Sejong University

Fast and Robust UAV to UAV Detection and Tracking From Video

AUTHORS

LE XUAN HOANG - 24114545

2025-05-20

Contents

List of Figures

List of Tables

1 Abstract

The proliferation of Unmanned Aerial Vehicles (UAVs) necessitates robust sense-and-avoid capabilities to ensure safe airspace operations, with vision-based systems offering a promising low-cost, lightweight solution. This project focuses on the in-depth study, reproduction, and performance evaluation of the "Fast and Robust UAV to UAV Detection and Tracking From Video" (U2U-D&T) framework developed by J. Li et al. The U2U-D&T system employs a computationally efficient, two-stage architecture that integrates motion analysis (optical flow, perspective transform, background subtraction) with appearance features (hybrid CNN and dense network classifiers) and robust tracking (flow point propagation, Kalman filtering) to detect and track other UAVs from a moving platform. Utilizing the original authors' publicly available codebase and pre-trained models, this work replicated the inference pipeline on the U2U-D&TD dataset. Performance was assessed using F1-score, precision, and recall across five cross-validation folds. The reproduced system achieved an average F1-score of 0.797, precision of 0.794, and recall of 0.818. While these results are slightly lower than those reported in the original publication (F1-score 0.890), they significantly outperform the EPFL baseline method and validate the general efficacy and robustness of the U2U-D&T approach for real-world UAV-to-UAV detection. This study underscores the practical viability of the framework and provides insights into the nuances of reproducing complex computer vision systems.

For the full code implementation, please refer to the GitHub repository: `https://github.com/lexuanhoang120/Fast-and-Robust-UAV-to-UAV-Detection-and-Tracking`

2 Introduction

2.1 The topic of project

Unmanned Aerial Vehicles (UAVs), commonly known as drones, are rapidly transitioning from niche applications to widespread use across diverse sectors, including surveillance, package delivery, remote sensing, and infrastructure inspection. This proliferation, however, brings significant operational challenges, most notably the increasing risk of mid-air collisions as airspace becomes more crowded. To ensure safe and efficient operation, particularly in shared or autonomous flight scenarios, robust "sense-and-avoid" or "see-and-avoid" capabilities are crucial.

While active sensors like Lidar and Radar offer precise environmental data, their size, weight, power consumption, and cost often make them impractical for smaller UAV platforms. Passive visual sensors, specifically low-cost, high-resolution video cameras, present a compelling alternative due to their lightweight nature and low power requirements. However, leveraging video for reliable UAV detection and tracking introduces its own set of complexities. The system must be able to detect potentially small, fast-moving targets from a moving platform, often against cluttered or dynamic backgrounds (like clouds or terrain), and do so with high sensitivity, robustness, and computational efficiency suitable for onboard processing.

Addressing this challenge, the paper "Fast and Robust UAV to UAV Detection and Tracking From Video" by J. Li et al. [1] proposes a novel framework (U2U-D&T) specifically designed for this UAV-to-UAV detection and tracking task using video input from a moving camera. Their approach utilizes a computationally efficient, two-stage architecture integrating motion analysis, appearance features, and temporal tracking to achieve reliable performance even for distant or intermittently visible targets.

2.2 Objective of the project

The primary objective of this project is to conduct an in-depth study and reproduction of the U2U-D&T system presented by Li et al. [1]. This involves:

- **Thoroughly understanding** the proposed two-stage methodology, including the specific algorithms used for target detection (optical flow, background subtraction, hybrid classification) and target tracking (flow point propagation, Kalman filtering).
- **Reproducing the implementation** of the U2U-D&T framework as described in the paper.
- **Evaluating the performance and practicality** of the implemented system, using metrics such as precision, recall, processing speed, and robustness, potentially leveraging the dataset referenced or similar simulated/collected data.
- **Analyzing the strengths and limitations** of the approach based on the results of the reproduction and evaluation, potentially identifying areas for future optimization or improvement.

Ultimately, this project aims to gain a comprehensive technical understanding of the state-of-the-art techniques presented in the paper and validate their effectiveness for real-world UAV sense-and-avoid applications.

2.3 Timeline

The project is conducted in a timeline of 10 weeks:

- Week 1, 2: Paper selection and review of the literature.
- Week 3, 4: Writing proposal as Professor's assignment .
- Week 5, 6: Implementation of the method.
- Week 7: Continue the implementation of the method.
- Week 8, 9: Writing the report and implementation of related methods.
- Week 10: Finalization of the report and implementation of related methods.

3 Literature Review

The increasing deployment of Unmanned Aerial Vehicles (UAVs) across various domains, from commercial delivery and remote sensing [2, 3] to security and coordination of multi-UAV systems [4, 5, 6], necessitates robust mechanisms for safe operation, particularly collision avoidance. As airspace becomes more congested with autonomous or semi-autonomous agents, the ability for a UAV to detect and track other nearby UAVs (UAV-to-UAV or U2U) is paramount for preventing mid-air collisions [5, 7]. This capability is often termed "sense-and-avoid" or, when relying on visual sensors, "see-and-avoid" [8, 9, 7].

Various sensor technologies can be employed for sense-and-avoid. Active sensors like Lidar and Radar can provide direct 3D environmental information, offering relatively accurate range and bearing data [10]. However, these systems often suffer from significant drawbacks for smaller UAV platforms, primarily their weight, power consumption, and cost, making them impractical for many applications [10].

Passive optical sensors, specifically digital video cameras, offer a compelling alternative. They are lightweight, low-power, low-cost, and provide rich scene information [8, 9, 7]. This aligns well with the operational constraints of many UAVs. However, extracting reliable detection and tracking information from video streams presents significant algorithmic challenges, especially when the camera itself is mounted on a moving platform.

The U2U detection and tracking problem using onboard cameras poses unique difficulties distinct from general object detection or tracking from static cameras:

- **Ego-Motion:** The camera platform (the host UAV) is typically moving, often rapidly and non-linearly. This induces a complex background motion that must be compensated for or distinguished from the true target motion [11, 12].
- **Target Motion:** The target UAV is also moving independently, often with a different motion profile than the background or the host UAV [1].
- **Target Appearance Variability:** Target UAVs can appear very small (occupying only a few pixels) when distant, making appearance-based features unreliable [13]. Their appearance can also change drastically with viewing angle and distance. Furthermore, they may blend into complex backgrounds like clouds or terrain (Figure 1 in [1]).
- **Computational Constraints:** Algorithms must be computationally efficient enough to run in real-time or near-real-time on resource-constrained onboard processors common in UAVs.
- **Data Scarcity:** Obtaining high-quality, annotated video datasets specifically for the U2U scenario, involving multiple simultaneously flying UAVs under various conditions, is challenging and expensive [14].

Significant research exists in the broader fields of moving object detection and tracking.

