Vision-based Aircraft Detection and Tracking for Detect-and-Avoid

Sourish Ghosh

CMU-RI-TR-22-11

May 5, 2022

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Sebastian Scherer, *chair*
Kris Kitani
Cherie Ho

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Detect-and-Avoid (DAA) capabilities are critical for autonomous operations of small unmanned aircraft systems (sUAS). Traditionally DAA systems for large aircraft have been ground and radar-based. Due to the size, weight, and power (SWaP) constraints of sUAS, current DAA systems rely mainly on vision-based sensors and ADS-B (Automatic Dependent Surveillance-Broadcast) transponders. However, not all flying objects have transponders. Therefore, a vision-based DAA capability needs to exist for safe low-altitude autonomous flight.

In this work, we present the vision-based DAA problem, particularly the problem of aircraft detection and tracking. At long distances, the visual appearance of planes and helicopters is usually tiny in the context of the whole image. The problem is detecting tiny objects in high-resolution videos taken from a moving camera. Historically, this problem has been tackled using a multi-stage image processing pipeline involving: (1) ego-motion estimation, (2) background/foreground detection using morphological operations, and (3) tracking using temporal filtering. With the advent of deep learning, modern approaches rely on learning-based object detection methods. However, traditional object detection approaches typically do not scale well for this task due to real-time performance constraints.

Our approach to solving this problem follows a two-stage pipeline: (1) ego-motion estimation and (2) detection and tracking. Both of these stages are fully-convolution neural networks that can scale to large resolution inputs. They are trained on a labeled dataset released by Amazon Prime Air containing 3.3M+ images of airplanes, helicopters, drones, and other flying objects. We also developed our own aircraft data collection systems and designed a custom vision-based DAA payload for in-flight encounters. Through empirical evaluation of real-world data, our approach is compared with two baseline detection and tracking architectures and is shown to be superior. Analyzing our quantitative results in the context of DAA industry standard (ASTM F3442/F3442M - 20) we also show that the proposed method can satisfy the visual DAA surveillance requirements for certain classes of unmanned aircraft with a minimum cruise speed of 60-90kts, minimum turn rate of 21-31deg/s, and a minimum climb rate of 250-500ft/min.

Acknowledgments

I would like to begin by thanking my advisor, Prof. Sebastian Scherer, for allowing me to join the AirLab in 2019 and thereafter giving me the responsibility to lead a new lab project (thus forming the basis of this thesis) in which I had minimal prior experience. He has been a great guide, and I have learned from him ever since. This work would not have been possible without his constant support and guidance.

Allegheny County Airport and Butler County Airport were very cooperative in allowing us to use their premises for data collection. This work would not have been possible without my amazing colleagues and collaborators at the AirLab. I want to thank them for extensive discussions on various subjects and helping with field experiments. I would like to thank Jay and Brady for helping with the DAA field tests; a massive credit for this work goes to them. I would also like to acknowledge Milad, Ian, and Joao for helping with the installation/repair of the airport data collection setup(s). Ojit, Jan, Shivam, and Anushka were terrific interns who also contributed to this work.

My Carnegie Mellon experience would have been incomplete without the Table Tennis club members. We made it to the college nationals for the first time in its history. Thanks to Michael, Leo, Logan, TL, and Jenna for training and making a little bit of CMU history.

I want to extend a big thank you to my roommates and neighbors; Manash, Joe, Raghav, Arushi, Pranshu, Bhavya, Shivam, Yashu, Rohan, Unni, Neil, and Siddharth for making my Pittsburgh stay very enjoyable and exciting. My gratitude toward Mrittika for being a constant and making sure I am in the best of health at all times with a balanced vegetable intake! Tapas, Samapika, and Akash thank you for your love and hospitality.

Lastly, I would like to thank my wonderful parents and Promila for their unconditional love and support throughout my life.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Detect and Avoid

Mid-air collision (MAC) and near mid-air collision (NMAC) risk are concerns for both manned and unmanned aircraft operations, especially in low-altitude shared airspace near busy non-towered airports. In order to mitigate these risks, Detect-and-Avoid (DAA) technology has been extensively researched for Unmanned Aircraft Systems (UAS) [6, 34]. DAA, also commonly referred to as *sense* (or *see*) and *avoid* (SAA) is defined as "the capability of a UAS to remain *well clear* from and avoid collisions with other airborne traffic." [11] The well clear boundary as defined by NASA [33] mathematically characterizes a volume, referred to as the Well Clear Violation (WCV) volume, such that aircraft pairs jointly occupying this volume are considered to be in a well clear violation. In visual flight conditions, NMAC/MAC threat mitigation is carried out by a pilot by visually *detecting* and *avoiding* other aircraft to remain *well clear* [34] of them. Typically for medium to large unmanned systems, an active onboard collision avoidance system such as the Traffic Alert and Collision Avoidance System (TCAS-II) or the Airborne Collision Avoidance System (ACAS-X) is used. For unmanned aircraft, ACAS-Xu or ACAS-sXu is used. However, these methods typically only work with transponder-equipped intruder aircraft or *cooperative aircraft*. The most commonly used transponder is the Automatic Dependent Surveillance - Broadcast (ADS-B). ADS-B out broadcasts information about an aircraft's GPS location, altitude, ground speed and other data

to ground stations and other aircraft. DAA is also an essential requirement for beyond visual line of sight (BVLOS) operations in the National Airspace System (NAS) such as autonomous drone delivery systems. However, not all intruder aircraft are equipped with ADS-B transponders (these are called *non-cooperative* aircraft) and thus sUAS usually have to rely on vision-based DAA systems due to size and weight limitations. Fig. 1.1 from NASA shows the key areas where airborne detect-and-avoid needs to be integrated into the National Airspace. One

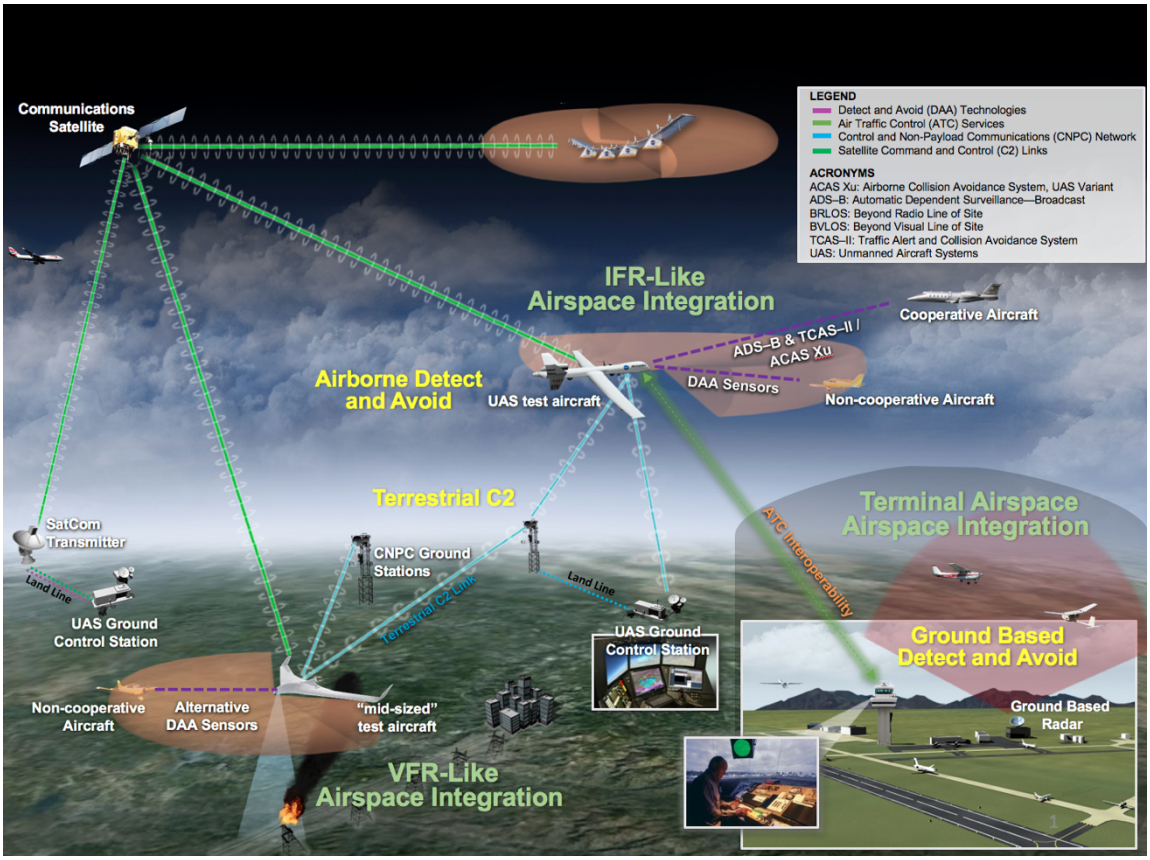


Figure 1.1: Image credits: NASA. Purple-dashed lines show the cases where DAA technology is needed to be used for safe sUAS integration into the National Airspace.

of the major roadblocks to civil UAS integration is the lack of complete Detect and Avoid (DAA) capability. Human vision is the last line of defence against a mid-air collision and is thus critical for aviation safety. Therefore, in order to assist pilots mitigate mid-air collision threats, machine vision can be used to alert the pilot of potential aircraft (and other objects) in the sky. Due to size, weight, and power

constraints of sUAS, radar is not a realistic solution cost-wise. Machine vision has been a promising direction for research in this domain [30]. In this thesis, we present a machine vision-based solution for detecting and tracking aircraft.

1.2 Contributions and Outline

1.2.1 Contributions

The key contributions of this thesis are as follows: (1) Two-stage visual DAA system with a frame alignment network for aligning successive image frames from a moving camera using predicted sparse optical flow, and CenterTrack-based object detection/tracking along with object distance prediction directly from the image. (2) Onboard inference on a six-camera visual DAA payload. (3) Automatic data collection setup at Allegheny County Airport (AGC) with an automatic labeling pipeline.

1.2.2 Outline

The organization of this thesis is as follows. Chapter 2 introduces prior work done in this domain, including classical approaches and more recent learning-based approaches; it also introduces prior work in the domain of small object detection since that forms a core component of the system. Chapter 3 contains information about the various datasets used for this project. It explicitly describes three datasets, two of which are used for training and evaluation. The third dataset describes our work in auto labeling with the help of a static 4-camera dataset. Chapter 4 describes each component of our visual DAA system in detail, along with supporting diagrams. Chapter 5 presents both qualitative and quantitative results of the proposed method compared with two baseline approaches. Finally, chapter 6 summarizes our work and outlines future directions to improve the system.

