Master MVA
Object Recognition and Computer Vision
**Assignment 1:** Instance-level recognition - 02/11/2016
Achari Berrada Youssef

## QI- Sparse features for matching specific objects in images

### A- SIFT features detections

**1-** It is important that we have similarity co-variant features because pictures in real life are usually taken from differents points of view and angles and we need to have a model that does not depend on these transformations to match object in pictures.

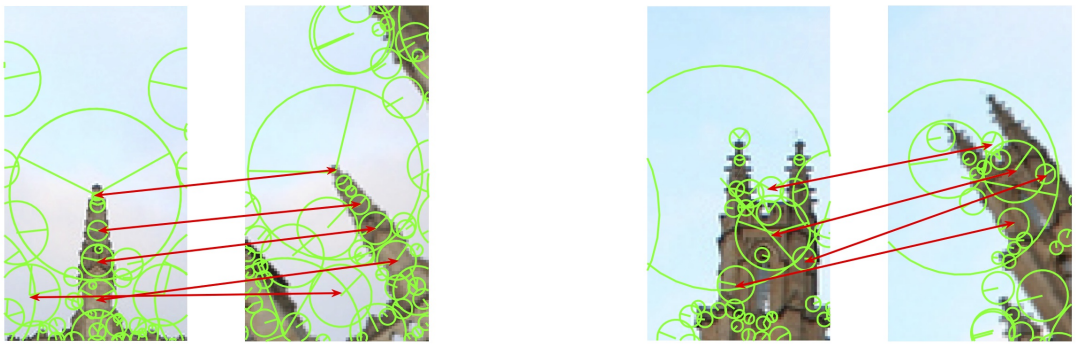
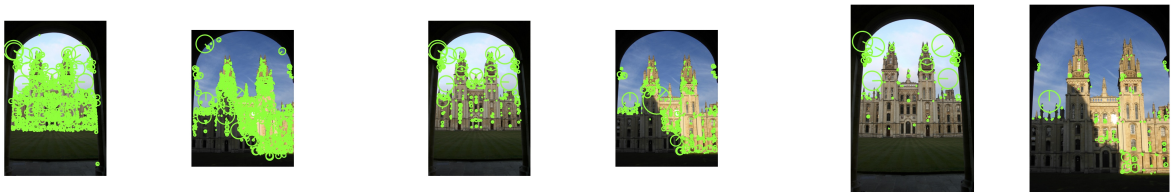


Figure 1: Similarity (equiform) of the detection process.

**2-** Varying the Peak Threshold for two images :



(a) Peak = 0.001 (b) Peak = 0.005 (c) Peak = 0.01

Figure 2: Different PeakThreshold for two images.

**3-** The density of detections across images changes depending on the threshold because the threshold fix the limit of the corner response function that we keep. So this makes the detection depends on the intensity of the image. So the matching of the same object may not work well if the images does not have the same intensity due to luminosity. To avoid this we can first try to preprocess the images by bringing them to the same intensity scale using for example Midway Image Equalization[1].

[1]Implementation of the Midway Image Equalization, Thierry Guillemot, Julie Delon, http://www.ipol.im/pub/art/2016/140/.

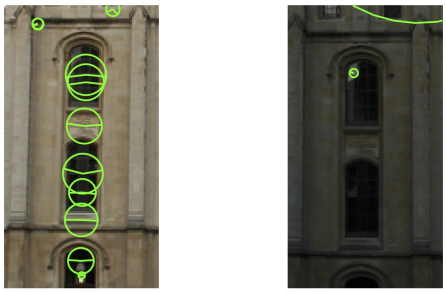


Figure 3: Detection depends on the luminosity for the same PeakThreshold 0.005

**B- SIFT features descriptors and matching between images**

**1-** It is a good strategy to have descriptors over a much larger region because then the matching will be more precise if we match descriptors than if we match frames.

Figure 4: Descriptors.

**2-** Mismatches could happen like in the figure due to change in lighting. To remove the mismatches, one can add constraints on the geometry of similar points. For example, we impose the constraint that three points in the image 1 are similar to three points in image 2 if and only if there is a similarity transformation of the plan that transform those three points from one image to the other.

Figure 5: Mismatches.

## C- Improving SIFT matching using Lowe's second nearest neighbour test

**1-** Mismatches can be removed using the second nearest neighbour test. $\frac{d_1}{d_2}$ is the ratio bounded by the threshold we fix. The number of matches we keep grows with the threshold. This method decreases the uncertainty we have on the first neighbour and thus remove some mismatches.

