RDFIA: Pattern recognition for image analysis and interpretation

TP 1 & 2 Report

Kevin Meetooa

Année 2020-2021

Contents

Chapter 1

Introduction

This report presents the work done and the results obtained during the first two TP sessions.

We initially implemented a vectorized representation of a small image patch. This representation is called SIFT (Scale-invariant feature transform): The SIFT is a local descriptor aimed to numerically characterize an image patch.

We then built a "visual dictionary" composed of average descriptors representing the frequent patterns among all the SIFTs within our image set.

For a given image, we then aggregated all the SIFTs of that image with the "Bag of Words" method. The aim was to represent each image as a condensed vector, using the visual dictionary and the SIFTs previously computed.

Chapter 2

SIFT

A SIFT (Scale-invariant feature transform) is a local visual descriptor. It transforms a small image patch of 16 x 16 pixels into a numerical representation (a vector of shape 128) which will be robust to perturbations and transformations. Therefore, two similar patches will have very similar (in terms of vector distance) SIFTs.

We define the following masks, $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ and $M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

We use those masks to compute the partial derivatives (in terms of light intensity) I_x and I_y of the image I.

1. We can "separate" the matrixes M_x and M_y as the product of two vectors:

   $M_x = h_y * h_x.^\intercal$ and $M_y = h_x * h_y.^\intercal$ with $h_x = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$ and $h_y = \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$

   We can interpret these two vectors: The vector h_x allows us to quantify the "difference" between the left and right pixel, whereas the vector h_y acts as a "Gaussian" which gives a higher weight to the middle pixel compared to the left and right pixels.

2. This separation of the convolution filter has several advantages:

   - Space complexity: It costs less to store two vectors of size 3 rather than a matrix of size (3,3) containing 9 elements.
   - Time complexity: During a convolution, we only do 6 multiplication when using the vector representation against 9 multiplications for the matrix representation.

3. The Gaussian mask gives more weight to the pixels located at the center of the patch compared to the pixels located at the edge, as shown on figure 2.1.

   The point of this mask is to "blur" the patch in order not to be too sensitive to small local changes.

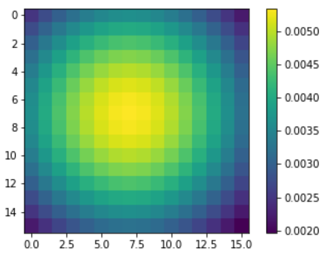

Figure 2.1: Illustration of the Gaussian mask weighting

4. The discretization of orientations in eight classes presents several advantages:

   - Space complexity: It costs less to store an array of size 8 rather than an array of size 360 (corresponding to all the 360 possible angles, rounded to the nearest degree).

   - We have a small "invariance" by rotation: All the angles between 0 and 45° will be classified in the same class.

     Thus, our SIFT is invariant by rotation for rotations lower than 45°, which makes it robust to eventual noises due to the angle of from which the photo was taken.

5. We now have a vector $P_{enc}^{I} \in \mathbb{R}^{128}$ describing the image I.

   - The first post-processing step is computing the norm of the vector P_{enc}^{I}. If that norm is lower than 0.5, then we consider the vector as uninteresting and set it to the null vector.

     In fact, we only want encodings with a high norm because a low norm may mean that the patch doesn't contain any interesting/discriminative information.

   - We then normalize the vector. We are going to use the vector P_{enc}^{I} to characterize an image. To compare our image to another image, we will compare their respective P_{enc} vectors. For this reason, it is preferable for these vectors to have the same order of magnitude for the comparison to make sense.

   - Finally, the values of the vector greater than 0.2 are set to 0.2 and the vector is then normalized again. This is because a too high value may be caused by an "artefact" noisening the patch.

6. The SIFT is invariant by small rotation (depending on the discretization step used) and by light intensity (we only look at the derivative of the light intensity, not the intensity itself) which makes it a robust descriptor, invariant by the noise caused by the angle/light/spot from which the picture was taken.

