# Assignment 1: Instance-level recognition

Adil Rhoulam

arhoulam@ens-paris-saclay.fr

**QIA.1: (i) Why is it important to have a similarity co-variant feature detector? (ii) How does this affect the descriptors computed at these detections? (iii) How does this affect the matching process?**

(i) It is important to have a similarity co-variant feature detector because it allows us to normalize the features to canonical form (standard shape) which ease the process of establishing features correspondences between multiple images. It ensures also the accuracy and stability in image matching and recognizing.

(ii) The fact that the feature detector is co-variant causes the descriptors computed at these detections to have invariant to a group of feature deformations like orientation, uniform scaling, change of illumination and partially invariant to affine intensity changes.

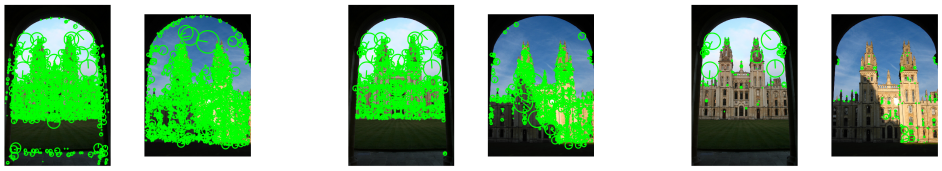
(iii) Having feature detectors with similarity co-variance and feature descriptors with the invariance property mentioned previously ease the matching process by using a similarity measure (i.e Nearest Neighborhood) to compare invariant descriptors to match regions in different images.

**QIA.2: Show the detected features in the two images for three different values of the peakThreshold option..**

(i) The detected features with three different values of $peakThreshold$ are shown in Figure 1.

**QIA.3: Note the change in spatial density of detections across images, for a given value of peakThreshold. (i) Is the density uniform? If not, why? (ii) Which implications for image matching can this have? (iii) How can it be avoided?**

(i) For a given value of peakThreshold, we can notice that for the same feature, their is no uniform density detection across the two images. This is because of the difference of viewpoint of the two images. This can causes some mismatching between features, because some invariance properties could be lost (like intensity), we can avoid this by computing the descriptor of those features over a larger region (i.e a square).



(a) $peakThreshold = 0.0001$ (b) $peakThreshold = 0.001$ (c) $peakThreshold = 0.01$

Figure 1: Feature detectors with three different values of $peakThreshold$

**QIB.1: Note the descriptors are computed over a much larger region (shown in blue) than the detection (shown in green). Why is this a good strategy?**

(i) The descriptors are computed over a much larger region than the detection to characterize it in a way that is partially invariant to intensity, contrast changes and small geometric deformations.

**QIB.2: Examine carefully the mismatches and try to understand the different causes of mismatch. (i) In your report, present at least 3 of them and explain why the mistakes are being made. For example, is the change in lighting a problem? (ii) What additional constraints can be applied to remove the mismatches?**

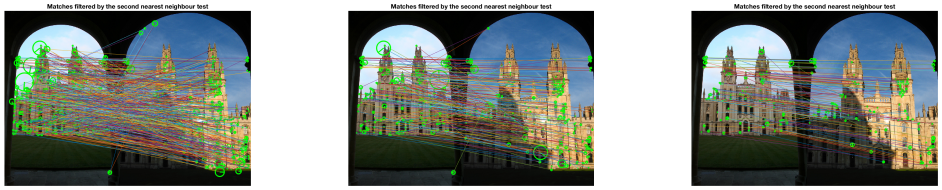
(i) We can notice that there is a lot of mismatching between features in the first image and the ones that are in the area exposed to the sun in the second image, this is because the significant change in illumination. We can see also that there is an ambiguity of matching the same descriptor to more than one pattern because of the repetitive patterns that are in the image (similar windows/ doors...), also we can notice that the noise added by the clouds in the first image which is detected by the feature detector lead to some mismatches. Eventually, the choice of the matching scheme is also a factor that can lead to a lot of mismatches if not well selected.

(ii) Some additional constraints that can be applied to remove the mismatches are: Thresholding the scores given by the NN approach (a match is not accepted if the NN score isn't bigger than the Threshold T), or if a feature has multiple matches, accept the one with the highest score, we can also use a different scheme like the ratio test and adding to it the thresholding condition also.Moreover, we. can add a geometrical constraint.

