Inverse Problems 1: convolution and deconvolution

Lesson 11: Convolution and deconvolution in 2D

Luca Ratti

October 8, 2019

University of Helsinki

Table of contents

Definition and examples

Signals in 2D, images and matrices

A **two-dimensional signal** is any scalar quantity depending on two variables. Examples: a 1D signal evolving in time (x and t), a function of two spatial variables (x and y).

Signals in 2D, images and matrices

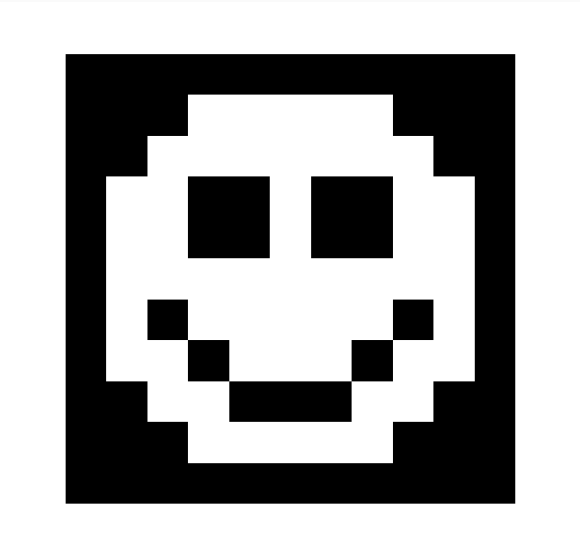
A **two-dimensional signal** is any scalar quantity depending on two variables. Examples: a 1D signal evolving in time (x and t), a function of two spatial variables (x and y).

We can represent 2D signals as bidimensional arrays (matrices)

Signals in 2D, images and matrices

A **two-dimensional signal** is any scalar quantity depending on two variables. Examples: a 1D signal evolving in time (x and t), a function of two spatial variables (x and y).

We can represent 2D signals as bidimensional arrays (matrices)

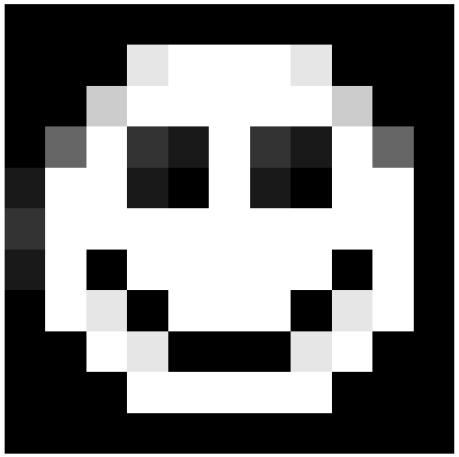


$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Signals in 2D, images and matrices

A **two-dimensional signal** is any scalar quantity depending on two variables. Examples: a 1D signal evolving in time (x and t), a function of two spatial variables (x and y).

We can represent 2D signals as bidimensional arrays (matrices)



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 1 & 1 & 1 & 1 & 1 & 0.8 & 0 & 0 \\ 0 & 0.4 & 1 & 0.2 & 0.1 & 1 & 0.2 & 0.1 & 1 & 0.4 & 0 \\ 0.1 & 1 & 1 & 0.1 & 0 & 1 & 0.1 & 0 & 1 & 1 & 0 \\ 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0.1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0.9 & 0 & 1 & 1 & 1 & 0 & 0.9 & 1 & 0 \\ 0 & 0 & 1 & 0.9 & 0 & 0 & 0 & 0.9 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

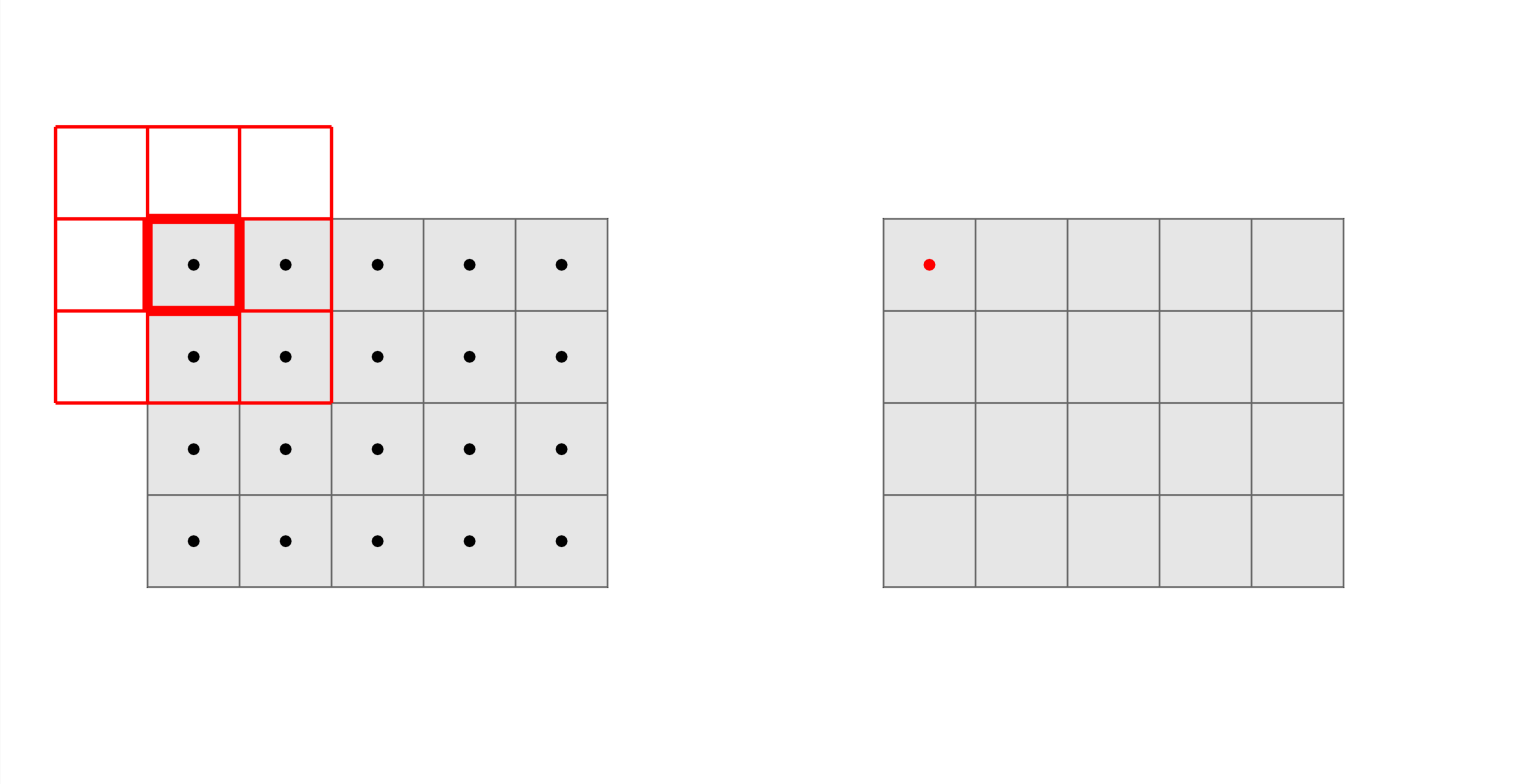
Signals in 2D, images and matrices

A **two-dimensional signal** is any scalar quantity depending on two variables. Examples: a 1D signal evolving in time (x and t), a function of two spatial variables (x and y).

We can represent 2D signals as bidimensional arrays (matrices)

Convolution - graphical intuition

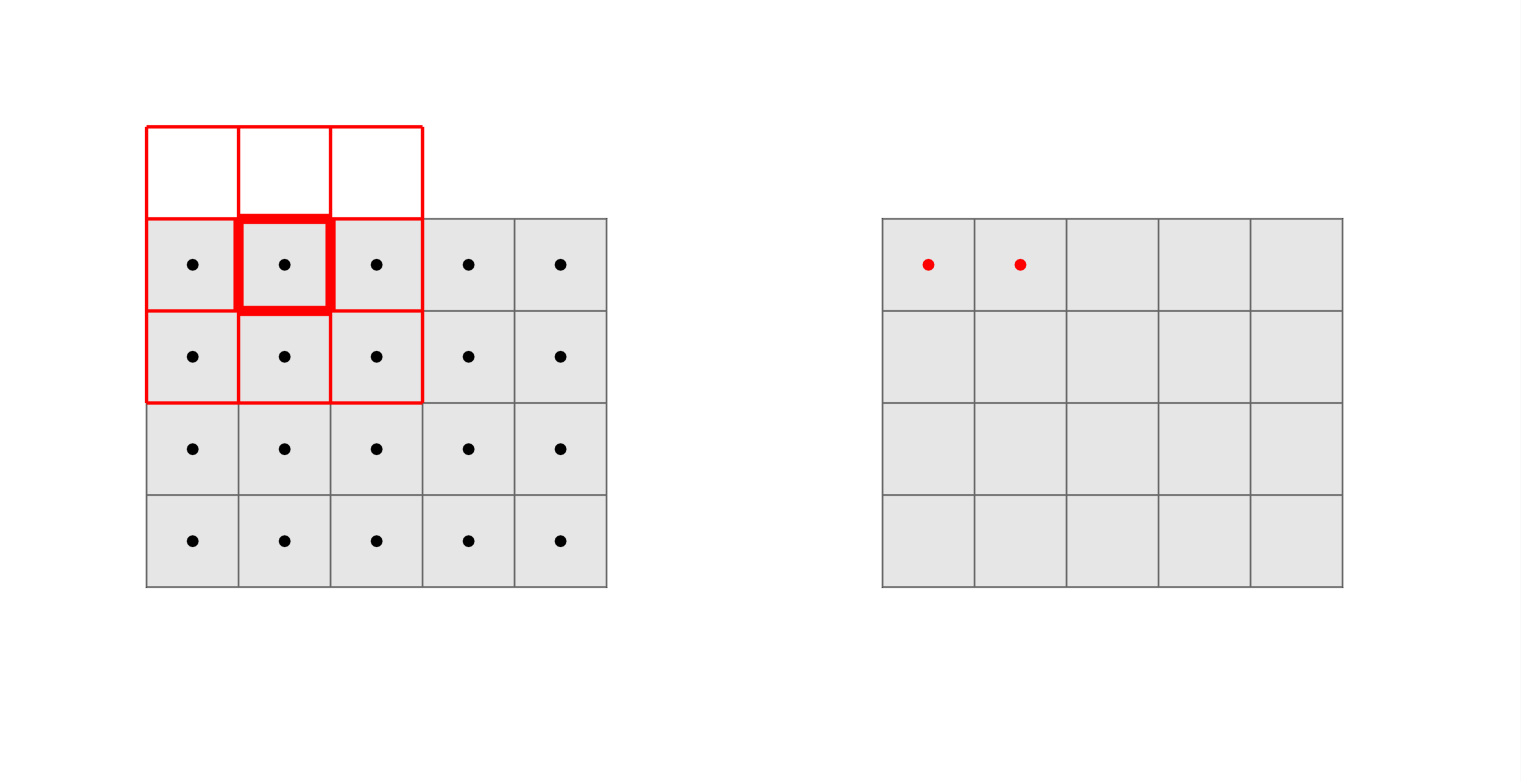
Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



2D convolution

Convolution - graphical intuition

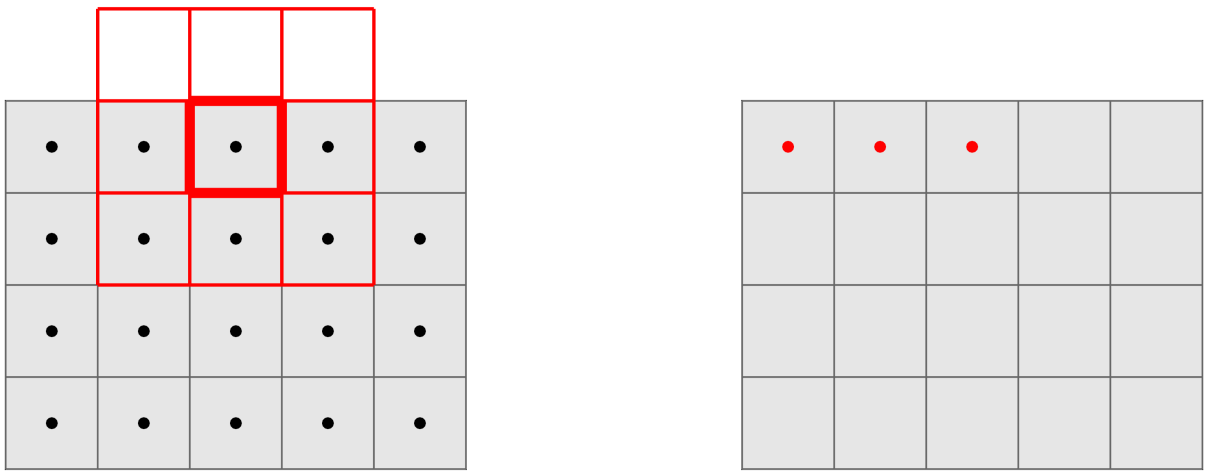
Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



