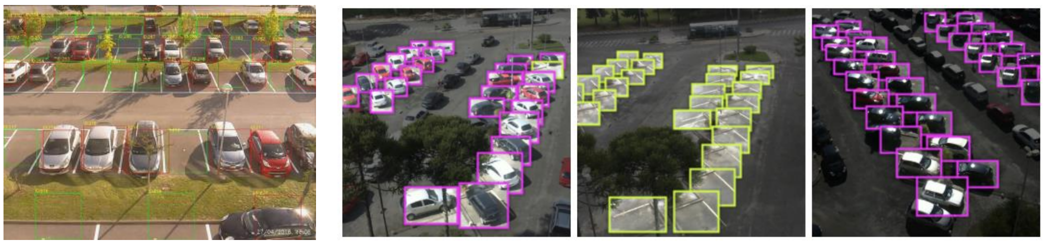
COMPUTER VISION - LAB 5

Computer Vision 2022 (P. Zanuttigh, M. Mel, M. Toldo, A. Simonetto)

Topic: Detect the free parking slots
Goal: Analyze the provided images and get how many free or occupied parking slots are present inside each image



You are requested to develop a computer vision algorithm in C++ analyzing the images and getting if the slots in the parking lot are free or occupied. You can use both traditional computer vision algorithms (e.g., edge, corner or feature detectors), or machine/deep learning strategies or a combination of the two things.

The workflow is as follows:

1. Download one or both the parking lot datasets from the corresponding URLs
    a. http://cnrpark.it/ (easier)
    b. https://web.inf.ufpr.br/vri/databases/parking-lot-database/ (a bit more advanced)

2. Get the data
    a. The CNRpark datasets contains both the full images to evaluate the approach and the single slot patches that you can use to train binary classifiers (i.e., free vs. occupied) based on ML techniques. The .csv files camera.csv contains the location of the slots while the CNRpark+EXT.csv and the .txt contain the ground truth solution.
    b. The PKLot dataset also contains both the full images and the slot patches (the latter in the “segmented” folders). It considers larger parking lots with more slots that can be also rotated in different ways. The slot positions and ground truth information are all inside the .xml files (for parsing have a look at the OpenCV FileStorage class or use any xml file parser).

3. Develop an algorithm for detecting if each parking slot is free or occupied. The choice of the best strategy is up to you, possible ideas are:
    a. Run edge and/or corner and/or feature detector into the slot and analyse found features (i.e., count number of corners, length of edges, etc….) or feed this information to a deterministic or ML-based classifier.
    b. Analyse image statistics into the slot (e.g., color variance, gradients…)
    c. Extract features (e.g. SIFT, ORB), then use ML classifiers (e.g., SVM or Random Forest, recall the bag-of-words approach).
    d. Use a deep learning (DL) binary classifier (e.g., a simple CNN with 3-4 layers). You can segment the slots, normalize them to a common size and orientation and then feed to the DL classifier.
    e. Use a deep learning object detector (e.g., Yolo, Mask-RCNN) and focus on the car-motorbike objects
    f. You can combine multiple strategies for optimal performances!!
4. Test your approach on various images and evaluate the performances (the CNRpark allows also to test in rainy or overcast conditions)

Notes:

- You can choose your preferred strategy, any idea different from the proposed ones is welcome.
- Develop the approach in C++ (in case use Python only for deep networks module but keep C++ for pre- and post-processing).
- Avoid directly cutting and pasting material from the web (in particular the cnrpark webpage provides a deep network already trained for the task, do not use it).
- Recall you need to deliver the code, the report (mandatory for the final project) and some output results.