TP 4: Appearance-Aware IoU-Kalman Object Tracker

Objective: Extend IoU-Kalman tracker to include object re-identification (ReID)

Instead of relying solely on geometric information (IoU) consider also the appearance similarity between the predicted object state (form frame t-1) and the detected objects (from frame t).

1. Implement Object Re-Identification
   - Feature Extraction**:** Use a pre-trained **lightweight** deep learning model (EfficientNet, OSNet or MobileNet) to extract features from detected objects (image patches). You can download checkpoints of REiD model from obtained link or search in the Git repository.
   - Patch preprocessing:
     - Generate a patch for each detected object. The size of each patch is defined by its bounding box => im_crops
     - Patch Resizing: Resize each patch (im_crops) to the size of the image patches used to train the ReID model you are using. If the provided checkpoints are utilized, resize to (64, 128), following the approach of the Market1501 dataset used to train this OSNet model.
     - Convert BGR to RGB
     - Normalize

     Here is an example function to preprocess image patches:

     ```
     def preprocess_patch(self, im_crops):
         roi_input = cv2.resize(im_crops, (self.roi_width, self.roi_height))
         roi_input = cv2.cvtColor(roi_input, cv2.COLOR_BGR2RGB)
         roi_input = (np.asarray(roi_input).astype(np.float32) - self.roi_means) /self.roi_stds
         roi_input = np.moveaxis(roi_input, -1, 0)
         object_patch = roi_input.astype('float32')
         return object_patch
     ```

   - Compute the ReID vector for each patch
   - Distance Metrics for ReID**:** Use metrics like cosine similarity or Euclidean distance to compare feature vectors of detected objects with those of tracked objects
2. Combine IoU and Feature Similarity to make the association more robust. One common approach is to use a weighted sum. You could define a combined score S as follows:

   $$S = \alpha \cdot IoU + \beta \cdot Normalized\ Similarity$$

   where: α and β are weights that you can tune based on your application

   Normalized Similarity is obtained by normalizing the feature similarity score (e.g., for cosine similarity, this could be directly used; for Euclidean distance, you would need to invert it to get a similarity score).

   $$Normalized\ Similarity = \frac{1}{1 + Euclidean\ Distance}$$

   This ensures that a lower distance results in a higher similarity score.

TP 5 Appearance-Aware IoU-Kalman Object Tracker: Detector Extension

1. Integrate more efficient **lightweight** deep Learning-based object detector for pedestrian detection. To accelerate tracker inference, you can regenerate a file named `det.txt` with the obtained detections and then read the file as you have been doing so far (Pay attention to the file format).

2. Extra: Evaluate the performance of your multi-object tracking system (the version you have been able to develop). "Utilize the ground truth data file (gt) to compute HOTA, IDF1, and ID_Switch metrics, as well as the tracking speed on your PC.
    - Download official evaluation kit TrackEval : https://github.com/JonathonLuiten/TrackEval
    - If you have any issues with the tool, feel free to ask me for a step-by-step tutorial guide

Evaluation

You are required to submit an **Individual Report** that outlines your experience with the project, accompanied by a **video** that visualizes the **tracking results**. In your report, clearly describe the tasks you undertook throughout the project. Identify any challenges you encountered and explain how you addressed them.

Deadline for project submission: 20.12.2024