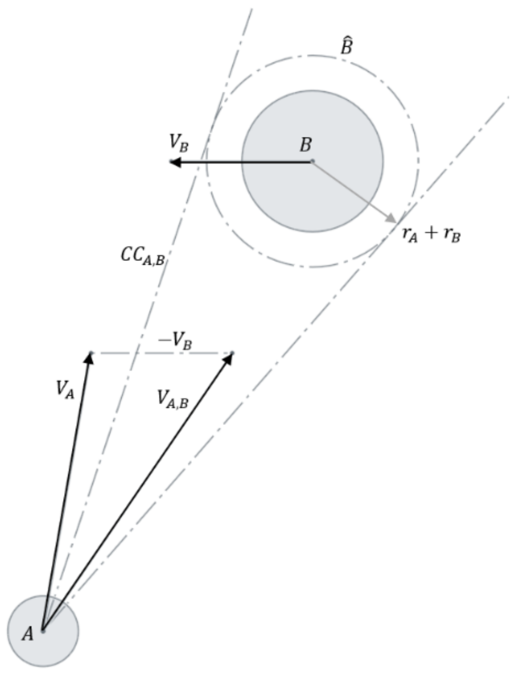


Figure 4.2: Velocity Obstacles, adopted from Wessendorp et al.(2021)[53]. In this image, the relative cone defined by the radii and velocity vectors well illustrates the principle. Vector $V_{A,B}$ needs to be outside of collision cone $CC_{A,B}$

iterative computing, ranges for the radius, bearing and velocity of an obstacle are defined. Then a large range of Velocity Obstacles is determined. It can be guaranteed that the new heading chosen is not a part of the 'true' Velocity Obstacles-generated headings as the actual collision courses are within the union of all probable courses. In Figure 4.3 one can see all the possible collisions that are generated. The authors stress that this method is no 100% foolproof as there are some exceptional cases where an object can not be avoided in time. However, considering the limitations in sensor availability, the method shows great innovative thinking and produces interesting results.

4.1.3. Optimisation-based avoidance methods

Next to aforementioned methods, a large group of algorithms exist that iteratively search for the most optimal path. These methods consider a longer path than Velocity Obstacles and allow more options than the Artificial Potential Field, to find an optimum. This section will shortly elaborate on a few variations.

Genetic Algorithm

A Genetic Algorithm is, in this context, a search algorithm based on an evolutional model. Different solutions, evolve over time and are selected or 'bred' according to the fitness score, which can be a function of e.g. distance to the objective and time to get there. The path can be planned ahead as far one wants, allowing the user to have some freedom in the computational cost.

In [24], a genetic algorithm is used for navigation in a dynamic environment. In every step, the algorithm checks whether a potentially new object is detected on the path. If so, the path is reconsidered and the robot follows the new path.

Multi-Step Look-Ahead Policy (MSLAP)

The MSLAP algorithm works similar to a genetic algorithm in a sense that it considers multiple paths and chooses the optimal one. However, MSLAP is not based on evolution and chooses its policy from a set of discretised actions. By considering a predefined amount of timesteps, the algorithm can decide which option in the long term will be most suitable.

