

# Deep Learning Practical Work 1-a and 1-b

## Images descriptors with SIFT and Bag of Words

Arthur Douillard

Asya Grechka

Alexandre Rame

Matthieu Cord

22 September & **TODO**

### Homework

- For all practical works, add all teacher assistants in copy to the mail ([arthur.douillard@lip6.fr](mailto:arthur.douillard@lip6.fr) and [asya.grechka@lip6.fr](mailto:asya.grechka@lip6.fr) and [alexandre.rame@lip6.fr](mailto:alexandre.rame@lip6.fr)).
- Email object must be **[RDFIA] [Homework-1]**.
- After doing 1-c, you will need to finish by yourself at home the whole homework I (1-a and 1-b). You will send by email this homework. You are encouraged to do it by group of two (please write both names in the email!)
  - You need to provide answers to all questions in one single PDF file. You can include inside the PDF code snippets (not a whole file!), graphs, or images that you deemed necessary.
  - Don't write long paragraphs of bullshit, we'll just get tired of it and be more severe.
  - Questions with a "Bonus" mark are optional. Questions with a ★ are worth double points.

Data, code, and PDF version of this file are available at  
<https://rdfia.github.io>

---

## Goals

---

During these first practical works, we are going to prepare the development of an initial image classification model. The goal is to produce an algorithm that is able to classify images from the dataset 15-Scenes. This dataset contains 4485 images belonging to 15 scenes categories (e.g. kitchen, bedroom, street, etc.)

First, in 1-a, we are going to use **SIFT ((Scale-invariant feature transform))** which are local visual descriptors and thus can encode a small *patch* of image. Then, still in 1-a, we'll explore some archetype descriptors that represent recurring patterns to build a **visual dictionary**. Then, we'll aggregate, in 1-b, the SIFT of each image with a **BoW (Bag of Words)** method that express in a compact way an image.

Finally, in 1-c, we'll learn to classify each image, from its BoW representation by using a **SVM (Support Vector Machine)**.

---

## Data · Code · Python

---

During our practical work sessions, we'll use Python3 alongside very popular *packages* in scientific computing and machine learning (numpy, matplotlib, scikit-learn, pytorch, ...). A quick introduction of Python is proposed here <http://webia.lip6.fr/~dancette/deep-learning/python>.

All the coding sessions will be done on Google colab, but you can still download a jupyter notebook from it if you prefer running the code on your local machine. Be aware that we won't help you if you have a problem related to your machine in that case.

## Section 1 – SIFT

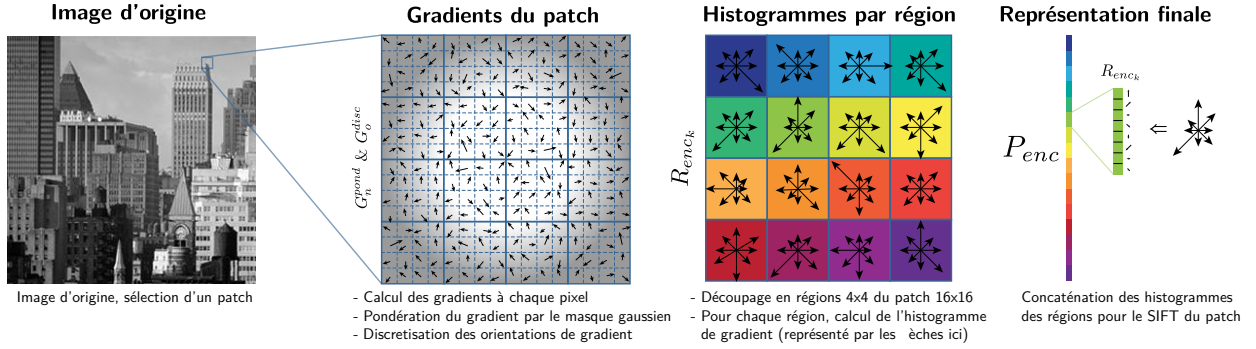


Figure 1: Illustration of the SIFT method applied to a patch.

