Machine Learning Engineer Nanodegree

Capstone Project: Image search Engine

Zhuo Chen

Jan 10th, 2017

Definition

Project Overview

The objective of this project is designed and implement an image search engine and test the performance on UKbench dataset.

In UKbench dataset, each set of pictures contains four images of the same object. The goal of the system is for a given picture from the dataset, find out the at least another picture in the same set.

Problem Statement

The objective of this project is designed and implement an image search engine and test the performance on UKbench dataset.

The problem can be decomposed into following steps as:

1. Implement a feature extractor to extract the features of each picture
2. Apply the feature extractor to establish the feature database of the whole dataset
3. For each searching image, extract the features. Iterate through the database and find the four 'closest' images. There should be three other images, however, the image itself will also be searched and it will be counted as the 'closest' image. The distance metrics will be explained in later sections.
4. In order to tune and improve the performance of the system, it will first be tuned on a training set. The final performance will be evaluated on the testing set.

Metrics

The performance of the system will be evaluated by accuracy. There are four images for a same object in the dataset, a correct search would be, for a given image, all the other three images will be found.

Thus, for each search operation, the accuracy is:

$$Accuracy = \frac{number\ of\ correct\ results}{4}$$

If the feature extractor works properly, the image itself will have the closest distance. The best-expected performance will be 1 and the worst should be at least 0.25.

Analysis

Data Exploration

The dataset adopt for this project is the UKbench dataset [1]. The dataset can be downloaded from the following link:

http://vis.uky.edu/~stewe/ukbench/ [2]

A glance of the dataset is shown as Figure 1:

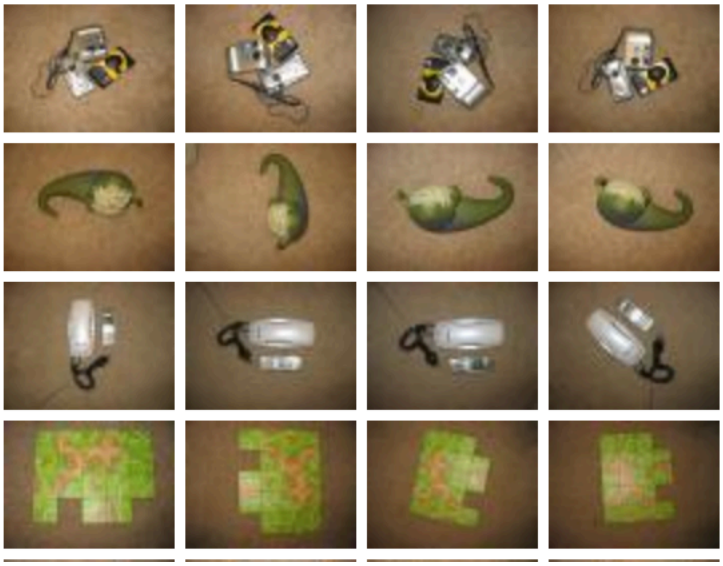


Figure 1 A few samples of the UKbench dataset

Some very important features of the dataset are:

1. Many of the images are focus of one object
2. For the same object, the images were taken in different directions with regards to the object.
3. For the same object, most of the images are taken with a similar background
4. For many of images of the same object, the light condition is similar.

The website mentioned that for each different subset of the dataset, the performance was different and plotted in the following image.