**QIC.1: Illustrate and comment on improvements that you can achieve with this step. Include in your report figures showing the varying number of tentative matches when you change the nnThreshold parameter.**

(i) Using the ratio scheme which compares between the first and the second nearest neighborhoods to match the features, we can see that for some values of Threshold, the matching process has improved significantly, we have to chose the threshold in a way that it doesn't delete the true positive matches and it doesn't accept the false positive matches, for this we can see that the value 0.8 (Figure 2 (b)) is a very good threshold comparing with the value 0.9 (Figure 2 (a))which contains a lot of mismatches and

the value 0.7 (Figure 2 (c))which delete some true positive matches, so the parameter threshold should be selected respecting the trade-off mentioned previously (we could learn it using a machine learning algorithm).



(a) $nnThreshold = 0.09$ (b) $nnThreshold = 0.08$ (c) $nnThreshold = 0.07$

Figure 2: Matching process with three different values of $nnThreshold$

**QID.1: Work out how to compute this transformation from a single correspondence. Hint: recall from Stage I.A that a SIFT feature frame is an oriented circle and map one onto the other.**

(i) If we denote the parameters of the two frames of correspondance frame0=($x_0$,$y_0$,$\theta_0$,$s_0$) and frame1=($x_1$,$y_1$,$\theta_1$,$s_1$). To obtain frame 1 from frame 0, the rotation angle would be ($\theta_1$-$\theta_0$) and the scale parameter would be $s_1/s_0$ (those transformations are assumed to be done from the origine of the image0), after this transform the new coordinates of the frame0 are $x_0$'=($s_1/s_0$)*($x_0$*cos($\theta_1$-$\theta_0$)-$y_0$*sin($\theta_1$-$\theta_0$)) and $y_0$'=($s_1/s_0$)*($x_0$*sin($\theta_1$-$\theta_0$+$y_0$*cos($\theta_1$-$\theta_0$)) then the translation parameters are $t_x$= $x_1$-$x_0$' and $t_y$=$y_1$-$y_0'$

**QID.2: Illustrate improvements that you can achieve with this step.**

(i) We can see that the there is a noticeable improvements in matching features when adding the geometric similarity constraint by comparing the figure(c) of Figure2 obtained without the geometrical contraint with the one which impose it. The method of geometrical transformation is an iterative model that in each iteration fit well the matching and delete the mismatches, it's then more robust than the ones which don't take geometrical considerations.

**QII.1: Include in your report images and a graph showing the number of verified matches with changing viewpoint. At first there are more similarity detector matches than affine. Why?**

(i) We can notice that when the viewpoint has a remarquable transformation, the affine method works very well in detecting valid matches than similarity method. At first there are more similarity detector matches than affine because the viewpoint wasn't dramatically changed, then the similarity method is working well in that case. We can compare between the two methods in Figure 4 and 5.