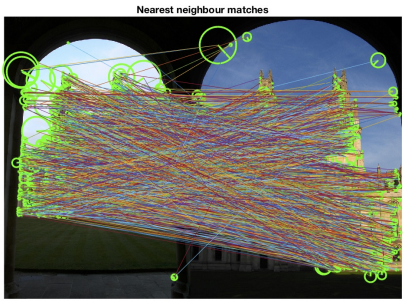
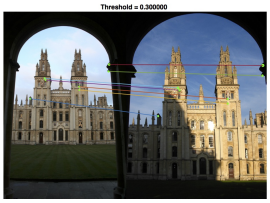
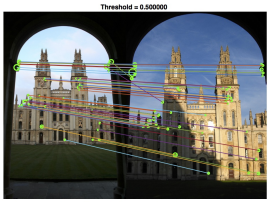


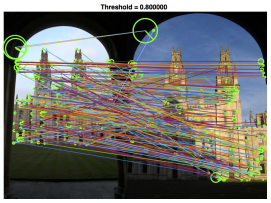

Figure 6: Matches by the first nearest neighbour.




(a) 0.3




(b) 0.5




(c) 0.8

Figure 7: Matches filtered by the second nearest neighbour test.
In figure a) the we have only few matches left but they are all correct matches. In figure b) matches are more than in a) but they are still few and correct. In c) some incorrect matches appear, but we have a correct number of matches.

## D- Improving SIFT matching using a geometric transformation

**1-** We have to find the parameters of the similarity transformation that transform each $x$ in the first image to its corresponding $x'$ in the second one.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = S.R(\Theta) \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \end{pmatrix}$$

is equivalent to :

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S.R(\Theta) & \begin{matrix} T_x \\ T_y \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{pmatrix} . \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

There is 7 parameters to be determined. So we need 4 pairs of correspondent points to determine these parameters.

But we can do an estimation from one pair of correspondent points $\begin{pmatrix} s \\ \theta \\ t_x \\ t_y \end{pmatrix} \leftrightarrow \begin{pmatrix} s' \\ \theta' \\ t'_x \\ t'_y \end{pmatrix}$, by doing the following computation :

$$S = \frac{s'}{s}$$
$$\Theta = \theta' - \theta$$
$$\begin{pmatrix} T_x \\ T_y \end{pmatrix} = -\frac{s'}{s}.R(\theta' - \theta).\begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} t'_x \\ t'_y \end{pmatrix}$$

This is equivalent to :

$$\begin{pmatrix} S.R(\Theta) & \begin{matrix} T_x \\ T_y \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{pmatrix} = \begin{pmatrix} s'.R(\theta') & \begin{matrix} t'_x \\ t'_y \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{pmatrix} . \begin{pmatrix} s.R(\theta) & \begin{matrix} t_x \\ t_y \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{pmatrix}^{-1}$$

**2-** From each correspondence, we compute an estimation of the similarity transformation, and by doing a RANdom SAmple Consensus, we select the transformation that has the highest count of "*inliers*".

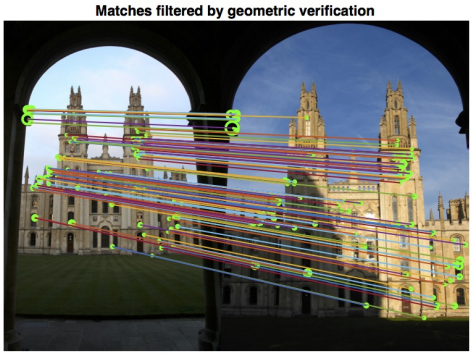



Figure 8: Matches filtered by geometric verification.

## QII- Affine co-variant detectors

**1-** At first there are more similarity detector matches than affine. It is due to the conditions imposed by affine matches. . And when the similarity transformation is enough to describe the matches between images that have approximately the same viewpoint, we may find more similarity matches than affine matches which have more relaxed conditions. Indeed, affine descriptors have 6 parameters against 4 parameters for similarity descriptors.

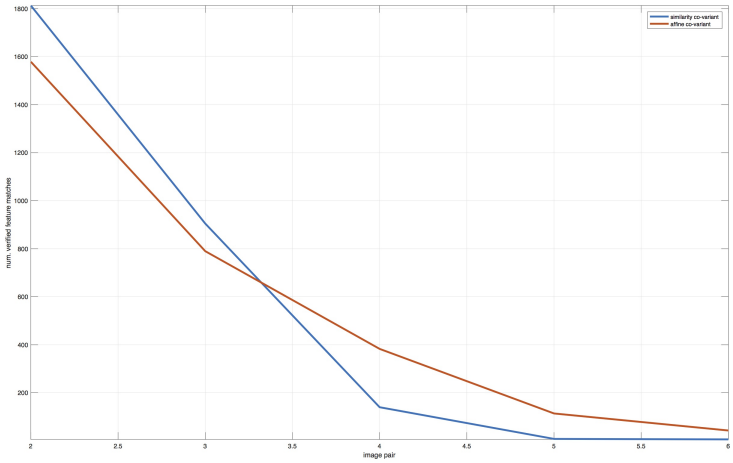



Figure 9: Number of verified matches with changing viewpoint.