Chapter 2

Background

With the advent of modern learning-based object detection and tracking, the *state-of-the-art* visual DAA approaches mainly rely on deep neural networks. However, there are many relevant methods and techniques that come from the pre-deep learning era. In this chapter, a summary of the classical methods along with the learning-based methods is presented. Finally, related work for small object detection, which lies at the core of the vision-based DAA is also summarized. Fig. 2.1 shows a typical three-stage pipeline that is commonly used in almost all vision-based DAA methods. If not all, most DAA pipelines used at least one of

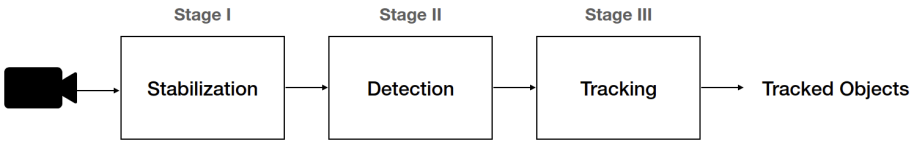


Figure 2.1: Commonly used multi-stage pipeline for vision-based DAA.

the three stages. The approach we describe in this thesis uses all the stages in the pipeline.

2.1 Classical Methods

Traditionally, the main solutions for visual DAA included a modular design with various well-known techniques commonly used in classical computer vision [5, 9,

12, 22, 23, 31]. The primary modules are the following in sequential order: frame stabilization, background-foreground estimation using morphological operations, temporal filtering, detection, and tracking. The frame stabilization step is typically handled using either optical flow-based approaches [29] or image registration using feature matching [43, 45]. We use a deep neural net in our approach to predict the flow between successive frames and use that flow to estimate a homography for frame alignment. [44] used regression-based motion compensation both in the horizontal and vertical directions. The morphological operations are used to enhance the signal of the intruder aircraft. [5, 9, 23] used background subtraction. Machine learning-based approaches have also been used to learn descriptors of the aircraft [9, 40, 44] using methods such as SVMs. Naively using morphological operations would result in false positives. Hence a temporal filtering stage is used. Track-before-detect is a commonly used technique for detecting low-SnR targets in infrared imagery [13], and several people have tried using it in vision-based DAA. Alternative approaches include tracking using Kalman filter banks [35], Hidden Markov Models [21, 23, 32], and Viterbi-based filtering [5, 23].

2.2 Deep Learning Methods

Modern visual DAA approaches heavily rely on deep learning-based object detection and tracking. Detection is done typically using CNNs [7, 17, 18, 47]. The key challenge in visual detection is detecting tiny objects in large resolution images. Due to the very low signal-to-noise ratio, standard anchor-based methods such as YOLO and R-CNN are not ideal for small object detection. Keypoint-based architectures [10] are much more suitable for small objects. Related work can also be found in the domain of face detection [16], aerial imagery [25], and pedestrian tracking [53]. Since computational efficiency is a key requirement for DAA systems, *state-of-the-art* approaches include fully convolution networks with heatmap prediction [18, 19].

Tracking-by-detection is the commonly used paradigm for aircraft tracking [3]. A tracking management system is typically maintained to account for the birth and death of new tracks and associate new detections to existing tracks using the Hungarian algorithm [9, 23]. In multi-object tracking, typically, a metric like

Intersection-over-Union (IoU) for bounding boxes is used to determine the assignment. However, the IoU metric becomes too sensitive to slight deviations in bounding box positions for small objects. In that case, an integrated detection and tracking approach such as [52] is a better choice and our approach is based on that.

2.3 Small Object Detection

DAA systems need to detect and track potential intruders at long distances for sufficient reaction time for planning and avoidance. Thus, at the core of the vision-based DAA lies the problem of small object detection. Modern neural network-based object detection methods do not perform better on small objects than on medium and larger-sized objects. This is mainly due to a lack of discernible features within the object bounding box context. Standard object detection architectures (such as YOLO or Faster R-CNN) do not have high-resolution feature maps to pick up the small object features. In order to address this issue, [4] uses a region context network to pool high-resolution image features from the early stage convolutions and then uses region proposal networks downstream (similar to R-CNN) for prediction. Feature pyramid networks (FPN) [15, 26] have been a popular approach to boost small object detection performance by combining higher and lower resolution feature maps, but they add computational burden. In the context of face detection, [16] uses a coarse 3-level pyramid and fixed-size templates on a fully convolutional network (FCN) to find tiny faces within an image.

One of the significant applications of small object detection is in the domain of satellite imagery and remote sensing [24, 38, 42]. Remote sensing is closely related to our work since the inputs to our vision-based DAA algorithm are very high-resolution images with tiny objects to detect. However, most remote sensing methods are very computationally expensive and run offline. Due to the online requirements of our use case, we cannot afford a high computational burden and, therefore, cannot handle heavy two-stage methods. We thus rely on a backbone architecture that facilities high-resolution feature maps [51] and use a single-stage detector.

Chapter 3

Datasets

There are not many widely available flying object datasets in the context of the DAA problem. The Amazon Prime Air Airborne Object Tracking Dataset is the only publicly available large-scale vision-based DAA dataset with annotated bounding box and range information. We designed two static data collection setups for this project and deployed them at local Pittsburgh airports (KAGC and KBTP) for automated data collection and annotation. We also developed our own DAA hardware for data collection and onboard inference. The following sections describe each of the data sources further.

3.1 Airborne Object Tracking (AOT) Dataset

This dataset was released in 2021 as part of the Airborne Object Tracking Challenge [2] organized by AIcrowd in partnership with Amazon Prime Air. Fig. 3.1 (which is taken from [2]) shows a quick overview of the dataset. It is a collection of around 5000 flight sequences of 120 seconds each at 10Hz resulting in 164 hours of total flight data. There are a total of 3.3M+ labeled image frames containing airborne objects. Helicopters and Airplanes are the primary labels, but there are also examples of birds and drones. The image resolution is 2448x2048, and the images are 8-bit grayscale. Along with bounding box and class labels, the annotations also include range information of the aircraft for a part of the dataset. The range mainly varies from 600 to 2000 meters (25-75 percentiles). The area of the objects labeled

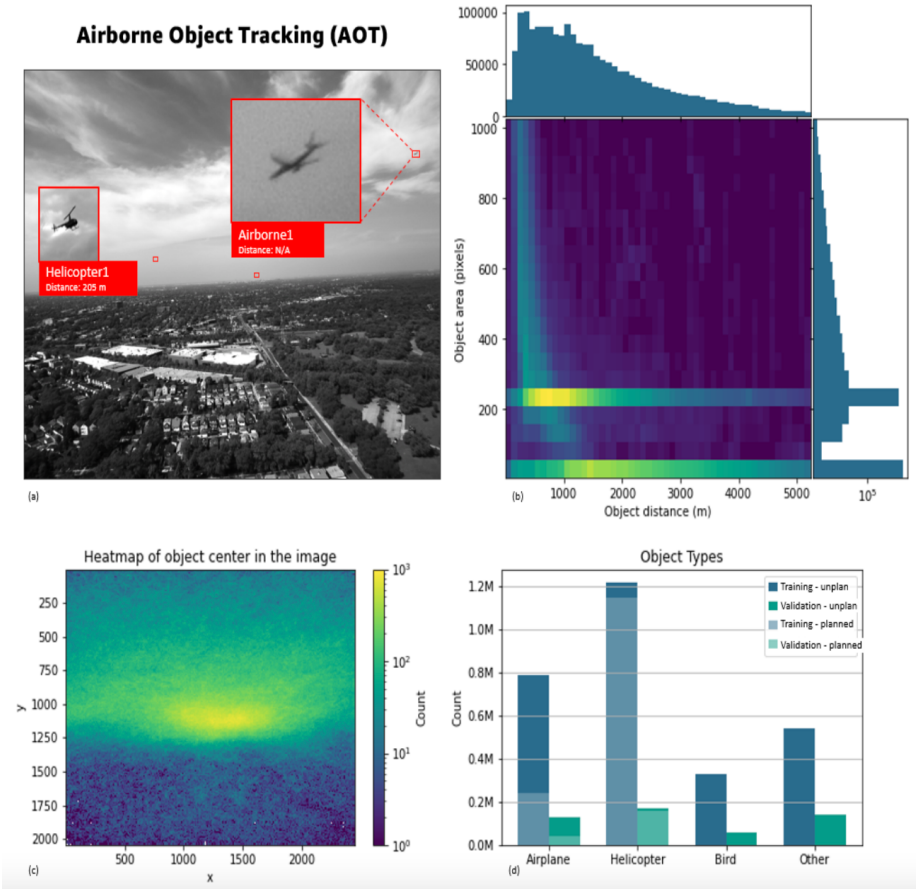


Figure 3.1: AOT dataset overview showing (a) sample objects in the full-resolution image with zoomed-in view, (b) object area vs distance plot, (c) distribution of object center within the image space, and (d) object type instances. Images are taken from the AOT challenge page [2].

vary from 4 to 1000 sq pixels. Among the planned airborne encounters, among 55% of them would qualify as potential collision trajectories. 80% of the targets are above the horizon, 1% on the horizon, and 19% below the horizon. The dataset also captures different sky and visibility conditions: 69% of the sequences have good visibility, 26% have medium visibility, and 5% exhibit poor visibility conditions. We use a train/val split of 90/10 to train and evaluate models on this dataset.

(a) X6C DAA payload

(b) DJI M600 ownship w/ payload

(c) Inflight Cessna w/ payload

Figure 3.2: In-house assembled vision-based DAA payload. (a) The onboard X6C payload is powered using an NVIDIA Xavier AGX dev kit. It has six Sony IMX264 global shutter cameras covering a total of around 220° horizontal FOV and 48° vertical FOV. (b) We mounted the payload onboard a DJI M600 to collect near collision encounters with other drones and helicopters. (c) The payload can also be mounted inside a Cessna aircraft for inflight data collection.

3.2 TartanX6C Dataset

We also designed our own vision-based DAA payload containing six Sony IMX264 cameras spanning a total of approximately 220° FOV horizontally in total and 48° vertically. The payload uses an NVIDIA Xavier AGX 32GB dev kit to handle the image processing. The payload also contains a 3DM-GQ7 GNS/INSS module with a dual-antenna setup for state estimation and ground truth. It also has an ADS-B

receiver. This vision payload was used in DAA field tests involving DJI drones and a Bell helicopter, where the vehicles were flown in near-collision trajectories. It was also used to collect about 7 hours of inflight data from inside a Cessna 172. We have manually labeled a subset of this dataset for evaluation. The evaluation set contains seven sequences of General Aviation (GA) aircraft, three sequences of Helicopters, and eight sequences of multi-rotor UAVs. This dataset was *only* used for evaluation and no training and therefore serves as our test dataset. Figs. [5.1, 5.2, 5.3, 5.4] later in this document show detection results on sample data collected by this payload.