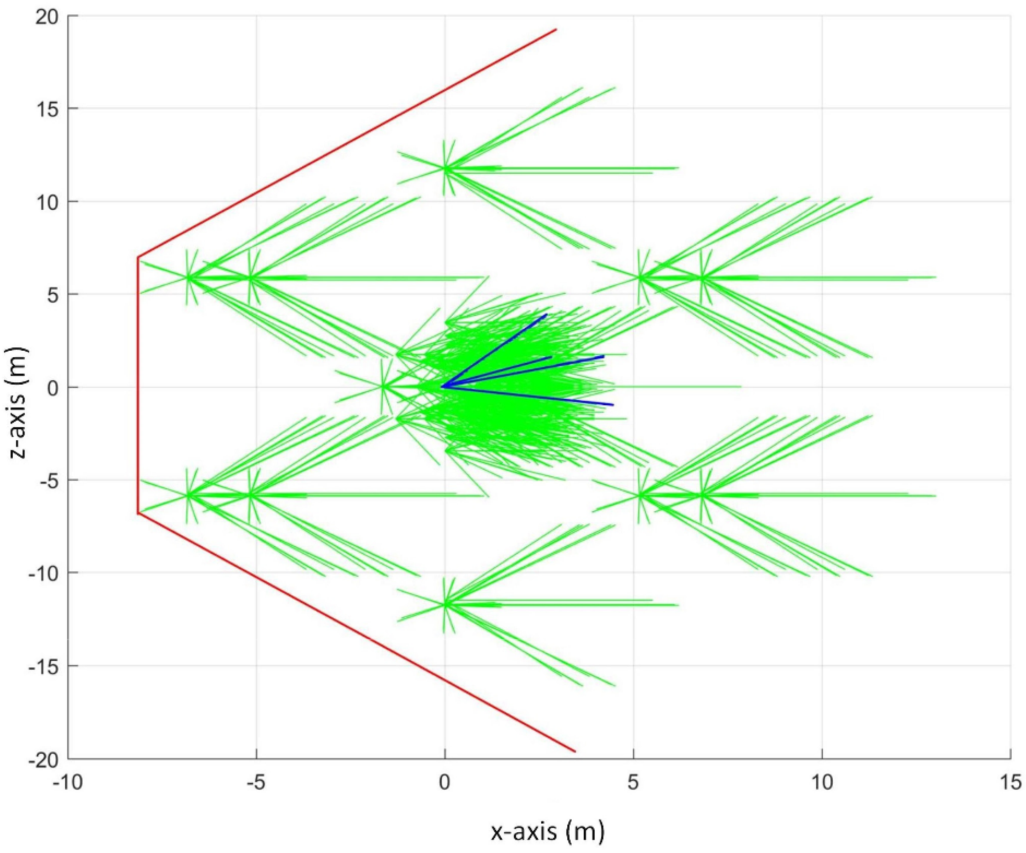


Figure 4.3: Projection of all possible collisions as calculated through the probabilistic Velocity Obstacles approach in [6]. The true collision course(s) in blue are part of the set of probable collision courses in green. A course outside of this set needs to be chosen in order to avoid collision.

Velocity Obstacles	**Artificial Potential Field**
Plans velocity	Plans points in space
Binary fly or no-fly zones	Relative indication safety
Does not define a grid	Defines a grid

Table 4.1: Concise overview of some key differences in Velocity Obstacles and Artificial Potential Field

Mixed-Integer Linear Programming (MILP)
The last method discussed is MILP. MILP is a well-known tool for finding solutions to linear problems. Many different solvers exist, where some provide optimal solutions and others also give non-optimal solutions. Generally, MILP approaches minimise a cost function representing the energy or distance required.

An example of MILP in drone applications is found in [21]. Here, communication constraints and terrain limitations are used in a MILP model. It is found that even under a large number of constraints, an optimal solution is found. Unfortunately, no online and dynamic constraints are used.

Many other approaches exist (e.g. A*, Reinforcement Learning, Floyd-Warshall), however, as the next section will elaborate upon, these often require lots of computational resources and will therefore not be elaborately discussed. For a complete overview, Radmanesh et al. (2018)[38] is recommended literature.

4.2. Discussion

In [38], a larger overview of obstacle avoiding and path-planning algorithms is given. Here, also the computational requirements for several scenario's are discussed. Potential Field methods, just as MSLAP, Genetic Algorithms and MILP are among the fastest methods and moreover, do not exponentially grow when a larger space is considered. Errors, defined as a percentage of the extra distance travelled compared to the optimal solution, are nonzero for APF, MSLAP and Genetic Algorithms. MILP is able to find the optimal solution in relatively short time, however MILP has as requirement that the problem is well-defined, which is not always the case. Furthermore, APF is cited as poor in dynamics and, as stated before, prone to get stuck in local minima.