A SIFT (Scale-invariant feature transform) is a visual and local descriptor. It can transform a small image *patch* of size  $16 \times 16$  pixels into a numerical representation (a vector of 128 dimension in that case) which will be robust to small perturbations and transformations. Meaning that two similar patches will have similar SIFTs.

### 1.1 Computing the gradient of an image

First, we want to compute the gradient of an image at a pixel  $(x, y)$ :

$$G(x, y) = \left[ \frac{\partial I}{\partial x} \quad \frac{\partial I}{\partial y} \right]^\top = [I_x \quad I_y]^\top \quad (1)$$

In practice, the partial derivatives will be approximated by finite differences: in that case, the partial derivatives  $I_x$  et  $I_y$  can be obtained using the convolution (denoted by  $\star$ ) of the image by a kernel  $I_x = I \star M_x$ ,  $I_y = I \star M_y$ . We'll use the following  $3 \times 3$  Sobel kernels:

$$M_x = \frac{1}{4} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \frac{1}{4} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (2)$$

From  $I_x$  and  $I_y$ , we can compute the norm of the gradient  $G_n = \|G\| = \sqrt{I_x^2 + I_y^2}$  and the direction  $G_o$  (code is given for that).

### Questions

1. Show that kernels  $M_x$  and  $M_y$  are separable, i.e. that they can be written  $M_x = h_y \times h_x^\top$  and  $M_y = h_x h_y^\top$  with  $h_x$  and  $h_y$  two vectors of size 3 to determine.
2. Why is it useful to separate this convolution kernel?

## 1.2 Computing the SIFT representation of a patch

Let's define the algorithm that can produce the SIFT representation of an image patch  $P$  of  $16 \times 16$  pixels. The whole method that we will cover is illustrated in the figure 1.

The patch  $P$  is going to be decomposed in 16 sub-regions  $R_i$  of  $4 \times 4$  pixels each. Each region will be encoded as a vector  $R_{enc_k} \in \mathbb{R}^8$ . To do so, we'll use the SIFT descriptor (Scale Invariant Feature Transform) which consists in computing a histogram of gradient directions.

The complete patch will be described by a vector  $P_{enc} \in \mathbb{R}^{128}$  which will be the concatenation of all regions encoding.

### Algorithm

The SIFT algorithm is as follow:

- For each pixel of the patch  $P$ , we compute the image gradients with respect to each pixel will be stored as a gradient direction matrix  $G_o \in \mathbb{R}^{16 \times 16}$  and a gradient norm matrix  $G_n \in \mathbb{R}^{16 \times 16}$ .
- We discretize the gradient direction matrix into 8 bins / values. The value 0 will be associated to the direction  $0^\circ$  (north), the value 1 to the direction  $45^\circ$  (north-east), the value 2 to the direction  $90^\circ$  (east), etc. Thus, we obtain a matrix  $G_o^{disc} \in \mathbb{N}^{16 \times 16}$ .
- We compose a 2d gaussian  $N \in \mathbb{R}^{16 \times 16}$  centered on the center of  $P$  and with standard deviation  $\sigma = 0.5 \times width$  with  $width = 16$  pixels. We weight  $G_n$  by  $N$  to obtain  $G_n^{pond}$ .
- For each region of size  $4 \times 4$ , we compute a histogram of gradient directions  $R_{enc_k} \in \mathbb{R}^8$ . Each bin of the histogram is the sum of the gradient weighted norms for a given direction. For example,  $R_{enc_k}[0]$  is the sum of values of  $G_n^{pond}[i, j]$  such as  $(i, j)$  is in the region  $k$  and that  $G_o^{disc}[i, j] = 0$ .
- We concatenate our 16 vectors  $R_{enc_k}$  to make the descriptor  $P_{enc} \in \mathbb{R}^{128}$ .