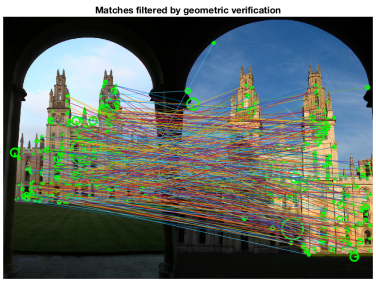



Figure 3: Matching process when adding the geometric transformation constraint

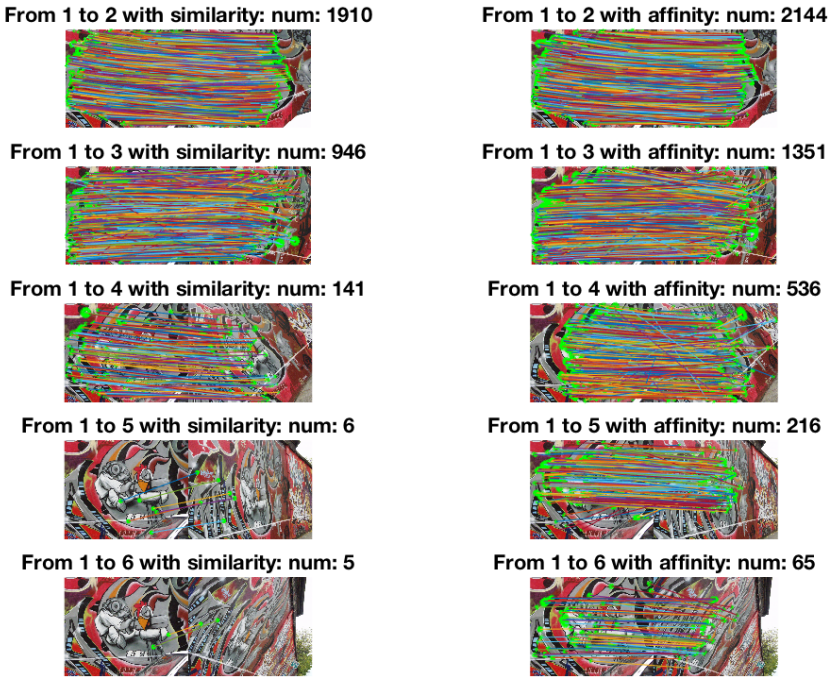



Figure 4: Images showing the comparison of Similarity and Affinity methods in matching features in different images with query image 1

**QIIIA.1: In the above procedure the time required to convert the descriptors into visual words was not accounted for. Why the speed of this step is less important in practice?**

(i) The speed of this step is not important in practice, because we can't avoid it (the step) if we don't want to footprint the descriptors on memory and it elapsed time could

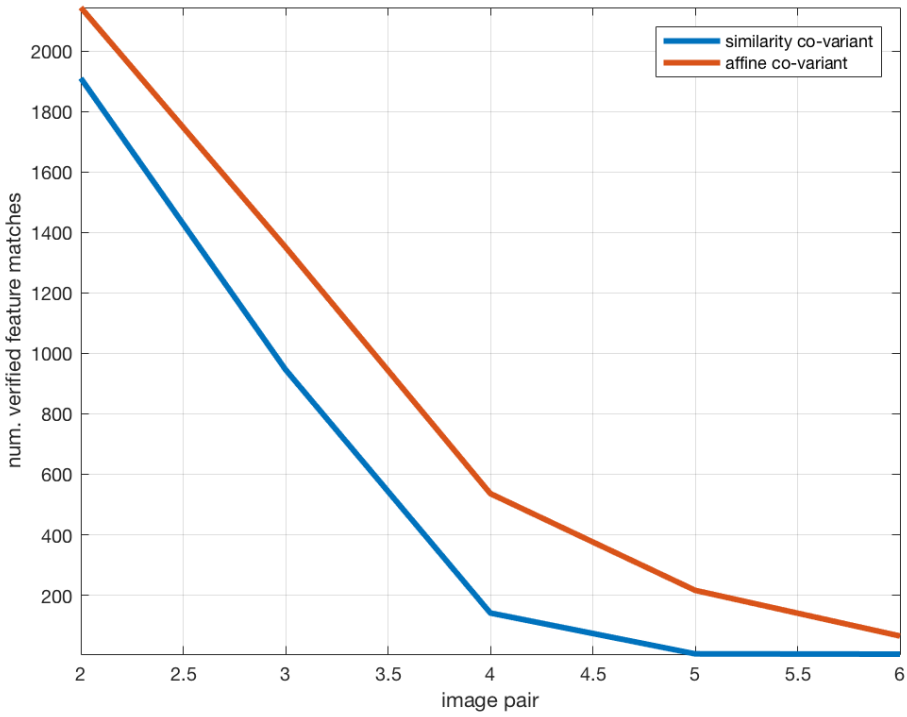



Figure 5: Number of verified matches with changing viewpoint: Affinity vs Similarity

be long, then we lose the comparison between the raw matching and the quantize matching, if we want to take it into account, we should then take also into account the spatial complexity of the descriptors footprint in memory which are incomparable w.r.t to their units of measure.

**QIIIA.2: What is the speedup in seconds in searching a large, fixed database of 10, 100, 1000 images? Measure and report the speedup in seconds for 10, 100 and 1000 images.**

(i) we can see in the Figure 6 that when matching between only two figures, the time elapsed is greater with the raw match method than the quantize one. The time elapsed from the example of matching two images is for Raw descriptors: 0.056s, and for Quantize descriptors: 0.00847s, if we consider that for each matching between any images in a large dataset the average time elapsed by matching two figures is 0.056s for Raw descriptors, and 0.00847s for Quantize descriptors, then for a dataset of 10, 100, 1000 will be 0.056s, 0.56s, 5,6s respectively for Raw descriptors and 0.00847s, 0.0847s, 0.847s respectively for Quantize descriptors.

