COMPUTER VISION LABORATORY

SESSION 8: Image Matching and Image Retrieval

June 17, 2020

Arlotta Andrea 4089306
Cantoni Francesca 4698289

Contents

Chapter 1

Introduction

Image matching allows, by different approaches, to get information about the similarities between two (or even more) images. Image is a powerful information medium: few other information methods are so direct and meaningful; however the information is **easily understandable for human beings** but this is not true for computers, where visual interpretation turns out too complex and intricate.

Data processing methods comes to help to simplify this procedure by characterizing the image with its interesting points or regions.

1.1 Local and Global Features

We can divide image matching processes into two big families:

- image matching algorithms based on **global image descriptors**: these are the element on which will be weight the similarity (e.g.: brightness/color mean and variance, color histograms, texture signatures, bag-of-features)
- algorithms that analyze the similarity between two images relying on **local features** that make up the image

Depending on the subject of our analysis and the scenarios that will (probably) surround it, our choice will fall to the one that proper fits the task.

Chapter 2

Local Feature Matching

This section will focus on the analysis and comparison of two different algorithms that are commonly used to extract functional parameters and consequently are useful for evaluating the affinity between two given images.

Firstly will be presented the SIFT and the NCC methods, employed to describe the images, and then it will be described the Affinity matrix for both described.

Secondly the relevant parameters that weight on the final results will be described.

Thirdly the Matlab code employed to test the two techniques will be presented.

Lastly the results of the code will be commented focusing on the similarities and differences between the two methods.

2.1 Employed Algorithms

2.1.1 SIFT

This method was patented in Canada by the University of British Columbia in 1999 and its main objective is to detect and describe local features in images.

In brief, the 128 dimensional SIFT descriptor is a histogram of responses to oriented gradient filters. By collecting the dominant orientation features in each state of the scale-space (starting form the higher scale value and then decreasing) it is possible to characterize them with a gradient orientation histogram (based on region around the key point).

The features extracted via Scale-invariant feature transform are associated to a data vector that describes them, maintaining their insensitive to geometric distortion.
Since the description of the feature is an histogram, it is possible to evaluate the affinity of two points by observing the intersection area.

2.1.2 NCC

This Algorithm is employed to measure the correlation of information between local windows surrounding the feature points.

The well-known function is:

$$NCC(f,t) = \frac{1}{n^2} \sum_{x,y=-n/2}^{+n/2} \frac{(f(x,y)-\overline{f})\ (t(x,y)-\overline{t})}{\sigma_f\ \sigma_t} \tag{2.1}$$

where $\overline{f}$ and $\overline{t}$ are the mean between the value of the patch, σ_f and σ_t are the standard deviations and n is the dimension of the side of the square patch.

2.2 Affinity Matrix

After this process the features are ready to be compared with other data structures: this procedure uses the Affinity matrix **A**, that is an essential statistical technique used to organize the mutual similarities between the two sets of feature points.
The affinity Matrix A has in each position ij the similarity value of the feature point P_i and P_j.

Affinity matrix A facilitates the detection of matches, in other words helps to check which are the couples of feature points with higher similarity value. To felicitate the analysis similarity should have values in a predefined compact range, e.g. $[0,1]$. As mentioned before there are different methods to define the similarity between two feature points, in this Lab we will focus on:

* **NNC Method + Euclidean patches distances:** each element of A shall be:

$$A(i,j) = E(i,j) \times \frac{1}{2}(NCC(f,t)-1) \tag{2.2}$$

$$E(i,j) = e^{\frac{\left\|p_i^2 - p_i^2\right\|^2}{2\sigma^2}} \tag{2.3}$$

where $E(i,j)$ is a Gaussian kernel that gives information about how spatially close two points are.

* **SIFT Method:** each element of A shall be:

$$A(i,j) = H(i,j) \tag{2.4}$$

where $H(i,j)$ is the intersection between two SIFT histograms.

