BTP-1 Final Report

FYLING TARGET DETECTION AND STATE ESTIMATION

PRESENTER NAME

Mayank Mittal (2022101094)
Adithi Samudrala(2022102065)

PROBLEM STATEMENT

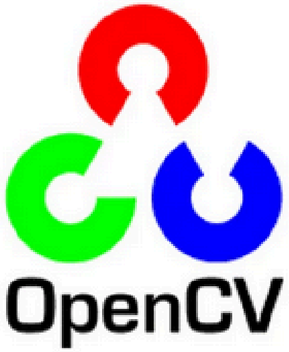
The goal of this project is to develop a system that can automatically detect, classify, and estimate the state of flying drones from images and videos. The system must identify the type of drone from three predefined categories and estimate its state, including velocity, trajectory, and other relevant parameters. This involves using object detection techniques to locate drones, classification models to identify their type, and state estimation algorithms to track their movement and predict their behavior over time.

Objectives

1) Detect and localize drones in real-time images and videos using object detection models (e.g., YOLO v8).
2) Classify drones into one of three types using deep learning-based classification models.
3) Estimate drone states, including velocity and trajectory, by applying optical flow or feature tracking techniques using Kalman Filters for smoothing the values.
4) Using LSTM/GRU sequential model , predicting the future path and Error Analysis.

TECH STACK USED

WITH ALL THESE TECHNOLOGIES ,WE CAN EASILY DO OBJECT DETECTION, CLASSIFICATION AND STATE ESTIMATION



PREPROCESSING TECHNIQUES OVERVIEW

Essential Steps for Effective Data Preparation

01 IMAGE NORMALIZATION

Normalization is an essential first step in image preprocessing, where pixel values are scaled to a common range, typically between 0 and 1. This process helps in reducing biases in model training and improves convergence speed, facilitating better performance in image classification tasks.

02 IMAGE RESIZING

Resizing images to a fixed dimension is crucial for ensuring uniformity across the dataset. This technique prevents computational inefficiencies and ensures that all images are processed uniformly, which is especially important when feeding images into neural networks that require consistent input sizes.

03 DATA AUGMENTATION

Data augmentation techniques, including rotation, scaling, and flipping, are applied to enhance the diversity of training data. By artificially expanding the dataset, models can learn to generalize better, thus improving their robustness and performance when encountering unseen data.

04 FRAME EXTRACTION FROM VIDEOS

In video processing, extracting frames is a vital step that allows for the analysis of individual image data over time. This enables models to understand motion and context, which is crucial for tasks such as action recognition and object tracking within video sequences.