2D convolution

Convolution - graphical intuition

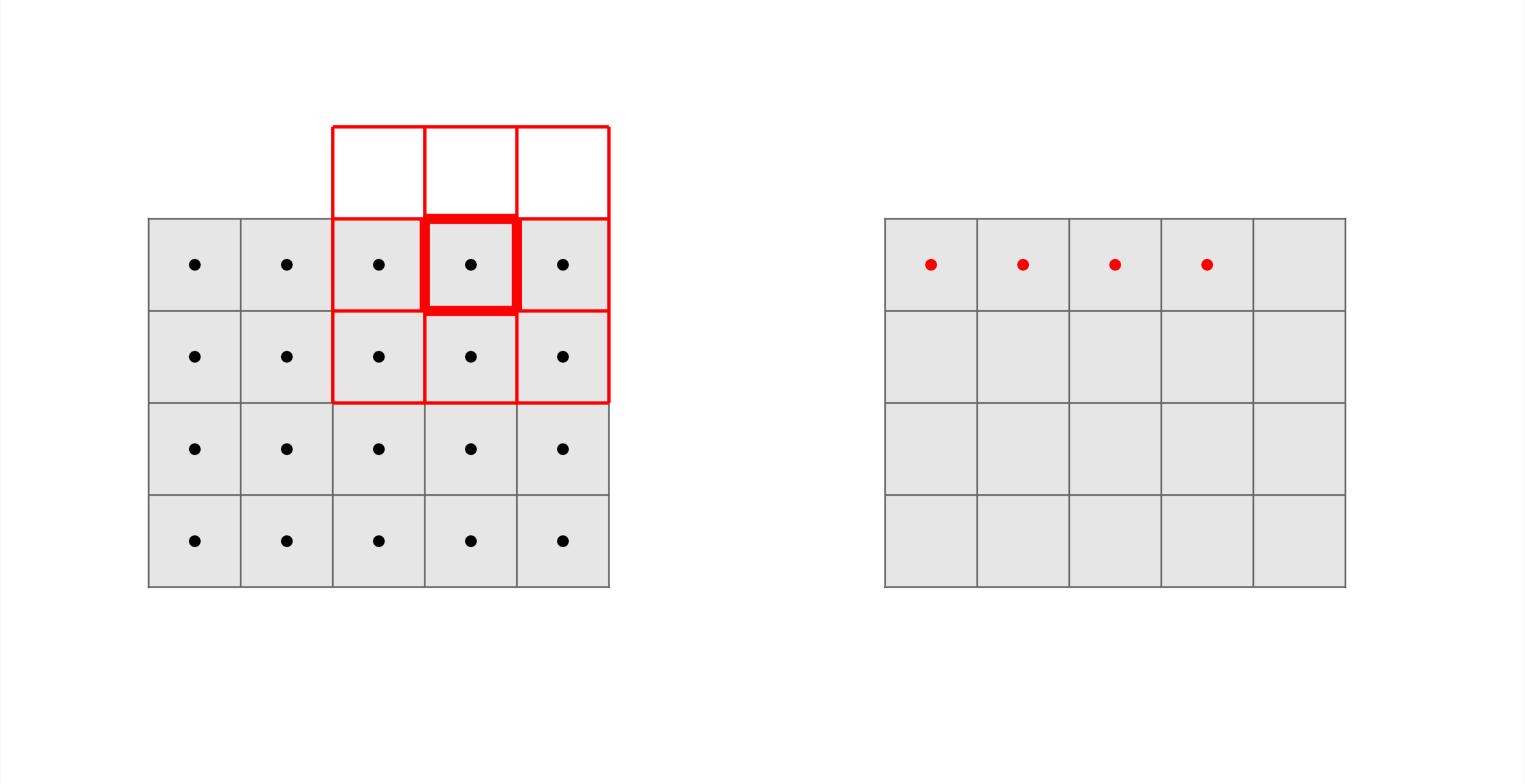
Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



2D convolution

Convolution - graphical intuition

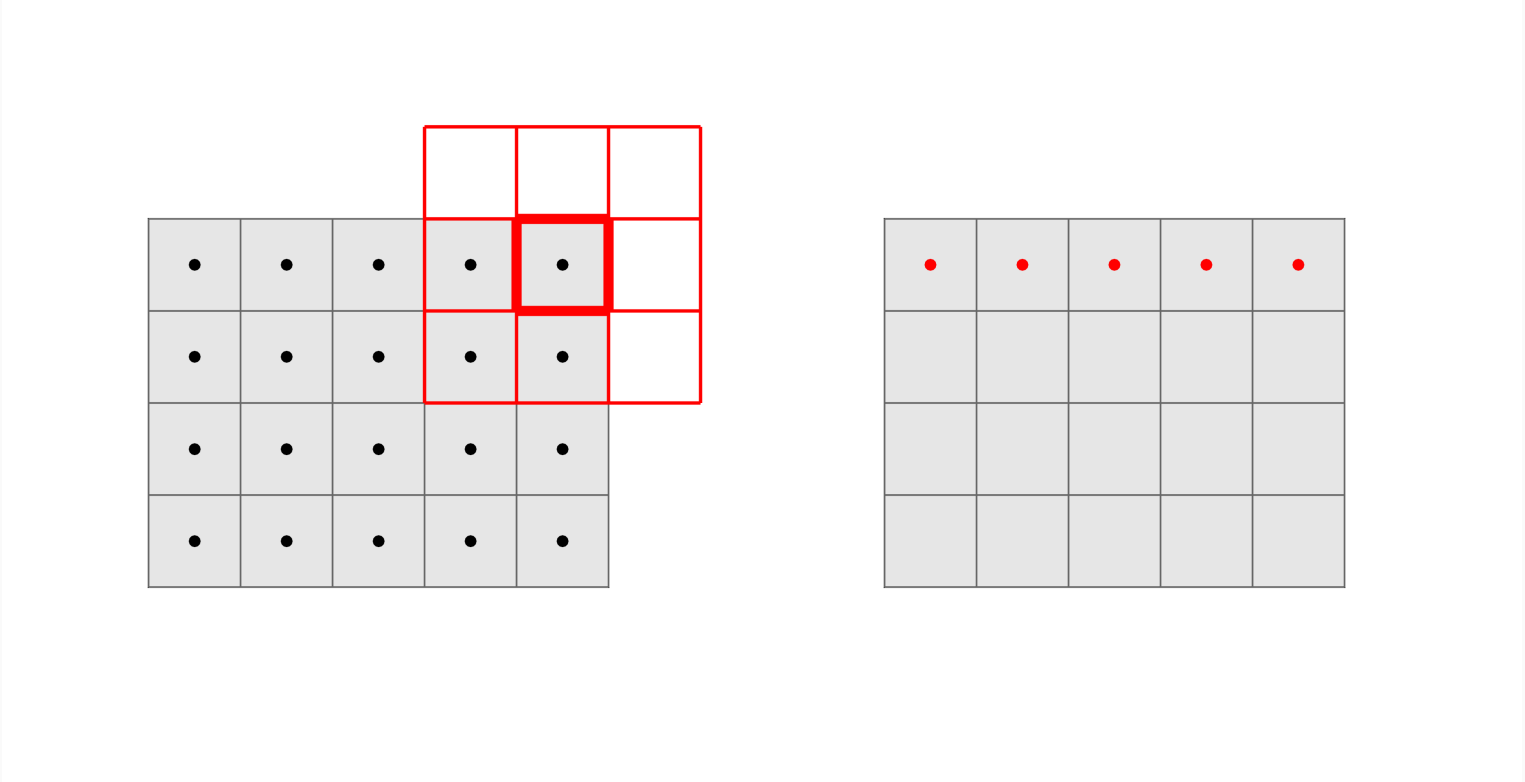
Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



2D convolution

Convolution - graphical intuition

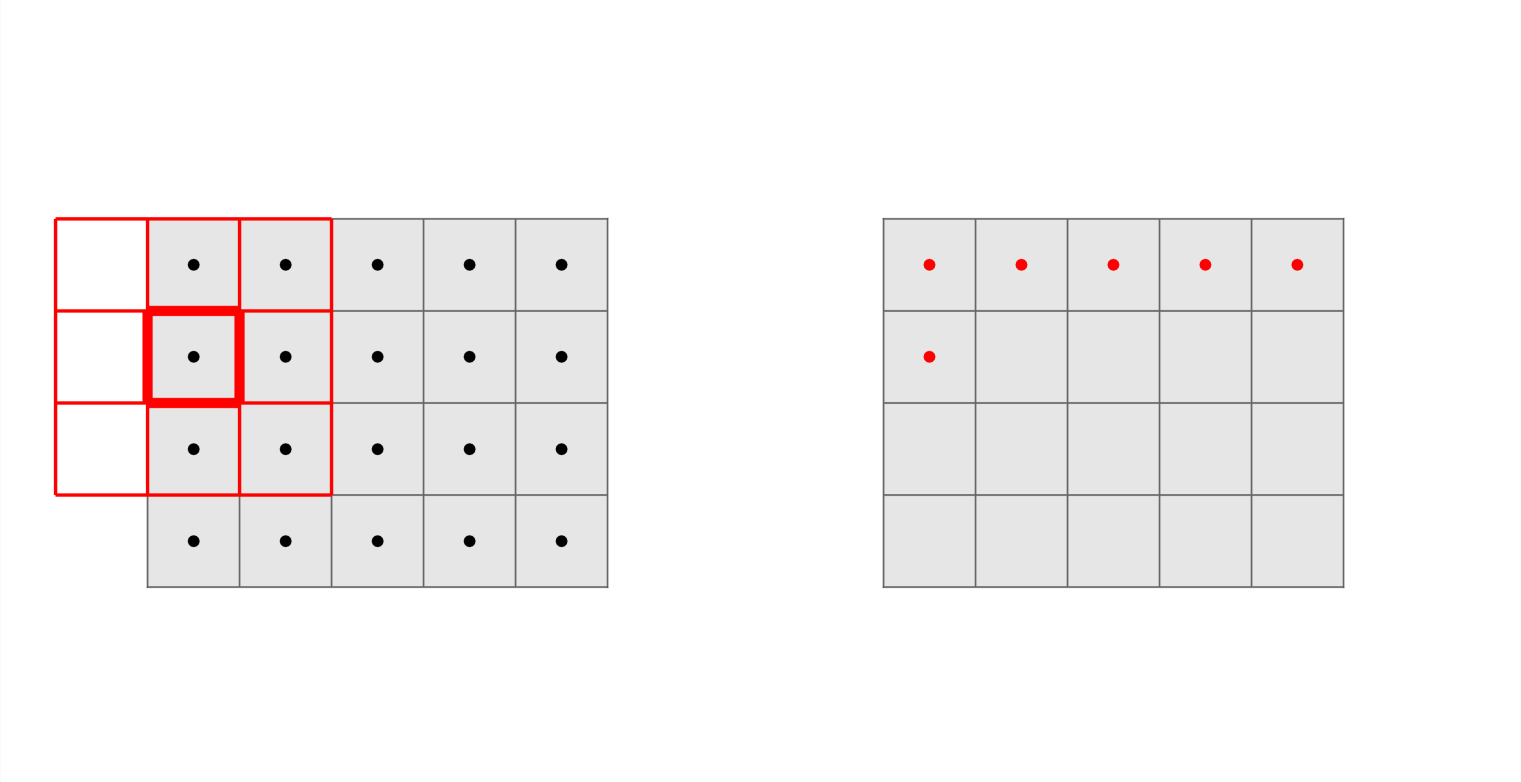
Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



2D convolution

Convolution - graphical intuition

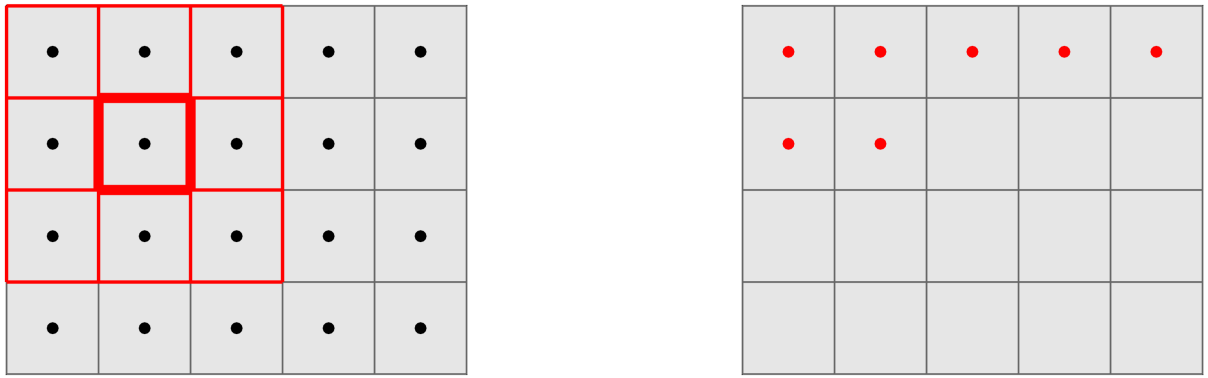
Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



2D convolution

Convolution - graphical intuition

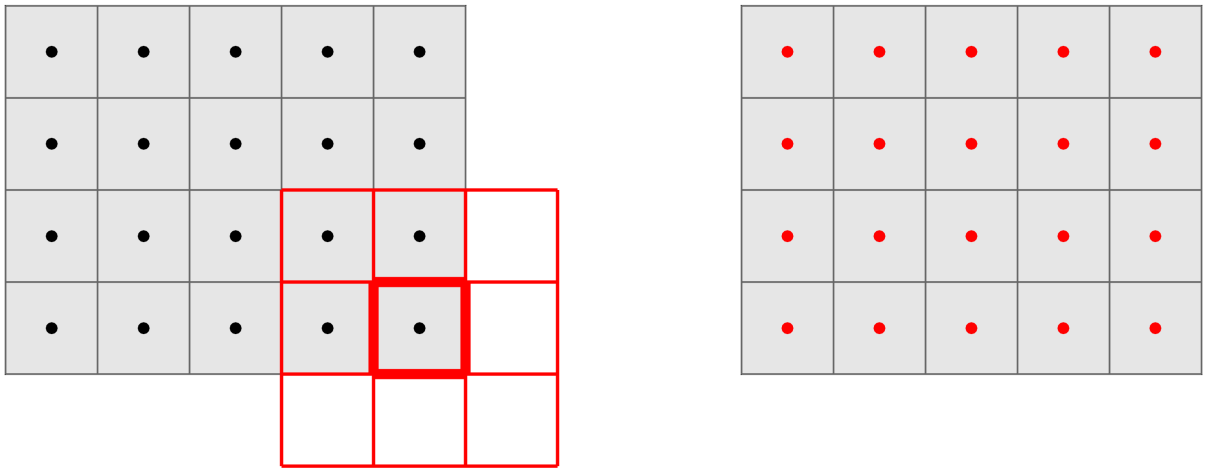
Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



2D convolution

Convolution - graphical intuition

Convolution: the value of the blurred picture in a pixel is influenced by the respective one in the original picture and by the points close to it.



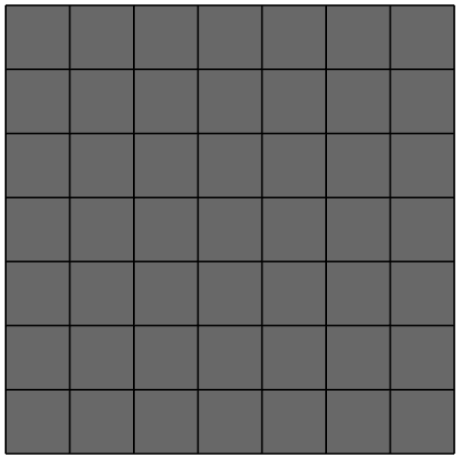
2D convolution

Point Spread Function

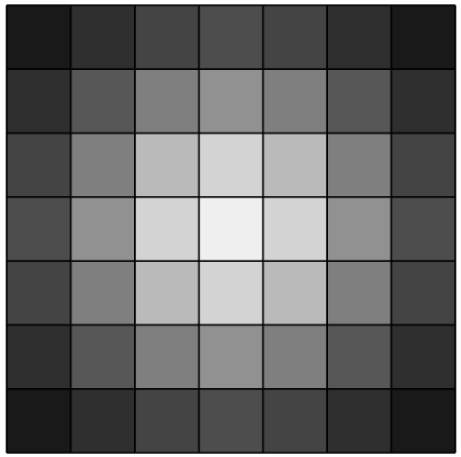
In 2D, the **Point Spread Function** can be interpreted as a matrix reporting the weights with which the points close to a pixel influence the convolution.