2.3 Relevant Parameters

The functions used in this lab relies on some characteristic parameters which burden on the global tolerance threshold of the match algorithm.

* **DELTA:** the square patch of the NCC method has side equals to 2*δ+1.
* **SIGMA E:** tuning parameter σ of the Gaussian kernel. Basically if σ is high, $E(x,x')$ will be close to 1 for any x, x'. If σ is low, a slight distance from x to x' will lead to $E(x,x')$ close to 0.
* **SIGMA SIFT:** tuning parameter of the *radial basis function* of the SIFT method.
* **Similarity threshold:** this parameter states the minimum acceptable value in matrix A for which two feature points can be considered similar.

2.4 Matlab Code

The proposed code allows the user the tuning of the relevant parameters directly from the main file *Labo8_part1*. The only different from the initial code is that the function *findMatches(...)* now take as input also *DELTA*, *SIGMA E*, *SIGMA SIFT*, *similarity threshold*.

2.5 Results and Comments

The collected results of this Lab session will be now exposed and commented, placing emphasis on the sigma tuning and the main peculiarity and differences of *SIFT* and *NCC* methods.

The template images employed to test the script are the following[1]

Figure 2.1: Monster.jpg

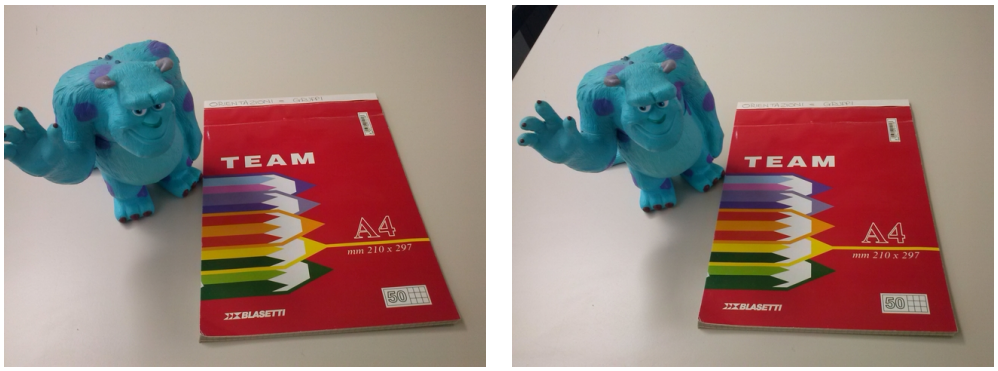

Figure 2.2: Ex01_01.jpg (left) and Ex01_02.jpg (right)

[1]Note that the images have been re-sized by 0.7 before the evaluation of the affinity matrix.

2.5.1 *Delta*

Patch value indicates how big is the kernel employed to check the correlation between two images. We execute NNC image matching maintaining *SIGMA E* equals to 1 and *similarity threshold* equals to 0.8 and by varying delta in that way:

* δ=1 $\longrightarrow$ # features=90
* δ=2 $\longrightarrow$ # features=114
* δ=4 $\longrightarrow$ # features=97
* δ=6 $\longrightarrow$ # features=97

A reasonable choice could be $\delta = 2$ due to the fact that the patch is not too big and the selection is not so strict.

2.5.2 *SIGMA E and similarity threshold:*

The code has been executed multiple times varying the relevant parameters that characterize the affinity value. Affinity has been estimated for *Ex01_01.jpg* and *Ex01_02.jpg* and δ has been set equals to 2.

Figure 2.4 shows clearly that, by lowering the affinity threshold, the number of detected matches increases. Moreover we can see that SIGMA E is inversely proportional to the number of detected matches: the more SIGMA E is high the more the selection is rigorous. Is it possible to observe a counter-trend behaviour when SIGMA E is 1.

