HALMSTAD UNIVERSITY

MASTER THESIS

Drone Detection and Classification using Machine Learning and Sensor Fusion

Fredrik Svanström

Halmstad University, June 3, 2020

SUPERVISORS:
PhD Cristofer Englund
PhD Fernando Alonso-Fernandez
PhD Eren Erdal Aksoy

EXAMINER:
Professor Slawomir Nowaczyk

ABSTRACT

This thesis explores the process of designing an automatic multi-sensor drone detection system using machine learning and sensor fusion. Besides the more common video and audio sensors, the system also includes a thermal infrared camera. The results show that utilizing an infrared sensor is a feasible solution to the drone detection task, and even with slightly lower resolution, the performance is just as good as a video sensor. The detector performance as a function of the sensor-to-target distance is also investigated.

Using sensor fusion, the system is made more robust than the individual sensors. It is observed that when using the proposed sensor fusion approach, the output system results are more stable, and the number of false detections is mitigated.

A video dataset containing 650 annotated infrared and visible videos of drones, birds, airplanes and helicopters is published. Additionally, an audio dataset with the classes drones, helicopters and backgrounds is also published.

CONTENTS

1 INTRODUCTION

Small and remotely controlled unmanned aerial vehicles (UAVs), hereinafter referred to as drones, can be useful and of benefit for society. Examples of their usefulness are to deliver automated external defibrillators [1], to more effectively fight fires [2] and for law enforcement purposes. Moreover, the low cost and ease of operation make drones suitable for entertainment and amusement purposes [3].

Nevertheless, such drones can also be intentionally or unintentionally misused, so that the safety and security of others are affected or threatened. In the worse case, this can cause severe structural damage to an aircraft if it collides mid-air with a consumer-sized drone even when flying at moderate speeds as shown by researchers at the University of Dayton [4].

Due to the rapid development of commercial and recreational drones the research area of drone detection and classification has emerged in the last few years [5] and, as can be seen in Figure 1 below, the Internet search trend for web pages with drone detection related content has been increasing over the last ten years.

Figure 1: Search trend for "drone detection" over the last ten years. From [6].

This thesis will explore the possibilities and limitations of designing and constructing a multi-sensor drone detection and classification system building on state-of-the-art machine learning methods and techniques. This will also include the collection and annotation of the necessary dataset to accomplish the training and evaluation of the system. Three different consumer-grade drones[1] are included in the dataset together with birds, airplanes and helicopters.

1 Hubsan H107D+, DJI Phantom 4 Pro and DJI Flame Wheel F450

1.1 RELATED WORK

To automatically detect flying drones, some form of sensor or sensor system is needed. As shown in [7], the fusion of data from multiple sensors, i.e. using several sensors in combination to achieve more accurate results than derived from single sensors, while compensating for their individual weaknesses, is well-founded when it comes to the drone detection task.

The sensors that may be considered for drone detection and classification tasks, and hence can be found in the related scientific literature, are: RADAR (on several different frequency bands and both active and passive), cameras for the visible spectrum, cameras detecting thermal infrared emission (IR), microphones for the detection of acoustic vibrations, i.e. sound, sensors to detect radio frequency signals to and from the drone and the drone controller (RF), Scanning lasers (Lidar) and, as mentioned in [7] and explored further in [8], even humans. Recently it has also been successfully demonstrated that animals can be trained to fulfil an anti-drone role [9].

Systems for drone detection utilizing one or more of the aforementioned sensors may also be combined with some form of effector to try to bring the flying drone down or, in some other way, take control of it. An effector component, however, is not part of this thesis work.

An introduction to the subject of drone detection is found in [10], published in 2018. It incorporates a comparison of advantages and disadvantages for different drone detection and tracking techniques. Regarding sensor fusion, the authors state that *"For accurate and quick detection/tracking of UAVs, data fusion techniques that can simultaneously use information from multiple types of sensors carry critical importance, and this constitutes an open research area"*.

One of the conclusions of [10] is also that *"Future research directions include developing effective machine learning methods for detecting, classifying, and tracking drones in various scenarios of interest"*. The paper also briefly provides an overview of ways to interdict unauthorized drones.

According to the authors, [5] *"presents a comprehensive review of current literature on drone detection and classification using machine learning with different modalities"*. This paper, published in 2019, also summarizes the results of the related work found in the 50 references into several tables, based on the type of sensor, to better describe the specifications and limitations of the different techniques.

One of the take-home messages is that the authors, on more than one occasion, stresses the current lack of publicly available reference datasets. Furthermore, they also write that *"No study was presented which investigated the classification performance as a function of drone distance not to speak of determining the range using regression models. This can be a very interesting research area in the future"*, and that *"Most of the research in visual drone detection fails to specify the type of the acquisition device, the drone type, the detection range, and the dataset used in their research. These details are key to validate the work and make it comparable with related literature"*.

A weakness of [5] is that it does not include any papers that make use of thermal infrared cameras and, as we will see next, this is something that will be of importance to this thesis.

Also published in 2019, [7] has an exhaustive 178 references. Within these are not only articles specific to drone detection and classification, but also literature regarding the general techniques employed, e.g. useful aspects of sensor fusion and machine learning.

This paper is directly reflected in the further extent of the thesis work with conclusions such as *"For the case of thermal cameras, we present deep learning based methods for general object detection and classification because, to the best of authors' knowledge, there is currently no work tackling the UAV detection and classification problem"*, and *"Deep learning based object detection and classification architectures have been successfully utilized on thermal imagery for generic targets yet not for UAVs. This could be the motivation for researchers to turn their attention to this novel subject"*.

1.1.1 *Thermal infrared sensors*

The work described in [11], from 2017, does not utilize any form of machine learning but instead the drone detection and classification is done by a human in real-time looking at the output video stream.

The sensor used is a low-cost FLIR Lepton 80x60 pixels thermal camera and with this connected to a Raspberry Pi the authors are able to detect three different drone types up to a distance of 100m. One of the things the authors mention is that it is the battery and not the motors, as one may presume, that is the most significant source of heat radiation.

With the background from this paper and the ones above, this thesis will try to extend these finding using a higher resolution sensor (FLIR Boson with 320x256 pixels) in combination with machine learn-

ing methods to build a system that automatically detects, classifies and tracks drones. The IR-camera will also be combined with at least one additional sensor.

The possible advantages of using not only video in the visible range but also a thermal infrared camera are explored to some extent in [12]. In the paper, from 2019, the authors describe how they combine the sensors with deep-learning-based detection and tracking modules. The IR-videos used are stated to have a resolution of 1920x1080, but unfortunately, the sensor is not specified in any further detail.

To detect the drones, a Faster-RCNN is implemented and since *"it is more difficult to acquire enough training data for training the thermal drone detector"*, the authors of [12] also successfully explore the possibility to use a modified Cycle-GAN (General Adversarial Network) to produce synthetic thermal training data.

The paper presents results using curves of the precision and recall of the detector and the authors conclude that *"In both real-world and synthetic thermal datasets, the thermal detector achieves better performance than the visible detector"*. Sadly any details on the distance between the sensor and the target drone is omitted.

Finally, the authors claim that the visible and thermal datasets used are made available as the *"USC drone detection and tracking dataset"*, without giving any link to it in the paper. This dataset is also mentioned in [13], but the link found in that paper is unfortunately not working.

Compared to the results of [12], as summarized above, this thesis will make use of three different drone types instead of just one in the data set. It will also expand the number of target classes of the detector to four and additionally explore the detection performance as a function of sensor-to-target distance.

A thermal infrared camera is also used as one of the sensors in [14], but just as pointed out in [5], this paper, like several others, fails to specify the type, field of view or even the resolution of the used sensors, so even if there are useful tables of the results in that paper, any comparison is unfortunately hard to achieve.

1.1.2 *Sensors in the visible range*

A widespread method to detect drones, as described in recent scientific papers, is to combine a video camera with a detector based on a

convolutional neural network (CNN).

One of these papers, published in 2017, is [15] making an in-depth study of the performance of six different CNN-models for the drone detection task. The paper presents metrics for training time and performance in speed (frames per second) as well as curves for precession and recall. After comparing the six different solutions, the authors conclude that *"considering speed and accuracy trade-offs, YOLOv2*[1] *might be the most appropriate detection model in our system"*.

As it turns out, when studying the subject of drone detection in images in the visible range, the YOLOv2-architecture [16] is very common. In [13], [17], [18] and [19] this is the preferred and implemented CNN-model. A lightweight version of the more recent YOLOv3 is utilized in [20].

The success of the YOLOv2 motivates the use of it in this thesis. This choice will also make it possible to compare the results to the papers mentioned above. To be able to match the results from the thermal infrared sensor, the YOLOv2 is also used in that part.

The fact that the two primary sensors (IR and visible) of the drone detector system in this thesis are placed on a pan/tilt platform also arises the need for a wide-angle sensor to steer the sensors in directions where suspicious objects appear. In [20] a static camera with 110° field of view (FoV) is used to align a rotating narrow-field camera (the images from this is analysed with the YOLOv3-model as pointed out above). To find the objects to be investigated further by the narrow-field camera, the video stream from the wide-angle one is analysed by means of a Gaussian Mixture Model (GMM) foreground detector in one of the setups evaluated in [20].

The GMM foreground detector will also be implemented in this thesis for the wide-angle sensor. However, the one used in this case has an even wider FoV (180°). Just as pointed out in [20], this setup is prone to produce false alarms in some situations, but as described in Section 3.1.3, this can be mitigated by tuning the associated detection and tracking algorithms.

Of the papers found using versions of the YOLO-architecture for drone detection, [17] has the most output target classes with three (drone, airplane and helicopter) followed by [19] with two classes (drone and bird). None of the YOLO-papers report the detection accuracy as a function of the sensor-to-target distance.

1 You Only Look Once in its second version

1.1.3 *Acoustic sensors*

Numerous papers are also exploring the possibility to detect drones using acoustic sensors. Some of these papers, like [21], [22] and [23] utilize the Fast Fourier Transform (FFT) to extract features from the audio signals.

However, it is observed that the Mel Frequency Cepstrum Coefficients (MFCC) seems to be the most common feature extraction technique, as used in [24], [25], [26] and [27]. The MFCC is *"a non-linear mapping of the original frequency according to the auditory mechanism of the human ear. It is the most commonly used audio feature in current recognition tasks"* [25].

When comparing three different classification models for the drone detection task, the authors of [26] comes to the conclusion that the Long Short-Term Memory achieves the best performance and F1-score. In that paper, the classification is binary (drone or background). This thesis expands the range of output classes of [26] by adding a helicopter class.

The maximum acoustic detection range reported in the reviewed paper is 290 m [28], this when using a 120-element microphone array and a DJI Phantom 2 drone. It is worth noting that the the DJI Flame Wheel F450, one of the drones used in this thesis, is detected at distance of 160 m by the microphone array.

1.1.4 *RADAR*

Since RADAR is the most common technology to detect flying objects, it is not far fetched to apply this also to the detection of drones. Doing this with a RADAR-system designed to detect aircraft is however not as straight forward as it might seem. This since those systems often have features to reduce unwanted echoes from small, slow and low-flying objects, which is precisely what characterise the drones in question.

The small Radar Cross Sections (RCS) of medium-sized consumer drones are well described in [29], and from [30] we have that the RCS of the DJI Flame Wheel F450 is -17 dBsm (0.02 m^2). The results of [31] points out that flying birds have similar RCS, something that can lead to false targets when detecting drones.

The F450 drone is also used in [32], where the micro-doppler characteristics of drones are investigated. These are typically echoes from the moving blades of the propellers and can be detected on top of the

bulk motion doppler signal of the drone. Since the propellers are generally made from plastic materials, the RCS of these parts are even smaller, and in [29] it is stated that the echoes from the propellers are 20 to 25 dB weaker than that of the drone body itself. Nevertheless, papers like [33], [34] and [35] all accompany [32] in exploring the possibility to classify drones using the micro-doppler signature.

1.1.5 *Other drone detection techniques*

Very few drones are autonomous in the flight phase. Generally, they are controlled by means of a manned ground equipment, and often the drones themself also send out information on some radio frequency (RF). The three drones used in this thesis are all controlled in real-time. The information sent out by the drones can range from just simple telemetry data such as battery level (DJI Flame wheel F450), a live video stream (Hubsan H107D+), to both a video stream and extensive position and status information (DJI Phantom 4 Pro).

Utilizing the RF-fingerprint is described in [36], and in [37] a CNN is used to process the information from an antenna array so that the direction to the drone controller can be calculated within a few degrees. In [38], signals from 15 different drone controllers are classified with an accuracy of 98.13% using only three RF-features[1] in a K-Nearest Neighbour (KNN) classifier.

The use of LiDAR (Light Detection And Ranging) and LADAR (LAser Detection And Ranging) in combination with background subtraction and a nearest neighbour classifier is shown to successfully detect drones up to a distance of 2 km in [39].

1.1.6 *Sensor fusion*

As shown in [7] the fusion of data from multiple sensors, i.e. using several sensors in combination to achieve more accurate results than derived from single sensors, while compensating for their individual weaknesses, is well-founded when it comes to the drone detection task.

As mentioned above, the benefits of sensor fusion are also pointed out in [10], and in [40] the authors describe that *"Audio detection, video detection, and RF detection are conducted in parallel. We consider that missed detection of the intruding drone will bring more security threats than false alarm. Therefore, we use logical or operation to fuse the results of the above three detectors and obtain the final decision"*.

1 Shape factor, kurtosis (the tailedness of the energy curve), and the variance

Furthermore under the section Challenges and Open Research Issues they conclude that *"Although we have realized basic functions for drone surveillance in our anti-drone system, new challenges have been raised, and there are still some open research issues to be addressed: Heterogeneous Information Fusion: The result of drone detection should be more than a simple combination of the results separately obtained by audio, video, and RF surveillance, which will induce great information loss. It is of great significance to develop reliable techniques for the fusion of audio, video, and RF information from the aspect of feature fusion and decision fusion"*.

Without any further specifications of the sensors used besides that they are visible, thermal and 2D-radar [14] presents promising results from experiments using a multilayer perceptron (MLP) to perform sensor fusion in a drone detector/classifier with just one output class.

Just as in [7] this thesis also considers early and late sensor fusion and differentiate these two principles based on if the sensor data is fused before or after the detection element.

1.1.7 *Drone detection datasets*

As mentioned above, [5] points out the lack of a publicly available dataset. This is also the case in [7] where the authors highlight this especially in the case of thermal infrared cameras and conclude that *"the creation of a dataset for UAV detection and classification based on thermal images without an increased budget might be out of reach for many universities and research centers"*. Furthermore, they also state that *"there is a strong need for the collection of a real-world UAV audio dataset that could serve as a benchmark for researchers to develop their algorithms"*.

In the references of [7], there are two useful links to datasets for visible video detectors. One of these is [41], where 500 annotated drone images can be found. The other link leads to the dataset [42] of the Drone-vs-Bird challenge held by the Horizon2020 SafeShore project consortium. However, the dataset is only available upon request and with restrictions to the usage and sharing of the data. The Drone-vs-Bird challenge is also mentioned in [18], [19] and by the winning team of the challenge in 2017 [43]. The results from the 2019 Drone-vs-Bird challenge is presented in [44].

The dataset used in [20] is not publicly available due to confidentiality. Since the work of that paper was founded by the French Ministry of Defence, one can presume that the dataset, in one way or another, is a property of the French Government or the French Armed Forces.

1.2 THESIS SCOPE

The scope of this thesis is twofold: First, to explore the possibilities and limitations of designing and constructing an automatic multi-sensor drone detection system building on state-of-the-art methods and techniques. In some cases, these methods will also be extended from conclusions and recommendations found in the related work. This extension is achieved by the use of a different sensor combination, exploring the performance based on the sensor-to-target distance, increasing the number of target classes, and incorporating a novel sensor fusion method for the drone detection task.

The second object of the thesis is to collect, compose and publish a drone detection dataset. This dataset should contain data from as many of the sensors used as possible. The data and the associated annotations should be in standard formats, so that the data can be imported, reviewed and even edited by others.

Thus, this thesis can be seen as incorporating all phases of designing an embedded and intelligent system. From the initial literature study, the subsequent assessment and procurement of suitable sensors and hardware components, the design and 3D-printing of parts that are not available, the programming and training of the system, and finally evaluating it and reporting the results. In this case, the thesis also includes the collection and composition of the required dataset.

A key doctrine and principle of this thesis will be that to effectively detect the sought after drones the system must also, by all possible means, detect and keep track of other flying objects that are likely to be mistaken for a drone. This follows directly from the conclusions of [18] and [19].

1.2.1 *Thesis system specifications and limitations*

The system designed in this thesis shall:

- Utilize thermal infrared, visible video and acoustic sensors to detect and classify drones
- Incorporate an ADS-B[1] receiver, so that objects flying with an active transponder can be tracked
- Have four output target classes: airplane[2], bird, drone and helicopter[3]

1 Automatic Dependent Surveillance–Broadcast
2 Manned fixed-wing aircraft
3 Manned rotary-wing aircraft

- Have a time constraint on the pan/tilt servo control loop so that it can track an object flying in front of the system at normal speeds
- Be close to real-time in the sense that the time from possible classification to output is no more than one second
- Use components of standard type, e.g. not of military-grade, so that it can easily be compared to other detection systems
- Be a stand-alone solution in the sense that it will work without an active internet connection

Moreover, the thesis shall include:

- An evaluation of detection and classification performance as a function of the sensor-to-target distance
- An assessment of the possibilities to use early or late sensor fusion
- The composition and publication of a multi-class, multi-sensor dataset

There are also some limitations to the scope of this thesis and some of these are that:

- The system will not be completely weather-proof
- It will not have any effector part, i.e. the system will not try to bring the flying drone down or in any other way take control of it
- In the work LIDAR, RF, humans or animals will not be considered as sensors to be used in the system setup
- Fixed-wing drones or drones of helicopter type will not be considered, neither very rare aircraft types such as lighter-than-air balloons etc.

A description of the methods and materials used in this thesis is found below in Chapter 2. The results are found in Chapter 3, followed by an interpretation of the findings in Chapter 4. Finally, in Chapter 5 the main points are summarised together with research questions raised by the results.

2

METHODS AND MATERIALS

This chapter describes the proposed methodology and the automatic drone detection system is also outlined. First, on a system-level and thereafter in deeper detail, both regarding hardware components, how they are connected to the main computational resource, and the involved software.

As pointed out in [5] *"Most of the research in visual drone detection fails to specify the type of the acquisition device, the drone type, the detection range, and the dataset used in their research. These details are key to validate the work and make it comparable with related literature"*. Hence the hardware used is initially specified in detail.

After that follows a description of the software running in the drone detection system when it is active, including the graphical user interface. The support software used for such tasks as to collect data, to set up the detectors and to evaluate the performance after training is also described. For all parts that include a machine learning feature, the training process is also presented.

Finally, the methods used for the composition of the drone detection dataset are described, including the division of the dataset according to the sensor type, target class and sensor-to-target distance bin.

Since Matlab (version R2019b) is used as the primary development environment, the thesis text complies with the Matlab terminology so that all programs are called scripts and what is commonly known as a thread is denoted as a worker. To indicate the Matlab functions used, these are printed in a different font, e.g. `trainYOLOv2ObjectDetector`. In addition to the basic Matlab environment, the following toolboxes are also utilized:

- Image acquisition, including package for video interface
- Image processing
- Computer vision
- Statistics and machine learning
- Deep learning, including model for MobileNetv2 network
- Audio
- Communication
- Parallel computing

2.1 PROPOSED METHODOLOGY

An efficient drone detection system must have the capability to both cover a large volume of airspace, and at the same time have the resolution enough to distinct the drone from other objects. Combining wide and narrow field of view-cameras is one way to accomplish this [20]. Another way, shown in [17], is to use an array of high-resolution cameras. Since this thesis incorporates only one infrared sensor, with a fixed field of view, there is no possibility to have neither a wide-angle infrared sensor or an array of such sensors. The proposed way to achieve the desired volume coverage with the IR-sensor is to have it on a moving platform. This platform can then either have objects assigned to it or search by itself, at moments in time when the sensors on it are not busy detecting and classifying objects.

To be able to react to moving objects and also to have the ability to track those, the combined time-constraints of the detection cycle and the control loop of the moving platform means that the system must work in close to real-time. Hence, all the detection and classification processes must be done efficiently and with as little delay as possible. The feedback loop of the moving platform must run at sub-second speed.

To be able to put together such a system, involving both several sensors and mechanical part makes the importance of choosing the right methods critical. While reviewing the theoretical foundations of machine learning techniques, and at the same time look at the practical results reported it is clear that such methods can be found.

This thesis will utilize three machine learning techniques. Two of these are supervised, and one is unsupervised. First, it has two detectors and classifiers build around the YOLOv2-architecture [16]. Both these can be seen as a form of transfer learning, since the feature extraction parts build on an already-trained network. The YOLOv2-networks are trained using a dataset with an annotated ground truth. Hereinafter, in this theses, by detection we mean the process of both detecting the area where the object is in an image and the classification that assigns it to one of the labels or classes found in the ground truth of the training dataset.

Furthermore, the second supervised machine learning technique used is a classification function building on the LSTM-architecture [45]. Since this is not detecting the position of an object, but only assigns the input to the network to one of three label classes, we will refer to this process just as classification.

Finally, an unsupervised technique is also used in the drone detection system. This is based on the segmentation of images using GMM background subtraction [46]. By employing this image segmentation, the system is able to find moving objects in the field of view of the wide-angle camera. This will also be referred to as detection even if it does not involve a classification of the object seen as in the YOLOv2-case. Since we are not providing the detector with any ground truth of what is foreground and background, this process falls under the category of unsupervised machine learning techniques.

One of the conclusions found in the related work is that to be able to detect the drones with high enough efficiency, the system must also recognize and keep track of other flying objects that are likely to be mistaken for a drone. For some of these drone-like objects, this is indigenous hard, e.g. birds. For others, it is technically possible since some of them announce their presence and location via radio. The technical system for this is the ADS-B, over which most aircraft regularly transmit messages with varied content.

Combining the data from several sensors under the time constraints described above must be kept simple and streamlined. This together with the fact that very few papers are exploring sensor fusion techniques is the motivation to have a system where the inclusion and weights of the sensors can be altered at runtime to find a feasible setting.

To be able to accomplish the necessary training of the detectors and classifiers, a dataset is also collected. The annotation of this is done so that the dataset can be inspected, edited and used by others. The fact that the datasets for the thermal infrared and the visible video sensors are collected under the same conditions, and using the same targets ensures that a comparison between the two sensor types is well-founded.

2.2 SYSTEM ARCHITECTURE

The system utilizes a thermal infrared camera (IRcam) and a video camera (Vcam) as the primary electro-optical sensors for the detection and classification. To keep track of cooperative aircraft[1] in the airspace, ADS-B information is made available in the system. Audio data is used to detect if a drone or a helicopter is in the vicinity of the system using their respective distinctive sounds when they are airborne. All computations in the system are made in a standard laptop. This is also used to present the detection results to the user. The sys-

1 Cooperative aircraft are here defined as being equipped with an equipment that broadcasts the aircraft's position, velocity vectors and identification information

tem architecture including the principal layout of the graphical user interface (GUI) is shown in Figure 2.

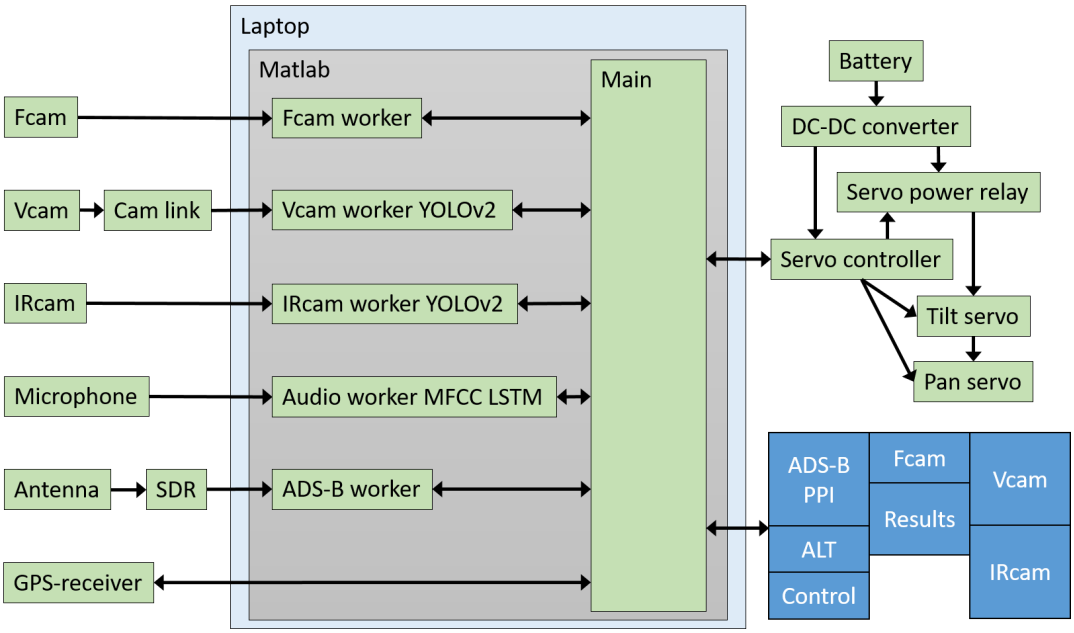


Figure 2: The system architecture with hardware and software components.

Due to the limited field of view of the IRcam, it is placed together with the Vcam on a pan/tilt platform that can be aimed in specific directions guided by information gathered by a fisheye lens camera (Fcam) covering 180° horizontally and 90° vertically. If nothing is detected and tracked by the Fcam, the platform can be set to move in two different search patterns to scan the skies around the system. Figure 3 shows the main parts of the system.

Figure 3: The main parts of the drone detection system. On the lower left the microphone and above that the fisheye lens camera. On the pan and tilt platform in the middle are the IR- and video cameras. The holder for the servo controller and power relay boards is placed behind the pan servo inside the aluminium mounting channel.