3.3 KAGC Dataset

We partnered with Allegheny County Airport (AGC) to conduct a research study on learning to predict aircraft traffic patterns in a socially-aware context [39]. As a part of this study, we have also installed a 4-camera setup (same hardware as TartanX6C) to capture videos of aircraft within an 8km range of the AGC runways using a Stratux ADS-B receiver [48]. Fig. 3.3 shows the setup. Videos and the ADS-B data (containing GPS and altitude information of aircraft) are saved to disk based on time synchronization. The system automatically collects data from sunrise to sunset every day. Using this data collection setup and the object detection models, we can automatically generate new bounding box labels of aircraft seen in the videos using their recorded 3D position data. The automatic label generation pipeline consists of the following two major components.

3.3.1 Extrinsic Camera Calibration

We assume that the 4-camera setup is located at the origin of a north-east-down (NED) world reference frame where the positive z-axis points towards the earth's center and the positive x-axis points towards True North. Given all the relevant information, such as GPS coordinates and altitudes, we can compute the 3D position of aircraft using geodesics. Given an aircraft 3D position $\mathbf{P} \in \mathbb{R}^3$ in the world reference frame, the corresponding image frame location in homogeneous coordinates

Figure 3.3: Static 4-camera setup mounted on a hangar at Allegheny County Airport (AGC). This data collection setup is remotely monitored and can automatically capture aircraft data (ADS-B + video) from sunrise to sunset everyday.

for camera i ($i = 1, \ldots, 4$) is given by the following perspective projection:

$$\mathbf{p}_i = \mathbf{K}_i[\mathbf{R}_i|\mathbf{t}_i]\mathbf{P} \tag{3.1}$$

where $\mathbf{K}_i$ is a known 3×3 intrinsic matrix for camera i, and $[\mathbf{R}_i|\mathbf{t}_i]$ constitutes the 3×4 extrinsic matrix of camera i. We use the PnP-RANSAC algorithm to estimate $[\mathbf{R}_i|\mathbf{t}_i]$ using eqn 3.1. Using time synchronization, the 2D correspondences for the recorded 3D observations are manually labeled using a custom labeling software. Since the setup is static, this is a one-time calibration step that gives us the projection matrix for camera i with which we can project new 3D aircraft data.

3.3.2 Automatic Label Generation

New bounding box labels for object detection can be generated by applying our object detectors to new aircraft videos. Since the precision of any neural network-based object detection method is not 100%, there are false positives. These false-positive labels can either be manually removed or they can be rejected automatically

based on a simple criterion where only the predicted bounding box closest to the projected image point is chosen. The goal of the auto labeling pipeline is to create

Figure 3.4: Visualization of auto-labeling pipeline. The reprojected point based on ADS-B is shown in the blue circle. The bounding boxes are predicted by the detector and it can be in the bottom of the image, there are false-positives. Using the reprojected point, we can reject those false-positives for new ground truth label generation (chosen label is highlighted in the blue).

a new large-scale labeled aircraft dataset with both bounding box and distance annotations.

3.4 Summary

In this chapter, we presented three different datasets relevant to the problem of aircraft detection and tracking. The first dataset, AOT, is a large-scale object detection and tracking dataset for aircraft and helicopters created by Amazon Prime Air. This serves as training data for the neural networks in our proposed system. The second dataset, TartanX6C, is our evaluation dataset collected by flying drones and helicopters towards each other in a near mid-air collision situation (with the help of safety pilots). The third dataset, KAGC, is a static 4-camera setup

that can automatically capture aircraft data based on their relative distance to the setup. This is mainly used to investigate the problem of generating bounding box labels automatically based on 3D aircraft position and the existing detector.

Chapter 4

Aircraft Detection and Tracking

The overall system design consists of the following four sequential modules: (1) Motion Estimation, (2) Detection and Tracking, (3) Secondary Classification, and (4) Intruder State Update. Fig. 4.8 outlines the overall system design in detail. The inputs are two successive grayscale image frames $I_t, I_{t-1} \in \mathbb{R}^{H \times W}$ where $H \times W$ are the dimensions of the input frames. We utilize the full image resolution of 2448×2048 during inference to maximize the chances of detection at long ranges ($\geq$ 1km). The final outputs of the system are a list of tracked objects with bounding box coordinates, track ID, 2D-KF state (containing pixel-level velocity and acceleration), estimated range, and time to closest point of approach (tCPA). The following subsections describe each of the modules in detail.

4.1 Frame Alignment

In order to separate the foreground objects from the background, one must align successive frames in a video so that the ego-motion of the camera can be discarded. This is done with the help of a frame alignment module that predicts the optical flow between two successive image frames and the confidence of the predicted flow. Fig. 4.1 describes the input/output structure of the frame alignment network. This module takes as input two fixed-size crops of the current and previous input frames $I_t, I_{t-1} \in \mathbb{R}^{H \times W}$, where H and W are the input image height and width respectively. Since the sky is mostly textureless, it is not great for computing the

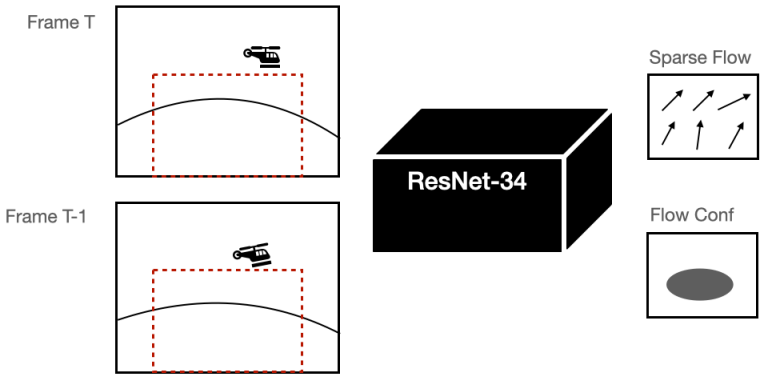


Figure 4.1: Frame alignment module with the inputs and outputs. The actual inputs to the module are cropped from the bottom-center of the images (shown in dashed-red) to provide good features for the optical flow estimate.

background flow. The input images are thus cropped from the center-bottom (thus covering most of the high-texture details present below the horizon) using a fixed-size crop of 2048×1280 (see dashed red lines in Fig. 4.1). The backbone architecture is ResNet-34 with two prediction heads: (1) optical flow offsets $\mathbf{F}_t \in \mathbb{R}^{2\times H\times W}$, (2) confidence heatmap of offsets $\mathbf{C}_t \in \mathbb{R}^{H\times W}$. The prediction is made at $1/32$ scale of the input. Low confidence offset predictions are rejected. Based on the flow offsets, current image points $\mathbf{p}_t$ can be equated with the previous image points $\mathbf{p}_{t-1}$ as:

$$\mathbf{p}_t = \mathbf{H}_t \mathbf{p}_{t-1} \tag{4.1}$$

where $\mathbf{p}_t, \mathbf{p}_{t-1}$ are in homogeneous coordinates and $\mathbf{H}_t$ is an affine homography from the previous frame to the current. $\mathbf{H}_t$ is a 3×3 matrix estimated from the over-determined system of eqn 4.1.

The network is trained using images from the AOT dataset. 75% of the input image tuples are created by simulation. The simulation involves generating a random affine homography by sampling the parameters from a normal distribution. This random homography is used to warp an image frame and thus create a training sample. The remaining 25% of the input image tuples are picked as successive frames from the AOT sequences. The target homography for the neural network is generated by first computing the Lucas-Kanade optical flow (OpenCV) and

then using it to find an affine transform. The training objective minimized is the following:

$$\frac{1}{N}\sum\left(\mathbf{C}_t \circ \text{MSE}\left(\mathbf{F}_t, \hat{\mathbf{F}}_t\right)\right)$$

where $\hat{\mathbf{F}}_t$ and $\mathbf{F}_t$ are the predicted and ground truth optical flow respectively, and MSE denotes the mean squared error.

Fig. 4.2 shows the visualization of successive frame differencing (*i.e.*, background subtraction) with and without frame alignment. We can clearly observe that with frame alignment, the background subtraction result is much better, whereas if we naively subtract successive frames, a lot of noisy artifacts, especially below the horizon, are seen.

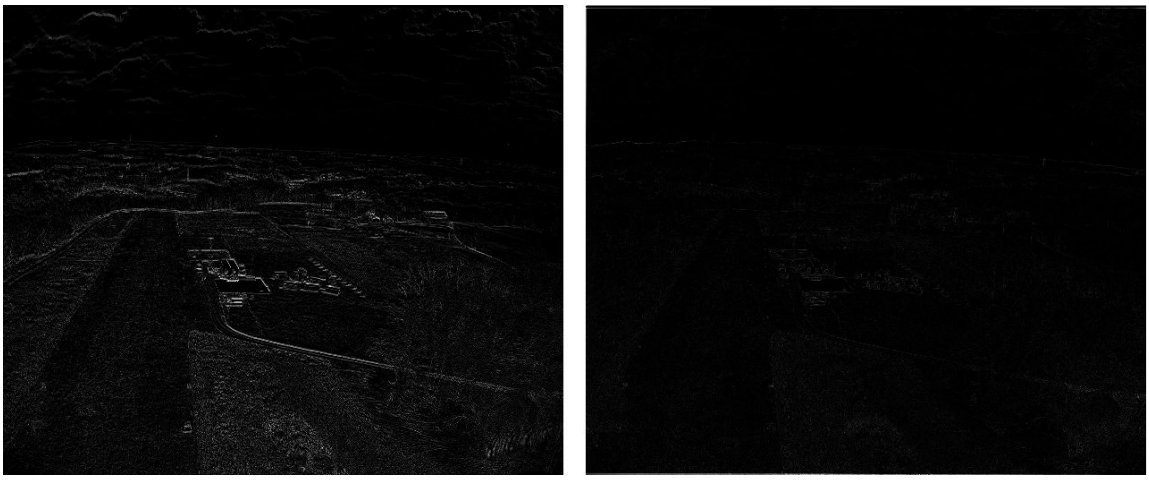


Figure 4.2: Successive frame differencing (background subtraction) with (right image) and without (left image) frame alignment. This snapshot is taken from a sequence in the TartanX6C dataset.