- **Object Detection:** Early methods relied on handcrafted features [15], while modern approaches heavily utilize Deep Convolutional Neural Networks (CNNs) like YOLO [16], SSD, and Faster R-CNN [17]. These have shown excellent performance for general object detection (e.g., pedestrians, cars) but often require substantial computational resources and large annotated datasets. Their effectiveness for detecting small, low-resolution, appearance-variable UAVs without specific fine-tuning can be limited. Some work has focused on detecting ground objects from UAVs [8, 18, 19, 20, 21, 22, 23], but detecting airborne targets presents different motion and background challenges.
- **Background Subtraction:** Techniques exist to separate foreground moving objects from the background. While simpler for static cameras [24], methods for moving cameras typically involve estimating the global ego-motion (e.g., using optical flow and fitting affine or perspective transforms [25]) and warping frames for alignment before subtraction [11, 12]. The accuracy of motion estimation is crucial.
- **Optical Flow:** Methods like Lucas-Kanade [26] estimate pixel-level motion between frames and are fundamental for both motion compensation and extracting motion cues of targets relative to the background.
- **Object Tracking:** Trackers like Kernelized Correlation Filters (KCF) [27] or Kalman Filters [2] are used to maintain object identity and predict positions over time, providing temporal coherence and robustness to intermittent detection failures.

Fewer studies have directly addressed the specific U2U video-based detection and tracking problem.

- **Rozantsev et al. [28, 29]:** Proposed a method involving motion compensation to center potential targets followed by a DNN for appearance classification. This approach was effective but primarily designed for relatively close/large targets where appearance features are discriminative and employed a computationally intensive sliding-window detection mechanism.
- **Saribas et al. [30]:** Combined an appearance-based detector (YOLOv3 [16]) with a KCF tracker [27]. This hybrid approach leverages deep learning for initial detection but may still struggle with small/distant targets where appearance is less reliable and inherits the computational demands of YOLOv3.
- **Li et al. [1]:** This work proposes a computationally efficient two-stage pipeline. It emphasizes distinguishing targets based on motion relative to the background using sparse optical flow, perspective motion modeling, and background subtraction. It combines this motion analysis with appearance features using hybrid classifiers (AdaBoost) applied sparsely only to salient candidate regions. Crucially, it incorporates tracking (flow point propagation and Kalman filtering) not just for continuity but also to improve the detection robustness of faint or intermittently visible targets. Their approach aims to balance accuracy, robustness (especially for small targets), and computational efficiency, and they also contribute a relevant U2U dataset [14].

The literature highlights the critical need for efficient and robust U2U detection and tracking systems for safe UAV operations. While general computer vision techniques provide foundational tools, the specific constraints of the U2U problem (ego-motion, small targets, computational limits) necessitate specialized approaches. Prior U2U systems have shown promise but often rely heavily on appearance features (limiting small target detection) or employ computationally expensive methods. The work by Li et al. [1] presents a well-reasoned approach that explicitly tackles these challenges by combining motion and appearance cues within an efficient, modular framework integrating detection and tracking. Therefore, studying and reproducing this system provides valuable insight into state-of-the-art solutions for this critical problem.

4 System Design

4.1 System Model

The UAV-to-UAV Detection and Tracking (U2U-D&T) system proposed by Li et al. [1] employs a modular, two-stage pipeline architecture to process video input from a camera mounted on a moving UAV platform. The primary goal is to detect and continuously track other UAVs within the camera's field of view in a computationally efficient and robust manner. All of the overall architecture of our U2U-D&T algorithm is illustrated in Figure 1:

- Stage 1: Target Detector: This stage processes frames periodically (every L_d frames) to generate a sparse set of high-probability candidate points that likely correspond to moving UAVs. Its primary function is to efficiently scan the frame and identify potential regions of interest while rejecting most of the background. The Target Detector is implemented in the following five steps detailed below: optical flow estimation on sparsely distributed points, global motion estimation, background subtraction, salient point extraction, and UAV target classification.
- Stage 2: Target Tracker: This stage operates on every frame. It takes the candidate points generated by Stage 1 (when available) to initiate new tracks and uses optical flow and filtering techniques to maintain and refine existing tracks over time, providing continuous and robust tracking even through intermediate frames where the Detector does not run or when detections are temporarily missed.

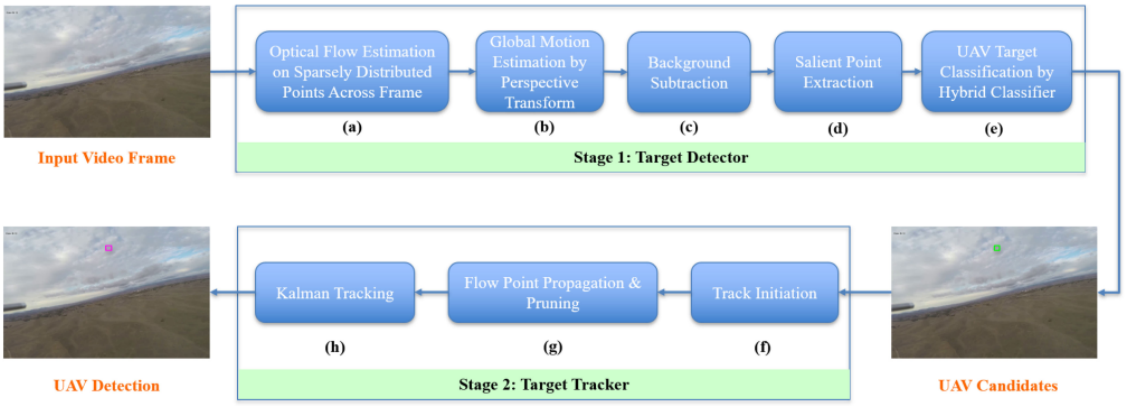

Figure 1: An overview of the U2U-D&T algorithm [1].

4.1.1 Stage 1: Target Detector (executed every L_d frames)

Goal: To efficiently scan the surroundings every few moments (e.g., every 6 frames) and identify potential locations where something interesting (possibly a target UAV) might be. It's designed to be

quick and avoid analyzing every single detail all the time.

Input: Current video frame X_n and the previous frame X_{n-1}.

(a) Optical Flow Estimation on Sparsely Distributed Points

Purpose: To get an initial, computationally cheap sense of how different parts of the scene are moving between the previous frame (X_{n-1}) and the current frame (X_n). This is the first step in understanding motion.