2.3 HARDWARE

To have a stable base for the system, all hardware components, except the laptop, are mounted on a standard surveyors tripod. This solution also facilitates the deployment of the system outdoors, as shown in Figure 4. Due to the nature of the system, it must also easily be transported to and from any deployment. Hence a transport solution is available where the system can be disassembled into a few large parts and placed in a transport box.

Figure 4: The system deployed just north of the runway at Halmstad airport.

Just as in the design of any embedded system, the power consumption of the system components have also been taken into consideration, not to overload the laptop's built-in ports and the added USB-hub. For this purpose, the current drawn by all components have been measured using a Ruideng AT34 tester.

2.3.1 *Thermal infrared camera*

The thermal infrared camera used is a FLIR Breach PTQ-136 using the Boson 320x256 pixels detector. The field of view of the IR-camera is 24° horizontally and 19° vertically. Figure 5 shows an image taken from the IRcam video stream.

Figure 5: The F450 drone flying at a distance of 3 m in front on the IRcam.

Notably, the Boson sensor of the FLIR Breach has a higher resolution than the one used in [11] where a FLIR Lepton sensor with 80x60 pixels was used. In that paper, the authors were able to detect three different drone types up to a distance of 100m, however this detection was done manually by a person looking at the live video stream and not, as in this thesis, by means of a trained embedded and intelligent system.

The IRcam has two output formats, a raw 320x256 pixels format (Y16 with 16-bit greyscale) and an interpolated 640x512 pixels image in the I420 format (12 bits per pixel). For the interpolated image format, the colour palette can be changed, and several other image

processing features are also available. In the system, the raw format is used to avoid the extra overlaid text information of the interpolated image. This choice also grants a better control of the image processing operations as they are implemented in Matlab instead.

The output from the IRcam is sent to the laptop via a USB-C port at a rate of 60 frames per second (FPS). The IRcam is also powered via the USB connection.

2.3.2 *Video camera*

A Sony HDR-CX405 video camera is used to record the scenes in the visible range of the spectrum. The output of this camera is an HDMI-signal and hence a frame grabber in the form of an Elgato Cam Link 4K is used to feed the laptop with a 1280x720 video stream in YUY2-format (16 bits per pixel) at 50 FPS.

The Vcam has an adjustable zoom lens, and with this, the field of view can be set to be both wider or narrower than that described above of the IRcam. The Vcam is set to have about the same field of view as the IRcam.

2.3.3 *Fisheye lens camera*

To monitor a larger part of the surroundings of the system an ELP 8 megapixel 180° fisheye lens camera is also used. This outputs a 1024x768 video stream in Mjpg-format at 30 FPS via a USB connector.

2.3.4 *Microphone*

To be able to capture the distinct sound that drones emit when flying a Boya BY-MM1 mini cardioid directional microphone is also connected to the laptop.

2.3.5 *RADAR module*

As described in the specification of the system architecture in Section 2.2 a RADAR module is not incorporated final design. However, since one was available, it was assessed to be included and hence also described in this section. In Section 3.1.5 the performance of the RADAR module is described, and the exclusion of it is motivated by the short practical detection range. Figure 6 shows the radar module and some of its features.

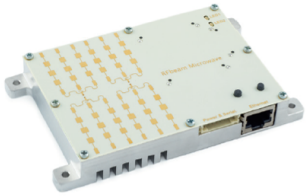
RFbeam MIcrowave GmbH

product info

K-MD2

Engineering Sample

radar transceiver
with integrated signal processing



Features

- 24 GHz FMCW radar with digital signal processing
- Angle of arrival in azimuth/elevation
- Serial target list output
- Detection distance: 100 m for persons/200 m for cars
- Distance range: 0–250 m, 1 m resolution
- Speed range: ± 130 km/h, 1 km/h resolution
- Angle range: ± 9.1° (elevation) ± 16.4° (azimuth), 0.1° resolution
- Compact size: 120 × 72 × 15 mm

Figure 6: The radar module K-MD2. From [47].

Furthermore, [47] describes the module as *"a high-end 3D radar transceiver with three receiving channels and a low phase noise, PLL[1] controlled transmitter. The target information from the three receive antennas is digitized and the high speed digital signal processing performs range and doppler FFT's with an update rate of 20 measurements per second. Using the serial interface, many operating parameters such as frequency, bandwidth and repetition rate can be adjusted. Results are available in target list format as well as in raw range-doppler matrices. Ethernet and a serial communication interfaces are included"*.

Interestingly, the K-MD2 radar module is also found to be used in another research project connected to drones. In [48] it is utilized, not to detect drones, but instead mounted on board one as part of the navigation aids in GNSS[2] denied environments.

2.3.6 *ADS-B receiver*

To track aircraft equipped with transponders, an ADS-B receiver is also used. This consists of an antenna and a NooElec Nano 2+ Software Defined Radio receiver (SDR). This is tuned to 1090 MHz so that the identification and positional data sent out as a part of the 1 Hz squitter message can be decoded and displayed. The Nano 2+ SDR receiver is connected to the laptop using USB.

1 Phase-locked loop
2 Global Navigation Satellite System

2.3.7 *GPS receiver*

To present the decoded ADS-B data in a correct way the system is also equipped with a G-STAR IV BU-353S4 GPS receiver connected via USB. The receiver outputs messages following the National Marine Electronics Association (NMEA) format standard.

2.3.8 *Pan/tilt platform including servo controller*

To be able to detect targets in a wider field of view than just 24° horizontally and 19° vertically the IR- and video cameras are mounted on a pan/tilt platform. This is the Servocity DDT-560H direct drive tilt platform together with the DDP-125 Pan assembly, also from Servocity. To achieve the pan/tilt motion two Hitec HS-7955TG servos are used.

A Pololu Mini Maestro 12-Channel USB servo controller is included so that the respective position of the servos can be controlled from the laptop. Since the servos have shown a tendency to vibrate when holding the platform in specific directions a third channel of the servo controller is also used to give the possibility to switch on and off the power to the servos using a small optoisolated relay board.

To supply the servos with the necessary voltage and power, both a net adapter and a DC-DC converter are available. The DC-DC solution is used when the system is deployed outdoors and, for simplicity, it uses the same battery type as one of the available drones.

Some other parts from Actobotics are also used in the mounting of the system and the following has been designed and 3D-printed: adapters for the IR-, video- and fisheye lens cameras, a radar module mounting plate and a case for the servo controller and power relay boards.

A lighter version of the IR- and video camera mounting, without the pan/tilt platform, has also been prepared. This is used on a light camera tripod when collecting data and hence simplifies both transporting the setup and also the possibility for a person to manually set the direction of it. The data collection setup is shown in Figure 7.

Figure 7: The data collection setup.

2.3.9 *Laptop*

The computational part of the system is done on a Dell Latitude 5401 laptop. This is equipped with an Intel i7-9850H CPU and an Nvidia MX150 GPU. The laptop is connected to all of the above-mentioned sensors and the servo controller using the built-in ports and an additional USB-hub, as shown in Figure 4.

2.4 SOFTWARE

The software used in the thesis can be divided into two parts. First, the software running in the system when it is deployed, as depicted in Figure 2. Additionally, there is also a set of support software used for tasks such as to form the training data sets and to train the system.

2.4.1 *System software*

The software running when the drone detector is deployed consists of the main script and five "workers", as shown in Figure 2. These are threads running in parallel, and having such is enabled by the Matlab parallel computing toolbox. The transfer of messages between the main program and the workers is done using pollable data queues. Only commands and results are sent and hence the displays of the video streams are part of the respective worker.

The Fcam worker utilizes a foreground/background detector together with a multi-object Kalman filter tracker, and after calculating the position of the best-tracked target[1], it sends the azimuth and elevation angles to the main program. Based on this, the main program can then control the pan/tilt platform servos via the servo controller, so that the moving object can be analysed further by the IR- and video cameras.

The IRcam and Vcam workers are similar in their basic structure and both import and run a trained YOLOv2 detector and classifier. The information sent to the main script is the class of the detected target, the confidence[2], and the horizontal and vertical offsets in degrees from the centre point of the image. The latter information is used by the main script to calculate servo commands when an object is being tracked by the system.

The Audio worker sends information about the class and confidence to the main script. Unlike the others, the ADS-B worker has two output queues. One is consisting of current tracks and the other of the history tracks. This is done so that the presentation clearly shows the heading and altitude changes of the targets.

All of the above workers also send a confirmation of the command from the main script to run the detector/classifier or to be idle. The number of frames per second currently processed is also sent to the main script.

Looking further into the different output classes or labels that the main code can receive from the workers, as shown in Table 1, it is clear that not all sensors can output all the target classes used in this thesis. Furthermore, the audio worker has an additional background class and the ADS-B will output the NoData-class if the vehicle category

1 Defined to be the one with the longest track history

2 This is the class-specific confidence score for each bounding box, i.e. the product of the conditional class probability and the individual bounding box prediction confidence, as defined in 49

field of the received message is empty (this is not a mandatory field as we will come back to in Section 2.4.1.6).

Table 1: The output classes of the sensors, and their corresponding class colours.

IRcam	AIRPLANE	BIRD	DRONE	HELICOPTER
Vcam	AIRPLANE	BIRD	DRONE	HELICOPTER
Audio			DRONE	HELICOPTER
ADS-B	AIRPLANE		DRONE	HELICOPTER

BACKGROUND
NO DATA

The main script and all the workers are set up so that they can be run independently of each other in a stand-alone mode. The reason for this is to facilitate the development and tuning without having to set up the whole physical system.

2.4.1.1 *Main script*

The main script is the core of the system software, as shown in Figure 2. Besides initially starting the five workers (threads) and setting up the queues to communicate with these, the main script also sends commands to and reads data from the servo controller and the GPS-receiver, respectively. After the start-up sequence, the script goes into a loop that runs until the program is stopped by the user via the graphical user interface (GUI).

Updating the GUI and reading user inputs are the most frequent tasks done on every iteration of the loop. At regular intervals, the main script also interacts with the workers and the servo controller. Servo positions are read and queues are polled ten times a second. Within this part, the system results, i.e. the system output label and confidence, are also calculated using the most recent results from the workers. Furthermore, at a rate of 5 Hz new commands are sent to the servo controller for execution. Every two seconds the ADS-B plot is updated. Having different intervals for various tasks makes the script more efficient since, for example, an aircraft sends out its position via ADS-B every second, and hence updating the ADS-B plots to often would only be a waste of computational resources. The main script pseudocode is shown in Algorithm 1.

Algorithm 1 Main script pseudocode

```
1: Start of main
2: connect to the servo controller
3: start a parallel pool with five workers
4: setup queues for data from the workers
5: start the workers
6: poll the queues from the workers to get the queues to the workers
7: set up and start the GUI
8: while GUI is not closed do
9:     if time since last execution > 0.1s then
10:        read servo positions
11:        communicate with workers
12:        calculate system output label and confidence
13:    end if
14:    if time since last execution > 0.2s then
15:        send control signals to the servos
16:        update the servo controller indicator
17:    end if
18:    if time since last execution > 2s then
19:        update the ADS-B plots
20:    end if
21:    read and execute user commands
22:    update status and control panels
23: end while
24: stop workers
25: shut down the parallel pool
26: turn off the power to the servos
27: disconnect from the servo controller
28: End of main
```

The use of workers will also allow the different detectors to run asynchronously, i.e. handling as many frames per second as possible without any inter-sensor delays and waiting time.

The sensor fusion is an essential part of the main code and every time the main script polls the worker's queues it puts the results in a 4x4 matrix, organized so that each class is a column and each sensor is a row. The value depends not only on the class label and the confidence but also on the setting of which sensors to include and the weight of the specific sensor, i.e. how much we trust it at the moment.

A new 1x4 array is then formed by taking the column-wise sums of the result matrix. This array is in turn placed as a new row in a 10x4 first-in-first-out time-smoothing matrix. Since we have close to ten FPS from the workers, this will be the results of approximately the last second. Once again the 10 columns are summarized into a

1x4 array, and the column with the highest value will be the output system class.

The system output confidence is calculated by normalizing the value found to be highest, over the number of sensors included at the moment. An additional condition before presenting the system result is also that the number of sensors that detects any object has to fulfil the GUI-setting, as shown in Figure 8.

Figure 8: Two examples of sensor fusion results.

2.4.1.2 *IRcam worker*

The start-up process of the IRcam worker is to first connect to the queue it gets from the main script in the function declaration. Since it is only possible for the main script to set up a queue from the worker, the IRcam worker uses that queue to send a queue to the main script. In this way, a bidirectional communication is established and hence the worker can both send information about the detections, and re-

ceive commands, such as when to run the detector and when to go idle.

If no queue is given to the IRcam worker function when it is started, the function assumes that it is running in a stand-alone mode and acts accordingly.

The next actions taken at start-up is loading the detector and connecting to the IR-camera. The video stream is thereafter started and set to be continuous using the `triggerconfig`-command, so that the worker can use the `getsnaphot`-function to read an image at any time within the loop it goes into when running.

The IR camera has several mounting points and when attached to the 3D-printed adapter, on which it and the video camera are mounted, the raw image output stream is upside down. Hence, the first image processing operation is `imrotate` followed by `imflatfield` with $\sigma = 30$ [1] and then `imadjust` to increase the contrast of the image.

Next, the input image is processed by the YOLOv2-detector, with a given detection threshold and the execution environment set to GPU. The output from the detector consists of an array of class labels, confidence scores and bounding boxes for all objects that have been detected and classified. The detector output may be no data at all, or just as well, data for several detected objects. In this implementation, only the detection with the highest confidence score is sent to the main script.

The image from the IR camera is just 320x256 pixels, so in order to present the result in the GUI using a window similar in size to the video camera output, the image is resized to 640x512 pixels. Then the bounding box together with the class label and confidence score are inserted in the image. To clearly indicate the detected class the inserted annotation uses the same colour scheme as presented in Table 1.

To give information about the current state of the detector and the performance in terms of frames per second (FPS), this information is also inserted in the top left corner of the image. The current frames per second processed by the worker is also sent to the main script together with the detection results.

The YOLOv2 detector is set up and trained using one of the scripts belonging to the support software. The YOLOv2 is formed by mod-

1 The flat-field correction uses Gaussian smoothing with a standard deviation of σ to approximate the shading component of the input image

ifying a pretrained MobileNetv2 following [50], so that the first 12 layers, out of 53, are used for feature extraction. The input layer is updated[1] so that is has the size of 256x256x3. The choice of this particular size is motivated by the fact that the image is always resized to fit the input layer. This resize operation will always make the image smaller or keep it at its original size, so using 256 causes the least information loss since it is the height of the IR images.

Six detection layers and three final layers are also added to the network. Besides setting the number of output classes of the final layers the anchor boxes used are also specified.

To estimate the numbers of anchor boxes to use and the size of these the training data is processed using the `estimateAnchorBoxes`-function. This function uses a k-means clustering algorithm to find suitable anchor boxes sizes given the number of boxes to be used. A script from the support software loops through the number of anchor boxes from one to nine to provide a plot over the mean intersection over union (IoU)[2] as a function of the number of boxes as shown in Figure 9.

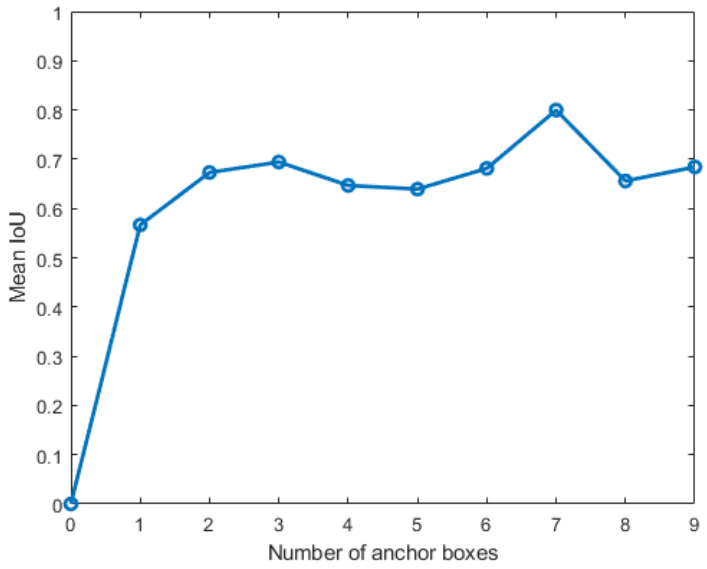


Figure 9: Plot used to assess the number of anchor boxes to be implemented in the IRcam worker YOLOv2 detector.

When choosing the number of anchor boxes to use the trade-off to consider is that a high IoU ensures that the anchor boxes overlap well with the bounding boxes of the training data but, on the other

1 The original input layer of the MobileNetv2 is 224x224x3 in size.

2 This is the intersection area of two bounding boxes divided by the union of their respective areas. This provides a scale invariant measurement of the bounding boxes matching.

hand, using more anchor boxes will also increase the computational cost and may lead to overfitting. After assessing the plot, the number of anchor boxes is chosen to be three and the sizes of these, with the scaling factor of 0.8 in width to match the downsize of the input layer from 320 to 256 pixels, are taken from the output of the `estimateAnchorBoxes`-function.

The detector is trained using data picked from the available dataset after the evaluation data have been selected and put aside, as described in Section 2.6. The training data for the IRcam YOLOv2 detector consists of 120 video clips, each one just over 10 seconds and evenly distributed among all classes and all distance bins, making the total number of annotated images in the training set 37428. The detector is trained for five epochs[1] using the stochastic gradient descent with momentum (SGDM) optimizer and an initial learning rate of 0.001.

Using a computer with an Nvidia GeForce RTX2070 8GB GPU, the time for five epochs is 2 h 15 min. The training time of the IRcam worker YOLOv2-detector for one epoch using different hardware is shown in Table 2.

Table 2: Training time for one epoch of the IR dataset.

Computing hardware	Time for one epoch
GPU GeForce RTX2070 8GB	39 min
GPU GeForce MX150 2GB	6 h 58 min
CPU i7 9850H 16GB RAM	8 h 53 min

Since the training data is specified as a table and not using the datastore-format the `trainYOLOv2ObjectDetector`-function performs preprocessing augmentation automatically. The augmentation implemented is reflection, scaling, and changing brightness, hue, saturation and contrast.

2.4.1.3 *Vcam worker*

The Vcam worker is very similar to the IRcam worker, with some exceptions. The input image from the Vcam is 1280x720 pixels, and directly after the `getsnapshot`-function, it is resized to 640x512 pixels. This is the only image processing operation done to the visible video image. The input layer of the YOLOv2 detector has a size of 416x416x3. The increased size, compared to the detector in the IRcam worker, is directly reflected in the FPS performance of the detector.

1 An epoch is a pass through the whole dataset

Due to the increased image size used to train the detector, the training time is also extended compared to the IR case. When using a computer with an Nvidia GeForce RTX2070 8GB GPU, the time for one epoch is 2 h 25 min, which is significantly longer than what is presented in Table 2. The training set consists of 37519 images, and the detector is trained for five epochs just as the detector of the IRcam worker.

2.4.1.4 *Fcam worker*

Initially, the fisheye lens camera was mounted facing upwards, but, as it turned out, this caused the image distortion to be significant in the area just above the horizon where the interesting targets usually appear. After turning the camera so that it faces forward, as can be seen in Figure 3, the motion detector is less affected by the image distortion, and since half of the field of view is not used anyway, this is a feasible solution. Initially, the image was cropped so that the half where the pan/tilt platform obscures the view was not used. Now instead, the part of the image covering the area below the horizon in front of the system is not processed.

The Fcam worker sets up queues and connects to the camera much in the same way as the IRcam and Vcam workers. The input image from the camera is 1024x768 pixels, and immediately after the `getsnapshot`-function the lower part of the image is cropped so that 1024x384 pixels remain to be processed.

The image is then analysed using the `ForegroundDetector`-function of the Computer vision toolbox. This uses a background subtraction algorithm based on Gaussian Mixture Models (GMM). The output from this function is a binary mask where moving objects are ones, and the stationary background is zero ideally. The mask is next processed with the `imopen`-function that first performs a morphological erosion followed by a dilation. The structural element parameter of the `imopen`-function is set to just 3x3 so that only very small objects are deleted.

To get rid of noise from the parts of the square-shaped image sensor that lie outside the circular area seen by the fisheye lens, the mask image is also multiplied with another binary mask before it is processed by the `BlobAnalysis`-function. This outputs an array of centroids and bounding boxes for all objects that are considered to be moving.

All these centroids and bounding boxes are sent to a Kalman filter multi-object tracker, which is a customised version of a script available in one of the Matlab computer vision toolbox tutorials [51]. Out of the tracks started and updated by the Kalman filter, the one with

the longest track history is picked out and marked as the best one. In the Fcam presentation window all tracks, both updated and predicted are visualised and the track considered to be the best is also marked with red colour. With the press of a button in the GUI, the user can choose to show the moving object mask in the presentation window instead of the normal fisheye lens camera image. This is shown in Figure 10

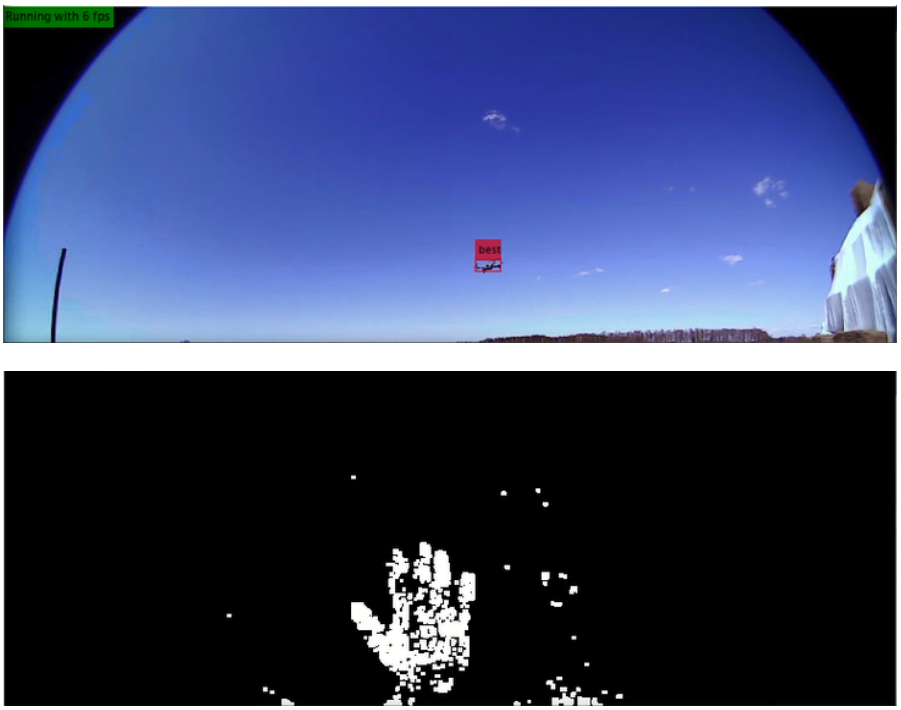


Figure 10: The normal Fcam image and track presentation above, and the moving object mask image below.

The output from the Fcam worker is the FPS-status, together with the elevation and azimuth angles of the best track, if such track exists at the moment. Out of all the workers, the Fcam is the one with most tuning parameters. This involves choosing and tuning the image processing operations, the foreground detector and blob analysis settings, and finally the multi-object Kalman filter tracker parameters.

2.4.1.5 *Audio worker*

The audio worker uses the attached directional microphone and collects acoustic data in a one-second long buffer (44100 samples), set to be updated 20 times per second. To classify the source of the sound in the buffer, it is first processed with the `mfcc`-function from the Audio toolbox. Based on empirical trails, the parameter `LogEnergy` is set to `Ignore`, and then the extracted features are sent to the classifier.

The LSTM-classifier used consists of an input layer, two bidirectional LSTM layers with a drop out layer in between, a fully connected layer, followed by a softmax layer and a classification layer. The classifier builds on [52] but with the extension of increasing the number of classes from two to three and an additional dropout layer between the bidirectional LSTM layers.

Since the audio database has 30 ten-second clips of each of the output classes, five clips from each class are set aside to be used for evaluation. The remaining audio clips are used for the training process. Out of each ten-second clip the last second is used for validation and the rest for training. The classifier is first trained for 250 epochs, but show signs of overfitting as can be seen in Figure 11 with the error on the training set being basically zero and the validation loss slightly increasing.

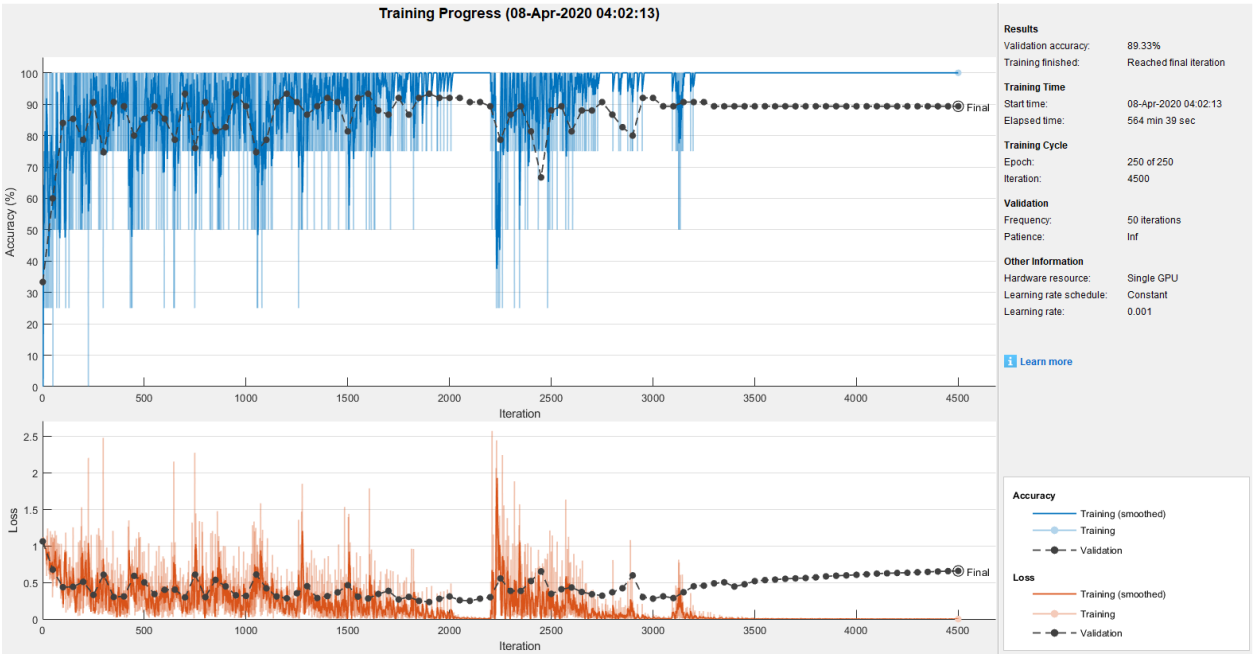


Figure 11: The audio classifier trained for 250 epochs showing signs of overfitting.