4.2 Detection

The detection architecture is cascaded where the primary detection module runs on the full resolution image of size 2448×2048 (horizontal padding of 56px is applied on each side) and the secondary detection module runs on a smaller crop of 512×512 around the top k track outputs (based on confidence) of the primary module (see Fig. 4.5). For our experiments, we have chosen $k = 4$. The architecture of both modules is based on [41, 52]. The inputs to the network are two aligned

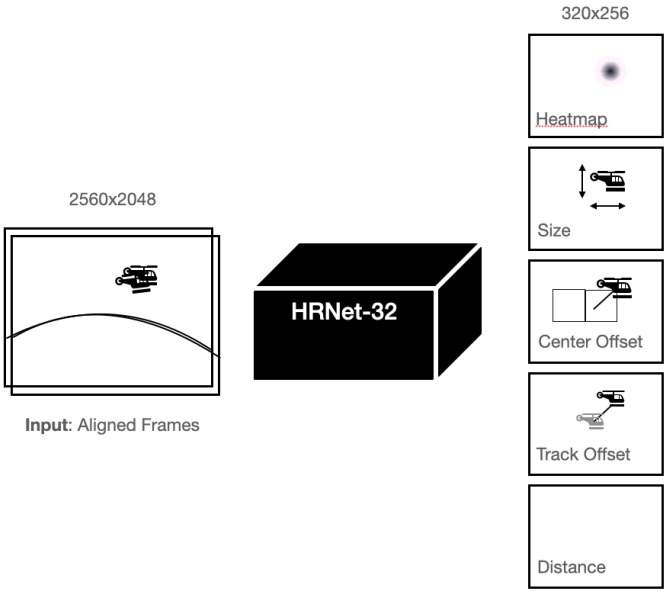


Figure 4.3: The detector is a fully-convolutional architecture with aligned input frames and five output maps.

image frames $I_t, I_{t-1} \in \mathbb{R}^{H \times W}$. The network is fully-convolutional, and the output scale is 1/8 of the input resolution. It outputs five maps: (1) center heatmap encoding the center of the object, (2) bounding box size, (3) center offset from grid to center of object, (4) track offset of object center from the previous frame, (5) distance of the object in log scale. Each of these five heads is trained with a separate loss function. The target center heatmaps during training are rendered using a Gaussian kernel. For distant objects, this results in a single-pixel render which is not helpful for training. Therefore a minimum box size of 3×3 is used for center rendering. The center heatmap is trained using the focal loss [27] for handling large class imbalances:

$$L_h = \frac{1}{N} \sum_{xy} \begin{cases} (1 - \hat{H}_{xy})^{\alpha} \log \hat{H}_{xy}, & \text{if } H_{xy} = 1 \\ (1 - H_{xy})^{\beta} \hat{H}_{xy}^{\alpha} \log(1 - \hat{H}_{xy}), & \text{otherwise} \end{cases}$$

where x, y are the pixel locations in ground-truth and predicted heatmaps $H, \hat{H}$, and α, β are the focal loss parameters. The peaks of this predicted heatmap are the object centers, and we choose the corresponding values from the other predicted heads based on the pixel locations of the peaks. Fig. 4.4 shows visualizations of

heatmap outputs. The bounding box size prediction is regressed by minimizing the following objective:

$$L_{\text{size}} = \frac{1}{N} \sum_{i=1}^{N} \left| \hat{s}_{\mathbf{p}_i} - \mathbf{s}_i \right|$$

where $\hat{s}_{\mathbf{p}_i}$ is the predicted box size at pixel location $\mathbf{p}_i$ and $\mathbf{s}_i$ is the ground truth box size. L1 error is also minimized for the center offset prediction:

$$L_{\text{off}} = \frac{1}{N} \sum_{i=1}^{N} \left| \hat{o}_{\mathbf{p}_i} - o_{\mathbf{p}_i} \right|$$

where $\hat{o}_{\mathbf{p}_i}$ is the predicted offset and $o_{\mathbf{p}_i}$ is the ground truth offset. The track offset loss is again an L1 loss:

$$L_{\text{track}} = \frac{1}{N} \sum_{i=1}^{N} \left| \hat{T}_{\mathbf{p}_i^{(t)}} - \left(\mathbf{p}_i^{(t-1)} - \mathbf{p}_i^{(t)} \right) \right|$$

where $\hat{T}_{\mathbf{p}_i^{(t)}}$ is the predicted track offset and $\mathbf{p}_i^{(t-1)} - \mathbf{p}_i^{(t)}$ denotes the change in the location of the center pixel from the previous frame. Finally, the distance prediction is trained by optimizing the following L2 loss:

$$L_{\text{dist}} = \frac{1}{N} \sum_{i=1}^{N} \left| \log \hat{D}_{\mathbf{p}_i} - \log D_{\mathbf{p}_i} \right|$$

where $\hat{D}_{\mathbf{p}_i}$ and $D_{\mathbf{p}_i}$ are the predicted and ground truth distance. We predict log distance because of large distance values and this helps with training stability. The overall training objective that is minimized is thus:

$$L_{\text{total}} = w_1 L_h + w_2 L_{\text{size}} + w_3 L_{\text{off}} + w_4 L_{\text{track}} + w_5 L_{\text{dist}}$$

where w_i ($i = 1, \ldots, 5$) are hyper-parameters.

Since we are dealing with tiny objects, the choice of the backbone architecture is very crucial for good performance. The key requirements are learning high-resolution image features that can be used to identify tiny objects against the sky (mainly) and clutter and also learning enough context to ignore false positives.

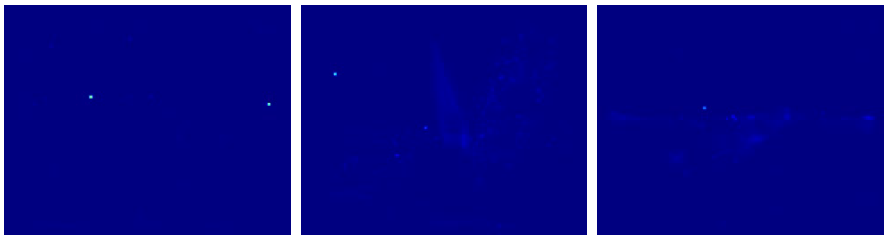


Figure 4.4: Example output heatmaps. The left-most heatmap showing two peaks (helicopter on the left and drone on the right) corresponds to the visualized detector output in Fig. 5.2 (shown later).

The backbone architectures used for the primary and secondary detectors are HRNet [51], and it is an ideal choice for this problem because it fuses high-level and low-level parallel convolutional feature maps using a unique fusion operation that preserves high-resolution features with enough low-level context. Fig. 4.5 shows the overall cascaded detection network. The full-resolution detector predicts the initial heatmap. Then 512×512 crops are taken around the top K ($K = 4$) heatmap peaks and formed into an input batch for the cropped detector. The cropped detector is typically chosen to be a heavier neural network with more parameters since it is operating on a lower-resolution image. It is possible to use

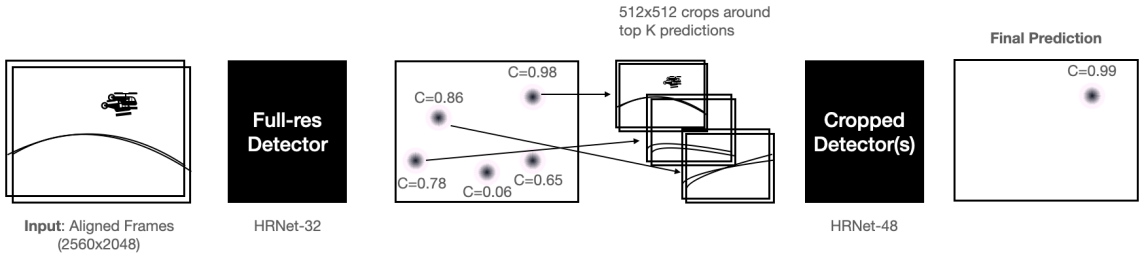


Figure 4.5: Cascaded detection with the first lightweight detector operating at full resolution and the secondary heavier detectors operating on a batch of smaller cropped images (taken around heatmap peaks). The final output is chosen from the cropped image detector.

an ensemble strategy of multiple cropped detectors (with different backbones) to combine the predicted heatmaps, using their mean. For this work and our onboard computational budget, we use a single cropped detector with HRNet as its backbone. The final heatmap output is used for generating the objects.

4.3 Training Details

All the neural networks (frame alignment and detection) are trained using the SGD optimzer [8] with cosine annealing warm restarts [28] as the learning rate scheduling strategy. 512×512 image crops are used to train all the detection networks. Since they are fully-convolutional, smaller random image-crop pairs are sufficient to train the networks. The sampling strategy of the image batches is important for a good overall performance of the detector. In a training batch, 50% samples are chosen with random crops, 25% samples are chosen around hard false-positives, and the remaining 25% are crops taken around true aircraft locations. Fig. 4.6 shows the relative learning-rate schedule and locations where hard false-positive samples are mined for re-training. False-positive predictions

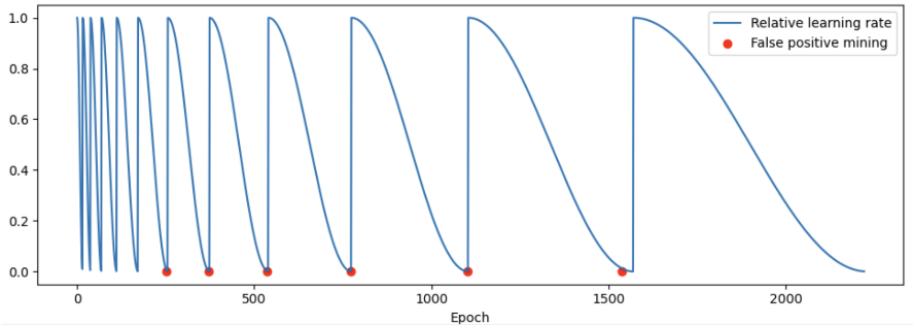


Figure 4.6: Plot showing epochs where hard false positives are mined (right before a "warm restart").

during training with a confidence score over 0.2 are saved for mining and are sampled for training in later epochs to improve the precision of the model. The models were trained on a Tesla P100 GPU with 16GB of GPU memory.

4.4 Tracking

We explored two approaches for tracking detected aircraft, both belonging to the *tracking-by-detection* class of algorithms [49] for multi-object tracking. The first approach, called SORT [3], is used as the baseline tracker for YOLOv5. The second approach is based on the offset tracking vector predicted by the CenterTrack [52]

detector.

4.4.1 SORT: Simple Online Realtime Tracking

SORT [3], as the name suggests, is a simple multi-object tracking algorithm. The strength of this algorithm lies in its simplicity. It uses a combination of the Kalman Filter and the Hungarian algorithm for tracking. The internal state of every tracked object is modeled as follows:

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]$$

where u, v represents the horizontal and vertical position of the intruder in the image, while s and r represent the scale and aspect ratio of the intruder bounding box. Whenever a detection is associated with an existing track, the update-step of the Kalman Filter is used to update the state with the detection bounding box coordinates. A constant acceleration model is used for the predict-step.

In the original implementation of SORT, the association metric used is Intersection-over-Union (IoU). However, the problem under consideration deals with very tiny objects, and thus the IoU metric becomes very sensitive (value becomes zero due to no overlap) to even small changes in bounding box position. Therefore, we use the generalized IoU (GIoU) metric for data association. We use a minimum GIoU threshold $GIoU_{\text{min}}$ to reject poor matches. Another option is to use the Euclidean distance between object centers. The data association step assigns detections in the current frame to the list of existing tracks. This is solved using the Hungarian algorithm. A tracking management system is used to handle the birth and death of intruder tracks. New tracks with unique identities are created when objects enter the image or when current frame detections don't have a valid associated track ($GIoU < GIoU_{\text{min}}$), they are put in a shadow tracking mode for T_{shadow} seconds. An existing object which is being shadow tracked leaves the probationary period only if it is consistently detected (*i.e.*, an assignment is found) for at least T_{shadow} seconds. Similarly, when an existing track leaves the image space or is not associated with any detection for at least T_{lost} seconds, the track is deleted.

