Computer Vision Laboratory Report n.7

# IMAGE MATCHING AND RETRIEVAL

June 14, 2019

Claudio Curti ID: 4216203
Nicola De Carli ID: 4198668
Alberto Ghiotto ID: 4225586
University of Genoa

# Contents

# Chapter 1

# Introduction

## 1.1 Defining the objective

The goal of this laboratory is to get familiar to different strategies for image matching and image retrieval by practicing with some pre-made MATLAB code. With the purpose of evaluating the results when modifying the key parameters the code was run on different image pairs and on the provided gallery.

# Chapter 2

# Implementation and Results

The provided code is organized in two different folders with a main script which calls its relative functions. The first part focuses on image matching whereas the second focuses on image retrieval.

## 2.1 Image Matching

- **Labo8_part1**: which loads the image pairs and then finds and displays the located matches.
- **findMatches**: which locates the matches with the method specified as a parameter, whether NCC (considering euclidean distance between positions and patches similarity) or SIFT (euclidean distance between SIFT vectors).
- **similarity**: which computes the similarities between a given pair of features according to the different inputs parameters.
- **show_matches**: which visualizes the two images and the list of matched features.

## 2.2 Image Retrieval

- **Labo8_part2**: which loads the images, extract the features for dictionary learning, constructs the dictionary, constructs the Bag of Features, perform the image retrieval and ranks the images.
- **load_images**: which simply loads the set of image descriptors.
- **showranking**: which simply displays the image ranking.

## 2.3 Results

This time the code was entirely provided as the starting material, hence, the required work was to analyze it by focusing on two different experiments:

- **Analysis 1** : which requires to focus on image matching and evaluate the effects of varying different parameters.
- **Analysis 2** : which requires to focus on image retrieval and evaluate the performance when varying the size of the dictionary.

### 2.3.1 Analysis 1 - Image matching

This task has been performed with two different strategies, in both cases at first, features are extracted using multiscale Harris, then a similarity measure is performed:
in one case by mixing the normalized cross correlation (NCC) applied to image patches of size $2 * delta + 1$ and a measure of the euclidean distance of the detected feature points, in the other by comparing the distance of the SIFT descriptors (which are 128-dimensional vectors) of the found features. In general, it is possible to notice that the correspondences provided by SIFT are more accurate than those provided by NCC.

#### 2.3.1.1 NCC

The parameters to be changed in this case were sigma, which controls the maximum distance between the positions of the features we are willing to consider, and the minimum threshold of the correspondence value in the affinity matrix to consider a correspondence as good. By increasing the sigma value the correspondences seem to decrease, this could happen since less one-to-one matches happen. When increasing the threshold, the correspondences with low similarity scores are discarded and therefore some incorrect correspondences disappear.

The table below refers each figure to its parameters configuration.

| FIGURE | METHOD | PARAMETERS |
|---|---|---|
| Figure 2.1 | NCC | $\sigma = 0.001$ $Th = 0.75$ |
| Figure 2.2 | NCC | $\sigma = 0.01$ $Th = 0.75$ |
| Figure 2.3 | NCC | $\sigma = 0.01$ $Th = 0$ |
| Figure 2.4 | NCC | $\sigma = 0.1$ $Th = 0.75$ |
| Figure 2.5 | NCC | $\sigma = 0.1$ $Th = 0$ |

