

# Tutorial-7: Bilateral filtering, Non-local means

Sanketh Vedula  
[sanketh@cs.technion.ac.il](mailto:sanketh@cs.technion.ac.il)

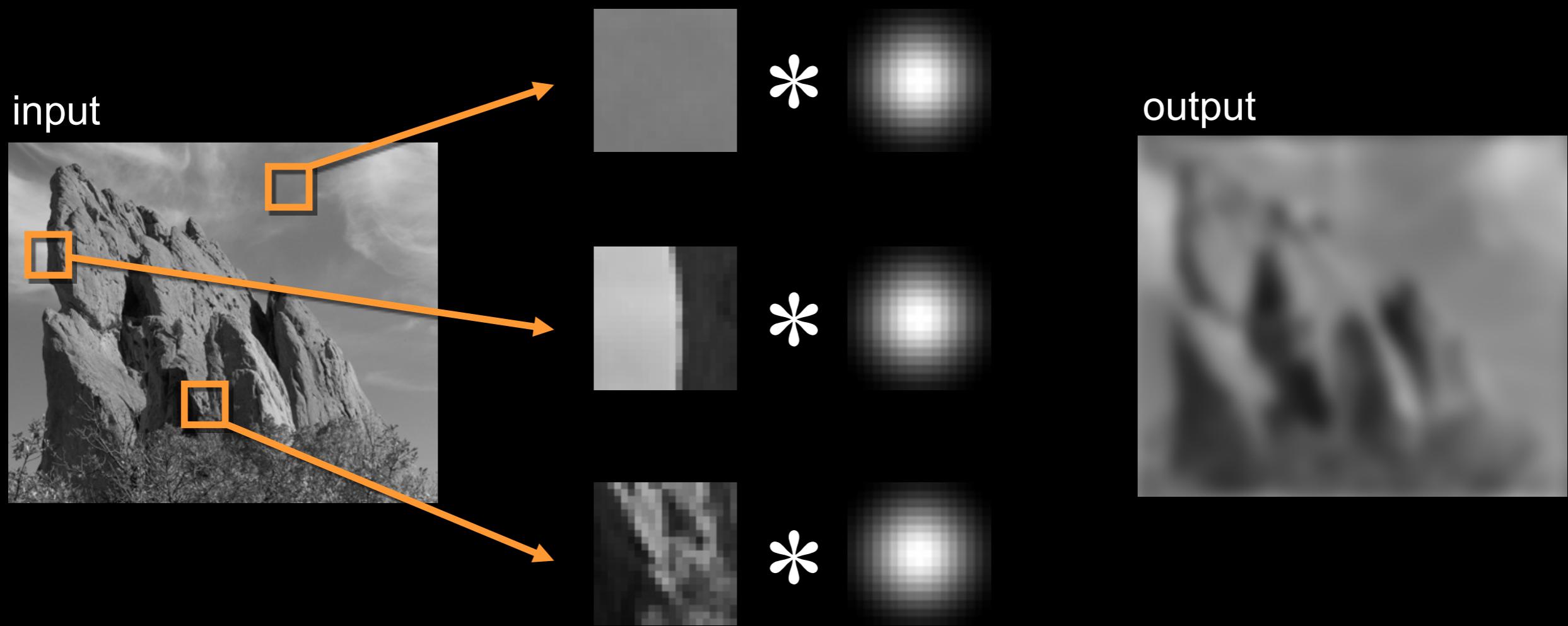
# Agenda

- Bilateral filtering
  - The concept
  - Complexity — Implementation details
  - Fast bilateral filter — code
  - Bilateral solver
- Non-local means
  - The concept
  - Implementation details — code
  - 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.

