Classification of Parking Spaces

1) Creating a sub-dataset

For this purpose, I wrote a Python script (create_sub_dataset.py) to subsample the dataset (original dataset is too large 4.5 GB).
My subset has **1600** training, **300** testing images.

- 1600 training images have 800 empty, 800 occupied images
- 300 test images have 150 empty, 150 occupied images.

2)Implementing the Bag of Words approach in Matlab

The procedure is as follows:

a) Extracting SIFT features of images in the training data
b) Perform k-means clustering (cluster means will be the "words")
c) Represent images with the histogram of these words (histogram becomes the feature vector)
d) Train a classifier by using those feature vectors

3) Classification

For classification, I used SVM with RBF Kernel

3a) Parameter Optimization for SVM

The parameters that I am optimizing is:

a) **k** (number of clusters from k-means)
b) **C** (C value of SVM)
c) **gamma** (gamma value of RBF Kernel for SVM)

To get best parameters for SVM, I did **5-fold cross-validation** with grid search for parameters C and gamma, I picked the parameters that gave the best result.

4)Results

Currently, the parameters in my system for SVM with RBF Kernel are:
k = 800
C = 500;
gamma = 0.00025;

With these parameters, I got the following classification results,
*Training Data
Accuracy = **95.875%** (1534/1600) (classification)
*Testing Data
Accuracy = **86.3333%** (259/300) (classification)

I realize that these results can be better. So I will work more on optimizing the parameters, or try different classifiers like Adaboost.

Notes:

As a side note, as we are doing 2-class classification problem, we can also use measurement metrics like False Positive, False Negative, True Positive, True Negative.

In the email attachment, I added my codes and also there are the results of my system applied to the actual full parking lot images. (p2Result.png, ...). You can see them below.

In those results, green rectangles represent available space, reds represent occupied space.

The procedure for that is as follows: as xml file is provided for each parking lot, I crop the parking space sub-images, make the necessary rotation(parking spaces are tilted) and put it into my classifier and make a prediction. If my system predicts that space as occupied, I put red, otherwise green

As you can see, the system works, however there are some mistakes, to overcome these problems I will try the thing that I mentioned above in this email.

(I did not set up a GitHub for this project because you told me not to share this project)