   Moreover, the SIFTs are discriminant, two different patches will most likely have two different SIFTs. Therefore, it is reasonable to use the SIFT to define an image's "signature".

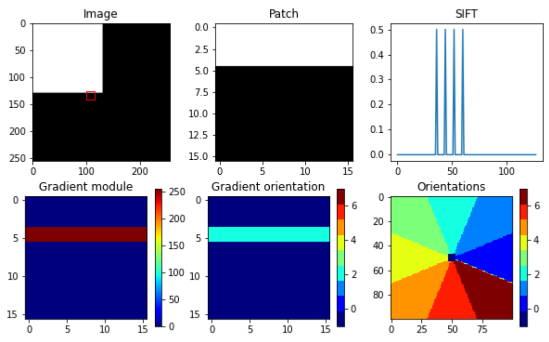

Figure 2.2: Computation of an image patch's SIFT

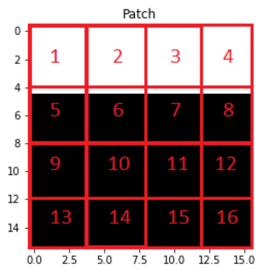

Figure 2.3: Division of the patch into 16 sub-patches

7. The figure 2.2 shows the different results obtained on a 16 x 16 pixels given patch.

   Intuitively, the SIFT graph obtained makes sense for this given patch. In fact, to compute the SIFT, we divide the patch into 16 sub-patches as shown on figure 2.3

   The first 4 and the last 8 sub-patches are respectively entirely white and entirely black. Therefore, for these patches, the light intensity is constant, leading to null values for the partial derivatives I_x and I_y. This leads to a null SIFT for these patches.

   However, the sub-patches n°5, 6, 7, 8 all contain both black and white at the same time, leading to strictly positive partial derivatives and therefore strictly positive SIFT values.

   The four spikes on the figure 2.2 correspond to these four patches.

Chapter 3

Visual dictionary

8. The goal of these TPs is to implement an image classification model. Each image is represented as a vector $P_{enc} \in \mathbb{R}^{128}$ as explained in 2. A first possible approach to compare images to each other would be to compare their vectors P_{enc} but this seems ineffective. In fact, to compare n images to each other, we would have to perform $\frac{n(n+1)}{2}$ comparisons.

   For our set of 4485 images, this corresponds to around 10 million comparisons, which is not feasible in a reasonable amount of time.
   The aim of the visual dictionary is to compute "average descriptors" on images.

   We would then only need a few descriptors to describe our image set. This means that a descriptor would correspond to an "average example" of several images.

   Our visual dictionary would thus be composed of "archetypal" descriptors describing the distribution of images.

   This approach has the advantage of having a much lower risk of over-fitting than the first one (where we would have one descriptor per image).

9. Let $L = \sum_{m=1}^{M} \sum_{i}^{128} \|x_i - c_m\|^2$ be the loss function associated with our clustering problem. We want to find the center of the cluster c_k which would minimize that function.

   To do this, we compute the partial derivative of L with respect to c_k.
   $\frac{\partial L}{\partial c_k} = \frac{\partial}{\partial c_k}(\sum_{m=1}^{M} \sum_{i}^{128} \|x_i - c_m\|^2) = \frac{\partial}{\partial c_k}(\sum_{i}^{128} \|x_i - c_k\|^2) = \sum_{i}^{128} 2(x_i - c_k)$

   We now want to cancel this derivative. $\sum_{i}^{128} 2(x_i - c_k) = 0 \Rightarrow c_k = \frac{1}{128} \sum_{i}^{128} x_i$

   Thus, the center of the cluster c_k that minimizes dispersion is the barycenter of the points x_i.

10. In practice, it is impossible to determine the "ideal" number of clusters a priori. We would need to experiment using an "elbow" type method: This would allow us to select a number of clusters that would minimize the dispersion of data without falling into overfitting.

