Tasks for Object Tracking

Objective

Develop hands-on skills in object tracking by implementing, tuning, and testing various algorithms. Complete each task step-by-step and observe how different scenarios affect tracking performance.

---

1. Basic Tracking in OpenCV

Description:
Use OpenCV's built-in tracking algorithms to track a single object in a video.

- **Steps**:
    1. Load a short video clip (you can use MOT17 or any custom video).
    2. Implement a basic tracking method using cv2.TrackerCSRT_create() or cv2.TrackerKCF_create().
    3. Track a single object throughout the video.
    4. Draw a bounding box around the tracked object in each frame and save the output video.
- **Goal**:
  Understand how simple tracking algorithms work in OpenCV.

---

2. Multiple Object Tracking with a Dataset

Description:
Practice multi-object tracking with real-world data to manage multiple objects and assign unique IDs.

- **Steps**:
    1. Choose a dataset, such as MOT17 or UA-DETRAC.
    2. Select a sequence with multiple objects (e.g., pedestrians or vehicles).
    3. Implement a multiple-object tracking algorithm (e.g., SORT or Deep SORT).
    4. Track and label each object with a unique ID, and save the output video with IDs visible.

- **Goal**:
  Gain experience in tracking multiple objects and maintaining identity across frames.

---

3. Occlusion Handling

Description:
Explore the challenge of tracking an object through occlusions.

- **Steps**:
  1. Use the same dataset, choosing a sequence with frequent occlusions (e.g., crossing objects).
  2. Track an object that goes behind other objects.
  3. Adjust algorithm settings or add Kalman filtering to help maintain tracking accuracy.
- **Goal**:
  Learn to manage occlusions, one of the biggest challenges in object tracking.

---

4. Tracking Under Different Lighting Conditions

Description:
Evaluate how lighting changes impact tracking performance.

- **Steps**:
  1. Select or create a video with varying lighting (e.g., moving from sunlight to shadow).
  2. Test two tracking algorithms, such as KCF and CSRT, on this video.
  3. Log observations on how each algorithm adapts to the lighting changes.
- **Goal**:
  Observe how lighting affects tracking accuracy and learn how different algorithms handle it.

---

5. Performance Tuning

Description:
Experiment with tuning algorithm parameters to improve tracking accuracy and speed.

- **Steps**:
  1. Select an algorithm (e.g., SORT or Deep SORT).
  2. Experiment with various parameters, like confidence thresholds or bounding box update intervals.
  3. Track the impact of each adjustment on tracking performance and document findings in a table.
- **Goal**:
  Practice parameter tuning and understand how it impacts algorithm performance.

---

6. Real-Time Tracking

Description:
Implement object tracking with a live video feed to simulate real-time conditions.

- **Steps**:
  1. Use a webcam or live video feed as the input source.
  2. Implement a fast tracking algorithm (e.g., KCF or CSRT).
  3. Track a moving object in real time (e.g., a hand or a face) and display a bounding box.
- **Goal**:
  Experience the unique challenges of tracking in a real-time environment.

---

7. Evaluation and Error Analysis

Description:
Analyze tracking accuracy by comparing predicted bounding boxes with ground truth.

- **Steps**:
  1. Track an object in a video, then manually label the ground truth bounding box for each frame (or use a labeled dataset).

2. Calculate the Intersection over Union (IoU) between the predicted and ground truth bounding boxes for each frame.
3. Determine the average IoU and identify frames where tracking accuracy decreased.

- **Goal**:
  Develop skills in evaluating and analyzing tracking accuracy.

Extra Credit (Optional)

1. **Algorithm Comparison**
   - Implement two different tracking algorithms on the same video sequence.
   - Compare their accuracy, speed, and robustness.
2. **Advanced Model**
   - If you have access to a GPU, implement a deep learning-based algorithm like YOLO with Deep SORT.
   - Track objects in a complex video and compare its performance to simpler algorithms.