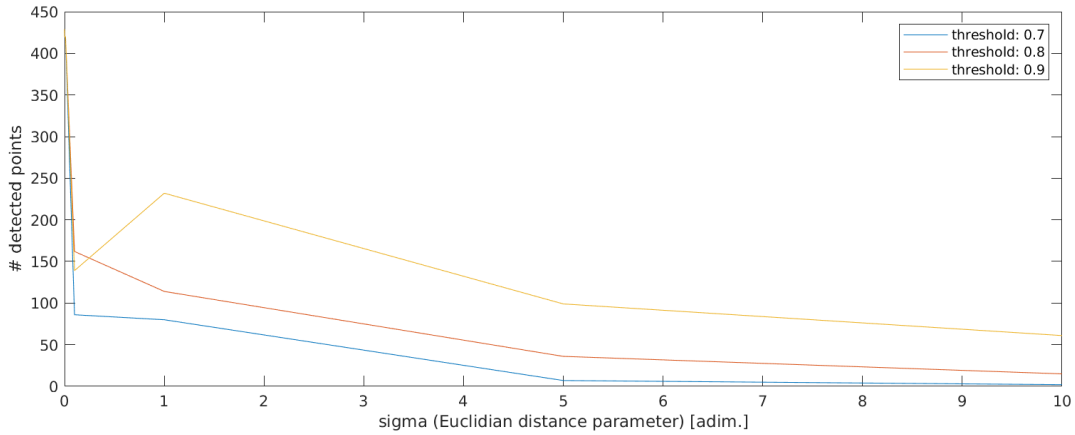

Figure 2.3: Number of the detected matches obtained by varying σ_E and affinity threshold

2.5.3 *Sigma SIFT*

Here again **Figure 2.5** we can observe that by lowering the affinity threshold the number of detected matches increase. For a better analysis a good choice is to use logarithm scale: detected matched does not decrease to zero if we choose high values of SIGMA E but, on the contrary to the previous method, it stabilizes on a certain level depending on the chosen threshold. We can say that a reasonable value for sigma can be chosen between 10^{-3} and 10^{0}.