Process

- **Select Feature Points:** In the previous frame (X_{n-1}), the algorithm selects K feature points. These are not just random pixels; they are chosen to be "good features to track" - typically corners or textured areas that are easy to distinguish. Initially, points are selected at uniformly distributed random locations and then evaluated using the Shi-Tomasi corner detector [31]. Points with higher saliency are preferred. To ensure spatial diversity and avoid clustering, any point within a distance D_o of a more salient point is discarded. This results in a set of K_n feature points, denoted $p_{n,i}$, in frame X_{n-1}. ($p_{n,i}$ is a 2D vector that determines a discrete pixel location in the frame.)
- **Forward Optical Flow:** For each feature point $p_{n,i}$ in X_{n-1}, the algorithm estimates its corresponding position in X_n using the Lucas-Kanade method [26]. This method finds the displacement $u_{n,i}$ by analyzing a local neighborhood around $p_{n,i}$, denoted $\mathcal{N}(p_{n,i})$. The estimated new position in X_n is:

$$\tilde{p}_{n,i} = p_{n,i} + u_{n,i} \tag{1}$$

  where: $\tilde{p}_{n,i}$ is a continuous 2D position specified to subpixel accuracy.
  This flow estimation is computed to sub-pixel accuracy by solving the least squares problem.

$$u_{n,i} = \arg\min_{u} \sum_{s \in \mathcal{N}(p_{n,i})} \|X_n(s+u) - X_{n-1}(s)\|^2 \tag{2}$$

- **Backward Optical Flow (Bi-directional Check):** To enhance reliability, the algorithm performs a backward flow estimation. Starting from the estimated position $\tilde{p}_{n,i}$ in X_n, the flow is tracked back to X_{n-1}, yielding the backward flow vector $v_{n,i}$.

$$v_{n,i} = \arg\min_{v} \sum_{s \in \mathcal{N}(\tilde{p}_{n,i})} \|X_n(s) - X_{n-1}(s-v)\|^2 \tag{3}$$

- **Pruning Unreliable Points:** Ideally, if a point is part of the rigid background and accurately tracked, then $u_{n,i}$ and $v_{n,i}$ should be nearly equal in magnitude and opposite in

direction. The algorithm computes the Euclidean distance:

$$\|u_{n,i} - v_{n,i}\|_2$$

If this value exceeds a threshold M_d, the point is considered unreliable (possibly due to occlusion, non-rigid motion, or errors) and is discarded.

Output A set of reliable point correspondences between X_{n-1} and X_n, along with their flow vectors. These points are assumed to primarily lie on the background.

(b) Global Motion Estimation by Perspective Transform

Purpose To model the overall motion of the background scene caused by the host UAV's movement (ego-motion). If we can accurately describe how the background should move, then anything moving differently is a potential target.

Process

- **Perspective Transform Model:** The algorithm assumes that background motion between two frames can be approximated by a 2D perspective transform (homography). This is a valid assumption if the background is planar (e.g., ground observed from altitude) or distant. The transform is represented by a 3×3 matrix H, with 8 degrees of freedom (typically normalized such that $h_{33} = 1$). The transformation of a point x is defined as:

$$y = T(x; H) = Hx = \begin{bmatrix} \frac{h_{11}x_1 + h_{12}x_2 + h_{13}}{h_{31}x_1 + h_{32}x_2 + h_{33}} \\ \frac{h_{21}x_1 + h_{22}x_2 + h_{23}}{h_{31}x_1 + h_{32}x_2 + h_{33}} \end{bmatrix} \tag{4}$$

where the homogeneous coordinates of x and y are used.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

H be the 3×3 homography matrix with elements h_{ij}, i.e.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}.$$

- **Fitting the Model:** Using the reliable point correspondences from the previous step, namely $p_{n,i}$ in frame X_{n-1} and their matched points $\tilde{p}_{n,i}$ in X_n, the goal is to estimate the transform H_n that minimizes the total reprojection error:

$$H_n = \arg\min_H \sum_{i \in S_n} \|\tilde{p}_{n,i} - T(p_{n,i}; H)\|_2^2$$

where the optimization is performed using an iterative least squares method [32], and S_n denotes a subset of original K_n detected points designed to remove outliers. The subset, S_n, is constructed by first removing points using the bi-directional pruning method described above.

- **RANSAC (RANdom SAmple Consensus) [33]:** Since some correspondences might belong to moving foreground objects (e.g., other UAVs), the algorithm employs RANSAC to robustly estimate the homography:
  (i) Randomly select a minimal subset of point correspondences (at least 4 for homography).
  (ii) Estimate H from this subset.
  (iii) Count how many other correspondences agree with this H (i.e., are inliers whose reprojection error is below a threshold).
  (iv) Repeat the above steps multiple times.
  (v) Select the H with the highest inlier count.
- **Final Estimation:** Once RANSAC identifies the inlier set S_n, the final homography matrix H_n is recomputed using all inliers through a standard least-squares optimization for better accuracy.

Output The estimated perspective transform matrix H_n that best describes the dominant background motion from frame X_{n-1} to X_n.

(c) Background Subtraction

Purpose To create an image that explicitly highlights regions that are not moving according to the estimated global background motion.

Process

- **Warping the Previous Frame:** Using the estimated perspective transform H_n, the previous frame X_{n-1} is warped into the coordinate system of the current frame X_n. The motion-compensated version of the previous frame is denoted:

$$\hat{X}_{n-1}(s) = X_n(T(s; H_n)) \tag{5}$$

  where $T(s; H_n)$ applies the homography H_n to pixel location s. Since $T(s; H_n)$ may yield non-integer coordinates, bilinear interpolation is used to sample pixel intensities from X_n.
- **Computing the Difference Image:** The background-subtracted image E_{n-1} is computed as the absolute difference between the original previous frame X_{n-1} and the warped version $\hat{X}_{n-1}$:

$$E_{n-1}(s) = \left| X_{n-1}(s) - \hat{X}_{n-1}(s) \right| \tag{6}$$

This formulation implies that for each pixel s in X_{n-1}, its intensity is compared to that of the corresponding (motion-predicted) location in X_n.

- **Interpretation:** The difference image E_{n-1} emphasizes regions where the global motion model fails to accurately predict pixel movement-i.e., where motion is inconsistent with the background. These discrepancies typically correspond to independently moving foreground objects (e.g., UAVs).

Output A background-subtracted image E_{n-1} in which regions of high intensity represent motion inconsistent with the global perspective transform-potential indicators of moving targets.

(d) Salient Point Extraction

Purpose To identify a sparse set of the most "promising" locations within the background-subtracted image E_{n-1} to pass on for more detailed classification. This reduces computational cost by avoiding dense pixel-wise processing.

Process

- **Corner Detection:** The Shi-Tomasi corner detector [31] is applied to the background-subtracted image E_{n-1}. Corners often correspond to distinctive, high-contrast regions that may indicate the presence of moving objects.
- **Parameter Control:** The detection is governed by the following parameters:
  - λ^c: Quality level threshold, representing the minimum accepted eigenvalue of the gradient matrix at a candidate point.
  - d^c: Minimum allowable distance between detected salient points, ensuring spatial diversity and preventing clustering.
  - m^c: Maximum number of salient points to be retained. The final number of selected points is denoted Q_n.
- **Patch Extraction:** For each of the Q_n detected salient points $q_{n,i}$, two image patches are extracted, centered at $q_{n,i}$:
  - A patch from the original current color image X_n, capturing appearance information.
  - A patch from the background-subtracted image E_{n-1}, capturing motion-highlighted information.

  Typically, these patches are of fixed size (e.g., 40×40 pixels).

Output A set of Q_n salient points $\{q_{n,i}\}$, each associated with a pair of corresponding patches from X_n and E_{n-1}, which are passed to the classification module for further analysis.

(e) UAV Target Classification by Hybrid Classifier I

Purpose To make a final decision for each salient point: is it a true moving target (UAV), or is it just noise, a background feature that was poorly compensated, or some other non-target motion?