We use SORT along with YOLOv5 as a baseline method for visual detection and tracking. Table 4.1 outlines the tracking parameter values used for the baseline

approach with YOLOv5:

Parameter	Value	Domain
$GIoU_{\min}$	-0.7	$[-1, 1]$
T_{shadow}	1.5s	$(0, \infty)$
T_{lost}	2.0s	$(0, \infty)$

Table 4.1: SORT parameters used in baseline approach

4.4.2 Offset Tracking

The offset tracking algorithm is a greedy algorithm [52] based on the predicted track offset from the detector. The key idea in this algorithm is to use the track offset vector to associate current frame detections to a list of existing tracks. Algorithm 1 outlines the offset tracking algorithm. The power of this algorithm lies in its

Algorithm 1 Offset Tracking

$T^{(t-1)} \leftarrow \{(\mathbf{p}, \mathbf{s}, \text{ID})_j^{(t-1)}\}_{j=1}^{M}$ ▷ tracked objects in previous frame (pos, size, ID)
$\hat{H}^{(t)} \leftarrow \{(\hat{\mathbf{p}}, \hat{\mathbf{d}})_i^{(t)}\}_{i=1}^{N}$ ▷ current frame heatmap peaks and track offsets
$T^{(t)} \leftarrow \emptyset$ ▷ list of tracked objects for current frame
$S \leftarrow \emptyset$ ▷ set of matched tracks
$W \leftarrow \text{Cost}(\hat{H}^{(t)}, T^{(t-1)})$ ▷ $W_{ij} = \|\hat{\mathbf{p}}_i^{(t)} - \hat{\mathbf{d}}_i^{(t)}, \mathbf{p}_j^{(t-1)}\|_2$
for $i \leftarrow 1, \ldots, N$ **do**
  $j \leftarrow \text{argmin}_{j \notin S} W_{ij}$
  $\kappa \leftarrow \min\left(\sqrt{\hat{w}_i \hat{h}_i}, \sqrt{w_j h_j}\right)$
  if $W_{ij} < \kappa$ **then** ▷ κ is a distance threshold
    $T^{(t)} \leftarrow T^{(t)} \bigcup \left(\hat{\mathbf{p}}_i^{(t)}, \hat{\mathbf{s}}_i^{(t)}, \text{ID}_j^{(t-1)}\right)$ ▷ propagate matched ID
    $S \leftarrow S \bigcup \{j\}$ ▷ mark j as matched
  else
    $T^{(t)} \leftarrow T^{(t)} \bigcup \left(\hat{\mathbf{p}}_i^{(t)}, \hat{\mathbf{s}}_i^{(t)}, \text{ID}_{\text{new}}\right)$ ▷ create a new track
  end if
end for
return $T^{(t)}$

simplicity. Using the predicted track offset vector, we subtract the offset from the

current object center to recover the center location in the previous frame. Then we compare the previous frame center with the offset-adjusted center and check if the distance between them falls below a certain threshold κ. If it does, we propagate the existing track ID to the current detection and mark it as matched. Otherwise, we spawn a new track ID with the current detection and proceed. The tracking management paradigm for handling birth and death of tracks is similar to SORT.

4.5 Secondary Classifier

Figure 4.7: Secondary classifier with inputs and outputs.

The secondary classifier is a ResNet-18 module used for false-positive rejection (see Fig. 4.7). The input to the network is a fixed-size 64x64 crop (padded and resized) around the bounding boxes detected by the object detectors. It is a binary classification network that predicts whether the input crop is an aircraft or not. The training data for this network is collected from the results of the detectors. We mine false-positive samples to train the classifier for better false-positive rejection. We use focal loss [27] to train the model (since a training batch contains more false-positive samples than true positives). The secondary classifier improves the overall precision of the system. It can also be used to quickly retrain using novel data rather than training the detector, which would take some time. This enables one to run the object detectors at a low confidence threshold which improves recall.

4.6 Intruder State Estimation

The final stage of our DAA module is intruder state estimation. This module maintains an internal state of each intruder detected using a 2D Kalman Filter [20] with a constant acceleration motion model. When using SORT as a tracker, this

step is unnecessary because SORT itself maintains an internal KF state. For offset tracking, this KF is necessary to compute pixel-level velocity and acceleration. The angular-rate is computed as:

$$r = \theta\sqrt{\dot{x}^2 + \dot{y}^2}$$

where θ is the degree/pixels ratio of the camera, and $\dot{x}, \dot{y}$ denotes the pixel velocity of the object center. This module also computes the time to closest point of approach for the intruder:

$$\text{tCPA}(i, j) = \frac{t_j - t_i}{-1 + \sqrt{\frac{a_j}{a_i}}}$$

where a_k, t_k denotes the area of the bounding box and time at frame k respectively. The formulation is based on [14] and the key idea behind this formulation is that as the bounding box area of the intruder increases in size (meaning that the object is getting closer), the tCPA value decreases. tCPA is inversely proportional to the rate of change of the square root of the bounding box area.

4.7 Summary

In this chapter, we presented our overall system design along with the various sub-components. In particular, we first describe Stage I of our system, Frame Alignment. This module takes image crops and produces a sparse optical flow and confidence map for frame alignment. The key insight here is that the predicted flow is actually the background optical flow, and thus, it can be used to find background homography mapping between successive image frames for alignment. Next, we describe the various components of our detector and how it uses the aligned frames as input to produce the necessary output maps for generating the bounding boxes. We then proceed to describe the tracking algorithms used for our method and the baselines. Finally, we describe our secondary classification module along with the intruder parameters update step.

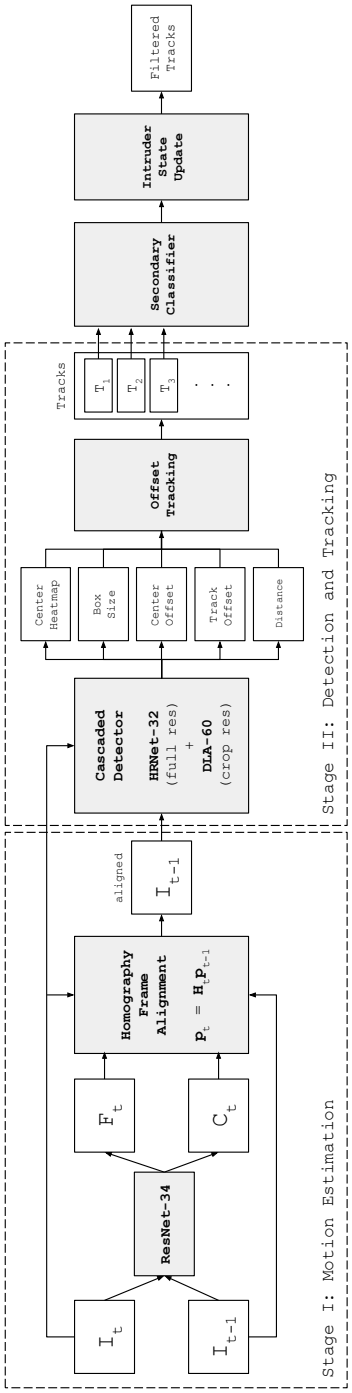


Figure 4.8: System architecture showing the various internal components. Shaded gray boxes are the modules are the white boxes are the input/output variables.

Chapter 5

Results

5.1 Qualitative Results

Figures [5.1, 5.2, 5.3, 5.4] show a visual snapshot of the proposed system in action. The detected bounding box in green is zoomed at the bottom-left corner to show a zoomed context of the object. The text display embedded with the bounding box contains information about the intruder being tracked. The legend is as follows: **ID**: track-id of the intruder, **C**: track confidence, **T**: time to closest point approach (tCPA) (in most of the images, this is 60s because a median filtered tCPA is computed which requires a few seconds of tracking before it changes, the max tCPA of 60s is thus shown), **D**: intruder distance (in nautical miles), **V**: angular rate in deg/s. Fig. 5.1 shows the detection of an intruder DJI M210 drone flying in a near-collision course above the horizon at a relative velocity of 8m/s. Fig. 5.2 shows an interesting below-the-horizon detection case. The zoomed-in view shows the difficulty of background clutter for below-the-horizon cases. Figs. 5.4 and 5.3 shows visual detection and tracking inflight from within a Cessna. An interesting observation we have made based on the qualitative results of the system is that we often see bird detections. Fig. 5.4 shows an example of a bird detection on the right side of the image. Birds are extremely hard to spot with the naked eye in the test videos, but our detector has picked up many such birds.

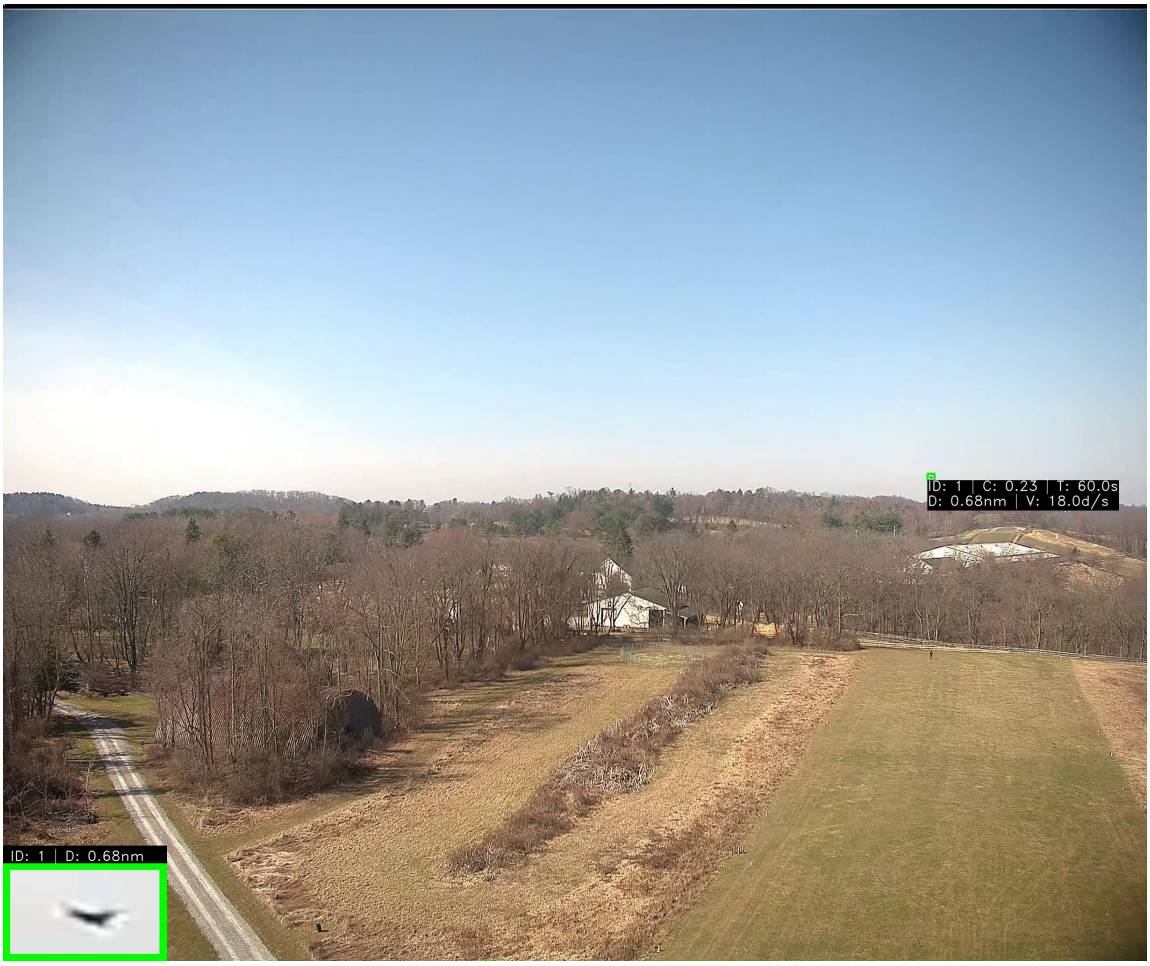


Figure 5.1: TartanX6C drone-drone encounter sequence with relative velocity of $\sim$8m/s.

