Tutorial 8:
Bilateral Filtering
Non-Local Means

Digital Image Processing (236860)

Agenda

- Bilateral filtering
  - The concept
  - Complexity and implementation details
  - Fast bilateral filter — code
  - Bilateral solver
- Non-local means
  - The concept
  - Complexity and implementation details
  - PCA-NLM — code

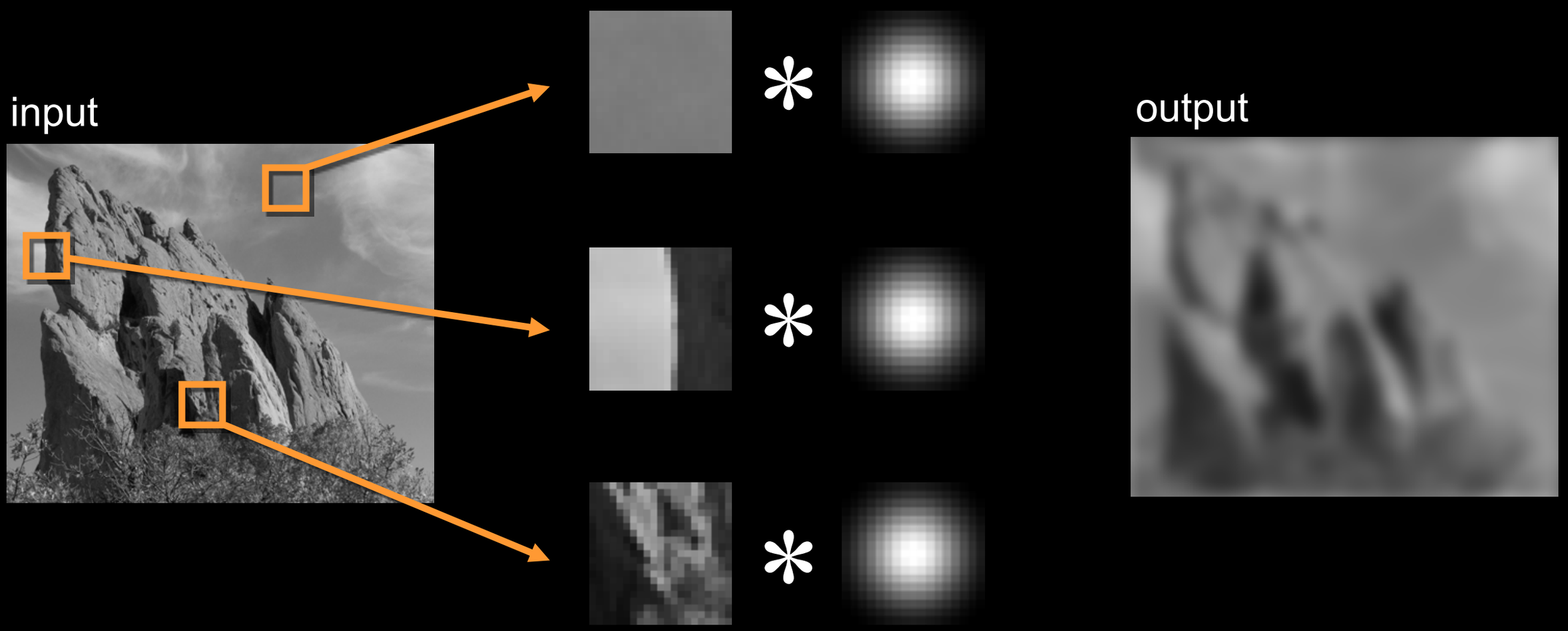
Bilateral Filter

- [IMPORTANT] Not to be confused with “bilinear” filter that we learned in the previous classes

- Bilinear interpolation — filling the missing values through linear interpolation (which is not in our best interests today).

- Bilateral filter — a non-linear, edge-preserving, noise-reducing filter.

Problem with Gaussian blur?

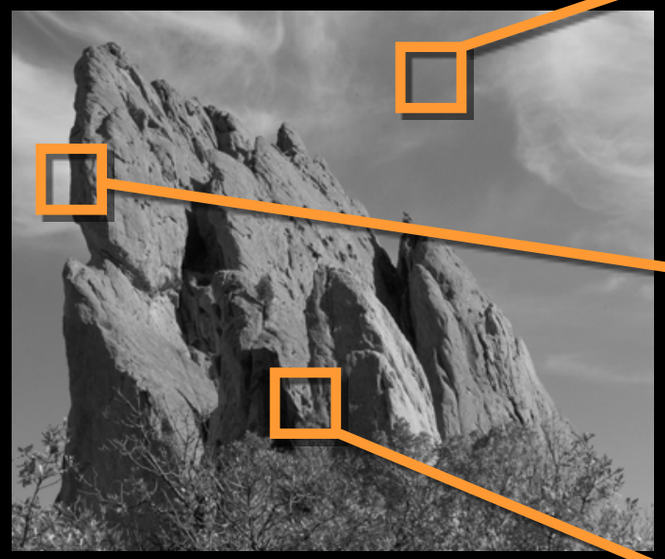


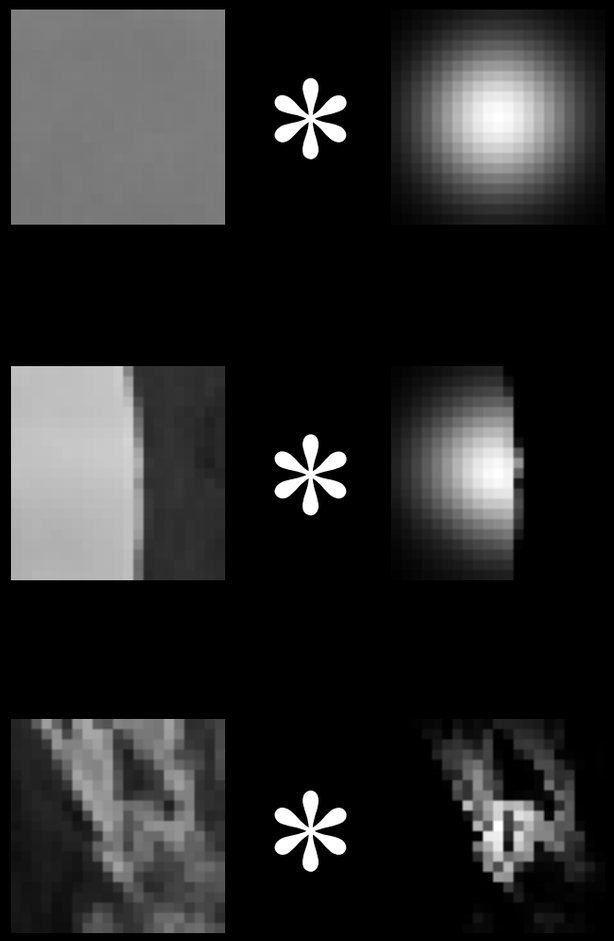
Same Gaussian kernel everywhere.

What do we rather want?

Bilateral filter!

input





output

The kernel shape depends on the image content.

[Aurich 95, Smith 97, Tomasi 98]

Bilateral filter: definition

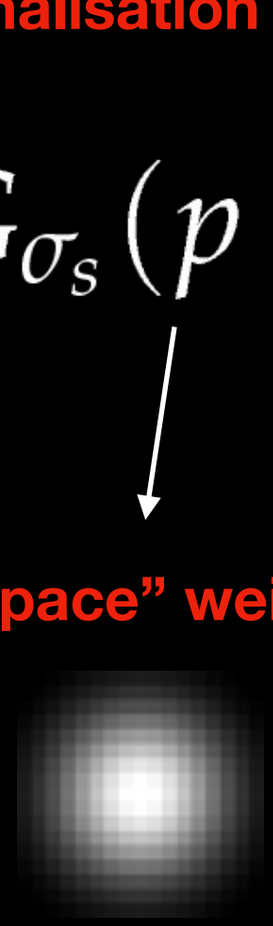
Same idea: weighted average of pixels

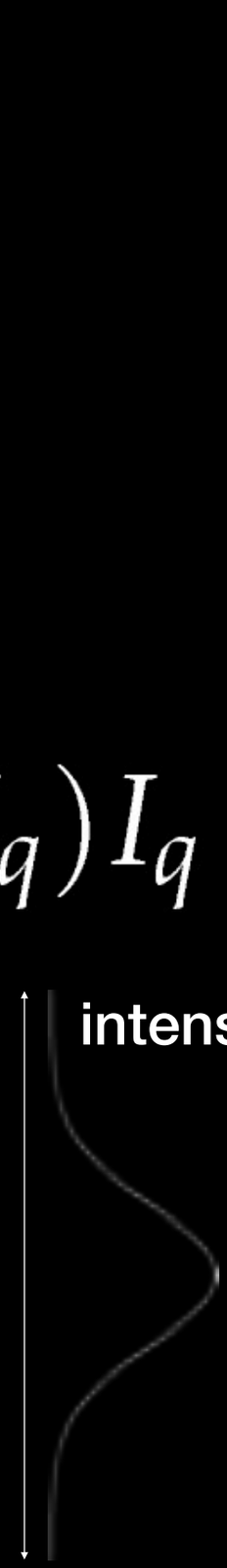
$$BF[I_p] = \boxed{\frac{1}{W}} \sum_{q \in S} G_{\sigma_s}(p-q) G_{\sigma_r}(I_p - I_q) I_q$$

