HomeWork-3: Parking Lot Vacancy Detector

Due Date: 11:59 PM, 12th April 2020

A public infrastructure has various parking lots. The parking lots get completely occupied very often and the public visiting the infrastructure spend too much time looking for a parking space, unaware that the parking lot is completely occupied. They would like to implement an automated solution to convey this information by displaying the number of available parking spaces at the entrance to the parking lot.

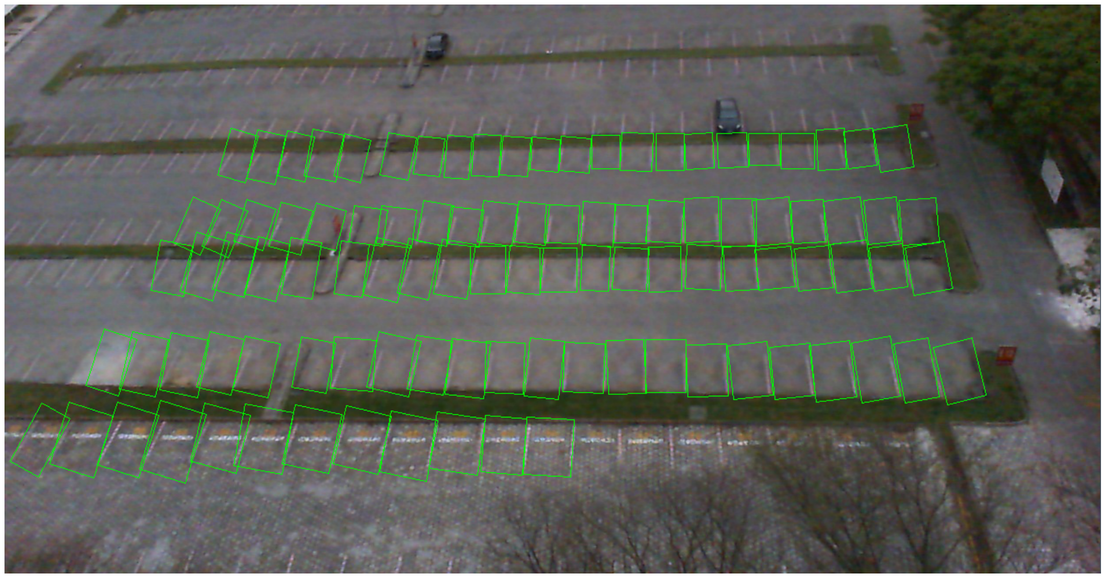
These parking lots are overlooked be surveillance cameras. The task is to leverage them to detect and count the empty parking spots.

Data set:

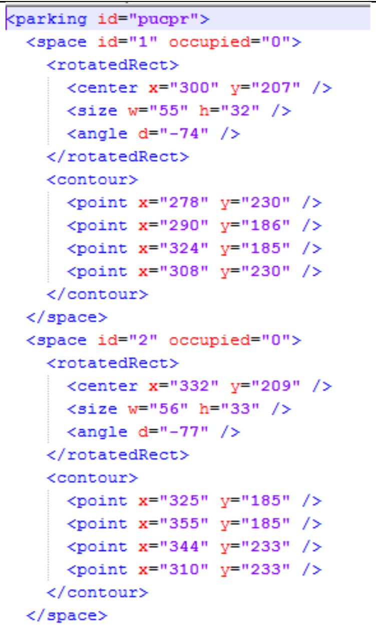
The data set consists of images from three separate parking lots parking1a, parking1b and parking2. Each of the parking has data set sorted for three different scenarios cloudy, rainy and sunny. Each single day is represented by the folder, for example the folder (e.g. PKLot/parking/cloudy/2012-12-12/) contains multiple snapshot images (like 2012_12_12_10_00_05.jpg) and the corresponding ground truth file (like 2012_12_12_10_00_05.xml).

Ground Truth(.xml)

Each image in the parking2 has 100 annotated parking spaces (see image below), along with the location, and occupancy information of the parking spot (Every image is supplemented with a ground truth file. For example (parking2/cloudy/2012-09-12/2012-09-12_06_05_16.jpg) has the ground truth file (parking2/cloudy/2012-09-12/2012-09-12_06_05_16.xml).