Point Spread Function

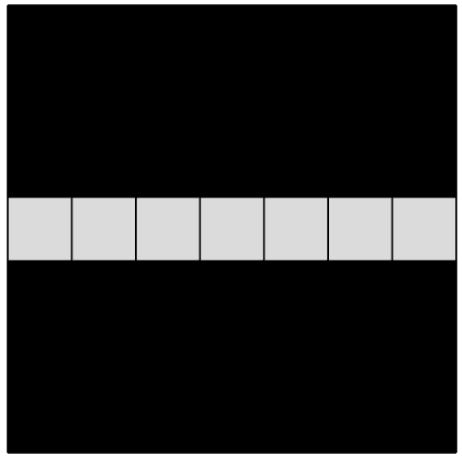
In 2D, the **Point Spread Function** can be interpreted as a matrix reporting the weights with which the points close to a pixel influence the convolution.
Some of the most important examples (please note: non-matching grayscales!)



Average PSF



Gaussian PSF



Motion PSF

A complicated formula

Let $F \in \mathbb{R}^{n \times m}$ be a 2D signal. Let $P \in \mathbb{R}^{q \times r}$ be the PSF.

Normalization: suppose $\sum_{i=1}^{q} \sum_{j=1}^{r} P_{i,j} = 1$

A complicated formula

Let $F \in \mathbb{R}^{n \times m}$ be a 2D signal. Let $P \in \mathbb{R}^{q \times r}$ be the PSF.

Normalization: suppose $\sum_{i=1}^{q} \sum_{j=1}^{r} P_{i,j} = 1$

For simplicity, let q and r be odd numbers: $q = 2\nu + 1$, $r = 2\mu + 1$. Rename the indices of P according to the following order:

$$\begin{bmatrix} P_{-\nu,-\mu} & .. & P_{-\nu,0} & .. & P_{-\nu,\mu} \\ .. & & .. & & .. \\ P_{0,-\mu} & .. & P_{0,0} & .. & P_{0,\mu} \\ .. & & .. & & .. \\ P_{\nu,-\mu} & .. & P_{\nu,0} & .. & P_{\nu,\mu} \end{bmatrix}$$

A complicated formula

Let $F \in \mathbb{R}^{n\times m}$ be a 2D signal. Let $P \in \mathbb{R}^{q\times r}$ be the PSF.

Normalization: suppose $\sum_{i=1}^{q}\sum_{j=1}^{r} P_{i,j} = 1$

For simplicity, let q and r be odd numbers: $q = 2\nu + 1$, $r = 2\mu + 1$. Rename the indices of P according to the following order:

$$\begin{bmatrix} P_{-\nu,-\mu} & .. & P_{-\nu,0} & .. & P_{-\nu,\mu} \\ .. & & .. & & .. \\ P_{0,-\mu} & .. & P_{0,0} & .. & P_{0,\mu} \\ .. & & .. & & .. \\ P_{\nu,-\mu} & .. & P_{\nu,0} & .. & P_{\nu,\mu} \end{bmatrix}$$

Convolution formula

The convolved signal $C = P * F$ is a matrix in $\mathbb{R}^{n\times m}$ s.t.

$$C_{i,j} = \sum_{\ell=-\nu}^{\nu}\sum_{k=-\mu}^{\mu} P_{\ell,k} F_{i-\ell,j-k}.$$

We always consider zero padding for the missing values of F.

Examples

Here are some examples of convolution kernels acting on a picture:

Original picture

Convolution with 9x9 Gaussian filter

Examples

Here are some examples of convolution kernels acting on a picture:

Original picture

Convolution with 9x9 average filter

Examples

Here are some examples of convolution kernels acting on a picture:

Original picture

Convolution with 9x9 motion filter

Implementation

Example 1

1. Import the picture `mri.tif`.
   Matlab image add on contains several pictures. The command to import one of those, or a picture from the current folder, is `imread`. For our purposes, it is better to deal with picture with values between 0 and 1 (sometimes the default is from -255 to 255): use the command `im2double`. Show the picture with `imshow`
2. Create 7x7 gaussian, average and motion PSFs.
   Use the command `fspecial`. For the Gaussian filter, impose variance equal to 2.
3. Show the result of convolution with those filters
   The command in Matlab is `conv2`. The default option is `'full'`, but we are interested in the option `'same'`
4. Add some Gaussian random noise to the picture with mean 0 and variance 0.005. Use the command `imnoise`.

Matrix formulation

Signals in 2D as vectors

Two-dimensional arrays (matrices) can be unfolded into simple arrays (vectors). Several strategies are possible, we will use te following one:

Signals in 2D as vectors

Two-dimensional arrays (matrices) can be unfolded into simple arrays (vectors). Several strategies are possible, we will use te following one:

$$\begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,1} & F_{2,2} & F_{2,3} \end{bmatrix}$$

Signals in 2D as vectors

Two-dimensional arrays (matrices) can be unfolded into simple arrays (vectors). Several strategies are possible, we will use te following one:

$$\begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,1} & F_{2,2} & F_{2,3} \end{bmatrix} \rightarrow \left[\begin{array}{c} F_{1,1} \\ F_{2,1} \\ \hline F_{1,2} \\ F_{2,2} \\ \hline F_{1,3} \\ F_{2,3} \end{array}\right]$$

Signals in 2D as vectors

Two-dimensional arrays (matrices) can be unfolded into simple arrays (vectors). Several strategies are possible, we will use te following one:

$$\begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,1} & F_{2,2} & F_{2,3} \end{bmatrix} \rightarrow \begin{bmatrix} F_{1,1} \\ F_{2,1} \\ \hline F_{1,2} \\ F_{2,2} \\ \hline F_{1,3} \\ F_{2,3} \end{bmatrix} \rightarrow \begin{bmatrix} f_1 \\ f_2 \\ \hline f_3 \\ f_4 \\ \hline f_5 \\ f_6 \end{bmatrix}$$

Signals in 2D as vectors

Two-dimensional arrays (matrices) can be unfolded into simple arrays (vectors). Several strategies are possible, we will use te following one:

$$\begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,1} & F_{2,2} & F_{2,3} \end{bmatrix} \rightarrow \begin{bmatrix} F_{1,1} \\ F_{2,1} \\ \hline F_{1,2} \\ F_{2,2} \\ \hline F_{1,3} \\ F_{2,3} \end{bmatrix} \rightarrow \begin{bmatrix} f_1 \\ f_2 \\ \hline f_3 \\ f_4 \\ \hline f_5 \\ f_6 \end{bmatrix} \rightarrow \begin{bmatrix} f_1 & f_3 & f_5 \\ f_2 & f_4 & f_6 \end{bmatrix}$$

Signals in 2D as vectors

Two-dimensional arrays (matrices) can be unfolded into simple arrays (vectors). Several strategies are possible, we will use te following one:

$$\begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,1} & F_{2,2} & F_{2,3} \end{bmatrix} \rightarrow \left[\begin{array}{c} F_{1,1} \\ F_{2,1} \\ \hline F_{1,2} \\ F_{2,2} \\ \hline F_{1,3} \\ F_{2,3} \end{array}\right] \rightarrow \left[\begin{array}{c} f_1 \\ f_2 \\ \hline f_3 \\ f_4 \\ \hline f_5 \\ f_6 \end{array}\right] \rightarrow \begin{bmatrix} f_1 & f_3 & f_5 \\ f_2 & f_4 & f_6 \end{bmatrix}$$

How to do it in Matlab:

- given a matrix $A \in \mathbb{R}^{n \times m}$, `a = A(:)` stores A in a nm dimensional vector;
- given a matrix $a \in \mathbb{R}^{nm}$, the command `A = reshape(A,[n,m])` creates the associated matrix with dimensions n and m.

Matrix representation

Also 2D convolution admits a representation via a suitable matrix.

Matrix representation

Also 2D convolution admits a representation via a suitable matrix.

- Unfold the original signal $F \in \mathbb{R}^{n \times m}$ as a vector $f \in \mathbb{R}^{nm}$.

Matrix representation

Also 2D convolution admits a representation via a suitable matrix.

- Unfold the original signal $F \in \mathbb{R}^{n \times m}$ as a vector $f \in \mathbb{R}^{nm}$.
- The convolution matrix A belongs to $\mathbb{R}^{nm \times nm}$. Each row `A(i,:)` contains the weights with which the elements of f contributes to the i-th component of the blurred vector, c_i.

Matrix representation

Also 2D convolution admits a representation via a suitable matrix.

- Unfold the original signal $F \in \mathbb{R}^{n \times m}$ as a vector $f \in \mathbb{R}^{nm}$.
- The convolution matrix A belongs to $\mathbb{R}^{nm \times nm}$. Each row `A(i,:)` contains the weights with which the elements of f contributes to the i-th component of the blurred vector, c_i.
- The result of the matrix product, $c = Af$, must then be reshaped into the blurred picture C.

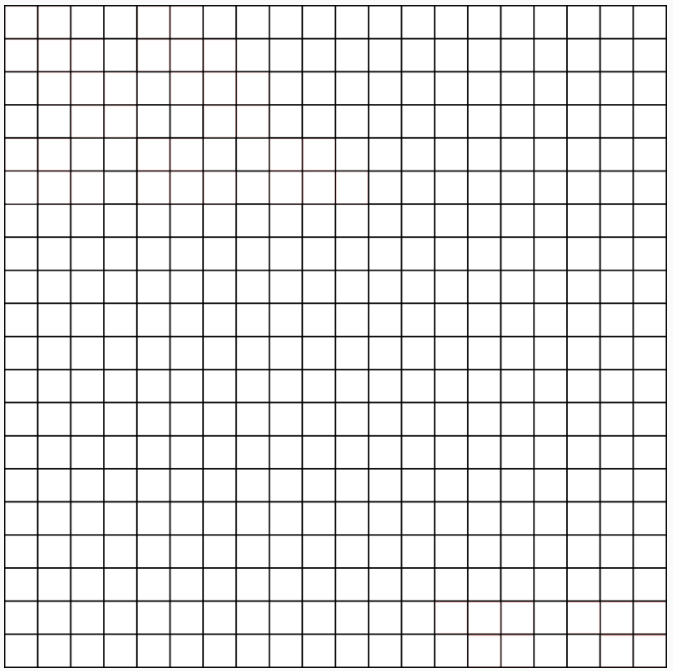
Matrix representation

Also 2D convolution admits a representation via a suitable matrix.

- Unfold the original signal $F \in \mathbb{R}^{n \times m}$ as a vector $f \in \mathbb{R}^{nm}$.
- The convolution matrix A belongs to $\mathbb{R}^{nm \times nm}$. Each row `A(i,:)` contains the weights with which the elements of f contributes to the i-th component of the blurred vector, c_i.
- The result of the matrix product, $c = Af$, must then be reshaped into the blurred picture C.

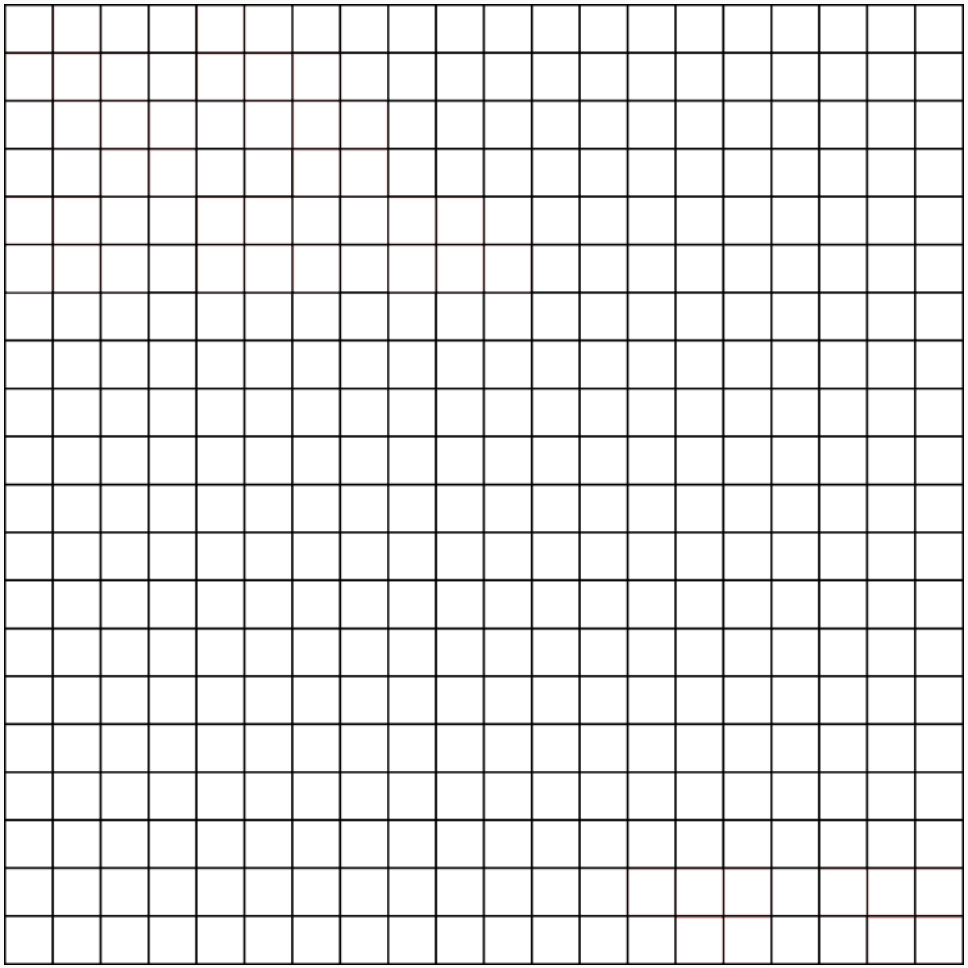
$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

$$\begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} \\ P_{2,1} & P_{2,2} & P_{3,3} \\ P_{3,1} & P_{3,2} & P_{3,3} \end{bmatrix}$$



Matrix representation - graphical intuition

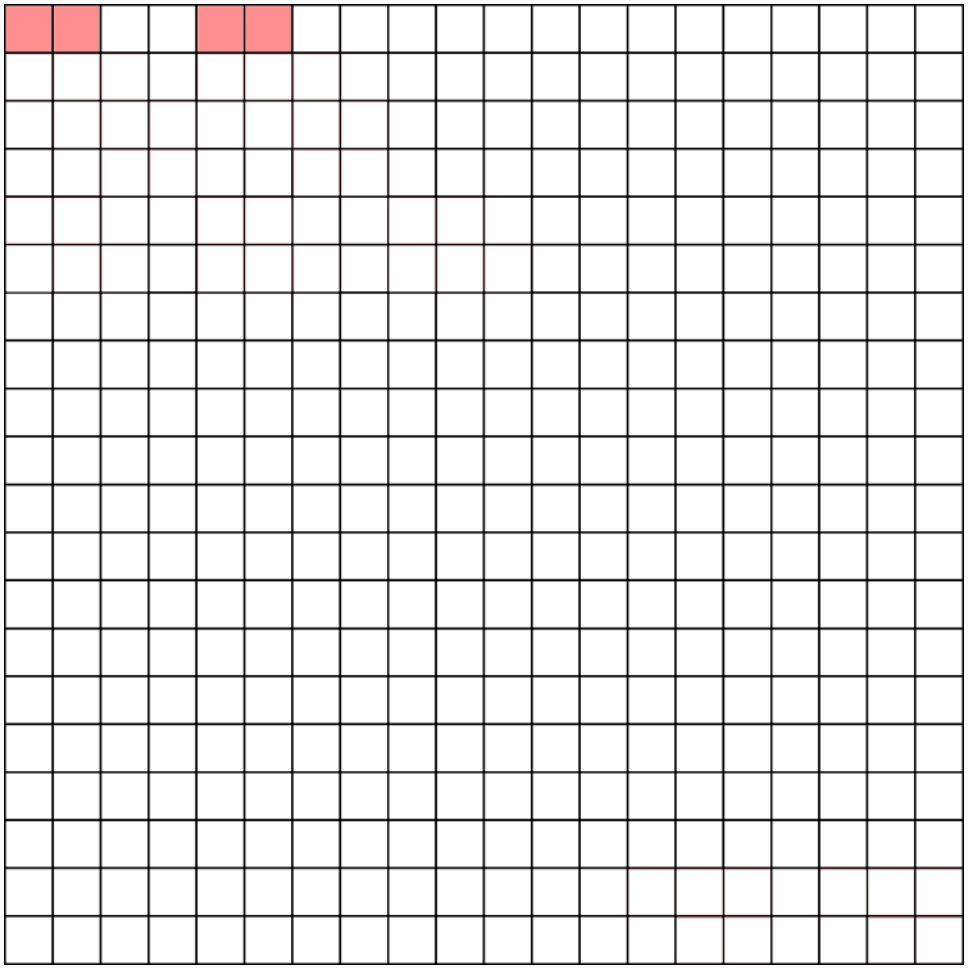
What elements of the matrix A are different from 0?