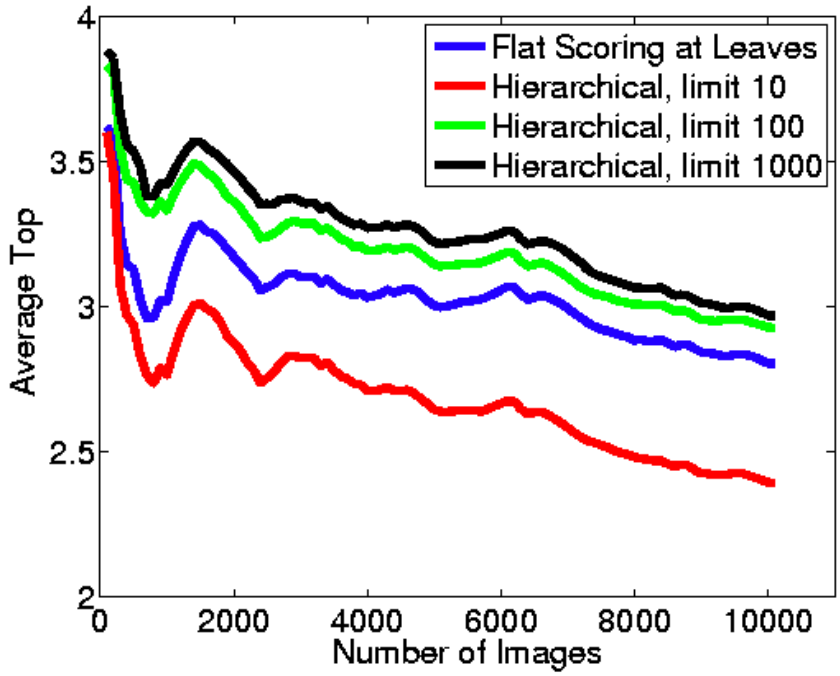


Figure 2 Plot of the performance versus subset [2]

Some information can be interpreted from this plot:

1. In general, the performance is decreasing alongside the number of images, which indicates the later subsets have higher difficulty.
2. The performances vary alongside the dataset, which indicates the samples are not distributed randomly in the dataset.

This project will target on the 'plateau' region of the plot above, which is 4000 to 5000, which could be suitable subsets to evaluate the performance and the difficulty of these subsets tend to be stable.

Also, it is necessary to split the training set and testing set randomly. The attributes of the samples data set are distributed un-randomly.

Exploratory Visualization

From the previous section, some features of the dataset was highlighted and these features could be suggestions that the color histogram could be a good candidate to expression the features of the image. A very common color representation is RGB, which the pixels have three

values each represents Red, Green, and Blue intensity. It a good format for storing and displaying images on the computer, however, there is another format, HSV, is more friendly for the way a human perceives color [3].

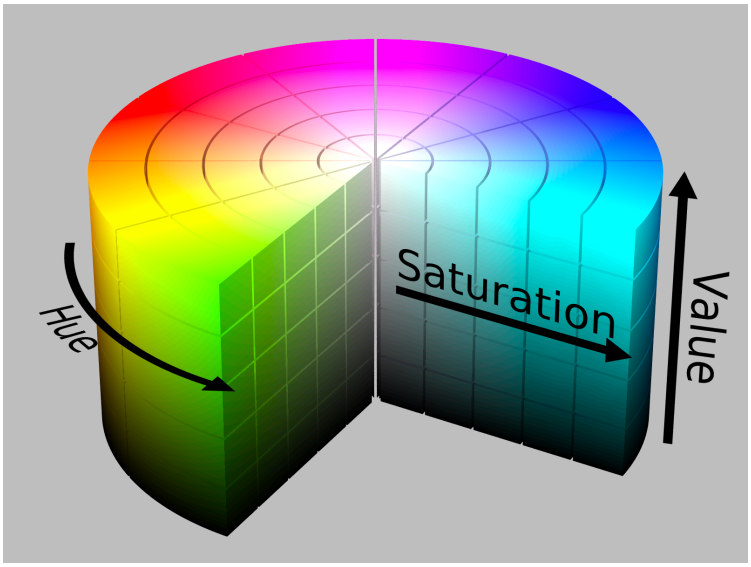


Figure 3 HSV cylinder [3]

Figure 3 is the plot of how colors are stored in HSV domain. Instead of R, G, B, the color is stored in terms of Hue, Value, and Saturation.

To make ann exploratory visualization, a sample image was selected from the dataset(No.50) as:

Figure 4 A sample image of the UKbench dataset

And the HSV histogram plot:

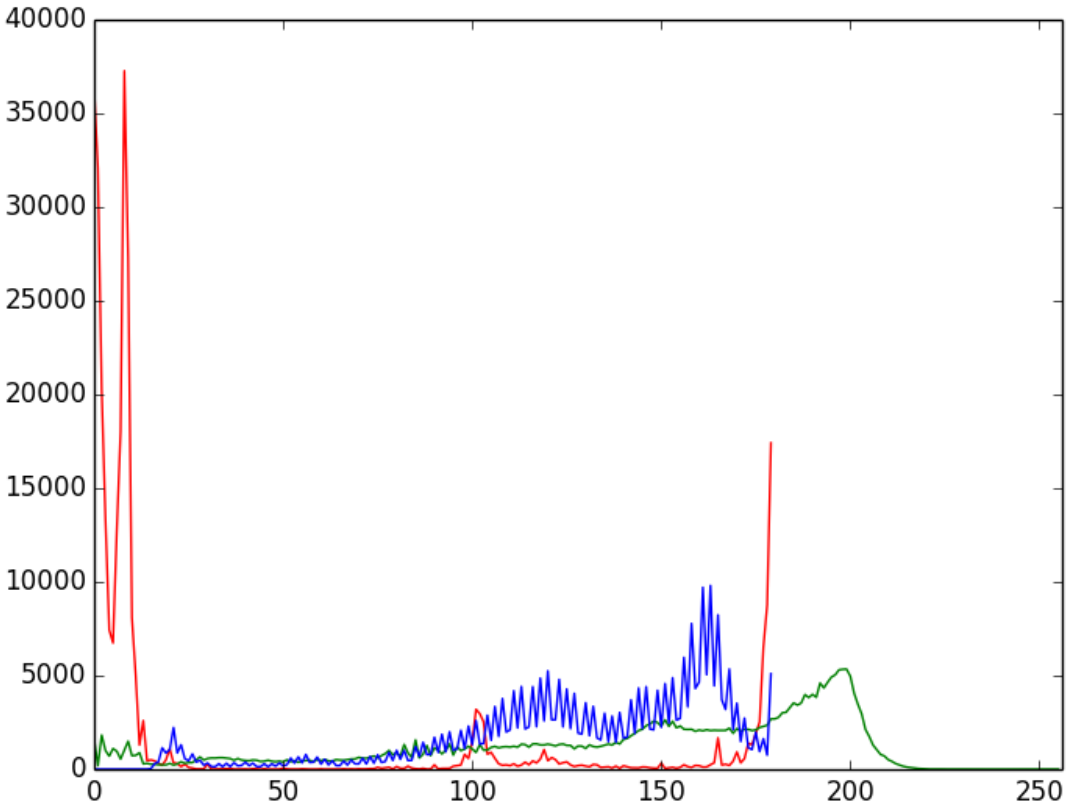


Figure 5 HSV histogram plot of No.50 image in the UKbench dataset

Methodology

General pipeline

The general workflow of the project can be summarized as following steps:

1. Implement a feature extractor to extract the features of each picture
2. Apply the feature extractor to establish the feature database of the whole dataset
3. For each searching image, extract the features. Iterate through the database and find the four 'closest' images. There should be three other images, however, the image itself will also be searched and it will be counted as the 'closest' image.

This is the pipeline given by Adrian [4] on his online tutorial.

Data Preprocessing

As it was discussed in a previous section, this project will use data from 4000 to 4999 from the full dataset.

As there are 1000 images involved, there would be 250 objects as each object has 4 images in the database. The 250 objects will be randomly split into a training set and test set in a fraction of 75% and 25% randomly. To ensure during every tuning and training process, we use the same samples as training set, the random state of the train-test set splitting was set to a fixed number.

Implementation

Feature extraction

As the most important part of bridging computer vision and machine learning, the feature extraction process provided the chance of the features can be learned and described by the model.

In this project, the HSV histogram of the image will be extracted from the image. Rather than extracting the histogram of the full image, some features of the images in this dataset would help improve the performance.

First, the object is placed in the center of the image. By applying a rectangle mask to the image, we can extract the most of the object. Also, the background of the image is also similar, which can also be used as a reference.

The parameter for this process is the bin size of each channel. If the number of the bin size is too large, the model may tend to over-describing the current sample, leading to over-fitting. Also, if the number of the bin size is too small, the model may fail to describe the image [4]. This will be the main tuning parameter for the refinement process.

An example has been generated to demonstrate the workflow of the feature extraction process. The right-side image is the original one, and the left-side is the image after HSV transformation.