Considering that Velocity Obstacles plans the velocity and the bearing instead of path points, and that Velocity Obstacles does not require a complete grid definition that may be computationally inefficient, the research will continue using Velocity Obstacles as implementation. It is believed that this fits the research framework very well.

The next steps to be taken are to develop a (variation of) Velocity Obstacles solution and implement it in the drone. Then, the output from the event-camera and the radar needs to be altered to allow Velocity Obstacles to change course when required.

5
Conclusion & Next steps

This report has outlined the state of the art in object avoidance and indicated what three main parts of the algorithm consist of: determining the presence of obstacles, fusing the information and acting upon a possible collision. The end goal of this research is to develop an algorithm that can be installed on a small drone, that allows the drone to efficiently and autonomously detect and avoid obstacles, under harsh environments. This is done in the context of Search & Rescue operations, where a drone autonomously searches an area that may be dark and/or filled with e.g. smoke or dust.

Previous research on this subject at this faculty has focused on a radar solution and an event-camera solution. This research will be continued in the form of fusion of these systems.

An event-camera and a radar together give an all-round vision system that performs relatively well in all circumstances. Both high and low lighting circumstances as well as obscured sight due to smoke or dust fall within the operational conditions of those sensors. They scan complete planes instead of points and are able to detect both bearing and distance.

Methods for object detection are already developed for both systems. For the event-camera, the novel FAITH method has been developed. This novel method uses negative half-planes in order to estimate the focus-of-expansion that is used to estimate Time To Contact and thus detect a collision. The radar-based method detects, clusters and tracks obstacles, in order to avoid them.

For fusion, a wide range of options exist. Fusion algorithms usually are tailored to the exact purpose of the research, standardised methods are usually sub-optimal or neglecting certain aspects. During the design process, one can decide upon a fusion strategy that best fits the exact hardware and software already included. This can be done by e.g. Kalman filtering or a variation thereof, implementation of a neural network or adding an optimisation scheme.

Velocity Obstacles is an avoidance scheme that is very well suited for this research. Velocity Obstacles gives a velocity or bearing command in order to avoid obstacles, which is relatively easy to integrate and is very well suited to the tracking and avoiding of dynamic obstacles.

Integration of these parts will provide a complete solution for object avoidance that is able to work in near all circumstances.

This brings us to the definition of the main research question that will be answered during the thesis:

> How can a radar system and and event-camera system be fused adequately to run on a small autonomous drone in order to achieve a robust object avoidance solution?

This can be divided into the following subquestions:

1. How can a fusion system effectively be implemented to fuse the radar implementation and the FAITH algorithm?
    - (a) What fusion algorithm provides an optimal balance in accuracy, computational time and robustness?
    - (b) How can the different outputs be effectively fused while requiring as little preprocessing as possible in order to achieve low latencies?

2. How can Velocity Obstacles be implemented in the system effectively, minimizing the extra processing required?
    (a) How can noise be efficiently handled and rejected if required?
    (b) How can the output of Velocity Obstacles be translated to a navigation command?

The end product of the thesis will be a drone flying autonomously, using aforementioned system. It is required to avoid obstacles robustly without human intervention.

Bibliography

[1] Mary B. Alatise and Gerhard P. Hancke. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods, 2020. ISSN 21693536.

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

[3] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2015-December, pages 298–304. Institute of Electrical and Electronics Engineers Inc., 12 2015. ISBN 9781479999941. doi: 10.1109/IROS.2015.7353389.

[4] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. ISSN 01628828. doi: 10.1109/TPAMI.1986.4767851.