05 TEMPORAL SMOOTHING

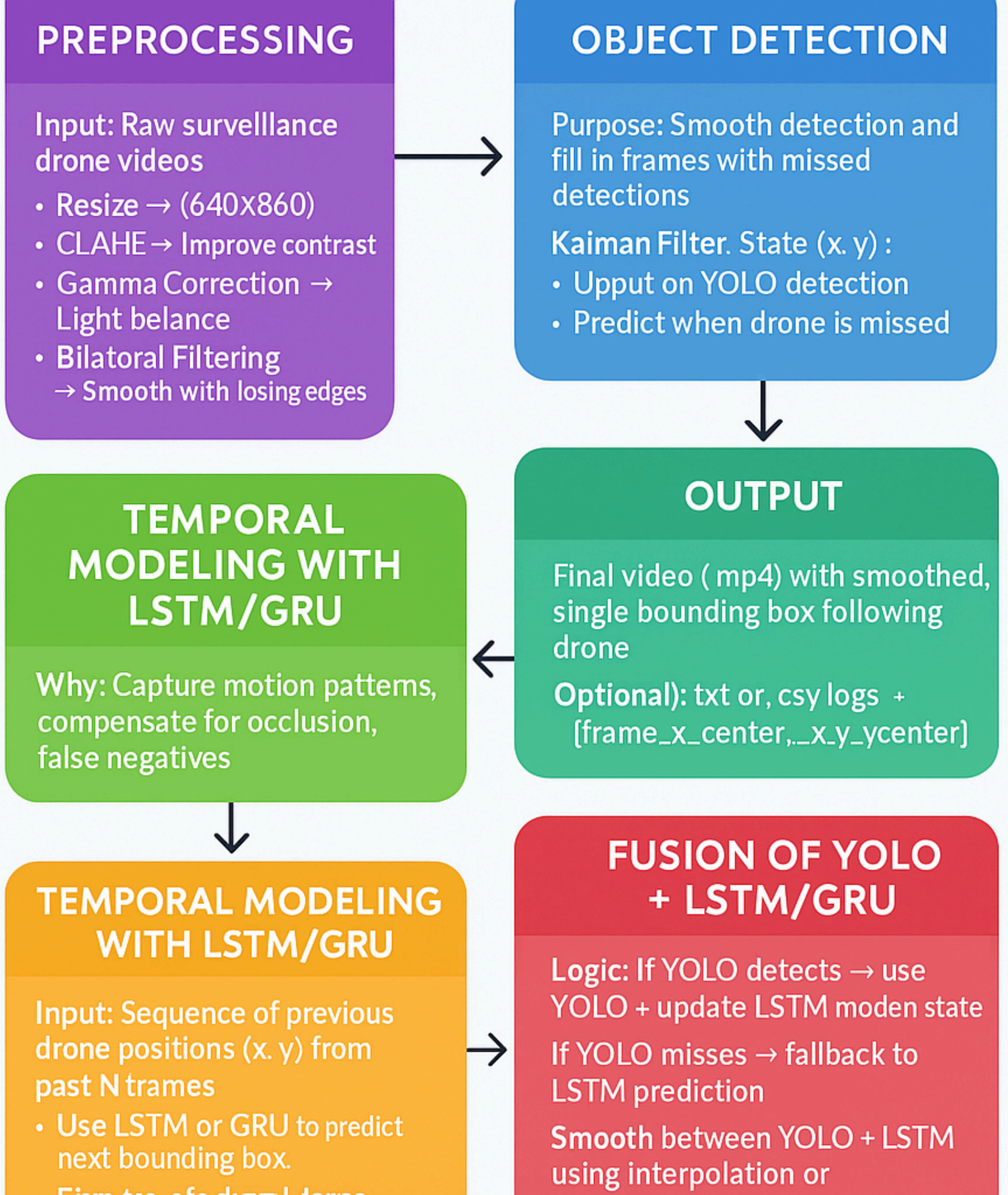
Temporal smoothing is applied to video data to reduce noise and fluctuations between frames. This technique enhances the quality of the data, allowing models to focus on significant changes and patterns, which is essential for accurate object detection and classification.

06 ANNOTATION FOR OBJECT DETECTION

Annotating video frames with bounding boxes and labels is crucial for supervised learning in object detection tasks. This process involves marking the location and identity of objects within frames, providing the necessary information for training models to recognize and classify objects accurately.

Project Pipeline

Using:-
1) YOLO v8 for drone detection
2) Kalman Filters for smoothing the detection
3) LSTM/GRU models with CNN and attention mechanism to do path prediction
4) Error Analysis



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

What is YOLO?

YOLO is a deep learning-based object detection algorithm known for its speed and accuracy. Unlike traditional detectors that use a two-step process (region proposal + classification), YOLO treats object detection as a **single regression problem**.

How YOLO Works:

- The input image is divided into an `S x S` grid.
- Each grid cell predicts:
  - **Bounding boxes** (coordinates + width/height),
  - **Confidence score** (how likely the box contains an object),
  - **Class probabilities**.
- YOLO applies a single neural network to the entire image and simultaneously outputs bounding boxes and class scores for those boxes.

Algorithm Steps:

1. **Feature extraction** using a CNN (e.g., CSPDarknet in YOLOv8).
2. **Bounding box prediction** (center_x, center_y, width, height).
3. **Objectness score**: Confidence whether an object exists in the box.
4. **Non-Maximum Suppression (NMS)** to remove duplicate detections.