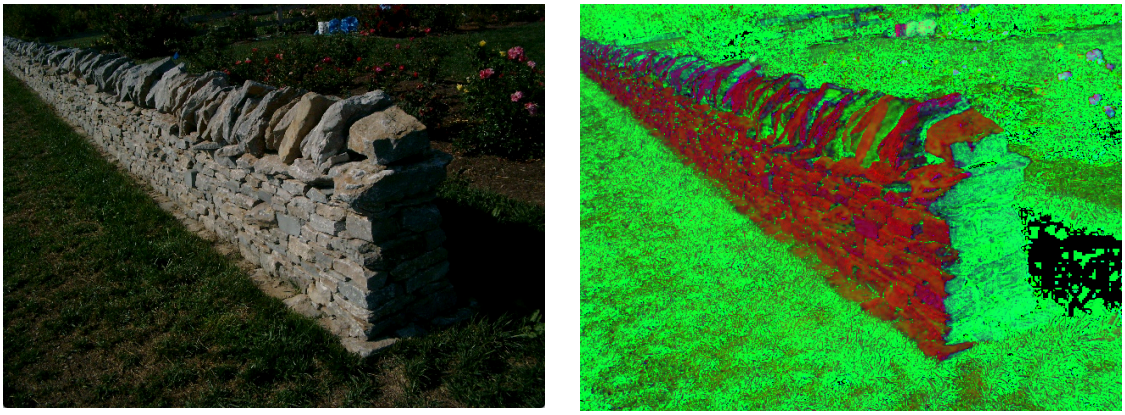


Figure 6 Original image and HSV transformed image.

