BTP-1 Final Report

Flying Target Detection and State Estimation

Submitted by:

Mayank Mittal (2022101094)

Under the guidance of:

Prof. Hari Kumar Kandath

1 Problem Statement

The goal of this project is to develop a system that can automatically detect and estimate the state of **Unmanned Aerial Vehicles** i.e. flying drones from images and videos. The system must identify the drone with the bounding box and estimate its state, including velocity, trajectory, and other relevant parameters. This involves using object detection techniques to locate drones and state estimation algorithms to track their movement and predict their behavior over time.

1.1 Objectives

1) Detect and localize drones in real-time images and videos using object detection models (e.g., YOLO v8).

2) Estimate drone states, including velocity and trajectory, by applying feature tracking techniques using Extended Kalman Filters for smoothing the values.

3) Using GRU + CNN hybrid sequential ensemble models, predicting the future path, and conducting error analysis on three types of drone motions - Frog Jump, Upward and Downward motion.

2 Technology Stack

With the following technologies, we can easily perform object detection and state estimation:

- **YOLO (You Only Look Once)**: For real-time object detection (v7,8 and 11)
- **OpenCV**: For image processing and computer vision tasks
- **Python**: Primary programming language
- **Matplotlib, Seaborn, Plotly**: For data visualization
- **TensorFlow/PyTorch**: For implementing deep learning models

Link of Final Codebase

3 Datasets Used

- **Dataset 1:** Drone images dataset from Kaggle used for initial training and pre-processing
  Link to Kaggle dataset **OneDrive link of Pre-processed dataset**
- **Dataset 2:** Drone videos dataset used for drone detection and path prediction
  OneDrive Link of drone videos dataset

4 Complete Pipeline

4.1 Data Preparation

Initial raw drone images were collected from Dataset 1, which already includes annotations.

4.2 Data Preprocessing (Applied only to Dataset 1)

All images were normalized and resized to 640×860, followed by CLAHE for contrast enhancement, gamma correction to balance lighting, and bilateral filtering for edge-preserving smoothing. We further applied standard data-augmentation techniques and extracted frames as needed.

4.3 YOLO v7, v8 and v11 Training

We trained object detection models (YOLO v7, v8 and v11) via transfer learning, monitoring key metrics throughout—mAP reached 0.92, IoU 0.75 and precision 0.88.

4.4 YOLO v8 Detection

Using the YOLO v8 model at a resolution of 640×640 and a confidence threshold of 0.25, we performed real-time drone detection on unseen footage from Dataset 2.

4.5 Extended Kalman Filter

To smooth motion and estimate state, an Extended Kalman Filter tracked each drone's position (x, y), velocity and acceleration, handling the system's non-linear dynamics and logging results to a TXT file.

4.6 Motion Feature Extraction

We partitioned video frames (60% for training, 40% for testing) and extracted characteristic movement patterns—namely "frog jump," "downward," and "upward" trajectories—for subsequent modeling.

4.7 Temporal Modeling with GRU

Feeding sequences of past N frame centers (x, y) into a GRU network, the model learns motion dynamics to predict the next bounding-box position, compensating for occlusions and occasional false negatives.

