Computer Vision

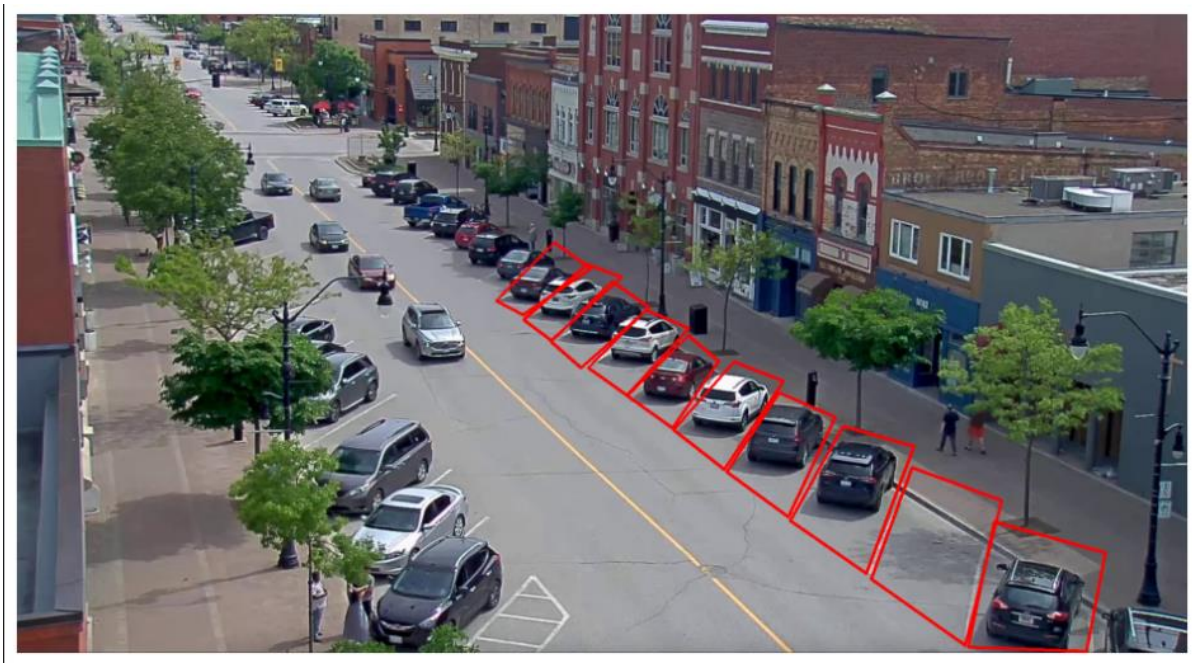
Project 2

Documentation

Alexandru – Radu Handac

Task 1

The objective of this task was to predict whether cars were placed on predefined parking slots. The main idea was to annotate the parking slots based on how the model would predict the cars, rather than annotating the exact area of the parking slot. This approach ensured that the annotations aligned with the model's detection patterns.



Having marked the parking slots, the next step was to detect the cars using a YOLO model from Ultralytics. By utilizing the latest model, Yolov9e, we could obtain accurate detections. With both detections and parking slots available, the next step was to check if a detected car was within a parking slot. This was achieved by calculating the center of each detection and checking if it lay within a parking slot.

This approach yielded satisfactory results, correctly detecting all vehicles in the parking slots.

Task 2

The second task involved handling videos and determining whether a car occupied a parking slot at the end of the video. Unlike Task 1, which dealt with images, Task 2 required processing videos and checking all ten parking slots.

The solution involved extracting the last frame of the video and applying the YOLO model to it. By calculating the center of each detection, we determined if a car occupied a parking slot. The results were recorded in text files, with "1" indicating an occupied slot and "0" indicating an empty slot.

Task 3

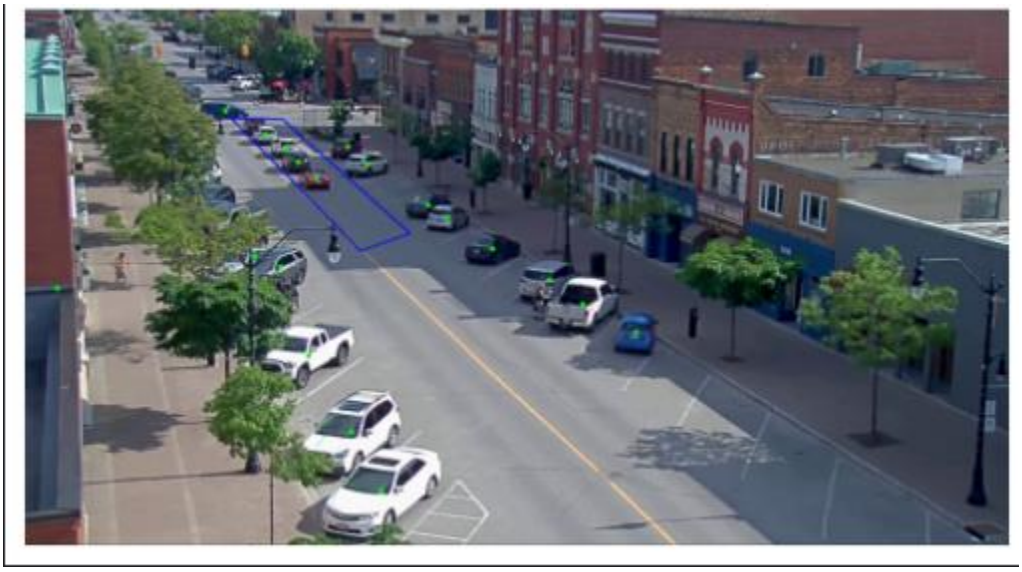
The third task focused on tracking a specified vehicle given its initial coordinates. The approach was to detect vehicles frame by frame, compute the IoU (Intersection over Union) for each detection, and update the bounding box if a higher IoU was found. The results were saved to a text file.

To compute the IoU, we determined the coordinates of the intersection rectangle by finding the maximum of the top-left corners and the minimum of the bottom-right corners of the two bounding boxes. We then computed the area of both bounding boxes and subtracted the intersection area to get the union area. Finally, we divided the intersection area by the union area to get the IoU.

For each bounding box detected in the frame, the IoU was computed with the current bounding box using the compute_iou function. If the IoU was higher than the current best IoU, we updated the best IoU.

Task 4

The final task was to count how many cars were waiting in a queue at a traffic light. The fastest approach was to extract the last frame of the video, apply the YOLO model, calculate the center of each detection, and check if it was in a hard-coded region of interest.



However, the model had limitations. Cars far away from the camera were not detected by YOLO, affecting the accuracy. While this approach was not perfect, fine-tuning the YOLO model with specific data could improve the detection accuracy for distant cars.

Conclusion

These are the four approaches for the tasks. The latest YOLO model performed well on these tasks but still has room for improvement. Fine-tuning the model with specific data could enhance its performance, especially for detecting distant cars in traffic queues.