Why YOLO is Useful in This Project:

- Provides **real-time detection** of drones in video frames.
- YOLOv8 performs best among YOLOv7 and v11 in your case.
- Highly accurate on well-defined drone shapes.

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

What is EKF?

EKF is a **recursive estimator** used to track an object's state (position, velocity, acceleration) over time when the model involves **non-linear dynamics** and noisy measurements. It is an extension of the classic Kalman Filter for non-linear systems.

Why EKF Instead of Kalman Filter?

Kalman Filter assumes linear models. But in real-world applications like drone motion (which is often non-linear), EKF linearizes the model at each step using **Taylor expansion (Jacobian matrix)**.

EKF Algorithm Overview:

1. **Prediction Step:**
   - Predict next state using a non-linear motion model:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1}, u_k)$$

$$P_{k|k-1} = F_k P_{k-1} F_k^T + Q_k$$

   - Where `f` is the motion model, `F_k` is the Jacobian of `f`, and `Q_k` is the process noise.
2. **Update Step:**
   - Compute Kalman Gain and correct the state using observations:

$$y_k = z_k - h(\hat{x}_{k|k-1})$$

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$$

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k y_k$$

$$P_k = (I - K_k H_k) P_{k|k-1}$$

   - `h` is the non-linear observation model, `H_k` is its Jacobian, and `R_k` is the measurement noise.

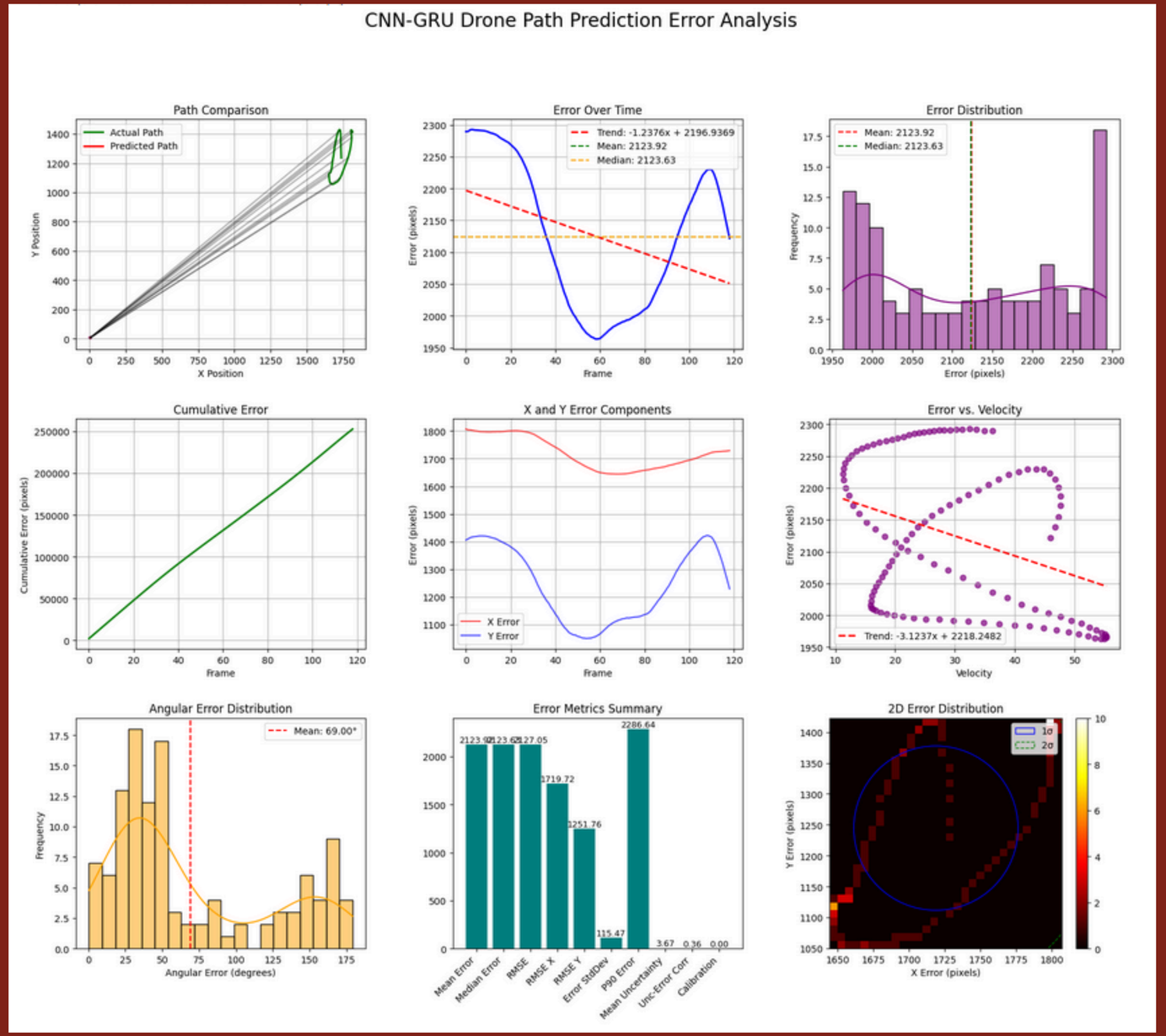
Why EKF is Useful in This Project:

- Smooths out YOLO's noisy frame-by-frame predictions.
- Provides continuous estimates of drone **position, velocity, and acceleration**.
- Enables **trajectory forecasting**, even when detections are briefly lost.

Error Analysis for video in which drone doing frog jump motion

Average confidence: 0.733
Frames with detections: 239
Average velocity: 1.96 m/s
Maximum velocity: 4.66 m/s

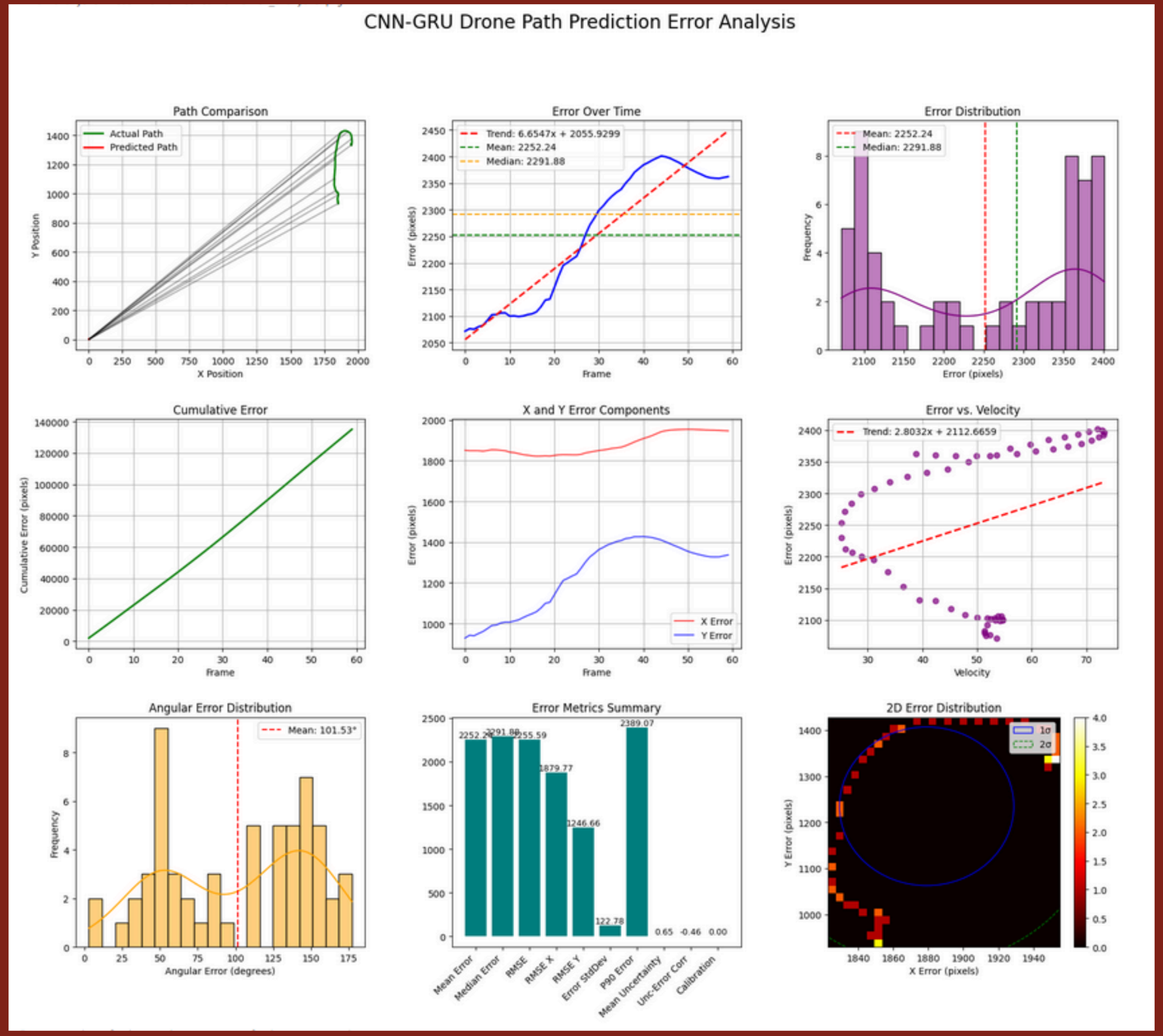
===== PREDICTION ERROR METRICS =====
Mean Distance Error: 2123.92 pixels
Median Distance Error: 2123.63 pixels
Maximum Distance Error: 2292.56 pixels
RMSE: 2127.05 pixels
Error Growth Rate: -1.2376 pixels/frame
Normalized Error: 246.6473
Mean Angular Error: 69.00 degrees
Uncertainty-Error Correlation: 0.3588
Calibration Score: 0.0000
==================================