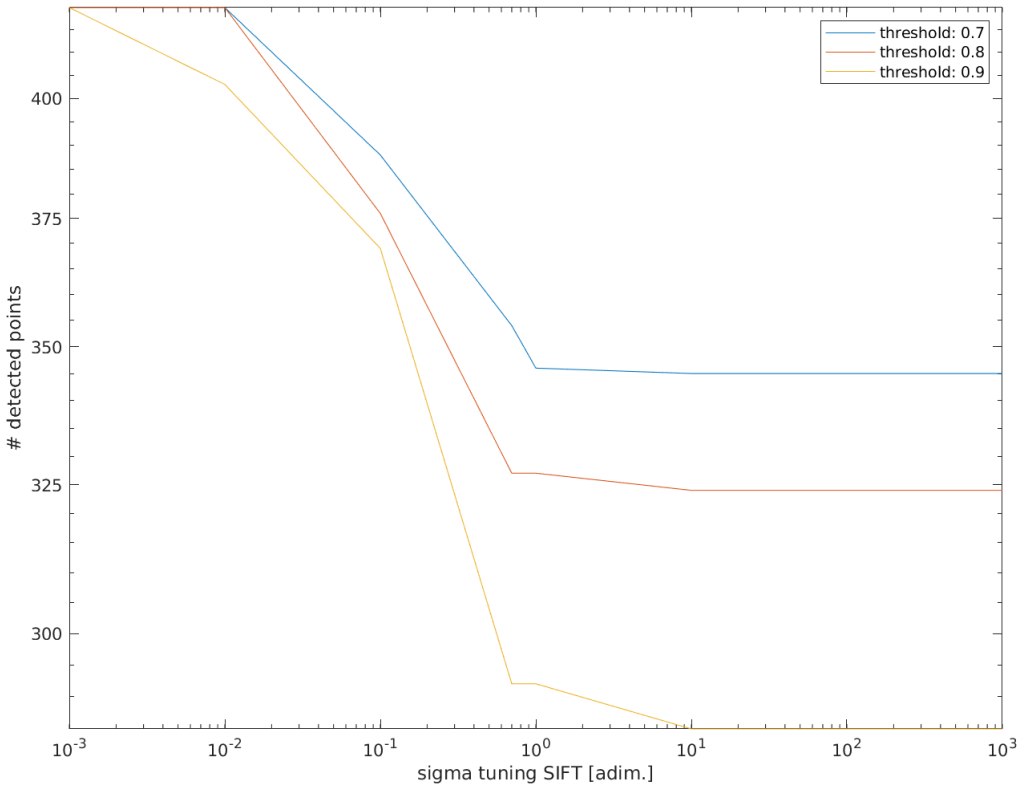

Figure 2.4: Number of the detected matches to vary of σ_{SIFT} and the affinity threshold

2.5.4 *Comparison between the two methods*

The two methods have both positive and negative aspects.

For this experience we choose the following values: **threshold =** 0.8**, SIGMA E =** 0.1**, SIGMA SIFT =** 1**, delta =** 2.

Figure 2.6 and **Figure 2.7** show the detection of match points between *Monster.jpg* and *Ex01_01.jpg* firstly via *SIFT* and secondly *NCC* methods.
It's obvious that SIFT executes a more correct and precise detection, in fact the notebook's feature does not match with the Monster's one.

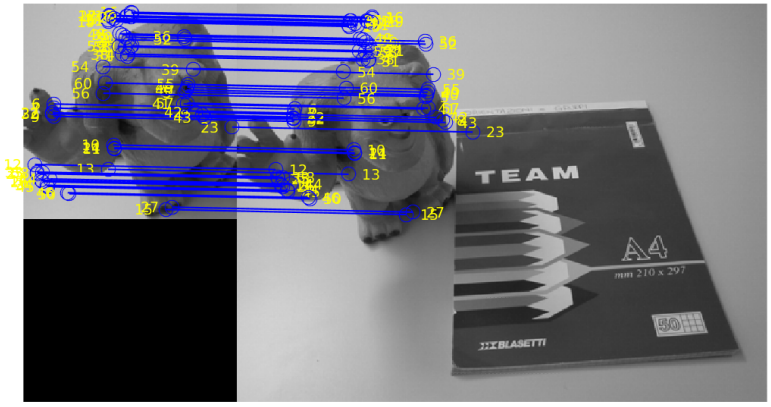

Figure 2.5: Detected matched via SIFT algorithm

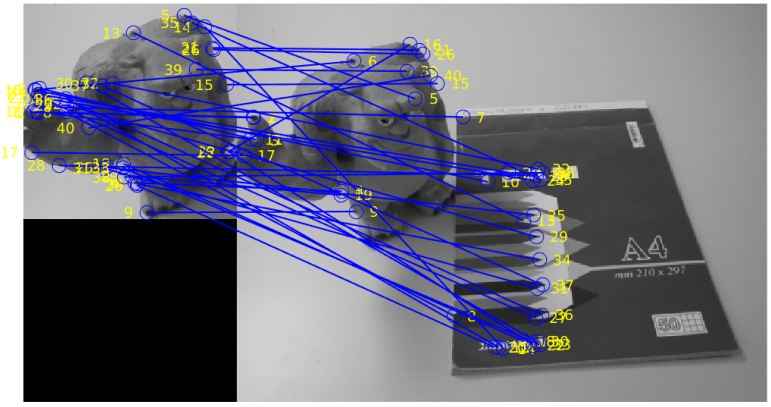

Figure 2.6: Detected matched via NCC algorithm

Another positive aspect of the SIFT algorithm is its speed with respect to the NNC method, in fact *findMatches()* function takes 7.738 s to be executed for NNC, while 0.471 s only for SIFT (this data has been generated using performance time tool).