Normalisation

"space" weight

"range" weight





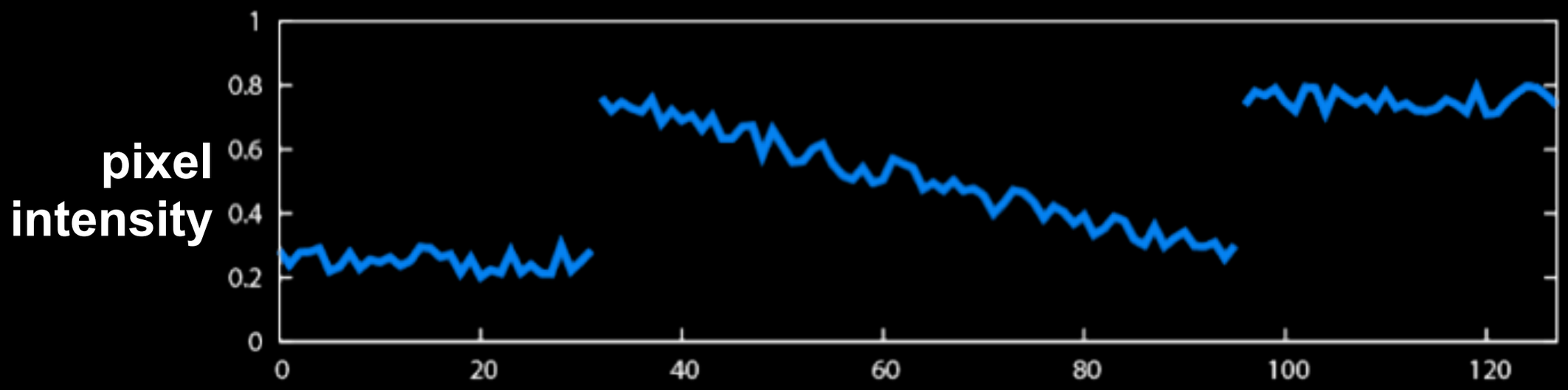
$$G_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}}$$