To overcome this the classifier is retrained from scratch with 120 epochs as shown in Figure 12.

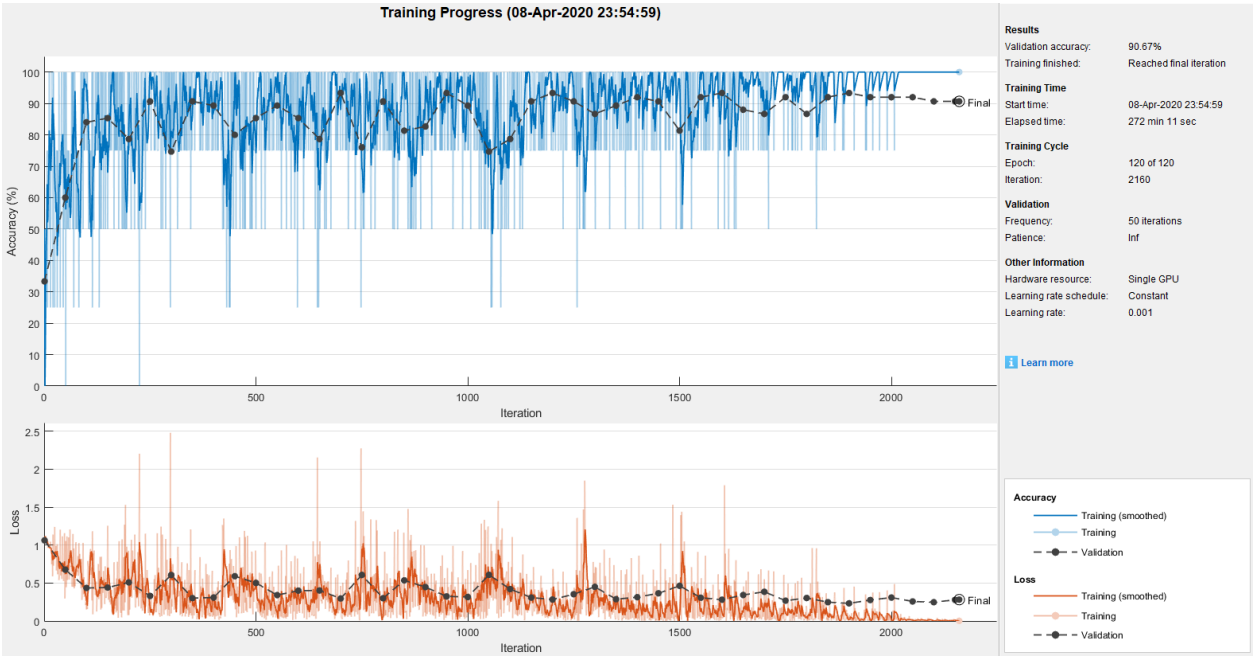


Figure 12: The audio classifier trained for 120 epochs.

Since the audio worker is essentially only a classifier and not a detector and classifier, like the YOLOv2-networks, it is trained with an additional class, as shown in Table 1, consisting of general background sounds. These are recorded outdoors in the typical deployment environment of the system, and also include some clips of the sounds from the servos moving the pan/tilt platform. Like the other workers, the output is a class label together with a confidence score.

A graphical presentation of the audio input and extracted features is also available as shown in Figure 13, but this is not included in the final GUI-layout.

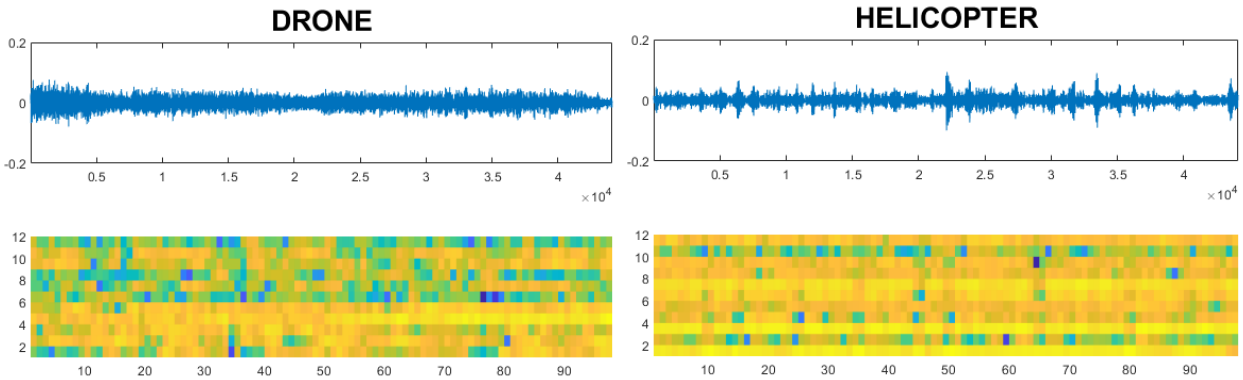


Figure 13: Two examples of audio worker plots with output classification labels, and below that the audio input amplitudes and the extracted MFCC-features.

2.4.1.6 *ADS-B worker*

As mentioned above, not all aircraft will send out their vehicle category as part of the ADS-B squitter message. Looking at how to implement the decoding of the ADS-B message, two alternative solutions arise. The first is to use the Dump1090-software [53] and then import the information into Matlab and have the worker just sorting the data to suit the main script. The other alternative is to implement the ADS-B decoding in Matlab using functions from the Communications toolbox.

The difference between the two solutions is that Dump1090 takes less computational resources, but even if the vehicle category field of the ADS-B squitter is filled, it is not available in the output JSON-format[1] message. The Matlab implementation, on the other hand, outputs all available information about the aircraft albeit using more processor power and hence slowing down the overall performance of the system.

To assess the best solution, data were recorded to investigate the fraction of aircraft sending out the vehicle class information. A script was set up as part of the support software to gather some statistical basis, and from this, it turned out that 328 aircraft out of 652, i.e. just over half sent out their vehicle category information. For the rest of these aircraft, the vehicle category was set to "NoData".

As mentioned in Section 1.2, a key principle of this thesis is to, by all possible means, detect and keep track of other flying objects that are likely to be mistaken for a drone, and the fact that about half of the aircraft sends out their vehicle category, the ADS-B decoding is implemented completely in Matlab despite the computational strain drawback.

All vehicle categories that can be seen as subclasses to the airplane target label are clustered together. These are all the classes "Light", "Medium", "Heavy", "HighVortex", "VeryHeavy" and "HighPerformanceHighSpeed". The class "Rotocraft" is translated into helicopter. Interestingly, there is also a "UAV" category label. This is also implemented in the ADS-B worker, although the label is translated into drone.

One might wonder if there are any such aircraft sending out that they belong to the UAV vehicle category. Examples are in fact found looking at the Flightradar24 service [54]. Here we can find one such drone as shown in Figure 14 flying at Gothenburg City Airport, one

1 JavaScript Object Notation

of the locations used when collecting the dataset of this thesis. The drone is operated by the company Everdrone AB [55], involved in the automated external defibrillators delivery trails of [1].

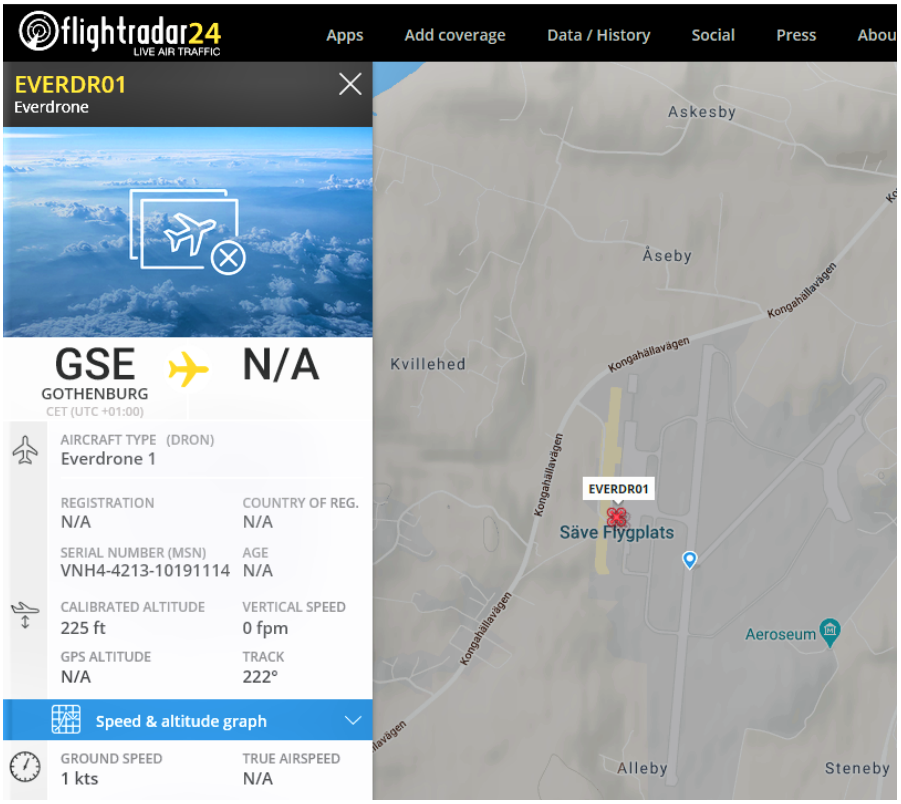


Figure 14: A drone flying at Gothenburg City Airport. From [54].

Another example seen in Figure 15 is the UK Coastguard/Border Force surveillance drone that is regularly flying at night over the straight of Dover since December of 2019. This is naturally a large drone with a wingspan of 7.3 m [56].

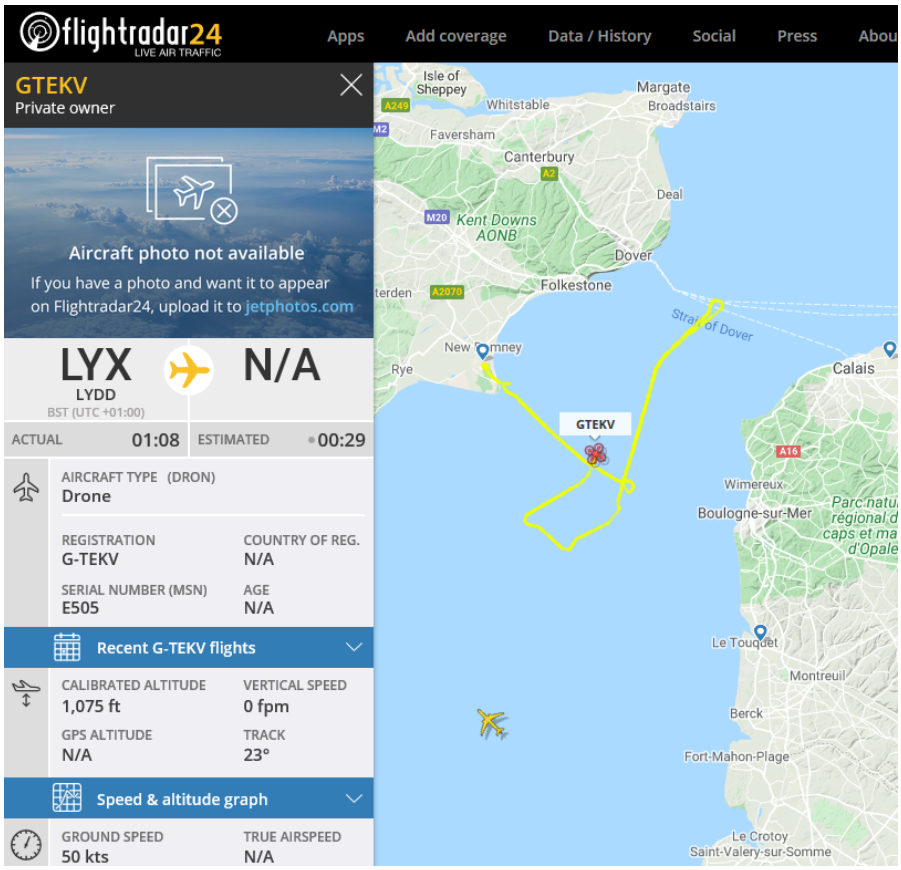


Figure 15: Surveillance drone over the straight of Dover. From [54].

In the output message of the ADS-B worker the confidence of the classification is set to 1 if the vehicle category message has been received, and if not, the label is set to airplane, since this is the most common aircraft type, with the confidence to 0.75 so that there is a possibility for any of the other sensors to influence the final classification.

2.4.2 *Support software*

The support software consists of a set of scripts to produce the training data sets, to set up the YOLOv2 network, and to do initial training of it. There are also scripts that import an already-trained network to perform additional training. Some of the tasks handled by the support software scripts are:

- Collection of ADS-B statistics
- Audio and video recordings and transformations
- Composition of the training datasets from the available data
- Estimation of the number and sizes of the anchor boxes for the setup of the YOLOv2 detectors
- Training and evaluation of the detectors and classifier

The Matlab video labelling app can also be considered to be a part of the support software, together with the Maestro Control Center [57] used to configure the servo controller, and the K-MD2-Radar Control Panel.

2.5 GRAPHICAL USER INTERFACE

The graphical user interface (GUI) is a part of the main script, but anyway described separately. The GUI presents the results from the different sensors/workers and also provides possibilities for the user to easily control the system without having to stop the code and changing it manually to get the desired configuration. The principal layout of the GUI is shown in Figure 16.

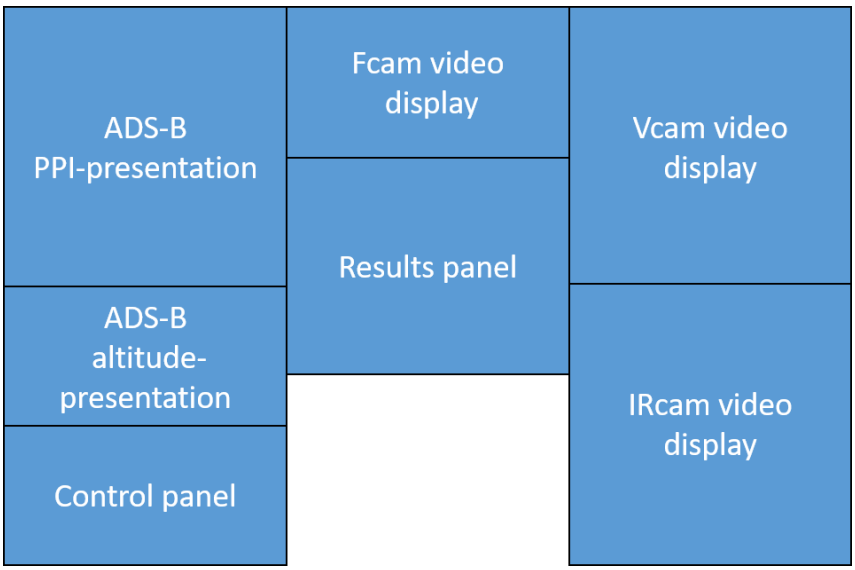


Figure 16: The GUI layout.

The gap under the results panel is intentional, making the Matlab command window visible at all times, so that messages, for example exceptions, can be monitored during the development and use of the system.

The ADS-B presentation area, shown in Figure 17, consists of a PPI-type[1] display and an altitude display. Besides presenting the ADS-B targets, the PPI display is also used to present system information. The solid green line is the main direction of the system relative to the north. The other green lines present the field of motion of the pan/tilt platform (dashed) and the field of view of the fisheye lens camera (dotted).

1 Plan Position Indicator

The actual direction of the pan/tilt platform is presented with a solid red line and the field of view of the IR- and video cameras are represented using dashed red lines. If any object is tracked by the fisheye lens camera worker its direction is indicated by a solid cyan line.

ADS-B targets are presented using the class label colours, as seen in Figure 17, together with the track history plots. The presentation of the altitude information is done in a logarithmic plot so that the lower altitude portion is more prominent.

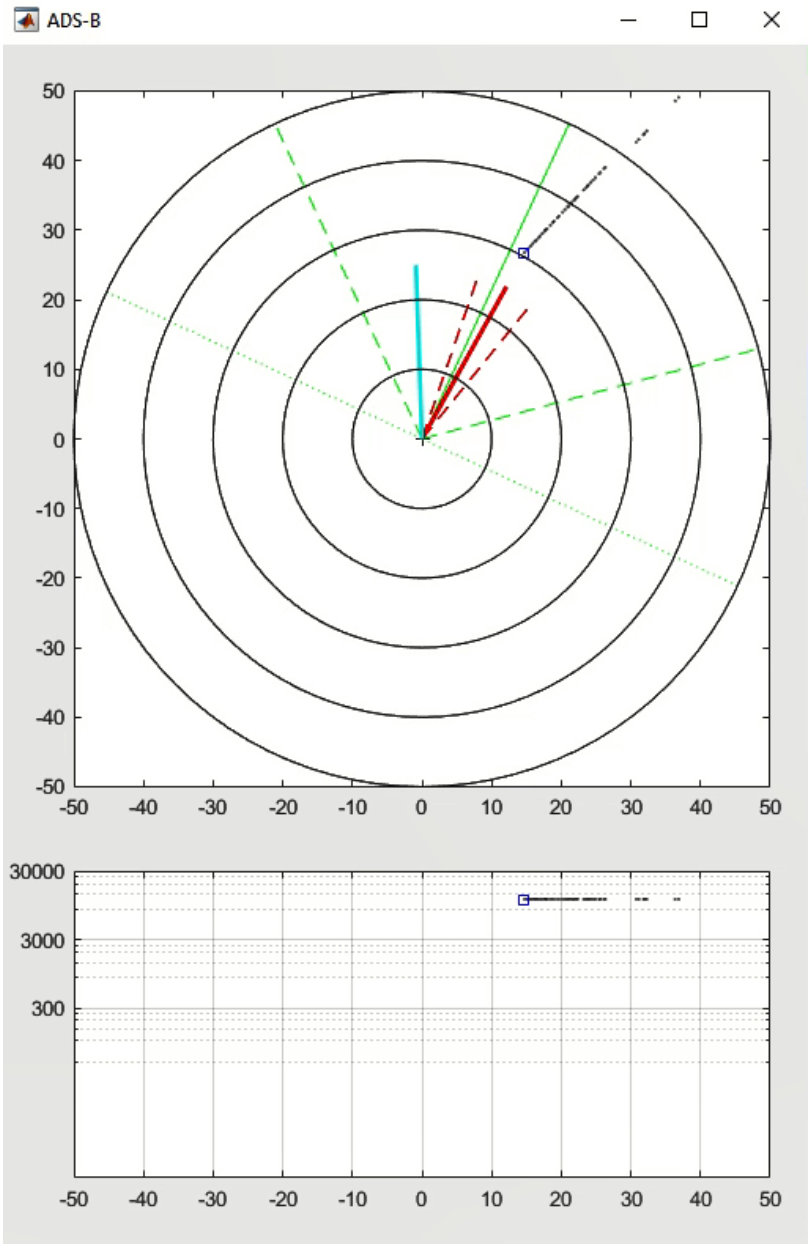


Figure 17: ADS-B presentation area. The circles of the PPI are 10 km apart.

Following the general GUI layout from Figure 16, the area directly below the ADS-B presentation is the control panel, as shown in Figure 18. Starting from the top left corner, we have radiobuttons[1] for the range settings of the ADS-B PPI and altitude presentations. Next is the number of ADS-B targets currently received and below that the set orientation angle relative to the north of the system. The Close GUI button is used to shut down the main script and the workers.

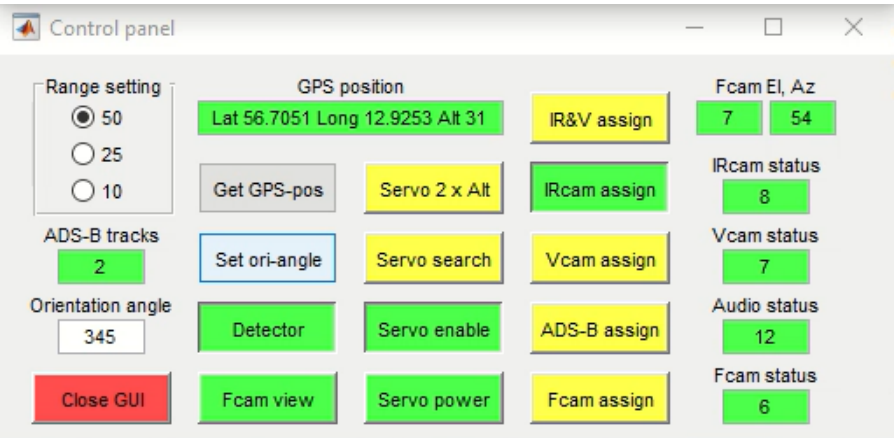


Figure 18: The control panel.

The GPS-position presentation changes colour to green when a valid position has been received from the GPS-receiver, after pressing the Get GPS-pos button. A press on the Set ori-angle opens an input dialogue box, so that the orientation angle of the system can be entered. Below that are buttons for switching the detectors between running and idle mode and the choice to display the normal Fcam image or the moving object mask, as shown in Figure 10.

The servo settings can be controlled with buttons in the mid column of Figure 18. To complement the Fcam in finding interesting object the pan/tilt can be set to move in two different search patterns. One where the search is done from side to side using a static elevation of 10°, so that the area from the horizon up to 20° is covered, and one where the search is done with two elevation angles to increase the coverage.

The pan/tilt platform can also be controlled by the elevation and azimuth angles from one of the workers. This is set by the assign-buttons, placed in priority order from the top and down. The IR&V assign setting means that a target has to be detected by both of the workers, and if so, the pan/tilt platform is controlled by the angular values from the IRcam worker.

1 In Matlab only one radiobutton in a radiobutton-group can be selected at once.

The rightmost column of Figure 18 is status information regarding the performance in FPS of the workers, and the elevation and azimuth angles of the Fcam worker target if such target exists. The status displays are red if the worker is not connected, yellow of the detector is idle, and green if it is running.

The results panel features settings for the sensor fusion and presents the workers and system results to the user. The servo controller column seen in Figure 19 indicates the source of information currently controlling the servos of the pan/tilt platform.

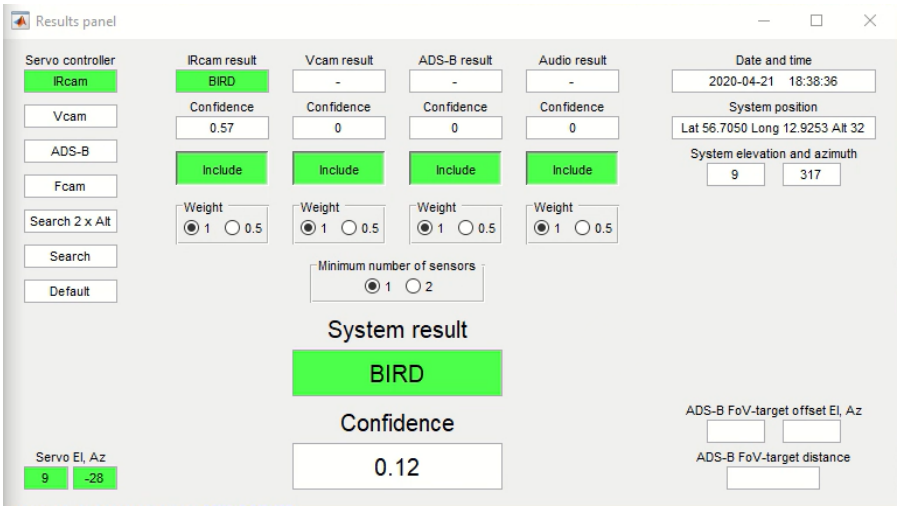


Figure 19: The results panel. Here a bird is detected and tracked by the IRcam worker.

In the lower-left corner of Figure 19, the angles of the servos are presented. The settings for the sensor fusion and the detection results presentation are found in the middle of the panel, as described in Section 2.4.1.1. The information in the right part of the panel is the current time and the position of the system. The system elevation and azimuth relative to the north are also presented here. Note the difference in azimuth angle compared to the lower-left corner where the system internal angle of the pan/tilt platform is presented.

The last part of Figure 19 presents offset angles for the ADS-B target, if one is present at the moment in the field of view of the IR- and video cameras. These values are used to detect systematic errors in the orientation of the system. The sloping distance to the ADS-B target is also presented here. See the bottom part of Figure 8 for an example of this.

Figure 20 shows the whole GUI, including the video displays. The image is turned 90° to use the space of the document better.