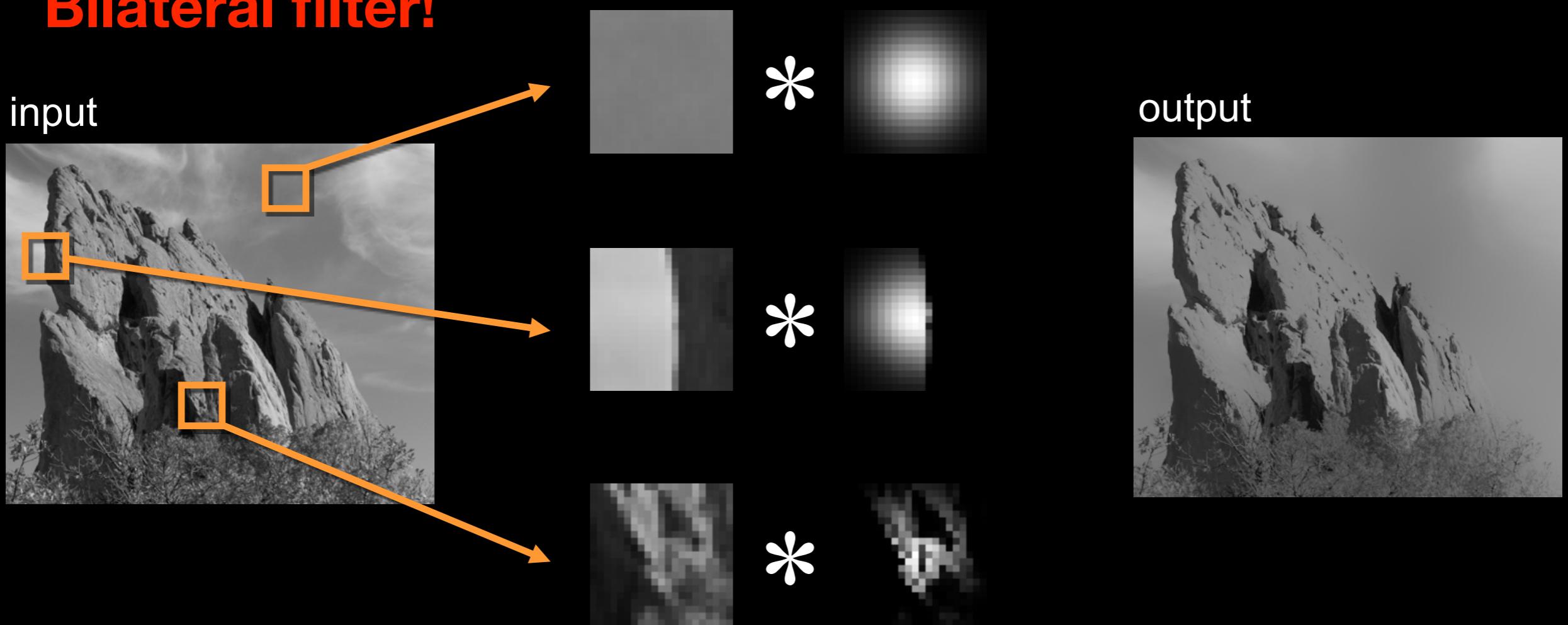
# Problem with Gaussian blur?



Same Gaussian kernel everywhere.

# What do we rather want?

**Bilateral filter!**



The kernel shape depends on the image content.

