EEE3032 – Computer Vision and Pattern Recognition

Lab Worksheet 3 – Visual Search and Coursework Introduction

Miroslaw Bober (m.bober@surrey.ac.uk)

Task and Learning Objective: The purpose of today's lab is to get started with the Visual Search coursework for this module, set in lectures this week. The content of this week's lectures, particular lecture 7, are directly relevant to the coursework and this lab.

Resources: The coursework and these exercises will be performed using Matlab. Download the Matlab code for the labs, if you have not done so before, and unzip the code into a separate folder. Ensure the code folder is in the Matlab search path (File -> Set Path.. -> Add with subfolders).

Ex1: Download the Skeleton code and Dataset *(5 minutes)*

You should download the "skeleton code" which serves as a starting point for adaptation, to form the solution for your coursework. This is available on ULearn next to the coursework specification.

Unzip the code into a new folder and add it to the Matlab path, as you did for the lab code.

Download the dataset of images you will be using for the coursework and unzip it into a separate folder in your home directory. Be aware the zip is 111Mb and expands to 157Mb when unzipped. Make sure you have enough space in your home folder. You can check your home space usage by typing `quota` at the command prompt.

You can download the zip from SurreyLearn at the same location you obtain the lab code

When unzipped you will find the dataset in a folder **MSRC_ObjCategImageDatabase_v2** within your space.

Ex2: Extract Image Descriptors from the Dataset *(15 minutes)*

You will now work with the program **cvpr_computedescriptors.m** from the Skeleton code. The purpose of this program is to iterate through all of the 591 images in the dataset, and extract an image descriptor fom each image. The descriptors will each be stored in a separate file, within a folder specified in the code.

Configure the program by locating the lines:-

```
DATASET_FOLDER = 'c:/visiondemo/cwsolution/MSRC_ObjCategImageDatabase_v2';
```

and

```
OUT_FOLDER = 'c:/visiondemo/cwsolution/descriptors';
```

These specify the input folder and the output folder respectively. The input folder contains the dataset. If you unzipped this into the top level of your home folder, you should replace the string with "`~/MSRC_ObjCategImageDatabase_v2`". Recall, in UNIX the ~ character is substituted for the full path of the top level of your home folder. If you unzipped the dataset somewhere else, adjust the string accordingly.

You now need to create a folder to store the descriptors extracted by this program. Create a folder 'descriptors' in the top level of your home folder, and adjust the OUT_FOLDER line to read "`~/descriptors`". Within that folder, please also create a subfolder called `globalRGBhisto`

Now you are ready to run the **cvpr_computedescriptors** program. Run the program, and check that the output has been generated in "`~/descriptors/globalRGBhisto`" or wherever else you located the descriptors. The purpose of the coursework is that you experiment with different descriptors, and so eventually you will want to create new subfolders within `~/descriptors` and fill these with output.

The program **cvpr_computedescriptors** calls the function **extractRandom.m** with each image in turn, to compute the descriptor. This may take a few minutes.

As supplied, this function just returns random numbers instead of a real image descriptor. In Ex.5, and later during your coursework, you will change this function to return something more useful as a descriptor.

Ex3: Run a Visual Search *(10 minutes)*

The program **cvpr_visualsearch** uses the extracted image descriptors to run a visual search. As with the descriptor extraction program, it must be configured before use. Find the line:-

```
DATASET_FOLDER = 'c:/visiondemo/cwsolution/MSRC_ObjCategImageDatabase_v2';
```

and replace the string with the one you used for `DATASET_FOLDER` in **cvpr_computedescriptors.**
Now find the line:-

```
DESCRIPTOR_FOLDER = 'c:/visiondemo/cwsolution/descriptors';
```

and replace it with the line you used for `OUT_FOLDER` in **cvpr_computedescriptors.**

Run the program. It will pick an image at random and load it's descriptor as a query. It will then compare the query descriptor to each of the 591 image descriptors previously extracted. The comparison is performed by calling `cvpr_compare.`

Edit the file `cvpr_compare`.m and inspect the code. This function takes 2 descriptors as input and returns the distance between them. At the moment this distance is a random number. In Ex.4, and later in the coursework, you will improve this code to return a value computed using a sensible distance measure such as L2 norm (Euclidean distance).

If you get an error at this stage, you either set the folders up incorrectly or you haven't yet extracted the descriptors. If successful, you will be presented with thumbnails of the top 20 matching images. These will be random at this stage because the distance is just a random number (and anyway the image descriptors are just random numbers).

Ex4: Modify the Distance measure *(10 minutes)*

Edit the file `cvpr_compare`.m and inspect the code. You need to compute the Euclidean distance between F1 and F2 which are two feature descriptors. One way to do this is via Euclidean distance (L2 norm).

Subtract each element of feature F1 from feature F2.

```
x=F1-F2;
```

Square these differences:

```
x=x.^2;
```

Sum up the square differences:

```
x=sum(x);
```

Now take the square root:

```
dst=sqrt(x);
```

Ensure you understand how this corresponds to Euclidean distance as discussed in Week 2 of lectures.

Re-run the visual search program. The left-hand image on the output corresponds to the most relevant image (closest match). Because all 591 images are compared against, this will always be the query image (with a descriptor that matches the query perfectly i.e. with distance zero). However the results are still random, because the image descriptor consist entirely of random numbers.

Ex5: Modify the Descriptor Computation *(15 minutes)*

Edit **extractRandom.m** which is the function responsible for computing an image descriptor from an image.

We will modify this function to compute a 3 dimensional image descriptor, comprising the average red green and blue values of the image. This is a very crude form of global colour descriptor. Within the function, compute the average red value of the image using:-

```
red=img(:,:,1);

red=reshape(red,1,[]);

average_red=mean(red);
```

do the same for green and blue. Concatenate these three values to form a feature vector F:-

```
F=[average_red  average_green  average_blue];
```

Now re-run the descriptor extraction process, and the visual search. You should find the system is returning images with roughly the same overall colour. This is particularly evident in the images containing large expanses of green grass – re-run the search a few times to appreciate it's operation, since the query image is selected at random.

Ex6: Modify the Descriptor to a Global Colour Histogram *(ongoing)*

The results of Ex.5 show some ability to discriminate between global colour in the image, but are not very good. A better way to create a colour based visual search system is to use the Global Colour Histogram described in Lecture 7 (slides 4-6) and in the associated code handout. Integrate this code into **extractRandom.m.**

With appropriate documentation and testing you are now well on the way to fulfilling requirement 1 of the coursework.