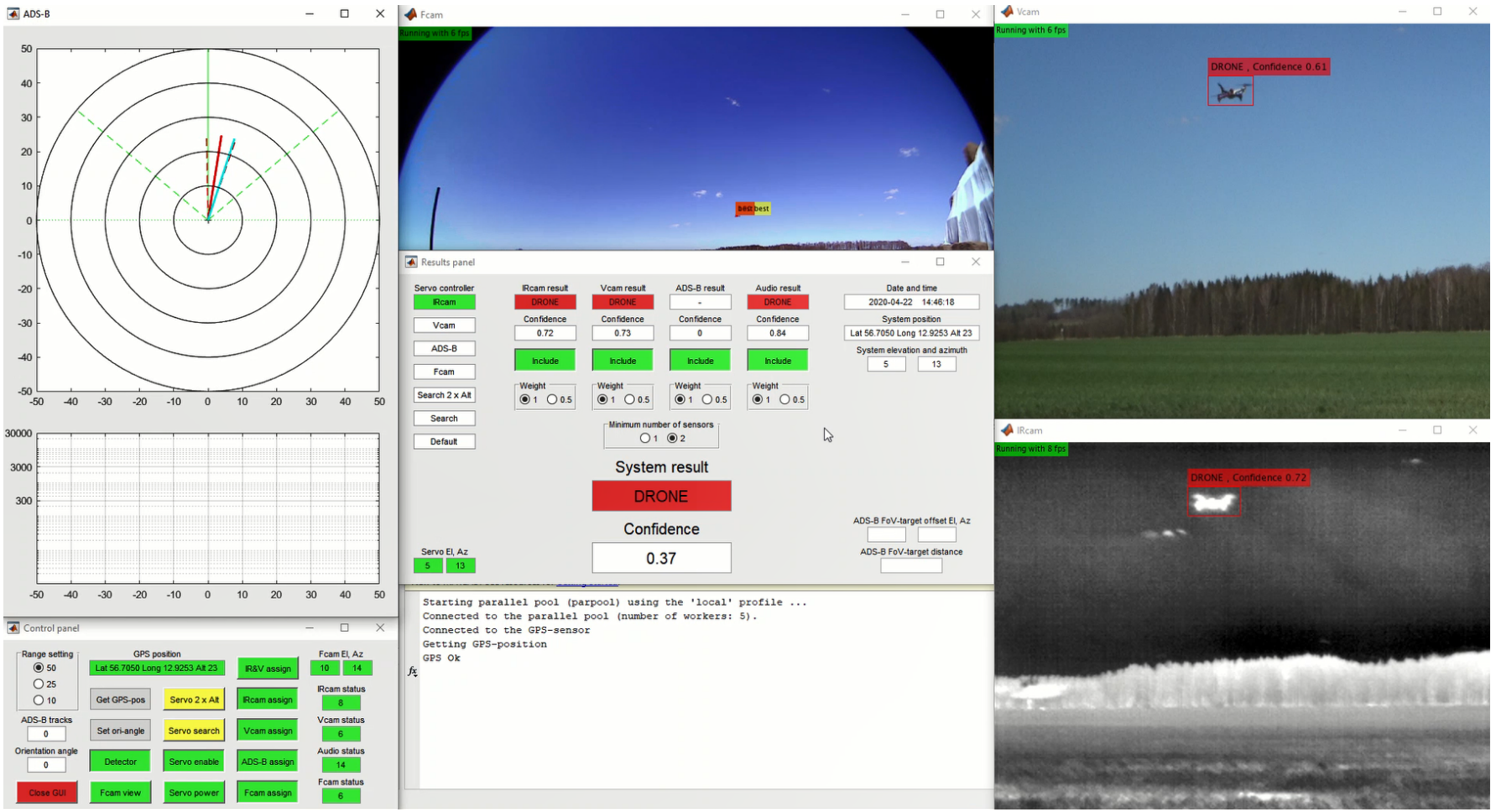


Figure 20: The graphical user interface. Here the IRcam, Vcam and audio workers all detect and classify the drone correctly. The Fcam worker is also tracking the drone, and the pan/tilt platform is for the moment controlled by the IRcam worker. The performance of the workers, in terms of frames per second, can also be seen in the status information.

2.6 DATASET FOR TRAINING AND EVALUATION

The videos in the dataset are recorded at locations in and around Halmstad and Falkenberg, at Halmstad Airport (HAD[1]/ESMT[2]), Gothenburg City Airport (GSE/ESGP) and Malmö Airport (MMX/ESMS).

Three different drones are used to collect and compose the dataset. These are of the following types: Hubsan H107D+, a small-sized first-person-view (FPV) drone, the high-performance DJI Phantom 4 Pro, and finally, the medium-sized kit drone DJI Flame Wheel. This can be built both as a quadcopter (F450) or in a hexacopter configuration (F550). The version used in this thesis is an F450 quadcopter. All three types can be seen in Figure 21.

(a) Hubsan H107D+ (b) DJI Phantom 4 Pro (c) DJI Flame Wheel F450

Figure 21: The three drone types of the dataset.

These drones differ a bit in size, with Hubsan H107D+ being the smallest having a side length from motor-to-motor of 0.1 m. The Phantom 4 Pro and the DJI Flame Wheel F450 are a bit larger with 0.3 and 0.4 m motor-to-motor side length, respectively.

The drone flights during the data collection and system evaluation are all done in compliance with the national rules for unmanned aircraft found in [58]. The most important points applicable to the drones and locations used in this thesis are:

- When flown, the unmanned aircraft shall be within its operational range and well within the pilot's visual line of sight
- When flown in uncontrolled airspace, the drone must stay below 120 m from the ground
- When flying within airports' control zones or traffic information zones and if you do not fly closer than 5 km from any section of the airport's runway(s), you may fly without clearance if you stay below 50 m from the ground

1 The airport code as defined by the International Air Transport Association (IATA)
2 The airport code as defined by the International Civil Aviation Organization (ICAO)

- For the protection of people, animals and property which are unrelated to the flight, there must be a horizontal safety distance between these and the unmanned aircraft throughout the flight

Since the drones must be flown within visual range, the dataset is recorded in daylight, even if the system designed in the thesis, to some extent can be effective even in complete darkness using the thermal infrared and acoustic sensors. The ADS-B information received is naturally also working as usual at night.

The weather in the dataset stretches from clear and sunny, to scattered clouds and completely overcast, as shown in Figure 22.

Figure 22: Examples of varying weather conditions in the dataset.

Common birds in the dataset are the rook (*Corvus frugilegus*), as shown in Figure 23, and the western jackdaw (*Coloeus monedula*) of the crow family (*Corvidae*), the european herring gull (*Larus argentatus*), the common gull (*Larus canus*) and the black-headed gull (*Chroicocephalus ridibundus*) of the *Laridae* family of seabirds. Occasionally occurring in the dataset are also the black kite (*Milvus migrans*) of the *Accipitridae* family and the eurasian skylark (*Alauda arvensis*) of the lark family (*Alaudidae*). On average these species have a wingspan of 0.8 m, making them about twice the size of the medium-sized consumer-grade drone.

Figure 23: A rook detected and classified correctly.

The audio in the dataset is taken from the videos or recorded separately using one of the support software scripts.

Both the videos and the audio-files are cut into ten-second clips to be easier to annotate. To get a more comprehensive dataset, both in terms of aircraft types and sensor-to-target distances, it has also been completed with non-copyrighted material from the YouTube channel "Virtual Airfield operated by SK678387" [59]. This is in total 11 plus 38 video clips in the airplane and helicopter categories, respectively.

Since one of the objectives of this thesis is to explore performance as a function of the sensor-to-target distance, the dataset has also been divided into the distance category bins: Close, Medium and Distant. The borders between these bins are chosen to follow the industry-standard Detect, Recognize and Identify (DRI) requirements [60], building on the Johnson criteria [61], as shown in Figure 24.

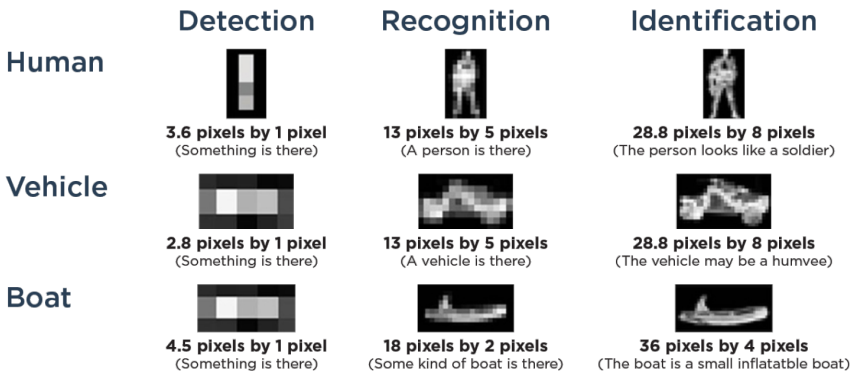


Figure 24: The DRI-requirements. From [60].

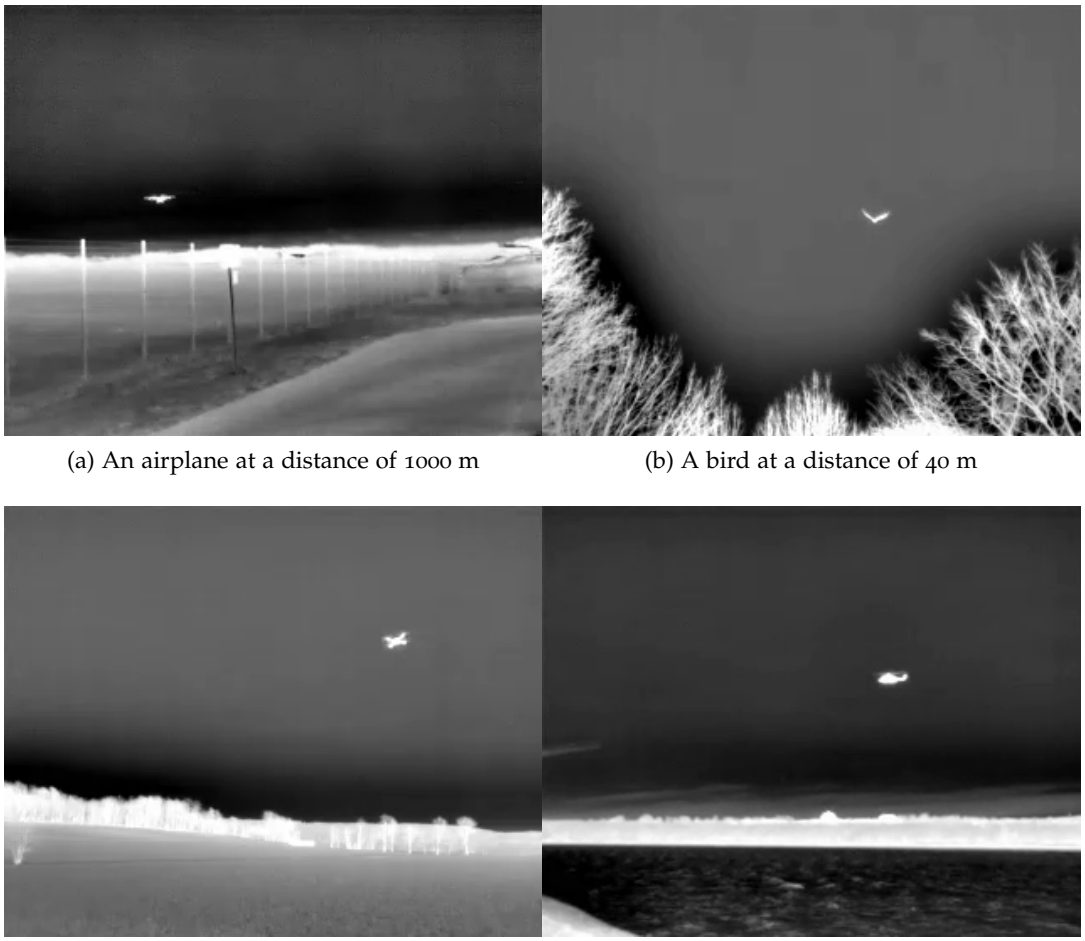
The Close distance bin is from 0 m out to a distance where the target is 15 pixels wide in the IRcam image, i.e. the requirement for recognition according to DRI. The Medium bin stretches from where the target is from 15, and down to 5 pixels, hence around the DRI detection point, and the Distant bin is beyond that. Given the resolution and field of view of the IRcam and the object class sizes: Drone 0.4 m, bird 0.8 m, helicopter[1] 10 m and airplane[2] 20 m, we get a distance division for the different object types summarized in Table 3.

Table 3: The distance bin division for the different target classes.

Bin	Class			
	AIRPLANE	BIRD	DRONE	HELICOPTER
Close	< 1000 m	< 40 m	< 20 m	< 500 m
Medium	1000-3000 m	40-120 m	20-60 m	500-1500 m
Distant	> 3000 m	> 120 m	> 60 m	> 1500 m

To illustrate the detect, recognize and identify concept, objects from all the target classes being 15 pixels i width are shown in Figure 25.

1 Bell 429, one of the helicopter types in the dataset, has a length of 12.7 m
2 Saab 340 has a length of 19.7 m and a wingspan of 21.4 m



(a) An airplane at a distance of 1000 m

(b) A bird at a distance of 40 m

(c) A drone at at distance of 20 m

(d) A helicopter at a distance of 500 m

Figure 25: Objects on the limit between the close and medium distance bins.

At this level, we can not only detect but also recognize the different objects, albeit without necessarily identifying them, i.e. explicitly telling what kind of helicopter it is an so on.

Note that exactly the same distances used in this division, are also implemented for the video data. This notwithstanding the fact that the input layer of the Vcam worker YOLOv2 detector has a resolution 1.6 times that of the IRcam ditto as described in Section 2.4.1.2 and Section 2.4.1.3.

The annotation of the video dataset is done using the Matlab video labeller app. An example from a labelling session is shown in Figure 26.

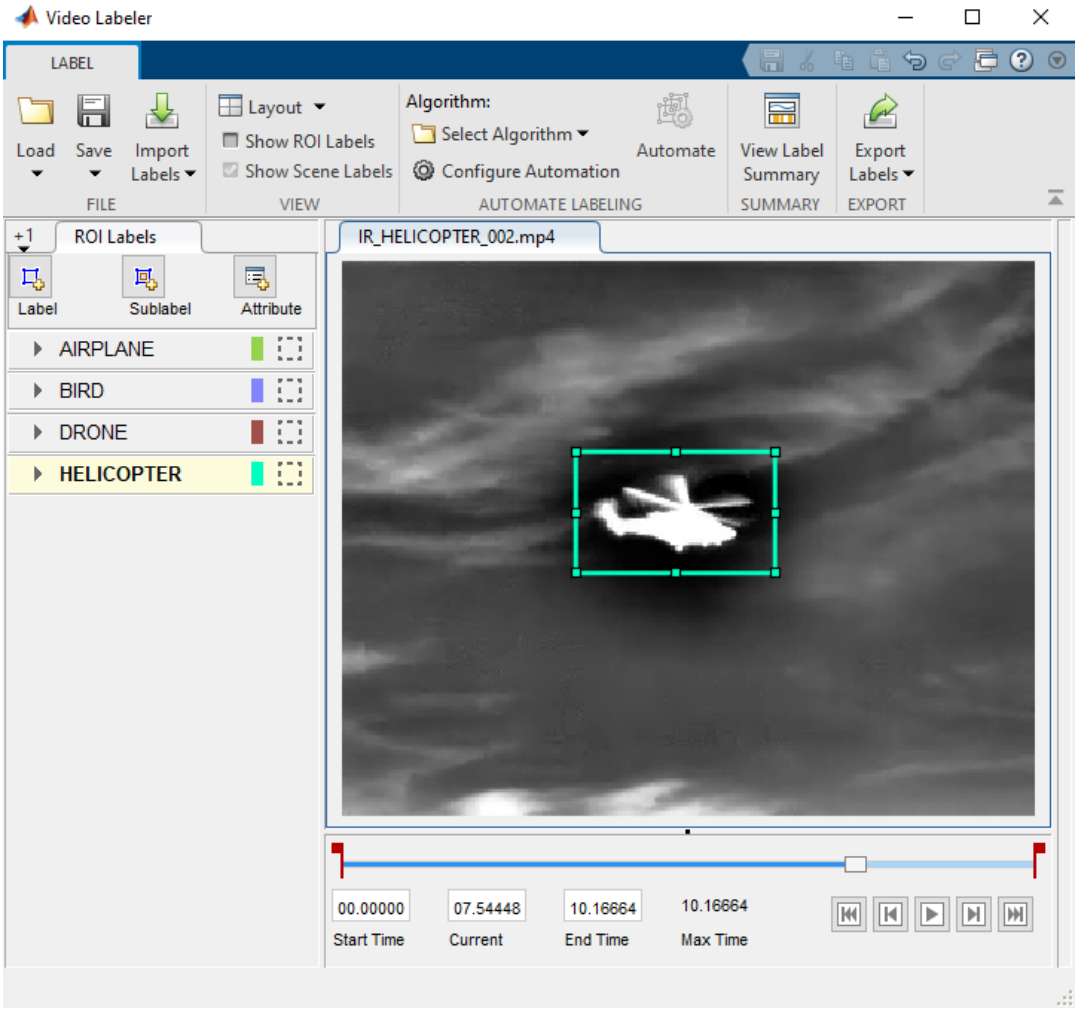


Figure 26: The Matlab video labeller app.

From the dataset, 120 clips (5 from each class and target bin) were put aside to form the evaluation dataset. Out of the remaining videos 240 were then picked as evenly distributed as possible to create the training set.

3

RESULTS

Recall that the scope of this thesis work was twofold: First, to explore the possibilities and limitations of designing and constructing a multi-sensor drone detection system while building on state-of-the-art methods and techniques. Secondly, to collect, compose and publish a drone detection dataset.

In what follows the results of the thesis are outlined, initially with the performance of the drone detection system, on both sensor and system levels, and finally, the dataset and its content is presented.

The evaluation of the system performance has been the hardest thing to do, both in terms of finding other work to compare with, but also since not all of the scenarios that the system should be able to handle are practically feasible to evaluate. For example, testing the drone detection capability of the system at a busy airport would not only be breaking the rules and regulation for drone flights but could also put people and property at risk.

First, the evaluation is done on detector level, this means measuring the performance in terms of precision[1], recall[2] and F1-score[3], of the individual sensor on a single image or audio signal. Secondly, the evaluation is done on system-level, i.e. the total output of the system after the sensor fusion part. Such results are sparse, but can be found in [40] and [14].

3.1 PERFORMANCE OF THE INDIVIDUAL SENSORS

To evaluate the individual sensors, a part of the dataset was put aside at an early stage and hence kept out of the training process. The evaluation set for the audio classifier contains five 10-second clips from each output category. Since the classifier processes a one-second input buffer, the evaluation set is also cut into that length, and using an overlap of 0.5, there are a total of 297 clips in the evaluation set, 99 from each class.

1 How many of the selected items are relevant? $\mathrm{Precision} = \frac{tp}{tp+fp}$

2 How many of the relevant items are selected? $\mathrm{Recall} = \frac{tp}{tp+fn}$

3 The F1-score is defined to be the harmonic mean of the precision and recall, hence $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

For the audio classifier, the performance is presented using a confusion matrix. The precision, recall and F1-score of the classifier are also calculated, so that this can be compared to related results.

For the detectors of the IRcam and Vcam workers, not only the classification label but also the placement of the bounding box must be taken under consideration. In this the IoU, as defined in Section 2.4.1.2, is used once more. The use similar IoU-requirements facilitates the results to be compared, and looking at the related work, we see that an IoU of 0.5 is used in [15], [18] and [19]. A lower IoU of 0.2 is used in [43].

The evaluation set is composed of five representative videos from each class and distance bin. Since the videos are not all precisely ten seconds[1], the total number of images in each evaluation set is as presented in Table 4.

Table 4: The number of images in the evaluation dataset.

Sensor	Distance bin			Total
	Close	Medium	Distant	
IR	6256	6226	6209	18691
Visible	6256	6254	6263	18773

The performances of the video detectors are both evaluated using the `bboxPrecisionRecall`-function to get the precision, recall and F1-score. Since we also have a confidence score of each detection the `evaluateDetectionPrecision` can be utilized to get plotted curves of precision as a function of the recall value, i.e. the PR-curve.

An important thing to keep in mind is that the ground truth is made by a person and will contain both errors and inconsistency regarding the placements and sizes of the bounding boxes.

3.1.1 *Thermal infrared detector and classifier*

Results regarding the use of an IR-sensor can be found in [11]. However, the sensor used in that article to detect drones, up to a distance of 100 m, has a resolution of just 80x60 pixels, and the system does not involve any form of machine learning feature. In that paper, three different drones were used, and a DJI Phantom 4 was detected by the setup up to a distance of 51 m.

1 10.4 seconds on average

In [12], curves for the precision and recall of a machine learning based thermal detector are presented. It is stated that the video clips used for training and evaluation have a frame resolution of 1920x1080. Unfortunately, the paper fails to mention if this is also the resolution of the sensor. Neither is the input size if the detection network specified in detail, other than that the images are rescaled so that the shorter side has 600 pixels.

The most substantial results to relate to in [12], is that since the system also contains a video camera with the same image size as the thermal one, the authors can conclude that the thermal drone detector has a performance over 8% better than the video detector.

Using the distance bin division described in Section 2.6 the precision and recall of the IRcam worker detector, with a confidence threshold set to 0.5 and an IoU-requirement of 0.5, are shown in Table 5, Table 6 and Table 7 below. This thesis conforms to the use of an IoU-requirement of 0.5.

Table 5: Precision and recall of the IRcam worker detector for the distance bin Close.

	Class				Average
	AIRPLANE	BIRD	DRONE	HELICOPTER	
Precision	0.9197	0.7591	0.9159	0.9993	0.8985
Recall	0.8736	0.8508	0.8790	0.8792	0.8706

Table 6: Precision and recall of the IRcam worker detector for the distance bin Medium.

	Class				Average
	AIRPLANE	BIRD	DRONE	HELICOPTER	
Precision	0.8281	0.5063	0.8951	0.9554	0.7962
Recall	0.7039	0.7033	0.8034	0.8355	0.7615

Table 7: Precision and recall of the IRcam worker detector for the distance bin Distant.

	Class				Average
	AIRPLANE	BIRD	DRONE	HELICOPTER	
Precision	0.7822	0.6161	0.8278	0.7982	0.7561
Recall	0.4043	0.7431	0.4836	0.4564	0.5218

Taking the average result from each of these distance bins and calculate their receptive F1-scores, we obtain Table 8.

Table 8: F1-scores for the IRcam worker detector

	Distance bin			Average
	Close	Medium	Distant	
F1-score	0.8844	0.7785	0.6175	0.7601

We can observe that the precision and recall values are well balanced using a detection threshold of 0.5, and altering the setting confirms that a higher threshold leads to higher precision, at the cost of a lower recall value, as shown in Table 9. The drop in recall with increasing sensor-to-target distance is also prominent.

Table 9: Precision and recall values of the IRcam worker detector, averaged over all classes, using a detection threshold of 0.8 instead of 0.5.

	Distance bin			Average
	Close	Medium	Distant	
Precision	0.9985	0.9981	1.0000	0.9987
Recall	0.2233	0.1120	0.0019	0.1124

To further explore the detection threshold setting, we can run the evaluation with values from 0.1 up to 1.0, in steps of 0.1. This is shown in Figure 27.

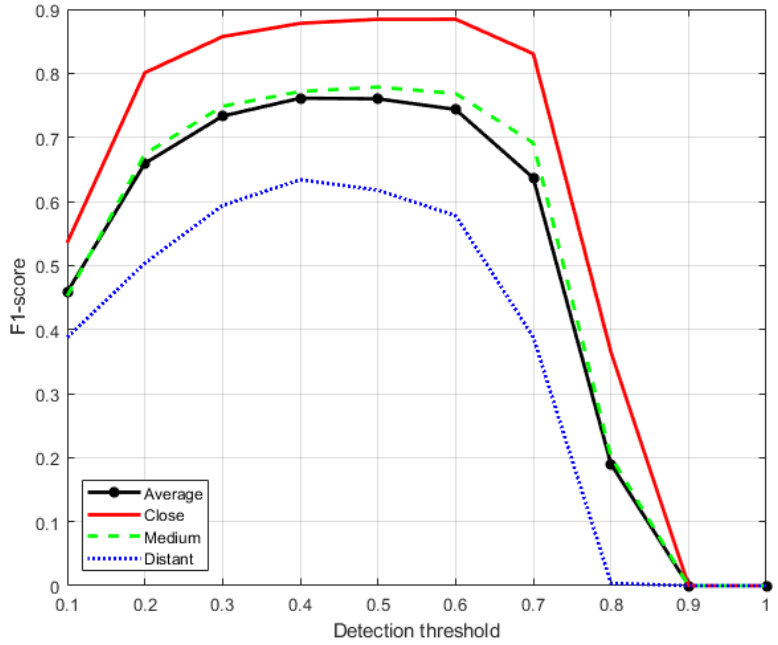


Figure 27: The F1-score of the IRcam worker detector as a function of detection threshold, using all the 18691 images in the evaluation dataset.

Using not only the bounding boxes and class labels, but also the confidence scores, the detector can be evaluated using the Matlab `evaluateDetectionPrecision`-function. From this, we obtain plots of the PR-curves[1], as shown below in Figure 28.

Note that the average precision results output from the Matlab `evaluateDetectionPrecision`-function is defined to be the area under the PR-curve, and hence it is not the same as the actual average precision values of the detector on the evaluation dataset, as presented in Table 5, Table 6 and Table 7 above.

To distinct that we mean the area under the PR-curve we denote this as the AP, just as in the original YOLO paper [49]. This is also the definition used in [18], and to further adopt the notation of these papers we denote the mean AP taken over all classes as the mAP.