$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Matrix representation - graphical intuition

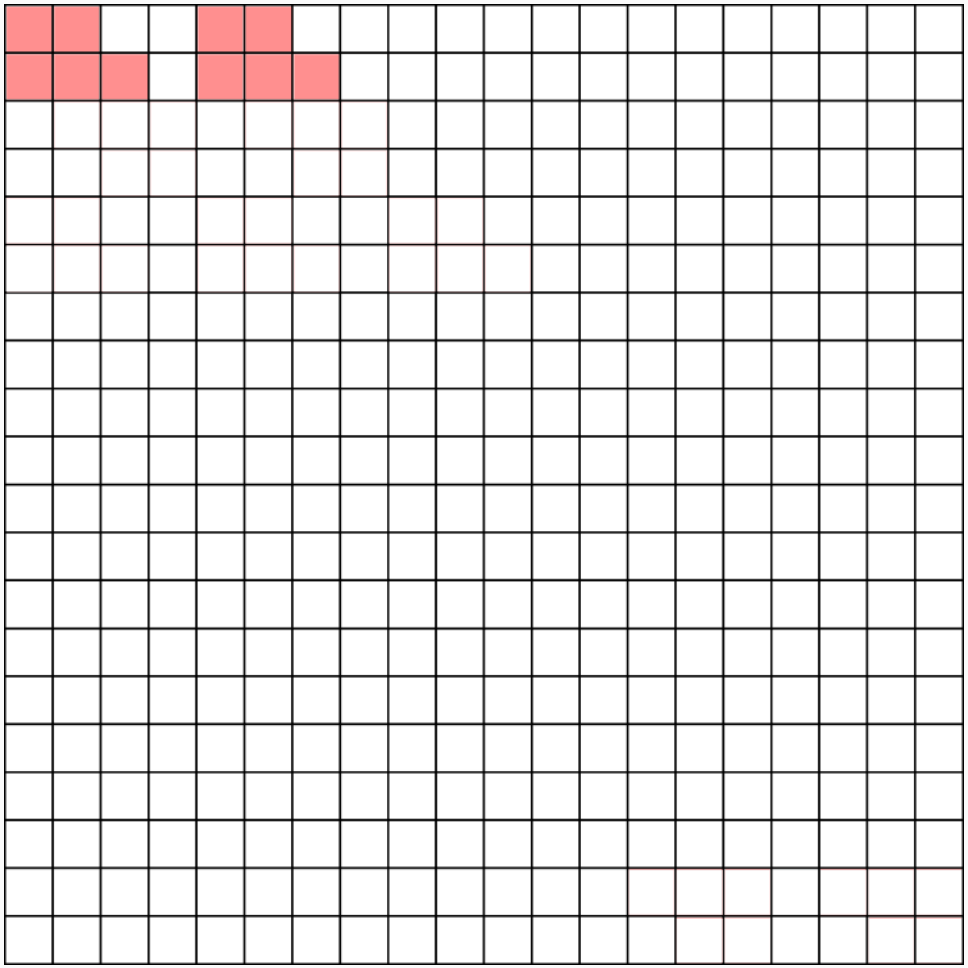
What elements of the matrix A are different from 0?



$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Matrix representation - graphical intuition

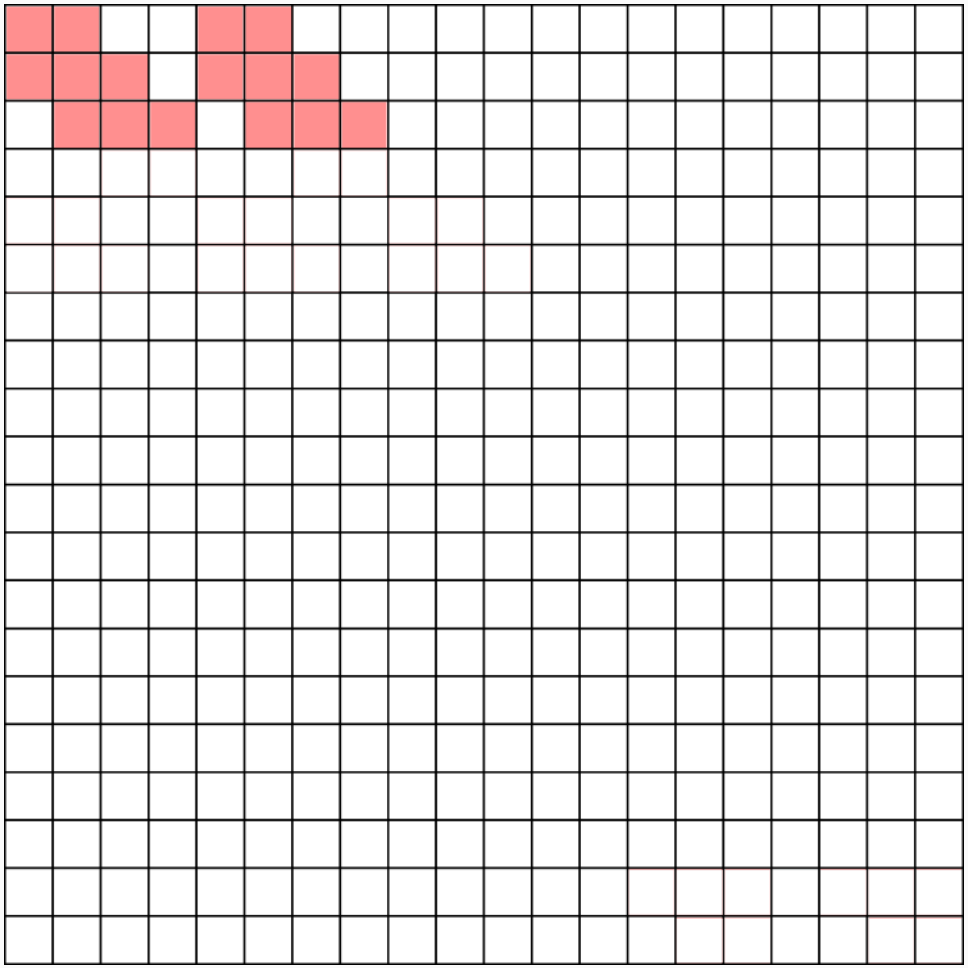
What elements of the matrix A are different from 0?



$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Matrix representation - graphical intuition

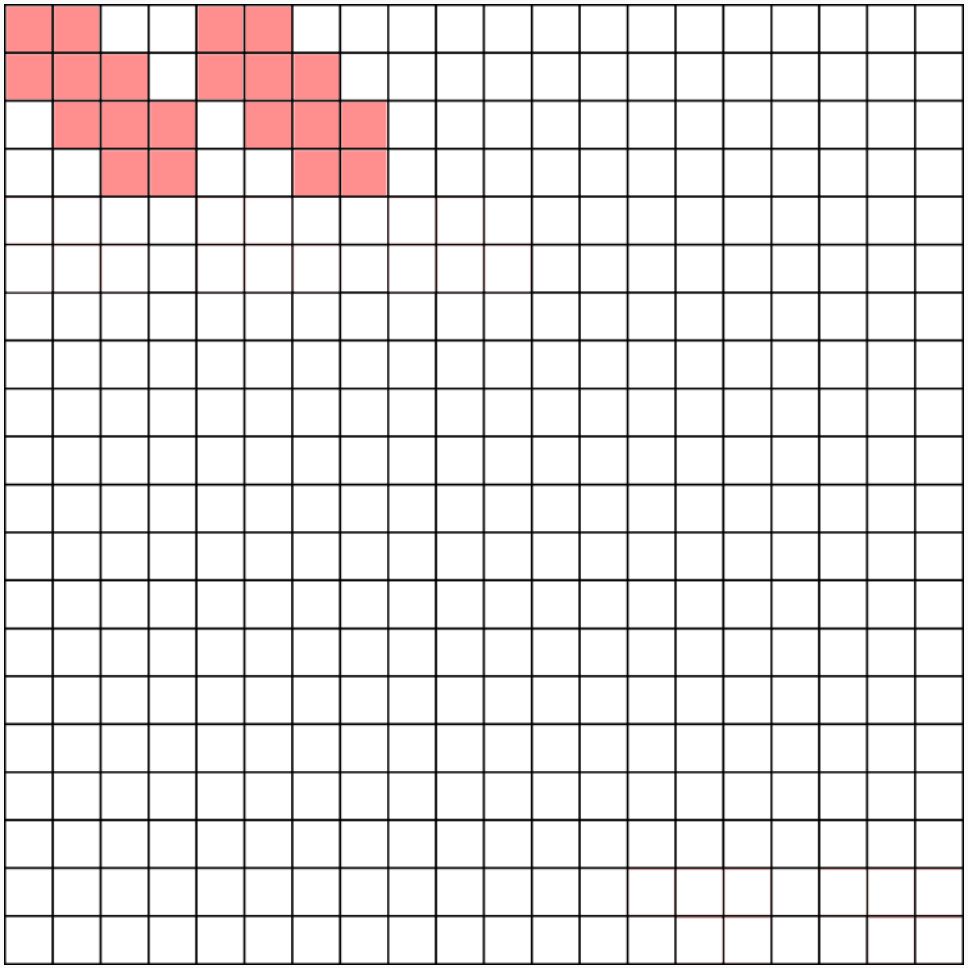
What elements of the matrix A are different from 0?



$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Matrix representation - graphical intuition

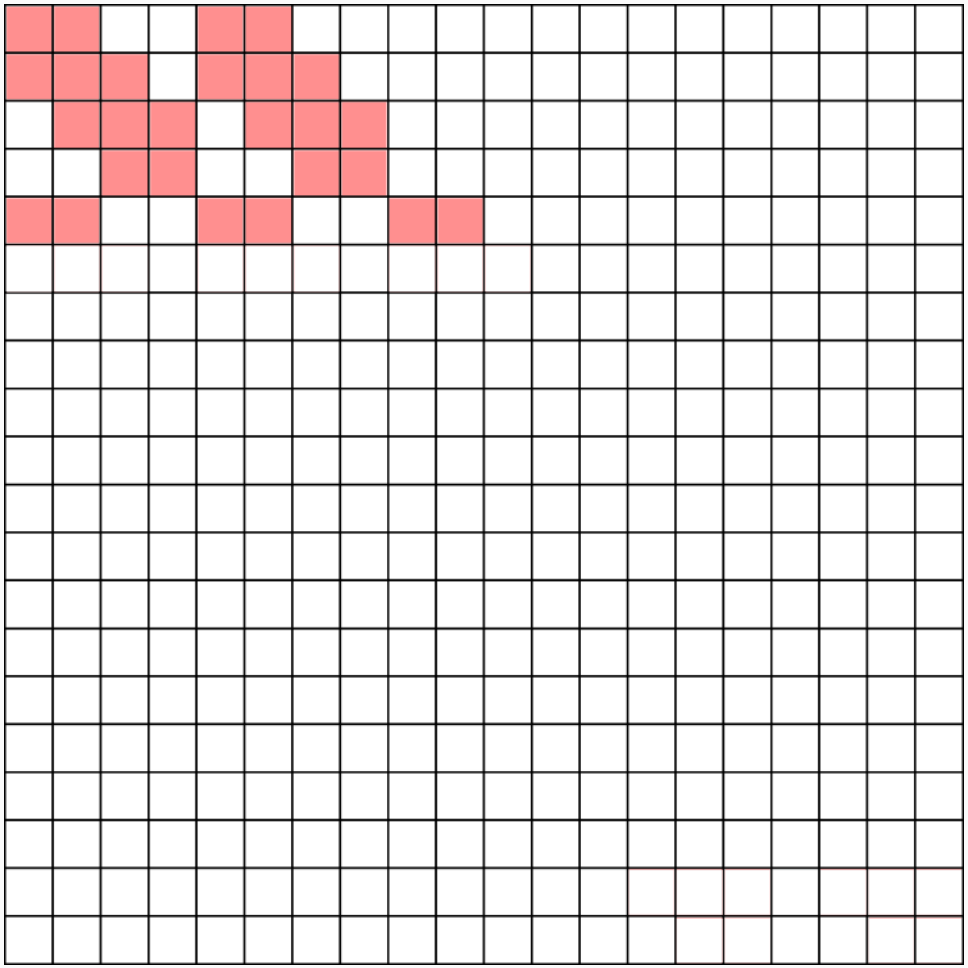
What elements of the matrix A are different from 0?