$$W = \sum_{q \in S} G_{\sigma_s}(p-q) G_{\sigma_r}(I_p - I_q)$$

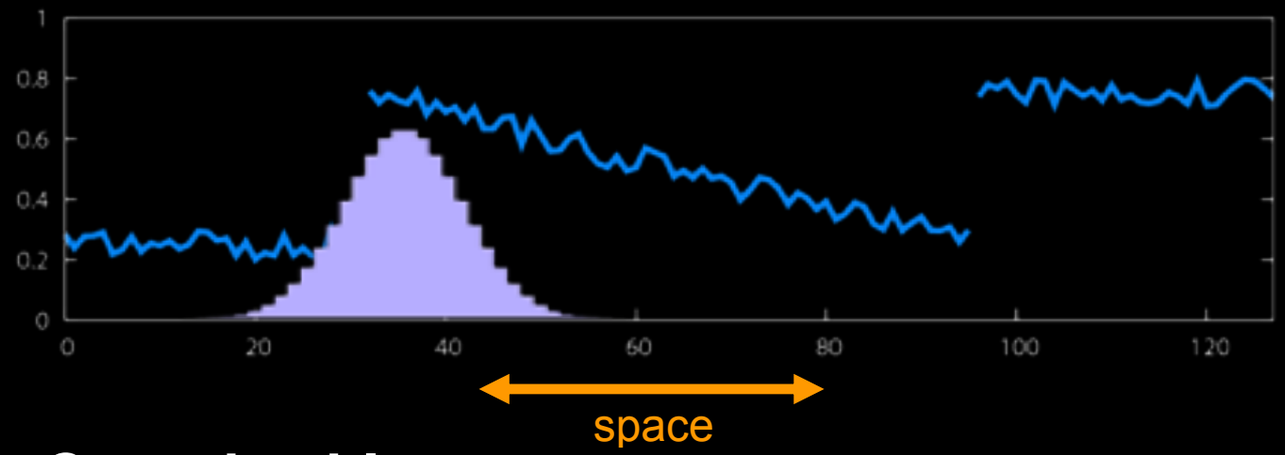
Illustration: 1D

1D image = line of pixels

Better visualised as a plot

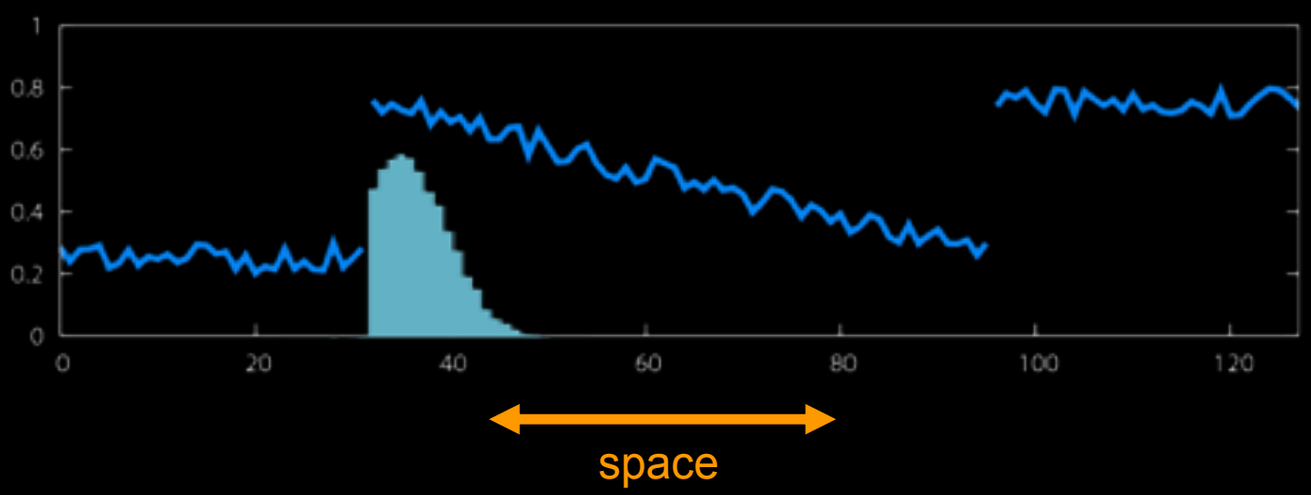


Gaussian blur vs. Bilateral filter



Gaussian blur

$$GB[I_p] = \sum_{q \in S} G_\sigma(p-q) I_q$$



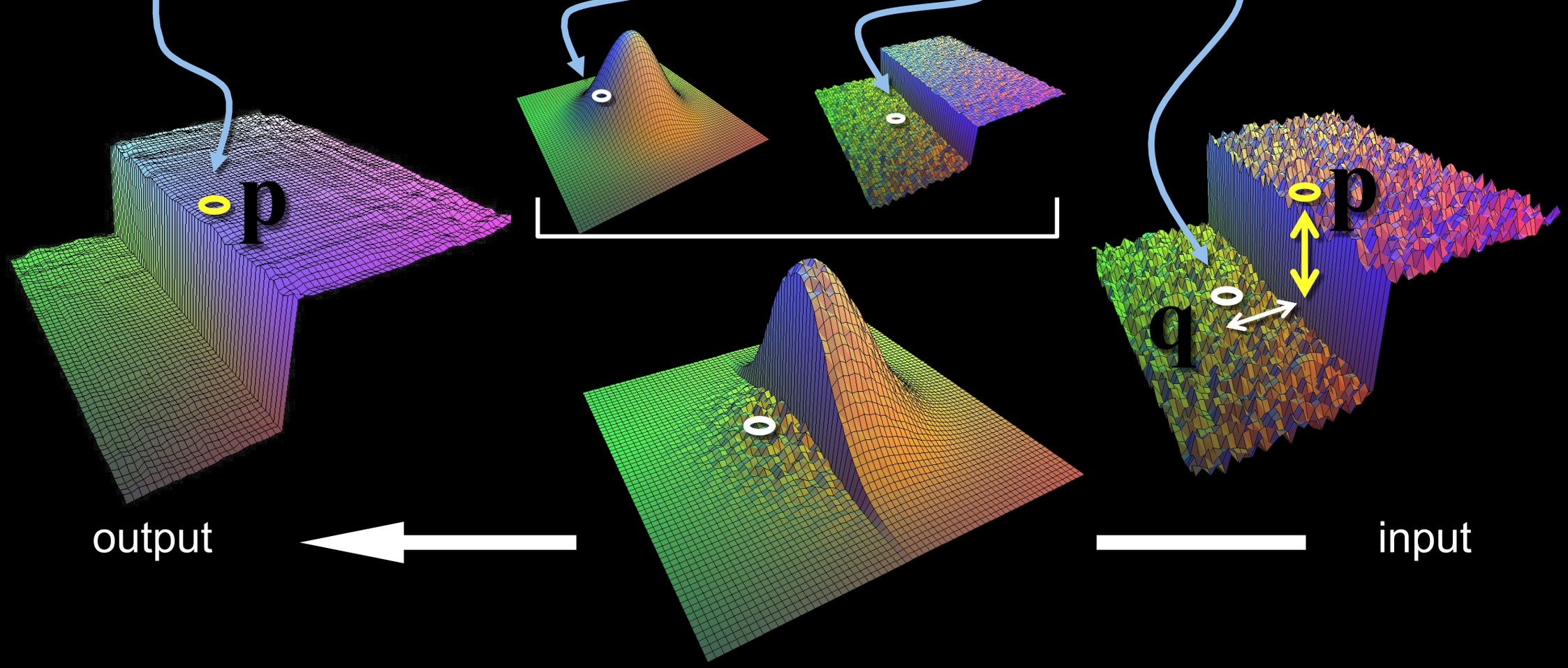
Bilateral filter

space range

$$BF[I_p] = \frac{1}{W} \sum_{q \in S} G_{\sigma_s}(p-q) G_{\sigma_r}(I_p - I_q) I_q$$

Bilateral filter on a height field

$$BF[I_p] = \frac{1}{W} \sum_{q \in S} G_{\sigma_s}(p - q) G_{\sigma_r}(I_p - I_q) I_q$$



(c) A course on bilateral filtering, SIGGRAPH 2008

reproduced from [Durand 02]

Space and range hyper-parameters

$$BF[I_p] = \frac{1}{W} \sum_{q \in S} G_{\sigma_s}(p - q) G_{\sigma_r}(I_p - I_q) I_q$$

space σ_s: spatial extent of the kernel, size of the considered neighbourhood.

range σ_r: the estimated standard deviation of the noise.

Complexity

$$BF[I_p] = \frac{1}{W} \sum_{q \in S} G_{\sigma_s}(p - q) G_{\sigma_r}(I_p - I_q) I_p$$

- for “each” pixel p
  - for “each” pixel q
    - compute: $G_{\sigma_s}(p - q) G_{\sigma_r}(I_p - I_q)$

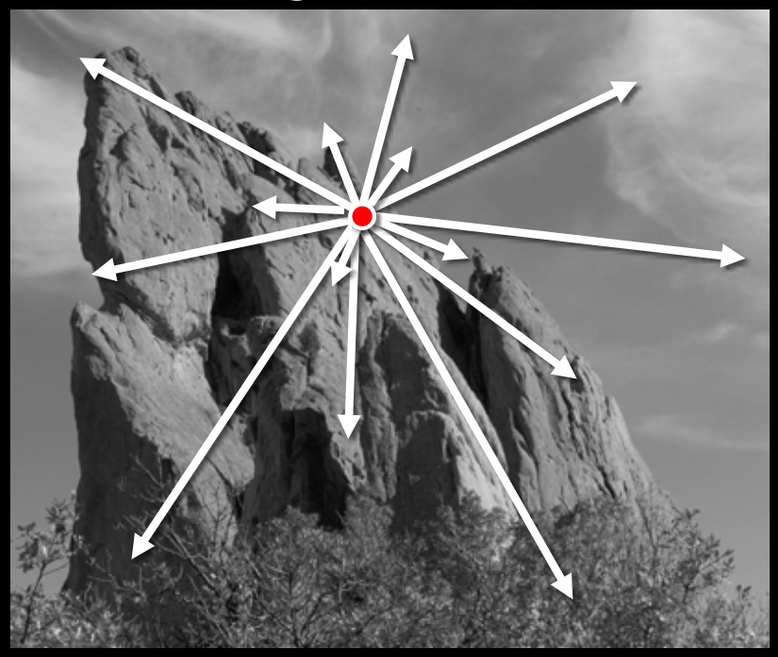
VERY SLOW!!