1 The precision of the detector plotted as a function of the recall value

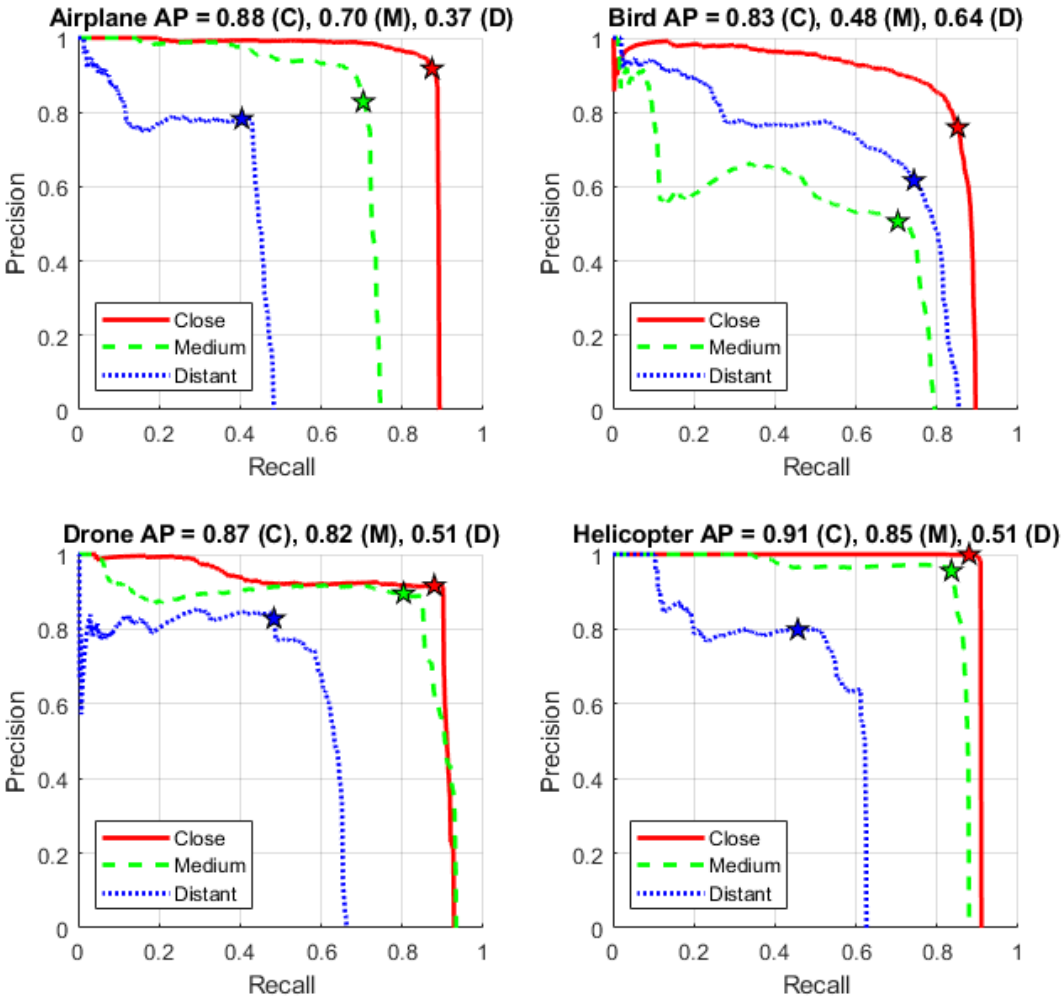


Figure 28: Precision and recall curves for the IRcam worker detector. The achieved values with a detection threshold of 0.5 are marked by stars.

The choice of the detection threshold will affect the achieved precision and recall values. By plotting the values from Table 5, Table 6 and Table 7 as stars in Figure 28, we can conclude that a threshold of 0.5 results in a balanced precision-recall combination near the top right edge of the respective curves. Compare this to the precision and recall values we obtain when using a detection threshold of 0.8, as shown in Table 9 above.

Calculating the mAP for the IRcam worker we obtain Table 10.

Table 10: The mean values, over all classes, of the area under the PR-curve (mAP) of the IRcam worker detector for the different distance bins including the average of these values.

	Distance bin			Average
	Close	Medium	Distant	
mAP	0.8704	0.7150	0.5086	0.7097

The results presented above will be compared to the results of the video detector in what follows below.

From observations of the behaviour when running the drone detection system, we can also point out that a common source of false alarms of the IRcam worker detector are small clouds and edges of large clouds lit up by the sun. An example of this can be seen in Figure 29.

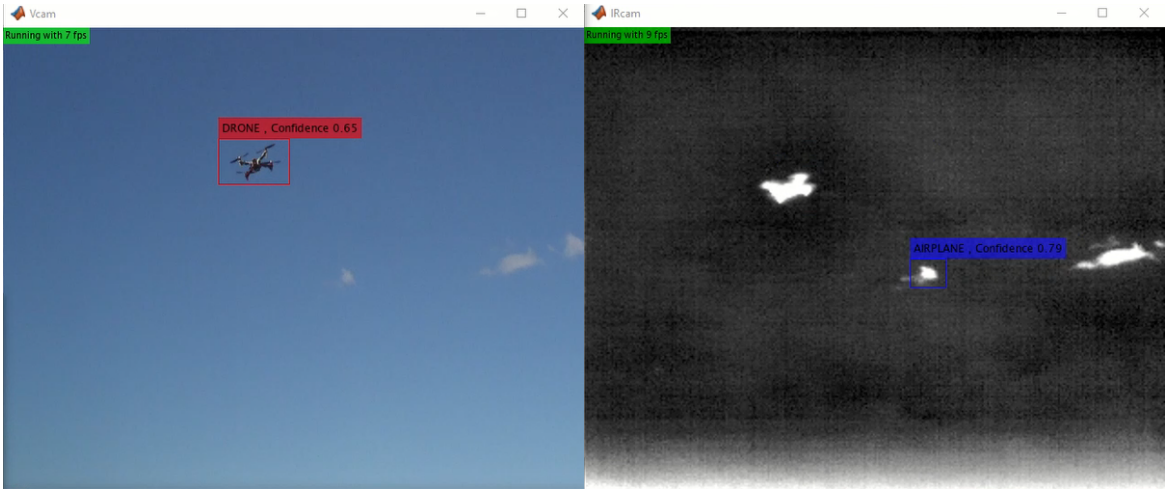


Figure 29: A false target of the IRcam worker detector caused by a small cloud lit by the sun.

3.1.2 *Video detector and classifier*

To be able to compare the results of the IRcam worker to the performance of the Vcam worker detector the same methods and settings as above are used to get Table 11, Table 12, Table 13 and Table 14.

Table 11: Precision and recall of the Vcam worker detector for the distance bin Close.

	Class				Average
	AIRPLANE	BIRD	DRONE	HELICOPTER	
Precision	0.8989	0.8284	0.8283	0.9225	0.8695
Recall	0.7355	0.7949	0.9536	0.9832	0.8668

Table 12: Precision and recall of the Vcam worker detector for the distance bin Medium.

	Class				Average
	AIRPLANE	BIRD	DRONE	HELICOPTER	
Precision	0.8391	0.7186	0.7710	0.9680	0.8242
Recall	0.7306	0.7830	0.7987	0.7526	0.7662

Table 13: Precision and recall of the Vcam worker detector for the distance bin Distant.

	Class				Average
	AIRPLANE	BIRD	DRONE	HELICOPTER	
Precision	0.7726	0.6479	0.8378	0.6631	0.7303
Recall	0.7785	0.7841	0.5519	0.5171	0.6579

Table 14: F1-scores for the Vcam worker detector.

	Distance bin			Average
	Close	Medium	Distant	
F1-score	0.8682	0.7942	0.6922	0.7849

These results differ no more than 3% from the results of the IRcam worker detector. Recall that the input layers of the YOLOv2-detectors are different and hence that the resolution of the Vcam worker[1] is 1.625 higher than that of the IRcam worker[2]. So even with a lower resolution, and the fact that the image is in greyscale and not in colour, the IR sensor performs as well as the visible one. This conforms well

1 416x416 pixels
2 256x256 pixels

with the conclusions in [12], where the IR detector outperforms the visible when the image sizes are the same.

Comparing the results to other papers we see that the YOLOv2-detector in [15] achieves an F1-score of 0.728 with exactly the same detection threshold and IoU-requirement. This F1-score is just below the results of the IRcam and Vcam workers.

However, one notable difference lies in that the detector in [15] has only one output class. This fact could confirm the doctrine of this thesis, i.e. that the detectors should also be trained in recognizing object easily confused for being drones. Unfortunately, there is no notation of the sensor-to-target distance other than that *"75% of the drones have widths smaller than 100 pixels"*. Since the authors implement an original YOLOv2 model from darknet, it is assumed that the input size of the detector is 416x416 pixels.

Just as for the IRcam, as shown in Figure 27, we can also explore the effects of the detection threshold setting. This can be seen in Figure 30 below.

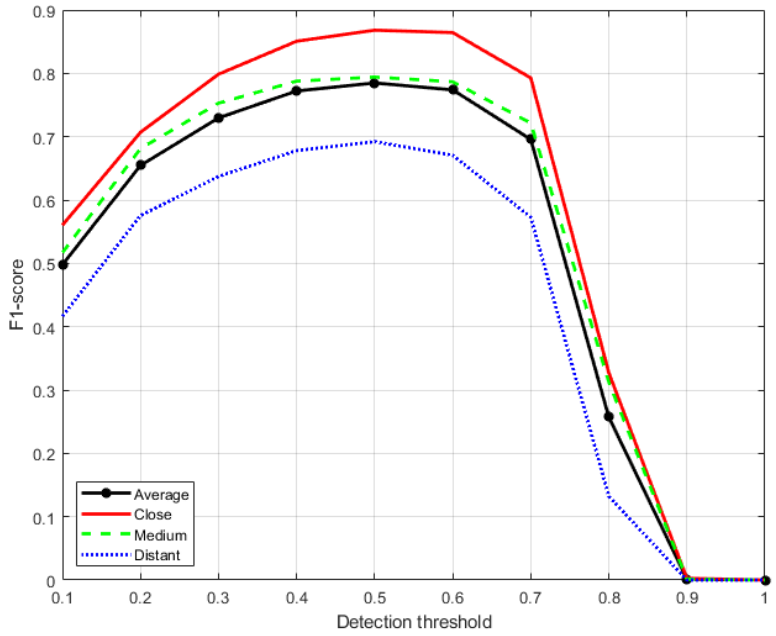


Figure 30: The F1-score of the Vcam worker detector as a function of detection threshold, using all the 18773 images in the evaluation dataset.

The PR-curves of the Vcam worker detector for the different target classes and distance bins are shown in Figure 31.

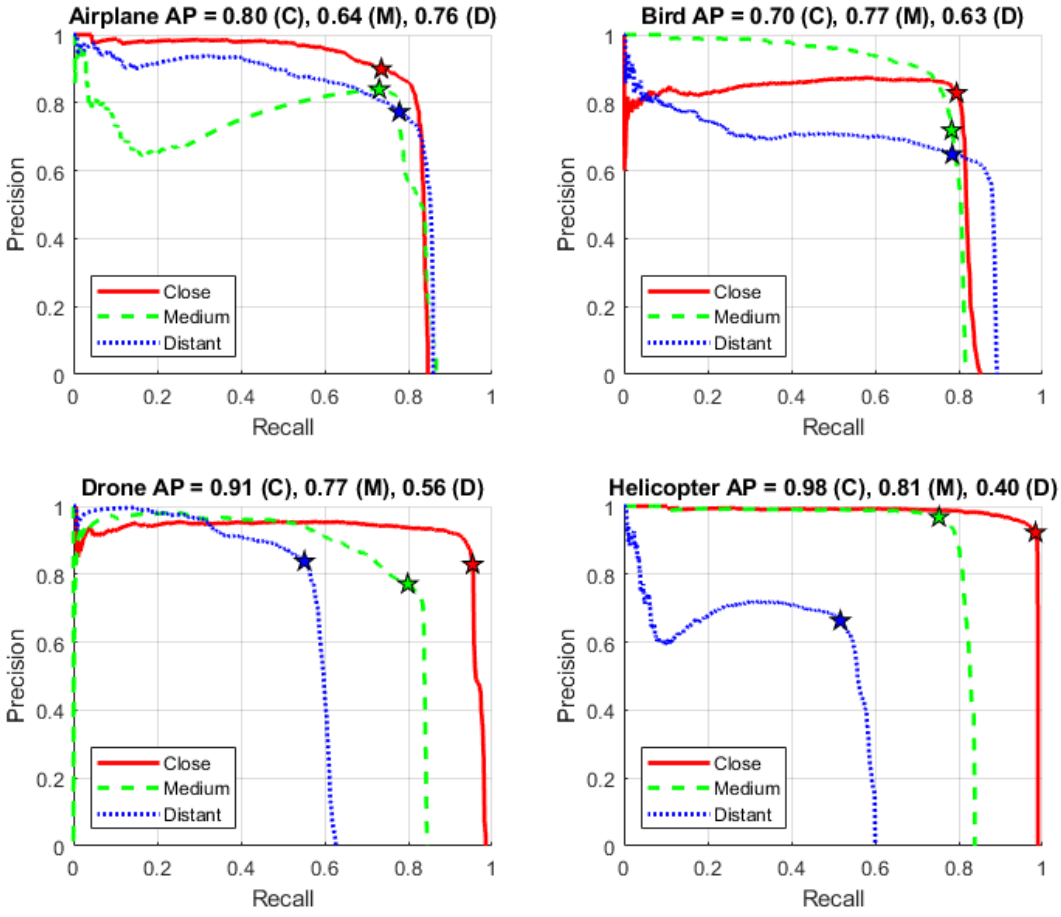


Figure 31: Precision and recall curves for the Vcam worker detector. The achieved values with a detection threshold of 0.5 are marked by stars.

Calculating the mAP for the Vcam worker from the results above we obtain Table 15.

Table 15: The mean values, taken over all classes, of the area under the PR-curve (mAP) of the Vcam worker detector for the different distance bins including the average of these values.

	Distance bin			Average
	Close	Medium	Distant	
mAP	0.8474	0.7477	0.5883	0.7261

Once again, this is not far from the 0.7097 mAP of the IRcam worker detector. The result is also close to what is presented in [18] where a mAP of 0.66 achieved, albeit using a detector with drones as the only output class and giving no information about the sensor-to-target distances.

A YOLOv2-architecture with an input size of 480x480 pixels is implemented in [19]. Moreover, the detector has two output classes, birds and drones. Based on the presented PR-curve, the authors state that they *"can understand from the curve that precision and recall can be achieved to be approximately 0.9 at the same time"*.

The performances of the IRcam and Vcam detectors together with the reported comparable results are summarized in Table 16. Additionally, the table shows the output classes used. [44] presents the results of the four teams participating in the 2019 Drone-vs-Bird detection challenge, so from that the results of the winning team [62] is also included in the Table 16. The approach of [62] is to first use a U-net to perform a two-class semantic segmentation that differentiates flying targets from the background. Then a Resnetv2 is implemented to classify flying targets as birds or drones.

Table 16: Results from the related work and the IRcam and Vcam worker detectors.

Source/Detector	Results				Classes[1]
	Precision	Recall	F1-score	mAP	
Park et al. 2017 [15]			0.73		D
Liu et al. 2018 [17]	0.99	0.80[2]			A, D, H
Saqib et al. 2017 [18]				0.66	D
Aker and Kalkan. 2017 [19]	≈ 0.9	≈ 0.9			B, D
Craye and Ardjoune. 2019 [62]			0.73		B, D
IRcam worker detector	0.82	0.72	0.76	0.71	A, B, D, H
Vcam worker detector	0.81	0.76	0.78	0.73	A, B, D, H

Notably, when inspecting the PR-curves in Figure 31, the Vcam worker detector performs outstandingly when it comes to distant airplanes. This has its explanation in that such targets often presents a very large signature consisting not only of the airplane itself but also contrails behind it. An example of this, from a system evaluation session, is seen in Figure 32. In this screenshot, an airplane is detected and classified at a sloping distance of more than 35000 m. At this distance even a large commercial airplane is only about 1.4 pixels, and hence well below the detection limit of the DRI-criteria.

1 Shortened to fit the table. A = Airplane, B = Bird, D = Drone and H = Helicopter

2 This is denoted as accuracy in the paper, but from the context and description it is assumed that it is actually the recall that is reported. The result in the table is the average of the following class results: Airplane 0.9603, Drone 0.5213 and Helicopter 0.9047

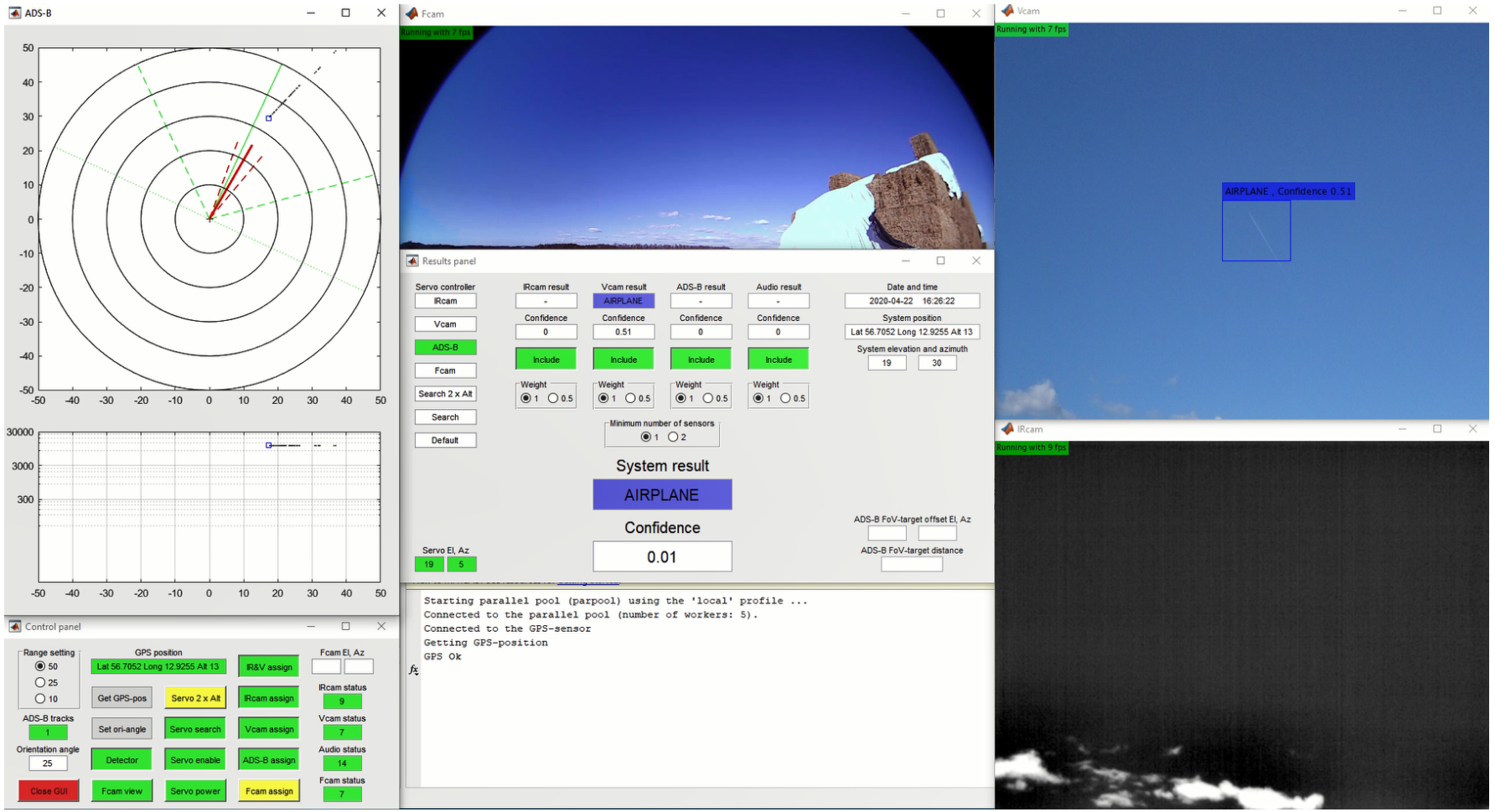


Figure 32: An airplane detected and classified correctly by the Vcam worker at a sloping distance of more than 35000 m. The reason behind the fact that the ADS-B FoV-target distance display is empty is that the main script will not present that information until the target is within 30000 m horizontal distance. This limit is set base on the assumption that no target beyond 30000 m should be detectable. An assumption that clearly turned out to be wrong.

The most frequent problem of the video part, when running the drone detector system outside, is the autofocus feature of the video camera. Unlike the Fcam and IRcam, clear skies are not the ideal weather, but rater a scenery with some objects that can help the camera to set the focus correctly. However, note that this fact is not affecting the evaluation results of the Vcam worker detector performance, as presented above, since only videos where the objects are seen clearly and hence are possible to annotate are used.

Figure 33 shows an example of when the focus is set wrongly, so that only the IRcam worker detects the drone.

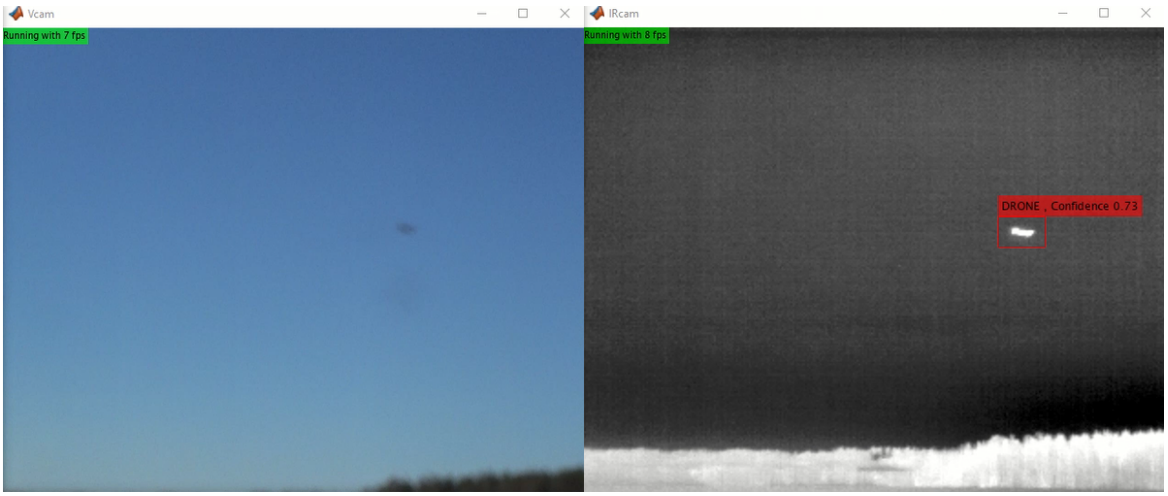


Figure 33: The drone detected only by the IRcam since the autofocus of the video camera is set wrongly.

A more subjective way to evaluate the performance of the Vcam worker detector in particular, is to observe the behaviour when running it as a stand-alone application on a randomly chosen aircraft video taken from the Internet. This procedure provides a form of assessment of the detector, without all the tedious work of annotating the video first. Four screenshots from such a session are shown in Figure 34.

Figure 34: Airplanes, a helicopter and birds detected when running the Vcam worker detector as a stand-alone application on an aircraft video from [59]. Note the FPS-performance.

The most striking observation is that the detector is surprisingly good at detecting birds not visible in the video at first glance. When used as a stand-alone application, the detector shows not only the object with the highest confidence score but all objects that exceed the detection threshold. Additionally, note the higher FPS-performance (16 FPS), with just one detector running on the computer, compared to Figure 33 (7 FPS) when running the whole system.

3.1.3 *Fisheye lens camera motion detector*

The fisheye lens camera is included in the system to be able to cover a larger airspace volume than covered by just the field of view of the IRcam and Vcam. However, the drone detection system is not totally reliant on the Fcam worker to detect objects of interest, since the system also include the search programs moving the pan/tilt platform when no other targets are detected. The search programs can easily be turned on or off using the control panel of the GUI.

It was initially observed that the foreground/background-based motion detector of the Fcam worker was sensitive to objects moving with the wind as shown in Figure 35

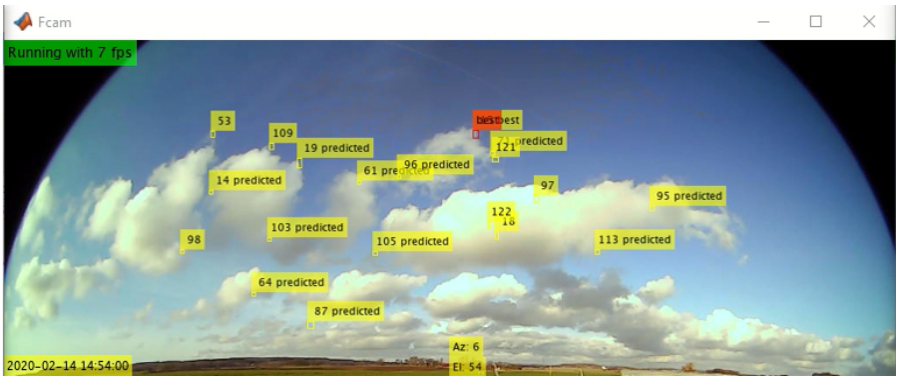


Figure 35: False targets in the Fcam image.

The number of false targets has been mitigated by extensive tuning of the image processing operations, the foreground detector, the blob analysis settings and the parameters of the multi-object Kalman filter tracker, from [51]. Some of the most critical points in this was first to remove the `imclose`- and `imfill`-functions that was initially implemented after the `imopen`-function in the image processing operations.

Furthermore, the minimum background ratio setting[1] of the foreground detector was increased together with the learning rate, so that the GMM more quickly adapts to changing conditions. In the blob analysis settings the minimum blob area was reduced, so that smaller objects are detectable and a maximum blob area was also implemented.

Tuning the parameters of the Kalman filter multi-object tracker was also important. These were altered to make the tracker slower to start new tracks and quicker to terminate them if no moving objects were present at the predicted positions.

As mentioned in Section 2.4.1.4 an extra mask was also introduced to eliminate noise from parts of the square-shaped image sensor that lie outside the circular area seen by the fisheye lens.

With a resolution of 1024x768 pixels, out of which 1024x384 are used as described in Section 2.4.1.4, the Fcam worker moving object detector and tracker has been found, during the evaluation sessions, to be an effective way to assign objects to the pan/tilt platform up to a distance of 50 m against drones. Beyond this distance, the drone is

1 This is threshold to control what pixels are to be considered as part of the foreground or the background

also theoretically so small, measured in pixels, so that it is deleted by the `imopen`-function.

The maximum resolution of the Fcam is 3264x2448, so a greater detection range should theoretically be achievable. However, using a higher resolution is also more demanding in terms of computational resources, leading to a reduction in the FPS-performance, not only for the Fcam worker, but also for the other workers. Since the fisheye lens camera is also complemented by the search program, the choice has been made to settle with this resolution and the limited detection range that follows from this.

Figure 36 shows a drone tracked by the Fcam worker. At this moment the pan/tilt platform is controlled by the output of the Fcam worker. Just a moment later, as seen in Figure 37, the drone is detected by the IRcam and Vcam workers, and the pan/tilt platform is therefore controlled by the IRcam worker output.