$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Matrix representation - graphical intuition

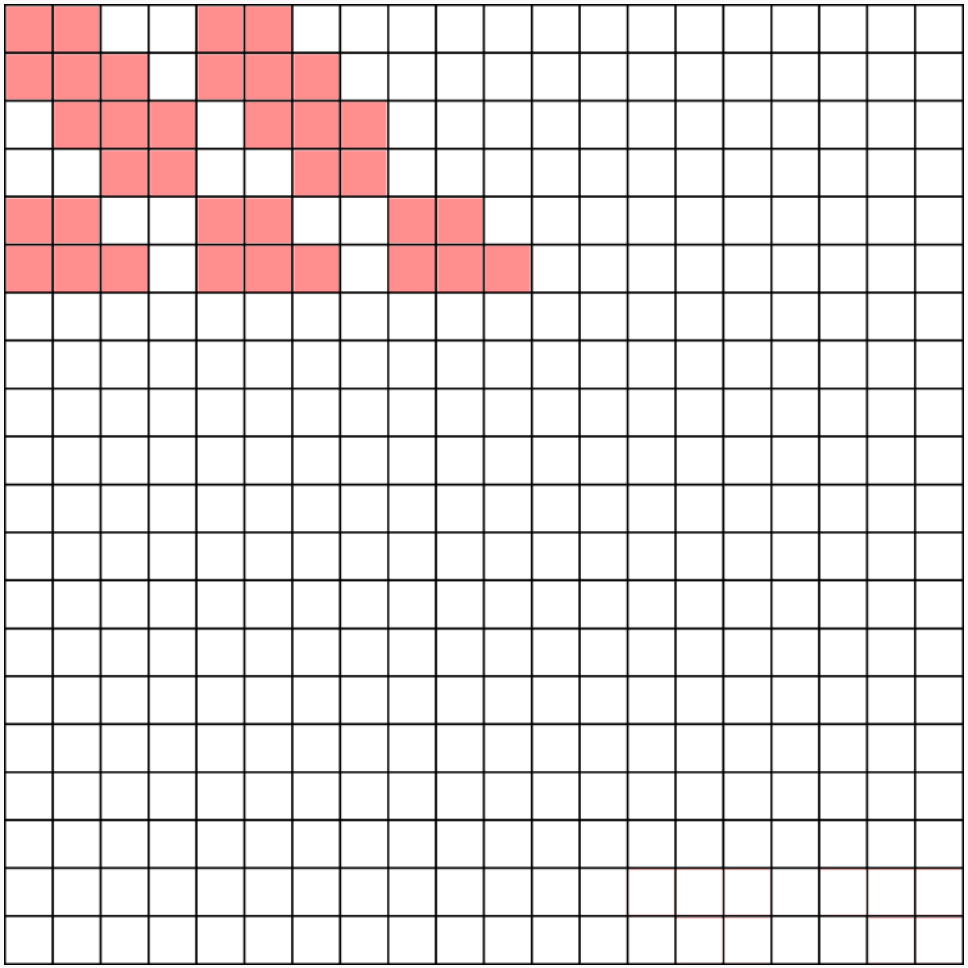
What elements of the matrix A are different from 0?



$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Matrix representation - graphical intuition

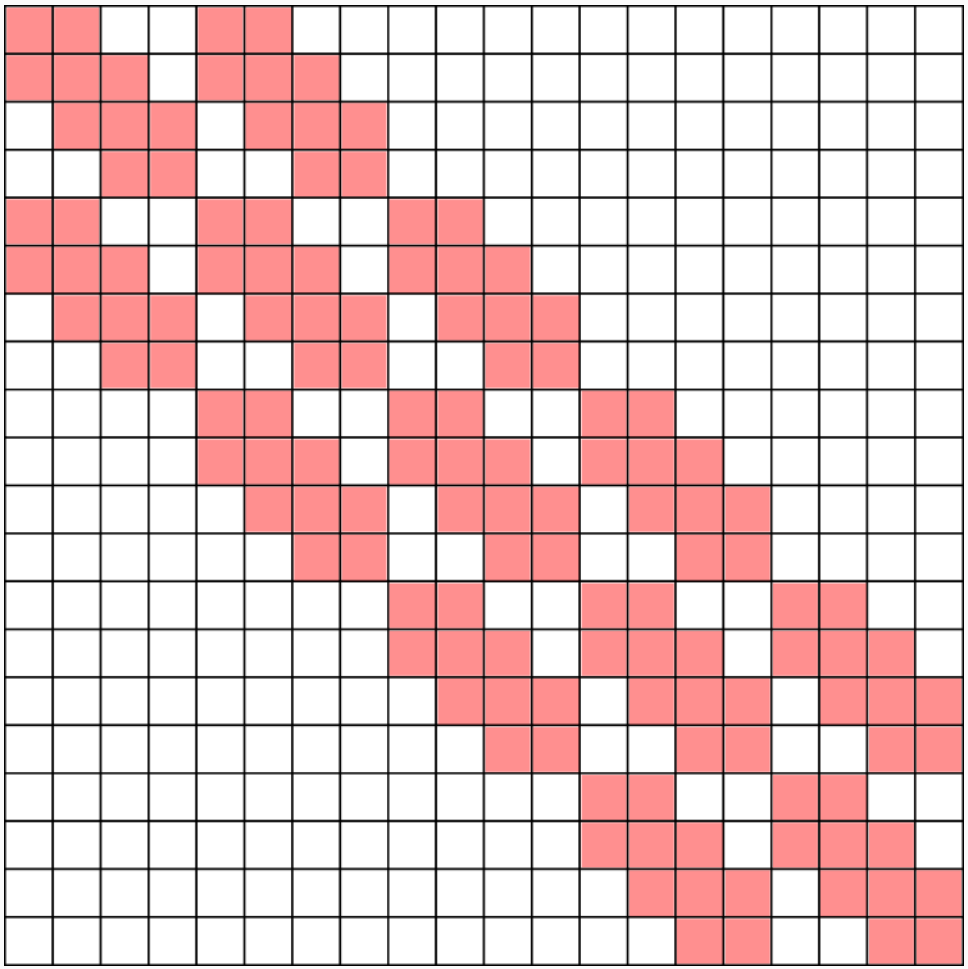
What elements of the matrix A are different from 0?



$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

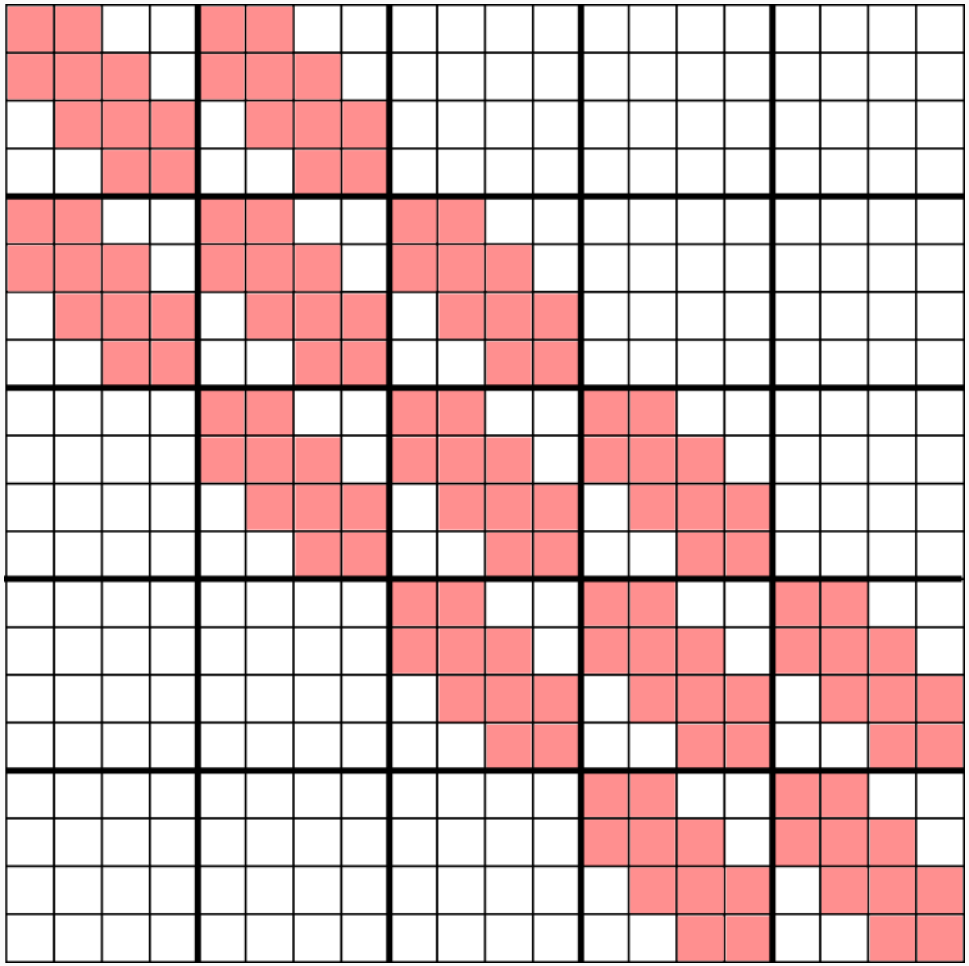
Matrix representation - graphical intuition

What elements of the matrix A are different from 0?



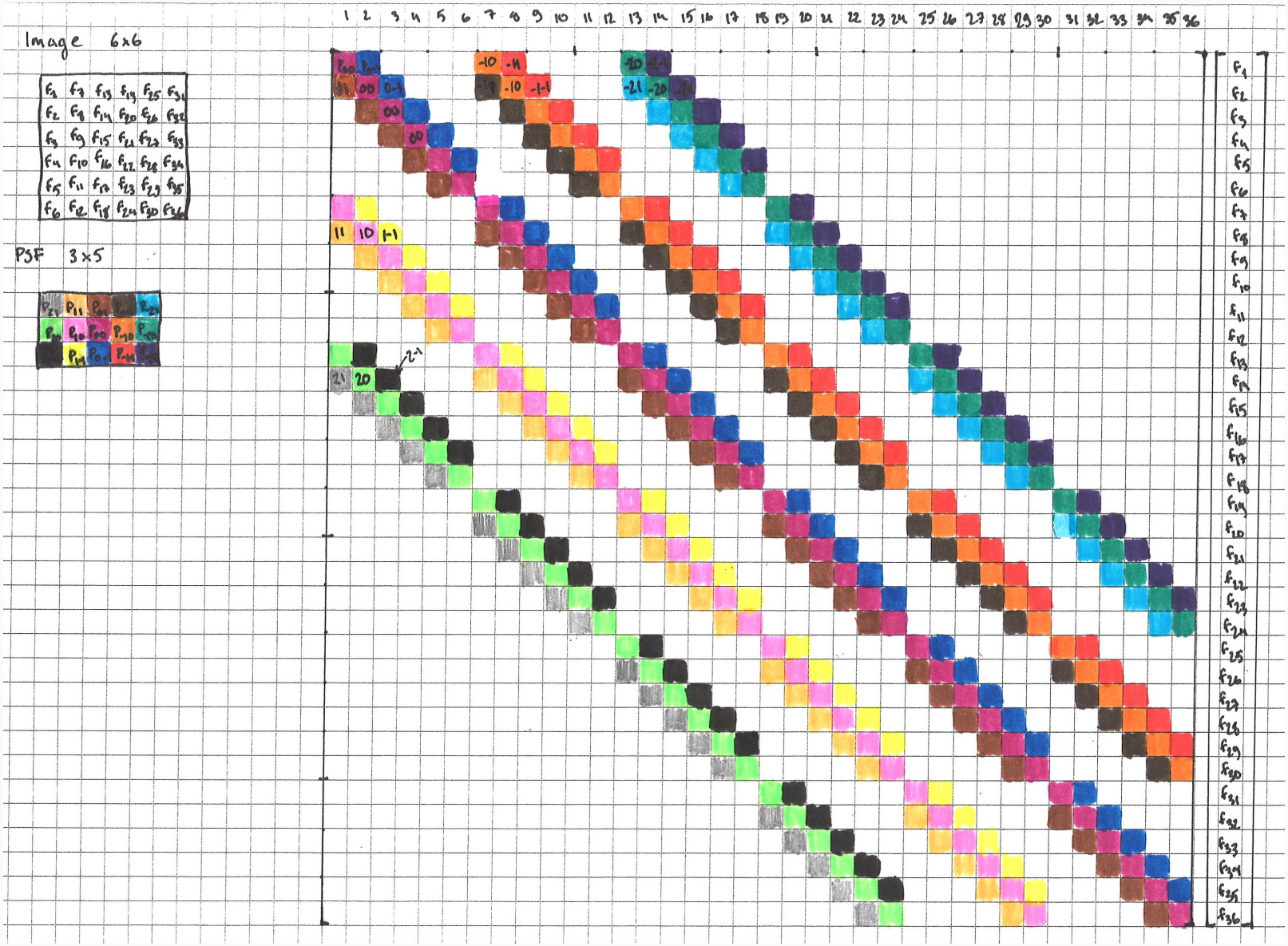
$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Matrix representation: some details



- A is a block matrix, consisting of $m \times m$ blocks of size $n \times n$;
- A is a block band matrix, with r (block) diagonals;
- every non null block of A is a band matrix with q diagonals;
- A is a **sparse** matrix, i.e., few elements are different from 0 (use the command `spy`).

Matrix representation: some details



Credits: Heli Virtanen

The Matlab command `convmtx2`

The command `A = convmtx2(P,n,m)` creates the convolution matrix associated with P and with an original signal of s $n \times m$.

The Matlab command `convmtx2`

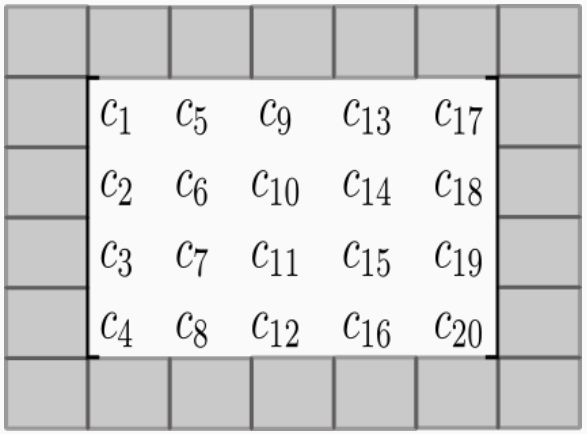
The command `A = convmtx2(P,n,m)` creates the convolution matrix associated with P and with an original signal of s $n \times m$.
The matrix A has $(n+2\nu)(m+2\mu)$ rows and nm columns. Computing A times f is equal to the command `conv2` with option `'full'`

The Matlab command `convmtx2`

The command `A = convmtx2(P,n,m)` creates the convolution matrix associated with P and with an original signal of s $n \times m$.
The matrix A has $(n+2\nu)(m+2\mu)$ rows and nm columns. Computing A times f is equal to the command `conv2` with option `'full'`

The result $c = Af$ can be reshaped in a matrix $C \in \mathbb{R}^{(n+2\nu)\times(m+2\mu)}$.
Goal: remove all the rows of A associated to undesired elements in the final signal.

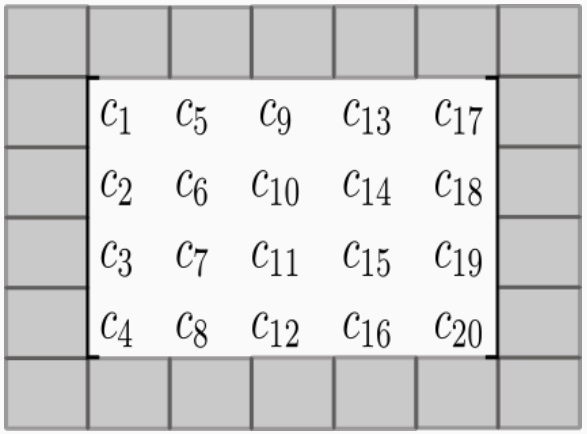
Proposed code:



The Matlab command `convmtx2`

The command `A = convmtx2(P,n,m)` creates the convolution matrix associated with P and with an original signal of s $n \times m$.
The matrix A has $(n+2\nu)(m+2\mu)$ rows and nm columns. Computing A times f is equal to the command `conv2` with option `'full'`

The result $c = Af$ can be reshaped in a matrix $C \in \mathbb{R}^{(n+2\nu)\times(m+2\mu)}$.
Goal: remove all the rows of A associated to undesired elements in the final signal.



