Assignment 1: Instance-level recognition

Victor Busa

victor.busa@ens-paris-saclay.fr

I Sparse features for matching specific objects in images

I.A SIFT features detections

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, in real life, images undergo various transformation such as scaling, rotation, translation (shear, lighting and so on) depending on the conditions and the point of view from which the pictures was taken. Hence, it is important to have a model that does not depend upon these transformations.

(ii) The fact that the feature detector is co-variant causes the descriptors computed at these detections to be the same. Hence the descriptors are invariants.

(iii) Having feature detectors with similarity co-variance and feature descriptors invariant under similarity transforms allows the matching process to find the same features of the query image even if this one undergoes a similarity transform.

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

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) The density of detections is not uniform. It actually depends upon the intensity of the image. The higher the contrast, the higher the detection density is.

(ii) Hence, the matching process won't work very well when the two images have different intensity

(a) $peakThreshold = 0.001$ (b) $peakThreshold = 0.005$ (c) $peakThreshold = 0.01$

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

(iii) To avoid this problem, we can preprocess the images in order to bring them to the same intensity.

I.B SIFT features descriptors and matching between images

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?

The fact that the descriptors are computed over a much larger region than the detection is a good strategy in the sense that the matching will occur on a broader space and hence the detection would likely be more accurate.

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) The mismatches can be due to: occlusion, lighting, scale, viewpoint, blur, non rigid deformations (such as clothes), or singularity (such as light reflecting on a surface). These mismatches are shown on Figure 2 and Figure 3

(ii) The change in lighting is a problem, as we've already highlighted in Q1A.3

(iii) To reduce the number of mismatches, one can use a cluster of at least 3 features that agree with the **geometrical position** of an object. These clusters are more likely to be correct than if we only focus on individual features matches.

I.C Improving SIFT matching using Loweś second nearest neighbor test

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.

We can considerably reduce the number of mismatches while discarding very few correct matches by using the second nearest neighbor test. The idea behind this technique relies on the fact that:

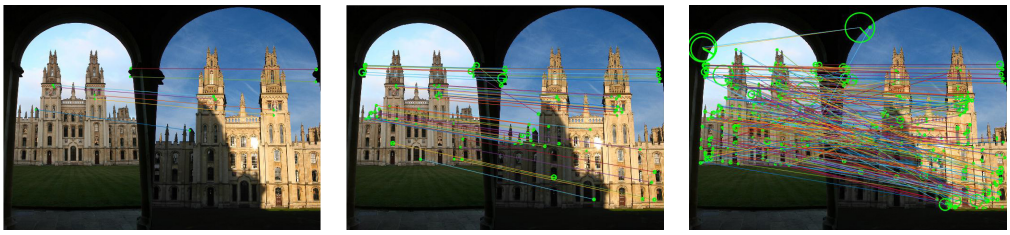
(a) Mismatches due to lighting (b) Mismatches due to partial occlusion

Figure 2: Mismatches due to lighting and occlusion

(a) Mismatches due to scale difference (b) Mismatches due to change in viewpoint

Figure 3: Mismatches due to change in scale and viewpoint

(i) correct matches should have the first nearest neighbor closer than the closest incorrect match (the second nearest neighbor).
(ii) erroneous matches should have the first and second nearest neighbor within the same order of magnitude. Hence, incorrect matches will have a ratio around 1.



(a) $nnThreshold = 0.3$ (b) $nnThreshold = 0.5$ (c) $nnThreshold = 0.8$

Figure 4: Effect of the second nearest neighbor with three different values of $nnThreshold$

I.D Improving SIFT matching using a geometric transformation

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.

As similarity transformation can be written mathematically as follow:

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

Where $R(\Theta)$ denotes a rotation by Θ, S refers to the isotropic scaling factor and T_x, T_y account respectively for the x-axis translation and the y-axis translation.