Figure 36: A drone tracked by the Fcam worker.

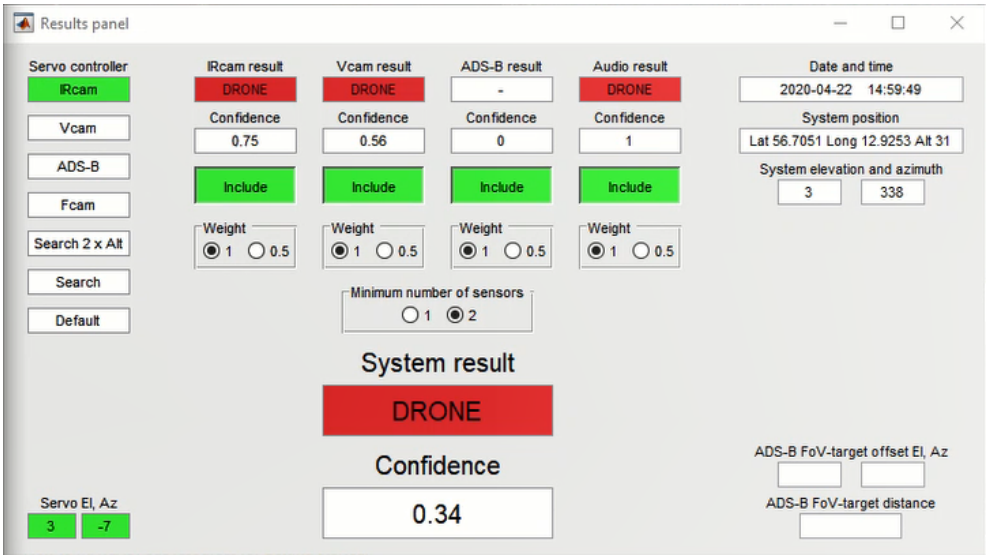


Figure 37: The results panel just after Figure 36. Note the time stamp.

3.1.4 *Acoustic classifier*

Using the 297 clips of the evaluation audio dataset and the Matlab `confusionchart`-function we obtain the confusion matrix shown in Figure 38.

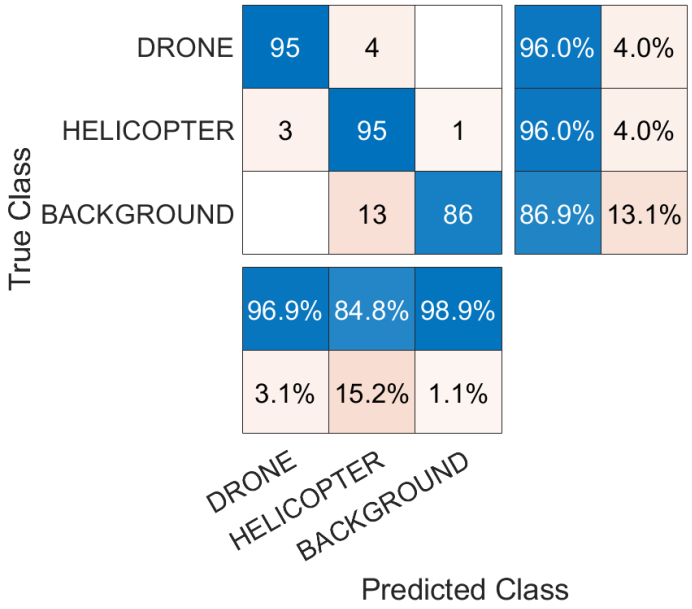


Figure 38: Confusion matrix from the evaluation of the audio classifier

So from this, we can put together Table 17 with the precision and recall results for the different classes and can thereby also calculate the average over all classes.

Table 17: Precision and recall from the evaluation of the audio classifier.

	Class			Average
	DRONE	HELICOPTER	BACKGROUND	
Precision	0.9694	0.8482	0.9885	0.9354
Recall	0.9596	0.9596	0.8687	0.9293

This gives a F1-score of 0.9323, which is higher compared to [26]. The classifier in that paper also utilize MFCC-features, and out of the three network model types tested, the one comprising a LSTM-RNN[1] performs the best with a F1-score of 0.6984. The classification problem in that paper is binary (drone or background).

Another paper utilizing MFCC-features is [27]. Using a Support Vector Machine (SVM) classifier, the authors report a precision of 0.983. Five output classes are used (drone, nature daytime, crowd, train passing and street with traffic), and the classification is based on a one-against-one strategy, hence ten binary SVM classifiers are implemented. The final output label is then computed using the max-wins voting principle. The results are summarized in Table 18.

Table 18: Results from the related work and the audio worker classifier.

Source/Detector	Results			Classes[2]
	Precision	Recall	F1-score	
Kim et al. 2017 [21][3]	0.88	0.83	0.85	D, BG
Jeon et al. 2017 [26]	0.55	0.96	0.70	D, BG
Bernardi et al. 2017 [27]	0.98			D, BG[4]
Audio worker classifier	0.94	0.93	0.93	D, H, BG

Against a drone, the practical range of the acoustic sensor is 35-45 m, depending on how the drone is flying. This is in parity with the 50 m of [23], but far from the 160 m against a F450 drone reported in [28] with its much more complex microphone configuration[5].

The classification range of the system against helicopters has not been tested practically.

1 Recurrent Neural Network
2 Shortened to fit the table. D = Drone, H = Helicopter, BG = Background
3 The results are calculated from the presented confusion matrix
4 Four background classes are defined: Nature daytime, crowd, train passing and street with traffic
5 A 120 element microphone array

3.1.5 *RADAR module*

From the datasheet of the K-MD2 [47], we have that it can detect a person with a Radar Cross Section (RCS) of 1 m^2 up to a distance of 100 m. Since we have from [29] that the RCS of the F450 drone is 0.02 m^2, it is straight forward to calculate that, theoretically, the F450 should be possible to detect up to a distance of

$$\sqrt[4]{\frac{0.02}{1}} \cdot 100 = 37.6\text{m}.$$

Furthermore, given that the micro-doppler echoes from the rotors are 20 dB below that of the drone body, these should be detectable up to a distance of

$$\sqrt[4]{\frac{0.02}{1 \cdot 100}} \cdot 100 = 11.9\text{m}.$$

Practically the F450 drone is detected and tracked by the K-MD2 up to a maximum distance of 24 m, as shown in Figure 39. This is however the maximum recorded distance, and it is observed that the drone is generally detected up to a distance of 18 m.

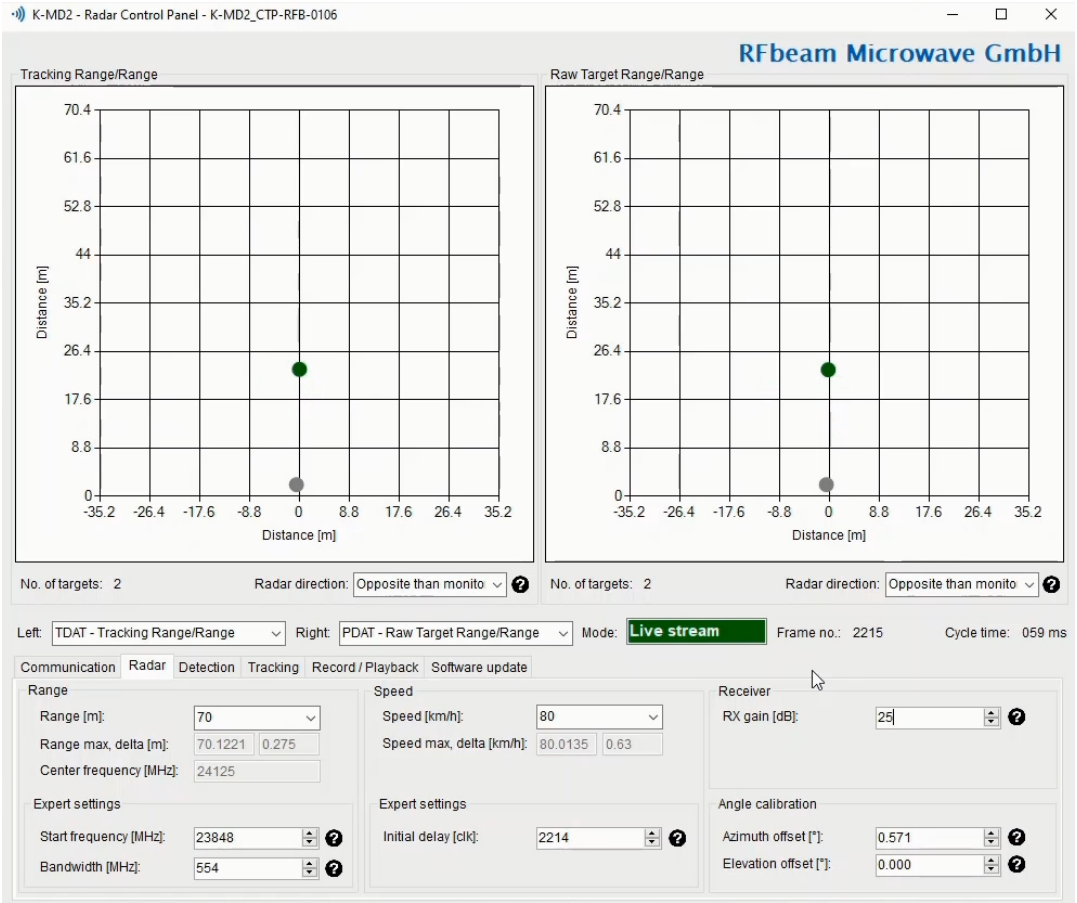


Figure 39: The maximum recorded detection distance of the K-MD 2 radar module against the F450 drone

The micro-doppler signature can also be detected at short distances, as shown in Figure 40. Corresponding range-doppler plots showing the micro-doppler of flying drones can be found in [34] and [35].

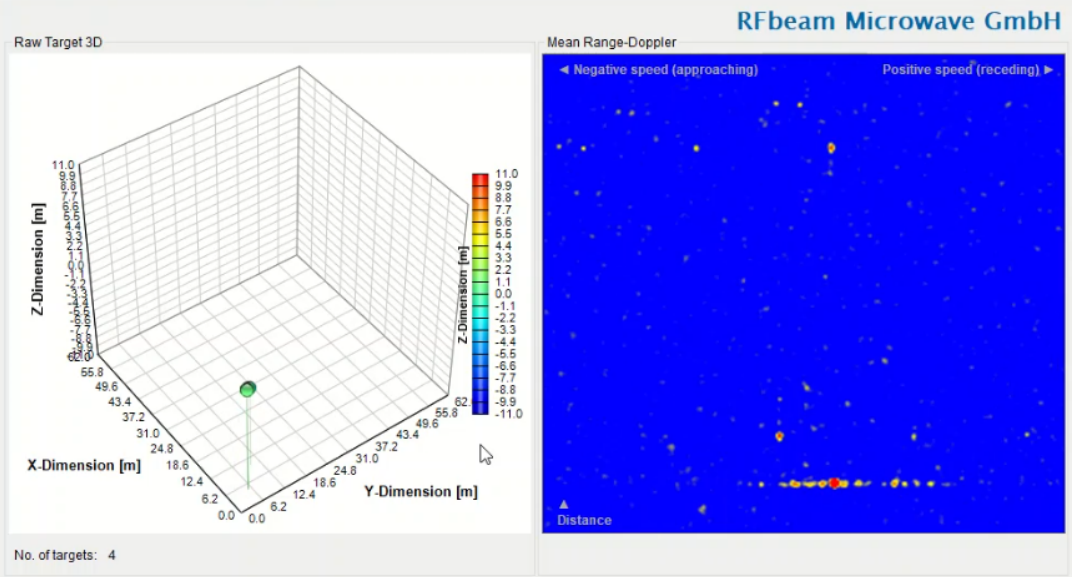


Figure 40: The micro-doppler signature the F450 drone

Compared to the echo of a person walking in front of the radar, as we can see in Figure 41, no such micro-doppler signature is present in that case.

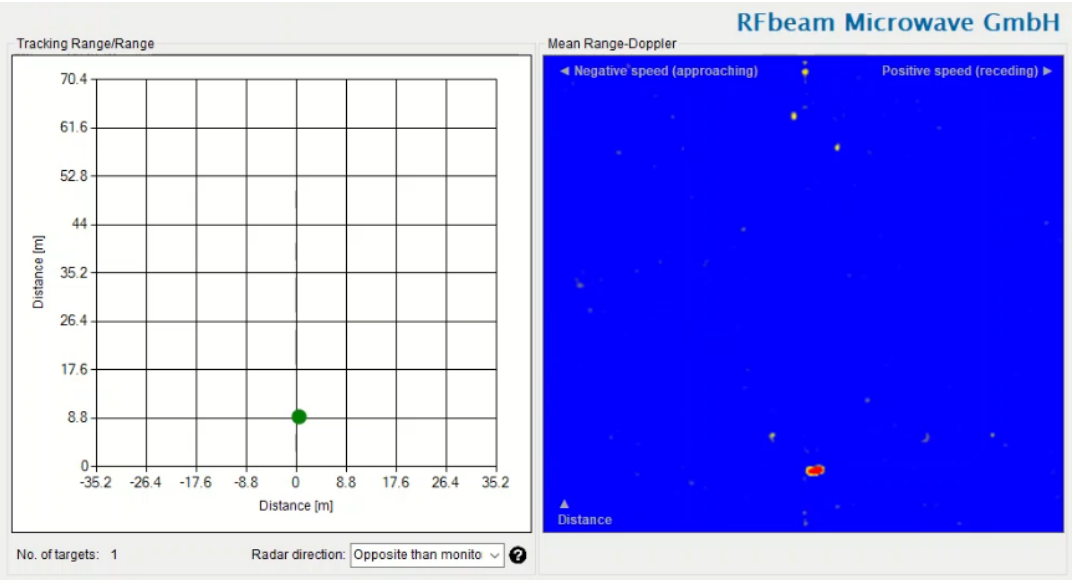


Figure 41: The echo of a person walking in front of the radar module

Due to the short practical detection range of the radar module, it is not included in the drone detection system of this thesis.

3.2 SENSOR FUSION AND SYSTEM PERFORMANCE

The choice of early or late sensor fusion has also been investigated. By early sensor fusion, we here mean to fuse the images from the IR- and video cameras before processing them in a detector and classifier. Late sensor fusion will, in this case, be to combine the output results from the separate detectors running on the IR- and video camera streams, respectively.

When implementing early sensor fusion, the pixel-to-pixel matching of the images turned out to be the biggest issue. Even if this is possible to do in a static scenario as shown in Figure 42, it has turned out not to be a feasible solution against moving objects, such as drones, with the available equipment. This stems from the small but still noticeable difference in latency between the two cameras. This is the motive for the implementation of late sensor fusion in the drone detection system of this thesis.

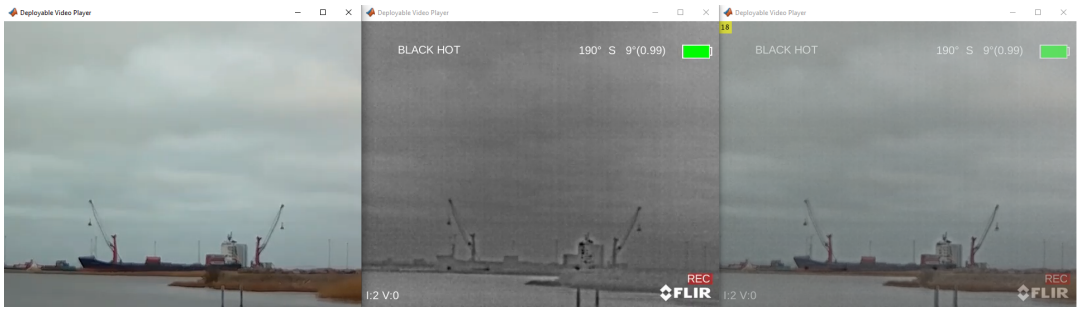


Figure 42: From the left: The video image, the thermal infrared camera image and the fused image.

An interesting sensor fusion method is also found in [20] where the image from the narrow-angle camera is inserted in the image from the wide-angle camera and then processed by a single YOLO-detector. It is a bit unclear how they avoid situations when the inserted image obscure the object found in the wide-angle image.

As described in Section 2.4.1.1, the sensor fusion method of the drone detection system is to utilize the class outputs and the confidence scores of the included sensors, and also to smooth the result over time (about one second). With the dynamical setting available in the GUI it is possible to use not only the or-function, as done in [40], but also more sophisticated variants by setting the number of sensors included and required for detection, including the weights for them.

To implement the sensor fusion using a trained network as in [14] would likely require much more training data, not only on individual sensor level, but especially on a system level. Such amounts of data

have not been possible to achieve within the scope of this thesis.

To evaluate the system level has also turned out to be even harder than expected due to the current situation. For example, all regular flights to and from the local airport have been cancelled, and hence the possibility for a thorough system evaluation against airplanes decreased drastically.

Using the material from the screen recorder[1], it is possible to do a frame-by-frame analysis of a typical drone detection. An example of this is found in Table 19 below.

Table 19: Table from a frame-by-frame analysis of a drone detection during one of the evaluation sessions. The servo column indicates the current servo controlling sensor. The next column specifies if the Fcam motion detector is tracking the drone or not. The respective output labels are shown in the rest of the columns. Note that the system output is more stable and lasts for more frames than the IRcam and Vcam individually, indicating the benefit of the sensor fusion. Figure 20 in Section 2.5 is the third frame from 14:46:18. Since there is no information from the ADS-B receiver in this case, that column has been omitted from the table.

Time	Servo	Fcam	IRcam	Vcam	Audio	System
14:46:17	Fcam	✓			DRONE	
14:46:17	Fcam	✓			DRONE	
14:46:17	IRcam	✓	AIRPLANE		DRONE	DRONE
14:46:17	IRcam	✓	AIRPLANE		DRONE	DRONE
14:46:17	IRcam	✓	AIRPLANE		DRONE	DRONE
14:46:17	IRcam	✓	AIRPLANE		DRONE	DRONE
14:46:17	IRcam	✓		DRONE	DRONE	DRONE
14:46:17	Vcam	✓		DRONE	DRONE	DRONE
14:46:18	Vcam	✓		DRONE	DRONE	DRONE
14:46:18	IRcam	✓	DRONE	DRONE	DRONE	DRONE
14:46:18	IRcam	✓	DRONE	DRONE	DRONE	DRONE
14:46:18	IRcam	✓	DRONE	DRONE	DRONE	DRONE
14:46:18	IRcam	✓	DRONE	DRONE	DRONE	DRONE
14:46:18	IRcam	✓	DRONE	DRONE	DRONE	DRONE
14:46:18	IRcam	✓	DRONE	DRONE	DRONE	DRONE
14:46:18	IRcam	✓	DRONE	DRONE	DRONE	DRONE
14:46:18	Vcam	✓		DRONE	DRONE	DRONE
14:46:19	Vcam	✓		DRONE	DRONE	DRONE
14:46:19	IRcam	✓	HELICOPTER	DRONE	DRONE	DRONE
14:46:19	IRcam	✓		DRONE	DRONE	DRONE
14:46:19	IRcam	✓	BIRD	DRONE	DRONE	DRONE
14:46:19	IRcam	✓	BIRD	DRONE	DRONE	DRONE
14:46:19	IRcam	✓	BIRD	HELICOPTER	DRONE	DRONE
14:46:19	IRcam	✓			HELICOPTER	
14:46:19	Fcam	✓			HELICOPTER	

1 The screen recorder saves images at 10 FPS

As an attempt to measure the efficiency of the sensor fusion, we can consider occasions such as the one described in Table 19, as being a *detection opportunity*. If we define this to be when the drone is continuously observable in the field of view of the IRcam and Vcam, and hence possible for the system (including the audio classifier) to analyse and track, we can find 73 such opportunities in the screen recordings from the evaluation sessions.

The duration of the individual detection opportunities is from just fractions of a second up to 28 seconds. This is highly dependent on how the drone is flying and if the system is able to track the drone or not. We can see that Table 19 describes the frame-by-frame analysis of a detection opportunity lasting for three seconds.

Comparing the system results after the sensor fusion[1] with the output from the respective sensors, we can observe that the system outputs a drone classification at some time in 78% of the detection opportunities. Closest to this is the performance of the Vcam detector that outputs a drone classification in 67% of the opportunities.

It is also possible to look at the system behaviour without a drone flying in front of it. This provides an opportunity to analyse the false detections that the system outputs. Out of the videos from the evaluation session, a ten minutes long section was reviewed frame-by-frame. Table 20 shows the timestamps, sensor types, causes of the false detections, and the resulting output labels. Setting the minimum number of sensors-option to two prevents all of the false detections in the table from becoming false detections on a system level.

1 Having all sensors included with weight 1.0, and the minimum number of sensors set to two

Table 20: Table of false detections appearing in a ten minutes long section of screen recording from an evaluation session, including the type of object causing the false detection.

Time	IRcam	Vcam	Object	Output label
00:31	✓		Cloud	BIRD
01:10	✓		Insect	AIRPLANE
01:46	✓		Cloud	HELICOPTER
02:05	✓		Insect	BIRD
02:25	✓		Cloud	AIRPLANE
02:34	✓		Insect	BIRD
02:58		✓	Cloud	AIRPLANE
03:00	✓		Cloud	BIRD
03:12	✓		Insect	AIRPLANE
03:13	✓		Insect	AIRPLANE
03:38	✓		Cloud	AIRPLANE
04:26		✓	Insect	BIRD
05:11	✓		Cloud	BIRD
05:12	✓		Cloud	AIRPLANE
05:15	✓		Cloud	BIRD
05:48	✓		Cloud	BIRD
06:11	✓		Cloud	BIRD
07:04	✓		Cloud	AIRPLANE
07:34	✓		Cloud	BIRD
07:38	✓		Cloud	BIRD
07:39	✓		Cloud	BIRD
08:11	✓		Cloud	DRONE
08:30	✓		Insect	BIRD
08:47	✓		Cloud	DRONE
08:49	✓		Cloud	DRONE
08:51	✓		Insect	BIRD
09:29		✓	Insect	BIRD
09:30	✓		Cloud	BIRD
09:31	✓		Insect	BIRD
09:33	✓		Cloud	BIRD
09:37	✓		Cloud	AIRPLANE

The false detections caused by insects flying just in front of the sensors are very short-lived. The ones caused by clouds can last longer, sometimes several seconds. Figure 43 shows the false detections of the IRcam at 02:34 and the Vcam at 02:58 from Table 20.

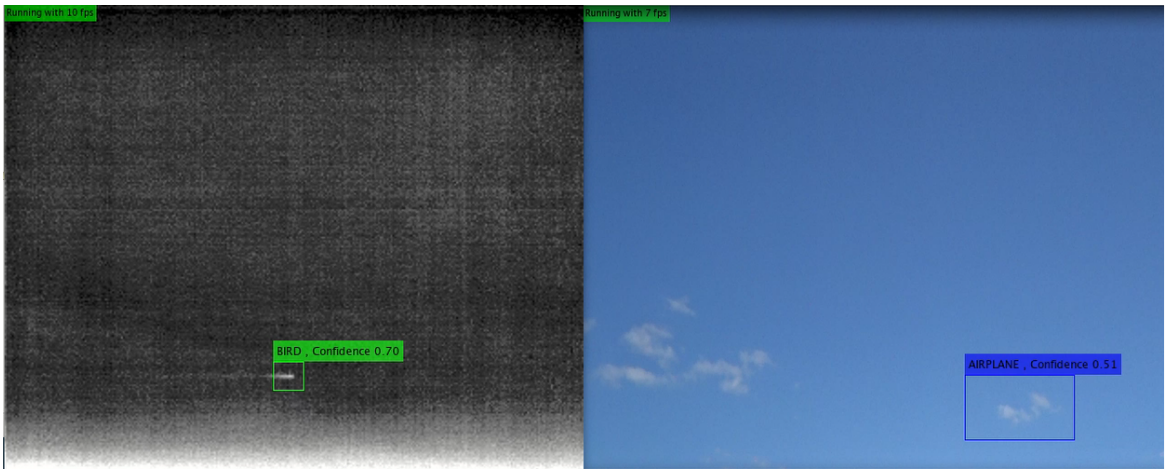


Figure 43: The false detections of the IRcam at 02:34 and the Vcam at 02:58.

As described above, the individual weaknesses observed for the primary sensors are the sensitivity to clouds of the IRcam and the autofocus problem of the Vcam. However, running the whole detection system has also shown that such individual shortcomings can be overcome using a multi-sensor solution. In what follows are some screenshots from the system evaluation sessions, and some interesting observations are pointed out.

These images also show the FPS-performance. The system is able to process 6 FPS or more from all cameras and over 10 processing cycles for the input audio stream per second. Note that the ADS-B solution chosen is the more computationally demanding out of the two alternatives evaluated. To be able to evaluate the system performance a screen recording software has also been running on the computer at the same time as the system software. This setup was a necessity since the drone flying took all the computational power of the thesis author during the evaluation sessions.

As shown by Figure 20, in Section 2.5, when describing the graphical user interface, the ideal situation is of course that all the sensors output the correct classification of the detected target and that the object is tracked by the Fcam. This is however far from the case at all times.

Nevertheless, after the sensor fusion the system output class is observed to be robust as shown in Figure 44 and Figure 45 where the IRcam and Vcam classifies the drone incorrectly but still with a correct system output.

Since the output class depends on the confidence score, the result is sometimes also the opposite, as shown in Figure 46, so that one very confident sensor causes the system output to be wrong. If this turns

out to be frequent, the weight of the sensor can easily be adjusted, or the sensor can be excluded completely from the system result. The time smoothing part in the sensor fusion will also reduce the effect of an occasional misclassification, so that the system output stays correct, as can be seen in Figure 47. Naturally, there are also times when all sensors are wrong, as evident in Figure 48.

To the author's knowledge the inclusion of an ADS-B receiver in a drone detection system has not yet been described in the scientific literature, so it is also of interest to see how this information is utilized. Recall Figure 32, in which the ADS-B information set the pan/tilt platform in the direction of the airplane, so that it was detected by the Vcam at a distance of more than 35000 m.