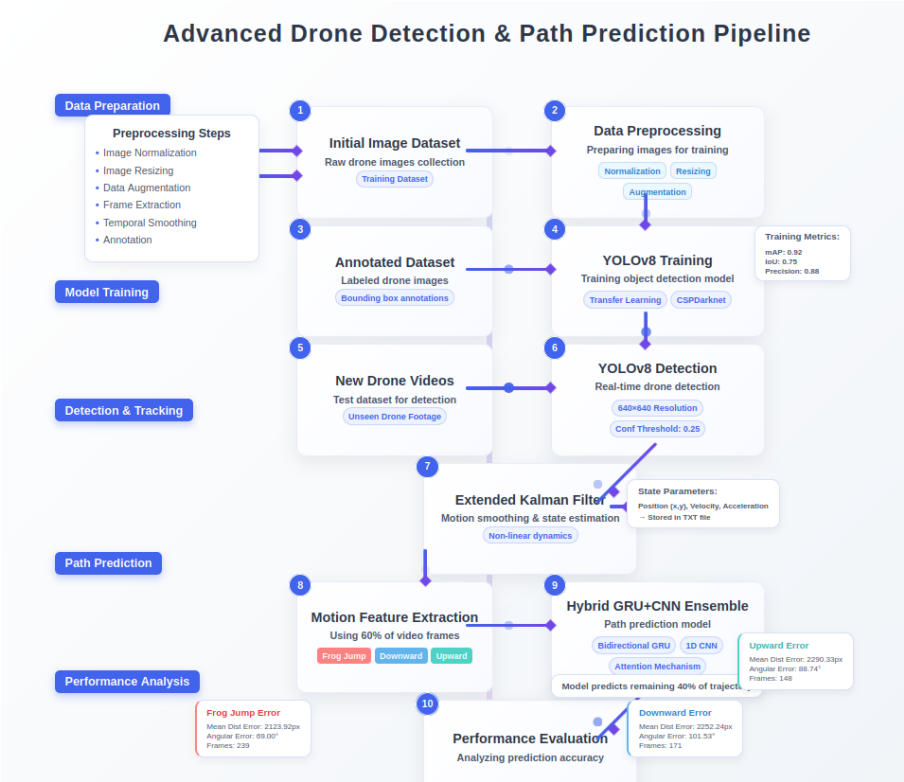
4.8 Hybrid GRU+CNN Ensemble for Path Prediction

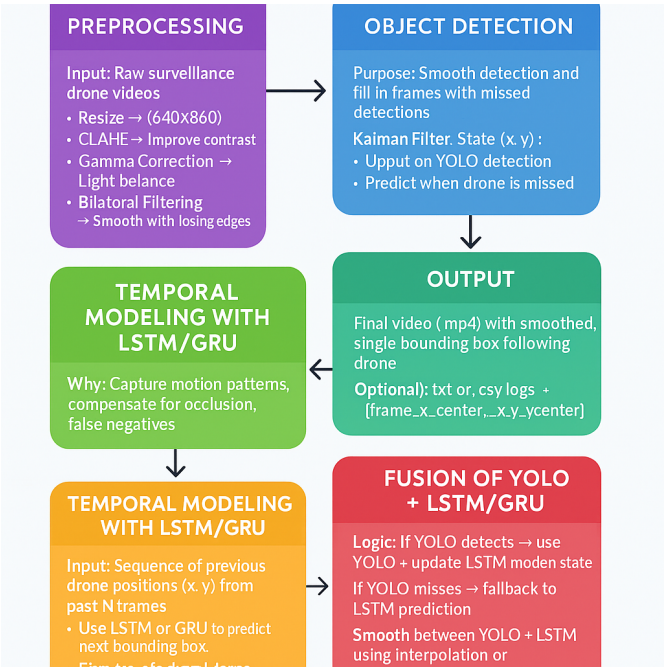
A bidirectional GRU coupled with a 3D CNN and an attention mechanism predicts the remaining 40% of each trajectory. We then performed error-analysis across the three movement types.

4.9 Output

The final deliverables include an MP4 video overlaying smoothed, predicted bounding boxes on drone footage, CSV logs of frame center coordinates [frame x center, frame y center], and saved model weights (.pth) for each movement category.

5 Project Pipeline Flowchart





6 Core Algorithms

6.1 YOLO (You Only Look Once) – Real-Time Object Detection

YOLO is a deep learning-based object detection algorithm that frames detection as a single-step regression task, unlike traditional methods that rely on separate region proposal and classifi-

cation stages, which are slower and less unified. It divides the image into grids, with each cell predicting bounding boxes, confidence scores, and class probabilities in one forward pass.