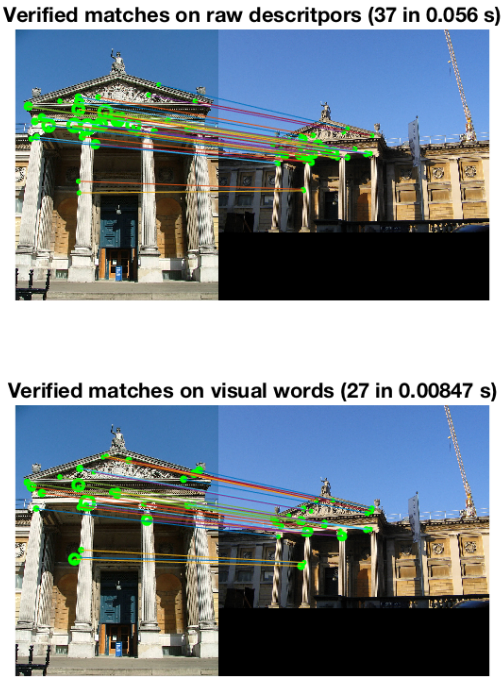



Figure 6: Raw matches vs Quantize matches of two images

**QIIIB.1: Why does the top image have a score of 1?**

(i) The top image have a score of 1 because it's the query image, and we are matching it to itself which gives a normalized similarity score of 1.

**QIIIB.2: Show the first 25 matching results, indicating the correct and incorrect matches. How many erroneously matched images do you count in the top results?**

(i) In the Figure 7, the incorrect matches are ranked 9, 10, 12, 13, 14, 15, 17, 18, 19, 22, 23, 25. They represent 12 from 25 first matches.

**QIIIC.1: Why is the top score much larger than 1 now?**

(i) Here the top score is match larger than 1, because the score is the number of inliers ( number of matches between the two images which is a huge number because it's the same image. We can see that the score is reasonable when it's an unseen query image Figure 9.


Figure 7: First 25 matching results and their scores

**QIIIC.2: Illustrate improvements of retrieval results after geometric verification.**

(i) In the Figure 8, we can see an improvement in scoring the matches, because with this method the incorrect matches are ranked in general in the last of the matches and have scores that are very low when comparing with the top of 4 first matches, they are ranked: 8, 10, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25. They represent 12 from 25 first matches.

**QIV.1: How many features are there in the painting database?**

(i) total number of features: 4.94 M.