Proposed code:

```
A = convmtx2(P,n,m);
Nr = (n+2*nu)*(m+2*mu);
I = [1:(n+2*nu)*mu, (Nr-mu*(n+2*nu)+1):Nr];
for i = 1:nu
I = [I, i:(n+2*nu):Nr, (n+2*nu-i+1):(n+2*nu):Nr];
end
A(I,:)  = [];
```

Implementation

Example 2

1. Import the picture `cameraman.tif`.
2. Create a 5x5 average PSF with the command `fspecial`.
3. Compute the blurred picture `C1` by the command `conv2`.
4. Assemble the matrix A and remove the undesired lines.
5. Compute the blurred picture `C2` by matrix product and rescaling.
6. Compare the results.

Deconvolution: regularization

Naïve inversion clearly fails

Matrix formulation of the 2D deconvolution problem

Given $M \in \mathbb{R}^{n \times m}$ (unfolded in a vector $m \in \mathbb{R}^{nm}$), find $F \in \mathbb{R}^{n \times m}$ (i.e., $f \in \mathbb{R}^{nm}$) such that $Af = m$.

Is Naïve inversion a good idea?

Naïve inversion clearly fails

Matrix formulation of the 2D deconvolution problem

Given $M \in \mathbb{R}^{n \times m}$ (unfolded in a vector $m \in \mathbb{R}^{nm}$), find $F \in \mathbb{R}^{n \times m}$ (i.e., $f \in \mathbb{R}^{nm}$) such that $Af = m$.

Is Naïve inversion a good idea?

Consider, the `cameraman` example. Use a 5x5 Gaussian PSF. Create synthetic data with a small amount of noise, $\delta = 0.005$

Naïve inversion clearly fails

Matrix formulation of the 2D deconvolution problem

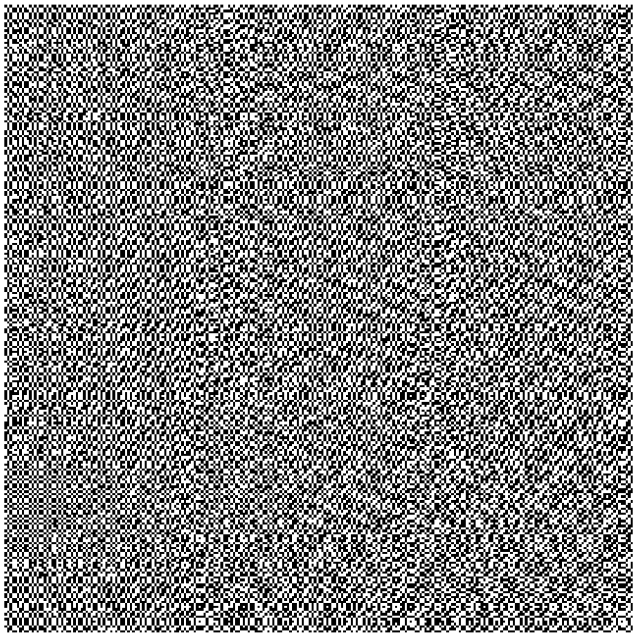
Given $M \in \mathbb{R}^{n \times m}$ (unfolded in a vector $m \in \mathbb{R}^{nm}$), find $F \in \mathbb{R}^{n \times m}$ (i.e., $f \in \mathbb{R}^{nm}$) such that $Af = m$.

Is Naïve inversion a good idea?

Consider, the `cameraman` example. Use a 5x5 Gaussian PSF. Create synthetic data with a small amount of noise, $\delta = 0.005$

Blurred and noisy data

No regularization



The Matlab commands `deconvwnr` and `deconvreg`

Matlab has two built-in methods for image deconvolution: `deconvwnr` uses a technique related to the study of Wiener involving the Fourier transform: this is extremely effective in absence of noise. The method `deconvreg` is studied to guarantee stability with noisy measurements.

The Matlab commands `deconvwnr` and `deconvreg`

Matlab has two built-in methods for image deconvolution: `deconvwnr` uses a technique related to the study of Wiener involving the Fourier transform: this is extremely effective in absence of noise. The method `deconvreg` is studied to guarantee stability with noisy measurements.

Tikhonov regularization

The 2D convolution problem is a (ill-posed) linear inverse problem. All the technique we studied in 1D can be applied also here.

Tikhonov regularization

Set $\alpha > 0$. Find $f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|f\|^2 \right\}$

Tikhonov regularization

The 2D convolution problem is a (ill-posed) linear inverse problem. All the technique we studied in 1D can be applied also here.

Tikhonov regularization

$$\text{Set } \alpha > 0. \text{ Find } f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|f\|^2 \right\}$$

The solution can be found via the **normal equations**:

$$f_{\alpha,\delta} = (A^T A + \alpha I)^{-1} A^T m$$

The other techniques are computationally impracticable (expecially computing the SVD).
Remark: the matrix $A^T A + \alpha I$ is sparse: use a sparse solver for the linear system!

Tikhonov regularization

The 2D convolution problem is a (ill-posed) linear inverse problem. All the technique we studied in 1D can be applied also here.

Tikhonov regularization

$$\text{Set } \alpha > 0. \text{ Find } f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|f\|^2 \right\}$$

The solution can be found via the **normal equations**:

$$f_{\alpha,\delta} = (A^T A + \alpha I)^{-1} A^T m$$

The other techniques are computationally impracticable (expecially computing the SVD).
Remark: the matrix $A^T A + \alpha I$ is sparse: use a sparse solver for the linear system!

Tikhonov, α = 0.15

Generalized Tikhonov

Generalized Tikhonov regularization

$$\text{Set } \alpha > 0. \text{ Find } f_\alpha = \underset{f \in \mathbb{R}^{nm}}{\arg\min} \left\{ \|Af - m\|^2 + \alpha \|L(f - f_*)\|^2 \right\}$$

Most interesting example: penalize the derivatives to enhance smoothness.

Generalized Tikhonov

Generalized Tikhonov regularization

$$\text{Set } \alpha > 0. \text{ Find } f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|L(f - f_*)\|^2 \right\}$$

Most interesting example: penalize the derivatives to enhance smoothness.

In this case, $L \in \mathbb{R}^{2nm \times nm} = \begin{bmatrix} L_x \\ L_y \end{bmatrix}$, where L_x is the discretization of the horizontal derivative and L_y the vertical one.

Generalized Tikhonov

Generalized Tikhonov regularization

$$\text{Set } \alpha > 0. \text{ Find } f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|L(f - f_*)\|^2 \right\}$$

Most interesting example: penalize the derivatives to enhance smoothness.

In this case, $L \in \mathbb{R}^{2nm \times nm} = \begin{bmatrix} L_x \\ L_y \end{bmatrix}$, where L_x is the discretization of the horizontal derivative and L_y the vertical one.

- it holds $\|Lf\|^2 = ([L_x f, L_y f]^T, [L_x f, L_y f]^T) = \|L_x f\|^2 + \|L_y f\|^2$;

Generalized Tikhonov

Generalized Tikhonov regularization

$$\text{Set } \alpha > 0. \text{ Find } f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|L(f - f_*)\|^2 \right\}$$

Most interesting example: penalize the derivatives to enhance smoothness.

In this case, $L \in \mathbb{R}^{2nm \times nm} = \begin{bmatrix} L_x \\ L_y \end{bmatrix}$, where L_x is the discretization of the horizontal derivative and L_y the vertical one.

- it holds $\|Lf\|^2 = ([L_x f, L_y f]^T, [L_x f, L_y f]^T) = \|L_x f\|^2 + \|L_y f\|^2$;
- **normal equations:** we easily see that $L^T L = L_x^T L_x + L_y^T L_y$, hence

$$f_{\alpha,\delta} = (A^T A + \alpha L_x^T L_x + \alpha L_y^T L_y)^{-1} A^T m$$

Generalized Tikhonov

Generalized Tikhonov regularization

$$\text{Set } \alpha > 0. \text{ Find } f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|L(f - f_*)\|^2 \right\}$$

Most interesting example: penalize the derivatives to enhance smoothness.

In this case, $L \in \mathbb{R}^{2nm \times nm} = \begin{bmatrix} L_x \\ L_y \end{bmatrix}$, where L_x is the discretization of the horizontal derivative and L_y the vertical one.

- it holds $\|Lf\|^2 = ([L_x f, L_y f]^T, [L_x f, L_y f]^T) = \|L_x f\|^2 + \|L_y f\|^2$;
- **normal equations:** we easily see that $L^T L = L_x^T L_x + L_y^T L_y$, hence

$$f_{\alpha,\delta} = (A^T A + \alpha L_x^T L_x + \alpha L_y^T L_y)^{-1} A^T m$$

- small issue: how to build L_x and L_y?

Generalized Tikhonov: details

Approximation of the derivatives:

$$\frac{\partial F}{\partial x}(x_i, y_j) \approx \frac{F_{i,j} - F_{i,j-1}}{\Delta x} \qquad \frac{\partial F}{\partial y}(x_i, y_j) \approx \frac{F_{i,j} - F_{i-1,j}}{\Delta y}$$

Generalized Tikhonov: details

Approximation of the derivatives:

$$\frac{\partial F}{\partial x}(x_i, y_j) \approx \frac{F_{i,j} - F_{i,j-1}}{\Delta x} \qquad \frac{\partial F}{\partial y}(x_i, y_j) \approx \frac{F_{i,j} - F_{i-1,j}}{\Delta y}$$

Some problems when unfolding the matrix ($\Delta x = \Delta y = 1$):

$$(L_x f)_i = f_i - f_i^{left} \qquad (L_y f)_i = f_i - f_i^{above},$$

where f_i^{left} is the component of f corresponding to the element that was on the left of the i-th component before unfolding, and f_i^{above} is the one above.

$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Generalized Tikhonov: details

Approximation of the derivatives:

$$\frac{\partial F}{\partial x}(x_i, y_j) \approx \frac{F_{i,j} - F_{i,j-1}}{\Delta x} \qquad \frac{\partial F}{\partial y}(x_i, y_j) \approx \frac{F_{i,j} - F_{i-1,j}}{\Delta y}$$

Some problems when unfolding the matrix ($\Delta x = \Delta y = 1$):

$$(L_x f)_i = f_i - f_i^{left} \qquad (L_y f)_i = f_i - f_i^{above},$$

where f_i^{left} is the component of f corresponding to the element that was on the left of the i-th component before unfolding, and f_i^{above} is the one above.

$$\begin{bmatrix} f_1 & f_5 & f_9 & f_{13} & f_{17} \\ f_2 & f_6 & f_{10} & f_{14} & f_{18} \\ f_3 & f_7 & f_{11} & f_{15} & f_{19} \\ f_4 & f_8 & f_{12} & f_{16} & f_{20} \end{bmatrix}$$

Proposed code:

```
Lx = spdiags([ones(n*m,1) -ones(n*m,1)],[0,-n],n*m,n*m);
Ly = spdiags([ones(n*m,1) -ones(n*m,1)],[0,1],n*m,n*m);
Ly((m:m:n*m-1)+1,(m:m:n*m)) = 0;
```

Generalized Tikhonov - results

$$f_{\alpha,\delta} = (A^T A + \alpha L_x^T L_x + \alpha L_y^T L_y)^{-1} A^T m$$

Generalized Tikhonov, α = 0.15

Generalized Tikhonov, α = 1.75

The noise gets removed, but all the discontinuity in the signal (sharp edges) are lost.

Total Variation

A good strategy to enhance the sharpness of the edges would be the Total Variation regularization:

Total Variation

Set $\alpha > 0$. Find $f_\alpha = \underset{f \in \mathbb{R}^{nm}}{\arg\min} \left\{ \|Af - m\|^2 + \alpha \|Lf\|_1 \right\}$, where $L = [L_x, L_y]^T$ is the matrix approximating the partial derivatives.

Total Variation

A good strategy to enhance the sharpness of the edges would be the Total Variation regularization:

Total Variation

Set $\alpha > 0$. Find $f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|Lf\|_1 \right\}$, where $L = [L_x, L_y]^T$ is the matrix approximating the partial derivatives.

How to solve it? Interpretation as quadratic programming:

$$\text{minimize } \left\{ \|Af\|^2 - 2m^T Af + \alpha \mathbf{1}^T v_+ + \alpha \mathbf{1}^T v_- + \alpha \mathbf{1}^T w_+ + \alpha \mathbf{1}^T w_- \right\}$$

$$\text{subject to: } v_+ - v_- = L_x f;\ w_+ - w_- = L_x f;\ v_+, v_-, w_+, w_- \geq 0.$$

where $\mathbf{1} \in \mathbb{R}^{nm}$ is a vector of nm ones.

Total Variation

A good strategy to enhance the sharpness of the edges would be the Total Variation regularization:

Total Variation

Set $\alpha > 0$. Find $f_\alpha = \arg\min_{f \in \mathbb{R}^{nm}} \left\{ \|Af - m\|^2 + \alpha \|Lf\|_1 \right\}$, where $L = [L_x, L_y]^T$ is the matrix approximating the partial derivatives.