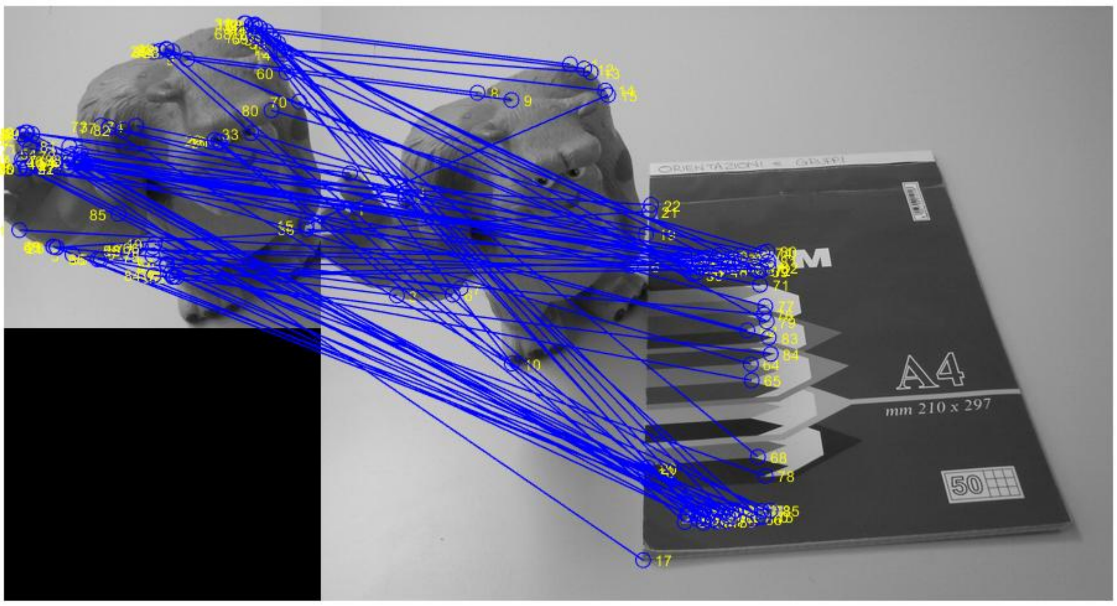

Figure 2.1: 85 matches found

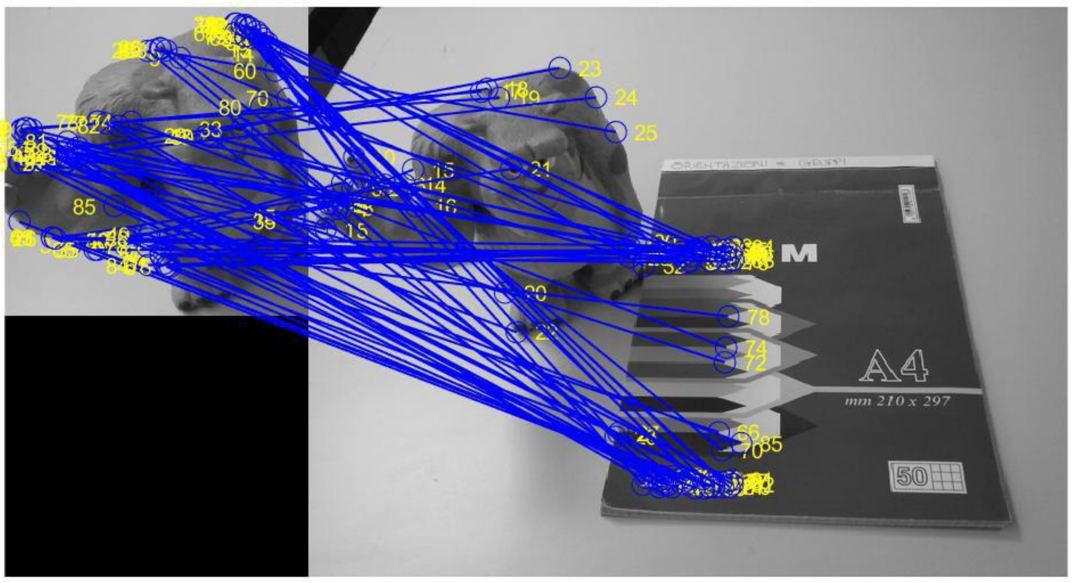

Figure 2.2: 85 matches found

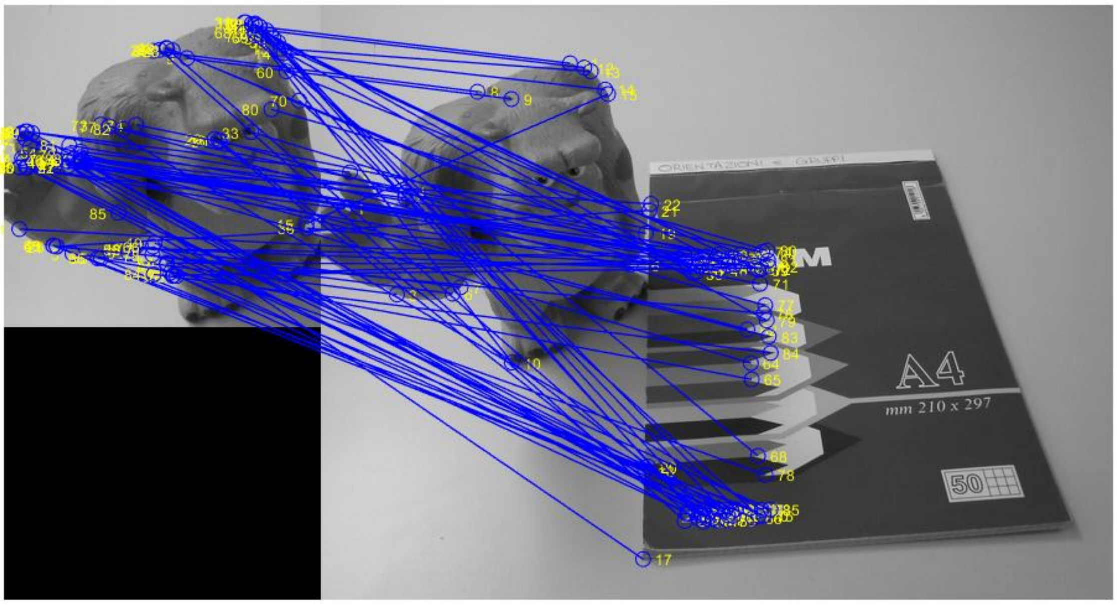

Figure 2.3: 85 matches found

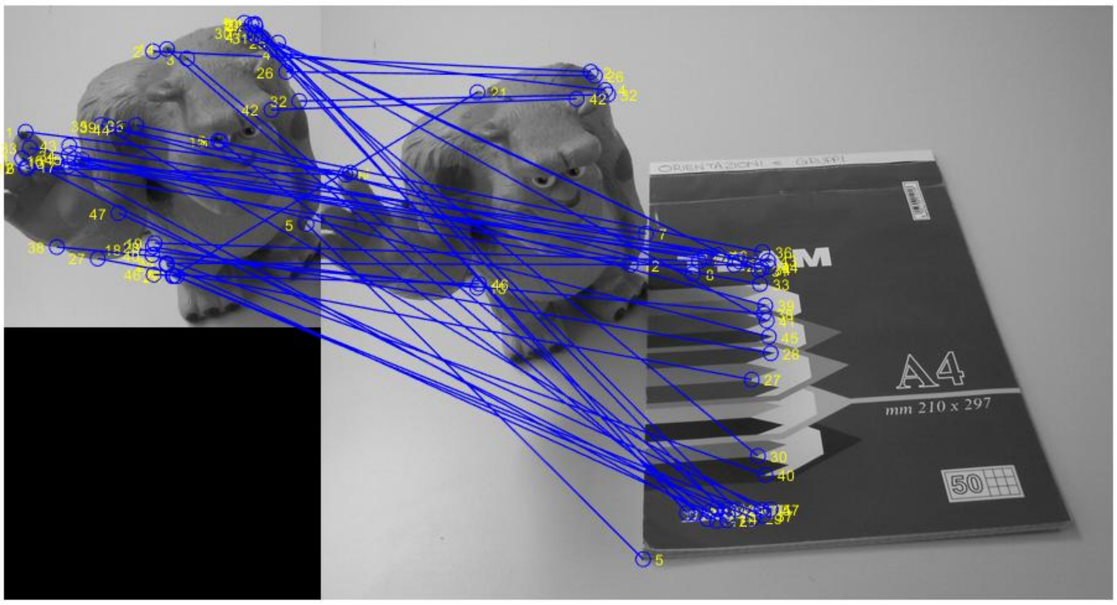

Figure 2.4: 47 matches found

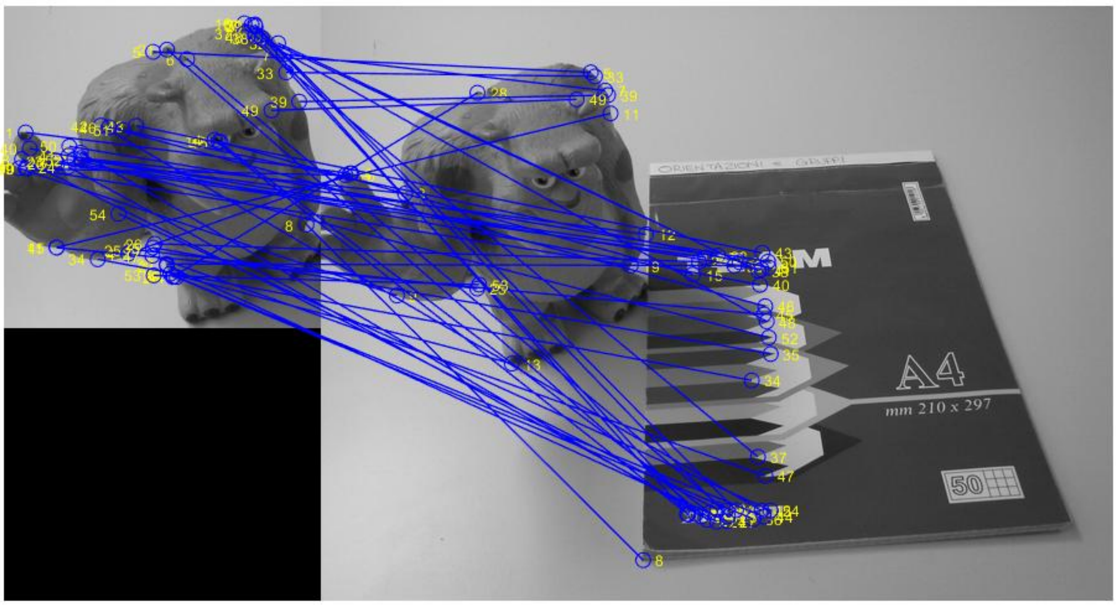

Figure 2.5: 54 matches found

#### 2.3.1.2 SIFT

The parameters to be changed in this part were sigma, which in this case has a different meaning with respect to the previous method and a less immediate interpretation, and the minimum threshold of the correspondence value in the affinity matrix to consider a correspondence as good, as before. In this case, when increasing the value of sigma, the number of correspondences decreases, in particular, those that disappear seem to be false correspondences.

The table below refers each figure to its parameters configuration, the threshold is fixed at 0.75.

| FIGURE | METHOD | PARAMETERS |
|---|---|---|
| Figure 2.6 | SIFT | $\sigma = 0.1$ |
| Figure 2.7 | SIFT | $\sigma = 0.5$ |
| Figure 2.8 | SIFT | $\sigma = 1$ |