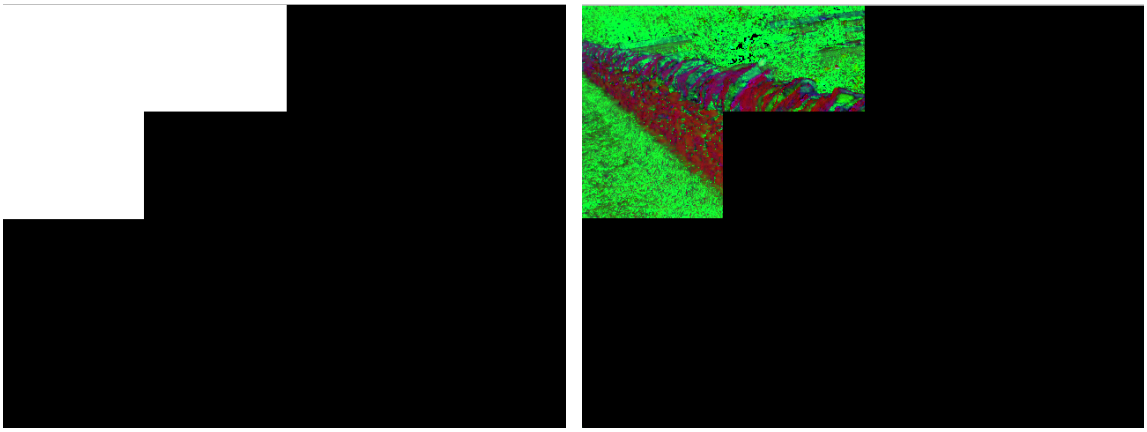


Figure 7 Corner mask and the effect of applying the mask on image

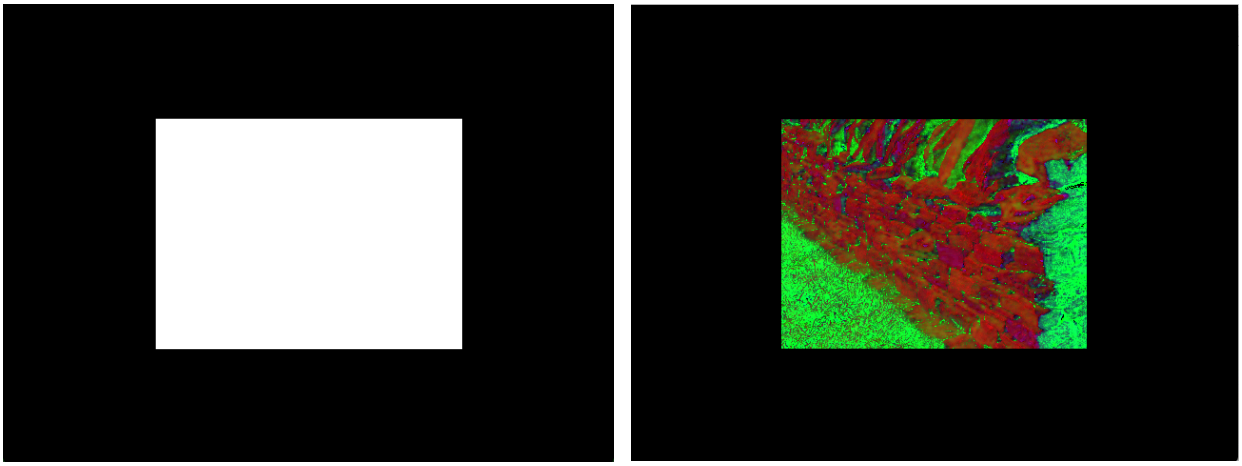


Figure 8 Center mask and the effect of applying the mask on image

Figure 7 and Figure 8 show the shapes of the masks and effect of applying the mask. All the histogram of the fragments will be calculated and stored as the feature vector of the image.

Distance comparison

The distance metric used in this project is the chi-square [5]:

$$d(H_1, H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)}$$

The chi-square [5] distance calculation is already implemented in openCV as:

cv2.compareHist(histA, histB, cv2.cv.CV_COMP_CHISQR)

this function will return the float as the distance.

Image searching

The image searching process can be generalized as the following process:

1. Apply feature extractor on the dataset and store the output as an index file
2. Apply feature extractor on the searching image and obtain the feature vector.
3. Compare the object feature vector with the vectors in the index file, find the closest 4 samples

The searching process is very close to the classic K-NN method, however, it is not a typical clustering problem. In a classic K-NN problem, a sample will be clustered by finding the K nearest neighbors. Here the problem is finding the k (k=4) neighbors and assume they are in a same cluster.