Looking at Figure 49, we can see that when the airplane comes within 30000 m horizontal distance from the system the ADS-B information will appear in the results panel. At this moment, the sloping distance is 32000 m and the offset between the camera direction and the calculated one is zero. Moreover, since the system has not yet received the vehicle category information at this moment the target is marked with a square in the ADS-B presentation area, and the confidence score of the ADS-B result is 0.75.

The next interesting event, shown in Figure 50, is when the system receives the vehicle category message. To indicate this, the symbol in the ADS-B presentation is changed into a circle, and the confidence is set to one since we are now sure that it is an airplane. At a distance of 20800 m, it is also detected and classified correctly by the IRcam worker, as shown in Figure 51.

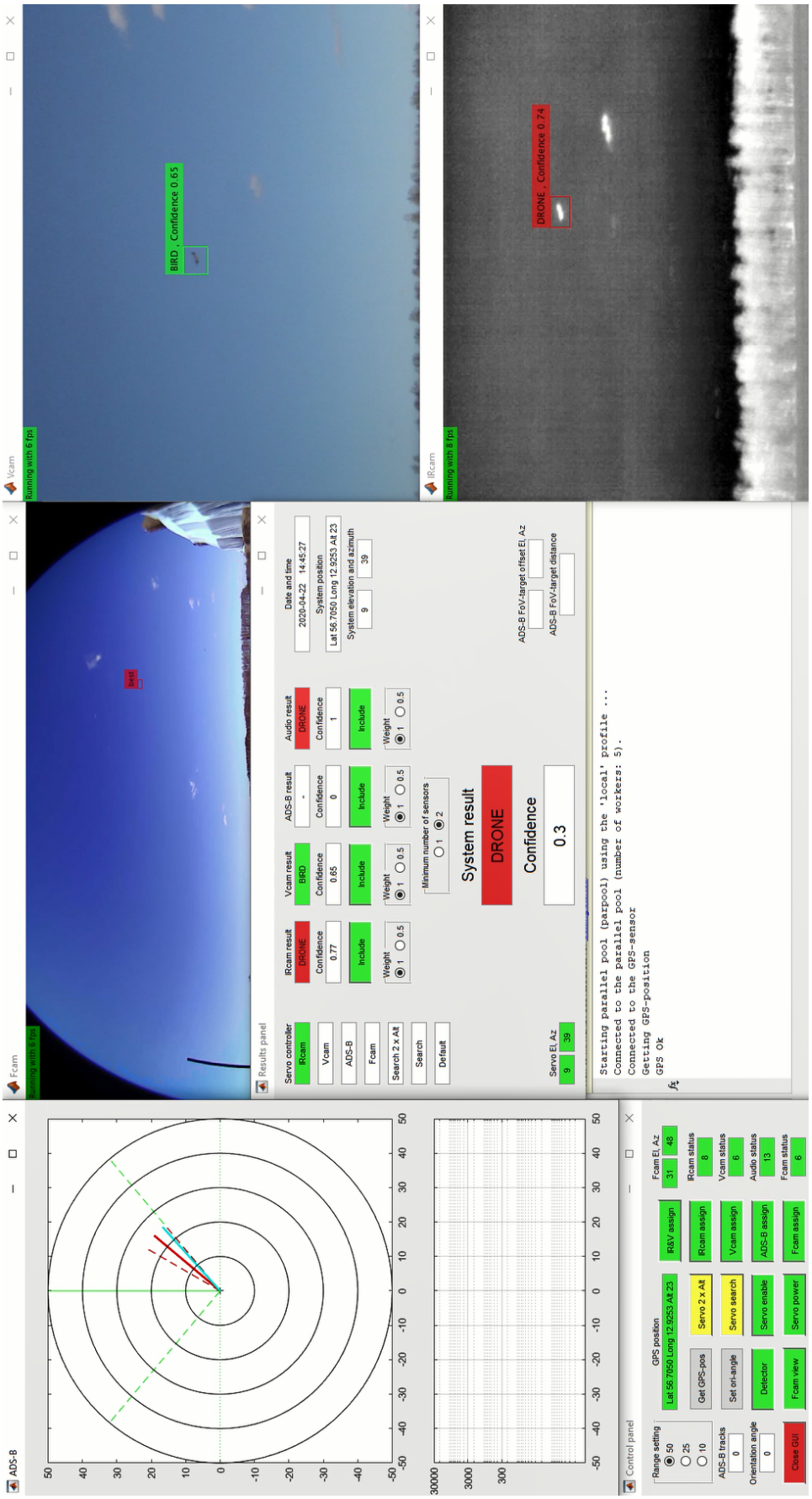


Figure 44: A correct system output even if the Vcam classifies the drone as being a bird.

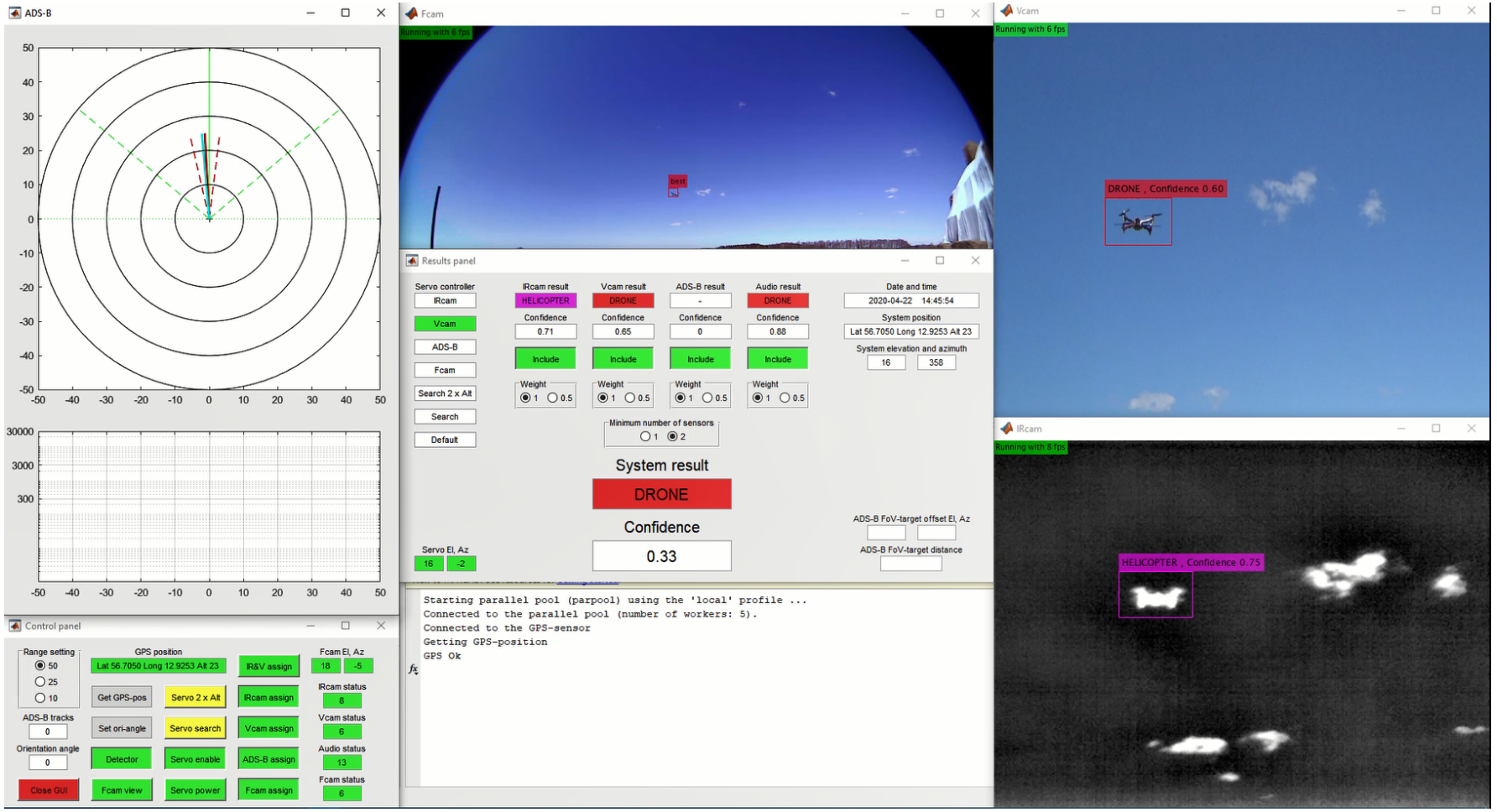


Figure 45: A correct system output even if the IRcam classifies the drone as being a helicopter.

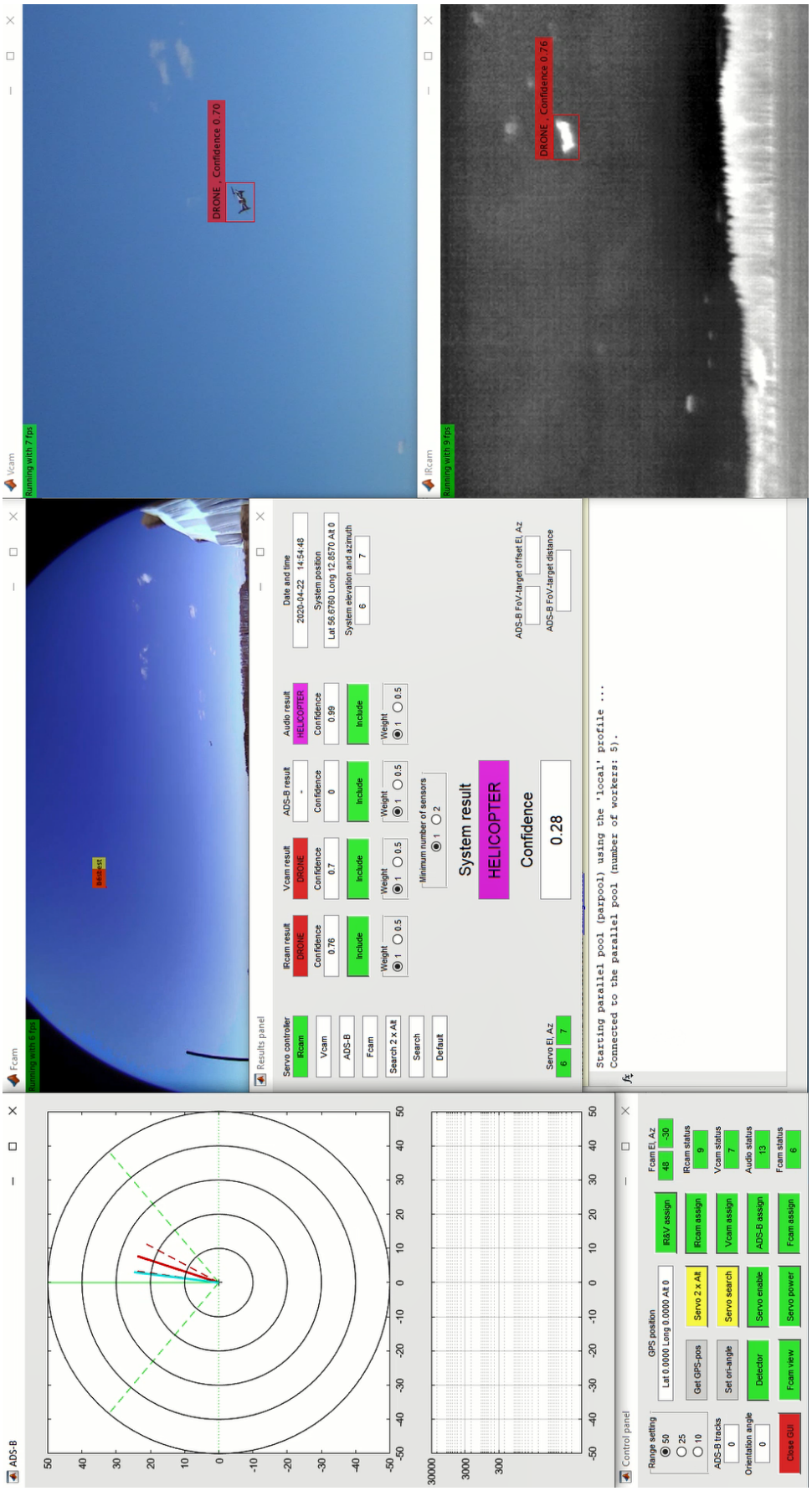


Figure 46: The high confidence score of the audio classifier cases the system output to be incorrect just as the audio output.

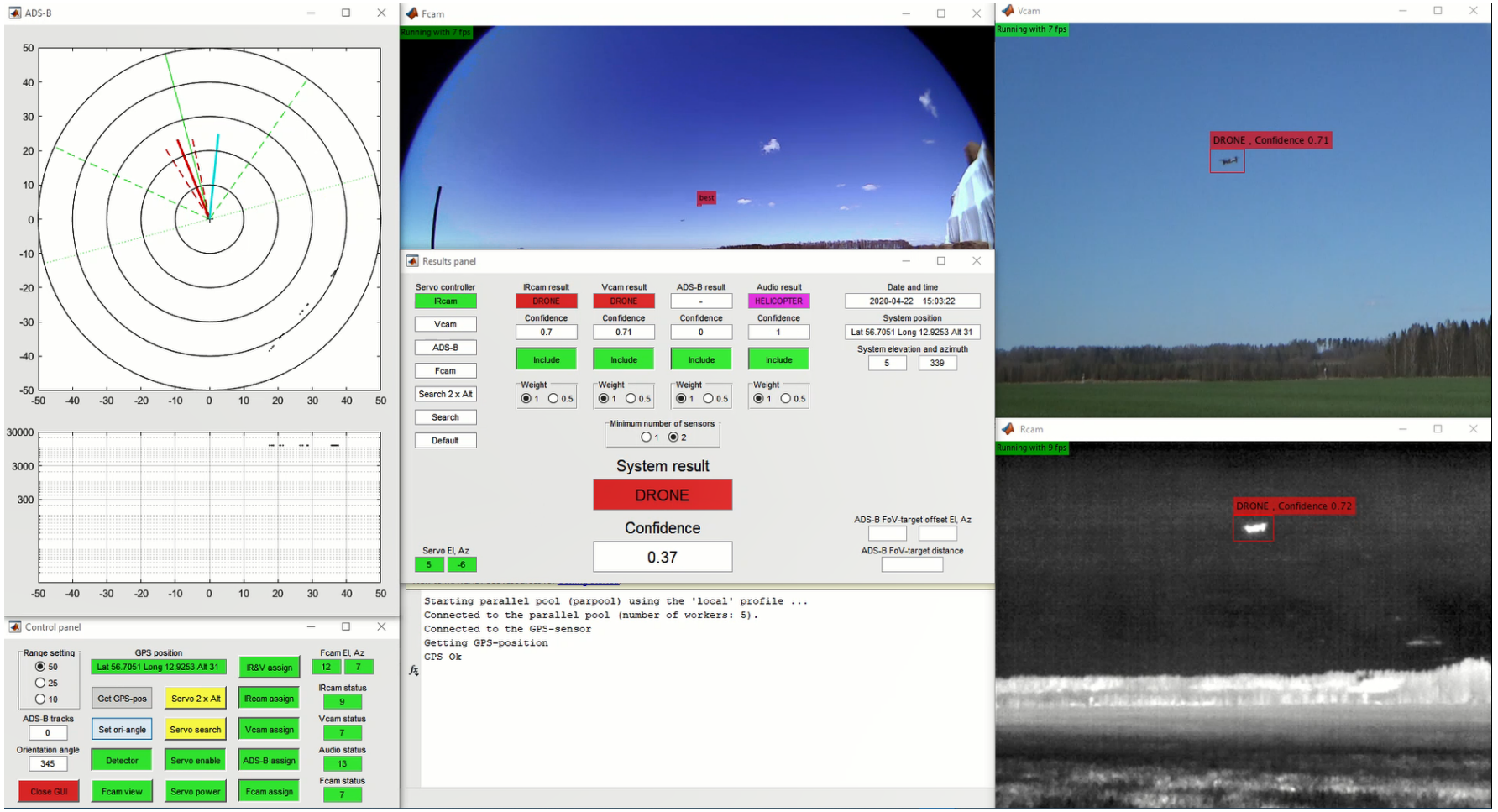


Figure 47: The time smoothing part of the sensor fusion reduces the effect of an occasional misclassification even if that has a high confidence scores.

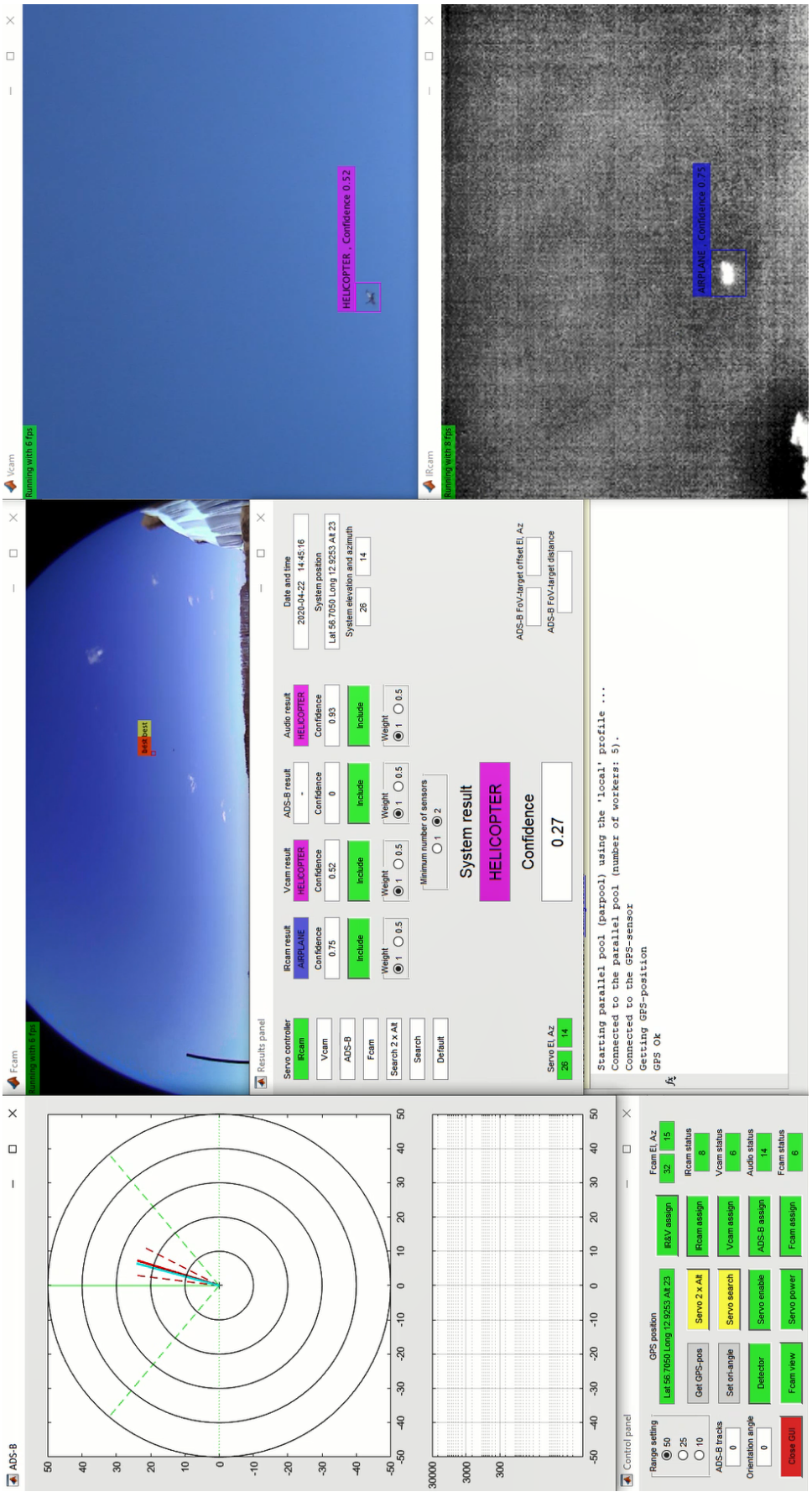


Figure 48: The drone is misclassified by several sensors at the same time.

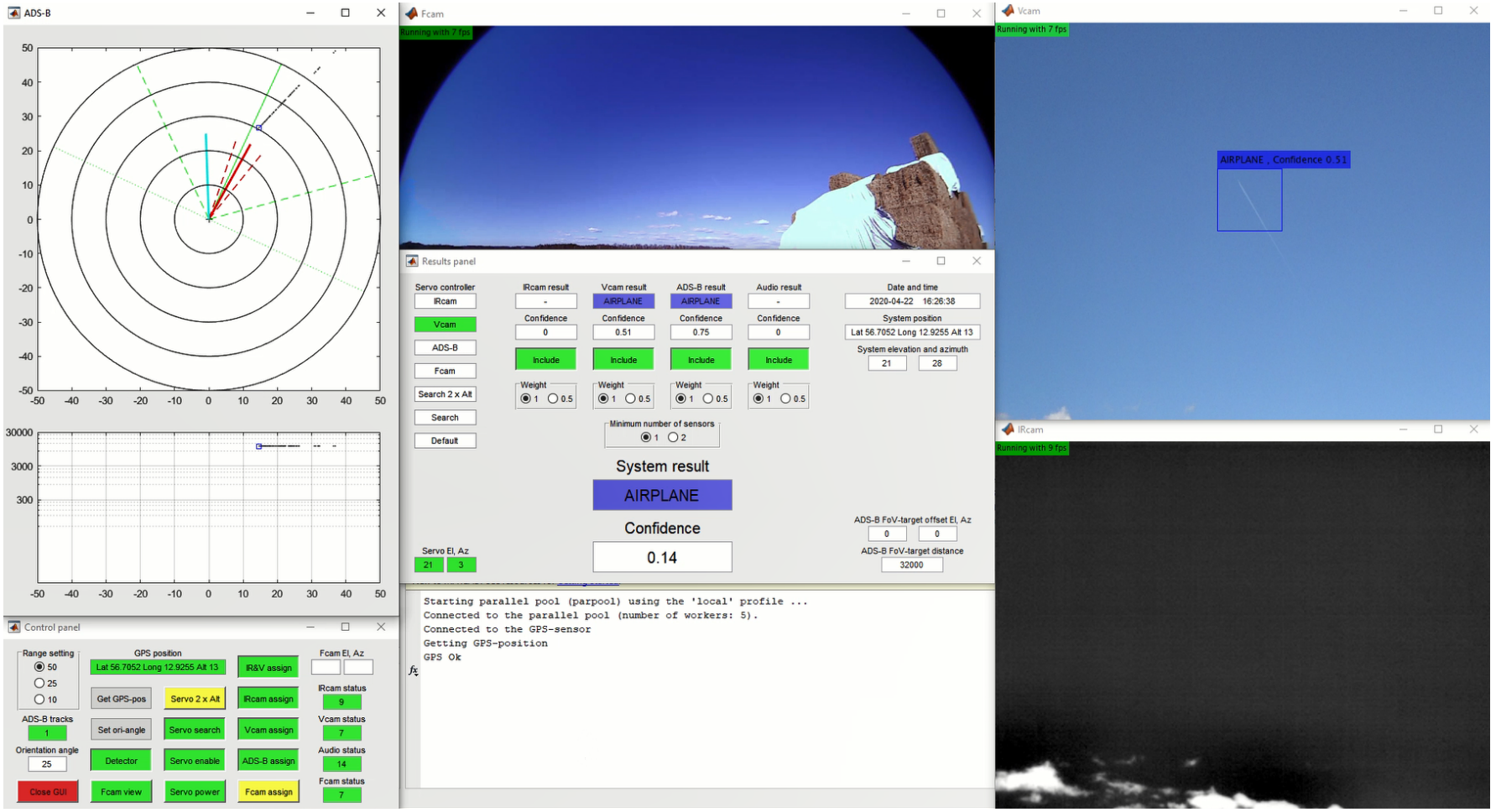


Figure 49: When the airplane is within 30000 m horizontal distance the ADS-B information is presented in the results panel.

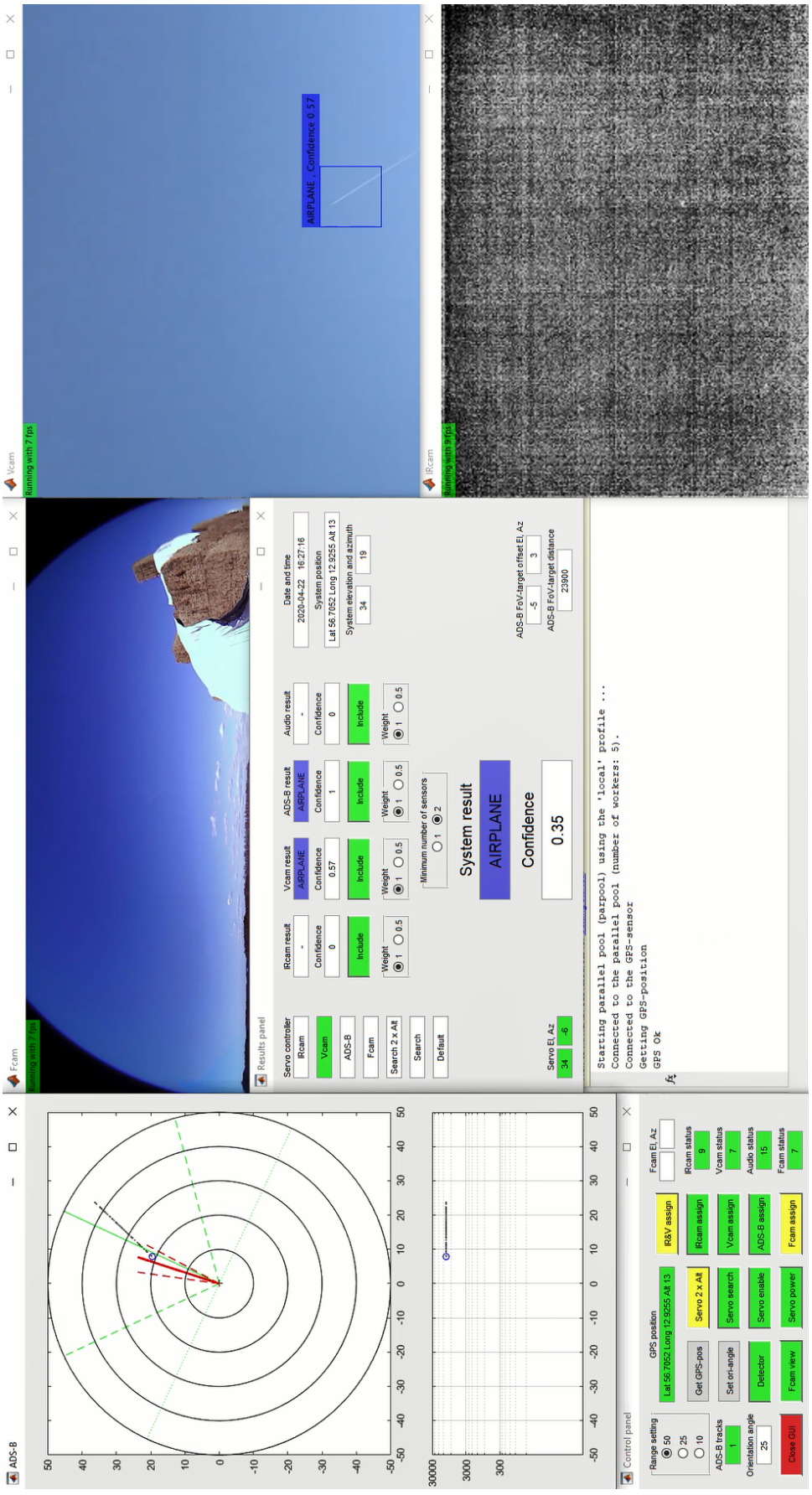


Figure 50: The vehicle category message has been received by the system, so the confidence for the airplane classification is set to 1. The airplane is now at a distance of 23900m.