**QIV.2: How much memory does the image database take?**

(i) size in memory: 281.8 MB

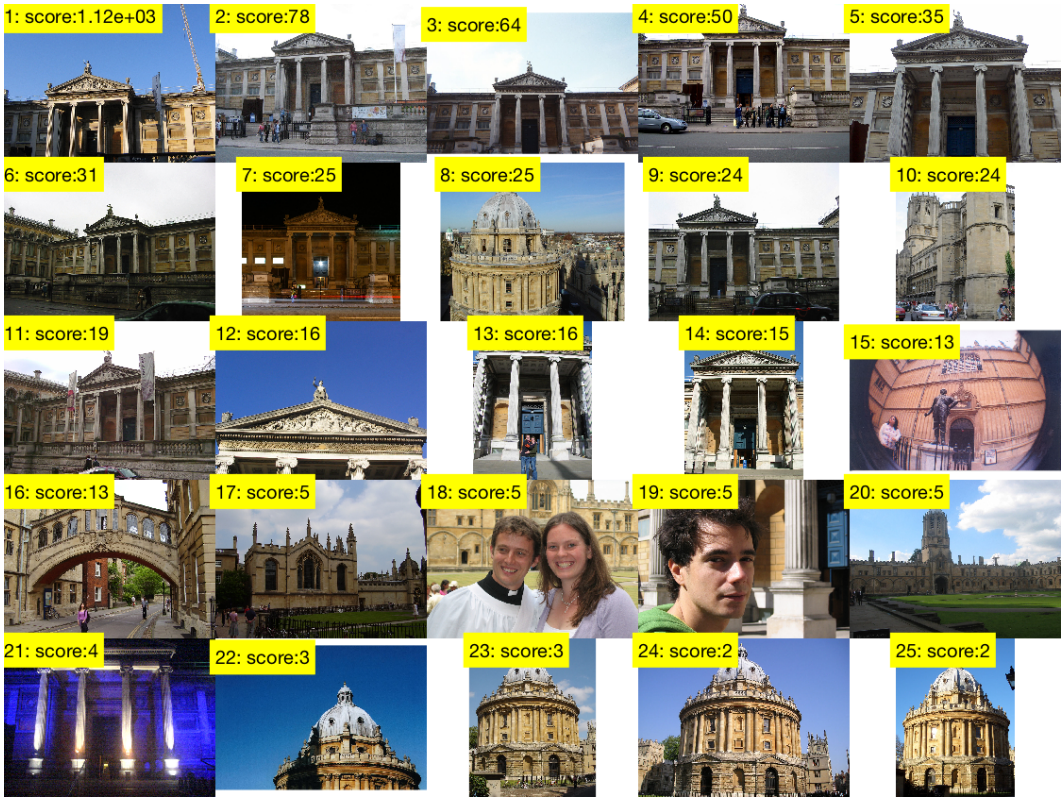



Figure 8: First 25 matching results and their scores with geometric constraint

**QIV.3: What are the stages of the search? And how long does each of the stages take for one of the query images?**

(i) search: feature time: 0.128 s
search: index time: 0.012 s
search: geometric verification time: 0.087 s

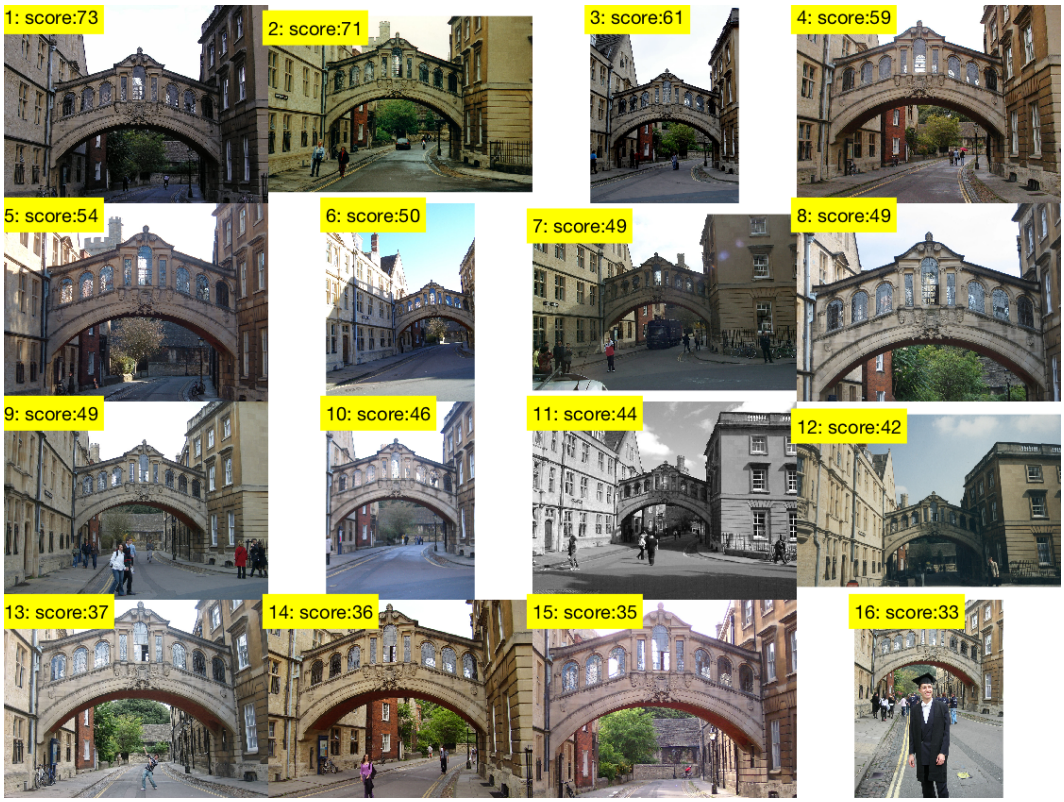



Figure 9: First 25 matching results and their scores with geometric constraint with an unseen query image