For an 8 MP image - takes 64,000,000,000,000 iterations

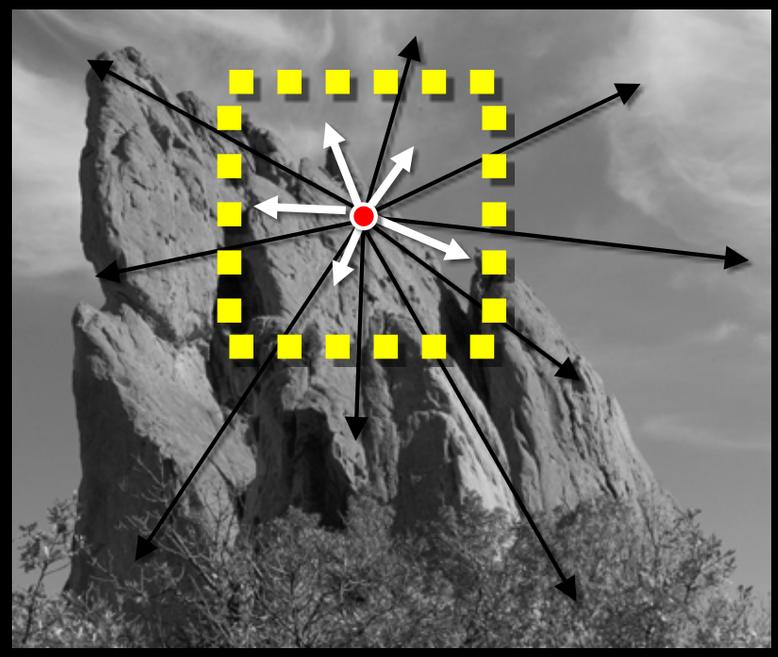
Better bilateral filter

Idea: Far away pixels are negligible, so don’t see them at all

looking at all pixels



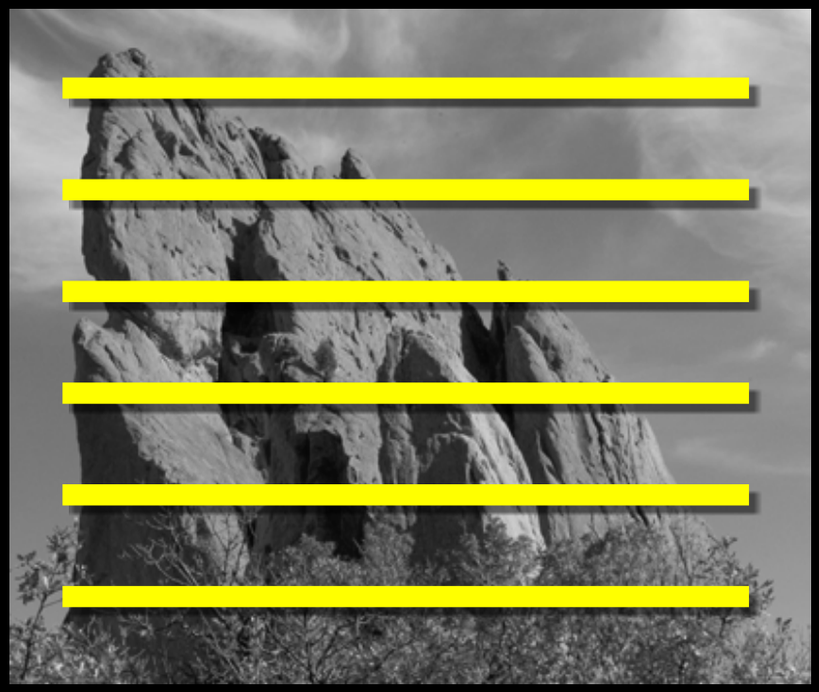
looking at neighbors only

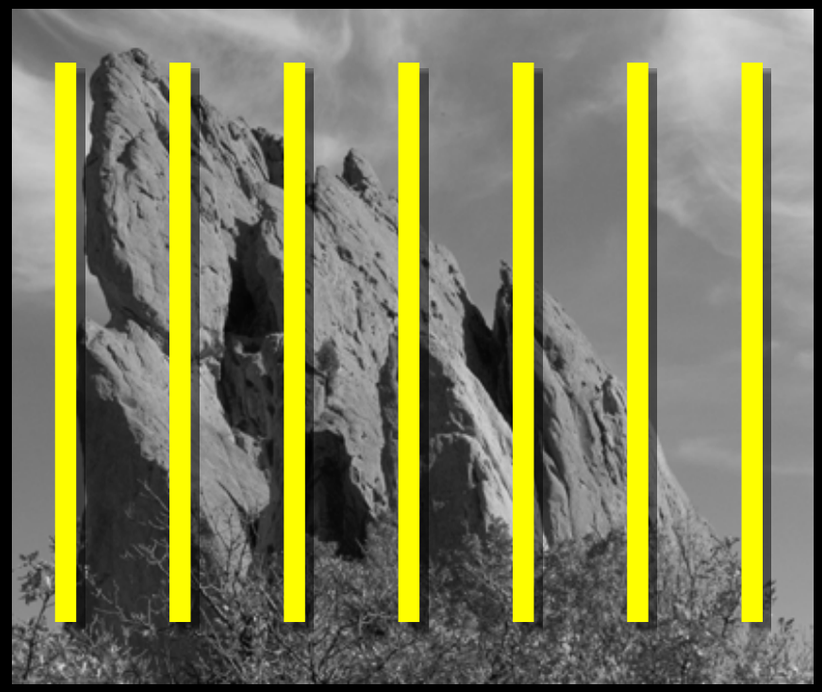


PyNotebook

Separable filter

Idea: filter the rows then the columns





Two “cheap” 1D filters instead of an “expensive” 2D filter

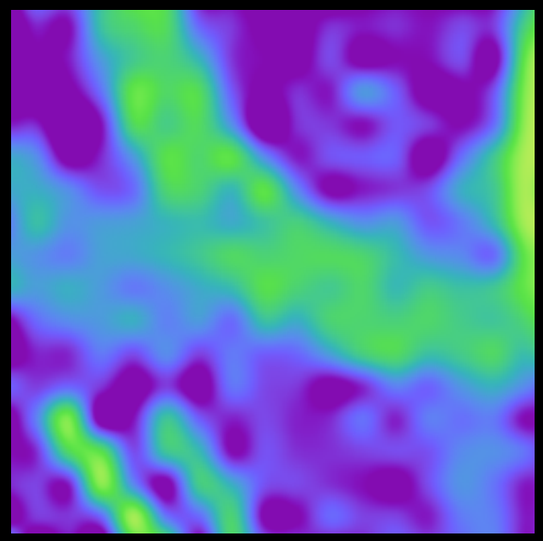
[Pham and Van Vliet 05]

Applications: denoising

Noisy input

Bilateral filter 7x7 window

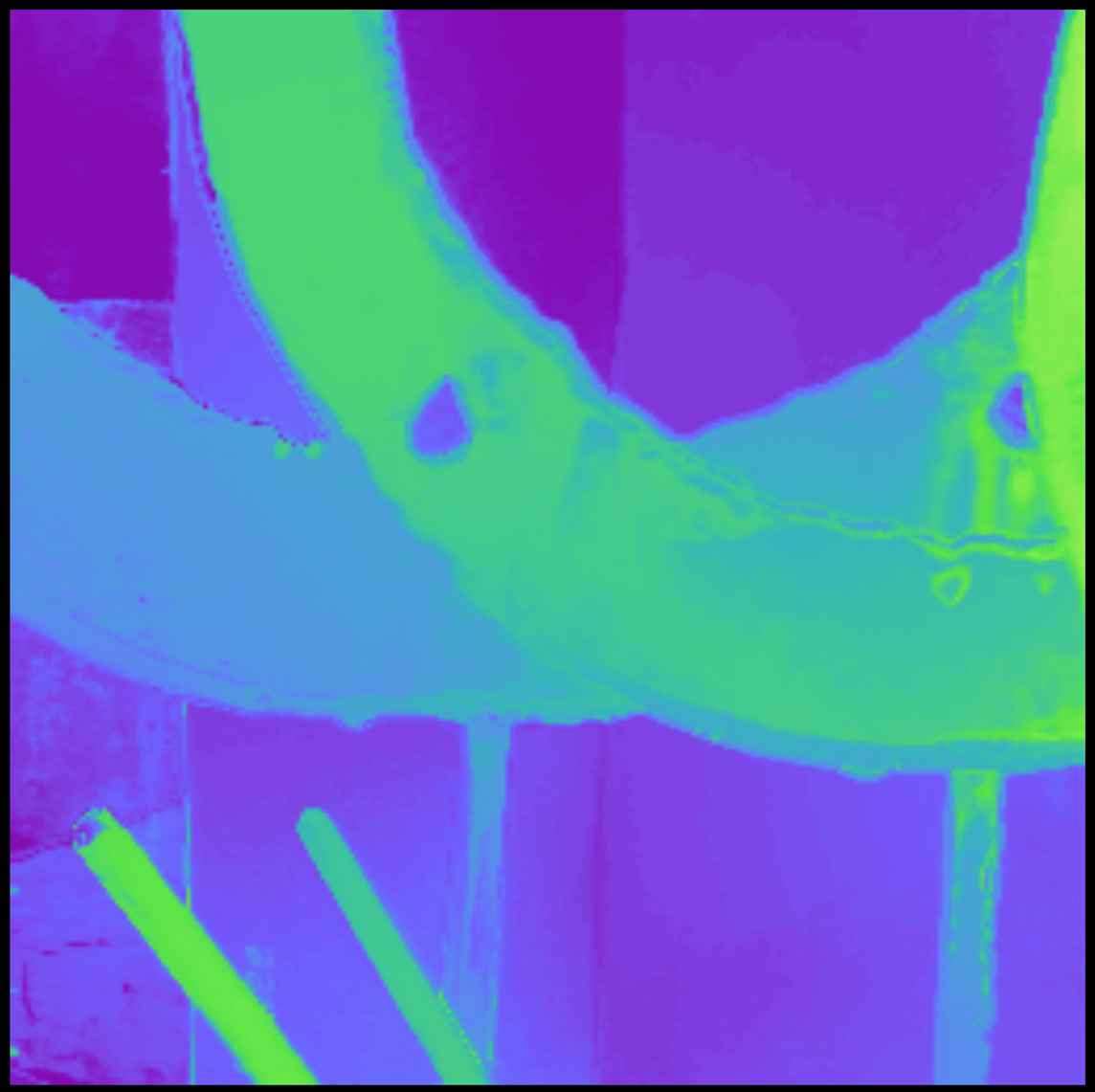
Semantic segmentation



Input $\mathbf{y}$

Referenc $\mathbf{R}$

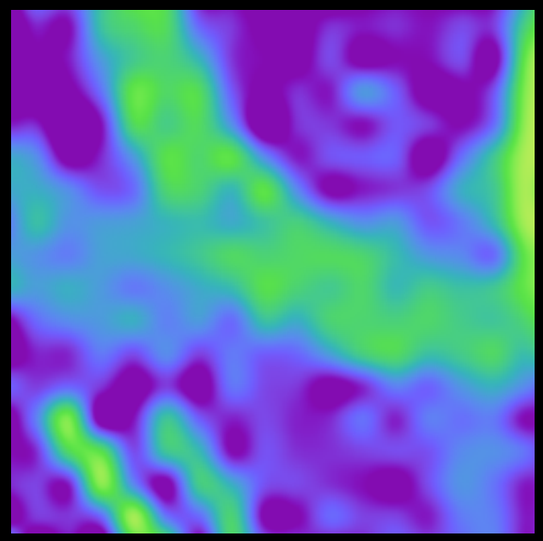
Bilateral Filter



Output $\mathbf{x}$

Blur the input with respect to the edges in the reference image.

