Inpainting problems

INSA Rennes - 5MA - Parcimonie

1 Introduction

In this TP, we consider images corrupted by some tag. The objective is to remove the tag by leveraging properties of natural images. This task is usually called an *inpainting* problem.

Figure 1: Tagged image.

The notebook `tp.ipynb` is the main file of this TP. Most of the code is already written and the "`#TODO`" comments locate parts to be filled. You are encouraged to read and try to understand the code that is provided with this TP. It should be enough documented.

2 Patches

Let $\mathbf{z} \in [0, 255]^{m_1 \times m_2 \times 3}$ be an RGB image. We can extract from $\mathbf{z}$ a collection of small areas of pixels called *patches*. They consists in small rectangles of pixels of dimension $p_1 \times p_2 \times 3$, where $p_i \ll m_i$. Patches are constructed from the image by taking overlapping areas in order to have redundancy between the different patches collected. From a formal point of view, we are not interested in the fact that a patch is also an image. We rather use patches because they usually admit a sparse representation through some well chosen dictionaries.

When we extract N patches from an image $\mathbf{z}$, we denote $\mathbf{P} \in \mathbf{R}^{3p_1p_2 \times N}$ the matrix obtained by concatenating all vectorized patches. The function

```
P = patches(z,[p1,p2,3],skip)
```

allows to perform such operation. The argument `skip` is the number of overlapping pixels in horizontal or vertical translation between consecutive patches. It is possible to get the image back from its patches using the function

```
z = patch2image(P,[p1,p2,3],skip,[m1,m2,3])
```

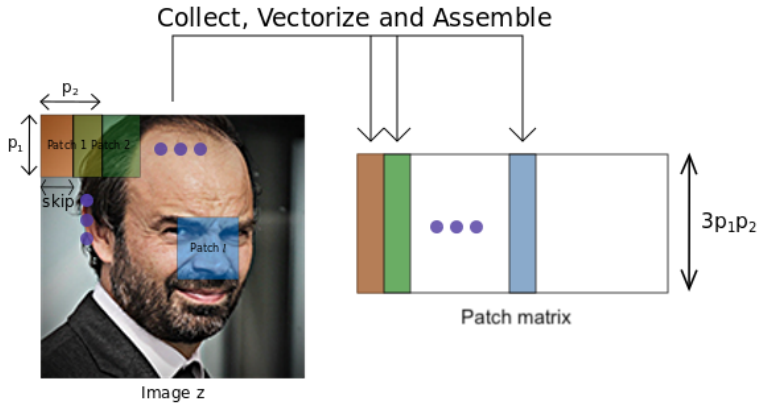

Figure 2: Patch extraction.

3 Inpainting with predefined dictionary

Patches extracted from natural images usually admit a sparse representation in some well chosen dictionaries $\mathbf{D}$. In this part, we consider the celebrated DCT (Discrete Cosine Transform) dictionaries that are strongly linked to Fourier transforms. You need not understand how they are constructed but you can dig into this subject if you want. DCT dictionaries can be generated using

```
D = genDCT(dims,fact)
```

The argument `dims` is the dimensions of the patches and `fact` defines the over-completeness of the DCT dictionary, that is how precise (but also larger) it is.

3.1 Asserting the sparse representation

In a first place, we will verify that the patches of the **un-tagged** picture indeed admit a sparse representation through a DCT dictionary $\mathbf{D} \in \mathbf{R}^{p\times d}$. To do so, we consider the problem

$$\min_{\mathbf{X}\in\mathbf{R}^{d\times n}} \ \tfrac{1}{2}\|\mathbf{P}-\mathbf{D}\mathbf{X}\|_F^2 \quad \text{s.t.} \quad \|\mathbf{X}_\ell\|_0 \leq k, \ \forall \ell \in \{1,\dots,n\} \tag{1}$$

where $\mathbf{P} \in \mathbf{R}^{p\times n}$ stores the patches. In few word, the above problem aims to find a sparse representation of all the patches (*i.e.,* all the columns of $\mathbf{P}$) through a unique dictionary $\mathbf{D}$. Since this problem is NP-hard, we rather approximate its solution using the ISTA algorithm previously seen in the lectures.

Question 1. Read and the `ista.py` template file and try to understand it.

Question 2. Complete the `soft_threshold(x,gamma)` function.

Question 3. Complete the ISTA implementation.

Question 4. For different k and λ, try to reconstruct the original image.

3.2 Dealing with missing pixels

To remove the tagged pixels from the image, you can just fill them with the ones of the reconstructed image obtained from the ISTA algorithm. To handle missing pixels the latter method, the simplest strategy is just to ignore them. Stated otherwise, **you have to filter which pixels to use in all computations involving Y and P in ISTA**.

Question 5. Modify the ISTA function so that you can pass an argument informing where missing pixels are.

Question 6. Update the algorithm to handle missing pixels.

Question 7. Try to calibrate k and λ in order to remove the tag from the image.

Question 8 (bonus). DCT matrices are orthogonal when they are squared. Propose a more efficient inpainting strategy with such dictionaries.

4 Learning an appropriate dictionary

Instead of fixing $\mathbf{D}$, we can try to construct a more suitable dictionary than the DCT one from training images. In this section, we study a simple *dictionary learning* algorithm allowing to perform such task.

To learn a suitable dictionary $\mathbf{D}$, the first thing to do is to construct some training data $\tilde{\mathbf{P}}$ similar to the real target $\mathbf{P}$. For instance, $\tilde{\mathbf{P}}$ can be the concatenation of patches corresponding to several faces. We can then solve

$$\min_{\mathbf{X},\mathbf{D}} \ \tfrac{1}{2}\|\tilde{\mathbf{P}} - \mathbf{D}\mathbf{X}\|_F^2 \quad \text{s.t.} \quad \|\mathbf{X}_\ell\|_0 \leq k, \ \forall \ell \tag{2}$$

Stated otherwise, we try to find both a tailored dictionary $\mathbf{D}$ and a sparse representation of the training data $\tilde{\mathbf{P}}$ through it. Obviously, there are infinitely many solutions to (2). Yet, we can leverage the MOD algorithm that aims to construct an "interesting" one. The algorithm alternates between minimization with respect to $\mathbf{D}$ for a fixed $\mathbf{X}$ and a minimization respect to $\mathbf{X}$ for a fixed $\mathbf{D}$ in the problem (2). Note that

- The minimization of (2) respect to $\mathbf{D}$ is a quadratic problem.
- The minimization of (2) respect to $\mathbf{X}$ can be roughly done with ISTA.

The algorithm outputs a dictionary $\mathbf{D}$ that may be more suited to our problem than a DCT one.

Question 9. Read the `mod.py` template file and try to understand it.

Question 10. Complete the implementation of the MOD algorithm.

Question 11. Try the learning of a suitable dictionary and compare it to the results with the DCT one. You can try to tune the different parameters.

Question 12 (bonus). There are other images in the `img` directory than faces one. Try to learn a dictionary with them.

Question 13 (bonus). Replace the ISTA algorithm by another one seen in the previous TP (OMP, IHT, ...).

Question 14 (bonus). Try to reconstruct the missing pixels by taking the average color of neighbors. Does this produces satisfying results ?

Credits for this TP : Jeremy Cohen