Thought Process and Instructions to run the code (Updated)

Author: Shubham Agarwal
Prepared for TrademarkVision Technical Interview
Date: 29th December 2016

Template Matching
The first thing thats comes to mind is template matching but considering different scales of of the probe images and logos, this will not be efficient. Somehow if we are able to detect the logo, then also we will be facing difficulty in drawing the bounding box. We need to take into account the scaling for sure. One more problem is that we will need a threshold so that we do not detect template in probe image without any logo. We can set some threshold for current set of images but for scalability that's not a good idea.

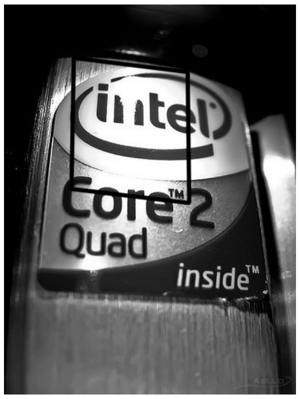
Keypoints based Detector(I used SIFT)
Considering the shortcomings of the template matching, a keypoint based detector seems to be a much better approach. I tried using the SIFT detector and discriptor from OpenCV in python. SIFT feature discriptor is scale, rotation and translation invariant so we can get better results compared to the template matching case. To account for skewness I used the skewed synthetic images as explaned in next section. Once we get good enough matches, we can compute corresponding homography to get the bounding box in the probe image. This approach is scalable as once we add new logos, we can just add the corresponding keypoints and label and utilize the same code.

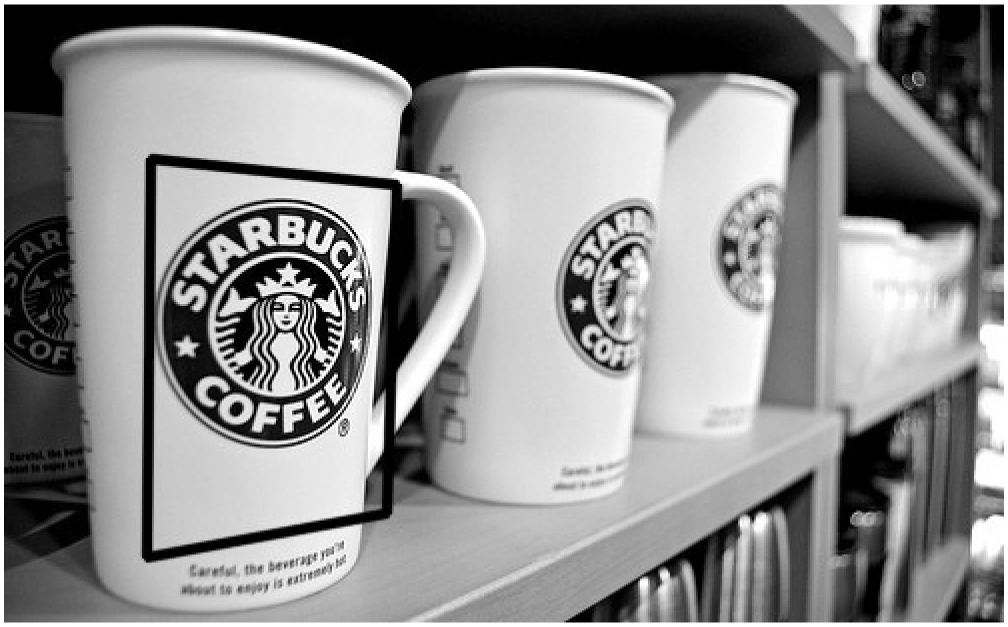
SETUP USED: Anaconda Python with OpenCV installed
Steps:
1) Extract the contents of zip file in some folder inside yor PC.
2) Open Python Console(I did with Spyder in Ubuntu 14.04).
3) Run “logoMatch.py” present in the extracted folder.

The accuracy will be printed as the console output. For me this code gave 61.11%(up from 52% earlier) accuracy. I drew the bounding box for only the images where I was able to obtain the homography matrix. Once you run the script from step 3) above, the images with bounding box will be generated in the “matchResults” folder.