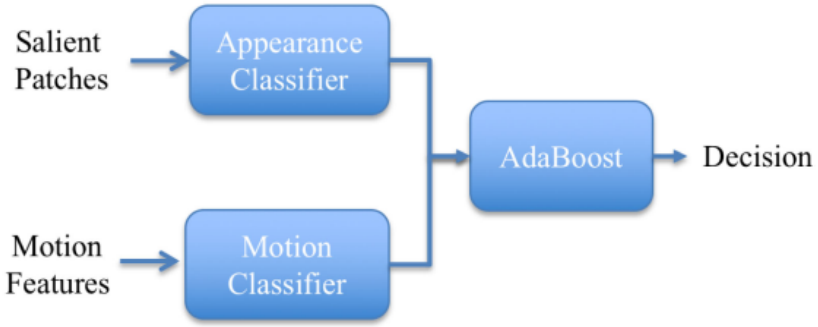

Figure 2: The Hybrid Classifier uses AdaBoost to compute a classification decision based on the combined results of an appearance and motion classifier [1]

Process

- **Hybrid Approach:** The classification system integrates two complementary classifiers-an appearance-based CNN and a motion-based dense network - whose outputs are fused using the AdaBoost algorithm [34] as depicted in Figure 2.
- **Appearance Classifier (CNN):**

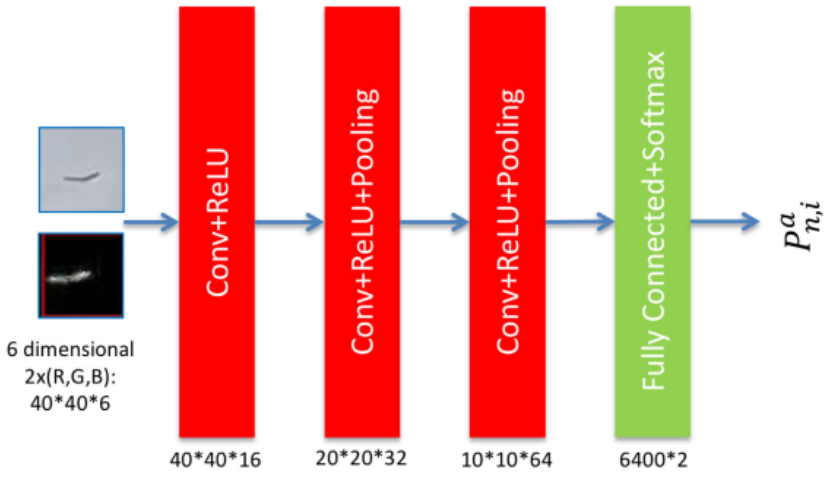

Figure 3: Appearance classifier architecture [1]

  - **Input:** A $40 \times 40 \times 6$ tensor, created by stacking the color patch from X_n and the corresponding motion patch from E_{n-1}.
  - **Architecture:** A Convolutional Neural Network (see Figure 3), consisting of three convolutional layers with ReLU activations, max-pooling, dropout, batch normalization, followed by a fully connected layer and a softmax output.

- **Output:** A probability $P^a_{n,i}$ indicating the likelihood that the patch of each salient point $q_{n,i}$, corresponds to a true moving target based on appearance.

- **Motion Classifier (Dense Neural Network):**

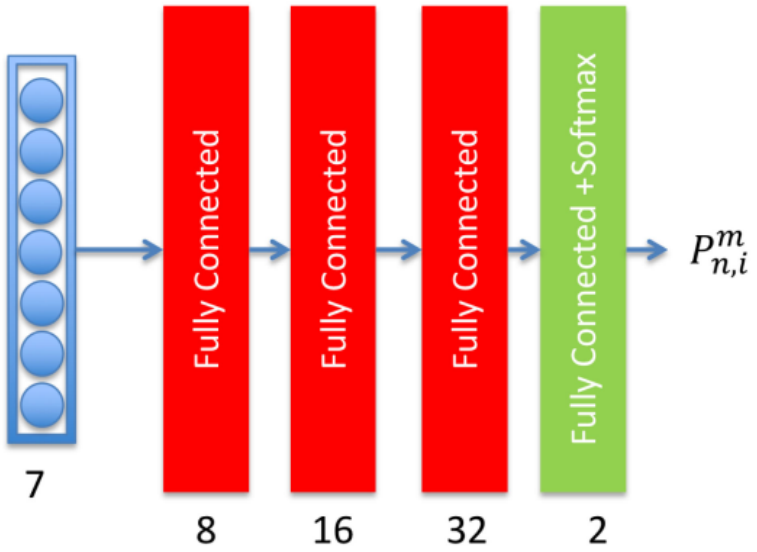

Figure 4: Motion classifier architecture [1]

  - **Input:** A 7-dimensional motion feature vector for each salient point $q_{n,i}$. These features are derived as follows:
    - Estimate the local optical flow vector $\mu_{n,i}$ at $q_{n,i}$ using the Lucas-Kanade method.
    - Compute the expected background motion at $q_{n,i}$ using the global transform: $h_{n,i} = T(q_{n,i}; H_n) - q_{n,i}$.
    - Calculate the residual (relative) motion: $d_{n,i} = \mu_{n,i} - h_{n,i}$.
    - Extract 7 features: magnitude and angle of $d_{n,i}$, bi-directional flow error for $\mu_{n,i}$, angle difference between $\mu_{n,i}$ and $h_{n,i}$, magnitude difference between $\|\mu_{n,i}\|$ and $\|h_{n,i}\|$, etc. (see Table 1).
  - **Architecture:** A dense neural network (see Figure 4) composed of three fully connected layers with dropout, batch normalization, and softmax.
  - **Output:** A probability $P^m_{n,i}$ that the point corresponds to a moving target based on motion characteristics.

- **AdaBoost Fusion:**
  - The two classifier outputs $(P^a_{n,i}, P^m_{n,i})$ are combined using AdaBoost, which learns to optimally weight and fuse the two inputs.The input to the AdaBoost classifier is the

2D vector and the parameters d_m and M_0 were used to specify the maximum tree depth and the maximum number of decision trees, respectively.

– The final decision is a hard classification: either the point $q_{n,i}$ is a UAV target or not. Discard any salient points that are classified as false alarms.

Output A sparse set of UAV candidate points $\{c_{n,i}\}$, each identified as a target by the Hybrid Classifier I. These candidates are passed to Stage 2 for temporal consistency checks and tracking.

Table 1: Table of motion features.

Equation	Description
1. $d_{n,i} = \mu_{n,i} - h_{n,i}$	Target motion
2. $l_{n,i} = \|d_{n,i}\|$	Magnitude of target motion
3. $\alpha_{n,i} = \arctan(d_{n,i})$	Angle of target motion
4. $\zeta_{n,i} = \|\mu_{n,i} - \nu_{n,i}\|$	Bi-directional verification distance
5. $\phi_{n,i} = \arctan(h_{n,i}) - \arctan(\mu_{n,i})$	Angle difference between global and local motion
6. $\delta_{n,i} = \|\|\mu_{n,i}\| - \|h_{n,i}\|\|$	Magnitude difference between global and local motion

4.1.2 Stage 2: Target Tracker (executed every frame)

Goal: To take the "Candidates" identified by the Spotter, confirm if they are actual targets, and then meticulously follow them frame-by-frame, maintaining a smooth track even if the Spotter isn't running or the target becomes temporarily hard to see.

