**MacV Object Tracker Task**

**Task Overview**

The objective of this task was to build a Python-based object tracking system capable of:

1. Detecting objects in a video using the YOLOv8n model.
2. Assigning unique IDs to objects using the DeepSort algorithm.
3. Drawing centroids and trails for each tracked object.
4. Calculating the time each object spends in the frame, specifically for individual subobjects (e.g., person1, handbag1, etc.).
5. Exporting the final statistics and displaying them in an HTML file alongside the output video.

**Initial Approach and Challenges**

**Why YOLOv8n?**

I chose YOLOv8n for object detection because it is one of the most efficient models available for this task. Its ease of use and performance gave me confidence that it would handle the object detection requirement effectively.

**Centroid and Trail Implementation**

The next step was to implement centroid tracking and draw trail lines for each object. Initially, this was challenging, but with the help of tools like Claude and ChatGPT, I resolved the implementation efficiently. The trails visually represent the movement path of each object.

**Time Calculation**

Calculating the duration of each object’s presence in the frame was the most critical part of the project. While it is straightforward to calculate overall durations, achieving this granularity for each unique object (like person1, handbag2) required deeper research. I referred to GitHub repositories, YouTube tutorials, and used AI tools (Copilot, Claude, ChatGPT) for guidance.

The solution came with the DeepSort algorithm. This algorithm assigns unique IDs to each detected object, enabling precise tracking. With unique IDs, I calculated the time by measuring the object's presence across frames (accounting for the 30 FPS video frame rate).

**Implementation Details**

**Object Detection**

The YOLOv8n model was used for object detection:

* Loaded using the ultralytics library.
* Configured to detect objects above a confidence threshold of 0.5.

**Object Tracking with DeepSort**

The DeepSort tracker was configured with:

* max\_age = 50: Maximum number of frames before an object is considered lost.
* n\_init = 3: Minimum detections required to initialize a track.
* embedder\_gpu = True: Enabled GPU acceleration for faster processing.

**Centroid and Trail Tracking**

* Centroids were calculated for each bounding box.
* Trails of objects were stored and updated using a TrailTracker class. This class uses a color-coding system to differentiate between object trails visually.

**Time Calculation**

* Each object’s presence time was calculated based on its first and last appearances in the frame.
* The time was converted from frame counts to seconds and milliseconds using the 30 FPS rate.

**Final Output**

* Statistics for each unique object (class name, ID, time in the frame) were displayed on the video.
* The final output video was saved as output.mp4.

**HTML Display**

To enhance the usability and presentation of results, I created an HTML file that:

* Displays the time and count of objects in a table format.
* Includes the final processed video embedded for easy visualization.