Chapter 3

Image retrieval

This second section focuses on image retrieval using a representation based on SIFT descriptors and the bag-of-keypoints paradigm. Bag of Features (BoF) methods are characterized by the use of an orderless collection of image features. BoF representation can be seen as definition of a visual vocabulary, which elements are the most meaningful and informative features of a set of images. Instead of words, this dictionary is based on features descriptors: in this case we will adopt SIFT method to characterize the elements. A vocabulary can be build by Vector Quantization (Clustering) of a features data-set (e.g.: a structure of feature points extracted by a gallery).

The choice of the vocabulary size is an important specific that may lead, if too small or even too big, to the construction of an non optimal visual dictionary; on the other hand the appropriateness of the data set is also crucial. The choice of these two aspects are strictly related to the requested task.

In summary, given a new image, features are detected and assigned to their nearest matching terms, from the visual vocabulary, so that a BoF term vector is obtained. The BoF term vector is a compact representation of an image which discards large-scale spatial information and the relative locations, scales, and orientations of the features.

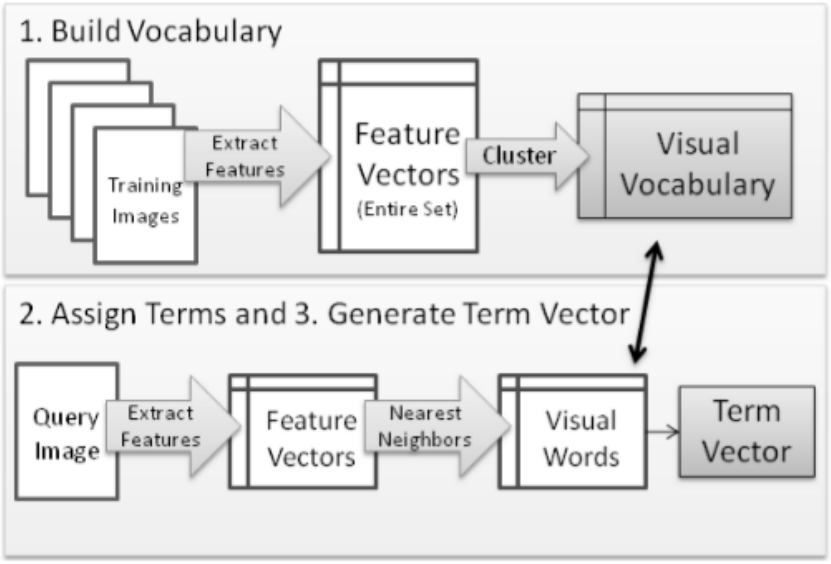

Figure 3.1: Process for Bag of Features Image Representation

3.1 Matlab code

The implemented script:

* Loads images
* Extracts features via SIFT
* Loads pre-computed Dictionary, if is not possible to load it generates new dictionary by K-means clustering the features detected previously. Note that it is possible to choose the K value which defines the size of the employed vocabulary.
* Creates a vector term for each query image and then normalizes the histogram of the quantized features
* Evaluate affinity between the query images' vector term and data-set images, relying on the function *vl_alldist2(...)* which returns a distance value. Note that it is possible to choose the METRIC of the procedure:

  L0: sum $(X \neq Y)$

  L0: sum $|X - Y|$

  L2: sum $(X - Y)^2$

* Draws up a similarity ranking list and returns the computed distance

3.2 Comments and results

The given code has been executed several times, trying to understand the main differences and the main similarities detected by varying the dictionary size.

The standard output (see **Figure 3.2**) of the code is a sequence of images: the top left one is the query image, while the remaining nine are taken from the data-set and they are ordered according to a gradient of similarity.