Input: Current video frame (X_n), UAV candidates ($c_{n,i}$ - only on detection frames), and the state of existing tracks from the previous frame ($t_{n-1,j}$, associated flow points $r_{n-1,j,l}$, Kalman state $b_{n-1,j}$).

(f) Track Initiation

Purpose: To decide whether a new UAV candidate point (from Stage 1) corresponds to an actual new UAV that is not already being tracked, and if so, to initiate a new tracking process.

Process

- **Receive Candidates:** The system receives the list of UAV candidate points $\{c_{n,i}\}$ from Stage 1.
- **Compare with Existing Tracks:** Each candidate $c_{n,i}$ is compared to all currently active tracks. Let $t_{n,j}$ be the current estimated position of the j-th track, and $k_{n,j}$ be its estimated relative velocity (as maintained by the Kalman filter in step (h)).

- **New Track Criteria:** A candidate $c_{n,i}$ is designated to initiate a new track if, for all existing tracks j, at least one of the following conditions holds:

$$\|c_{n,i} - t_{n,j}\| > T_d \quad \text{(spatial distance threshold)}$$
$$\|d_{n,i} - k_{n,j}\| > T_v \quad \text{(velocity difference threshold)}$$

  where $d_{n,i}$ is the estimated relative velocity of candidate $c_{n,i}$, as computed in Stage 1(e). This ensures that the candidate is sufficiently distinct from any existing track, either spatially or in motion characteristics.

- **Flow Point Extraction for New Tracks:** If a new track is initiated:
  - A small image patch (e.g., $N_c \times N_c$ pixels) centered at $c_{n,i}$ is extracted from the current frame X_n.
  - Within this patch, multiple trackable points (called flow points) are extracted using the Shi-Tomasi corner detector. Let these be denoted $r_{n,j,l}$, where j is the index of the new track and l indexes the flow points.
  - The corner detection parameters (λ^t, d^t, m^t) may be different from those used in Stage 1(d) to allow for denser or more localized feature selection.

Output An updated list of active tracks. This list may now include newly initiated tracks, each initialized with a set of flow points $\{r_{n,j,l}\}$ for subsequent tracking.

(g) Flow Point Propagation & Pruning

Purpose To accurately track the individual features (flow points) of each tracked target from the current frame to the next and to remove any flow points that are no longer reliably tracking the target. This process runs on every frame for all active tracks.

Process

- **Flow Point Propagation:** For each active track j and each of its associated flow points $r_{n,j,l}$ in the current frame X_n:
  - The Lucas-Kanade optical flow method [26] is applied to estimate the local motion vector $u_{n,j,l}$ for the flow point.
  - The flow point is then propagated to its new estimated position in the next frame as:

$$r_{n+1,j,l} = r_{n,j,l} + u_{n,j,l}$$

- **Flow Point Pruning (Hybrid Classifier II):** After propagation, each flow point $r_{n+1,j,l}$ needs to be validated to ensure it still represents the target and hasn't drifted or become occluded.
  - **Input to Classifier:**

    - An image patch is extracted from the original color image X_{n+1}, centered at the propagated flow point $r_{n+1,j,l}$.
    - A 7D motion feature vector $f_{n+1,j,l}$ is computed for the flow point $r_{n+1,j,l}$, similar to how motion features were computed in Stage 1(e). This involves comparing its local optical flow with the predicted global background motion at its location.
  - **Classifier Action:** Hybrid Classifier II (similar in structure to Hybrid Classifier I - an AdaBoost fusion of an appearance CNN and a motion DNN, but likely trained specifically for this flow point pruning task) classifies the flow point as either:
    - *Target*: The flow point is still reliable and corresponds to the target.
    - *No Target*: The flow point is considered unreliable, likely due to drift or occlusion.
  - **Pruning:** If a flow point is classified as "no target," it is removed from the set of flow points for that track.
- **Track Termination (Empty Track Check):** If, after pruning, a track j has no remaining flow points, it is marked as "empty."
  - If a track remains "empty" for a predefined number of consecutive frames L (the "maximum unseen frames" parameter), the track is considered lost and is terminated (deleted from the list of active tracks).

Output An updated set of valid, propagated flow points $r_{n+1,j,l}$ for each active track in the next frame. Some tracks may be terminated if they are empty.

(h) Kalman Tracking

Purpose To provide a smoothed and robust estimate of each target's overall position and, crucially, its velocity relative to the background. The Kalman filter helps to predict the target's motion, bridge short gaps in tracking, and reduce the impact of noisy measurements. This process runs on every frame for all active tracks.

Process

- **Preliminary Track Location:** After flow point propagation and pruning, the valid flow points $r_{n+1,j,l}$ for a track j are aggregated (e.g., by taking their centroid) to compute a preliminary estimate of the track's location in the next frame, $\tilde{t}_{n+1,j}$.

$$\tilde{t}_{n+1,j} = \frac{1}{|T_{n,j}|} \sum_{l \in T_{n,j}} r_{n+1,j,l} \tag{7}$$

  where $T_{n,j}$ is the set of current flow points for j^{th} track in frame n.
- **Measure Local Motion of Track:** The local motion of the track itself is given by:

$$v^l_{n,j} = \tilde{t}_{n+1,j} - t_{n,j} \tag{8}$$

which represents the difference between the preliminary new position and the previous position.

- **Measure Predicted Global Motion at Track Location:** The motion that the background would have had at the track's previous location $t_{n,j}$ is calculated using the global perspective transform H_n (estimated in Stage 1 but used here for the current interval):

$$v_{n,j}^g = T(t_{n,j}; H_n) - t_{n,j} \tag{9}$$

- **Calculate Measured Relative Velocity:** The key measurement for the Kalman filter is the target's velocity relative to the background:

$$k_{n,j} = v_{n,j}^l - v_{n,j}^g \tag{10}$$

This isolates the target's own motion from the host UAV's ego-motion.

- **Kalman Filter Update:**
  - **State Vector:** The Kalman filter maintains a state $b_{n,j} = [k_{n,j}^T, \dot{k}_{n,j}^T]^T$ for each track. This state vector includes the estimated relative velocity $k_{n,j}$ and its derivative, the relative acceleration $\dot{k}_{n,j}$.
  - **Prediction Step:** The filter predicts the next state based on a motion model (e.g., constant relative acceleration, meaning k changes linearly, or constant relative velocity, meaning k is constant). The paper uses a "constant velocity model" for $k_{n,j}$, which assumes constant relative acceleration of the target. The state transition matrix A and measurement matrix M define this model.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{12}$$

  - **Measurement Update Step:** The measured relative velocity $k_{n,j}$ (from Equation 10) is used to correct the predicted state. The filter optimally combines the prediction with the noisy measurement based on their respective uncertainties ($\sigma_w^2 I$ for process noise, $\sigma_e^2 I$ for measurement noise).
  - **Estimate Final Track Position:** The final, smoothed position of the track for the next frame $t_{n+1,j}$ is estimated by:

$$t_{n+1,j} = t_{n,j} + v_{n,j}^g + k_{n,j} \tag{13}$$

where $k_{n,j}$ is the velocity component from the updated Kalman state $b_{n,j}$.

