Car and Pedestrian Detection Documentation

Overview

This Python script demonstrates car and pedestrian detection using Haar Cascade classifiers from OpenCV. The script reads frames from a video file (e.g., ‘cars.avi’), processes each frame, detects cars, and displays bounding rectangles around them.

Prerequisites

- Python 3.x
- OpenCV (cv2) library

Implementation Details

1. Imports:
    - numpy (as np): Numerical computations library.
    - cv2: OpenCV library for computer vision tasks.
    - time: Although imported, it’s not used in this snippet.

2. Loading the Haar Cascade Classifier:
    - The car_cascade variable holds the path to the pre-trained Haar Cascade classifier XML file for car detection.
    - car_classifier is initialized using cv2.CascadeClassifier(car_cascade).

3. Video Capture:
    - The script opens a video file (e.g., ‘cars.avi’) using cv2.VideoCapture('cars.avi').

4. Processing Frames:
    - Inside the while loop:
        - Reads frames from the video using capture.read().
        - Converts each frame to grayscale using cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY).
        - Detects cars in the grayscale frame using car_classifier.detectMultiScale(gray, 1.2, 3).
        - Draws rectangles around detected cars using cv2.rectangle().
        - Displays the modified frame with rectangles using cv2.imshow('Cars', frame).
        - If the ‘q’ key is pressed, the loop breaks.
        - If no frame is read (end of video), the loop also breaks.

5. Cleanup:
    - After the loop, the video capture is released (capture.release()), and all OpenCV windows are closed (cv2.destroyAllWindows()).

Usage

1. Ensure you have Python and OpenCV installed.
2. Place the pre-trained Haar Cascade XML file (e.g., ‘haarcascade_car.xml’) in the ‘cascades’ folder.
3. Replace ‘cars.avi’ with your video file path.
4. Run the script.

Notes

- Adjust the detection parameters (scale factor, minNeighbors, etc.) for optimal performance.
- Evaluate accuracy using metrics like IoU, precision, recall, and F1 score.