In this project, YOLO—particularly YOLOv8—offers fast and accurate drone detection in video streams. Its real-time performance, robustness to varied drone shapes, and high accuracy make it ideal for drone surveillance, unlike classical detectors (e.g., HOG + SVM or RCNNs), which are computationally expensive and less suited to dynamic scenes.

6.2 Extended Kalman Filter (EKF) – Tracking and State Estimation

The EKF is a recursive filter designed for systems with non-linear dynamics and noisy observations, unlike the standard Kalman Filter which assumes linearity. EKF linearizes motion models using a Jacobian and predicts the drone's position, velocity, and acceleration even when detections are intermittent.

In this project, EKF smooths noisy YOLO outputs and maintains consistent state tracking across frames. This is crucial for accurate trajectory prediction, which traditional frame-by-frame tracking methods fail to achieve due to their inability to handle noise, uncertainty, and occlusion in real-world drone videos.

6.3 CNN-GRU Hybrid Ensemble – Drone Path Prediction

The CNN-GRU hybrid model combines the spatial understanding of Convolutional Neural Networks with the temporal modeling power of Gated Recurrent Units. While CNNs extract motion features from video frames, GRUs model sequential dependencies to predict future drone positions. Attention mechanisms further enhance performance by focusing on key motion patterns.

This ensemble outperforms traditional models like standalone LSTMs or regression-based predictors, which often struggle with long-term dependencies and spatial variation. In this project, it enables accurate forecasting of the drone's trajectory, even in cases with erratic motion or partial visibility, making it a critical component for robust path prediction.

7 Error Analysis

7.1 Error Analysis for Different Drone Motions

We conducted comprehensive error analyses for three different types of drone motions:

7.1.1 Frog Jump Motion

- Average confidence: 0.733
- Average velocity: 1.96 m/s
- Maximum velocity: 4.66 m/s

7.1.2 Downward Motion

- Average confidence: 0.838
- Average velocity: 1.77 m/s

- Maximum velocity: 3.98 m/s

7.1.3 Upward Motion

- Average confidence: 0.868
- Average velocity: 1.77 m/s
- Maximum velocity: 3.66 m/s