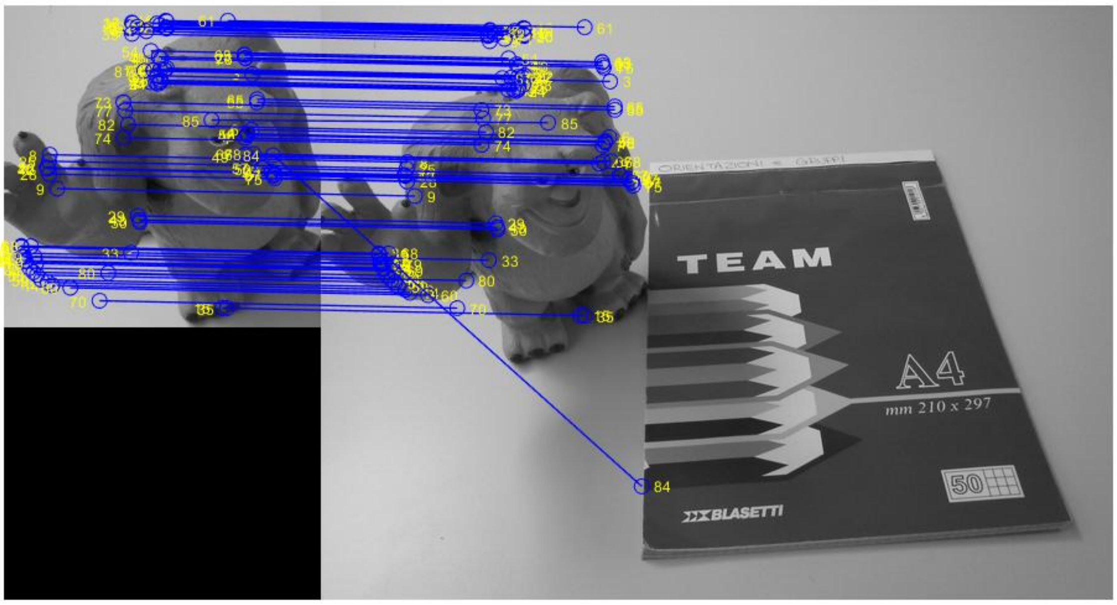

Figure 2.6: 85 matches found

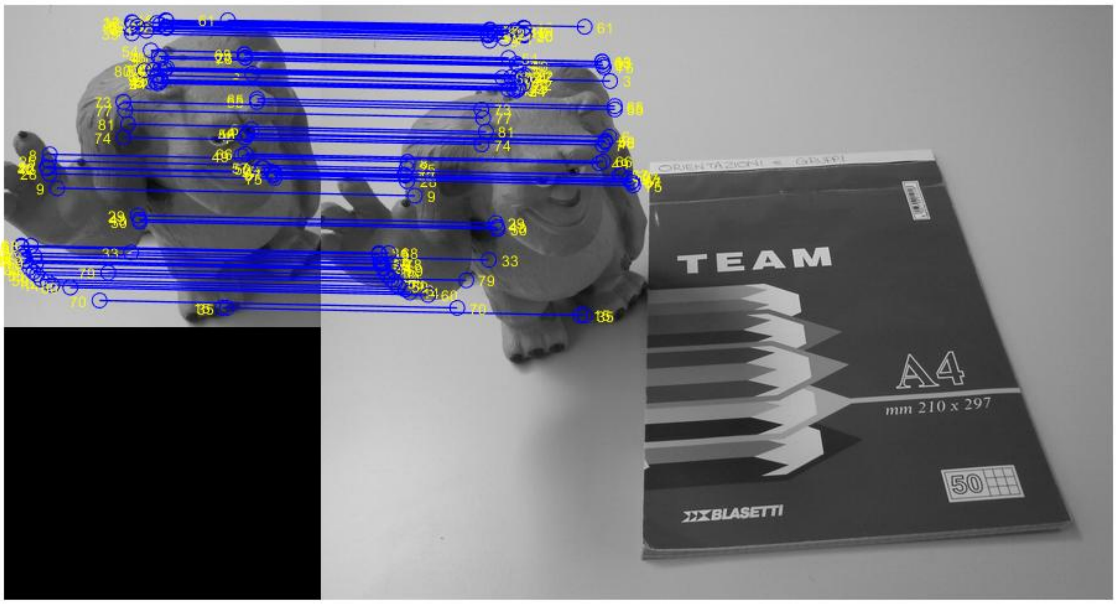

Figure 2.7: 82 matches found

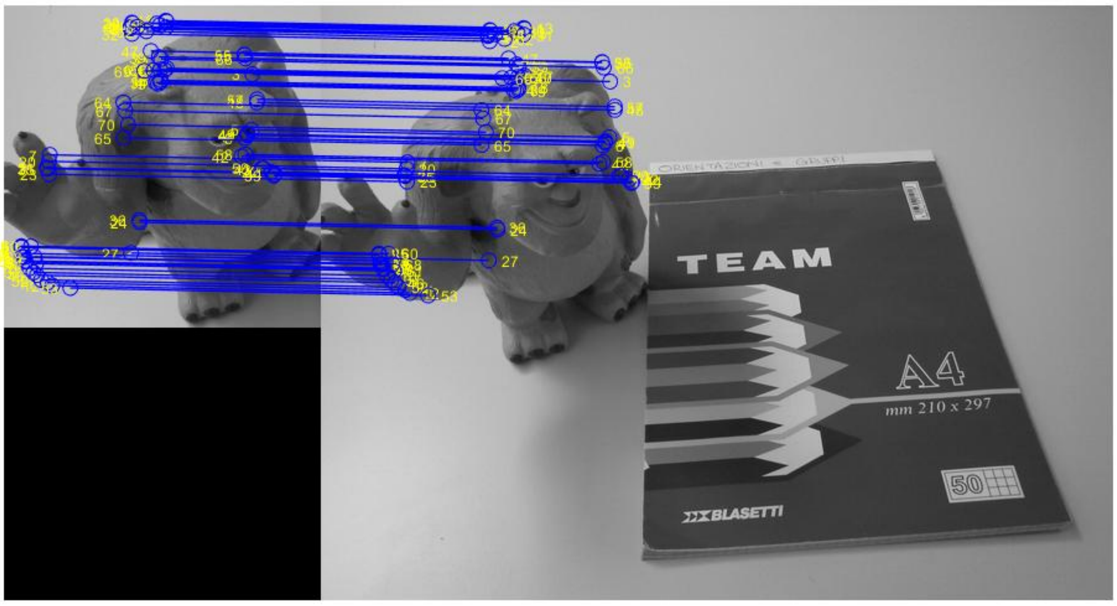

Figure 2.8: 70 matches found

### 2.3.2 Analysis 2 - Image retrieval

This task has been performed by extracting features from each image and then by building the bag of words, which are sorts of histograms used to measure the similarity between images, for each image by comparing the SIFT description of the features to those of the codewords of a codebook provided in the files. Then, a ranking based on these measurements is built. As requested from the text, the goal of this part was to explain the differences in the results when changing the size of the dictionary within the possible values (30, 100, 200, 300, 500, 1000). Given a query image, the code estimates a rank of the images which are more similar to the one provided. Therefore, the number of codewords used (the size of the dictionary) influences the precision of the image description as a sort of resolution of the image description. Thus, by increasing its size, the precision of the ranking should improve.

The table below refers each figure to its image query and threshold configuration.

| FIGURE | PARAMETERS |
|---|---|
| Figure 2.9 | Query 1 $Th = 30$ |
| Figure 2.10 | Query 1 $Th = 300$ |
| Figure 2.11 | Query 1 $Th = 1000$ |
| Figure 2.12 | Query 2 $Th = 30$ |
| Figure 2.13 | Query 2 $Th = 300$ |
| Figure 2.14 | Query 2 $Th = 1000$ |