Error analysis for drone showing downward motion

Average confidence: 0.838
Frames with detections: 171
Average velocity: 1.77 m/s
Maximum velocity: 3.98 m/s

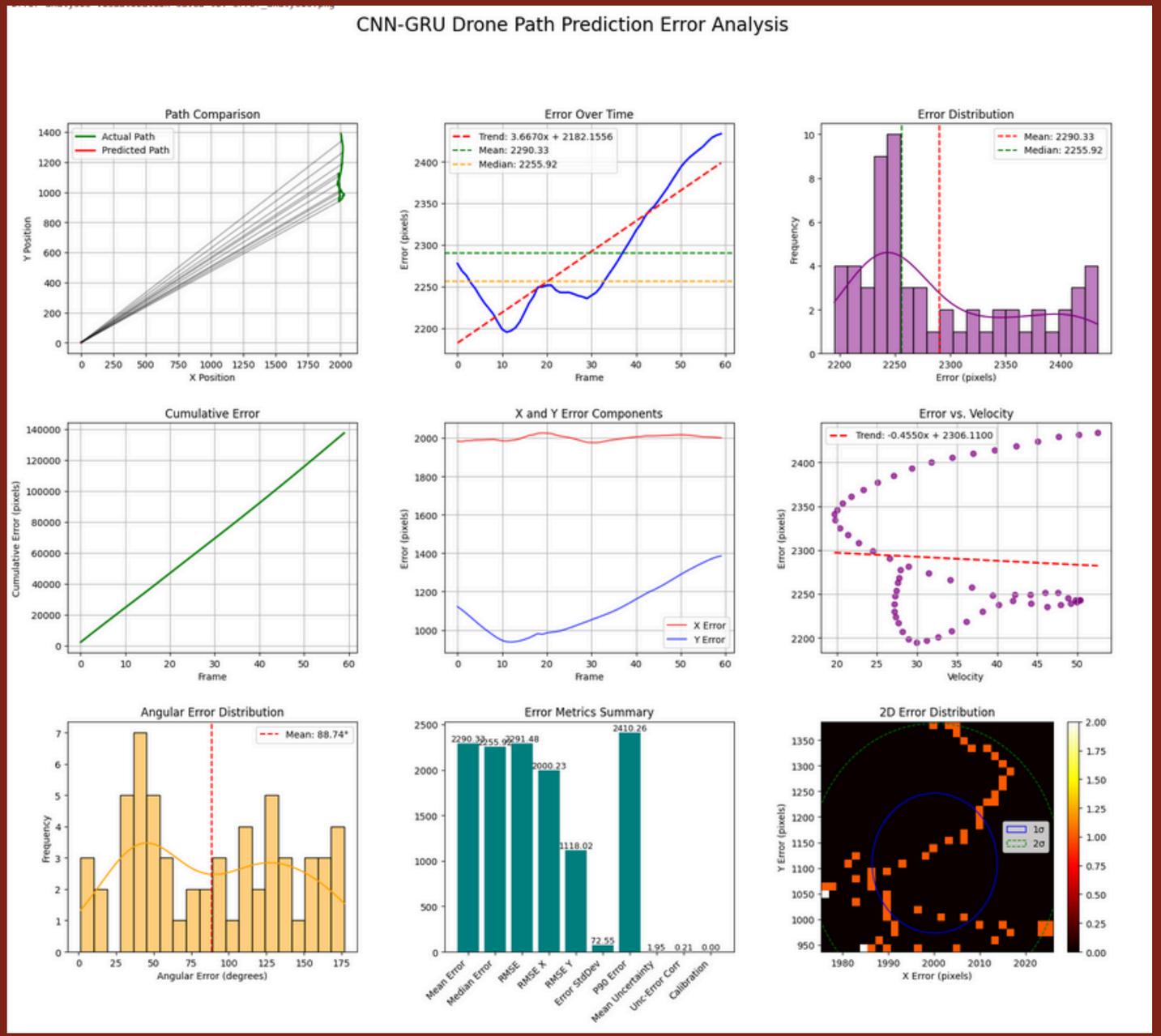
===== PREDICTION ERROR METRICS =====
Mean Distance Error: 2252.24 pixels
Median Distance Error: 2291.88 pixels
Maximum Distance Error: 2401.23 pixels
RMSE: 2255.59 pixels
Error Growth Rate: 6.6547 pixels/frame
Normalized Error: 196.8719
Mean Angular Error: 101.53 degrees
Uncertainty-Error Correlation: -0.4578
Calibration Score: 0.0000
==================================



Error analysis for drone showing upward motion

Average confidence: 0.868
Frames with detections: 148
Average velocity: 1.77 m/s
Maximum velocity: 3.66 m/s

===== PREDICTION ERROR METRICS =====
Mean Distance Error: 2290.33 pixels
Median Distance Error: 2255.92 pixels
Maximum Distance Error: 2433.58 pixels
RMSE: 2291.48 pixels
Error Growth Rate: 3.6670 pixels/frame
Normalized Error: 199.8974
Mean Angular Error: 88.74 degrees
Uncertainty-Error Correlation: 0.2054
Calibration Score: 0.0000
==================================



Some Setbacks

01

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.

02

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.

Drone Path Prediction: Challenges & Improvements

Current Challenges

1. Camera Motion Issues - When your 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

Quick Improvements

- Better Features :- Add explicit angular acceleration and jerk features
- Use quaternion representation for orientation instead of angles
- Calculate curvature metrics for the flight path
- Enhanced Architecture:- Implement a Physics-Informed Neural Network layer
- Use a longer sequence length to capture complex maneuvers
- Add skip connections between temporal layers
- Training Optimizations:- Use weighted loss functions that emphasize physical plausibility
- Start with simple trajectories before complex ones (curriculum learning)
- Apply stronger regularization to prevent overfitting

Thank You