How to solve it? Interpretation as quadratic programming:

$$\text{minimize } \left\{ \|Af\|^2 - 2m^T Af + \alpha \mathbf{1}^T v_+ + \alpha \mathbf{1}^T v_- + \alpha \mathbf{1}^T w_+ + \alpha \mathbf{1}^T w_- \right\}$$

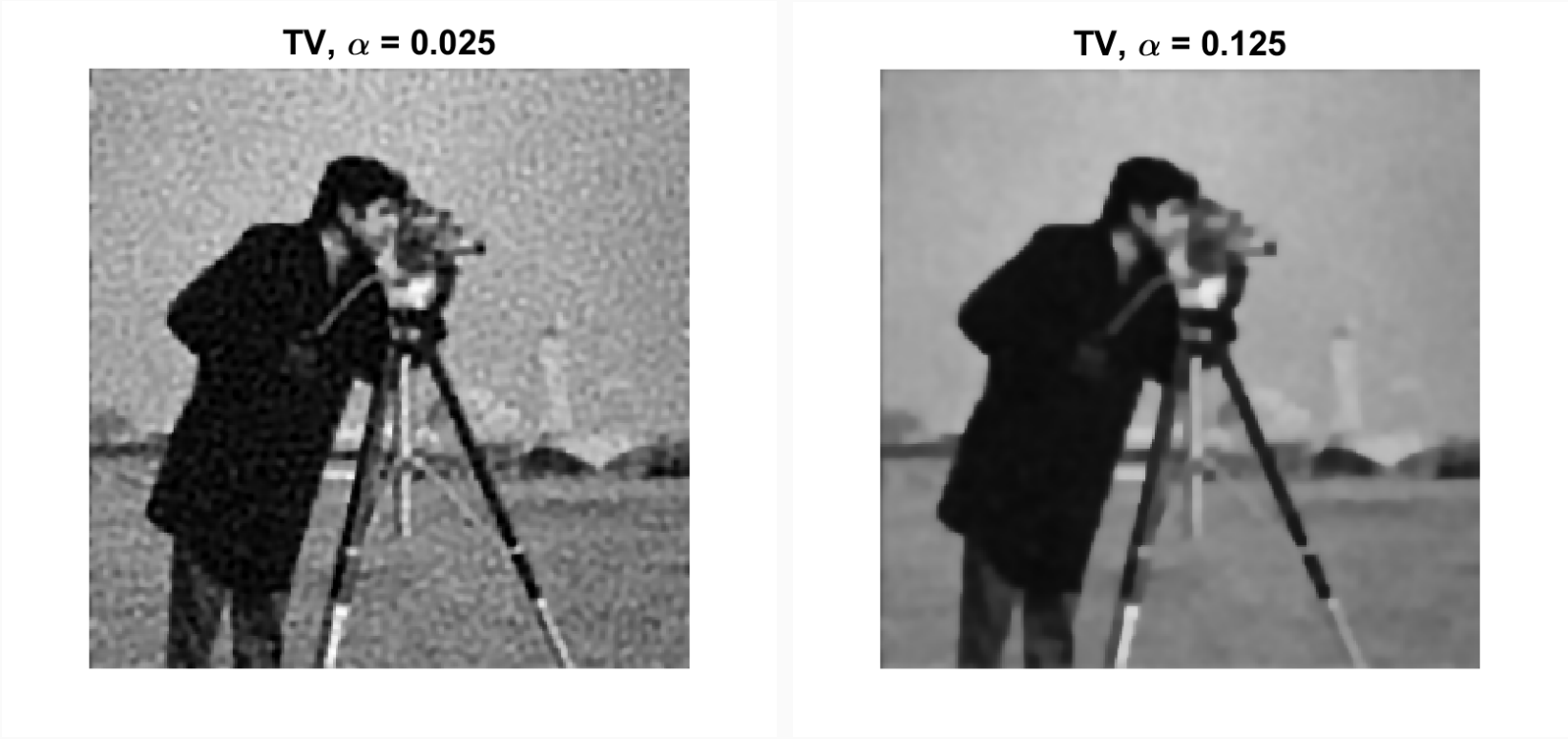
$$\text{subject to: } v_+ - v_- = L_x f;\ w_+ - w_- = L_x f;\ v_+, v_-, w_+, w_- \geq 0.$$

where $\mathbf{1} \in \mathbb{R}^{nm}$ is a vector of nm ones.

The implementation in Matlab via `quadprog` requires to define a matrix $H \in R^{5mn \times 5mn}$. Without sparse matrix solvers, this would be too demanding for most of the computers.

Total Variation: details

The time required by `quadprog` to process a 256x256 picture on a common laptop is around 1 minute.



The noise is removed the edges are clearly visible, but many small details are completely missing.

What regularization strategy would you pick? - I

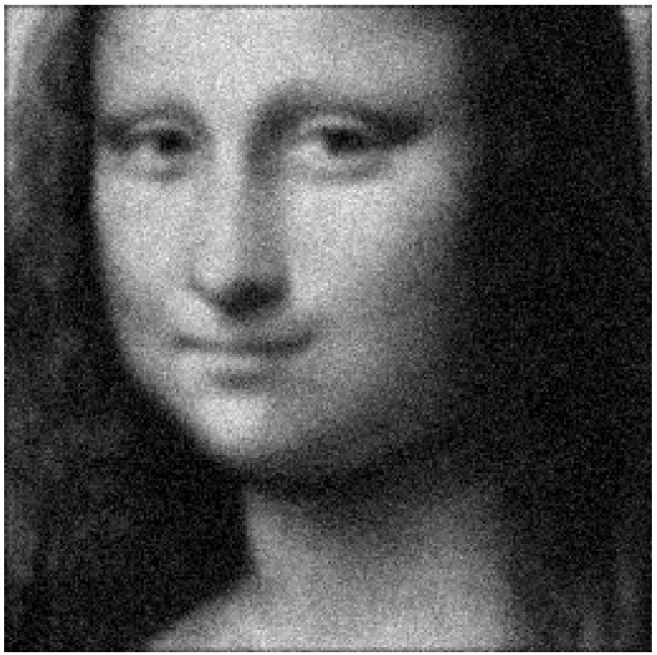
Example 3

1. We are given a blurred and noise picture to deconvolve (filter: 5x5, Gaussian).
2. According to a priori knowledge, what regularization strategy would you recommend?

What regularization strategy would you pick? - I

Example 3

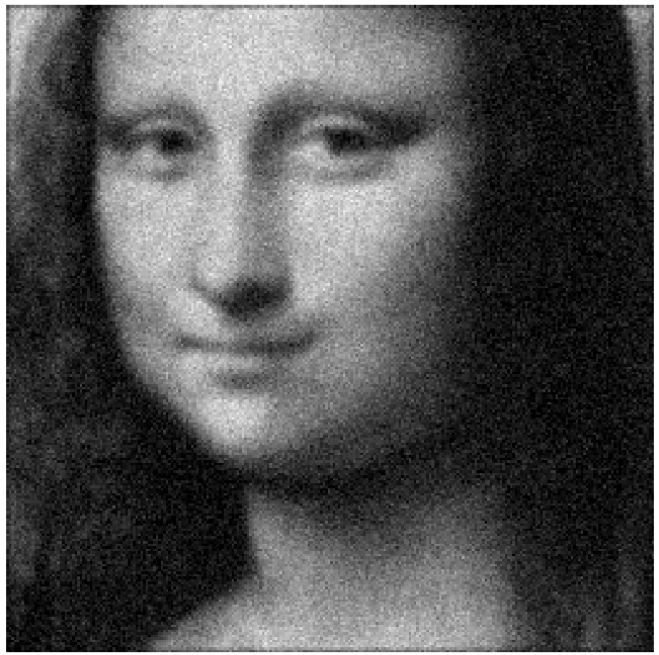
1. We are given a blurred and noise picture to deconvolve (filter: 5x5, Gaussian).
2. According to a priori knowledge, what regularization strategy would you recommend?



What regularization strategy would you pick? - I

Example 3

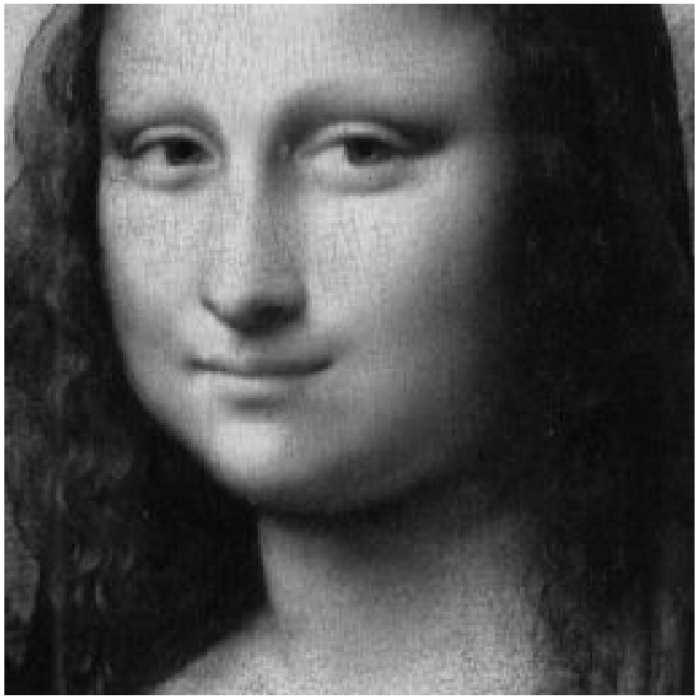
1. We are given a blurred and noise picture to deconvolve (filter: 5x5, Gaussian).
2. According to a priori knowledge, what regularization strategy would you recommend?



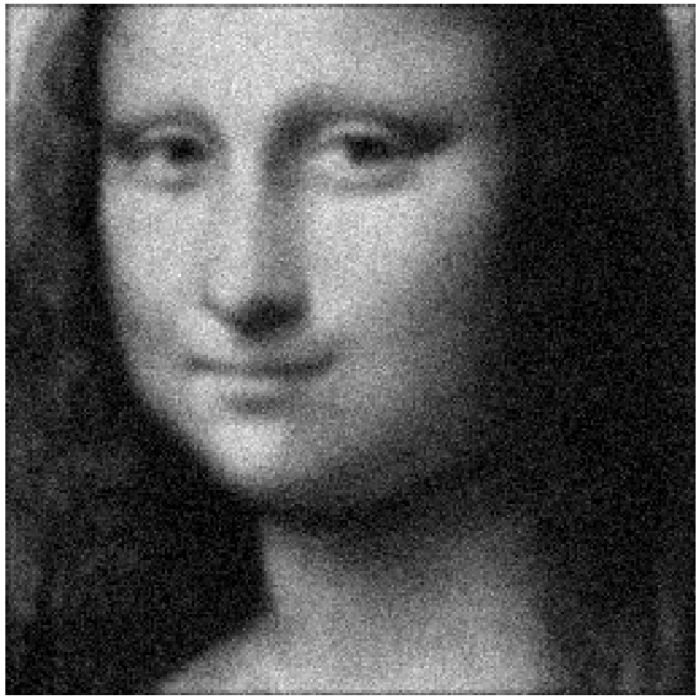
Hint: Leonardo Da Vinci was an expert of *sfumato*, a painting technique consisting in softening the transition between colours.