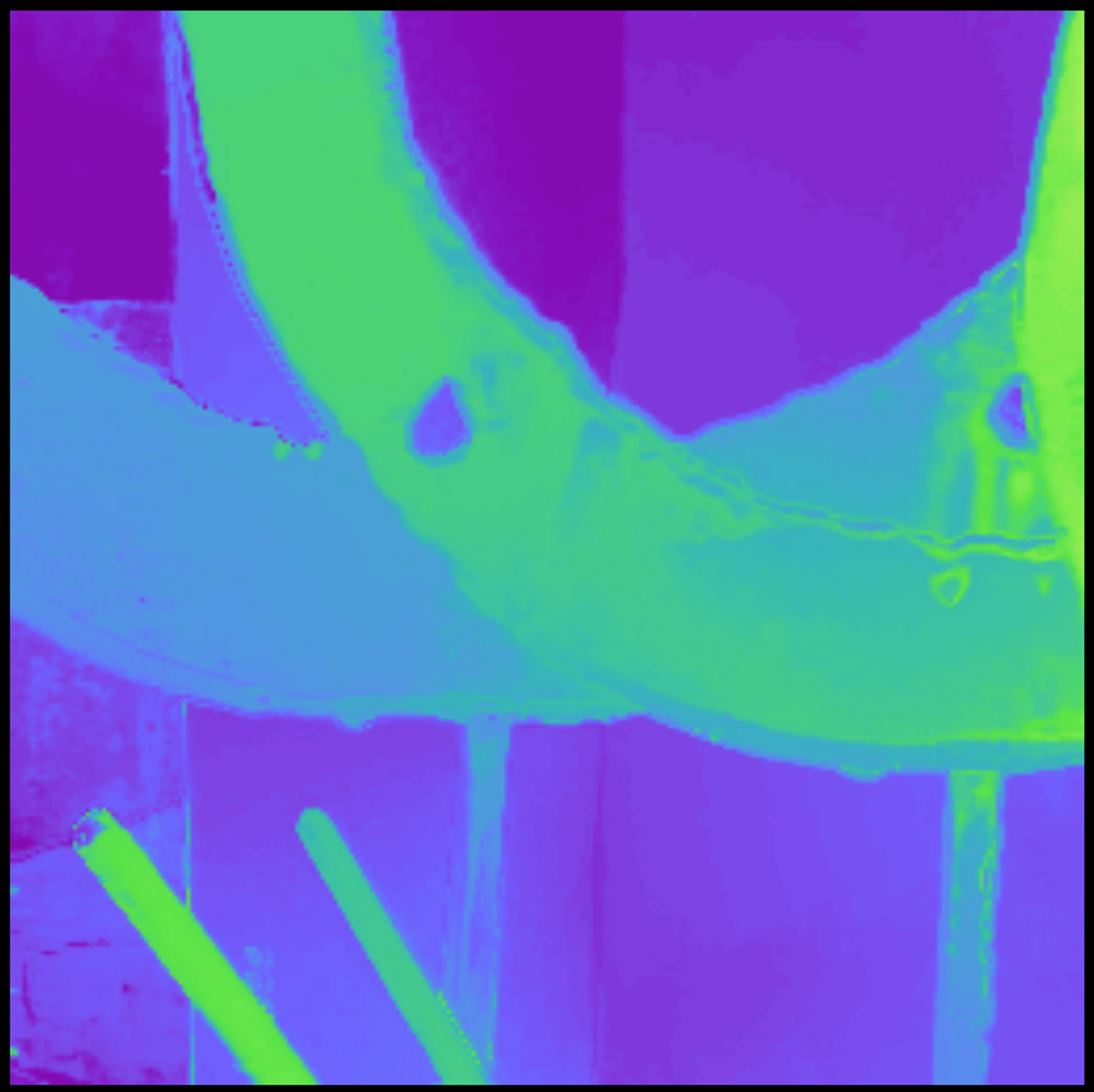
Semantic segmentation



Input $\mathbf{y}$

Bilateral Filter

Referenc $\mathbf{R}$

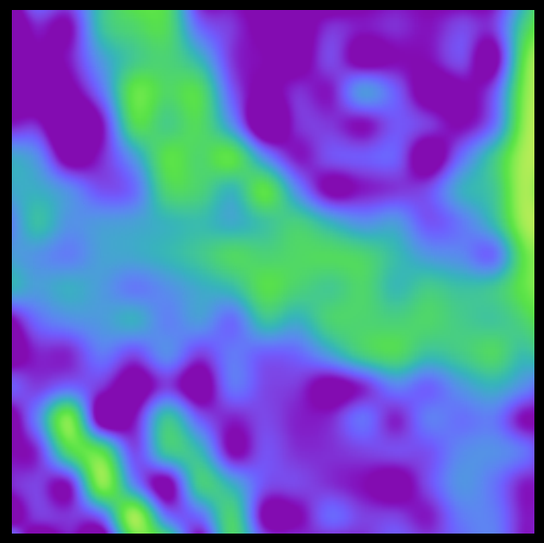


Output $\mathbf{x}$

$$\mathbf{x} \leftarrow \mathbf{W}\mathbf{y}$$

$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma_d^2}\right)$$

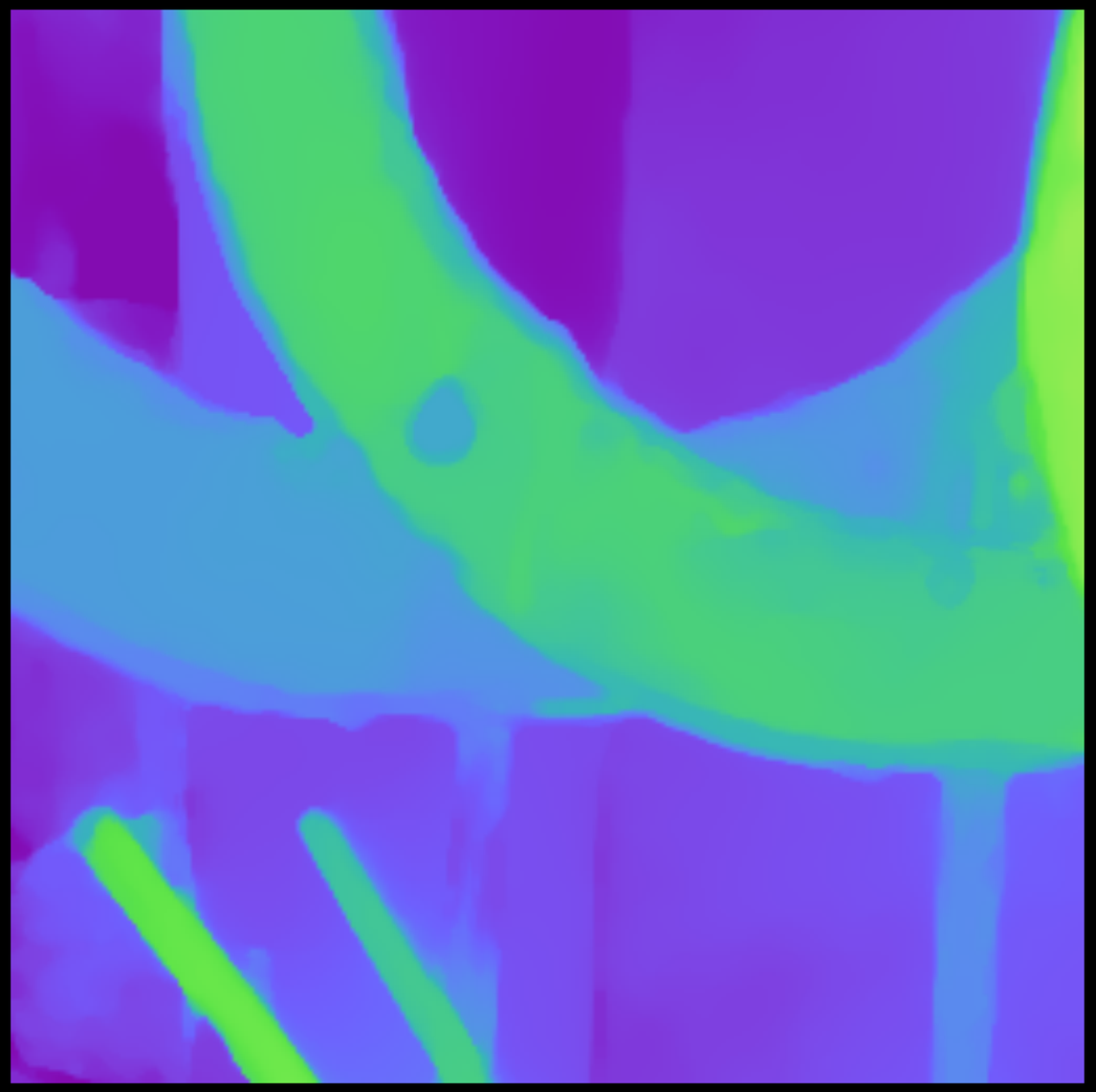
Bilateral Solver



Input $\mathbf{y}$

Referenc $\mathbf{R}$

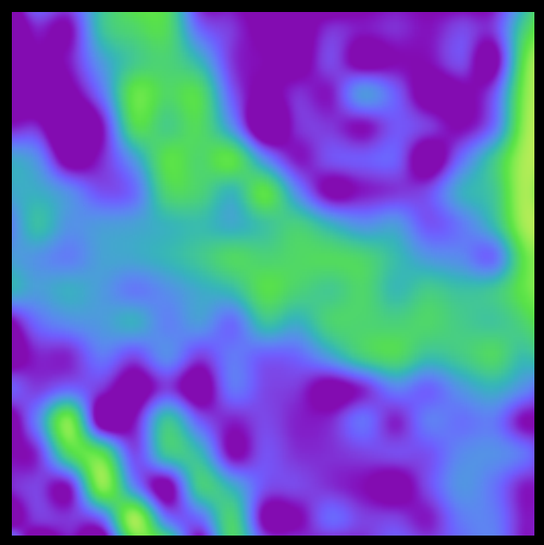
Bilateral Solver



Output $\mathbf{x}$

Find the image that is as smooth as possible with respect to the reference image, and as close as possible to the input.