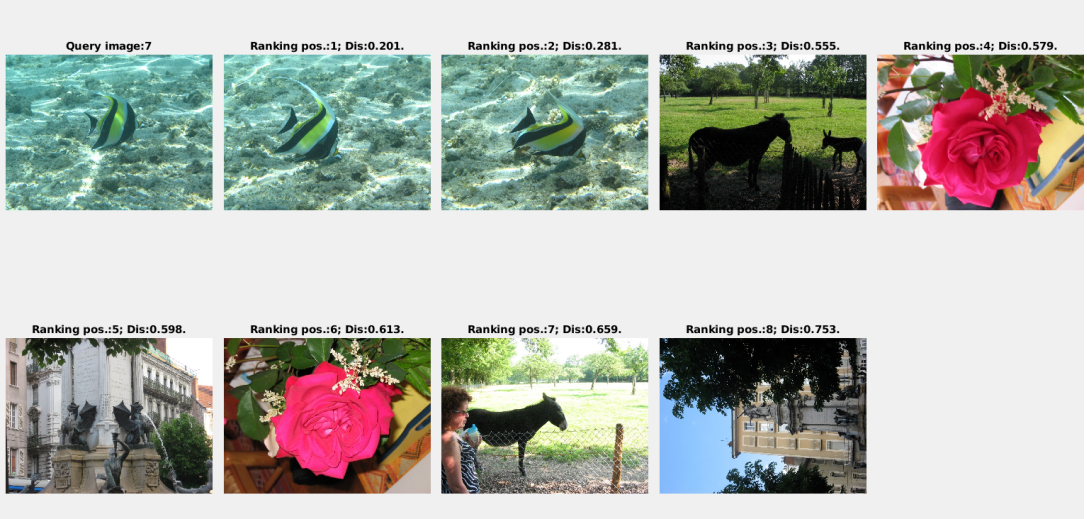

Figure 3.2: Standard output of the script

Now will be listed the main comments about this Lab experience:

* **Affinity distance:** it has been noted that the distance between query images and data-set image is directly proportional to the dictionary size, e.g. the distances computed considering a dictionary with 1000 elements are approximately 5 times those computed employing a dictionary with 30 elements.

* **Larger dictionaries may be more effective:** sometimes well described images can be correctly processed also with poor dictionaries.
  Figure 3.3 shows that the query image n.3 is correctly processed by 30 and 1000 elements both; this is probably due to the fact that the query

image is well characterized, in the sense that its term vector has elements that differs from all the other images of the data-set.

Figure 3.3: Flower one (left) and flower two (right)

On the other hand **Figure 3.4** makes clear that sometimes larger dictionaries are more powerful; here, by employing a 30-elements dictionary, we have that the correct image is ranked only fourth, while with a 1000-elements dictionary is ranked first.

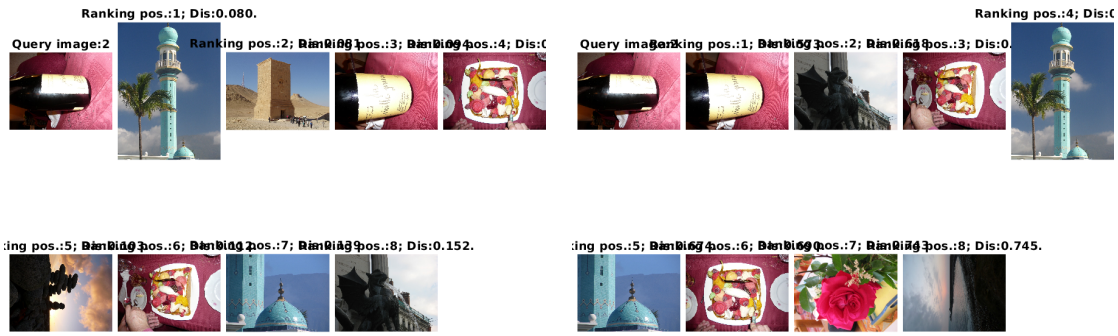

Figure 3.4: Flower one (left) and flower two (right)

* **Light intensity shifts affect SIFT effectiveness:** SIFT technique is employed to detect main local features in images. Is a fact that this method is not robust against brightness variations.

Figure 3.5: Image retrieving with 1000-elements dictionary

This can be shown in **Figure 3.5**, where beneath the usage of a 1000-elements dictionary, the correct data-set image is not ranked in first position. Moreover the third classified is really similar to the query image if it were not for completely different lighting.