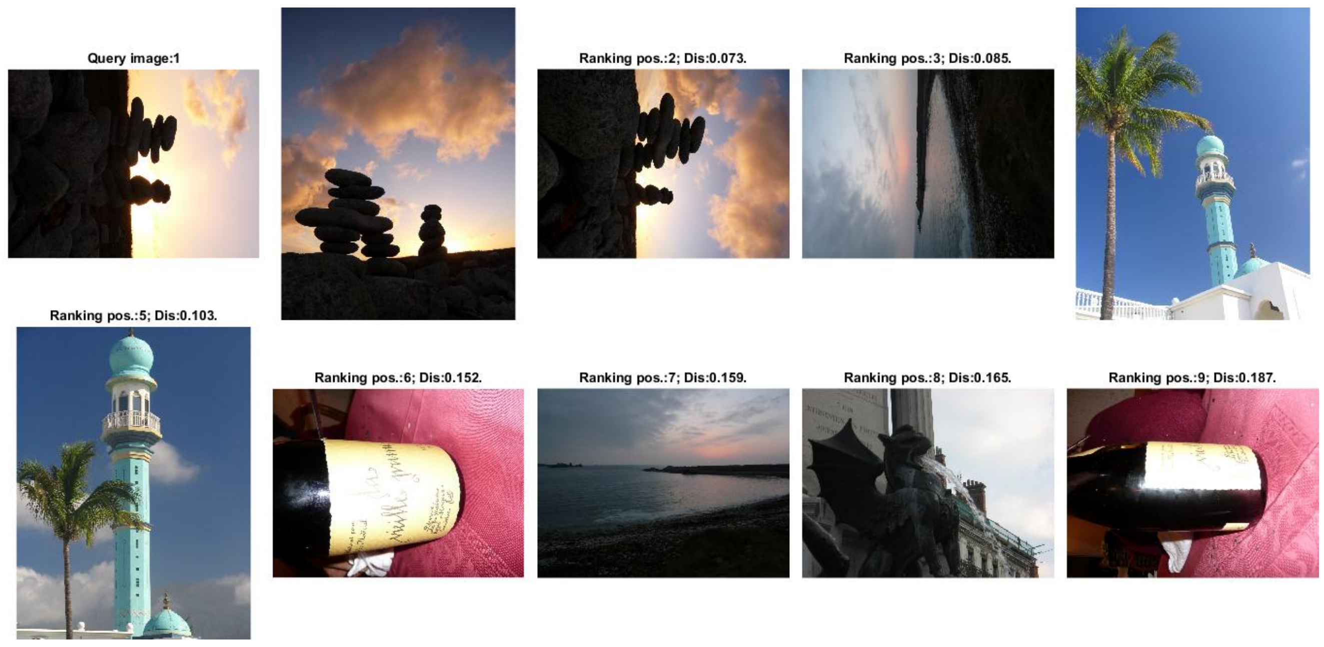
Query image:1
Ranking pos.:2; Dis:0.073.
Ranking pos.:3; Dis:0.085.
Ranking pos.:5; Dis:0.103.
Ranking pos.:6; Dis:0.152.
Ranking pos.:7; Dis:0.159.
Ranking pos.:8; Dis:0.165.
Ranking pos.:9; Dis:0.187.

Figure 2.9: Image query 1 $Th = 30$

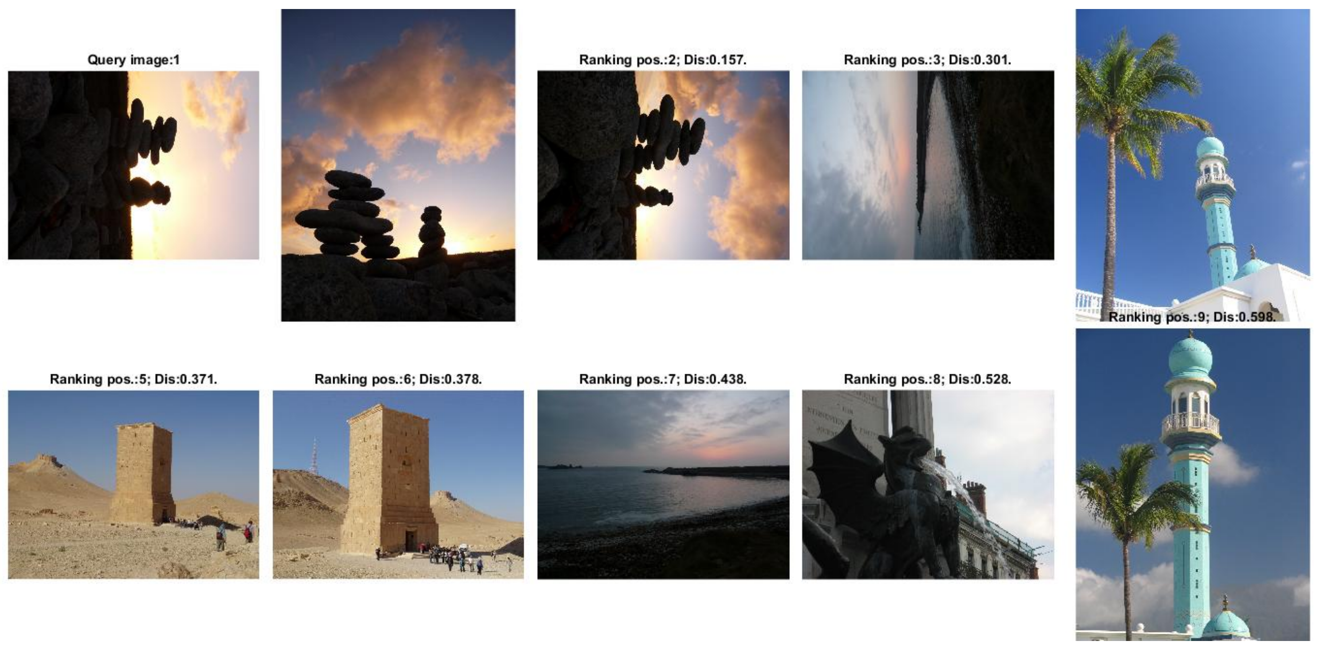
Query image:1
Ranking pos.:2; Dis:0.157.
Ranking pos.:3; Dis:0.301.
Ranking pos.:9; Dis:0.598.
Ranking pos.:5; Dis:0.371.
Ranking pos.:6; Dis:0.378.
Ranking pos.:7; Dis:0.438.
Ranking pos.:8; Dis:0.528.