[5] Yong-Bo Chen, Guan-Chen Luo, Yue-Song Mei, Jian-Qiao Yu, and Xiao-Long Su. International Journal of Systems Science UAV path planning using artificial potential field method updated by optimal control theory UAV path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 47(6):1407–1420, 2016. ISSN 1464-5319. doi: 10.1080/00207721.2014.929191. URL https://www.tandfonline.com/action/journalInformation?journalCode=tsys20http://dx.doi.org/10.1080/00207721.2014.929191.

[6] Myungwhan Choi, Areeya Rubenecia, Taeshik Shon, and Hyo Hyun Choi. Velocity Obstacle Based 3D Collision Avoidance Scheme for Low-Cost Micro UAVs. In *2017 International Conference on Platform Technology and Service (PlatCon)*, 2017. doi: 10.3390/su9071174. URL www.mdpi.com/journal/sustainability.

[7] Cevahir Cigla, Rohan Thakker, and Larry Matthies. Onboard Stereo Vision for Drone Pursuit or Sense and Avoid. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.

[8] Xavier Clady, Charles Clercq, Sio Hoi Ieng, Fouzhan Houseini, Marco Randazzo, Lorenzo Natale, Chiara Bartolozzi, and Ryad Benosman. Asynchronous visual event-based time-to-contact. *Frontiers in Neuroscience*, 2014. ISSN 1662453X. doi: 10.3389/fnins.2014.00009.

[9] Raoul Dinaux, Nikhil Wessendorp, Julien Dupeyroux, and Guido de Croon. FAITH: Fast iterative half-plane focus of expansion estimation using event-based optic flow. 2 2021. URL http://arxiv.org/abs/2102.12823.

[10] Jason Durrie, Tristan Gerritsen, Eric W. Frew, and Stephen Pledgie. Vision-aided inertial navigation on an uncertain map using a particle filter. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4189–4194, 2009. ISBN 9781424427895. doi: 10.1109/ROBOT.2009.5152778.

[11] Gabriele Facciolo, Gabriel Pacianotto, Martin Renaudin, Clement Viard, and Frédéric Guichard. Quantitative measurement of contrast, texture, color, and noise for digital photography of high dynamic range scenes. In *IS and T International Symposium on Electronic Imaging Science and Technology*. Society for Imaging Science and Technology, 2018. doi: 10.2352/ISSN.2470-1173.2018.12.IQSP-170.

[12] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5:9712, 2020. URL `http://robotics.sciencemag.org/`.

[13] Giuseppe Fedele, · Luigi D'alfonso, Francesco Chiaravalloti, · Gaetano D'aquila, Luigi D ' Alfonso, and Gaetano D'aquila. Obstacles Avoidance Based on Switching Potential Functions. *J Intell Robot Syst*, 90:387–405, 2018. doi: 10.1007/s10846-017-0687-2. URL `https://doi.org/10.1007/s10846-017-0687-2`.

[14] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998. ISSN 02783649. doi: 10.1177/027836499801700706.

[15] Sobers Lx Francis, Sreenatha G Anavatti, and Matthew Garratt. Real-Time scene flow estimation and obstacle avoidance under dim-light conditions using Time of Flight sensor. *Journal of Critical Reviews*, 7(14):2020, 2020.

[16] Muhammad Fuad, Trihastuti Agustinah, and Djoko Purwanto. Collision avoidance of multi modal moving objects for mobile robot using hybrid velocity obstacles. *International Journal of Intelligent Engineering and Systems*, 13(3):407–421, 2020. ISSN 21853118. doi: 10.22266/IJIES2020.0630.37.

[17] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based Vision: A Survey. In *EEE Transactions on Pattern Analysis and Machine Intelligence*, 4 2020. doi: 10.1109/TPAMI.2020.3008413. URL `http://arxiv.org/abs/1904.08405http://dx.doi.org/10.1109/TPAMI.2020.3008413`.

[18] Chris Harris and Mike Stephens. A COMBINED CORNER AND EDGE DETECTOR. Technical report, The Plessey Company, 1988.