– **Detection Bounding Box:** A detection bounding box of a fixed size N_c (or dynamically adjusted) is centered at the final estimated track position $t_{n+1,j}$.

Output For each active track, a refined estimate of its position $t_{n+1,j}$ and its relative velocity/acceleration state $b_{n+1,j}$.

Table 2: Table of parameters.

Notation	Description	Value
K	Number of selected feature points	600
D_0	Minimum distance between feature points	25
M_d	Bidirectional check threshold for local motion	1
c	Quality level of salient patch	0.01
d_c	Minimum distance between salient patches	50
m_c	Maximum number of salient patches	600
L_0	Detection interval	6
d_m	Maximum depth of decision tree	2
N_c	Size of candidate patch	40
M_0	Number of estimators in AdaBoost	20
t	Quality level of flow points	0.001
d_t	Minimum distance between flow points	1
m_t	Maximum number of flow points	50
σ_v	Transition modeling error level	0.1
σ_e	Measurement error level	1.0
T_d	Minimum distance to initiate new track	20
T_v	Minimum motion difference to initiate new track	0.5
L	Maximum unseen frames for Kalman tracking	6

4.1.3 Key Characteristics of the Model

- **Efficiency:** Achieved through sparse processing (sparse optical flow, salient points) and periodic execution of the computationally heavier detection stage.
- **Hybrid Feature Usage:** Integrates both motion cues (relative motion, optical flow consistency) and appearance cues (CNN features) for robust classification, particularly important for distinguishing small targets from noise.
- **Tracking for Robustness:** Uses continuous tracking (flow points, Kalman filter) not just to link detections over time but also to maintain tracks during temporary detection misses (e.g., due to occlusion or appearance changes), enhancing the ability to follow faint or intermittently visible targets.

- **Modularity:** The distinct detector and tracker stages allow for potential independent improvements or modifications.

This system model provides a structured overview of how the U2U-D&T algorithm processes video data to achieve fast and robust UAV detection and tracking.

4.2 Dataset for U2U Detection and Tracking

The experimental validation and performance evaluation of the proposed U2U-D&T algorithm, as detailed in Li et al. [1], were conducted using a specialized video dataset, referred to as U2U-D&TD. This dataset was specifically created by the authors to address the unique challenges of UAV-to-UAV detection and tracking from a moving airborne platform, a scenario for which publicly available datasets were scarce.

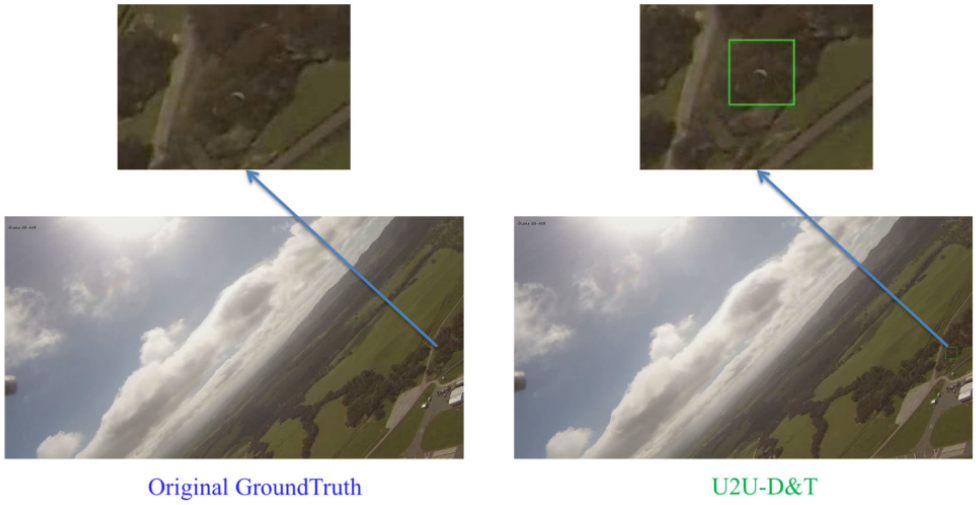

Figure 5: Missed annotation in the first round of annotations. Left is original ground truth, Right is U2U-D&T's detection results (Green box is detection). U2U-D&T detects UAVs missed by human eye [1].

Dataset Characteristics:

- **Content and Collection:** The U2U-D&TD dataset comprises 50 distinct video sequences. These were captured using a GoPro 3 camera mounted on a custom delta-wing airframe (the host UAV). Up to 8 UAVs may appear in a single frame. The sequences were sourced from approximately 100 hours of flight footage.
- **Recording Conditions:** The dataset includes outdoor flight conditions, capturing real-world challenges such as:

  - *Illumination Variations:* Changing lighting conditions typical of outdoor environments.
  - *Background Clutter:* Diverse and complex backgrounds including sky, clouds, and terrain.
  - *Small Target Objects:* Target UAVs often appear small, especially when distant.
  - *Varied Target Appearances and Shapes:* Includes UAVs with different physical designs and appearances.

- **Video Specifications:**
  - *Frame Rate:* 30 frames per second (fps).
  - *Duration:* Approximately one minute per sequence.
  - *Resolution:* HD videos at either 1920×1080 or 1280×960 pixels.

- **Preprocessing:** A step was taken to mask the pitot tube of the host UAV—visible in some frames—to prevent it from being misidentified as a target.

- **Ground Truth Annotation:** Manual annotations of target UAVs were performed using VATIC software [35]. Bounding boxes were refined in some cases after algorithmic detections helped identify missed targets as depicted in Figure 5.

- **Public Availability:** The dataset and its ground truth annotations are publicly available at `https://engineering.purdue.edu/~bouman/UAV_Dataset/`, facilitating reproducibility and further research.

4.3 Performance Evaluation Metrics

To quantitatively assess the performance of the proposed UAV-to-UAV Detection and Tracking (U2U-D&T) algorithm, Li *et al.* [1] employed standard and widely recognized metrics from the field of object detection and computer vision. These metrics allow for a comprehensive evaluation of the algorithm's accuracy, robustness, and efficiency.

4.3.1 Adjusted Intersection over Union (AIoU)

A key metric for evaluating detection accuracy is the *Adjusted Intersection over Union (AIoU)*, which is a modified version of the original *Intersection over Union (IoU)*. In this report, AIoU is defined as:

$$\text{AIoU} = \frac{\text{Area}(\text{Detected Box} \cap \text{Ground Truth Box})}{\text{Area}(\text{Detected Box})} \tag{14}$$

A detection is considered a *True Positive (TP)* if its AIoU with a ground truth object is greater than or equal to a predefined threshold, typically 0.5 (i.e., IoU ≥ 0.5).

4.3.2 Detection Outcome Categories

- **True Positives (TP)**: Ground truth UAVs correctly detected by the algorithm (AIoU ≥ 0.5).
- **False Positives (FP)**: Detections with AIoU < 0.5 or no corresponding ground truth (false alarms).
- **False Negatives (FN)**: Ground truth UAVs missed by the algorithm.