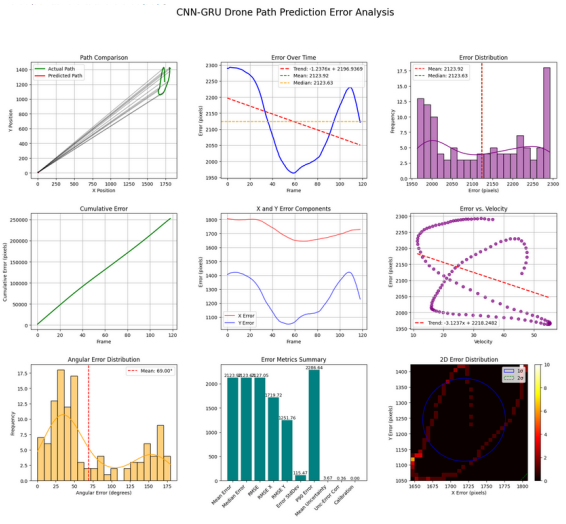


Figure 1: Frog Jump Motion

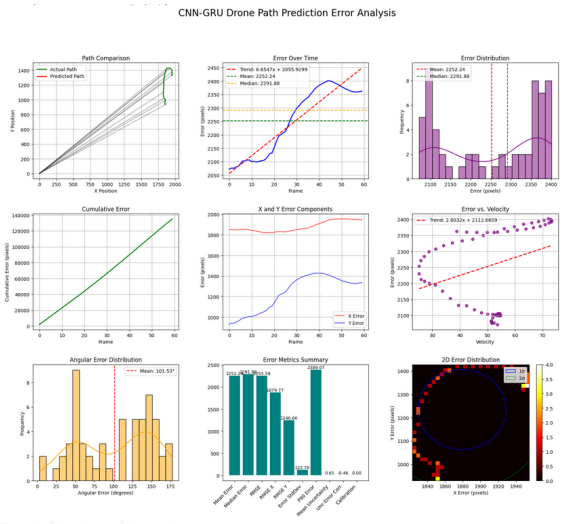


Figure 2: Downward Motion

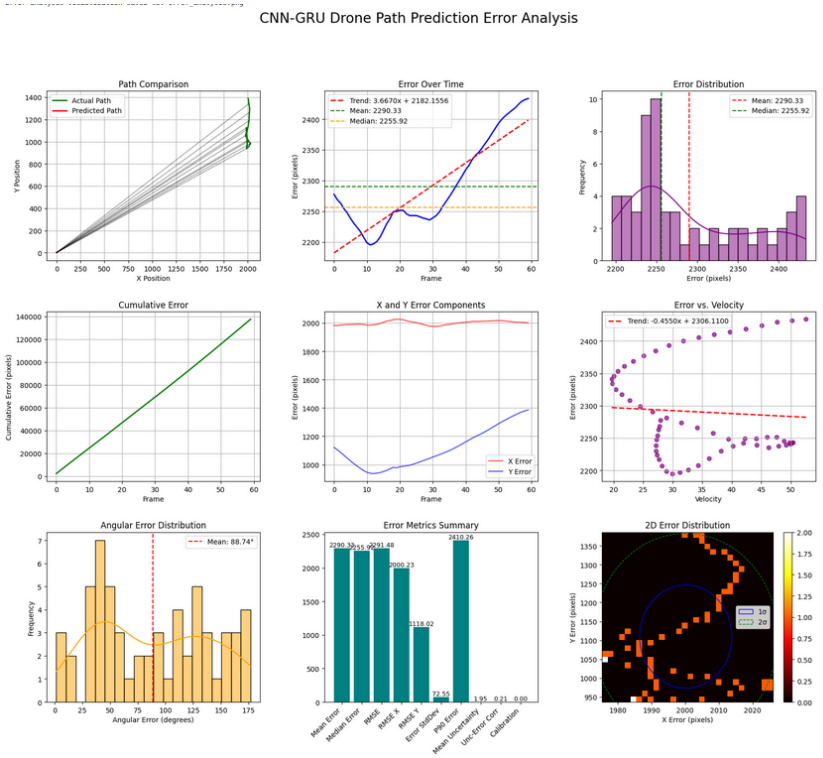


Figure 3: Upward Motion

8 Detailed Model Description

Our architecture integrates object detection, state estimation, and motion prediction components for robust drone tracking.

8.1 Object Detection

We use YOLOv8 for real-time drone detection due to its optimal speed-accuracy tradeoff compared to YOLOv7 and YOLOv11.

Configuration:

- Backbone: CSPDarknet; Resolution: 640×860
- Confidence threshold: 0.25; NMS IoU: 0.45
- Anchor-free architecture

8.2 State Estimation (EKF)

An Extended Kalman Filter smooths YOLO outputs and estimates position, velocity, and acceleration using a constant acceleration motion model.

State Vector: $(x, y, v_x, v_y, a_x, a_y)$

Measurement: YOLO-predicted (x, y)

EKF handles noise and non-linear drone motion for consistent tracking, even with temporary detection loss.

8.3 Path Prediction (CNN-GRU Hybrid)

We use a CNN-GRU ensemble to predict future drone positions:

- **Input:** Sequence of past 20 positions
- **Network:** 1D CNN + Bidirectional GRU + Attention
- **Output:** Next 5-frame trajectory

Training: MSE + angular loss, Adam optimizer, curriculum learning from simple to complex motion types.

9 Final Results and Resources

The following resources are available in the link **Link to Final Codebase** , providing a complete overview of our final implementation and results:

- **Model Files:** Three `.pth` files, each trained specifically for one of the drone motion patterns: `up`, `down`, and `frogjump`.