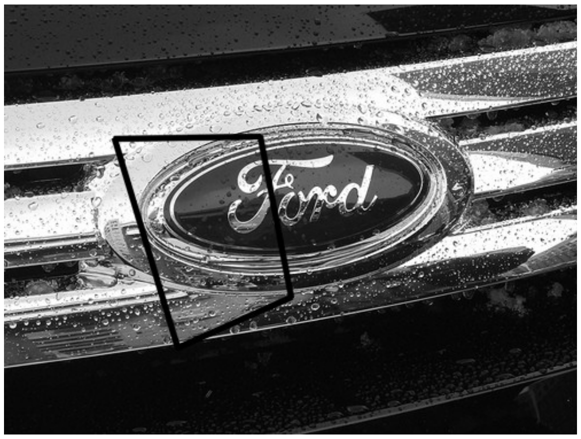
I have given the images I obtained using these steps in “matchResults_Shubham” folder. Here, I obtained proper bounding box for good images(where the trademark is visible properly). Some example images are shown below:

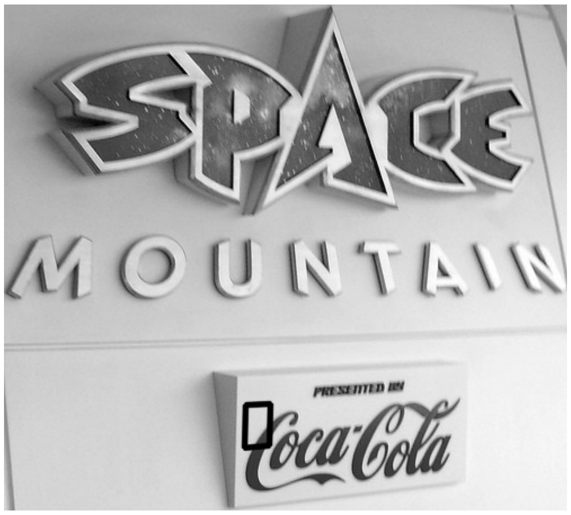
Good Matches and Bounding Box




STARBUCKS COFFEE
Careful, the beverage you're about to enjoy is extremely hot

Correct Matches but bad bounding box





Convolutional Neural Networks
The final approach I used is using convolutional neural networks in TensorFlow. CNN's are state of art is recognition field so I decided to give it a try. Since we have very less training data, we need to generate sufficiently varied training images for the neural network. Also, we need to make sure that we cover as many cases as possible to account for skew, rotation, traslation etc. For this I used an affine transformation with random parameters.

Generating Synthetic Images from given data
Setup Used: Python with OpenCV and TensorFlow
Steps:
1) Run “moreImages.py”.
2) The new Images will be obtained in the “test” directory.
Note: The logos used to compute the images are present in the “synthetic” directory in the extracted folder.

PORSCHE
Sprite
B M W

These synthetic images can be very useful with other approches also. SIFT showed considerable improvement with synthetic images

Network
I used TensorFlow with 5 layer convolutional neural network. The input image size is 128x64. The network architecture I used has been successfully deployed for number plate recognition systems. The network training is very slow and hard to do on laptop without GPU. Currently, I did the comparison for only two classes(ford and vodafone) with 1000 syntheticaly generated images.

With this five layer network, I got 59.9% accuracy if I consider only two classes. Although, this will surely reduce if we include more classes, we can still improve the recognition if we use more training data. Another thing which we can do is generate synthetic images with logo overlayed on some random background and train the network using these synthetc images with logo and background.

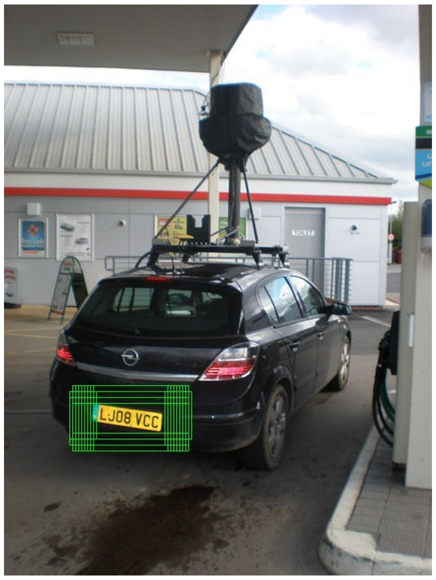
Setup Used: Anaconda with Python 2.7 with opencv and tensor flow installed
Steps:

...continue after generation of synthetic images.
Train and Test the network
4) Ensure the test images are there in probes directory in the extracted folder.
5) Run "5layer_cnn.py"
6) This will print the training accuracy after each iteration of training.

This approach is slower while training as we will be using thousands synthetic Images for the network training. Scale is also an issue because if we need to include more logos, we need to model and train the network again. Although, if we consider we have millions of images that cover every logo, this approach is best. Inspite of these shortcomings, once trained at full scale this approach gives more

accuracy compared to the previous two given sufficient variety in the training data.
To be implemented:
For compuing the bounding box we can use the sliding window approach. Here we pass only a portion of image to the network and take the windows that match more than the threshold. When we consider the intersection of such windows, we can get the bounding box. This will also make the network more accurate.

An example of this used for number plate recognition is shown below:



source: https://matthewearl.github.io/2016/05/06/cnn-anpr/

Overall I think that with current set of constraints SIFT seems to be the best approach. However, if I have more resources(data and processing power), convolutional neural network can overtake the SIFT in terms of accuracy.