11. The dictionary items, called centroids, are average descriptors from a mixture of several patches. However, a centroid is only a vector of $\mathbb{R}^{128}$. Nothing more can be said about the centroid and it cannot be transformed into an image.

    To analyze a cluster, it is better to look at the examples of patches in the cluster since we can't look at the centroid itself.

    Indeed, by looking at the images of the patches in the cluster, we could get a pretty good idea of the cluster and the common characteristics of its patches.

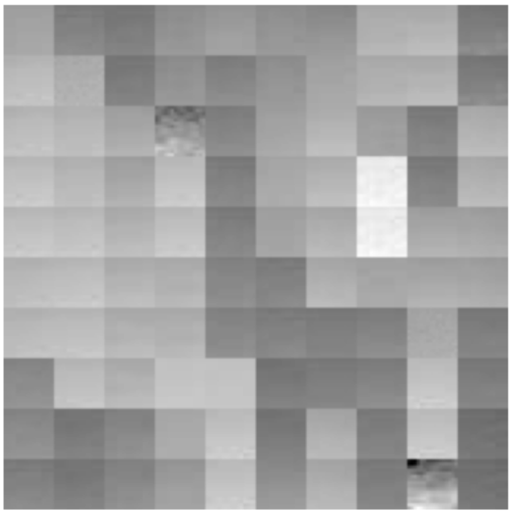

Figure 3.1: Elements from a given cluster

12. The figure 3.1 shows the 100 closest elements of a randomly chosen cluster center. The results obtained seem to make sense.

    Indeed, the patches are all of the same nature: they are all patches of uniform color with no particular pattern. The clustering algorithm seems to have worked relatively well.

Chapter 4

Bag of Words

13. The z vector represents the distribution (the number of elements) of the patches in each cluster.

14. The figure 4.1 shows the results obtained after running the Bag of Words algorithm on an image from our dataset. We can see that the algorithm manages to capture the different patterns of the image quite well: We can see that the black building, the white cloud and the gray sky belong respectively to the red, blue and orange clusters.

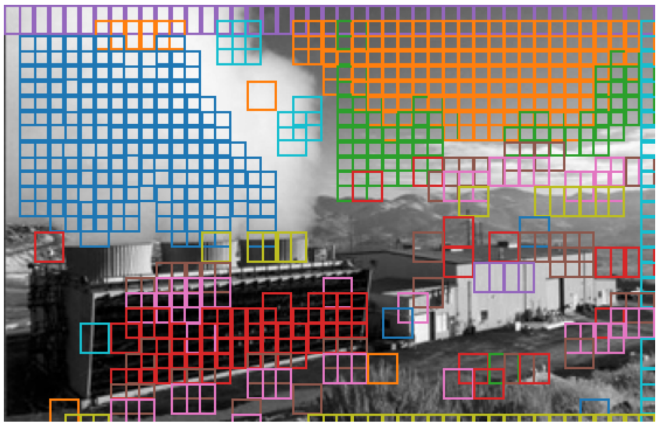

Figure 4.1: Result of the Bag of Words algorithm on an image

15. The nearest neighbor coding is intuitive and simple to implement. However, this coding does not take into account certain subtle points. For example, if the closest neighbor is very close to the second closest neighbor, this is not taken into account by this coding.

    A "softmax" type coding would solve this problem by assigning a weight to each cluster.

16. The "sum" pooling is intuitive and uses the frequency of the elements in an image. A "max" pooling could also have been used.

    The interest of the max pooling is to identify the dominant class within an image. This approach can be interesting for one-class image classification or feature detection problems for example.

17. Normalization allows us to have a z vector with a norm equal to 1. This is essential if we want our image comparisons to make sense.

    Indeed, the z vector depends on the number of patches in an image. Without normalization, the z vectors of two images of different size would have different norms.

    This would lead to ineffective comparisons, even if the images were similar.

    Normalization solves this issue. We have chosen to use the L_2 norm but any other norm (ex: L_1, L_∞) would also have worked.