University of Bucharest
Faculty of Mathematics and Computer Science
Department of Computer Science

Alexandru Tindeche
Alexandra Diaconu
Bogdan Alexe
MSC in Computer Science

Computer Vision - Laboratory class 7

Extracting traffic information from surveillance cameras

Objective

The goal of this lab is twofold:

- introduce basic operations in video processing such as reading, writing and displaying a video using OpenCV;
- extract traffic information (such as clustering trajectories) from video data provided by a surveillance camera positioned to take footage from an intersection.

In this laboratory class we will analyze the video data provided by a surveillance camera positioned to take footage from an intersection (Figure 1). The video data is given in the

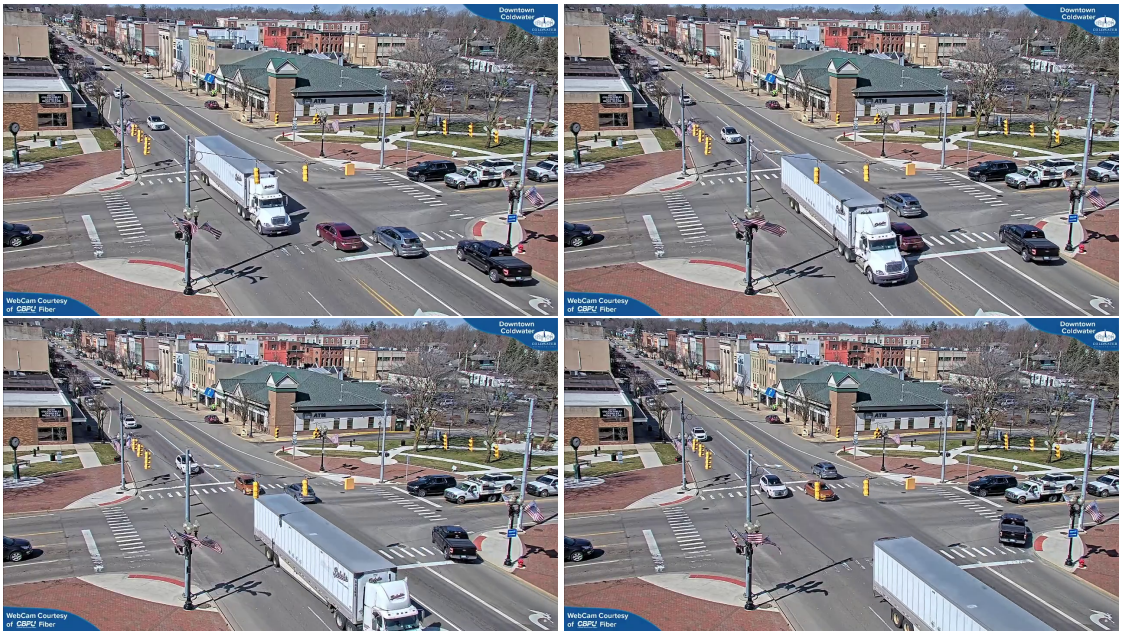

Figure 1: *Visualizing different frames of the video captured by a surveillance camera. We show different frames for the video file analyzed in this laboratory class.*