[19] Berthold K.P. Horn and Brian G. Schunck. Determining Optical Flow. Technical report, MIT, 4 1980.

[20] Minjie Hua, Yibing Nan, and Shiguo Lian. Small Obstacle Avoidance Based on RGB-D Semantic Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[21] Esten Ingar, Grøtli · Tor, Arne Johansen, E I Grøtli, and T A Johansen. Path Planning for UAVs Under Communication Constraints Using SPLAT! and MILP. *J Intell Robot Syst*, 65:265–282, 2012. doi: 10.1007/s10846-011-9619-8. URL `http://www.gurobi.com`.

[22] Khatib. REAL-TIME OBSTACLE AVOIDANCE FOR M.ANIPULATORS AND MQUIEE ROBOTS. Technical report, Stanford University, 1985.

[23] Young K Kwag and Chul H Chung. UAV based Collision Avoidance Radar Sensor. In *2007 IEEE International Geoscience and Remote Sensing Symposium*, 2007.

[24] Hoi-Shan Lin, Jing Xiao, and Zbigniew Michalewicz. Evolutionary Navigator for a Mobile Robot. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994.

[25] Jiahao Lin, Hai Zhu, and Javier Alonso-Mora. Robust Vision-based Obstacle Avoidance for Micro Aerial Vehicles in Dynamic Environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. ISBN 9781728173955. doi: 10.0/Linux-x86{_}64.

[26] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 11 2004. ISSN 09205691. doi: 10.1023/B:VISI.0000029664.99615.94.

[27] Chris Xiaoxuan Lu, Stefano Rosa, Peijun Zhao, Bing Wang, Changhao Chen, John A Stankovic, Niki Trigoni, and Andrew Markham. See Through Smoke: Robust Indoor Mapping with Low-cost mmWave Radar. In *MobiSys '20: The 18th Annual International Conference on Mobile Systems, Applications, and Services*, page 14. ACM, 2020. ISBN 9781450379540. doi: 10.1145/3386901.3388945. URL https://doi.org/10.1145/3386901.3388945.

[28] Bruce D Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI) An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981. URL https://www.researchgate.net/publication/215458777.

[29] Simon Lynen, Markus W. Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to MAV navigation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3923–3929, 2013. ISBN 9781467363587. doi: 10.1109/IROS.2013.6696917.

[30] Alfian Ma, Oyas Wahyunggoro, Adha Imam Cahyadi, and Corresponding Author. Artificial Potential Field Algorithm Implementation for Quadrotor Path Planning. *IJACSA) International Journal of Advanced Computer Science and Applications*, 10(8), 2019. URL www.ijacsa.thesai.org.

[31] Sina Sharif Mansouri, Christoforos Kanellakis, Emil Fresk, Björn Lindqvist, Dariusz Kominiak, Anton Koval, Pantelis Sopasakis, and George Nikolakopoulos. Subterranean MAV Navigation based on Nonlinear MPC with Collision Avoidance Constraints. *IFAC-PapersOnLine*, 53(2):9650–9657, 2020. URL https://youtu.be/-MP4Sn6Q1uo.

[32] Julien Marzat, Sylvain Bertrand, Alexandre Eudes, Martial Sanfourche, and Julien Moras. Reactive MPC for Autonomous MAV Navigation in Indoor Cluttered Environments: Flight Experiments. *IFAC-PapersOnLine*, 50(1):15996–16002, 7 2017. ISSN 24058963. doi: 10.1016/j.ifacol.2017.08.1910.

[33] Kimberly McGuire, Guido De Croon, Christophe De Wagter, Karl Tuyls, and Hilbert Kappen. Efficient Optical Flow and Stereo Vision for Velocity Estimation and Obstacle Avoidance on an Autonomous Pocket Drone. *IEEE Robotics and Automation Letters*, 2(2):1070–1076, 4 2017. ISSN 23773766. doi: 10.1109/LRA.2017.2658940.