We conclude the algorithm by a **post-processing** on  $P_{enc}$ :

- We compute the  $L_2$  norm of the descriptor  $P_{enc}$ . If it is smaller that a threshold of 0.5,  $P_{enc}$  is set to a null vectore and is immediately returned.
- Otherwise, we normalize the descriptor to have a unit euclidean ( $L_2$ ) norm ( $\|P_{enc}\|_2 = 1$ ).
- Finally, values greater than 0.2 will be clamped to 0.2, and then the descriptor is  $L_2$  normalized a second time.

### Practice

In the beginning of the colab, a file `tools.py` is downloaded which contains many useful functions that you can use. Write your code where indicated in the colab notebook.

- Write the function `compute_grad(I)` which computes the gradient of the input image and returns  $I_x$  and  $I_y$ . You'll use the function `conv_separable(I, ha, hb)` which computes  $I \star M$  with  $M = h_a \times h_b^\top$ .
- Write the function `compute_grad_mod_ori(I)` which return  $G_n$  and the discretized orientations  $G_o^{disc}$  of the input image. You'll use `compute_grad(im)` and `compute_grad_ori(Ix, Iy, Gn)`.

- From now on, we don't work anymore on the whole image but rather on a patch of size  $16 \times 16$ . Write the function `compute_sift_region(Gn, Go, mask=None)` which returns the representation  $P_{enc}$  of a patch given  $G_n$  and  $G_o^{disc}$  and also an optional gaussian mask which can help produce  $G_n^{pond}$  if provided. Note that this function will contain the post-processing of SIFT. The gaussian mask to give in input is generated by the function `gaussian_mask()` which is provided. Test your function manually and visually thanks to the function `display_sift_region(I, compute_grad_mod_ori, compute_sift_region, x, y)` which takes an image in input, coordinates  $x, y$  where to cut a patch, and your computing functions.

## Questions

3. What is the goal of the weighting by gaussian mask?
4. ★ Explain the role of the discretization of the directions.
5. Justify the interest of using the different post-processing steps.
6. ★ Explain why SIFT is a reasonable method to describe a patch of image when doing image analysis.
7. Interpret the results you got in this section.

## 1.3 Computing SIFTs on the image dataset

To compute the SIFT representation of an image, we must choose center points whose surrounding will be used to compute the SIFTs. Multiple methods<sup>1</sup> exist to efficiently choose them but a simple way is to densely sample them. i.e. to take a patch every  $n$  pixels. In our case, we will sample one  $16 \times 16$  patch every 8 pixels. Thus note that the sampled patches will have some overlap!

- Write the function `compute_sift_image(I)` which computes the SIFTs of an image. The function `x, y = dense_sampling(I)` provides two lists `x` and `y`. The upper-left coordinates of every patches are the pairs  $(x_i, y_i)$ . The SIFTs are stored in a numpy array of size  $n_x \times n_y \times d_{SIFT}$  with  $n_x$  the number of coordinates alongside the  $x$ -axis and  $n_y$  alongside the  $y$ -axis, and  $d_{SIFT}$  the size of a SIFT.

---

## Section 2 – Visual Dictionary

---

### Theory

We are going to use all SIFTs descriptors  $P_{enc}$  extracted from our dataset to compute a **visual dictionary**. A *word* in this dictionary is somekind of SIFT-like descriptor likewise  $c_m \in \mathbb{R}^{128}$ . The goal of this dictionary is to represent as fidely as possible all the SIFTs but using a limited amount of words. Thus, those words will be frequent patterns in the extracted SIFTs.

Concretely, our goal is to determine  $K$  centers (clusters)  $c_m$  of the SIFT space that minimize a distance from the actual SIFTs  $x_i$ . This is our objective:

$$\min_{c_m} \sum_{m=1}^M \sum_{x_i \in C_m} \|x_i - c_m\|_2^2 \quad (3)$$

---

<sup>1</sup>for the curious see the *Difference of Gaussian* at [https://docs.opencv.org/master/da/df5/tutorial\\_py\\_sift\\_intro.html](https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html)

A classic, yet efficient, way to solve this problem is to use the **K-Means** algorithm whose behavior is rather simple:

- Initialize  $M$  centers  $c_m$  from randomly-sampled points  $x_i$ .
- Alternate until convergence:
  - Assign to each point  $x_i$  the closest center
  - Re-compute the centers  $c_m$  as the average of their assigned points

## In Practice

With the function `compute_load_sift_dataset`, we can obtain a list where each element is a list of the SIFTs of a single image.