(a) Image1 (b) Image2 (c) Image6

Figure 10: There is more similarity matches between image1 and image2.
And there is more affine matches between image1 and image6.

## QIII- Towards large scale retrieval

### A- Accelerating descriptor matching with visual words

**1-** In practice, the convertion from descriptors to visual words is done a priori such that at the moment of doing the query, we have already the visual words computed for the decriptors of each image.

**2-** For large database, using visual words is 100 times more efficient than comparing descriptors directely between the query image and each image in the database.

Table 1: Speedup in searching for different random database.

| Search Number | 1 | 10 | 50 | 100 | 200 | 500 |
|---|---|---|---|---|---|---|
| Time for raw | 0.0359 | 0.2252 | 1.9618 | 2.2731 | 5.3128 | 11.0840 |
| Time for visual words | 0.0010 | 0.0067 | 0.0322 | 0.0274 | 0.0577 | 0.1104 |

### B- Searching with an inverted index

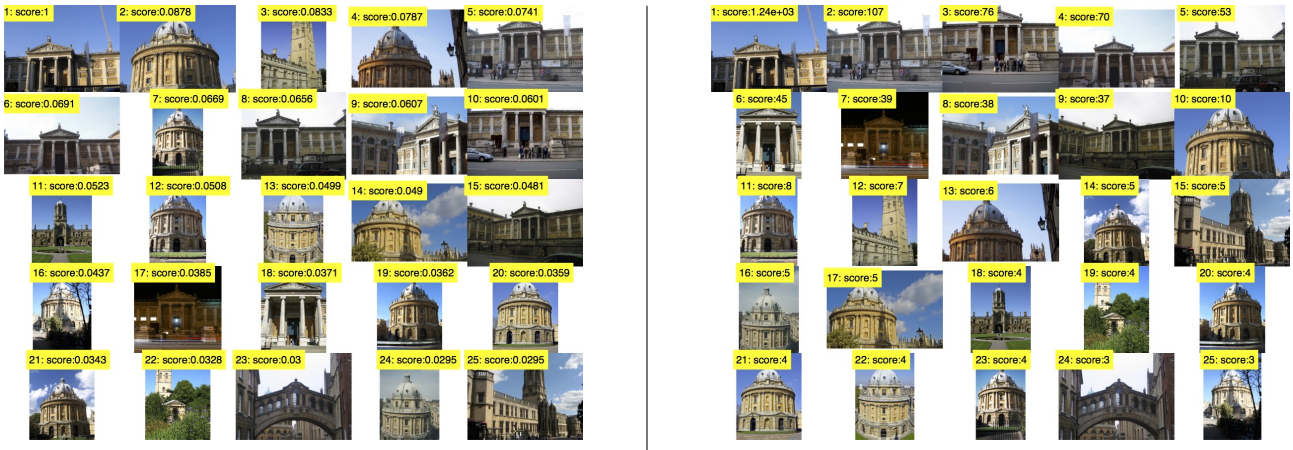
**1-** The top image has a score of 1 because it correspond to the same image of the query and so it's the norm of a normalized histogram which is a unit vector, it's why it's one.

**2-** Among the 25 top results, there is 16 erroneously matched images.

### C- Geometric rescoring

**1-** The top score is much larger than one because now we are rescoring based on the number of inlier matches after a geometric verification step. This is indeed much larger than one especially for the first image scored.

**2-** After geometric verification among the previous 25 top scores, all the erroneously results appears after true ones whereas before they were mixed with the true ones as it is illustrated in the following figure.




(a) Before Geometric Verification.

(b) After Geometric Verification.

Figure 11: Search results using histogram similarity measure.

### D- Full system

**QIV- Large scale retrieval**

**1-** The painting database stores an **inverted file index** of size **NumberOfWords** $\times$ **NumberOfImages**. For each image, it stores the visual words it contains and each word is weighted by its ***inverse document frequency***. It does not store the images locally.

**2-** The image database take O(NumberOfWords $\times$ NumberOfImages) memory. In our experiment, NumberOfWords = $10^5$ and NumberOfImages = 1734.

**3-** First, we compute the histogram of visual words our query image contains. Second we score each image in the database by the inverted index method. This step is computed efficiently in our experiement by a sparse matrix multipliation in Matlab. And finally, we re-score using a geometric verification.

Table 2: Search Times by step.

| Step | Painting1 | Painting2 | Painting3 |
|---|---|---|---|
| Feature time | 0.239 | 0.397 | 1.621 |
| Index time | 0.022 | 0.023 | 0.027 |
| Geometric verification time | 0.358 | 0.543 | 1.981 |