XML File description:

Space: each parking spot Id: Space ID Occupied: 0 – unoccupied, 1-occupied rotatedrect: Center: Of the rectangle Size: of the rectangle Angle: orientation of the rectangle Contour: Points on the contour Please note that OpenCV python does not have rotatedRect data structure. One would have to come up with your own logic to extract the bounding box.	

```
<parking id="pucpr">
  <space id="1" occupied="0">
    <rotatedRect>
      <center x="300" y="207" />
      <size w="55" h="32" />
      <angle d="-74" />
    </rotatedRect>
    <contour>
      <point x="278" y="230" />
      <point x="290" y="186" />
      <point x="324" y="185" />
      <point x="308" y="230" />
    </contour>
  </space>
  <space id="2" occupied="0">
    <rotatedRect>
      <center x="332" y="209" />
      <size w="56" h="33" />
      <angle d="-77" />
    </rotatedRect>
    <contour>
      <point x="325" y="185" />
      <point x="355" y="185" />
      <point x="344" y="233" />
      <point x="310" y="233" />
    </contour>
  </space>
```

Assignment steps:

1. Create a training dataset
2. Train a cascade classifier.
3. Car Detection: Use the classifier to detect cars in a test image.
4. Parking spot analysis application: Use the detected cars to decide if a parking spot is empty.

Cascade Classifier Training:

1. Training set (3 Pts):
    a. Use **Sunny** scenario in each parking lot set to create your training set.
    b. Write a python program to extract training images using the XML file.
        i. Use any XML parser library to read the xml file and extract training images. Example below

2. Train your cascade classifier (3 Pts)
(OpenCV provides utilities to train cascade classifier: https://docs.opencv.org/4.2.0/dc/d88/tutorial_traincascade.html)
    a. Please provide examples of the sample dataset used for positives and negatives and report and justify the number of positive and negative samples used along with the no of stages used to train the cascade classifier and justify in the report.
    b. Use two sets of features to train your classifier
        i. HAAR - Haar-like features
        ii. LBP - Local binary patterns
        c. Report performance of the classifier that is chosen
3. Car Detection (3 Pts):
    a. Use images from rainy and cloudy datasets to test your algorithm
    b. Write a python program that takes the cascade classifier parameters as input and a test image (from rainy and cloudy datasets) and performs car detection
    c. Test your detection using both sets of features, Haar and LBP
4. Parking lot analysis (3 Pts):
    a. Using the detected cars, and the location of the parking spots, determine, if a parking spot is occupied or empty.
    b. Use the ground truth to compare and summarize your results.

        i. Test 25 images each from rainy and cloudy (total 50 tests) dataset and report the true positives, false positive, accuracy.
        ii. Some scenarios might have multiple cars but the parking spots might not be annotated, report your results only with respect to the annotated parking spaces and ignore the rest.
        iii. Populate the table below with your findings.

Test Image	Classifier Feature	Training			TP	FP	Accuracy
		Stages	No. of Positives	No. of Negatives			

5. Report (3 Pts):
    a. Document your findings in the report
    b. In each of the steps of training, testing, and performance analysis, please include in your report, with example images, a discussion of the following:
        i. what challenges or limitation do you see in each step and what could be done to improve the same
        ii. compare/show or discuss expected difference(s) before and after the improvement action.

Instruction:

1. Submit code on Github
2. All coding must be done using **python**
3. Use tools available in OpenCV.

Dataset:

1. Dataset can be downloaded at: https://www.dropbox.com/sh/5wd3eb35dt7yp2h/AACyHVKO4AQ4oGN6fe-5Wo1Ka?dl=0

Submission Instruction:

1. Submit only the source code files, the cascade classifier (.xml) files and the report.

2. The TA will have the OpenCV setup in his development environment and will paste your code and run it. Do not use any third-party libraries (exception XML parser) as the TA will not have access to it.

Include a readme file to describe any required set up, like location of image files that was hard coded, or any other information that is required to run the code

Note. The training and testing can take unusually long time. Please start ahead of the due date.