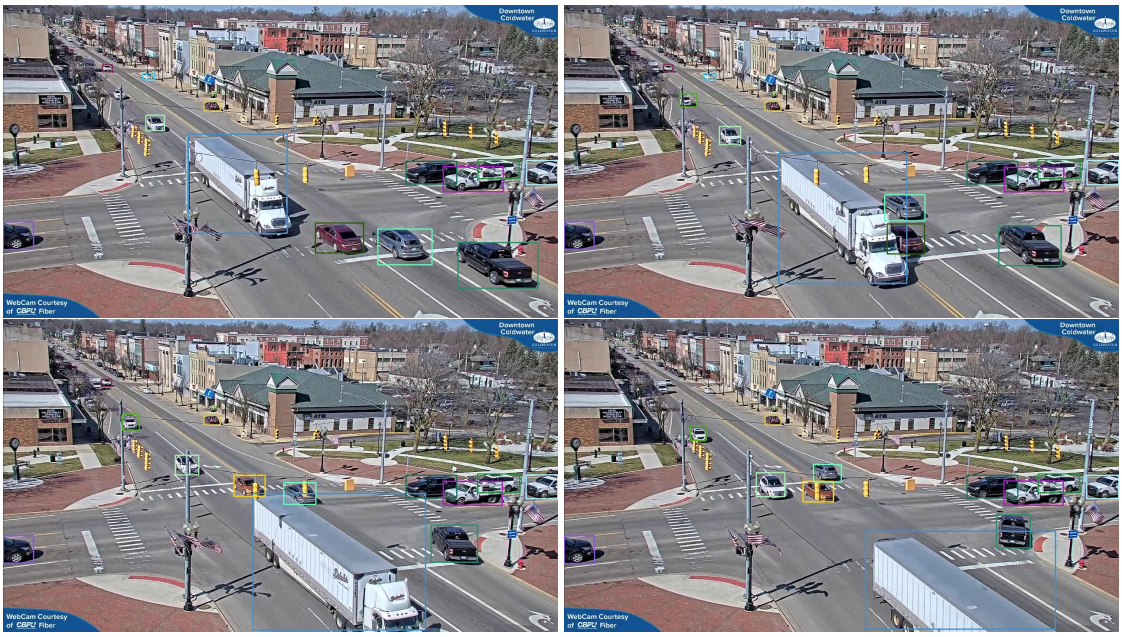

Figure 2: *Visualizing different frames of the video captured by a surveillance camera with overlayed bounding-boxes fitting tightly the detected vehicles.*

file *video-traffic.mp4*. The desired goal is to extract different information from this video (count the numbers of vehicles, cluster trajectories, etc.). As a pre-processing stage we have run an object detector trained to detect vehicles and gathered all the detections in the file *detections.csv*. Moreover, we have also grouped the detections in tracks, such that it easy to visualize the trajectory of each vehicle in the scene.

The data stored in the file *detections.csv* follows the format [frame x1 y1 x2 y2 track_id class_id class_name] where:

- *frame* represents the current frame number;
- the detected vehicle is represented as a bounding boxes [x1 y1 x2 y2], where (x1, y1) is the top left corner and (x2, y2) is the bottom right corner of the bounding-box;
- *track_id* represents the id of the track to which the detected vehicles was assigned;
- *class_id* represents the class id assigned to the detected vehicle by our used object detector;
- class_name represents the class name assigned to the detected vehicle by our used object detector.

Visualizing detected vehicles

The first task is to visualize detected vehicles in the scene. Your task is to write code that overlays in each frame of the video the corresponding bounding-box to each detected ve-

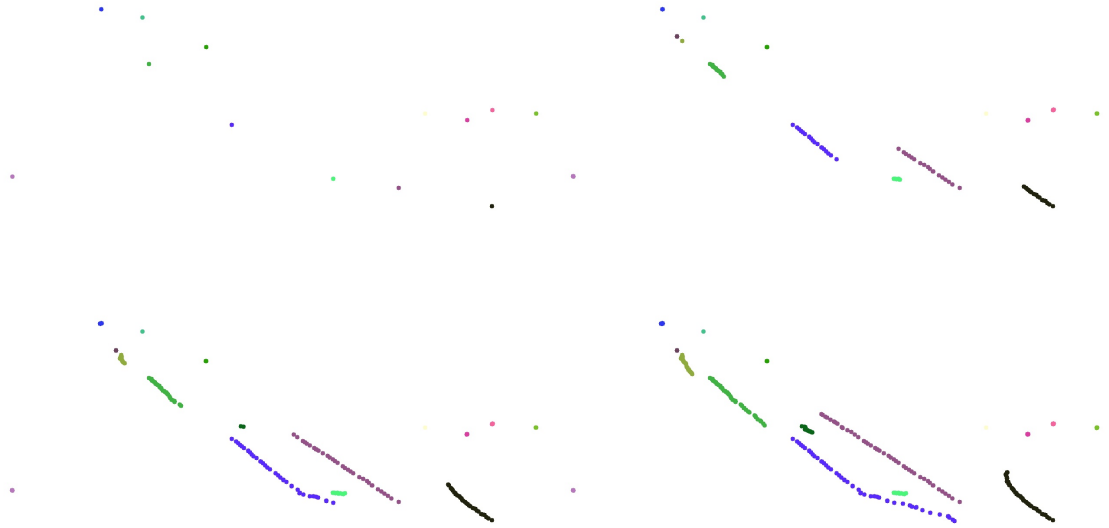

Figure 3: *Visualizing trajectories of the detected vehicles in video.*

hicle (Figure 2). Using a different color for each bounding box color will make your visualization look nicer. Visualize then the obtained video to check that everything is fine.

Visualizing trajectories

The second task is to visualize trajectories followed by the detected vehicles in the scene. As these detected vehicles are grouped in tracks your job is to plot each track by representing each detection in the track with a point (usually a good representation is to take the center of the bounding-box, see Figure 3). You can obtain a nice visualization by plotting trajectories with colors that fade away as you move to the next frames.

Clustering trajectories

The main task is to cluster trajectories of the vehicles in the scene (intersection). This enables then to extract traffic information, such as counting the number of vehicles following a specific trajectory or detect vehicles whose trajectory departs from a usual one (for example vehicles who violate traffic rules).