[34] Sherif A.S. Mohamed, Mohammad Hashem Haghbayan, Tomi Westerlund, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. A Survey on Odometry for Autonomous Navigation Systems. *IEEE Access*, 7:97466–97486, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2929133.

[35] Erikson Morais, Anselmo Ferreira, Sergio Augusto Cunha, Ricardo M.L. Barros, Anderson Rocha, and Siome Goldenstein. A multiple camera methodology for automatic localization and tracking of futsal players. *Pattern Recognition Letters*, 39(1):21–30, 4 2014. ISSN 01678655. doi: 10.1016/j.patrec.2013.09.007.

[36] Christos Papachristos, Frank Mascarich, and Kostas Alexis. Thermal-Inertial Localization for Autonomous Navigation of Aerial Robots through Obscurants. In *2018 International Conference on Unmanned Aircraft Systems, ICUAS 2018*, pages 394–399. Institute of Electrical and Electronics Engineers Inc., 8 2018. ISBN 9781538613535. doi: 10.1109/ICUAS.2018.8453447.

[37] Oliver Pink, Frank Moosmann, and Alexander Bachmann. Visual features for vehicle localization and ego-motion estimation. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 254–260, 2009. ISBN 9781424435043. doi: 10.1109/IVS.2009.5164287.

[38] Mohammadreza Radmanesh, Manish Kumar, Paul H. Guentert, and Mohammad Sarim. Overview of Path-Planning and Obstacle Avoidance Algorithms for UAVs: A Comparative Study. *Unmanned Systems*, 6(2):95–118, 4 2018. ISSN 23013869. doi: 10.1142/S2301385018400022.

[39] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (1):105–119, 2010. ISSN 01628828. doi: 10.1109/TPAMI.2008.275.

[40] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 2011. ISBN 9781457711015. doi: 10.1109/ICCV.2011.6126544.

[41] P Sakthivel. Integration of Vision and LIDAR for Navigation of Micro Aerial Vehicle. In *Proceedings of IEEE Third International Conference on Multimedia Processing*, page 14, 2020. ISBN 9781665419871. doi: 10.1109/MPCIT51588.2020.9350494/20/{\$}31.00.

[42] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. EVDodgeNet: Deep Dynamic Obstacle Dodging with Event Cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. ISBN 9781728173955. doi: 10.0/Linux-x86{_}64. URL http://prg.cs.umd.edu/.

[43] Arman Savran, Bertrand Higy, Raffaele Tavarone, Leonardo Badino, and Chiara Bartolozzi. Energy and Computation Efficient Audio-Visual Voice Activity Detection Driven by Event-Cameras | IEEE Conference Publication | IEEE Xplore. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 333–340, 2018. doi: 10.1109/FG.2018.00055. URL https://ieeexplore.ieee.org/document/8373848.

[44] A. F. Scannapieco, A. Renga, G. Fasano, and A. Moccia. Ultralight radar for small and micro-UAV navigation. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 333–338. International Society for Photogrammetry and Remote Sensing, 2017. doi: 10.5194/isprs-archives-XLII-2-W6-333-2017. URL https://ui.adsabs.harvard.edu/abs/2017ISPAr42W6..333S/abstract.

[45] Antonio Fulvio Scannapieco, Alfredo Renga, Giancarmine Fasano, and Antonio Moccia. Experimental Analysis of Radar Odometry by Commercial Ultralight Radar Sensor for Miniaturized UAS. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 90(3-4):485–503, 6 2018. ISSN 15730409. doi: 10.1007/s10846-017-0688-1. URL https://doi.org/10.1007/s10846-017-0688-1.

