AssistX Enterprise – AI Engineer (Computer Vision) Assessment

1. Introduction

The purpose of this system is to automatically detect two Beyblades in a top-down battle video, measure the spinning duration of each Beyblade, and output the results to a structured CSV file for further analysis. By analyzing motion using image blur as an indicator, the system can determine when a Beyblade stops spinning, which means the end of the battle. Once the battle ends, the system identifies the winner, loser, winner spin duration, loser spin duration, and the difference duration between both, then stores the results in a structured CSV file. This data can then be used by analysts to evaluate Beyblade performance.

2. System Overview

The system is designed to analyze Beyblade battle videos and extract meaningful data by following a sequence of well-defined processing steps. Below is a textual overview of the system workflow:

1. **Video Input**
   The user provides a video file of 1v1 Beyblade with top-down view. This video serves as the main input for the system.
2. **Object Detection and Object Tracker**
   Each frame of the video is processed using a trained object detection mode to identify and localize the two Beyblades. The model outputs bounding boxes and unique track IDs for each detected object. Besides object detection, this system also use object tracker to assign and maintain consistent IDs for each detected Beyblade.
3. **Spin Analysis**
   After detection, the system extracts cropped regions of each Beyblade from the frame. A blur-based metric is applied to these crops to estimate whether each Beyblade is still spinning. A predefined threshold determines when the Beyblade has stopped.
4. **Game End Detection**
   The game is considered over when one of the Beyblades stops spinning. At that moment, the system records the time and identifies the winner and loser.
5. **Data Aggregation**
   When the battle ends, relevant statistics such as total spinning duration of each Beyblade, and the difference in spin time is calculated.
6. **CSV Output**
   All extracted data is saved to a CSV file. This includes:
   - Winner and loser ID.
   - Spinning duration of both Beyblades.
   - Total battle duration.
   - Duration difference.

3. Detection Model and Object Tracker

3.1 Model Used (1)

For the detection task, I decided to use YOLOv8s (You Only Look Once version 8) developed by Ultralytics as the object detection model. YOLOv8s is good balance between speed, accuracy, and ease of integration for computer vision tasks.

The main reasons for selecting YOLOv8s are:

- **Real-time Performance**: The model is lightweight and fast, making it suitable for video processing with minimal delay.
- **High Accuracy**: YOLOv8s achieves high mean Average Precision (mAP) across multiple datasets, which helps ensure reliable detection of fast-moving Beyblades.
- **Robust Pretrained Weights**: YOLOv8s comes with pretrained weights on various datasets, which helps with transfer learning and fine-tuning on a smaller, custom Beyblade dataset.
- **Ease of Use**: The Ultralytics framework provides a simple interface for training, testing, and deployment, reducing development time.

3.2 Object Tracker

For the tracking task, I chose ByteTrack, which is natively supported by the Ultralytics YOLO framework. In addition to being lightweight, ByteTrack offers reliable tracking performance and seamless integration with YOLOv8 detection outputs.

3.3 Dataset

To train the model, I collected some datasets from open-source repositories such as Roboflow. These datasets were then combined and preprocessed to form a unified training set tailored for Beyblade detection. The dataset contains only one object class (“Beyblade”).

Dataset Detail:

- Annotation tool = Roboflow
- Number of Image = 359 images
- Augmentation = Rotation, Brightness, Blur, Noise
- Split = 84% train, 10% validation, 5% Test

3.4 Training Process

- Base model = YOLOv8s
- Epoch = 30
- Batch size = 16
- Image size = 640 x 640
- Results:
    - mAP50 = 0.99201
    - mAP50-95 = 0.89074