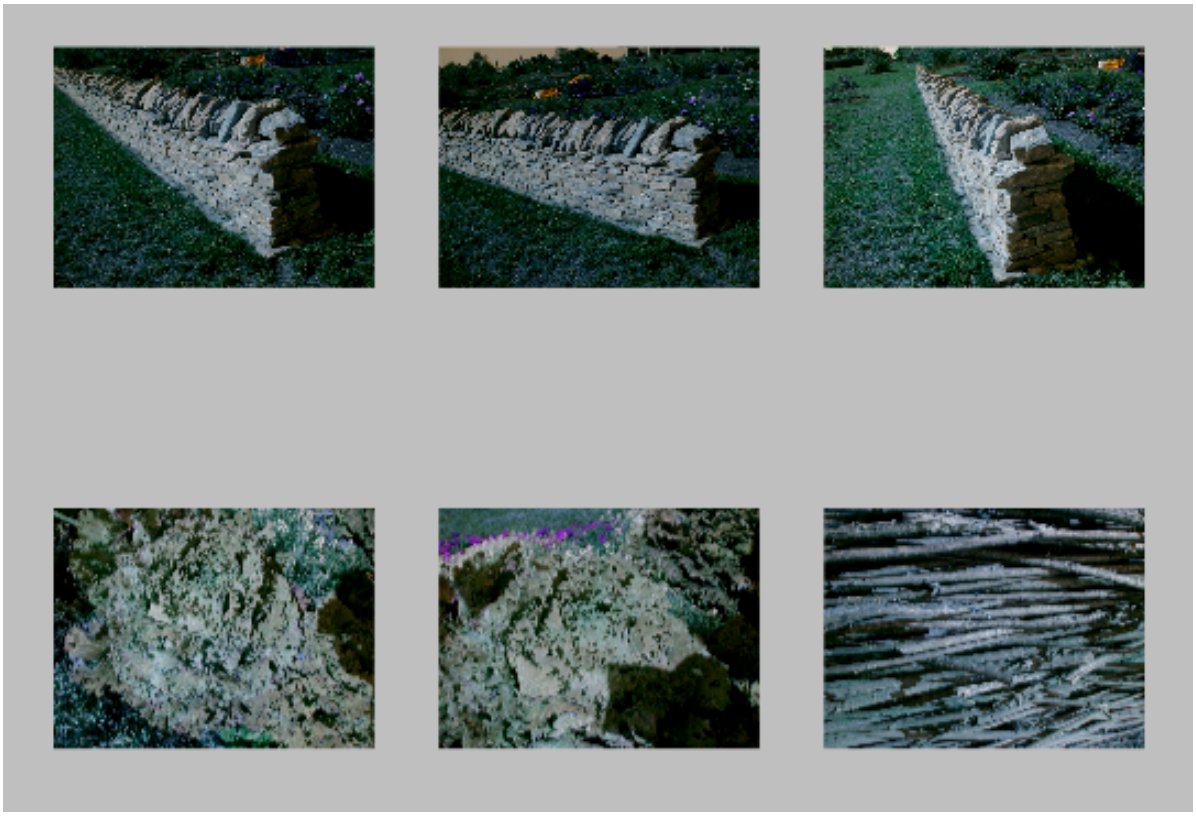


Figure 9 Searching result

To give a better demonstration, the first 6 results are selected rather 4 and some of the correct samples may appear the 5th or the 6th for a non-perfect search engine. Figure 9 is the result of the sample image. The first 3 results are correctly found out. However, the search engine failed to find the 4th image, it even did not appear till the 6th result.

Refinement

The parameter to be tuned for this model is the bin numbers of the histograms for Hue, saturation and value.
During the early stage of the implementation, a starting point was set to (2, 4, 4)
A table has been generated during the tuning process.

H	S	V	Accuracy
2	4	4	0.546
3	4	4	0.493
4	4	4	0.466
2	2	4	0.58
2	3	4	0.544
2	5	4	0.553
2	2	2	0.595
2	2	3	0.596

An assumption has been made that the effect of changing the H, S and V bin numbers are independent. For each feature, it was found out that the accuracy was decreasing with the bin number increasing and the resulted best parameter for the bin numbers are (2, 2, 3) and this gave an accuracy of 0.596.

Results

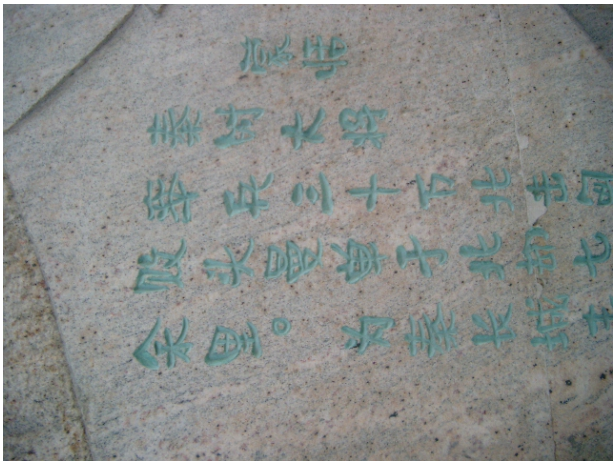
Model Evaluation and Validation

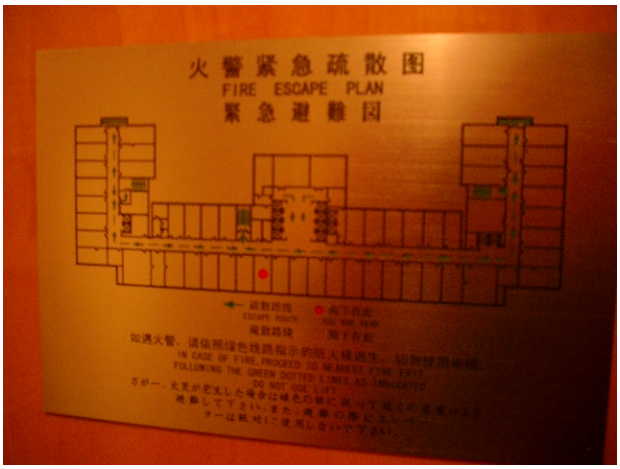
After tuning, the best parameter found for the HSV bin numbers is (2, 2, 3). Apply the model with this parameter to the testing set, the final accuracy is 0.558, which indicated that on average, at least one other image in the dataset of the same object can be found.

During the manually testing, which is randomly selected some samples to perform testing, it was found out that the model ‘s performance varies on different samples.