Considering two corresponding points: $\begin{pmatrix} s \\ \theta \\ t_x \\ t_y \end{pmatrix}$ and $\begin{pmatrix} s' \\ \theta' \\ t'_x \\ t'_y \end{pmatrix}$

we can estimate the parameters $S, R(\Theta)$ and $\begin{pmatrix} T_x \\ T_y \end{pmatrix}$ as follow:

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

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

For each tentative correspondence, we compute the similarity transformation. Then we apply the same idea as in the RANSAC algorithm. At the end, after testing several affine transformations, we select the one with the highest count of inliers. The resulting matches after applying a geometric verification are shown in Figure 5

II Affine co-variant detectors

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?

At first, there are more similarity detector matches than affine because similarity transformations are invariants through rotation, translation, and zoom and the first 2 images undergo only simple variations similar to these transformations. Then, when the initial image undergoes a more complex transformation like **shear**, affine co-variant detectors tend to be more effective as affine transformation can take into account a perspective deformation. Indeed, a square is mapped to any parallelogram through any affine transformation.

Figure 5: Resulting matches after geometric verification

III Towards large scale retrieval

III.A Accelerating descriptor matching with visual words

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?

In practice, we precompute beforehand the visual words of each descriptor, so that we don't need to account for the time required to convert the descriptors into visual words.

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.

We can notice that, for a large database, visual words are 50 times faster than comparing descriptors between the query image and the images in the database.

