ATHENS : CTU 15
Digital signal and image processing with applications

Two-dimensional discrete Fourier Transform in image analysis

Arnaud Capitan - Télécom Paris

Two-dimensional discrete Fourier Transform

For a 2-dimensional Fourier Transform of a discrete two-dimensional array of size (N,M) :

$$\{x(n,m)\}_{n=0,m=0}^{N-1,M-1}$$

- Direct DFT:

$$X(k,l) = \sum_{m=0}^{M-1} \underbrace{\left(\sum_{n=0}^{N-1} x(n,m) e^{-ikn\frac{2\pi}{N}} \right)}_{\text{1D DFT}} e^{-ilm\frac{2\pi}{M}}$$

- Inverse DFT:

$$x(n,m) = \frac{1}{M} \sum_{l=0}^{M-1} \underbrace{\left(\frac{1}{N} \sum_{k=0}^{N-1} X(k,l) e^{ikn\frac{2\pi}{N}} \right)}_{\text{1D IDFT}} e^{ilm\frac{2\pi}{M}}$$

Basic properties of Fourier Transform

We also denote the Discrete Fourier Transform :

$$\mathcal{F}(u) = \hat{u}$$

Basic properties :

DFT of a convolution

$$\mathcal{F}(u * v) = \hat{u}\hat{v}$$

DFT of a product

$$\mathcal{F}(uv) = \frac{1}{N}\hat{u} * \hat{v}$$

Parseval Equality

$$\|u\|_2 = \frac{1}{\sqrt{N}}\|\hat{u}\|_2$$

Basic properties of Fourier Transform

Basic properties :

Expression of a convolution

$$(u * v)_n = \sum_{m \in \mathbb{Z}} u_m v_{n-m}$$

Since we have this property :

$$\mathcal{F}(u * v) = \hat{u}\hat{v}$$

We can also express the convolution with DFT :

$$u * v = \mathcal{F}^{-1}(\hat{u}\hat{v})$$

Image reconstruction - Introduction

When an image is captured by a physical process, it is often deteriorated :

- Noise from the sensor
- Blur from the camera lens
- Blur from movement

We can express those detoriations with a simplified model :

$$v = Au_0 + w$$

- v is the image captured
- A is the blur matrix
- w is a noise
- u0 is the image we want to find

Image reconstruction - Introduction

When If we consider a uniform blur, we can express it as a convolution :

$$Au = k * u$$

We can express those detoriations with a simplified model :

$$v = k * u_0 + w$$

- v is the image captured
- k is the blur kernel
- w is a noise
- u0 is the image we want to find

We wil assume that we know the blur kernel, we won't tackle the blind deconvolution problem.

Image reconstruction - Finding blur kernel

To find the blur kernel, we can consider using Richardson-Lucy algorithm, which is an iterative algorithm that converges.

There are few conditions required to have an optimal solution :

- The more iterations we use, the better the sharpness of the kernel but we risk amplifying noise. Less than 100 iterations work best.
- Edge effects need to be dealt with, often with a zero padding.

Image reconstruction - Optimization problem

This single problem may have multiple solutions ; since most of of them are unusable, we need to add another constraint, such as the smoothness of the output image :

We can this the regulation function.

$$R(u) = \sum_{\mathbf{x} \in \Omega} |\partial_1 u(\mathbf{x})|^2 + |\partial_2 u(\mathbf{x})|^2$$

$$\begin{cases} \partial_1 u(x, y) = u(x+1, y) - u(x, y) \\ \partial_2 u(x, y) = u(x, y+1) - u(x, y) \end{cases}$$

$$\begin{cases} k_1(0, 0) = -1 \\ k_1(-1, 0) = 1 \end{cases}$$

$$\partial_1 u = k_1 * u$$

Image reconstruction - Optimization problem

This single problem may have multiple solutions ; since most of of them are unusable, we need to add another constraint, such as the smoothness of the output image.

We call this the regulation function.