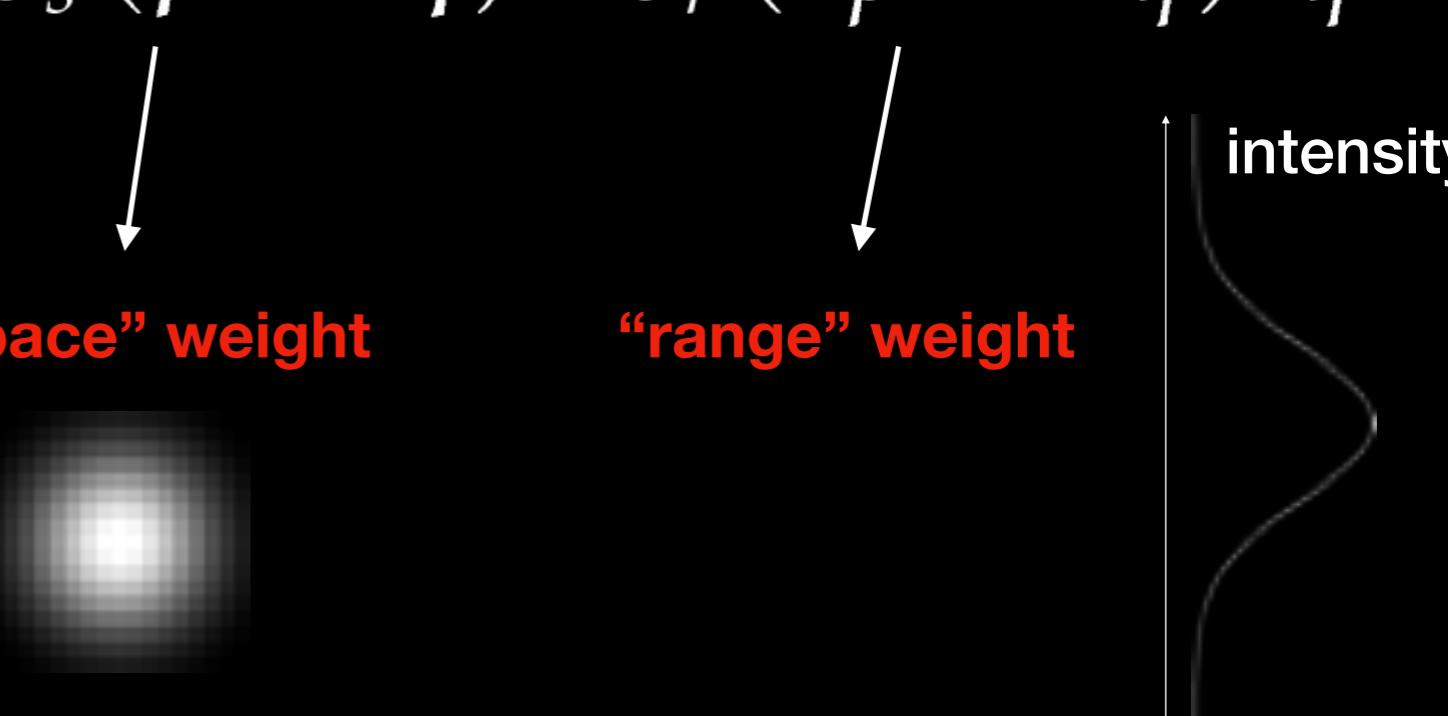
# Bilateral filter: definition

Same idea: weighted average of pixels

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

Normalisation

“space” weight      “range” weight



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

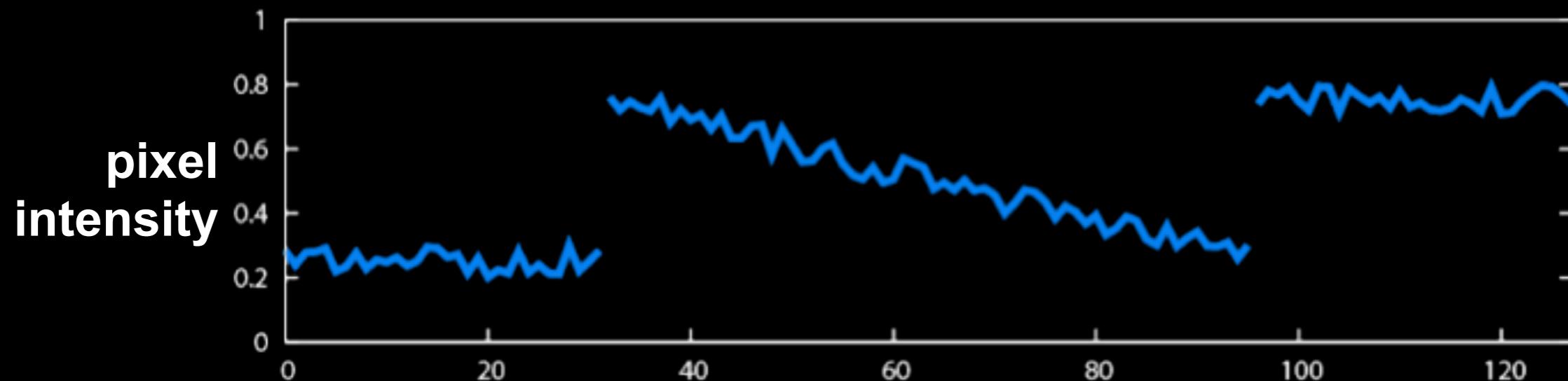
$$W = \sum_{q \in S} G_{\sigma_s}(p - q) G_{\sigma_r}(I_p - I_q)$$