Figure 5.2: TartanX6C drone-helicopter encounter sequence with relative velocity of ∼40m/s. This shows an example of below-the-horizon detection. The detector also picks up the hovering filming drone of the encounter on the right.

Figure 5.3: TartanX6C in-flight Cessna sequence capturing another aircraft in a non-collision course.

Figure 5.4: TartanX6C in-flight Cessna sequence chasing another aircraft during takeoff. Bird detection is observed on the right as well.

5.2 Quantitative Results

The detection and tracking performance of the proposed method is evaluated on a held-out subset of 200 flights in the AOT dataset and all the sequences in the TartanX6C dataset. Siam-MOT [46] is the baseline (based on Siamese multi-object tracking) provided by [2] and YOLOv5 [50] is trained on the same dataset. The confidence threshold for prediction was set at 0.5.

5.2.1 Detection and Tracking

Table 5.1 outlines the key detection and tracking metrics over which we evaluate the methods (some of these metrics are taken from the AOT challenge [2]): For the

Metric	Description	Domain
P	Precision	$[0, 1]$
R	Recall	$[0, 1]$
EDR	Encounter Detection Rate	$[0, 1]$
FPPI	False positives per image	≥ 0
IDSPI	Track ID switches per image	≥ 0
ARE	Angular rate error	≥ 0 (deg/s)

Table 5.1: Key Performance Metrics

encounter detection rate, an encounter is considered valid if the airborne object is consistently tracked for at least 3 seconds before its range falls below 2000ft. Table 5.2 outlines the overall results of the proposed system along with the two baselines. From the table, it can be deduced that YOLOv5+SORT has the worst performance. This is because YOLOv5, which is an excellent *state-of-the-art* object detector in general, is not the most ideal for detecting small objects. Our system outperforms the baselines in all the metrics mentioned earlier. The secondary classifier (SC) is an optional module in our system that can either be used or not. Using it greatly improves the precision of the system while only taking a slight hit in the recall.

Detecting aircraft above and below the horizon poses significantly different challenges, the latter being the more difficult due to the presence of background clutter. Table 5.3 outlines the detection metrics based on *above* or *below* the horizon

Metric	Baselines		Our System	
	YOLOv5+SORT	Siam-MOT	SC: off	SC: on
P	0.8436	0.9758	0.9532	**0.9916**
R	0.3326	0.4253	**0.4776**	0.4634
EDR	0.8867	0.938	0.9542	**0.9639**
FPPI	0.0987	0.0050	0.0111	**0.0018**
IDSPI	0.0872	0.0011	0.0010	**0.0009**
ARE	1.15	0.89	0.87	**0.85**

Table 5.2: Comparison of proposed method w/ baselines.

cases. We can observe that the precision is steady across both domains while the recall is much lower for below-horizon detections being a more difficult task. This is a key area to improve upon in the future. We can also observe the impact of

	Above Horizon		Below Horizon	
	SC: off	SC: on	SC: off	SC: on
P	0.94	**0.99**	0.94	**0.99**
R	**0.63**	0.59	**0.38**	0.36
FPPI	0.0122	**0.0012**	0.0175	**0.0026**
IDSPI	**0.0071**	0.0073	0.0015	**0.0011**

Table 5.3: Performance of the proposed system based on metrics calculated separately for above and below horizon scenarios.

the secondary classifier improving false-positive rejection by reducing the FPPI number by almost 10x. It however does reduce the recall slightly.

5.2.2 Range Estimation and Angular-rate Error

Fig. 5.5 shows the box plot of the range error in the TartanX6C dataset. The error is computed as a fraction of the ground truth range. For the TartanX6C dataset we observe that the median range error obeys the allowable error threshold of 15% up to a range of 1.5km. Fig. 5.6 shows individual range plots for detected aircraft for certain sequences in both datasets. We can observe that the range prediction quality gets worse with increasing object distance, especially with distances of over 1.25km. Fig. 5.7 shows example angle-rate plots (computed via forward-differencing of the position) compared to the ground truth (computed using the five-point stencil).

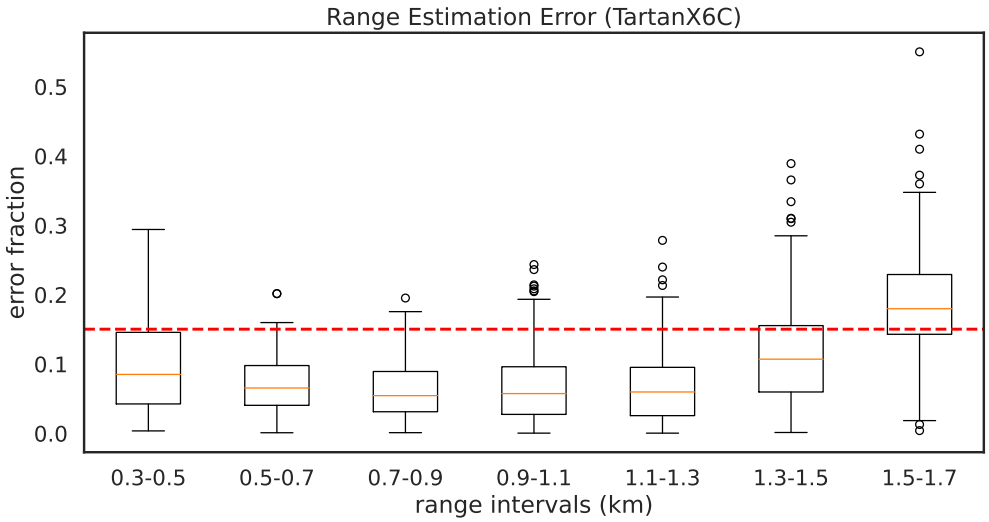
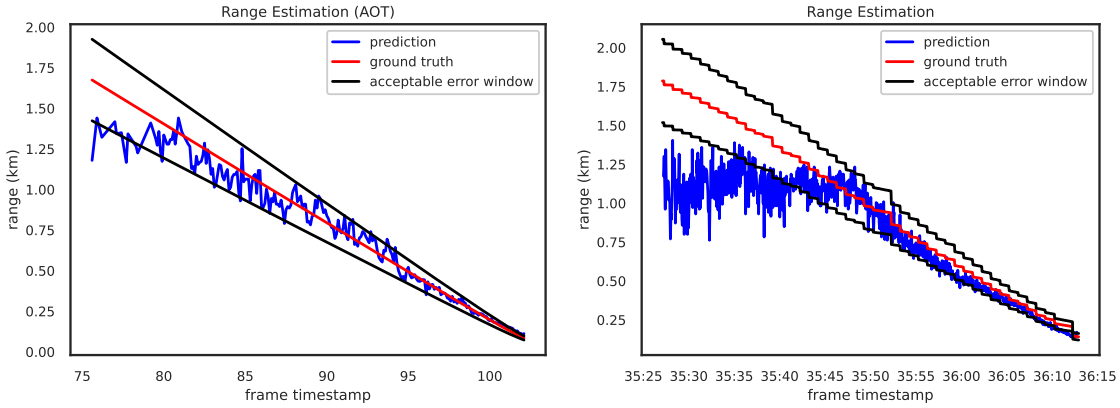


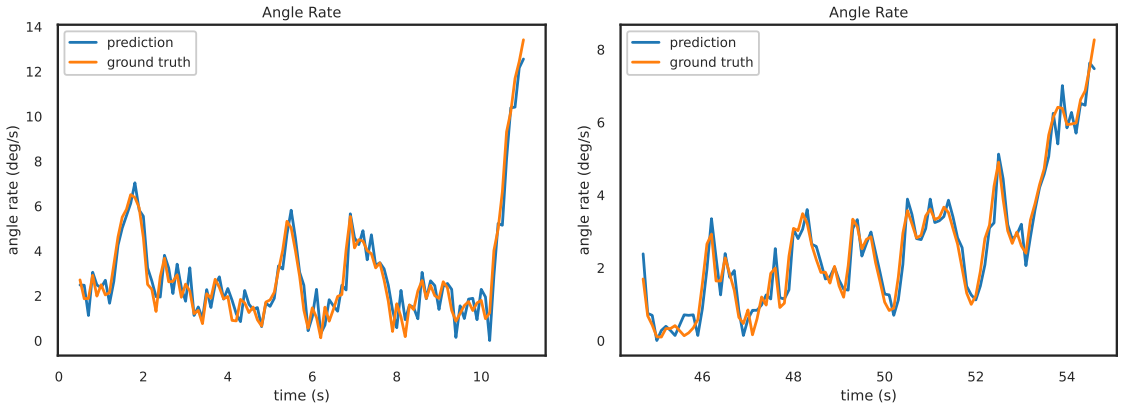
Figure 5.5: Box plot showing the range estimation error as a fraction of the ground truth range for different intervals. Dashed red line is the allowable error threshold of 15% according to ASTM F3442/F3442M. Distance estimation error gets worse with increasing range.



(a) AOT sample sequence

(b) TartanX6C sample sequence

Figure 5.6: Range estimation plots compared to ground-truth for two sequences. We note that the range estimation deteriorates after 1.2km typically. Black line denotes the 15% error margin (relevant for interpretation with regards to ASTM F3442/F3442M standard)



(a) AOT sample sequence

(b) TartanX6C sample sequence

Figure 5.7: Angle-rate prediction vs ground truth (in deg/s).

Fig. 5.8 shows the probability of track (recall) of the proposed system based on range intervals. The probability of track remains more than 95% up to a range of 700m.

