ECSE 415 - Introduction to Computer Vision
Assignment 5: Video Analysis

DEADLINE: Wednesday, 3 December 2025, 11:59 PM

Please submit your assignment solutions electronically via the **myCourses** assignment dropbox. The submission should include a **single** Jupyter notebook (**.ipynb** file) and **output videos** as separate **.mp4** files. More details on the submission format can be found below. Submissions that do not follow the format will be penalized by 10%. Attempt all parts of this assignment. The assignment will be graded from a total of **100 points**. You can use **PyTorch**, **TenserFlow**, **OpenCV**, **Matplotlib**, **Scikit-Image**, and **Numpy** library functions for all parts of the assignment. Students are expected to write their own code. You may use any tools for your assignments, but you are responsible for any misrepresentation, inaccuracy, or plagiarism. Citations of non-existent material or obvious factual inaccuracies may result in no credit. (Academic integrity guidelines can be found **here**). Assignments that are handed in late without permission will receive a penalty of 10% per day.

Note: Members within each group will receive the same score for the assignment.

Submission Instructions

On myCourses:

- Submit a single **Jupyter notebook** consisting of the solution of the entire assignment and **output videos** as separate .mp4 files.
- Comment your code appropriately.
- Give references for all codes which are not written by you. (Ex. the code is taken from an online source or from tutorials)
- Do not forget to run **Markdown** ('Text') cells.
- Do not submit input/output images. The output images should be displayed in the Jupyter Notebook itself.
- Make sure that the submitted code runs without errors. Clearly describe any specific requirements for executing the code in your notebook.
- If you use external libraries in your code, please specify their names and versions within the notebook.

On Kaggle:

- Submit the predicted count results as a **.csv** file to the competition.

Preliminaries

1. **Group Enrollment:** Enroll in a group of up to 2 people on myCourses to access the assignment files.
2. **Join the Kaggle Competition:** Join the competition through this link. (If you don't have a Kaggle account, you will need to create one first.)
3. **Team Naming:** Use your myCourses group number as your Kaggle team name (e.g., *Group 3*).

Objective

The goal of this assignment is to implement and evaluate an object tracking pipeline on pedestrian video sequences, thereby gaining practical experience in combining YOLOv8 with DeepSORT. The assignment involves assessing tracking performance using MOTA and performing per-frame pedestrian counting.

Dataset Description

Download the dataset from the Kaggle competition. The structure should be:

```
Tracking/
 Task1/
    images/    # 429 images
    gt/        # Ground truth annotations (gt.txt)
 Task2/
    images/    # 1050 images
```

Use only the first 6 columns from the "Task1/gt/gt.txt" file:

```
<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>,
<class>, <visibility>
```

1 Data Preparation (10 points)

1. Convert the images in the "Task1/images/" directory into a video at **14 FPS**. You can use the `cv2.VideoWriter()` function from OpenCV.
2. Save the output video as `task1_input.mp4`.

2 Model Implementation (40 points)

1. Apply **DeepSORT with YOLOv8** to detect and track pedestrians in the video.
2. Draw a bounding box for each detected pedestrian and assign a unique **tracking ID**.
3. Save the output video as `task1.mp4`.
4. Save tracking results as a .txt file in the following format:

```
<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>
```

3 Model Evaluation (40 points)

Use the ground truth "Task1/gt/gt.txt" to evaluate your tracker using the **Multiple Object Tracking Accuracy (MOTA)** metric:

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t}$$

Where:

- FN_t is the number of false negatives in frame t.
- FP_t is the number of false positives in frame t.
- IDSW_t is the number of identity switches in frame t.
- GT_t is the total number of ground-truth objects in frame t.

Implementation Details:

1. Use **IoU** ≥ 0.5 to match predicted bounding boxes with ground-truth boxes.
2. Implement the **Hungarian Algorithm** (or equivalent) to find optimal assignments.
3. Track and count:
   - **False Positives (FP)**: Predicted boxes with no ground truth match.
   - **False Negatives (FN)**: Ground truth boxes with no prediction match.
   - **Identity Switches (IDSW)**: Ground truth objects assigned a different predicted ID in consecutive frames.
4. Print the MOTA score and a breakdown of FP, FN, IDSW, and GT counts.

4 Prediction & Kaggle Competition (10 points)

1. Repeat steps 1 and 2 on the "Task2" dataset and save the annotated video as `task2.mp4` (you are recommended to use **other tracking methods** for better performance).
2. Count the number of people in each frame (based on bounding boxes).
3. Save your pedestrian count in a .csv file with the following format (Number is the frame ID):

```
Number,Count
1,12
2,15
3,13
...
```

4. Submit the .csv file to Kaggle to see your public leaderboard score, which is calculated on 30% of the test images. Final rankings will be based on the remaining 70%, revealed after the competition closes.
5. **Submission Limit:** Each team can submit up to 5 times per day, so plan accordingly.
6. **Grading:** Your grade for this section will be determined by your ranking, with the top 3 teams receiving 10 points, and the lowest rank receiving 0.

5 Deliverables

1. **Jupyter Notebook**: Contains all code, outputs, and explanations.
2. **Output Videos**:
   - `task1_input.mp4`
   - `task1.mp4`
   - `task2.mp4`
3. **Pedestrian Count**: `task2_count.csv` (prediction file for Kaggle).