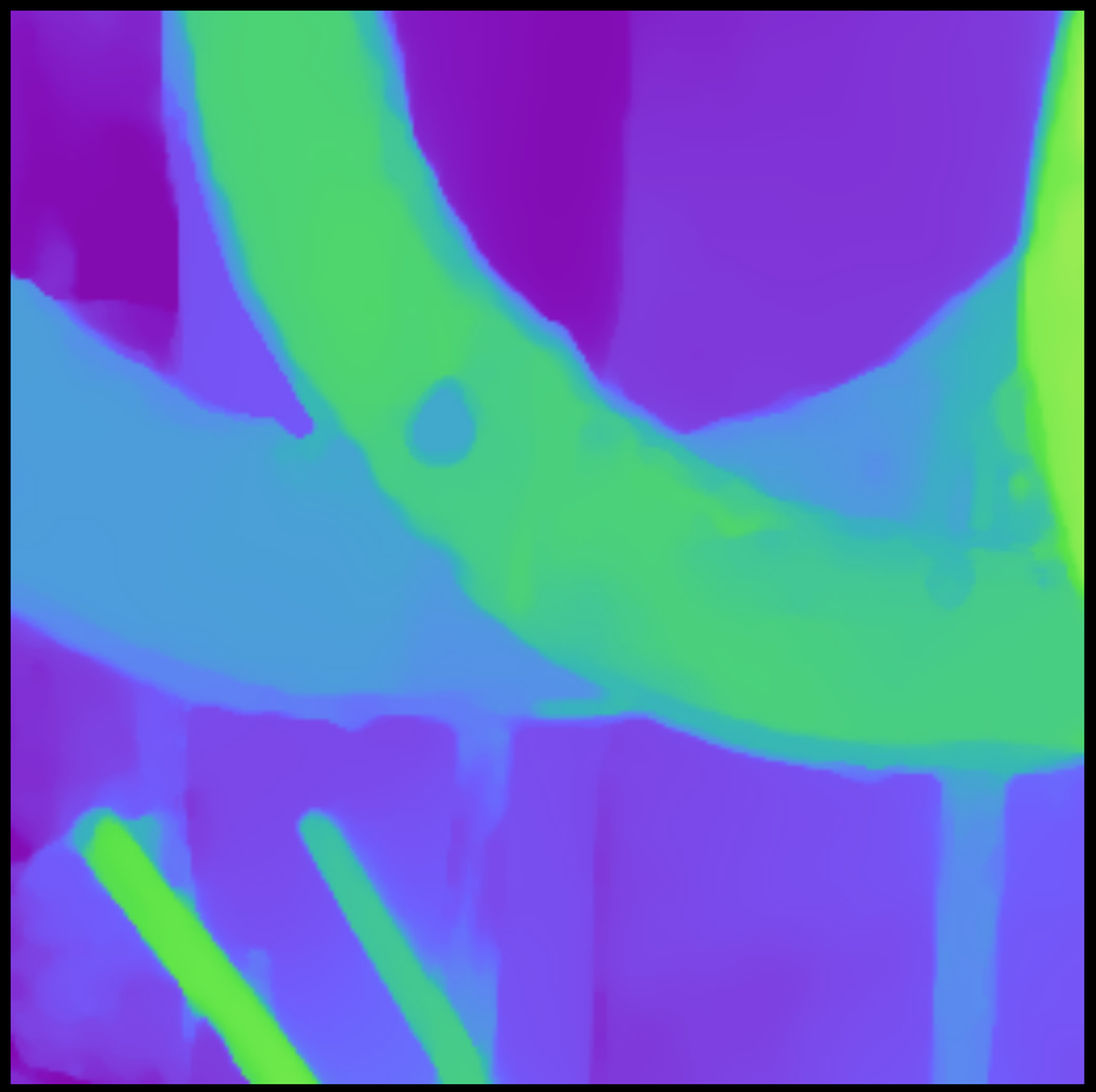
Bilateral Solver



Input $\mathbf{y}$

Referenc $\mathbf{R}$

Bilateral Solver



Output $\mathbf{x}$

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2} \sum_{i,j} W_{i,j} \left(x_i - x_j\right)^2 + \sum_i (x_i - y_i)^2$$

Non-local means

Bilateral filter: Average neighbours with similar intensities
Non-local means: Average neighbours with similar neighbourhoods

for every pixel p

Buades, 2005

Non-local means

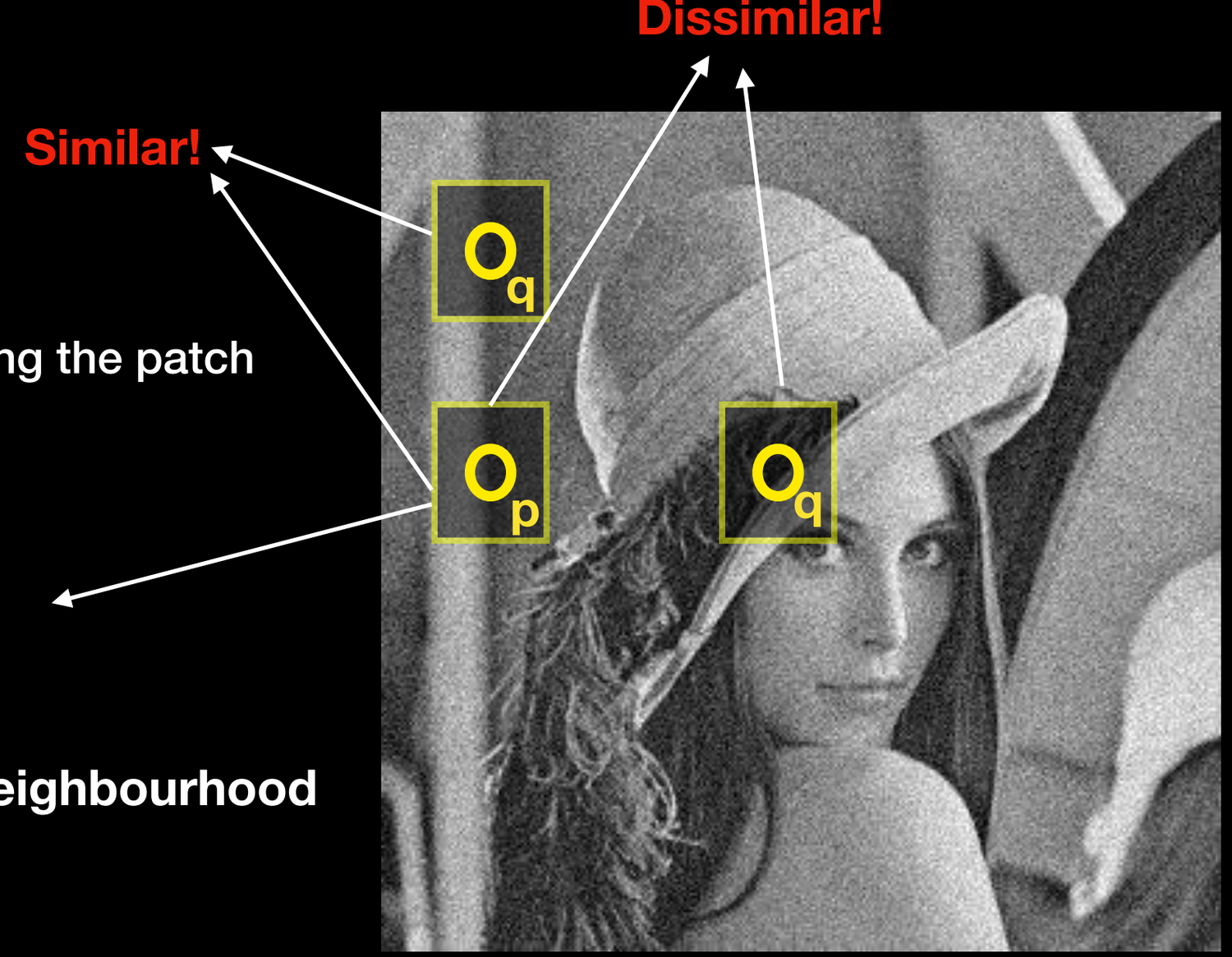
Bilateral filter: Average neighbours with similar intensities

Non-local means: Average neighbours with similar neighbourhoods!