Figure 2.10: Image query 1 $Th = 300$

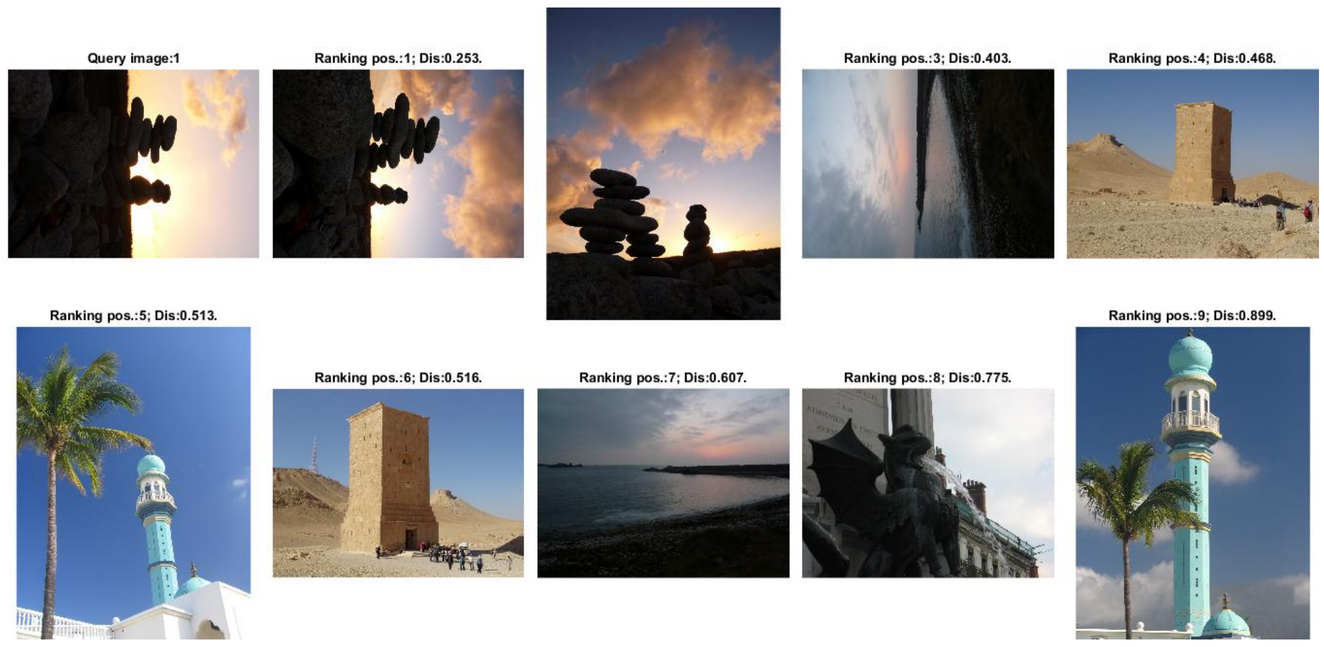
Query image:1
Ranking pos.:1; Dis:0.253.
Ranking pos.:3; Dis:0.403.
Ranking pos.:4; Dis:0.468.
Ranking pos.:5; Dis:0.513.
Ranking pos.:6; Dis:0.516.
Ranking pos.:7; Dis:0.607.
Ranking pos.:8; Dis:0.775.
Ranking pos.:9; Dis:0.899.

Figure 2.11: Image query 1 $Th = 1000$

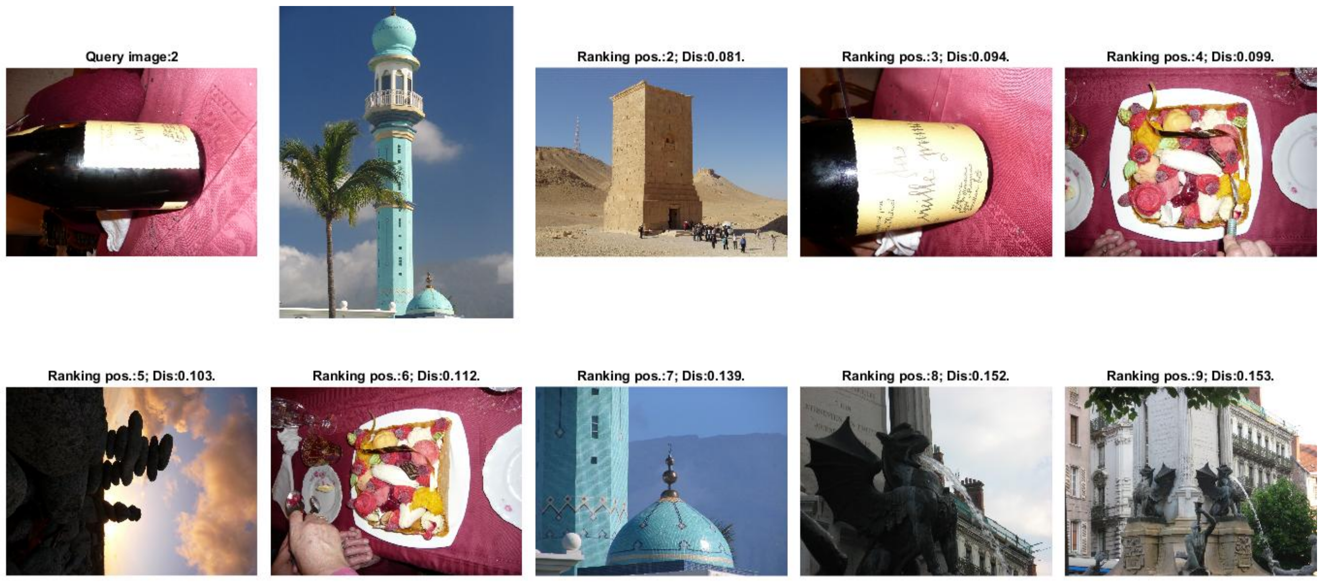
Query image:2
Ranking pos.:2; Dis:0.081.
Ranking pos.:3; Dis:0.094.
Ranking pos.:4; Dis:0.099.
Ranking pos.:5; Dis:0.103.
Ranking pos.:6; Dis:0.112.
Ranking pos.:7; Dis:0.139.
Ranking pos.:8; Dis:0.152.
Ranking pos.:9; Dis:0.153.