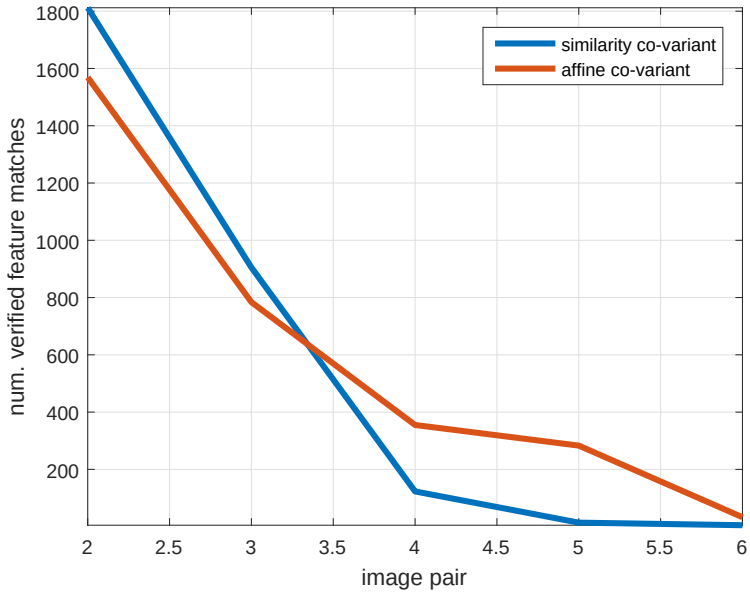


Figure 6: Number of verified feature matches with changing viewpoint

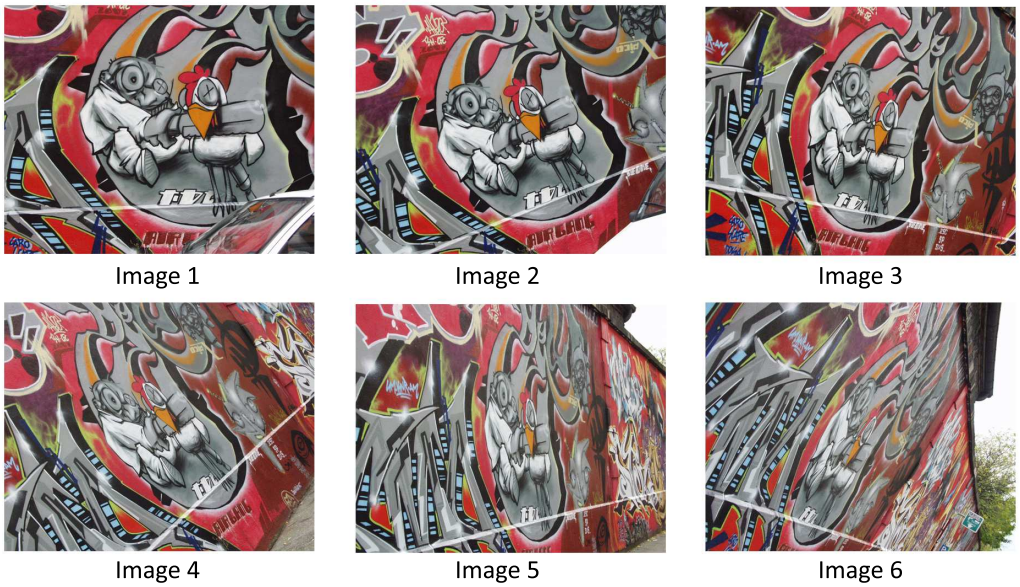


Figure 7: The first 2 transformations of Image 1 (Image 2 and Image 3) are approximately similar to a rotation, while others transformations are more complex and involve changes in perspective