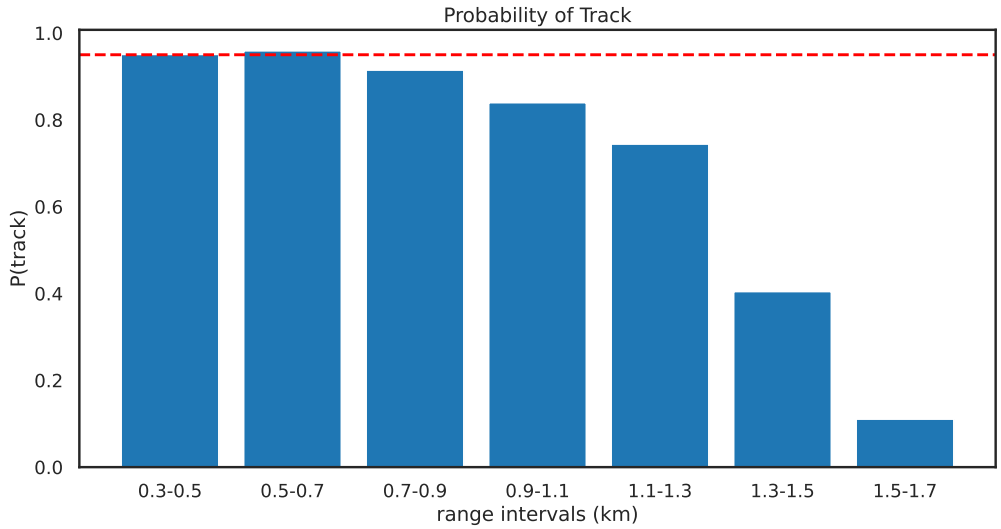


Figure 5.8: P(track) vs range. The probability of track (recall) reduces with increasing distance. Dashed red line shows the 95% level. P(track) $\geq$ 95% upto 700m.

5.3 Summary and Interpretation of Results

In this chapter, we presented some qualitative results (snapshots) of the proposed visual DAA system on some of the evaluation sequences. Below and above horizon detection cases were shown, and an interesting case of bird detection was visualized. Individual range estimation and angular-rate plots for sample sequences from both AOT and TartanX6C dataset were shown along with ground truth. The system angular-rate error is 0.85deg/s, and the probability of track is more than 95% for objects up to a range of 700m. We further investigate our system based on quantitative analysis and comparison with two baseline approaches on key performance metrics defined previously.

Based on our quantitative results, we interpret our results with regard to the ASTM F3442/F3442M standard [1]. This standard was created for unmanned aircraft (UA) with a maximum dimension $\leq$ 25ft, operating at airspeeds below 100kts, and of any configuration or category. This standard specifies surveillance requirements (based on vehicle configuration) for low risk-ratio UA operations. In particular, based on multiple simulation studies of near-collision encounters for different configurations of UA, this standard was created, and it found that only range estimation and angular-rate error and only two key performance indicators for DAA. It also specifies a minimum intruder tracking probability of 95%. Although they do not specify how this probability was computed, we interpret it as the recall value of the tracker (see Fig. 5.8). Based on the recall value, our system has a probability of track of more than 95% up to a range of 700m (based on empirical data). We also know that our system angular-rate error is within 0.9deg/s. Thus based on these two error metrics, we can use the standard to check which classes of aircraft can be used for low risk-ratio operations using our DAA surveillance system. Table 5.4 outlines the minimum track range requirement for different autonomous UA specs based on a maximum angular-rate error of 0.9deg/s. Based on Table 5.4 we can observe that our DAA surveillance system can satisfy low-risk UA operations for two specification categories: (1) **(60kts, 31.53deg/s, 250-500ft/min)**, and (2) **(90kts, 21.02deg/s, 250-500ft/min)**.

Ownship Specifications			**Min. Required Range**
Cruise Speed	**Turn Rate**	**Vertical Speed**	
30 kts	63.05 deg/s	250-500 ft/min	1222m
60 kts	10.51deg/s	250-500 ft/min	963m
60 kts	**31.53 deg/s**	**250-500 ft/min**	**703m** ($\sim$ 700m)
90 kts	7.01 deg/s	250-500 ft/min	1018m
90 kts	**21.02 deg/s**	**250-500 ft/min**	**666m** ($\leq$ 700m)

Table 5.4: Required min. detection range with P(track) $\geq$ 95% based on maximum angular-rate error of 0.9deg/s for different UA configurations. These values are taken from the ASTM F3442/F3442M standard.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this thesis, we introduced the general problem of Detect-and-Avoid and why it is necessary for future operations of small unmanned aerial systems in the national airspace. We presented prior work in this domain ranging from classical image processing methods to more modern deep learning-based methods. We introduced our two-stage visual DAA system that is specifically designed to detect and track aircraft at long distances (*i.e.*, small objects in high-resolution input images) with a frame alignment module and fully-convolutional object detector based on CenterTrack [52]. Our method was evaluated on two real-world datasets and compared to two baselines for performance evaluation, and we observed that the proposed method performs significantly better than the baselines. We provide both qualitative and quantitative results of our system. We also interpreted our quantitative results with regard to the DAA standard, ASTM F3442/F3442M [1]. The system has a minimum tracking probability of 95% up to a range of 700m and an overall angular-rate error of 0.85deg/s. Based on these surveillance parameters, according to the standard, our DAA system can be integrated with two categories of ownship specifications satisfying low risk-ratio UA operations. The categories are: (1) min. 60kts cruise speed, min. 31.53deg/s turn rate, min. 250-500ft/min climb rate, and (2) min. 90kts cruise speed, min. 21.02deg/s turn rate, min. 250-500ft/min climb rate.

6.2 Future Work

There are different areas in which this work can be further improved. The broad categories are: (1) Synthetic Dataset Generation, (2) Multi-frame Detection, and (3) Optimization for Deployment.

6.2.1 Synthetic Dataset Generation

While the AOT dataset is an excellent dataset for initial training of the detection models, it is still limited in the number of encounters with intruders. Collecting real-world data for DAA experiments is costly and time-consuming. Therefore, innovation in this regard can be done with a smart synthetic dataset generation pipeline. Relying fully on simulation is not ideal, as we have seen with X-Plane 11 data. Training on full simulation makes it harder for the models to transfer to real-world images. Therefore a hybrid approach involving simulated aircraft objects rendered on real-world images would be a great next step to creating a new large-scale DAA dataset. In the context of the DAA problem, actual intruder aircraft only occupy a tiny space in the image, and most of it is background. Therefore, real background images have true problem-domain complexity and can be easily collected using cameras (*i.e.*, the X6C setup) onboard a drone or some other aircraft. Obtaining a variety of different foreground objects, such as intruder aircraft, can be very hard to obtain but is easy to simulate using rendering. Computer graphics techniques such as match moving can be used to achieve this task.

6.2.2 Frame Alignment Replacement

When working with a fixed computational budget such as our use-case, it might be better to replace the optical flow-based frame alignment with hardware-based solutions such as using IMU/Gyro readings along with the camera model to estimate the ownship ego-motion.

6.2.3 Multi-frame Detection

Our current method relies on only two successive frames for alignment, and then both the frames are passed as inputs to the detector. In order to boost the performance of below-horizon detections, the cropped resolution (512×512) detectors can potentially use more than two frames as input to boost the object signal within the background clutter. This can be done by extending the CenterTrack [52] architecture to accommodate more than two frames as input. However, this would require an extra computational burden along with GPU memory, and a trade-off analysis needs to be performed in terms of computation and performance.

6.2.4 Ensemble Models

The overall performance of the detector can be improved by using an ensemble of networks rather than a single one. With different backbones, an ensemble model would be more powerful and accurate. The individual heatmap outputs can be combined using their mean, and a similar strategy can be applied to the other output heads as well. This could, however, violate the computational and memory budget of the hardware, and a trade-off study can be done in this regard.

6.2.5 Extension to other object classes

The current system is *only* trained on examples of general aviation aircraft and helicopters. Although our system can reliably pick up drones and birds in the sky, we can extend our detection and tracking dataset to include drones and birds as well. The AOT dataset already has a few examples of those classes but without distance annotations. Since our model jointly learns the distance estimate of objects, distance annotations are necessary for training good models. Therefore, a synthetic rendering scheme can help in this regard which can also provide distance annotations.

Furthermore, patterns of bird motion within the image space is significantly different from that of aircraft and helicopters. Since image-based classification is challenging for this problem, developing a classifier based on object motion within the image space might be a direction worth exploring.

6.2.6 Multi-camera Tracking

Currently, detection and tracking only work for a single camera image. The X6C payload has six forward-facing cameras spanning a horizontal FOV of 220°. The overlap between successive cameras is roughly 10°. The final goal is to run the system on all six cameras and enable tracking across cameras. This can be approached in two ways. Either all the six images can be fused into one, and then the problem reduces to the current single-camera case. However, the image resolution can become pretty large (roughly 14000x2000), and inference might be extremely slow. The second option is to run inference on a batch of individual camera images and then solve the association problem across image frames (using a greedy tracking strategy similar to SORT or Offset Tracking).

6.2.7 Optimization for Deployment

Fig. 6.1 shows the current deployment used for onboard inference on the X6C hardware. We leverage NVIDIA's DeepStream [36] pipeline for hardware-accelerated

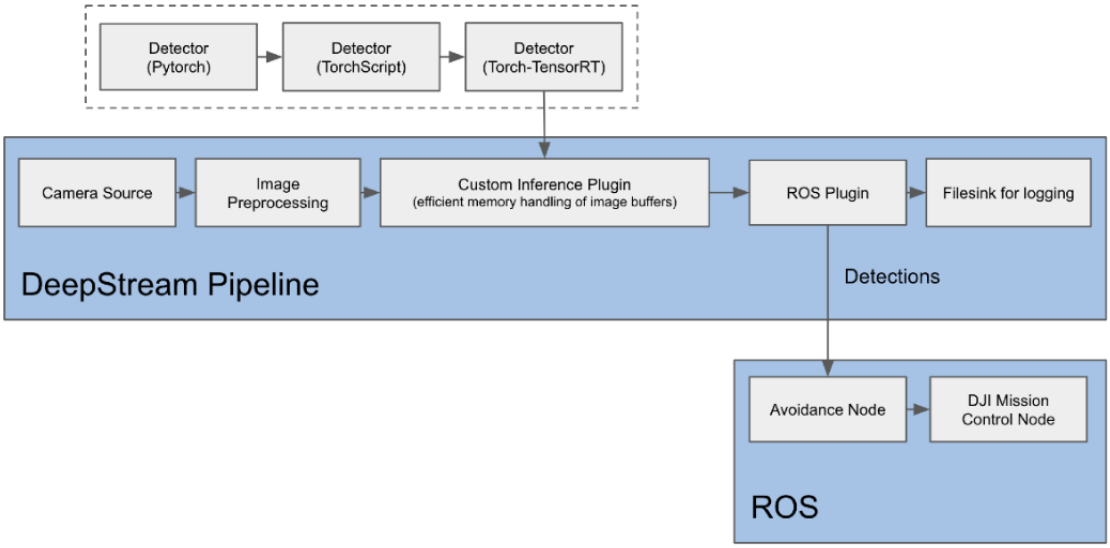


Figure 6.1: Deployment flow diagram for onboard inference on X6C.

encoding/decoding on the NVIDIA Xavier AGX, which is the main compute device on the X6C device. The detection models are first converted from native PyTorch code to TorchScript (which generates serializable models that can be loaded in

non-python-dependent processes, such as C++). We then further convert the Torch-Script models to TensorRT using the Torch-TensorRT [37] package. This pipeline traces the computation graph of the neural networks based on the input size and shape during inference and optimizes the computational flow by freezing the graph. It also optimizes individual layer operations by combining multiple ops into one and so on.