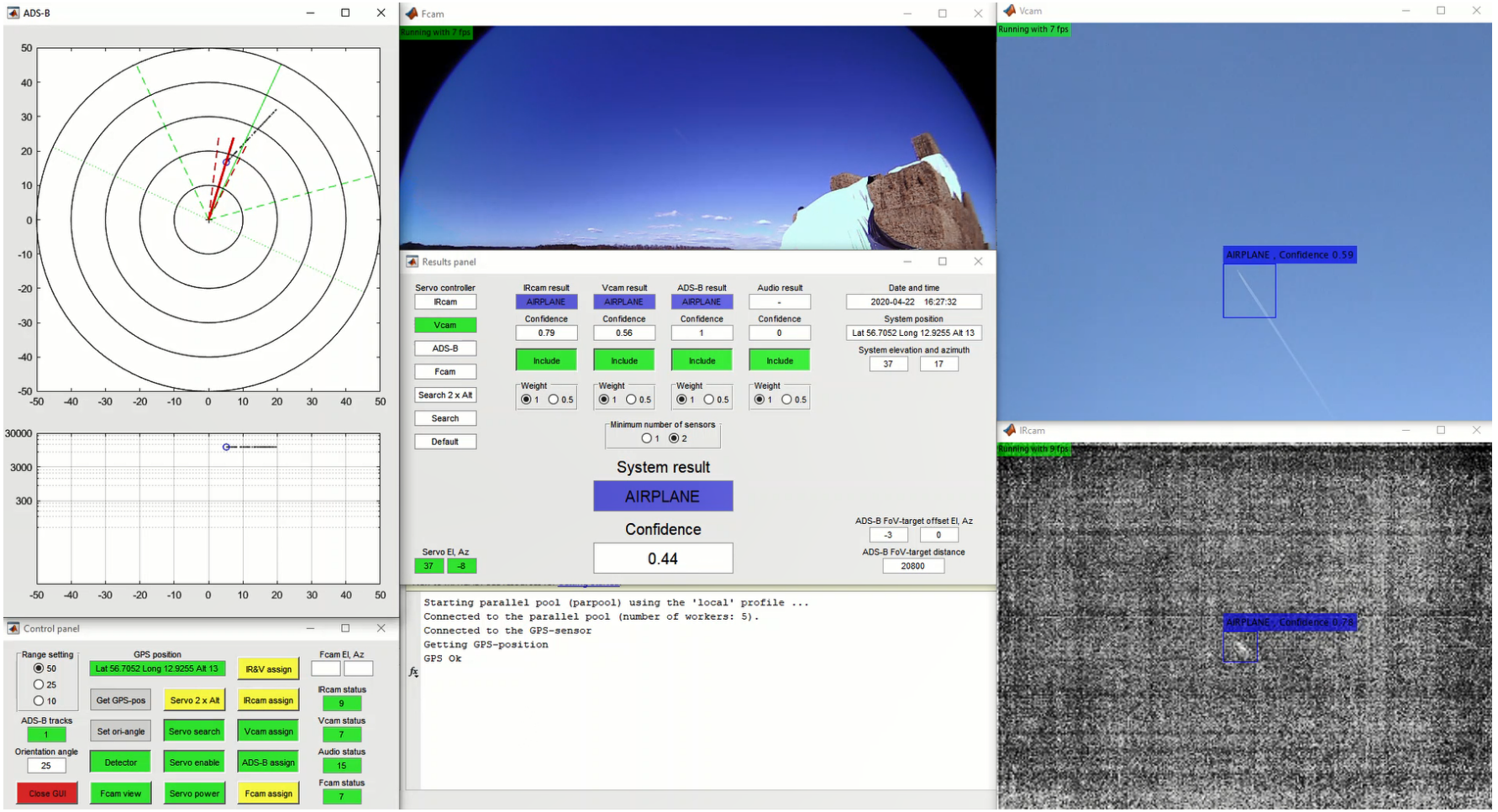


Figure 51: The airplane is detected by the IRcam worker at a distance of 20800 m.

3.3 DRONE DETECTION DATASET

Since one of the objectives of this thesis is to collect, compose and publish a multi-sensor dataset for drone detection this is also described as part of the results chapter.

As mentioned in Section 2.4.1.5, the audio dataset consists of a total of 90 ten-seconds clips, 30 from each class, in wav-format with a sampling frequency of 44100 Hz. The clips are not annotated in any other way than the filenames themself, e.g. DRONE_001.wav. Included in the background category are also system-specific sounds, in this case, the sounds of the pan/tilt platform servos and the power relay board.

The video dataset is more extensive and consists of a total of 650 video clips, out of which 365 are IR and 285 are visible video. Each of these has a duration of about 10 seconds[1]. The ambition has been to have a proper distribution between sensors, target types and distance bins. The distribution is presented in Table 21 and Table 22.

Table 21: The distribution of the 365 IR videos.

Bin	Class			
	AIRPLANE	BIRD	DRONE	HELICOPTER
Close	9	10	24	15
Medium	25	23	94	20
Distant	40	46	39	20

Table 22: The distribution of the 285 visible videos.

Bin	Class			
	AIRPLANE	BIRD	DRONE	HELICOPTER
Close	17	10	21	27
Medium	17	21	68	24
Distant	25	20	25	10

All videos are in mp4-format and comprise in total 203328 annotated images. The IR videos have a resolution of 320x256 pixels and are all recorded so that no further actions are needed before the YOLOv2-detector, hence, all image processing operations described in Section 2.4.1.2 are already implemented before the video is saved.

1 The average duration of the videos is 10.4 s

This is done to ensure that the objects are clearly visible when annotating the videos. The visible videos have a resolution of 640x512 pixels.

The filenames of the videos start with the sensor type, followed by the target type and a serial number, e.g. IR_DRONE_001.mp4. The annotation of the respective clip has the additional name LABELS, e.g IR_DRONE_001_LABELS.mat. These files are Matlab `GroundTruth`-objects, and using the Matlab video labeller app, the videos and respective label-file can easily be opened, inspected and even edited. If the dataset is to be used in another development environment, the label-files can be opened in Matlab, and the content can be copy-pasted into Excel and saved in the desired format, for example, .csv. Importing .mat-files into the Python-environment can also be done using the `scipy.io.loadmat`-command.

To retrieve the images from the videos and form a dataset of the individual images the `selectLabels`- and `objectDetectorTrainingData`-functions are recommended.

It has not been possible to film all types of suitable targets, so, as mentioned in Section 2.6, some clips are taken from longer aircraft videos downloaded from [59]. This is a total of 49 clips (11 airplane and 38 helicopter clips) in the visible video dataset.

Since the distance bin information of the clip is not included in the filename, there is also an associated excel-sheet where this is shown in a table. This table also contains information about exact drone type, and if the clip comes from the Internet or not.

The greatest sensor-to-target distance for a drone in the dataset is 200 m. Within the dataset, there are also eight clips (five IR and three visible videos) with two drones flying at the same time. Figure 52 shows the IRcam worker running in stand-alone mode on one of these videos. The clip chosen is not part of the training dataset. Note also the FPS-performance.

Figure 52: Two drones detected and classified correctly by the IRcam worker.

4

DISCUSSION

An unforeseen problem occurring when designing the system was actually of mechanical nature. Even though the system uses a pan/tilt platform with ball-bearings and very high-end titanium gear digital servos, the platform was observed to oscillate in some situations. This phenomena was mitigated by carefully balancing the tilt platform and by introducing some friction in the pivot point of the pan segment. It might also be the case that such problems could be overcome using a servo programmer[1]. Changing the internal settings[2] of the servos can also increase their maximum ranges from 90° to 180°. This would extend the volume covered by the IRcam and Vcam, so that all targets tracked by the Fcam could be investigated, not just a portion of them, as now.

It is also observed, regarding the sensors, that there is a lower limit of around 5 FPS where the system becomes so slow so that the ability to track flying objects is lost. All actions taken by the system have to be well balanced, and just such a simple thing as plotting the ADS-B panel with a higher frequency than necessary can cause a drop in the FPS-rate. Such problems can be overcome by the use of more than one computational resource.

One thing not explored in this thesis is the use of the Fcam together with the audio classifier as a means to output the position and label of a system detection. Implementing a YOLOv2-detector on the Fcam could also be considered. However, a dataset for the training of this must either be collected separately or by skewing images from the visible video dataset, so that the distortion of the Fcam image is matched. Neither is the performance of the audio classifiers performance as a function of sensor-to-target distance explored in the same way as the IR and visible sensors.

There is also a need for a better method to evaluate the whole drone detection system. To assess the performances of the individual sensors is pretty straight forward, and in most cases, there are also numerous results to relate to, except for the IR-sensor, at least at the moment. On a system level, however, the results are very sparse, making any comparison difficult.

1 This is not the same as the servo controller
2 Also using the servo programmer

Due to the very short practical detection range, the RADAR module was unfortunately not included in the final system setup. Having a RADAR with a useful range would have contributed significantly to the system results since it is the only one of the available sensors being able to measure the distance to the target efficiently. Another way to increase the efficiency of the system could also be to exploit the temporal dimension better, i.e. to use the flight paths and the behaviour of the object to better classify them.

As we have seen, most of the false detections are caused by either insects or clouds. Adding these classes to the dataset might be a way to overcome this annoyance. Using three different quadcopter drones makes the system effective against such drones. Extending the drone dataset by including also hexacopters, octocopters, fixed-wing and single rotor drones would be necessary before using the system in a real world application.

It would also be of interest to implement YOLOv3 in the system, since it is more efficient in detecting small objects according to [20], and then compare the performance to the results reported in this thesis.

Where there are measures, there are also countermeasures. Looking at the drone market today, it is easy to find equipment purposely built to avoid detection by systems such as the one in this thesis. This can range from just the installation to low noise propellers [63], to drones disguised as birds, as shown in Figure 53. Testing the thesis system against such drones would be appealing.

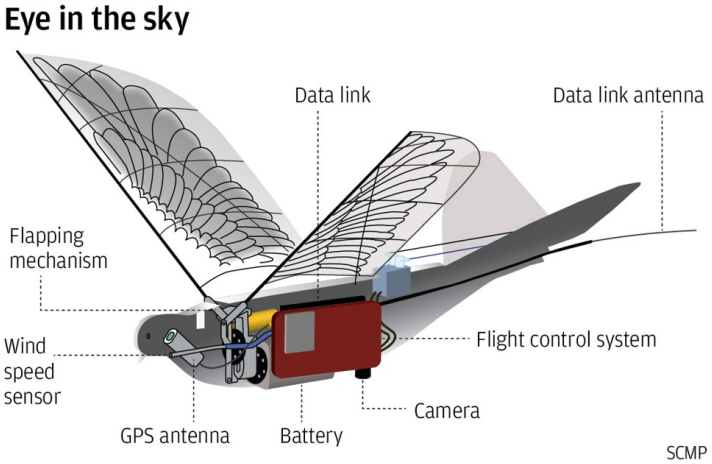


Figure 53: A drone from the "spy birds" programme lead by Song Bifeng, professor at the Northwestern Polytechnical University in Xian. From [64].

Likewise, it would be interesting to test the system against a drone equipped with a transponder to see the performance of the ADS-B worker at a short distance. Such transponders, as exemplified in Figure 54, weighs as little as 20 grams [65]. However, the price[1] of such equipment is still an issue for the non-professional drone user.

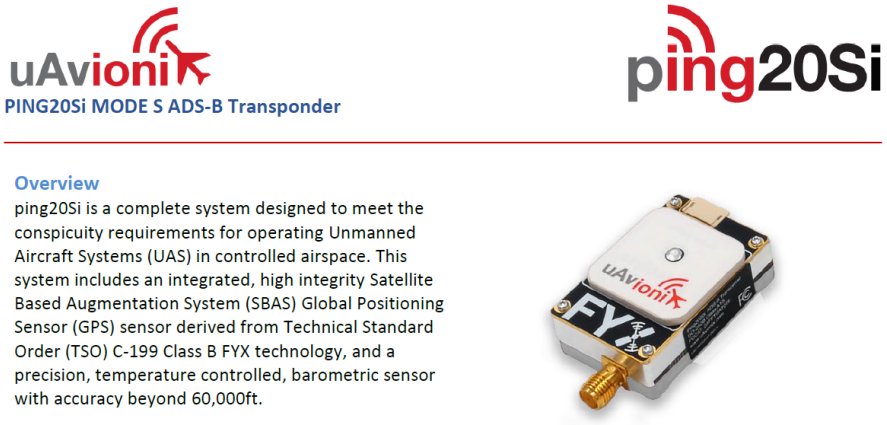


Figure 54: The ping20Si Mode S ADS-B transponder. From [65].

Since ADS-B information is included in the system, this could also be implemented in the form of ground truth for an online learning feature. Images of ADS-B targets could be saved, and on regular intervals, the detectors would be retrained using these additional images. To further enlarge the training data set all images of detected objects could be saved and annotated manually at a later stage.

The work done in this thesis is also applicable to other areas. One such that immediately springs to mind, is road traffic surveillance. Except for the ADS-B receiver, all other parts and scripts could be adopted and retrained to detect and track pedestrians or just a specific vehicle type, e.g. motorcycles.

1 2790 euro as of 2020-05-10

5 CONCLUSIONS

The increased use of drones and the raising concerns of safety and security issues following from this highlights the need for efficient and robust drone detection systems.

This thesis explores the possibilities to design and build a multi-sensor drone detection system utilizing state-of-the-art machine learning techniques, and sensor fusion. The performance of the individual sensors are evaluated in terms of F1-score and mean average precision (mAP).

The results confirm that general machine learning techniques can be applied to input data from infrared sensors making them well suited for the drone detection task. The infrared detector achieves an F1-score of 0.7601, showing similar performance as the visible video detector with an F1-score of 0.7849. The audio classifier achieves an F1-score of 0.9323.

Besides the analysis of the feasibility of an infrared sensor, this thesis expands the number of target classes utilized in the detectors compared to the related papers. The thesis also comprise a novel investigation of the detection performance as a function of sensor-to-target distance, with a distance bin division derived from the Detect, Recognize and Identify (DRI) requirements based on the Johnson criteria.

The proposed sensor fusion makes the detection and classification more robust than that of any of the sensors individually. It also shows the efficiency of sensor fusion as a means to mitigate false detections. To the author's knowledge, the system is also the first to explore the benefits of including ADS-B data to better separate targets prone to be mistaken for drones.

Due to the lack of a publicly available dataset, the other main concern of this thesis was to contribute with such a multi-sensor dataset. This dataset is especially suited for the comparison of infrared and visible video detectors due to the similarities in conditions and target types in the set.

The dataset is made up of 650 video clips containing a total of 203328 annotated images, and has four target classes: Drones, birds,

airplanes and helicopters. Furthermore, the dataset has a distance bin division, making it possible to explore the sensor-to-target distance performance of the detector. There is also an audio dataset with 900 seconds of acoustic data from drones, helicopters and backgrounds.

Further research could be to implement a distance estimation function based on the output from the detectors. Such research could also include the investigation of distributed sensors and to further make use of the temporal dimension to separate drones form other targets based on the behaviour over time as they are tracked by the system.

This thesis also discloses the need for a benchmark or standard when it comes to the evaluation of machine learning and sensor fusion applications on a system level.

The drone detection dataset is available at:
https://github.com/DroneDetectionThesis/Drone-detection-dataset
The author of this thesis can be contacted at:
DroneDetectionThesis@gmail.com

BIBLIOGRAPHY

[1] J. Sanfridsson et al. Drone delivery of an automated external defibrillator – a mixed method simulation study of bystander experience. *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine*, 27, 2019.

[2] M. Innocente and P. Grasso. Self-organising swarms of firefighting drones: Harnessing the power of collective intelligence in decentralised multi-robot systems. *Journal of Computational Science*, 34:80–101, 2019.

[3] The World Air Sports Federation FAI. World drone racing championship. https://www.fai.org/world-cups/drone-racing-2020, April 2020.

[4] Aviation International News. What happens when a drone hits an airplane wing? About the University of Dayton research institute. https://www.youtube.com/watch?v=QH0V7kp-xg0, April 2020.

[5] B. Taha and A. Shoufan. Machine learning-based drone detection and classification: State-of-the-art in research. *IEEE Access*, 7:138669–138682, 2019.

[6] Google Trends. https://trends.google.com/trends/explore?date=all&q=dronedetection, April 2020.

[7] S. Samaras et al. Deep learning on multi sensor data for counter UAV applications — a systematic review. *Sensors*, 19(4837), 2019.

[8] S. Boddhu et al. A collaborative smartphone sensing platform for detecting and tracking hostile drones. *Proceedings Volume 8742, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IV*, 2013.

[9] Guard From Above BV. https://guardfromabove.com/, February 2020.

[10] I. Güvenc et al. Detection, tracking, and interdiction for amateur drones. *IEEE Communications magazine*, 56:75–81, 2018.

[11] P. Andrasi et al. Night-time detection of UAVs using thermal infrared camera. *Transportation Research Procedia*, 28:183–190, 2017.

[12] Y. Wang et al. Towards visible and thermal drone monitoring with convolutional neural networks. *APSIPA Transactions on Signal and Information Processing*, 8, 2017.

[13] M. Wu et al. Real-time drone detection using deep learning approach. *Machine Learning and Intelligent Communications. MLICOM 2018. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 251:22–32, 2018.

[14] D. Tzovaras et al. Multimodal deep learning framework for enhanced accuracy of UAV detection. *LNCS*, 11754:768–777, 2019.

[15] J. Park et al. A comparison of convolutional object detectors for real-time drone tracking using a PTZ camera. *17th International Conference on Control, Automation and Systems (ICCAS 2017)*, pages 696–699, 2017.

[16] J. Redmon and A. Farhandi. YOLO9000: Better, faster, stronger. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] H. Liu et al. A drone detection with aircraft classification based on a camera array. *IOP Conference Series: Materials Science and Engineering*, 322(052005), 2018.

[18] M. Saqib et al. A study on detecting drones using deep convolutional neural networks. *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[19] C. Aker and S. Kalkan. Using deep networks for drone detection. *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[20] E. Unlu et al. Deep learning-based strategies for the detection and tracking of drones using several cameras. *IPSJ Transactions on Computer Vision and Applications volume*, 11, 2019.

[21] J. Kim et al. Real-time UAV sound detection and analysis system. *IEEE Sensors Applications Symposium (SAS)*, 2017.

[22] N. Siriphun et al. Distinguishing drone types based on acoustic wave by IoT device. *22nd International Computer Science and Engineering Conference (ICSEC)*, 2018.

[23] S. Park et al. Combination of radar and audio sensors for identification of rotor-type unmanned aerial vehicles (UAVs). *IEEE SENSORS*, 2015.

[24] M. Anwar et al. Machine learning inspired sound-based amateur drone detection for public safety applications. *IEEE Transactions on Vehicular Technology*, 68:2526–2534, 2019.

[25] H. Liu et al. Drone detection based on an audio-assisted camera array. *IEEE Third International Conference on Multimedia Big Data (BigMM)*, 2017.

[26] S. Jeon et al. Empirical study of drone sound detection in real-life environment with deep neural networks. *25th European Signal Processing Conference (EUSIPCO)*, 2017.

[27] A. Bernardini et al. Drone detection by acoustic signature identification. *Electronic Imaging: Imaging and Multimedia Analytics in a Web and Mobile World*, pages 60–64, 2017.

[28] J. Busset et al. Detection and tracking of drones using advanced acoustic cameras. *Proc. SPIE 9647, Unmanned/Unattended Sensors and Sensor Networks XI; and Advanced Free-Space Optical Communication Techniques and Applications*, 2015.

[29] J. Patel et al. Review of radar classification and RCS characterisation techniques for small UAVs or drones. *IET Radar, Sonar and Navigation*, 12:911–919, 2018.

[30] A. Herschfelt et al. Consumer-grade drone radar cross-section and micro-doppler phenomenology. *IEEE Radar Conference (RadarConf)*, 2017.

[31] J. Gong et al. Interference of radar detection of drones by birds. *Progress In Electromagnetics Research M*, 81:1–11, 2019.

[32] L. Fuhrmann et al. Micro-doppler analysis and classification of UAVs at Ka band. *18th International Radar Symposium (IRS)*, 2017.

[33] S Björklund. Target detection and classification of small drones by boosting on radar micro-doppler. *15th European Radar Conference (EuRAD)*, 2018.

[34] J. Drozdowicz et al. 35 GHz FMCW drone detection system. *17th International Radar Symposium (IRS)*, 2016.

[35] S. Rahman and D. Robertson. Radar micro-doppler signatures of drones and birds at K-band and W-band. *Scientific Reports*, 8, 2018.

[36] S. Birnbach et al. Wi-fly?: Detecting privacy invasion attacks by consumer drones. *NDSS Symposium*, 2017.

[37] A. Shorten et al. Localisation of drone controllers from RF signals using a deep learning approach. *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, pages 89–97, 2018.

[38] M. Ezuma et al. Detection and classification of UAVs using RF fingerprints in the presence of interference. *IEEE Open Journal of the Communications Society*, 1:60–76, 2019.

[39] B. Kim et al. V-RBNN based small drone detection in augmented datasets for 3D LADAR system. *Sensors*, 18(11), 2018.

[40] X. Shi et al. Anti-drone system with multiple surveillance technologies: architecture, implementation, and challenges. *IEEE Communications Magazine*, pages 69–74, April 2018.

[41] C. Reiser. Github page. https://github.com/creiser/drone-detection, January 2020.

[42] SafeShore consortium. http://safeshore.eu/dataset/, January 2020.

[43] A. Schumann et al. Deep cross-domain flying object classification for robust uav detection. *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 68, 2017.

[44] A. Coluccia et al. Drone-vs-bird detection challenge at IEEE AVSS2019. *16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2019.

[45] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computations*, 9(8):1735–1780, 1997.

[46] W. Grimson and C. Stauffer. Adaptive background mixture models for real-time tracking. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:2246–2252, 1999.

[47] RFbeam Microwave GmbH. Datasheet of K-MD2. https://www.rfbeam.ch/files/products/21/downloads/Datasheet_K-MD2.pdf, November 2019.

[48] M. Mostafa et al. Radar and visual odometry integrated system aided navigation for uavs in gnss denied environment. *Sensors*, 18(9), 2018.

[49] J. Redmon et al. You Only Look Once unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[50] MathWorks. https://se.mathworks.com/help/vision/examples/create-yolo-v2-object-detection-network.html, October 2019.

[51] MathWorks. https://se.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html, October 2019.

[52] MathWorks. https://se.mathworks.com/help/audio/examples/classify-gender-using-long-short-term-memory-networks.html, October 2019.

[53] M. Robb. Github page. https://github.com/MalcolmRobb/dump1090, December 2019.

[54] Flightradar24. https://www.flightradar24.com, April 2020.

[55] Everdrone AB. https://www.everdrone.com/, April 2020.

[56] YouTube channel "Jake's wings". https://www.youtube.com/watch?v=XoQ79hcZrb8, April 2020.

[57] Pololu. Maestro servo controller, Windows drivers and software (release 130422). https://www.pololu.com/product/1352/resources, November 2019.

[58] TSFS 2017:110 Transportstyrelsens föreskrifter om obemannade luftfartyg. 2017.

[59] YouTube channel "VIRTUAL AIRFIELD operated by SK678387". https://www.youtube.com/channel/UCx-PY5Q1Z5sJOQ9e8wvwvWQ, January 2020.

[60] Infiniteoptics. Whitepaper on thermal DRI. https://www.infinitioptics.com/sites/default/files/attachments/Infiniti%20DRI%20Whitepaper.pdf, April 2020.

[61] P. Chevalier. On the specification of the DRI requirements for a standard NATO target. *Researchgate publication*, 2016.

[62] C. Craye and S. Ardjoune. Spatio-temporal semantic segmentation for drone detection. *16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2019.

[63] Master Airscrew from the Windsor Propeller LLC. https://www.masterairscrew.com/products/dji-mavic-air-stealth-upgrade-propellers-x4-black, May 2020.

[64] South China Morning Post. https://www.scmp.com/news/china/society/article/2152027/china-takes-surveillance-new-heights-flock-robotic-doves-do-they, May 2020.

[65] uAvioni. Datasheet of ping20Si. https://uavionix.com/downloads/ping20s/Ping20Si-DataSheet.pdf, May 2020.

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se