Parking Space Detector @ NETLINK

This Python software is used to process and analyse video footage from a parking lot surveillance camera. It counts the number of cars in different parking places and determines how many parking spaces are empty using computer vision techniques and an object identification model. The code is explained in detail below:

1. Import the required libraries:

- 'cv2': This is OpenCV, a widely used computer vision library for image and video processing.

- 'pandas': A data manipulation and analysis library. It has been imported but is not utilised in the code.

- 'numpy': A numerical computing and arrays library.

- 'ultralytics.YOLO': It appears to be using the Ultralytics library's YOLO (You Only Look Once) object identification methodology.

2. Define the function'main()':

- Use the pre-trained model file 'yolov8s.pt' to train the YOLO object identification model.

- Launch the 'SaveTube.App-CCTV' video file. Using OpenCV's 'VideoCapture', read 'Video of a Parking Lot.mp4'.

- Using the 'load\_class\_list()' function, load class names from the file 'coco.txt'.

- Using the 'load\_areas()' function, load parking area coordinates (polygon vertices).

- Set variables to count cars and vacant parking spaces.

3. Enter the main processing loop:

- The loop uses cap.read() to read frames from the video and resizes each frame to a fixed size (1020x500).

- The YOLO model is used to forecast items in the frame, and bounding boxes are created as a result.

- The process\_objects() function is used to determine and count the objects in each parking space.

- The count\_total\_cars() method returns the total number of vehicles in the parking lot.

- To draw polygons around parking places and mark them with counts, the draw\_objects() function is utilised.

- The frame shows the total number of cars and empty parking spaces.

4. The loop is continued until a key press event (Enter key) happens, at which point the loop is terminated.

5. The video capture is released and all OpenCV windows are closed once the loop is completed.

6. Define utility functions as follows:

- load\_class\_list(filename): Reads class names (e.g., 'car','motorcycle', 'cellphone') from a text file and returns a list of class names.

- load\_areas() returns a dictionary with keys corresponding to parking area numbers and values corresponding to lists of polygon vertices defining each parking area.

- process\_objects(boxes, areas, class\_list): Examines the bounding boxes of identified objects to determine which parking area they belong to and what type of object they are.

- count\_total\_cars(area\_objects): Returns the total number of cars in the parking lot based on the objects with the highest number in the parking area.

- draw\_objects(frame, areas, object\_counts, total\_cars, space): This function draws polygons around parking areas, color-coding them based on the number of objects and displaying total car and vacant parking space counts.

7. Check the main() function to see if the script is being run as the main programme.

*In summary, this code analyses video footage from a parking lot surveillance camera, detects objects (e.g., cars), counts them in various parking areas, and computes the total number of cars and empty parking spaces in the parking lot. The outcomes are shown in the video frames.*