Filter with this!

$$||v_p - v_q||^2$$

v_p, v_q are vectors describing the patch



for every pixel p

- **define a small neighbourhood**

Buades, 2005

Non-local means

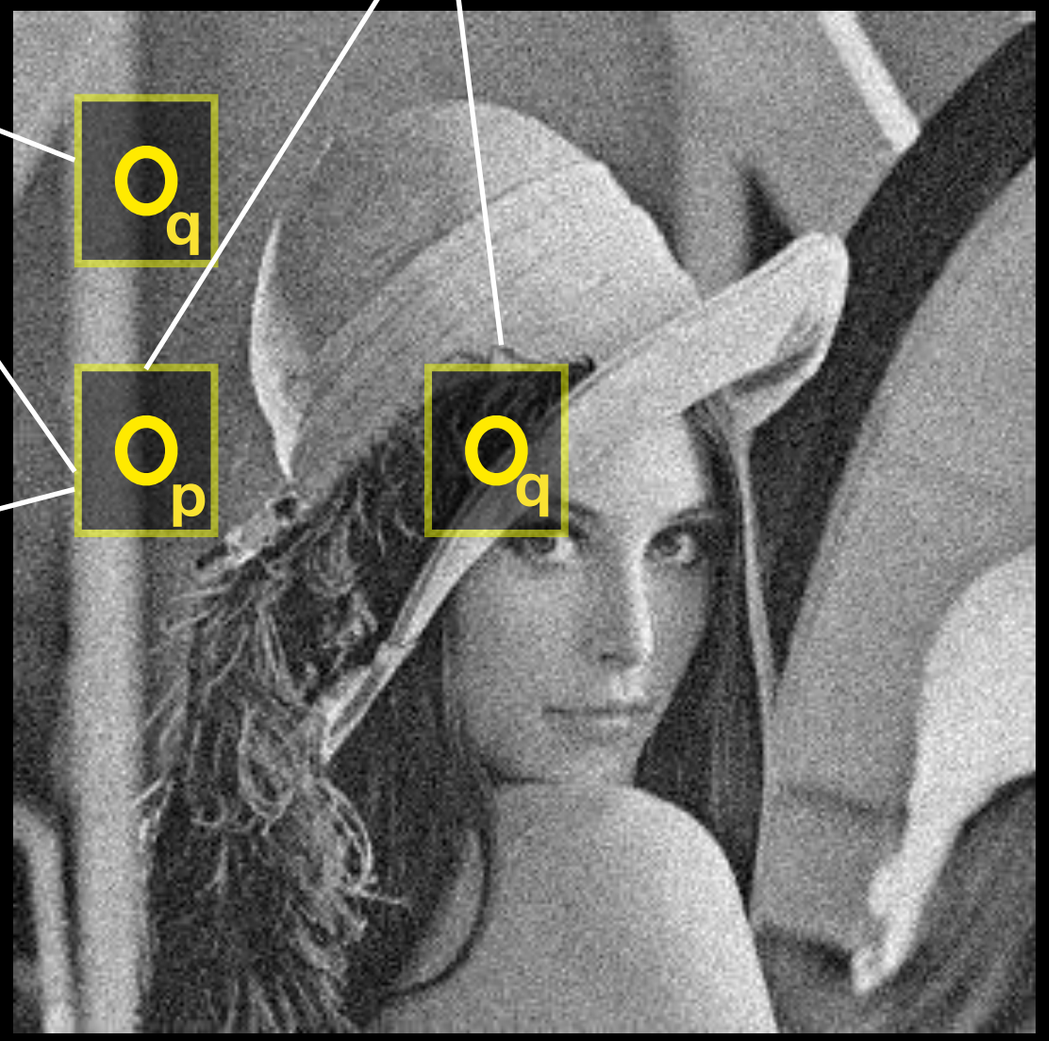
Filter with this!

$$||v_p - v_q||^2$$

v_p, v_q are vectors describing the patch

for every pixel p

- define a small neighbourhood
- define vectors v_p, v_q



No spatial term

$$NLM[I_p] = \frac{1}{W} \sum_{q \in S} \cancel{G_{\sigma_s}(p - q)} G_{\sigma_r}(||v_p - v_q||^2) I_p$$

Buades, 2005

Non-local means

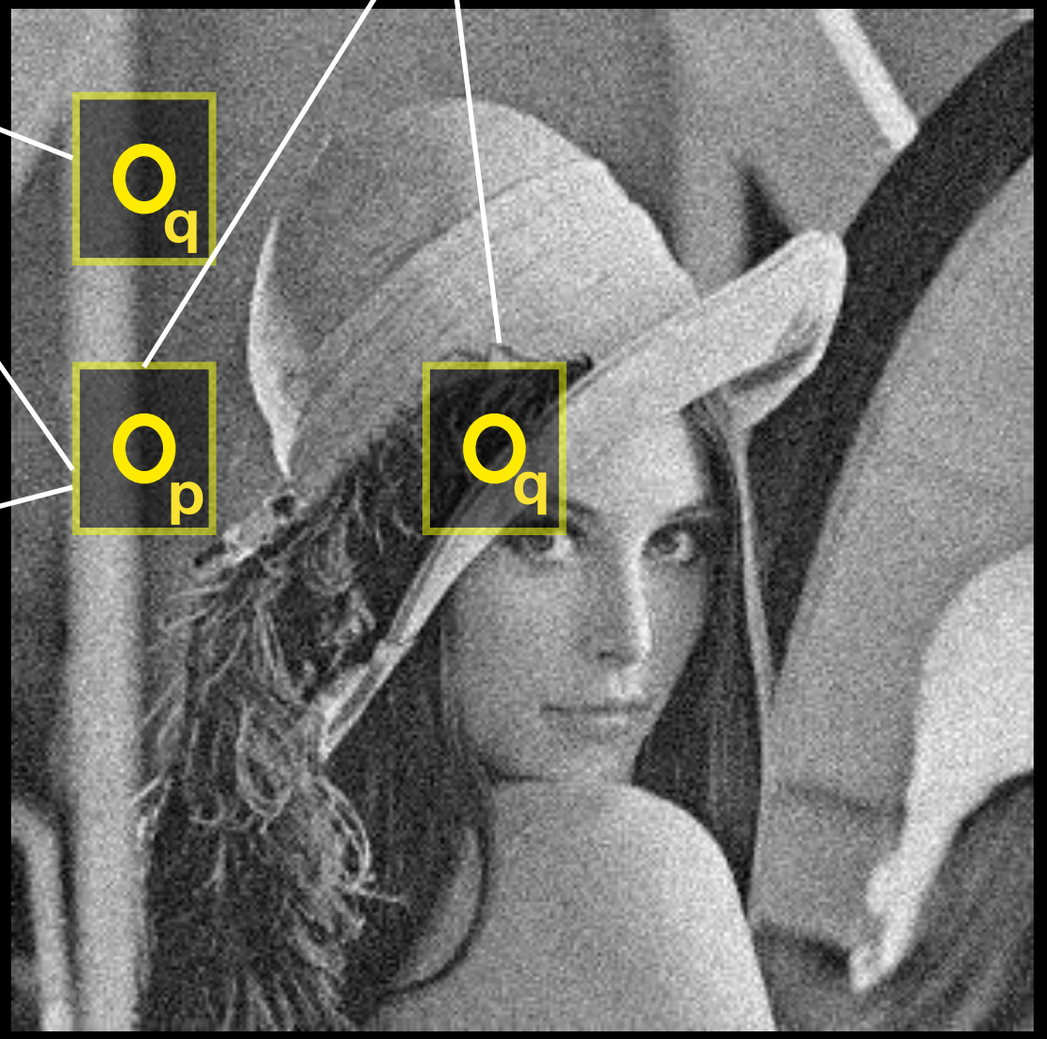
Filter with this!