Below are some of the images scored 100% accuracy. Those images tend to be simply in color and dominated by a few colors with high contrast to each other. Also, many of them also follow the assumption that the object appeared in the center of the image.

Below are some examples of the images scores 0.25, which is the lowest possible score. Those pictures do not follow the assumption that the object is located in the center of the picture with a clear background. Those images are dominated by a few colors which have low contrast rate in H, S, V domain, which makes it hard for the model to distinguish them. This may explain the low performance on these pictures.





After tuning the HSV bin number parameter, the performance has increased from 0.546 to 0.596.

At this point, this model has meet the expected performance, which on average, for each searching image, at least one image can be found. However, this model only has a good performance on certain conditions, where the object is located in the center of the image and

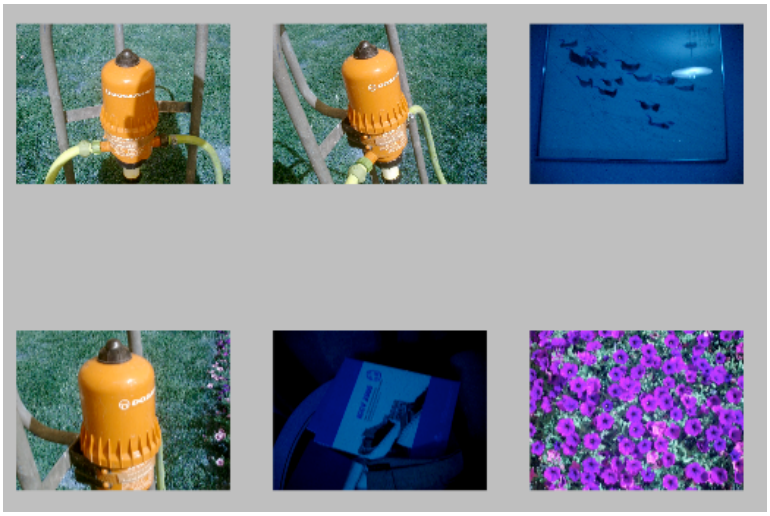
has a clear background. Also, the ideal colors in the image should be simple and have high contrast in HSV domain.

Conclusion

This project is carried out on the UKbench dataset, where for each object, there are 4 images representing that image in the dataset. A sample of the image is shown as below:

The object of the project is, for each given searching image, find out at least one image representing the same object.

For example, for image 4859, the searching result is shown below. 3 out 4 images have been found out.



Testing the model with the tuned parameter (2 ,2, 3), the final performance of the system achieved 0.54 in terms of accuracy, which indicates that on average, the object has been achieved.

Reflection and improvement

In summary, this project only went to a very simplified process of applying machine learning technics on computer vision area. Though this project only involved histogram distance and nearest neighbor method, it illustrated the basic workflow:

- Feature extraction with a proper feature extractor
- Feature comparison with a distance metrics
- Find the nearest neighbors with regards to the distances

However, it was found out that the histogram comparison is rather a naive technic for any slightly complicate task. The feature extraction process could be improved with some advanced descriptors for example, SIFT or SURF combined with some key point detectors.
When testing with one example, the searching speed is acceptable. However, during the tuning and testing progress, for each sample the program need to iterate through the whole dataset, it took around minutes to finish the iteration. The size of the dataset is only 1000. For practical use, the size of the dataset could achieve millions or even billions. Some more searching technics, for example reverse index, could be helpful to improve the searching efficiency.

Bibliography

[1] Henrik Stewénius, David Nistér, " Scalable recognition with a vocabulary tree," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), ,volume 2,* pp. 2161-2168, Jun 2006.

[2] H. Stewénius, "UKbench dataset," [Online]. Available: http://vis.uky.edu/~stewe/ukbench/.

[3] Wiki, "HSL and HSV," [Online]. Available: https://en.wikipedia.org/wiki/HSL_and_HSV. [Accessed 15 Jan 2017].

[4] A. Rosebrock, "The complete guide to building an image search engine with Python and OpenCV," 1 Dec 2014. [Online]. Available: http://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/. [Accessed 11 Jan 2017].

[5] openCV, "Histograms," [Online]. Available: http://docs.opencv.org/2.4/modules/imgproc/doc/histograms.html.