As trajectories of vehicles in the scenes contain different number of points we obtained new trajectories with exactly 150 points. We use in this sense the *interpolatePPolyfrom_spline* function. We can use the structure of the scene (intersection) to derive similar trajectories by taking the first and last 10 points in each trajectory. This will be sufficient to describe precisely the trajectory of a vehicle in the scene (from which part of the intersection comes and to which part of the intersection moves). By applying the t-SNE algorithm we can project the first 10 points and the last 10 points from each trajectory in a 2D feature space (a plane) where it is very easy to visualize the clusters (Figure 5). As we have 4 possible

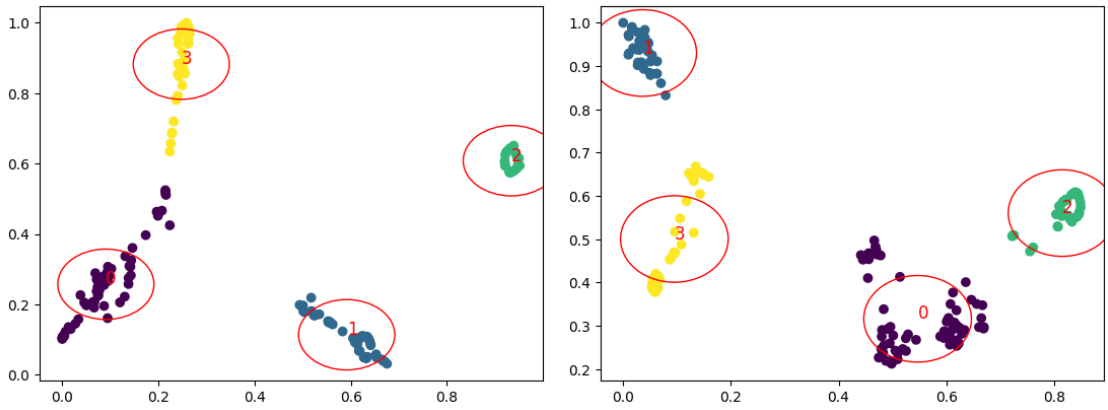

Figure 4: *Visualizing clusters for the first 10 points (left) and last 10 points (right) after using t-SNE to project the beginning and ending of each trajectory in a 2D space.*

directions in the intersections we will clusters the 2D points in 4 clusters using the well known *k-means* algorithm. We do this for the first 10 points in each trajectory and also for the last 10 points in each trajectory, building thus two different sets of points.

Building trajectories clusters. We can use the 2D clusters to build the trajectories clusters. By taking the cartesian product of the 4 clusters of the first 10 points with the 4 clusters of the last 10 points we obtain 16 prototypes containing 20 points, the first 10 points for the beginning of the trajectory and the last points for the last part of the trajectory. We thus can classify each trajectory of 150 points based on the first and last 10 points in one of the 16 classes (clusters). For each cluster we can then compute the mean trajectory (Figure 5).

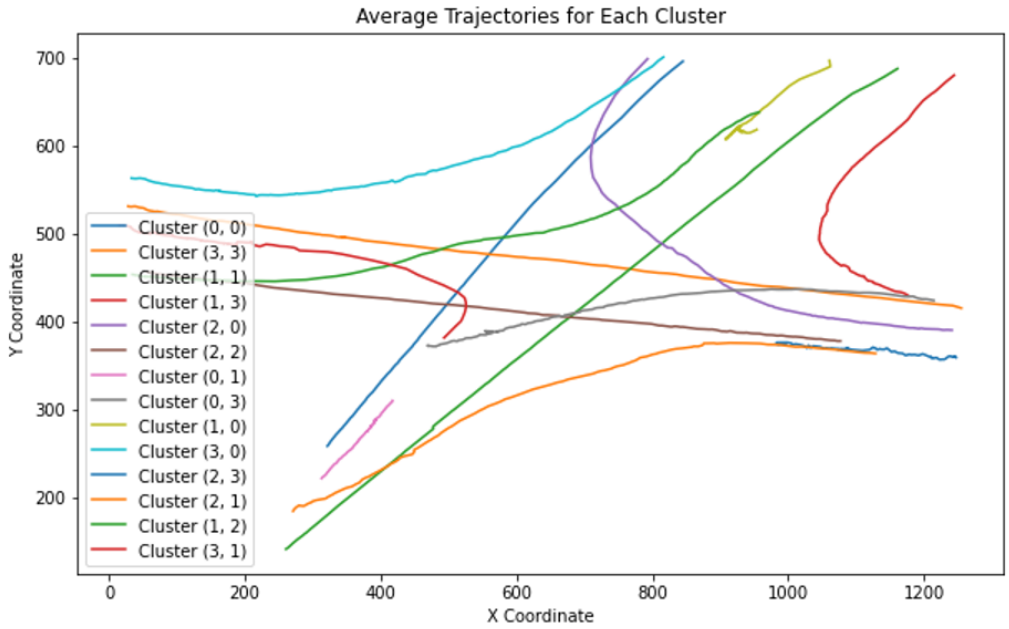

Figure 5: *Computed average trajectories to each cluster.*