4.3.3 Evaluation Metrics

Based on the above counts, the following metrics are computed:

- **Precision**: Measures the proportion of correct detections among all algorithm-generated detections.
$$\text{Precision} = \frac{TP}{TP + FP} \tag{15}$$
- **Recall** (Sensitivity or True Positive Rate): Measures the proportion of ground truth UAVs that are successfully detected.
$$\text{Recall} = \frac{TP}{TP + FN} \tag{16}$$
- **F-Score** (F1-Score): Harmonic mean of Precision and Recall, providing a balanced performance measure.
$$\text{F-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}$$

The paper reports these metrics, Precision, Recall, and F-Score, along with their mean and standard deviation, typically obtained via a 5-fold cross-validation procedure.

5 Code Implementation

The implementation of this project is based on the work presented in [1]. All of the code was written in Python and used to infer the results. All the work from original paper is here[1]. In the range of this project, the training code is not implemented. The model is taken from the author's trained models.

5.1 Device specification

- **CPU:** Intel® Core™ i9-10900K CPU @ 3.70GHz×20
- **GPU:** 2×RTX 3090 (24GB)
- **Memory:** 128 GB
- **Storage:** 2.5 TB

5.2 Environmental Setup

These libraries must be installed prior to running the experiment. The version of Python used in this project is `3.7.16`. Below are some essential libraries required for the project:

- `tensorflow==2.11.0`
- `pandas==1.3.5`
- `opencv-python==4.11.0.86`
- `scikit-learn==0.22.1`
- `numpy==1.21.6`
- `openpyxl==3.1.3`

We can install the required libraries using the following command in the terminal:

```
pip install -r requirements.txt
```

[1] https://github.com/jingliinpurdue/Fast-and-Robust-UAV-to-UAV-Detection-and-Tracking

5.3 Structure of project folder

Figure 6 describe the structure of project folder. Below is the description for each element in the project.

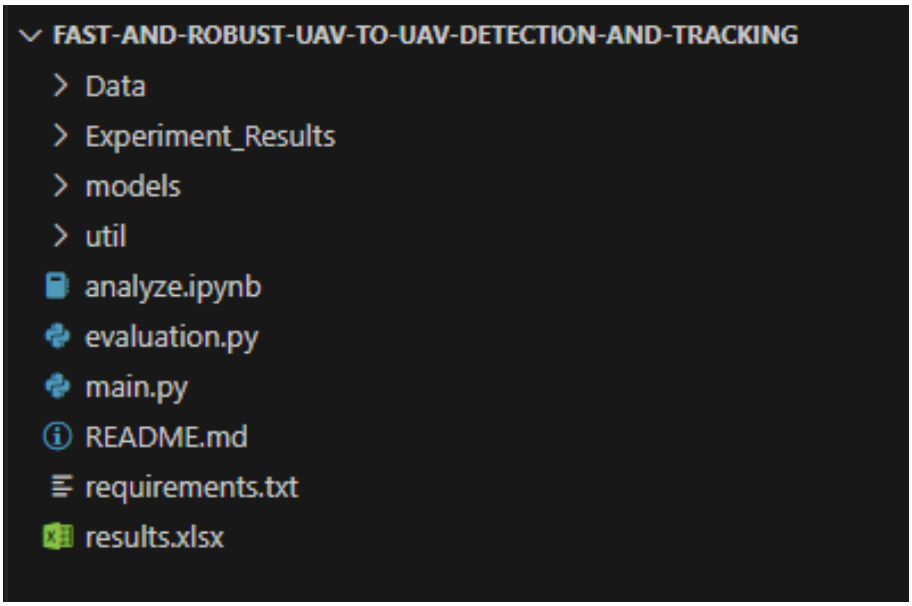

Figure 6: The structure of project folder

- **Data/:** Contains the datasets, video file, and the corresponding annotations. The video file is downloaded from https://engineering.purdue.edu/ bouman/UAV_Dataset/.
- **Experiment_Results/:** Stores evaluation results for each fold from 1 to 5.
- **models/:** Stores the trained models related to the components described in Section 4.1. These are provided by the authors.
- **util/:** Contains utility functions used throughout the project.
- **analyze.ipynb:** Contains some initial analysis of the evaluation results from `results.xlsx` file.
- **evaluation.py:** Contains code to compute the F1-score, precision, and recall for each fold. Running this code generates the `results.xlsx` file.
- **main.py:** Used to run inference and predict objects for each video in the folds.
- **README.md:** Provides an overview, setup instructions, and usage guide for the project.

- **requirements.txt:** Lists all Python dependencies required for the project.
- **results.xlsx:** Contains the final evaluation results.

5.4 Replicating the Experiments

All files are organized to align with the experimental setup described in this report. To reproduce the results, simply follow the steps below in order. For a detailed understanding of each script's functionality, refer directly to the source code and in-line comments.

- **Step 1: Infer the models for each fold**
  Run the following command in the terminal to perform inference for all folds (from 1 to 5):

```
python main.py
```

- **Step 2: Evaluate the inference results**
  After inference, use the following command to evaluate the results by computing the F1-score, precision, and recall for each fold:

```
python evaluation.py
```

- **Step 3: Analyze the results**
  For analysis, open and inspect the generated Excel file:

```
results.xlsx
```

6 Performance Analysis

In this project, we aimed to reproduce the results reported in the original paper by Li *et al.* [1]. Using the pre-trained models provided by the authors, we evaluated the performance across five cross-validation folds. The evaluation metrics considered include F1-score, precision, and recall, as summarized in Table 3.

Table 3: The results of evaluation metrics per fold

Fold	F1-score	Precision	Recall
1	0.813540 ± 0.083980	0.830319 ± 0.098427	0.803663 ± 0.096336
2	0.805948 ± 0.111985	0.791894 ± 0.139918	0.829417 ± 0.099560
3	0.829120 ± 0.109578	0.852153 ± 0.125755	0.812112 ± 0.110633
4	0.788515 ± 0.144919	0.756433 ± 0.205855	0.852876 ± 0.075412
5	0.748895 ± 0.110603	0.739906 ± 0.168534	0.793840 ± 0.117211

Overall, the results demonstrate relatively consistent performance across folds, with Fold 3 achieving the highest F1-score (0.8291) and precision (0.8522), while Fold 5 shows the lowest values across all metrics. Fold 4 achieves the highest recall (0.8529), indicating strong sensitivity in that subset, albeit with more variability in precision.

The standard deviations across folds are moderately high, especially for precision in Fold 4 (± 0.2059), suggesting that the model's performance may vary depending on the specific subset of the data. This variability highlights the importance of using multiple folds to provide a robust estimate of model performance.