You must complete the function `compute_visual_dict` so that it computes the visual dictionary, i.e. the  $M$  clusters. To do so, we are going to use scikit-learn : <http://scikit-learn.org/stable/modules/clustering.html#k-means>

Remember to add a dummy vector filled with zeros as “cluster” which will represent the null SIFTs.

Choose (but you can try other values) 1000 clusters to determine with K-Means. At the end, you should have 1001 clusters by adding the zero cluster.

You can then use the function `compute_or_load_vdict` to compute the clusters on all the SIFTs.

The function `get_regions_and_sifts` will get you the patches of an image alongside the corresponding SIFTs. To analyse the clusters of a visual dictionary, we could search for the image patches that are their closest (in the SIFT space!). Try different things to get a deeper understanding of your visual dictionary. For example, you could look for the 50 patches closest to a particular cluster, or for each cluster display the closest patch, etc. To help you, the function `display_images` can display regions.

## Questions

8. ★ Justify the need of a visual dictionary for our goal of image recognition that we are currently building.
9. Considering the points  $\{x_i\}_{i=1..n}$  assigned to a cluster  $c$ , show that the cluster's center that minimize the dispersion is the barycenter (mean) of the points  $x_i$ :

$$\min_c \sum_i ||x_i - c||_2^2 \quad (4)$$

10. In practice, how to choose the “optimal” number of clusters?
11. Why do we create a visual dictionary from the SIFTs and not directly on the patches of raw image pixels?
12. Comment the results you get.

## Section 3 – Bag of Words (BoW)

### Theory

In this section, the goal is to obtain a numerical representation of each image that will help us later to classify this image (e.g. tell what does the image represent).

For now, we have for each image an ensemble of local SIFT descriptors (not necessarily the same amount per image), and a visual dictionary made of “mean” descriptors.

The goal of BoW is to summarize all the local descriptors of an image into a global descriptor with the help of the visual dictionary.

For an image  $I$ , we have all its descriptors  $x_i$  in a matrix  $X \in \mathbb{R}^{n_{patch} \times d}$ . We also have the visual dictionary under a matrix form  $C \in \mathbb{R}^{M \times d}$ .

The creation of the BoW descriptor of the image  $I$  is done in two steps:

- The **encoding** step: each descriptors  $x_i$  is encoded as a vector  $h_i \in \mathbb{R}^M$  by using the visual dictionary. We get a new matrix  $H \in \mathbb{R}^{n_{patch} \times M}$ .  
In our case, we will use a “nearest-neighbors” encoding: the vector  $h_i \in \mathbb{R}^M$  is a one-hot vector indicating which word of the dictionary is the closest:

$$h_i[j] = \begin{cases} 1 & \text{if } j = \arg \min_k \|x_i - c_k\|^2 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

- The *pooling* step: we aggregate the  $h_i$  over all local descriptors to obtain a vector  $z \in \mathbb{R}^M$  that globally describe the image.  
In our case,  $z$  will simply be the sum of  $h_i$ .

The descriptors will finally be normalized with a euclidean norm ( $L_2$ ).

### Pratique

Write a function `compute_feats` that takes in input the visual dictionary matrix  $C$  and the matrix  $X$  containing all SIFTs of an image. This function will return the BoW representation  $z$  of the image.

Use the code given in the notebook to visualize the BoW representation.

### Questions

- ★ Concretely, what does the vector  $z$  represent of the image?
- Show and discuss the visual results you got.
- What is the interest of the nearest-neighbors encoding? What other encoding could we use (and why)?
- What the interest of the sum pooling? What other pooling could we use (and why)?
- What is the interest of the  $L_2$  normalization? What other normalization could we use (and why)?