|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | November 15, 2013 |  | | |
|  | |  | | |
| Report for Week 1  *Introduction to the basics* | | | | |
|  |  | |  | |
|  |  | | Patrick de Kok Georgios Methenitis | 5640318  ????????? |

Report for Week 1

Introduction to the basics  
Note: For this assignment, we have chosen not to use Canopy. We work directly with Python, as we have some experience with this language already. Therefor it might be possible that certain GUI elements look slightly different.

# Step A

We have been able to run the main\_script\_week1.py file without any syntax errors. This required removing the line

cd “CHANGE THIS PATH TO YOUR WORKING DIRECTORY $main\_dir/python/week1”

as this is not proper Python syntax. When the Python session is started in the same directory as where the script resides, the current working directory is already set correctly.

The data we are going to use is located in the data/objects/flowers folder, two folders up from the current working directory.

# Step B

Now we know how to locate the images, we can load them and display them to the user. As an example, .../.../data/objects/flowers/1.jpg is displayed in Figure 1. This requires the usage of matplotlib.pyplot.imshow(image) function. The function reads a 2D or 3D numpy.array object and displays it as an image.

When we count the number of elements in the first layer of the array, we obtain the number of rows the image has. Each of the first layers' elements should have another array, representing a column position at that row. All these nested arrays should have the same length, as it would be strange to have an image with fringed edges. These nested arrays most often contain either a scalar value for single-channel images, such as gray-scale or depth map images, or a triple of scalars which is appropriate for RGB, HSV and other three-channel representations.

Inspection shows that the image of Figure 1 contains 1024 rows and 684 columns. As imshow automatically interprets three-channel images as RGB and the image does seem to represent something from nature, it most probably is in the RGB color space.

**Figure 1:** Shown is the first object from the flowers data set.

# Step C

One common used image representation which is easy to compare, is the combined histogram of color channels. For each possible value per color channel, the frequency of that value is counted in the image. This value-frequency pair is stored. On the end, the frequencies are normalized, such that all frequencies sum up to 1. In this way, the histogram of an image with many pixels is very similar to that of the same image with a lower resolution. With these single channel histograms we can see easily the amount of light and dark pixels per color. Each such histogram contains 256 different values (bins). Because all color channels are important, we combine these to a single histogram of 768 bins, where the single channel histograms are concatenated.

The relative pixel value frequencies (so seeing each triple as a unique value in the image) does not generalize enough over the image. This would result in very low frequencies. If someone takes a picture from the same object from the same position, but only with a slight difference in colors (because of the moving Sun, clouds or something else), these values will be different and the histogram will be quite different.

**Figure 2:** Relative color channel frequencies of Figure 1. Bin labels 0 to 256 correspond to the relative frequency of the red values 0 to 256 in the image, labels 256 to 512 correspond to the relative frequency of the green values 0 to 256, and labels 512 to 768 correspond to the relative frequency of blue values 0 to 256.

# Step D

To compare the image representations, one can apply different distance measures. We have implemented five different distance measures. For two histograms with bins, we have the following distance measures:

* Euclidean distance: where [[1]](#footnote-2)
* distance:where
* distance:
* Histogram intersection distance:
* Hellinger distance:

Note that for lower outcomes indicate thatare more similar (“more nearby” in the space of histograms with the given distance measure as inner product), but for more similar objects have higher outcomes. Actually, are not distance measures (one rule to be a true distance measure is that the distance of a vector to itself is ).are *similarity measures*. Any distance measures can easily be transformed to a similarity measure:

In our code, we have transformed each distance measure in a similarity measure.

# Step E

Ranking the images for similarity to a chosen image is now very easy. First, we have to compute the combined histogram of each image in the data set. Then, compute the similarity of each image with the chosen image. Sort all images in descending order with respect to their similarity with the chosen image. The first image on your list is most similar to the provided query image.

This has been done for each similarity measure with query images 5.jpg, 10.jpg and 15.jpg. The results summarized in Table 1. Corresponding images are included in Appendix A. More text on results?

| **Query id** | **Sim. meas.** | **Rank 1** | **Rank 2** | **Rank 3** | **Rank 4** | **Rank 5** |
| --- | --- | --- | --- | --- | --- | --- |
| 5 |  | 5 | 53 | 35 | 37 | 54 |
| 5 |  | 5 | 53 | 35 | 37 | 54 |
| 5 |  | 5 | 37 | 54 | 57 | 56 |
| 5 |  | 5 | 56 | 37 | 54 | 51 |
| 5 |  | 5 | 37 | 56 | 54 | 27 |
| 10 |  | 10 | 60 | 17 | 2 | 54 |
| 10 |  | 10 | 60 | 17 | 2 | 54 |
| 10 |  | 10 | 60 | 17 | 2 | 35 |
| 10 |  | 10 | 60 | 17 | 2 | 43 |
| 10 |  | 10 | 60 | 17 | 43 | 30 |
| 15 |  | 15 | 27 | 12 | 18 | 31 |
| 15 |  | 15 | 27 | 12 | 18 | 31 |
| 15 |  | 15 | 27 | 18 | 31 | 19 |
| 15 |  | 15 | 27 | 18 | 20 | 31 |
| 15 |  | 15 | 27 | 12 | 54 | 18 |

**Table 1:** Overview of image ids similar to the query under the provided similarity measure.

# Step F

More text...

# Appendix A: Similar images

**Figure A.1: **Similar images to 5 under the Euclidean and l2 measure.

**Figure A.2:** Similar images to 5 under chi2 measure.

**Figure A.3:** Similar images to 5 under histogram measure.

**Figure A.4:** Similar images to 5 under Hellinger measure.

**Figure A.5:** Similar images to 10 under Euclidean and l2 measure.

**Figure A.6:** Similar images to 10 under chi2 measure.

**Figure A.7:** Similar images to 10 under histogram measure.

**Figure A.8:** Similar images to 10 under Hellinger measure.

**Figure A.9:** Similar images to 15 under Euclidean and l2 measure.

**Figure A.10:** Similar images to 15 under chi2 measure.

**Figure A.11:** Similar images to 15 under histogram measure.

**Figure A.12:** Similar images to 15 under Hellinger measure.

1. This extra normalizing constant is needed because …?! [↑](#footnote-ref-2)