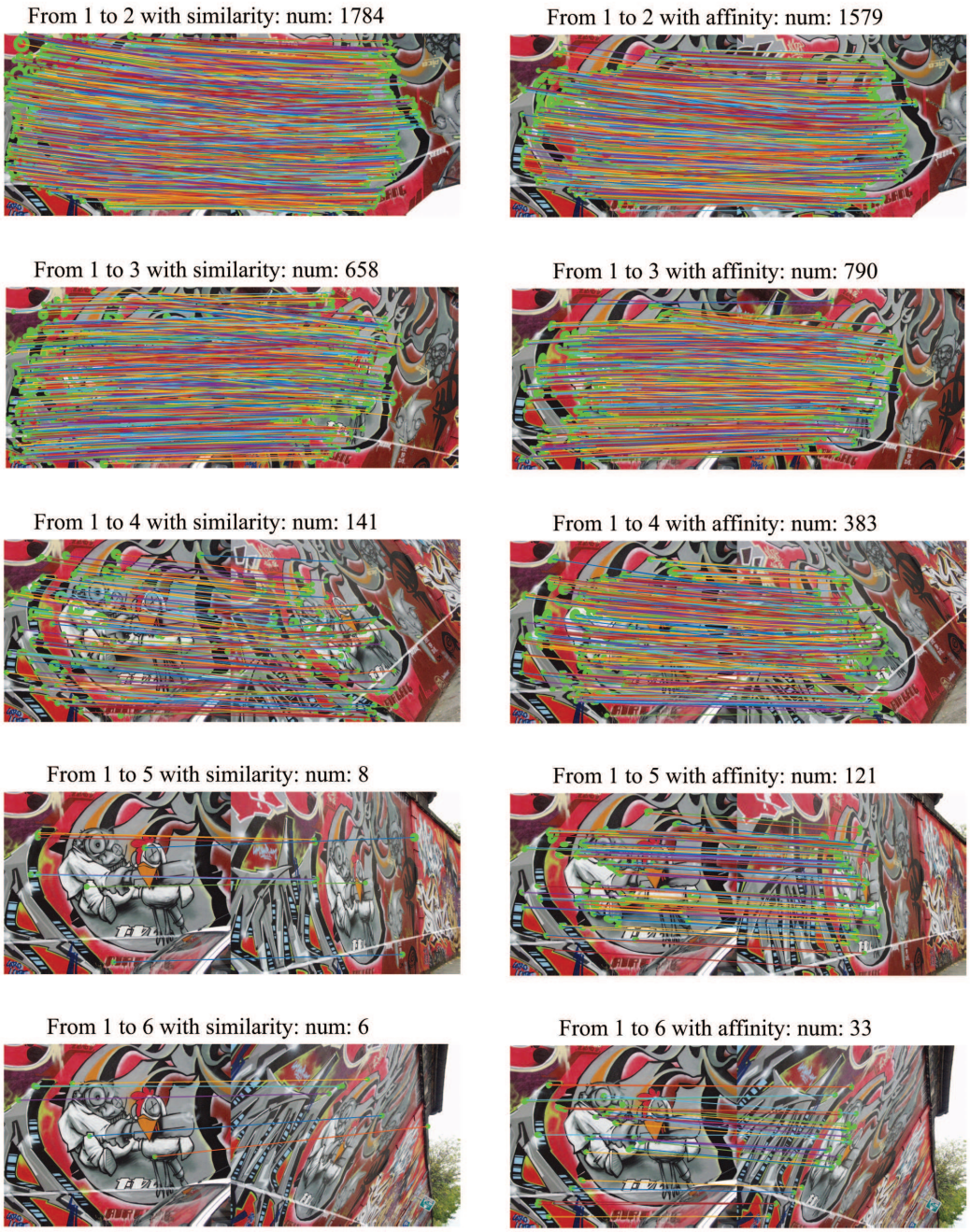


Figure 8: similarity and affine detector matches between the original graffiti image and its 5 deformations

Database size	1	5	10	50	100	250	500
Raw SIFT Time	0.038	0.137	0.243	1.242	2.399	2.399	11.305
Visual words Time	0.0016	0.0039	0.0067	0.0297	0.0574	0.1329	0.2684

Table 1: Speed comparison (in second) of Raw SIFT descriptors and Visual words methods on database of different sizes

III.B Searching with an inverted index

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

The first image has a score of one because it is both the query image and the target image. As the similarity between a query image and a target image is calculate as the inner product of the normalized histograms of theses images, that explains why the score of the first image is 1 ($< u, u >= |u|^2 = 1$)

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?

There are **16 erroneous** images among the 25 top results. The erroneous images are highlighted in red as shown in Figure 9.

III.C Searching with an inverted index

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

We are now re-scoring the images based on the number of inlier matches after a geometric verification step. Hence, the measure is not normalized anymore and that explains why the score is much higher than 1.

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

Using geometric verification allows improving the recognition a lot. Indeed, the erroneous images appear after the correct one whereas when we were searching with an inverted index the correct and incorrect images were mixed. Also, we can notice that there is a gap in term of geometric scoring between the last correct image and the first erroneous image: respectively 37 and 10. See Figure 10.

IV Large scale retrieval

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

the painting database actually stores an inverted file index with 100000 visual words for each image and there are 1734 images. Also, we can notice that each word is weighted by its inverse document frequency as shown in Figure 11

Figure 9: Erroneous images are highlighted in red

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

the image database takes $\mathcal{O}(NumOfWords \times NumOfImages)$ Memory.
Here we have: **NumOfWords** = 100000 and **NumOfImages** = 1734.

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

Stage of the search:

(i) We compute the histogram of the visual words in the query image and in the database images

(ii) We score each image of the database using an inverted file index. This can be done quite efficiently using Matlab's built-in sparse matrix engine.

(iii) We re-score the top-ranked images from the previous step using a geometric verifications step.

Figure 10: Erroneous images are highlighted in red

```
>> imdb

imdb =

              dir: ''
     featureOpts: {'method'  'hessian'  'affineAdaptation'  [0]  'orientation'  [0]}
           images: [1x1 struct]
           kdtree: [1x1 struct]
         numWords: 100000
            vocab: [128x100000 single]
    shortListSize: 100
    sqrtHistograms: 1
            index: [100000x1734 double]
              idf: [100000x1 double]
```

NumOfWords x NumOfImages

inverse document frequency

Figure 11: Image database structure

Table 2: Search Times (in second) for each mystery images

Image	**The Starry Night**	**Wheat Field with Cypresses**	**Sedie**
Feature time	0.235	0.424	1.770
Index time	0.028	0.032	0.034
Geometric verification	0.372	0.541	1.967