Table 4: Comparison of precision, recall and F1-score on this project, original paper, and EPFL

Metric	This report	Original paper	EPFL
F1-Score	0.797 ± 0.11	0.890 ± 0.07	0.515 ± 0.14
Precision	0.794 ± 0.152	0.880 ± 0.10	0.648 ± 0.16
Recall	0.818 ± 0.09	0.892 ± 0.06	0.427 ± 0.15

Table 4 presents a comparative summary between our results, those reported in the original paper, and a baseline method (EPFL) [1]. Although our reproduced results fall short of the original paper - especially in terms of F1-score (0.797 vs. 0.890) - they significantly outperform the EPFL baseline (0.797 vs. 0.515). Notably, the recall in our results (0.818) remains fairly close to that in the original paper (0.892), suggesting that our model retains good sensitivity, even if overall classification balance (as reflected in F1-score) is slightly reduced.

The performance gap between this report and the original publication may be attributed to several factors, including differences in data pre-processing, evaluation conditions, or subtle implementation

details not fully captured in the replication. Nonetheless, the consistency of the results across folds and the significant improvement over the EPFL baseline confirm the effectiveness and generalizability of the proposed method.

For all detail result, please refer this link.

7 Conclusion

This project successfully undertook the reproduction and performance evaluation of the *Fast and Robust UAV to UAV Detection and Tracking* (U2U-D&T) system proposed by Li et al.[1], a significant contribution to vision-based sense-and-avoid technology for Unmanned Aerial Vehicles. The core of this endeavor involved a comprehensive study of the original paper's methodology, particularly its innovative two-stage architecture that synergizes motion analysis with appearance-based classification and Kalman filter-based tracking. By leveraging the authors' provided pre-trained models and codebase, we replicated the inference process on the specialized U2U-D&TD dataset.

The empirical evaluation across five cross-validation folds yielded an average F1-score of 0.797, precision of 0.794, and recall of 0.818. These metrics, while indicating a performance slightly below that reported in the original paper (F1-score of 0.890), robustly demonstrate the system's effectiveness, significantly outperforming the EPFL baseline. This confirms the U2U-D&T framework's capability to reliably detect and track UAVs under challenging, real-world airborne conditions. The observed discrepancies may stem from subtle differences in environmental setup, evaluation scripts, or minor undocumented pre-processing steps, highlighting the inherent challenges in achieving exact replication in complex deep learning and computer vision systems.

This project did not involve retraining the models; rather, it focused on understanding and validating the inference capabilities of the provided system. Future work could delve into retraining the models to potentially bridge the performance gap or explore optimizations within the existing modules.

Ultimately, this project provided valuable insights into the practical application of advanced computer vision techniques for UAV safety. It allowed for a deep understanding of the U2U-D&T system's components, their interplay, and the overall effectiveness of the approach. The process of reproducing the experiments underscored the strengths of the original work and offered a practical learning experience in evaluating state-of-the-art research in UAV detection and tracking.

References

[1] J. Li, D. H. Ye, M. Kolsch, J. P. Wachs, and C. A. Bouman, "Fast and robust uav to uav detection and tracking from video," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, pp. 1519–1531, July-Sept 2022.

[2] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for uavs: Current developments and trends," *Journal of Intelligent & Robotic Systems*, vol. 87, no. 1, pp. 141–168, 2017.

[3] G. Loianno *et al.*, "Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1576–1583, July 2018.

[4] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "A hierarchical detection and response system to enhance security against lethal cyber-attacks in uav networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, pp. 1594–1606, September 2018.

[5] Y. Lin and S. Saripalli, "Sampling-based path planning for uav collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 3179–3192, November 2017.

[6] S. Rabinovich, *Multi-UAV coordination for uncertainty suppression of natural disasters*. PhD thesis, University of California, Santa Cruz, 2018.

[7] D. Bratanov, L. Mejias, and J. J. Ford, "A vision-based sense-and-avoid system tested on a scaneagle uav," in *Proceedings of the International Conference on Unmanned Aircraft Systems*, pp. 1134–1142, 2017.

[8] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *Proceedings of the International Conference on Robotics and Automation*, pp. 2992–2997, 2011.

[9] S. Roelofsen, D. Gillet, and A. Martinoli, "Reciprocal collision avoidance for quadrotors using on-board visual detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4810–4817, 2015.

[10] K. May and N. Krouglicof, "Moving target detection for sense and avoid using regional phase correlation," in *Proceedings of the International Conference on Robotics and Automation*, pp. 4767–4772, 2013.

[11] A. Elqursh and A. Elgammal, "Online moving camera background subtraction," in *Proceedings of the European Conference on Computer Vision*, pp. 228–241, 2012.

[12] D. Zamalieva and A. Yilmaz, "Background subtraction for the moving camera: A geometric approach," *Computer Vision and Image Understanding*, vol. 127, pp. 73–85, 2014.

[13] R. LaLonde, D. Zhang, and M. Shah, "Clusternet: Detecting small objects in large scenes by exploiting spatio-temporal information," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 4003–4012, 2018.

[14] T. H. Chung, M. R. Clement, M. A. Day, K. D. Jones, D. Davis, and M. Jones, "Live-fly, large-scale field experimentation for large numbers of fixed-wing uavs," in *Proceedings of the International Conference on Robotics and Automation*, pp. 1255–1262, 2016.

[15] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," 2019. arXiv:1905.05055.

[16] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision*, pp. 404–417, 2006.

[17] L. Liu *et al.*, "Deep learning for generic object detection: A survey," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, 2020.

[18] T. M. Cheng, T. S. Lim, and M. Y. S. Liew, "Vision-based detection and tracking of aerial objects for uav collision avoidance," *Procedia Computer Science*, vol. 76, pp. 349–354, 2015.

[19] C. Torance and K. Bowyer, "Background modeling for computer vision applications," in *Proceedings of the European Conference on Computer Vision*, pp. 337–346, 1996.

[20] J. Ferryman and A. Shahrokni, "Pets2009: Dataset and challenge," *Proceedings of the 12th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pp. 1–6, 2009.

[21] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 246–252, 1999.

[22] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 157–164, 1997.

[23] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," in *Handbook of Pattern Recognition and Computer Vision*, pp. 53–82, 2016.

[24] Z. Zivkovic and F. der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[25] Y. Xu, Y. Shi, Y. Fang, and J. Zou, "Edge detection based on convolutional neural networks with multi-scale feature fusion," in *Proceedings of the International Conference on Artificial Intelligence and Big Data*, pp. 102–106, 2018.

[26] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision*, pp. 740–755, 2014.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," *Proceedings of the International Conference on Computer Vision*, pp. 2564–2571, 2011.

[28] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," in *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, 2014.

[29] M. Piccardi, "Background subtraction techniques: A review," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3099–3104, 2004.

[30] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[31] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[32] Y. Wang, W. Yang, Y. Sun, and Y. Chen, "Moving object detection for unconstrained scenes based on generative adversarial and convolutional neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 2233–2245, 2021.

[33] L. Yu, W. Zhang, J. Gao, and W. Fan, "Multi-object tracking by detection and segmentation with enhanced global association," in *Proceedings of the European Conference on Computer Vision*, pp. 20–35, 2020.

[34] P. Hu and D. Ramanan, "Finding tiny faces," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 1522–1530, 2017.

[35] X. Zhou, D. Wang, and P. Krahenbuhl, "Objects as points," in *Proceedings of the European Conference on Computer Vision*, pp. 1–17, 2020.