Tychonov regularization

$$R(u) = \sum_{\mathbf{x}\in\Omega} |\partial_1 u(\mathbf{x})|^2 + |\partial_2 u(\mathbf{x})|^2$$

Total variation regularization

$$\mathrm{TV}(u) = \|\nabla u\|_1 = \sum_{\mathbf{x}\in\Omega} \|\nabla u(\mathbf{x})\| = \sum_{\mathbf{x}\in\Omega} \sqrt{|\partial_1 u(\mathbf{x})|^2 + |\partial_2 u(\mathbf{x})|^2}.$$

Image reconstruction - Optimization problem

The objective is now to find u that minimizes :

$$F(u) = \frac{1}{2}\|u - v\|^2 + \lambda R(u)$$

With R(u) a regulation function, λ>0 its weight,
||u-v||² the data fidelity term

To compute u, we use gradient descent to optimize F(u).

Image reconstruction - Example

We will consider the following blur kernel :

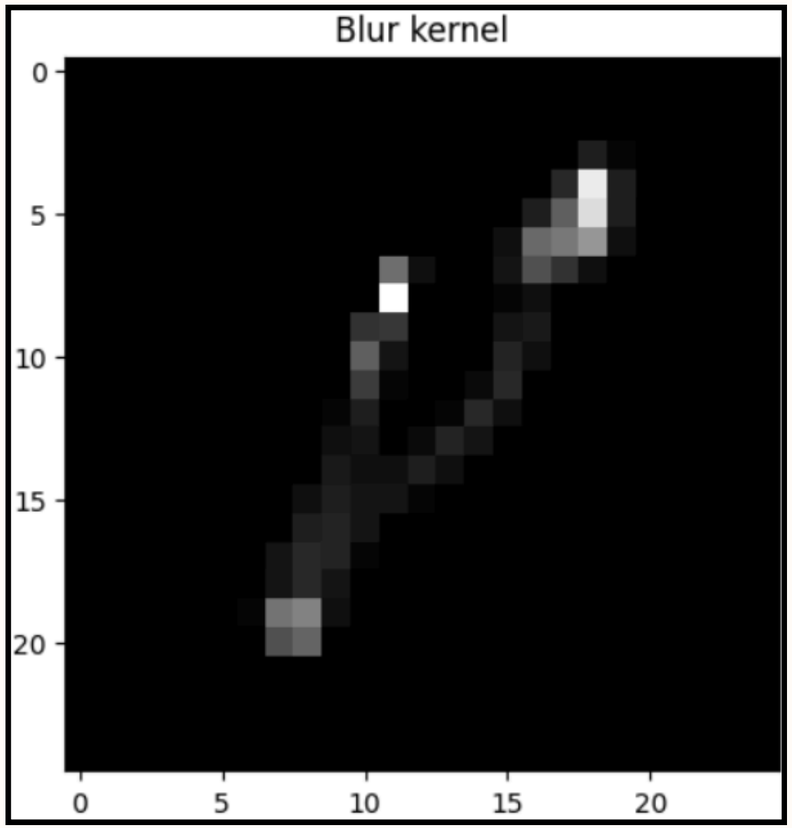
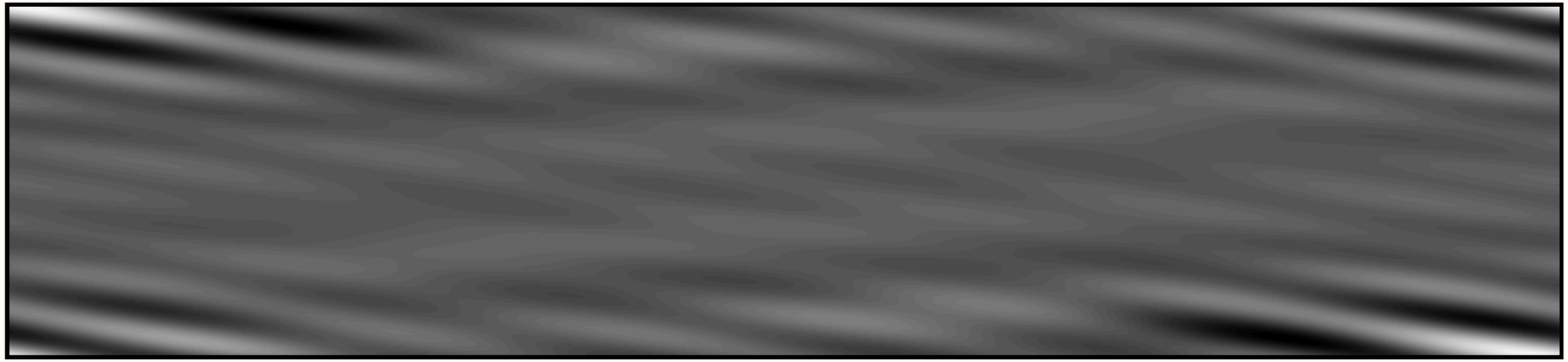