- **Final Output Videos:**
  - `up.mp4`, `down.mp4`, and `frogjump.mp4` — predicted drone trajectories over time.
  - A separate `YOLO_Kalman_detection.mp4` video visualizes red (YOLO) and green (Kalman) bounding boxes for comparative tracking.
  - Cut-short versions of each motion video with trimmed durations (seconds mentioned in filenames).
- **CSV Output Files:** Each video has a corresponding `.csv` file containing:

  `frame_id, x, y, ekf_vx, ekf_vy, ekf_ax, ekf_ay, confidence, detected, track_id, height_meters, smoothed_vx, smoothed_vy, smoothed_velocity, smoothed_direction_angle, smoothed_direction, velocity_mps`

  These files capture detection and state estimation data per frame.
- **YOLO Model:** The `YOLOv8 .pt` file used for object detection.
- **Initial Training Results:** The `Initial_training/` folder contains:
  - `YOLOv7` and `v11` model outputs and predictions used in the early phase.
- **Code:**
  - Full code in both `.ipynb` and `.py` formats.
  - In the notebook, refer directly to the section titled `Final Code` for the main implementation.
- **Presentation:** A `presentation.pdf` summarizing the whole project.

10 Setbacks and Challenges

10.1 Setbacks

10.1.1 YOLO-Based Detection

Initially, we applied previously trained YOLOv7, YOLOv8, and YOLOv11 models on drone videos, experimenting with bounding box fusion. However, fusion degraded performance, and further testing revealed that YOLOv8 alone provided the most reliable and accurate drone detection.

10.1.2 Handling Flapping/Bird-Shaped Drones

Some videos contained flapping or bird-shaped drones, which our models struggled to detect due to lack of relevant training data. Since suitable datasets for such drones are unavailable online, we attempted detection using the 'bird' class as a workaround, but with limited success. Future work includes collecting custom data and improving detection for these cases.

10.1.3 Limited Path Tracking Coverage

The current path prediction system has been evaluated on a subset of available videos and covers only a few predefined drone motion types (e.g., frog jump, upward, downward). However, real-world drones may exhibit a wider range of motion patterns. In future work, we plan to scale the system to handle more diverse trajectories across a larger set of drone videos.

10.2 Drone Path Prediction: Challenges

10.2.1 Current Challenges

1. **Camera Motion Issues** - When the camera moves while tracking the drone, it creates shifting reference frames and perspective distortion
2. **Angular Velocity Complexity** - Rotational movements create non-linear trajectories that are difficult to predict accurately
3. **Error Accumulation** - Small prediction errors compound over time, especially during complex maneuvers

11 Future Work

- Incorporate richer motion features like angular acceleration, jerk, curvature metrics, and quaternion-based orientation.
- Explore advanced architectures such as ViT-based Transformers and BERT-style models to replace or complement GRU for better temporal understanding.
- Integrate Physics-Informed Neural Networks and add skip connections to improve modeling of complex drone dynamics.
- Extend training beyond frame-wise tracking to video-wise generalization: train on 70% of videos and evaluate on the remaining 30% for each drone motion type.

12 References

1. YOLOv7 Drone Detection GitHub Repository:
https://github.com/doguilmak/Drone-Detection-YOLOv7
2. YOLOv8 Drone Detection GitHub Repository:
https://github.com/doguilmak/Drone-Detection-YOLOv8x
3. Codebasics YouTube Tutorial on YOLO Algorithm:
https://youtu.be/ag3DLKsl2vk
4. YouTube Guide on Training Custom YOLOv5, YOLOv8, and YOLOv11 Models:
https://youtu.be/r0RspiLG260
5. Research Paper: *Drone Detection from Video Streams Using Image Processing Techniques and YOLOv7* (IJIGSP):
https://www.mecs-press.org/ijigsp/ijigsp-v16-n2/IJIGSP-V16-N2-7.pdf
6. Research Paper: *Application of Image Processing Techniques for UAV Detection Using Deep Learning and Distance-Wise Analysis* (MDPI Drones Journal):
https://www.mdpi.com/2504-446X/7/3/174
7. UAV Drone Object Tracking using Kalman Filter
https://github.com/yudhisteer/UAV-Drone-Object-Tracking-using-Kalman-Filter