On average, we see a 5X improvement in performance speed when using TensorRT models compared to native PyTorch. However, the current bottleneck of the system is inefficient memory copies of image buffers between GPU and CPU. Currently, the images are loaded in GPU memory, but before being sent to the detector for inference, they are mapped into the CPU and then loaded back into the GPU via a Torch tensor. This is an expensive operation and can be streamlined to avoid the unnecessary memory copy to increase the overall FPS of the system.

Bibliography

[1] Standard Specification for Detect and Avoid System Performance Requirements. Standard ASTM F3442/F3442M-20, ASTM International, 2020. 5.3, 6.1

[2] AICrowd. Airborne object tracking challenge. URL https://www.aicrowd.com/challenges/airborne-object-tracking-challenge. (document), 3.1, 3.1, 5.2, 5.2.1

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016. 2.2, 4.4, 4.4.1

[4] Brais Bosquet, Manuel Mucientes, and Víctor M Brea. Stdnet: A convnet for small target detection. In *BMVC*, page 253, 2018. 2.3

[5] Ryan Carnie, Rodney Walker, and Peter Corke. Image processing algorithms for uav" sense and avoid". In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2848–2853. IEEE, 2006. 2.1

[6] James P Chamberlain, Maria C Consiglio, and César Muñoz. Danti: detect and avoid in the cockpit. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 4491, 2017. 1.1

[7] Xueyun Chen, Shiming Xiang, Cheng-Lin Liu, and Chun-Hong Pan. Aircraft detection by deep convolutional neural networks. *IPSJ Transactions on Computer Vision and Applications*, 7:10–17, 2014. 2.2

[8] Aaron Defazio and Samy Jelassi. Adaptivity without compromise: a momentumized, adaptive, dual averaged gradient method for stochastic optimization. *arXiv preprint arXiv:2101.11075*, 2021. 4.3

[9] Debadeepta Dey, Christopher Geyer, Sanjiv Singh, and Matt Digioia. Passive, long-range detection of aircraft: towards a field deployable sense and avoid system. In *Field and Service Robotics*, pages 113–123. Springer, 2010. 2.1, 2.2

[10] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the*

IEEE/CVF international conference on computer vision, pages 6569–6578, 2019. 2.2

[11] FAA Sponsored Sense and Avoid Workshop. Sense and avoid (saa) for unmanned aircraft systems (uas), 2009. 1.1

[12] Giancarmine Fasano, Domenico Accardo, Anna Elena Tirri, Antonio Moccia, and Ettore De Lellis. Morphological filtering and target tracking for vision-based uas sense and avoid. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 430–440. IEEE, 2014. 2.1

[13] Manuel F Fernandez. Detecting and tracking low-observable targets using ir. In *Signal and Data Processing of Small Targets 1990*, volume 1305, page 193. International Society for Optics and Photonics, 1990. 2.1

[14] Tanya Glozman, Anthony Narkawicz, Ishay Kamon, Franco Callari, and Amir Navot. A vision-based solution to estimating time to closest point of approach for sense and avoid. In *AIAA Scitech 2021 Forum*, page 0450, 2021. 4.6

[15] Yuqi Gong, Xuehui Yu, Yao Ding, Xiaoke Peng, Jian Zhao, and Zhenjun Han. Effective fusion factor in fpn for tiny object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1160–1168, 2021. 2.3

[16] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017. 2.2, 2.3

[17] Sunyou Hwang, Jaehyun Lee, Heemin Shin, Sungwook Cho, and David Hyunchul Shim. Aircraft detection using deep convolutional neural network in small unmanned aircraft systems. In *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, page 2137. 2018. 2.2

[18] Jasmin James, Jason J Ford, and Timothy L Molloy. Learning to detect aircraft for long-range vision-based sense-and-avoid systems. *IEEE Robotics and Automation Letters*, 3(4):4383–4390, 2018. 2.2

[19] Jasmin James, Jason J Ford, and Timothy L Molloy. Below horizon aircraft detection using deep learning for vision-based sense and avoid. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 965–970. IEEE, 2019. 2.2

[20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. 4.6

[21] John Lai, Jason J Ford, Peter O'Shea, and Rodney Walker. Hidden markov model filter banks for dim target detection from image sequences. In *2008 Digital Image Computing: Techniques and Applications*, pages 312–319. IEEE, 2008.

2.1

[22] John Lai, Luis Mejias, and Jason J Ford. Airborne vision-based collision-detection system. *Journal of Field Robotics*, 28(2):137–157, 2011. 2.1

[23] John Lai, Jason J Ford, Luis Mejias, and Peter O'Shea. Characterization of sky-region morphological-temporal airborne collision detection. *Journal of Field Robotics*, 30(2):171–193, 2013. 2.1, 2.2

[24] Rodney LaLonde, Dong Zhang, and Mubarak Shah. Clusternet: Detecting small objects in large scenes by exploiting spatio-temporal information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4003–4012, 2018. 2.3

[25] Dong Liang, Zongqi Wei, Dong Zhang, Qixiang Geng, Liyan Zhang, Han Sun, Huiyu Zhou, Mingqiang Wei, and Pan Gao. Learning calibrated-guidance for object detection in aerial images. *arXiv preprint arXiv:2103.11399*, 2021. 2.2

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2.3

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4.2, 4.5

[28] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 4.3

[29] Jeffrey W McCandless. Detection of aircraft in video sequences using a predictive optical flow algorithm. *Optical Engineering*, 38(3):523–530, 1999. 2.1

[30] Aaron Mcfadyen and Luis Mejias. A survey of autonomous vision-based see and avoid for unmanned aircraft systems. *Progress in Aerospace Sciences*, 80: 1–17, 2016. 1.1

[31] Luis Mejias, Scott McNamara, John Lai, and Jason Ford. Vision-based detection and tracking of aerial targets for uav collision avoidance. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 87–92. IEEE, 2010. 2.1

[32] Timothy L Molloy, Jason J Ford, and Luis Mejias. Detection of aircraft below the horizon for vision-based detect and avoid in unmanned aircraft systems. *Journal of Field Robotics*, 34(7):1378–1391, 2017. 2.1

[33] César Munoz, Anthony Narkawicz, James Chamberlain, Maria C Consiglio, and Jason M Upchurch. A family of well-clear boundary models for the integration of uas in the nas. In *14th AIAA Aviation Technology, Integration, and*

Operations Conference, page 2412, 2014. 1.1

[34] César Muñoz, Anthony Narkawicz, George Hagen, Jason Upchurch, Aaron Dutle, María Consiglio, and James Chamberlain. Daidalus: detect and avoid alerting logic for unmanned systems. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pages 5A1–1. IEEE, 2015. 1.1

[35] Andreas Nussberger, Helmut Grabner, and Luc Van Gool. Aerial object tracking from an airborne platform. In *2014 international conference on unmanned aircraft systems (ICUAS)*, pages 1284–1293. IEEE, 2014. 2.1

[36] NVIDIA. Deepstream sdk, . URL https://developer.nvidia.com/deepstream-sdk. 6.2.7

[37] NVIDIA. Torch-tensorrt, . URL https://github.com/NVIDIA/Torch-TensorRT. 6.2.7

[38] Jiangmiao Pang, Cong Li, Jianping Shi, Zhihai Xu, and Huajun Feng. $\mathcal{R}^2$-cnn: Fast tiny object detection in large-scale remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5512–5524, 2019. 2.3

[39] Jay Patrikar, Brady Moon, Jean Oh, and Sebastian Scherer. Predicting like a pilot: Dataset and method to predict socially-aware aircraft trajectories in non-towered terminal airspace. *arXiv preprint arXiv:2109.15158*, 2021. 3.3

[40] Stavros Petridis, Christopher Geyer, and Sanjiv Singh. Learning to detect aircraft at low resolutions. In *International Conference on Computer Vision Systems*, pages 474–483. Springer, 2008. 2.1

[41] Dmytro Poplavskiy. Seg tracker. URL https://gitlab.aicrowd.com/dmytro_poplavskiy/airborne-detection-starter-kit. 4.2

[42] Ran Qin, Qingjie Liu, Guangshuai Gao, Di Huang, and Yunhong Wang. Mrdet: A multi-head network for accurate oriented object detection in aerial images. *arXiv preprint arXiv:2012.13135*, 2020. 2.3

[43] Vladimir Reilly, Haroon Idrees, and Mubarak Shah. Detection and tracking of large number of targets in wide area surveillance. In *European conference on computer vision*, pages 186–199. Springer, 2010. 2.1

[44] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Flying objects detection from a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4128–4136, 2015. 2.1

[45] Falk Schubert and Krystian Mikolajczyk. Robust registration and filtering for moving object detection in aerial videos. In *2014 22nd International Conference on Pattern Recognition*, pages 2808–2813. IEEE, 2014. 2.1

[46] Bing Shuai, Andrew Berneshawi, Xinyu Li, Davide Modolo, and Joseph Tighe. Siammot: Siamese multi-object tracking. In *Proceedings of the IEEE/CVF*

conference on computer vision and pattern recognition, pages 12372–12382, 2021. 5.2

[47] Vladan Stojnić, Vladimir Risojević, Mario Muštra, Vedran Jovanović, Janja Filipi, Nikola Kezić, and Zdenka Babić. A method for detection of small moving objects in uav videos. *Remote Sensing*, 13(4):653, 2021. 2.2

[48] Stratux. Stratux ads-b. URL http://stratux.me/. 3.3

[49] Zhihong Sun, Jun Chen, Liang Chao, Weijian Ruan, and Mithun Mukherjee. A survey of multiple pedestrian tracking based on tracking-by-detection framework. *IEEE Transactions on Circuits and Systems for Video Technology*, 31 (5):1819–1833, 2020. 4.4

[50] Ultralytics. Yolov5. URL https://github.com/ultralytics/yolov5. 5.2

[51] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020. 2.3, 4.2

[52] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020. 2.2, 4.2, 4.4, 4.4.2, 6.1, 6.2.3

[53] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 840–849, 2019. 2.2