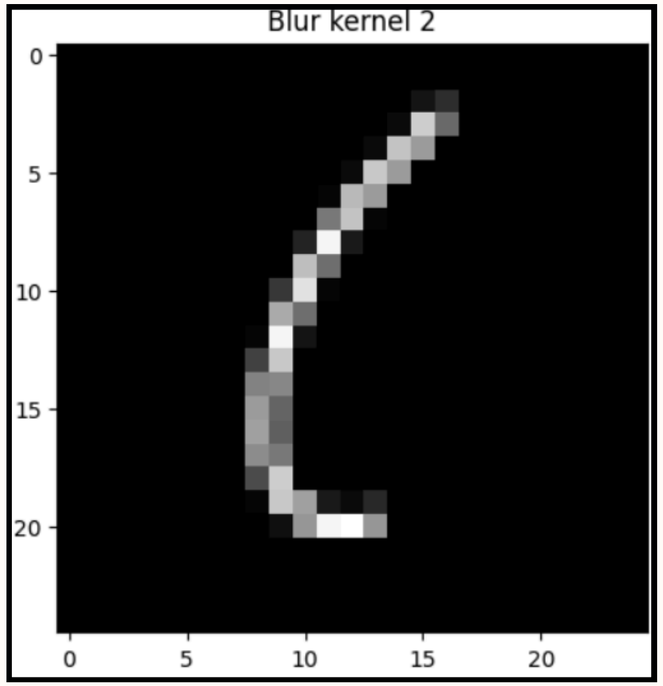
Image reconstruction - Example

Doing the discrete fourier transform of blur kernels often gives texture :



Texture - Example

Doing the discrete fourier transform of blur kernels often gives texture :



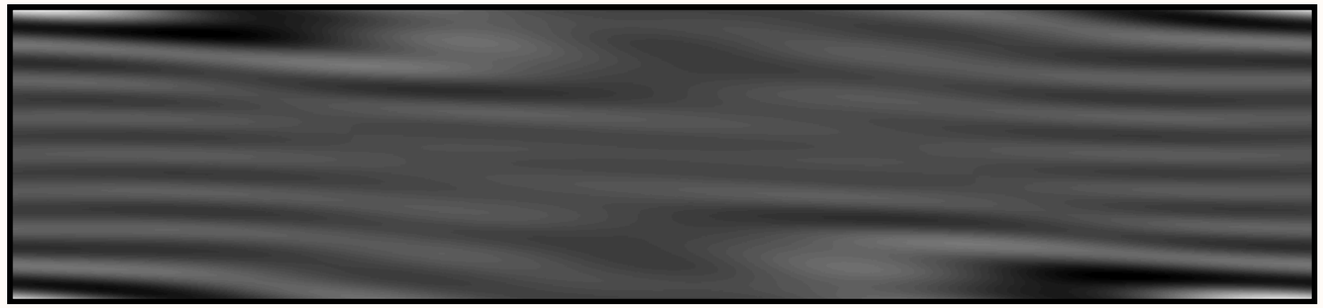


Image reconstruction - Example

We will consider the following image :

Image reconstruction - Example

The noise used is a Gaussian Noise of variance 1 with weight 0.01

We get the blurred image v :

Image reconstruction - Example

Using gradient descent with Tychonov regulation,

Image reconstruction - Example

Using gradient descent with Total variation regulation,

Image reconstruction - Example

Thanks for listening !