$$||v_p - v_q||^2$$

v_p, v_q are vectors describing the patch

for every pixel p

- define a small neighbourhood
- define vectors v_p, v_q

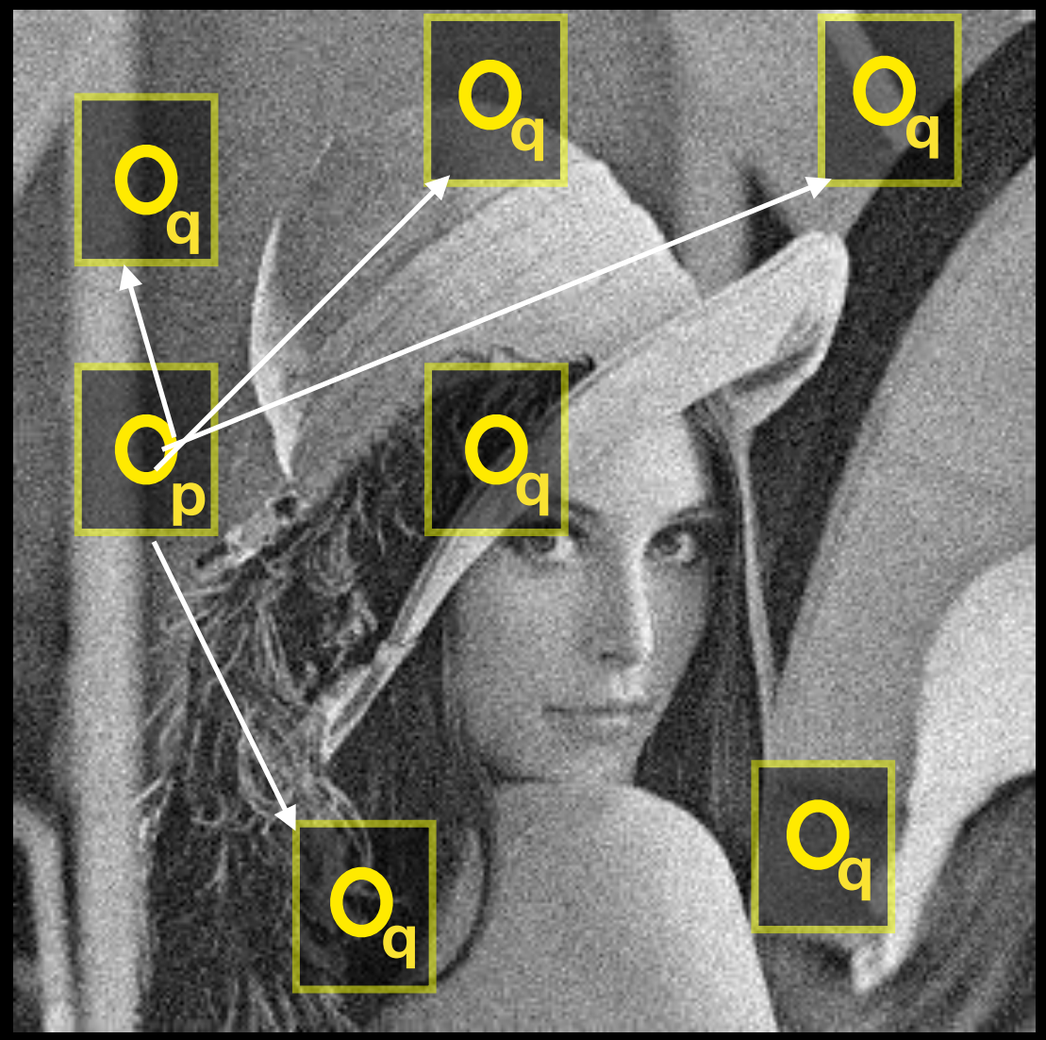


$$NLM[I_p] = \frac{1}{W} \sum_{q \in S} G_{\sigma_r}(||v_p - v_q||^2) I_p$$

Buades, 2005

Non-local means: complexity?

$$NLM[I_p] = \frac{1}{W} \sum_{q \in S} G_{\sigma_r}(||v_p - v_q||^2) I_p$$

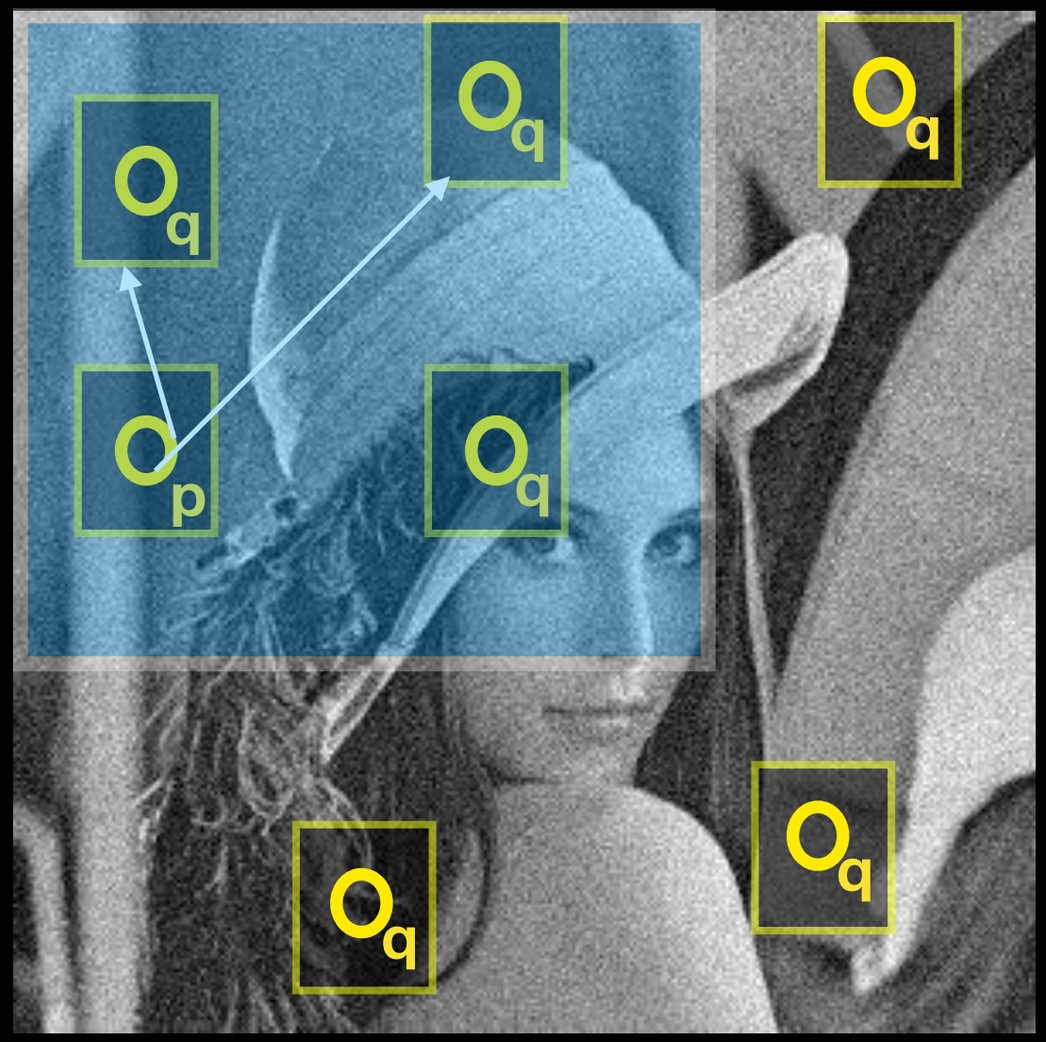


Problem: Too many patches to search!

Buades, 2005

Non-local means: complexity?

$$NLM[I_p] = \frac{1}{W} \sum_{q \in S} G_{\sigma_r}(||v_p - v_q||^2) I_p$$



Problem: Too many patches to search!

Simple remedy: confine the search to an extended neighbourhood.

Buades, 2005

Non-local means: faster?

PyNotebook

- Get all the patches from the image
- Compute their PCA — know which patterns are more descriptive than others.
- Since low-importance components are mostly noise — we can discard them. We reduce the patch size to a smaller number.
- Index all the reduced patch-vectors in a tree and query over it.