What regularization strategy would you pick? - I

Original Image

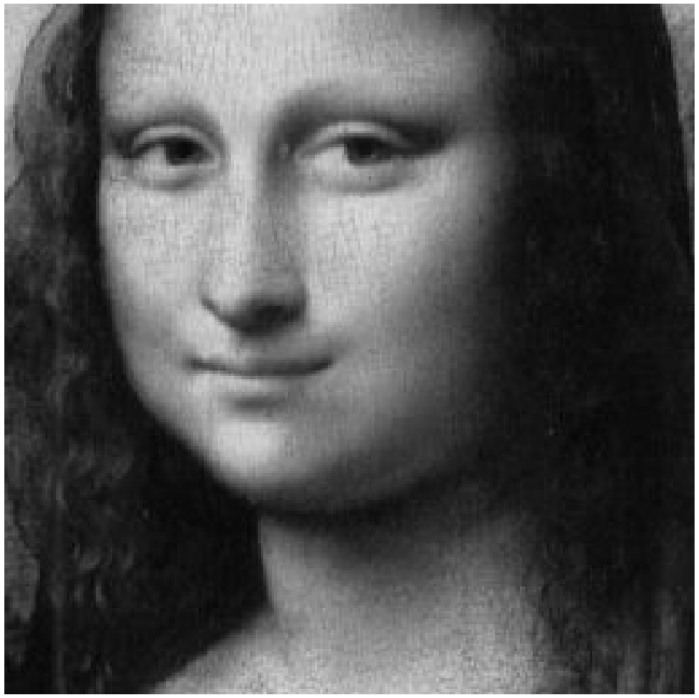


Blurred and noisy data

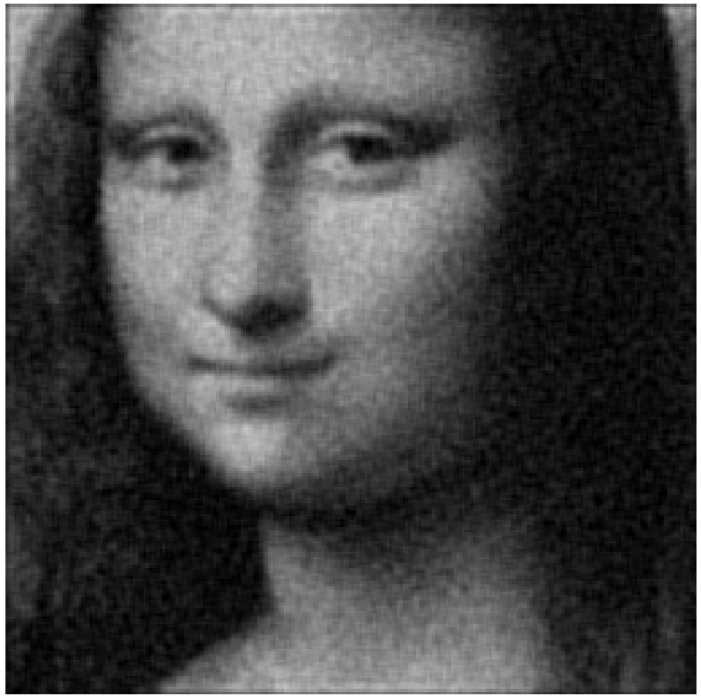


What regularization strategy would you pick? - I

Original Image

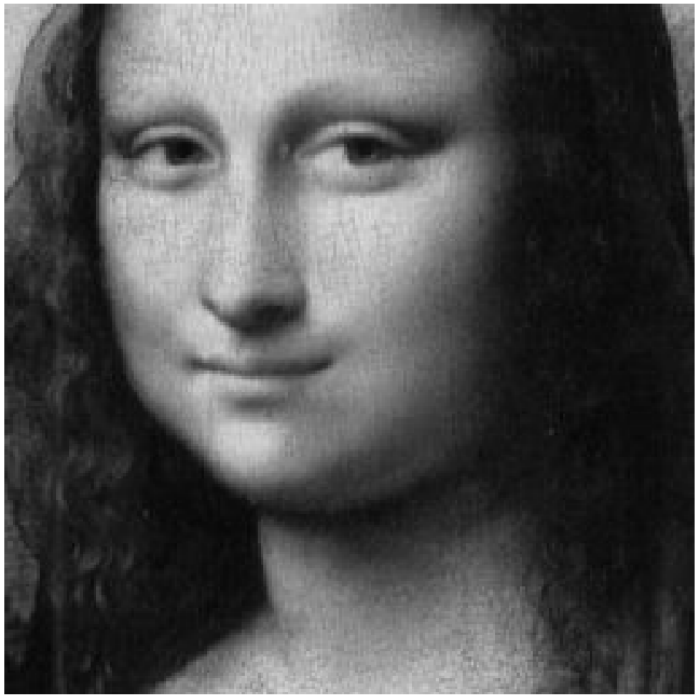


Tikhonov, α = 0.15



What regularization strategy would you pick? - I

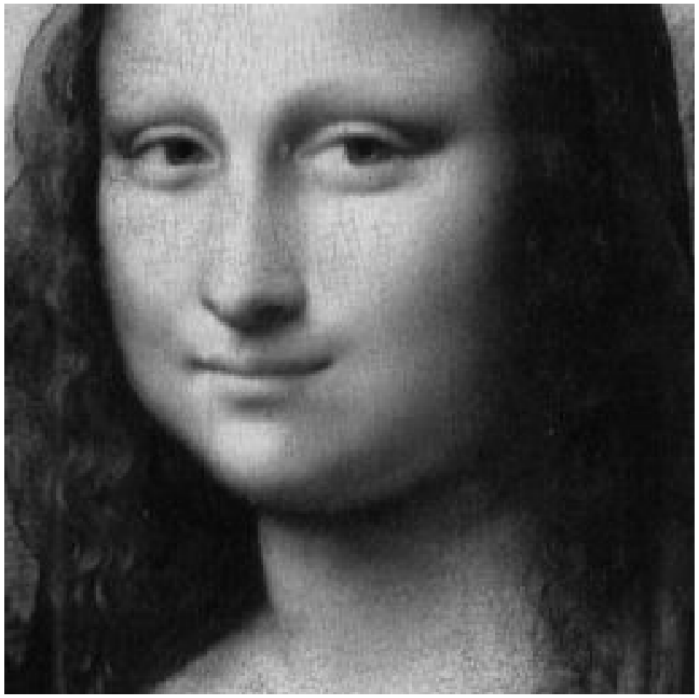
Original Image



Generalized Tikhonov, α = 0.9

What regularization strategy would you pick? - I

Original Image



TV, α = 0.125

What regularization strategy would you pick? - I

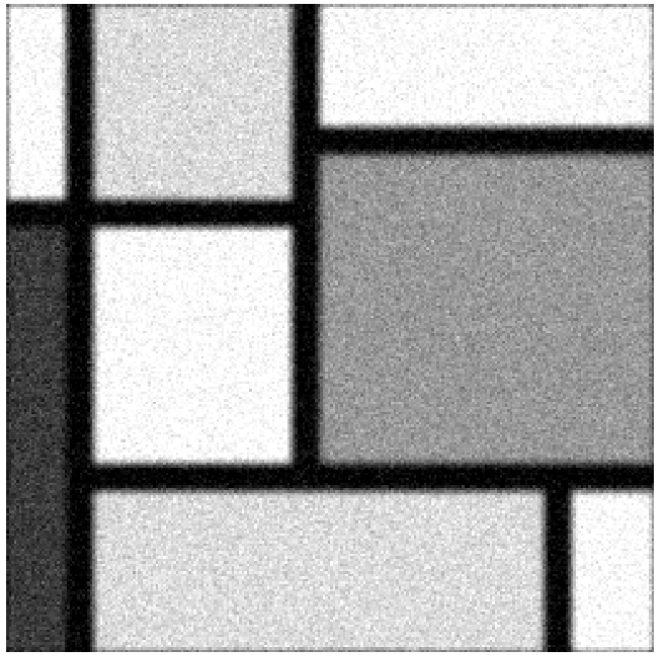
Example 3

1. We are given a blurred and noise picture to deconvolve (filter: 5x5, Gaussian).
2. According to a priori knowledge, what regularization strategy would you recommend?

What regularization strategy would you pick? - I

Example 3

1. We are given a blurred and noise picture to deconvolve (filter: 5x5, Gaussian).
2. According to a priori knowledge, what regularization strategy would you recommend?

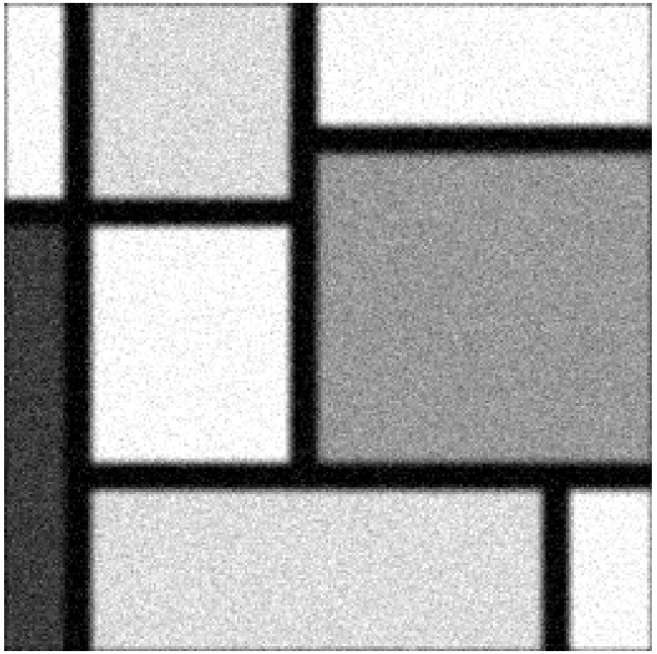


What regularization strategy would you pick? - I

Example 3

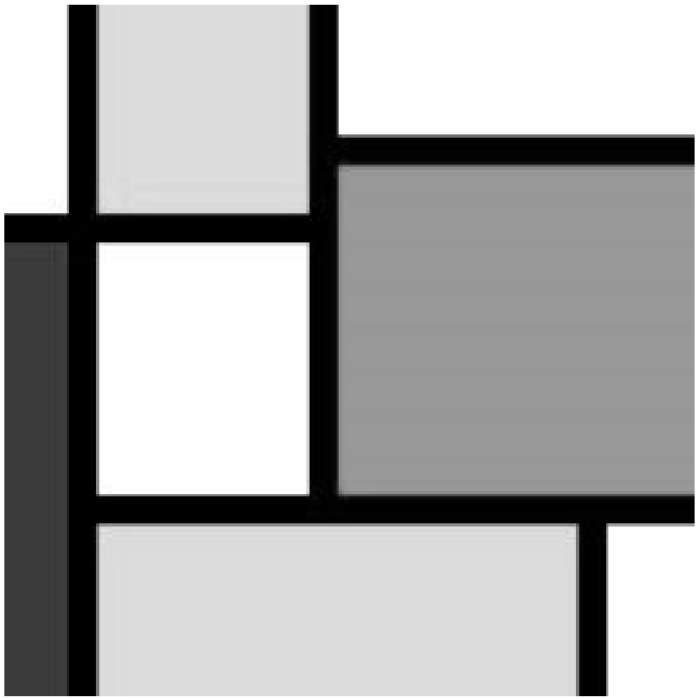
1. We are given a blurred and noise picture to deconvolve (filter: 5x5, Gaussian).
2. According to a priori knowledge, what regularization strategy would you recommend?

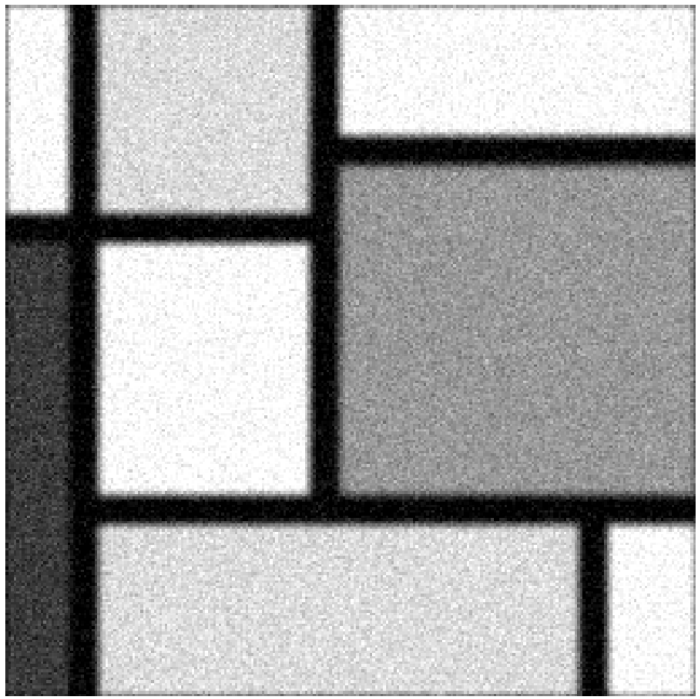
Blurred and noisy data



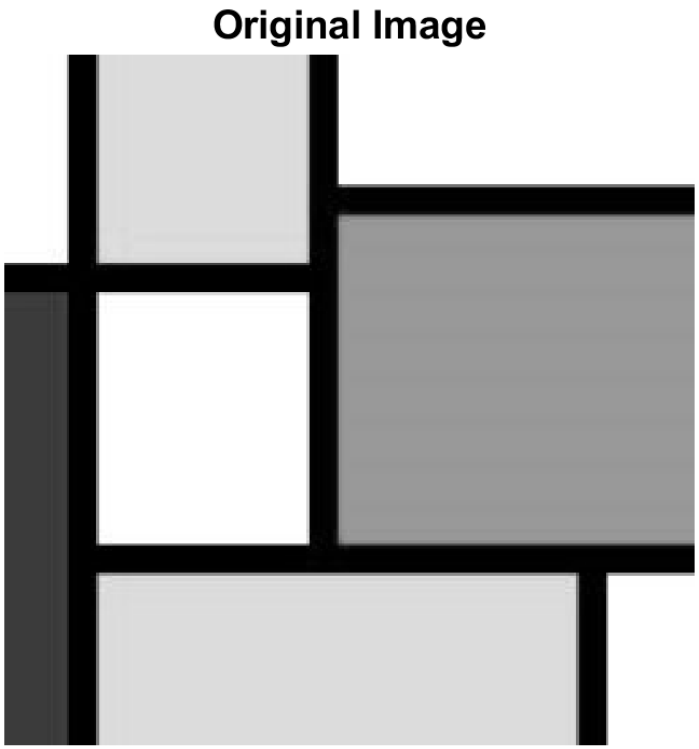
Hint: Piet Mondrian was the founder of Neoplasticism, whose main elements are the use of primary colors (red, blue and yellow), three primary values (black, white and gray) and two primary directions (horizontal and vertical).

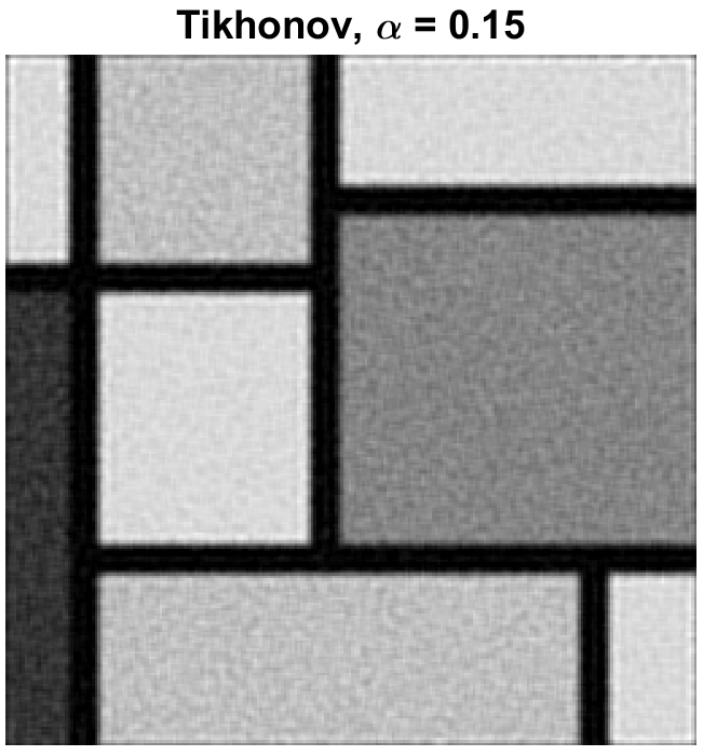
What regularization strategy would you pick? - I



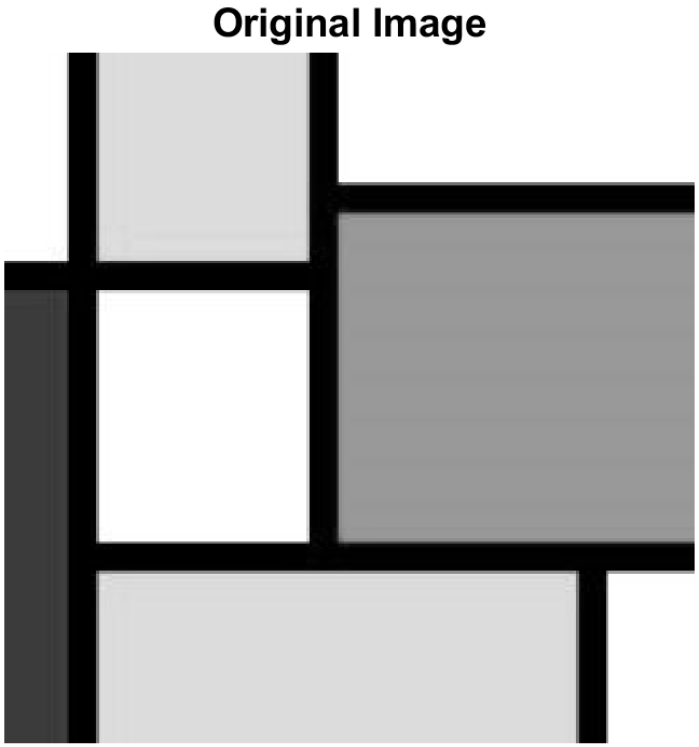


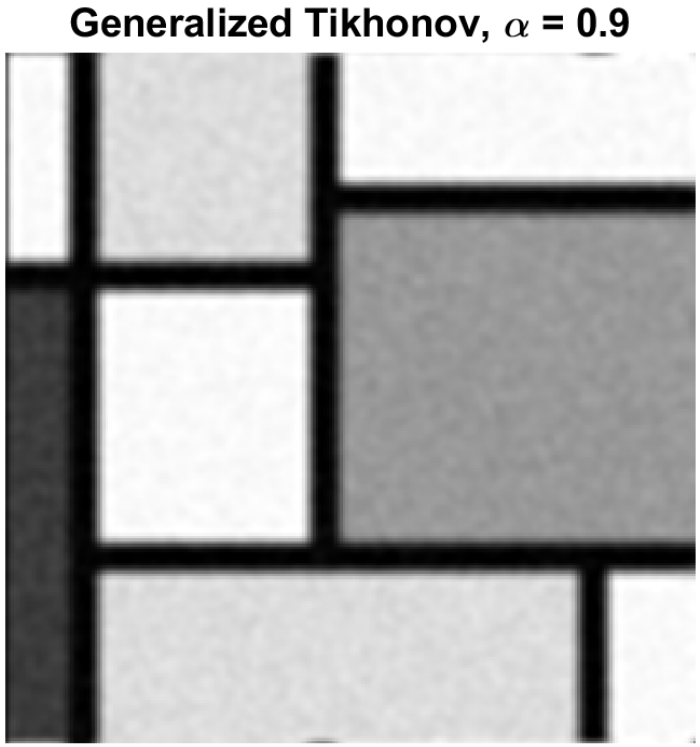
What regularization strategy would you pick? - I





What regularization strategy would you pick? - I





What regularization strategy would you pick? - I