Figure 2.12: Image query 2 $Th = 30$

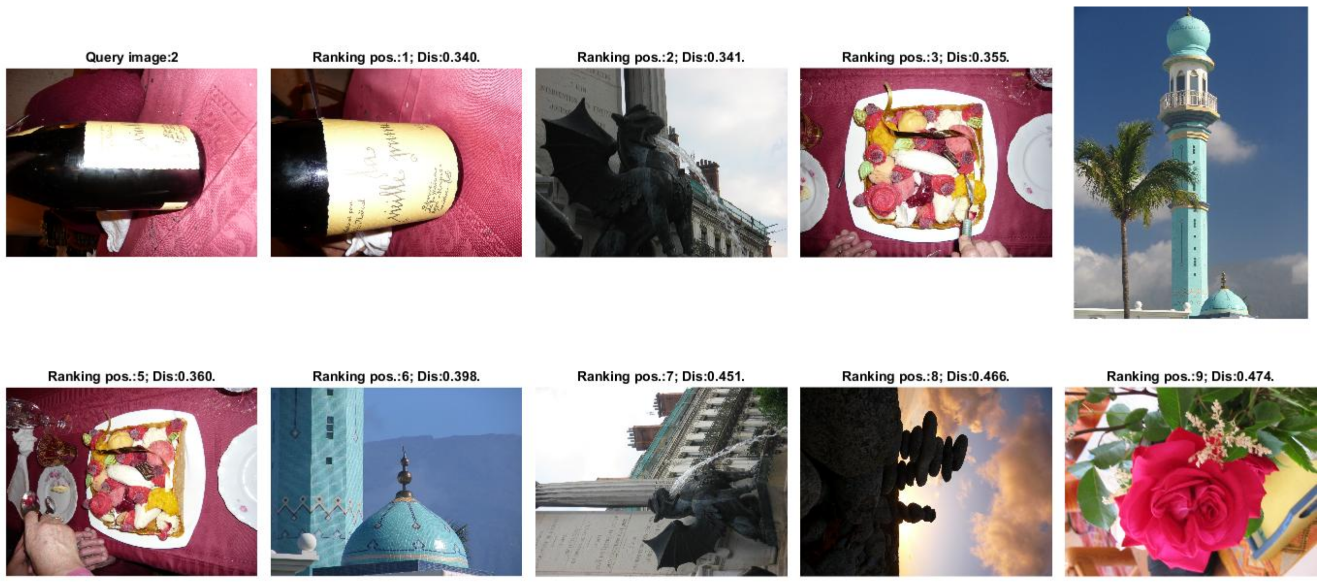
Query image:2
Ranking pos.:1; Dis:0.340.
Ranking pos.:2; Dis:0.341.
Ranking pos.:3; Dis:0.355.
Ranking pos.:5; Dis:0.360.
Ranking pos.:6; Dis:0.398.
Ranking pos.:7; Dis:0.451.
Ranking pos.:8; Dis:0.466.
Ranking pos.:9; Dis:0.474.

Figure 2.13: Image query 2 $Th = 300$

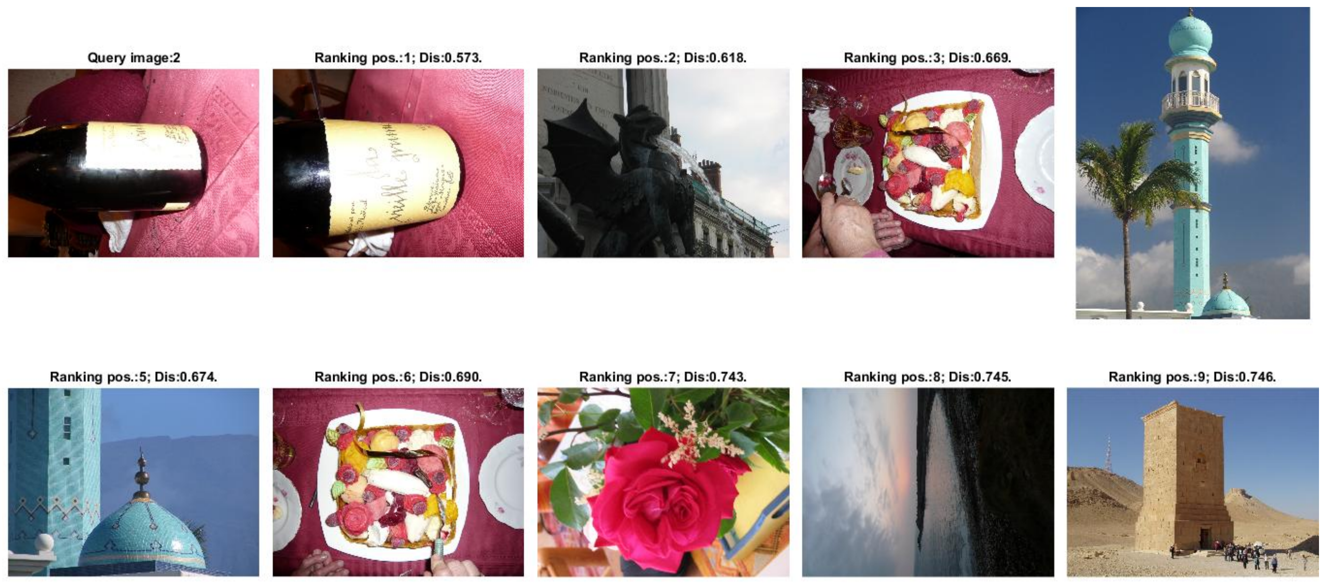
Query image:2
Ranking pos.:1; Dis:0.573.
Ranking pos.:2; Dis:0.618.
Ranking pos.:3; Dis:0.669.
Ranking pos.:5; Dis:0.674.
Ranking pos.:6; Dis:0.690.
Ranking pos.:7; Dis:0.743.
Ranking pos.:8; Dis:0.745.
Ranking pos.:9; Dis:0.746.

Figure 2.14: Image query 2 $Th = 1000$

### 2.3.3 Comparison with findMatches

The last part consisted in implementing a code to execute the ranking of similar images by using the function *"findMatches"*. The query image has been compared to all the images of the gallery and these has been ordered on the basis of the number of matches found with respect to the query image. The resulting ranking is different with respect to the case with the bag of features, but the top ranked images are more or less similar. The big difference lies in the execution time, in fact, when using the bag of feature a big part of the job (building the dictionary) is computed offline, while when using the function *"findMatches"* all the work is performed online and therefore the computations take a huge amount of time. Below are shown the resulting rank with the Bag of Feature and the *"findMatches"* method.

