TP 2: IOU-based Tracking (Bounding-Box Tracker)

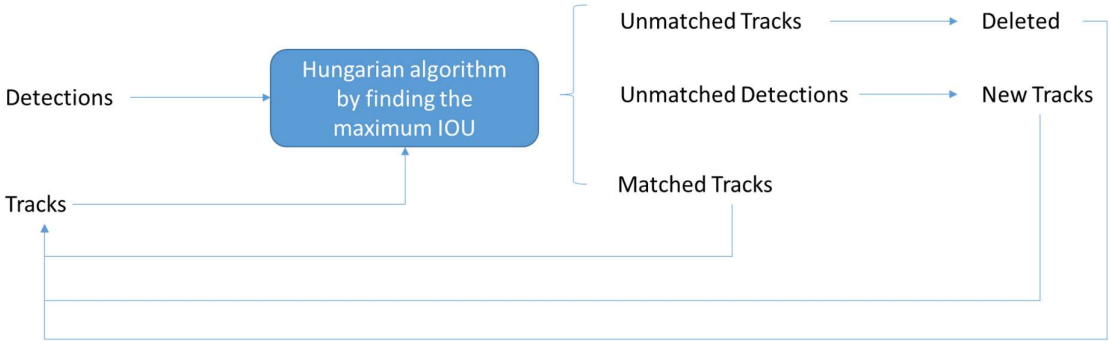
Objective: Develop a Simple IoU-Based Tracker and extend it for Multiple Object Tracking

- Object representation: bounding box
- MOT (Multiple Object Tracker)
- Data: pre-generated detections loaded from text file

Data:

- ADL-Rundle-6 sequence (img1 folder). All images are in JPEG and named sequentially to a 6-digit file name (e.g. 000001.jpg)
- Detections (det folder) and ground truth annotations (gt folder) are simple comma-separated value (CSV) files. For the detections, three different folders are available: public dataset (file download from MOT Challenge site), Yolov5s (using small model), Yolov5l (using large model). Choose the data that suits you best but be aware that the detection quality varies. The files are named the same way as their corresponding images.

1. Load detections (det) stored in a MOT-challenge like formatted text file. Each line represents one object instance and contains 10 values (fieldNames = [<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>]
   - frame = frame number
   - id = number identifies that object as belonging to a trajectory by assigning a unique ID (set to −1 in a detection file, as no ID is assigned yet).
   - bb_left, bb_top, bb_width, bb_height: bounding box position in 2D image coordinates *i.e.* the top-left corner as well as width and height
   - conf: detection confidence score
   - x,y,z: the world coordinates are ignored for the 2D challenge and can be filled with -1.
2. Create similarity matrix:
   2.1 Initialization
      - Define the lists or arrays to store the current tracked bounding boxes and the new detections for the current frame.
   2.2 Calculate IoU for all pairs
      - Create a similarity matrix (a 2D array) where each entry (i,j) corresponds to the IoU value between the i^{th} tracked object and the j^{th} new detection
      - The dimensions of this matrix will be (N×M), where N is the number of tracked objects and M is the number of new detections. Compute similarity score using the Jaccard index (intersection-over-union) for each pair of bounding boxes
3. Associate the detections to tracks
   - Apply the Hungarian algorithm using existing libraries (e.g. function linear_sum_assignement from scipy library for Python) to find the optimal assignment of detections to tracked objects
4. Implement track management
   - Each object can be assigned to only one trajectory (ID)
   - Create and update lists for matches, unmatched detections and unmatched tracks
      - ✓Matched -> update existing tracks based on associations
      - ✓Unmatched tracks -> remove tracks that exceed the maximum missed frames
      - ✓Unmatched detections -> create new tracks



5. Develop an interface for tracking results check to see if the tracker properly keeps track of objects:
   - Display Video Frames
   - Draw Bounding Boxes: Overlay bounding boxes for each tracked object on the frames
   - Show Track IDs: Label the bounding boxes with track IDs for identification
   - Save tracking video

6. Save tracking results in a txt file.

The file name must be exactly like the sequence name. The file format should be the same as the ground truth file (gt.txt), which is a CSV text-file containing one-object instance per line. Each line must contain 10 values. Update the id column (2th value) with the unique ID assigned to the track. The 7th value (*conf*) act as a flag 1.

TP 3: Kalman-Guided IoU Tracking (Bounding-Box Tracker)

Objective: Extend IoU-based MOT with Hungarian Algorithm by adding Kalman Filter

1. Integrate Kalman Filter (from Exercise 1) into the tracking System (from Exercise 2)
   - Modify the tracking algorithm according to the diagram below. Practical tip: represent the bounding boxes by their centroids to be able to apply the Kalman filter
   - Integrate the IoU calculation with the Kalman Filter's prediction step. The association process will now first predict the state of each track