Exercise 10 - Object Recognition

Anders Nymark Christensen Anders Dahl Mads Doest
02504 - Computer Vision

April 13, 2018

The aim of this exercise is to give you a feel for how image database search can be carried out, and often is. In order for the workload of this exercise to be reasonable, you will have to make use of a code package from the internet. An outline of this exercise is, that you are given 100 images. This dataset should be split into two parts, a database of 99 images and a single image used as a query.

1. Extract SIFT features from all 100 images.
2. Find K clusters of the SIFT feastures on the 99 database images (there are typically 100.000 to 200.000 features in these 99 images), via k-means clustering.
3. Based on these K clusters, a K-bin histogram is formed for each image, denoting how many extracted sift features are closest to a given cluster. This is seen as a descriptor for the image.
4. Calculate the image descriptor for the query image and find the images from the database with the closest descriptors.

1 Extracting Sift Features

Unzip the images located in ukbench.zip, these images are provided by Henrik Stewenius et al. from the university of Kentucky, see `http://www.vis.uky.edu/~stewe/ukbench/`. This data set has been extended, and the small original version is located together with this exercise. In the version we are using, we have 100 images divided into 25 classes. For calculating the SIFT features you should use the VLfeat framework, that was used in a preavius exercise. A sample script for extracting the features is given below:

```
clear all; close all; clc;
run('../vlfeat/toolbox/vl_setup');

%%
image_count = 10;

images = zeros(image_count, 480, 640);
SIFT_features = {image_count};
SIFT_descriptors = {image_count};

for image_index = 1:image_count
    if ~mod(image_index , 10)
        fprintf('Extracting features from image %03d of %03d\r',
            image_index, image_count);
    end
    filename = sprintf('ukbench%05d.jpg', image_index - 1); %
        MATLAB 1-index to images 0-index.
    filepath = fullfile('ukbench', filename);
    im = rgb2gray(imread(filepath));
    images(image_index, :, :) = im;
    [f, d] = vl_sift(single(im));
    SIFT_features{image_index} = f;
    SIFT_descriptors{image_index} = d;
end

save('SIFTdescr.mat', 'SIFT_descriptors', 'SIFT_features', 'images
    ')

%%

load('SIFTdescr.mat');
```

Get your sample script working - test it on e.g. 10 images. Then: Extract all the features and save them for later use.

2 K-Means Clustering

We use K-means clustering to reduce the dimensionality of our data. Instead of 100.000 to 200.000 SIFT features in a 128-dimensional space, we reduce it to K clusters. It is noted that 50 clusters is very few, and most decent size systems will have considerably more. It is recommended that you use so few here for speed of computing. You can use the inbuilt function in MatLab. For more details see section '7.2.1 K-Means Clustering', in the lecture notes, or MatLabs documentation.

First try with e.g. 20 images and 50 clusters, to get your code running, then try with all the images. *Remember not to include the query image in the clustering!*

3 Constructing Image Descriptors

For each SIFT feature in each image determine which cluster it is closest to. Then - for each image - make a normalised K-bin histogram of how many SIFT features are closest to each of the K clusters. Lastly compare the distance between the histogram of the query image to the rest of the images. This should be done via a χ^2 distance, i.e.

$$dist(x, y) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i} \tag{1}$$

where the term is set to zero if $x_i + y_i = 0$.

How well does it perform? How do you evaluate this? Does the performance change, if you change the number of clusters used?