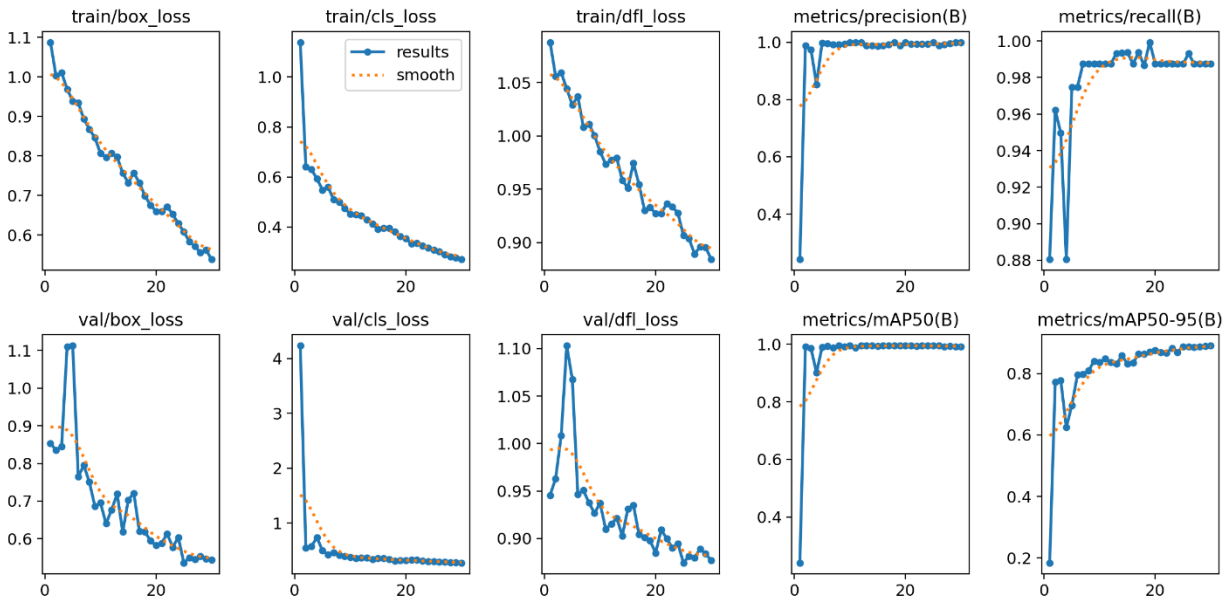

Figure 1 Train Curves

4. System Performance (2)

4.1 Detection Performance

After training the YOLOv8 model on the custom dataset, I evaluated its performance using standard object detection metrics on the validation set.

Key performance results:

- mAP50 = 0.99201
- mAP50-95 = 0.89074
- Precision = 1
- Recall = 0.9874

In battle video tests, the model maintained stable detections across consecutive frames providing reliable bounding boxes

4.2 Tracker Performance

The system uses **ByteTrack** to maintain consistent object identities across video frames. In qualitative evaluation, the tracker performed well under ideal conditions, with no sudden object jumps or identity switches, and successfully preserved consistent track_ids for both Beyblades throughout the video.

5. Spin / Motion Detection Logic

After detecting and tracking the Beyblades, the system estimates whether each Beyblade is spinning by analyzing the motion blur present in its cropped image region across video frames.

5.1 Crop Extraction per Object

Using the track_id assigned by ByteTrack, each detected object is cropped from the original video frame:

- The bounding box (bbox) is used to extract a region of interest (RoI).
- This crop is saved per track_id for further analysis.
- By relying on tracking, the system ensures crops correspond to the same Beyblade over time.

5.2 Blur-based Spin Detection

To determine if a Beyblade is spinning, the system computes the blur score of each crop using the variance of Laplacian method.

- If the blur score is low, the object is likely spinning fast (blurred).
- If the blur score is high, the object is likely stationary (sharp image).

The output from this process is used to determine the state of each Beyblade (Spinning or Still) based on a predefined motion threshold. However, since the raw blur-based detection can be unstable due to noise and rapid motion changes, a filtering mechanism is applied to stabilize the results.

5.3 Spin Duration Tracking

The system continuously monitors the spin state of each Beyblade over time. At each frame, it compares the current state with the previous one.

- If the state changes from False to True, the system records the timestamp as the **start time**.
- If the state changes from True to False, the system records the timestamp as the **end time**.

The spin duration is then calculated by subtracting the end time and start time. This logic ensures that only continuous spinning periods are counted.

5.4 Summary

Feature	**Method Used**
Object Identity Tracking	ByteTrack
Spin Detection	Variance of Laplacian (Blur Score)
Spin Status Decision	Threshold on blur score
Final Output	CSV with spin duration per Beyblade

6. Output Data

After completing the detection, tracking, and spin duration analysis, the system generates a CSV file as the final output. This file contains key information about the battle result between two Beyblades.

6.1 CSV Ouput Format

The CSV file contains one row per battle, with the following columns:

Column	Description
win_id	The track_id of the Beyblade that spin the longest (winner)
lose_id	The track_id of the Beyblade with the shorter spin duration (loser)
win_dur	Total spin duration (in seconds) of the winning Beyblade
lose_dur	Total spin duration (in seconds) of the losing Beyblade
battle_dur	The total duration of the battle, marked by the moment the first Beyblade stops
dur_difference	The difference in spin duration between winner and loser

Example output:

```
win_id,lose_id,win_dur,lose_dur,battle_dur,dur_difference
2,1,25.53,25.23,25.23,0.30
```

6.2 How the Output Generated (3)

- The system continuously monitors the spin state of each track_id.
- When a Beyblade starts spinning, the system records the start time.
- When the same Beyblade stops, the end time is recorded.
- The difference between the end time and the start time gives the total spin duration for each Beyblade.
- The battle data is stored in a dictionary and sorted in descending order based on spin duration. The Beyblade at the first index is considered the winner, and the second is the loser.
- The result is then written to a CSV file based on the sorted dictionary.

7. Challenges and Improvement

- **Challenges:** Limited availability of clean top-down videos for testing.
- **Improvement:**
    - Add detection data from multiple viewpoints.
    - Improve tracking stability under fast motion or jumps.