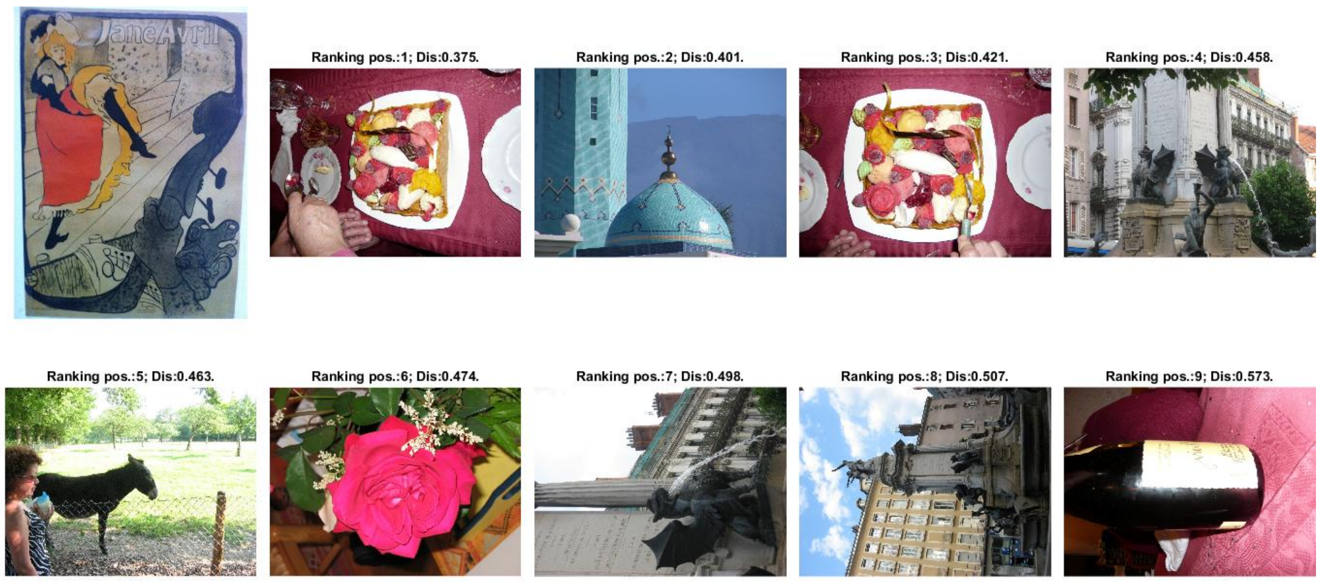
Jane Avril
Ranking pos.:1; Dis:0.375.
Ranking pos.:2; Dis:0.401.
Ranking pos.:3; Dis:0.421.
Ranking pos.:4; Dis:0.458.
Ranking pos.:5; Dis:0.463.
Ranking pos.:6; Dis:0.474.
Ranking pos.:7; Dis:0.498.
Ranking pos.:8; Dis:0.507.
Ranking pos.:9; Dis:0.573.

Figure 2.15: Rank with BoF method

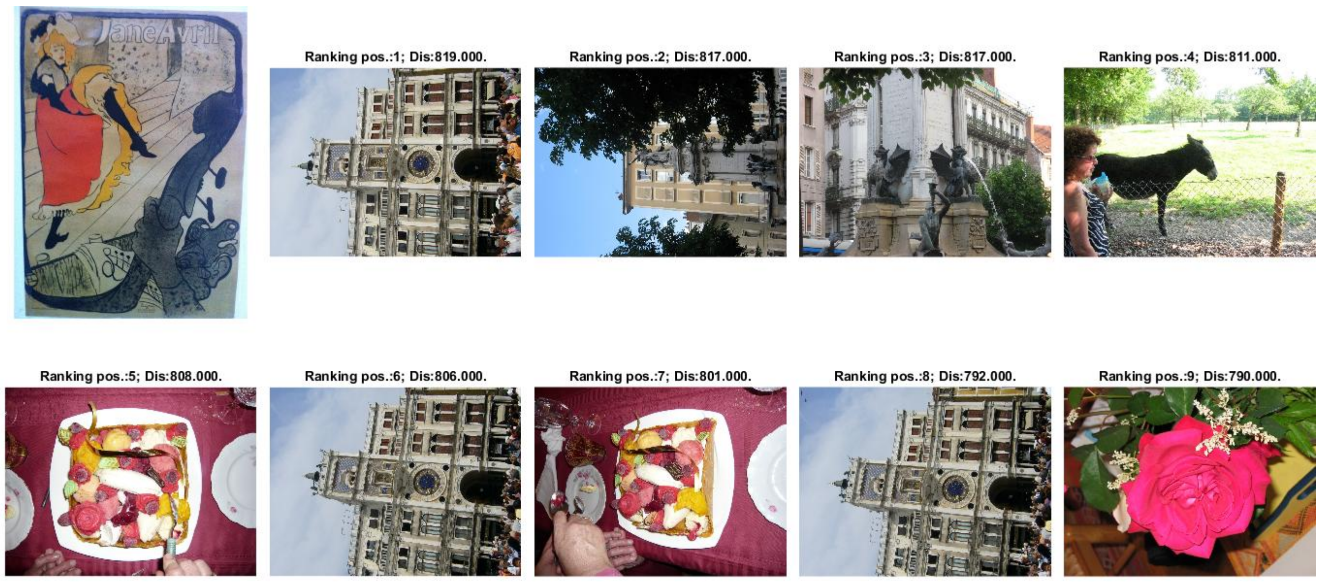
Jane Avril
Ranking pos.:1; Dis:819.000.
Ranking pos.:2; Dis:817.000.
Ranking pos.:3; Dis:817.000.
Ranking pos.:4; Dis:811.000.
Ranking pos.:5; Dis:808.000.
Ranking pos.:6; Dis:806.000.
Ranking pos.:7; Dis:801.000.
Ranking pos.:8; Dis:792.000.
Ranking pos.:9; Dis:790.000.

Figure 2.16: Rank with the *"findMatches"* method