[46] Antonio Fulvio Scannapieco, Maria Daniela Graziano, Giancarmine Fasano, and Alfredo Renga. Improving radar-based mini-UAS navigation in complex environments with outlier rejection. In *AIAA SciTech Forum*, 2019. doi: 10.2514/6.2019-2379. URL http://arc.aiaa.org.

[47] Antonio Fulvio Scannapieco, Alfredo Renga, Maria Daniela Graziano, and Giancarmine Fasano. Preliminary performance assessment of Radar-aided monocular Visual Odometry for small aerial platforms. *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, 2019.

[48] Joseph W. Starr and B. Y. Lattimer. Evaluation of Navigation Sensors in Fire Smoke Environments. *Fire Technology*, 50(6):1459–1481, 10 2014. ISSN 15728099. doi: 10.1007/s10694-013-0356-3.

[49] Fabio Stefanini, Arren Glover, Chetan Singh Thakur, Elisa Donati, Enea Ceolini, Charlotte Frenkel, Sumit Bam Shrestha, Gemma Taverni, Lyes Khacef, and Melika Payvand. Hand-Gesture Recognition Based on EMG and Event-Based Camera Sensor Fusion: A Benchmark in Neuromorphic Computing. *Frontiers in Neuroscience | www.frontiersin.org*, 1:637, 2020. doi: 10.3389/fnins.2020.00637. URL www.frontiersin.org.

[50] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 4 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2793357.

[51] Min Wang and Holger Voos. Safer UAV Piloting: A Robust Sense-and-Avoid Solution for Remotely Piloted Quadrotor UAVs in Complex Environments. In *19th International Conference on Advanced Robotics (ICAR)*, 2019. ISBN 9781728124674. doi: 10.0/Linux-x86{_}64.

[52] Min Wang, Holger Voos, and Daobilige Su. Robust Online Obstacle Detection and Tracking for Collision-free Navigation of Multirotor UAVs in Complex Environments. In *15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018. ISBN 9781538695821. doi: 10.0/Linux-x86{_}64.

[53] Nikhil Wessendorp, Raoul Dinaux, Julien Dupeyroux, and Guido de Croon. Obstacle Avoidance onboard MAVs using a FMCW RADAR. 3 2021. URL http://arxiv.org/abs/2103.02050.

[54] Wonkeun Youn, Hayoon Ko, Hyungsik Choi, Inho Choi, Joong-Hwan Baek, and Hyun Myung. Collision-free Autonomous Navigation of A Small UAV Using Low-cost Sensors in GPS-denied Environments. *International Journal of Control, Automation and Systems*, 19(2):953–968, 2021. doi: 10.1007/s12555-019-0797-7. URL http://dx.http://www.springer.com/12555.

[55] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. 3D registration in dark environments using RGB-D cameras. In *2013 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2013*, 2013. ISBN 9781479921263. doi: 10.1109/DICTA.2013.6691470.

[56] Hang Yu, Fan Zhang, Panfeng Huang, Chen Wang, and Li Yuanhao. Autonomous obstacle avoidance for UAV based on fusion of radar and monocular camera. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5954–5961. Institute of Electrical and Electronics Engineers Inc., 10 2020. ISBN 9781728162126. doi: 10.1109/IROS45743.2020.9341432.

[57] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, Davide Scaramuzza, and Eth Zurich. Semi-Dense 3D Reconstruction with a Stereo Event Camera Multimedia Material. In *European Conference on Computer Vision (ECCV)*, 2018. URL https://youtu.be/Qrnpj2FD1e4.

[58] Lihua Zhu, Xianghong Cheng, and Fuh Gwo Yuan. A 3D collision avoidance strategy for UAV with physical constraints. *Measurement: Journal of the International Measurement Confederation*, 77: 40–49, 1 2016. ISSN 02632241. doi: 10.1016/j.measurement.2015.09.006.

[59] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. MAV navigation through indoor corridors using optical flow. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3361–3368, 2010. ISBN 9781424450381. doi: 10.1109/ROBOT.2010.5509777.