# Illustration: 1D

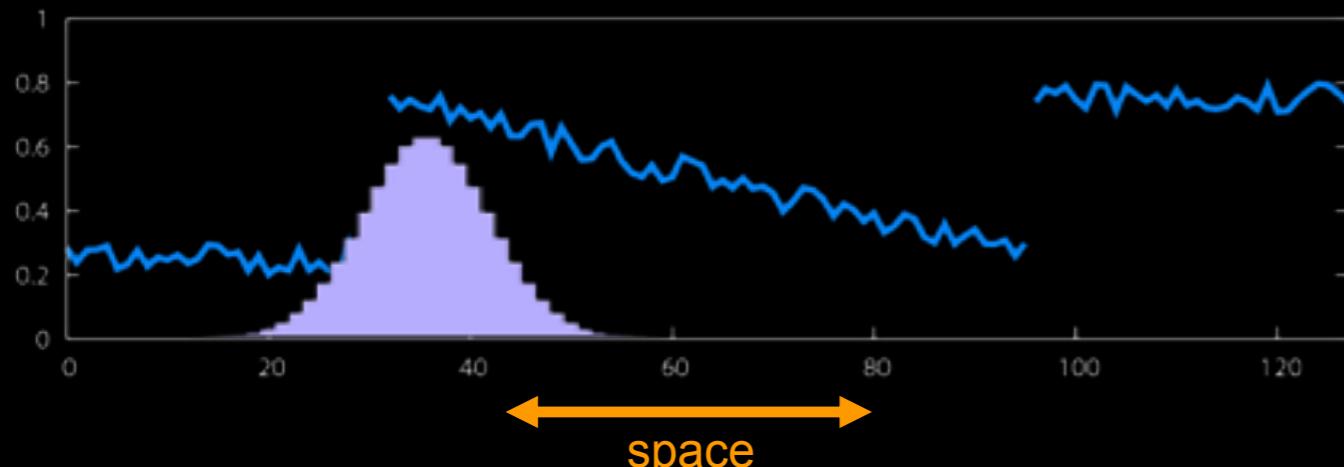
1D image = line of pixels



Better visualised as a plot

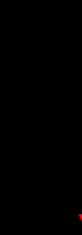


# Gaussian blur vs. Bilateral filter



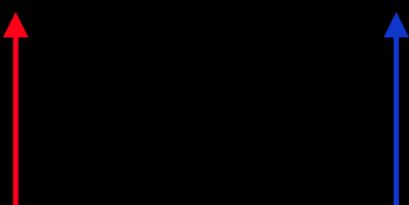
**Gaussian blur**

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

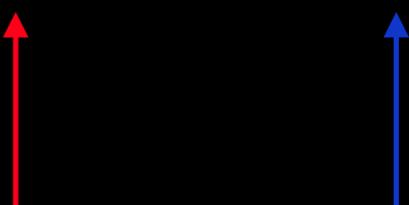


**space**      **range**

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

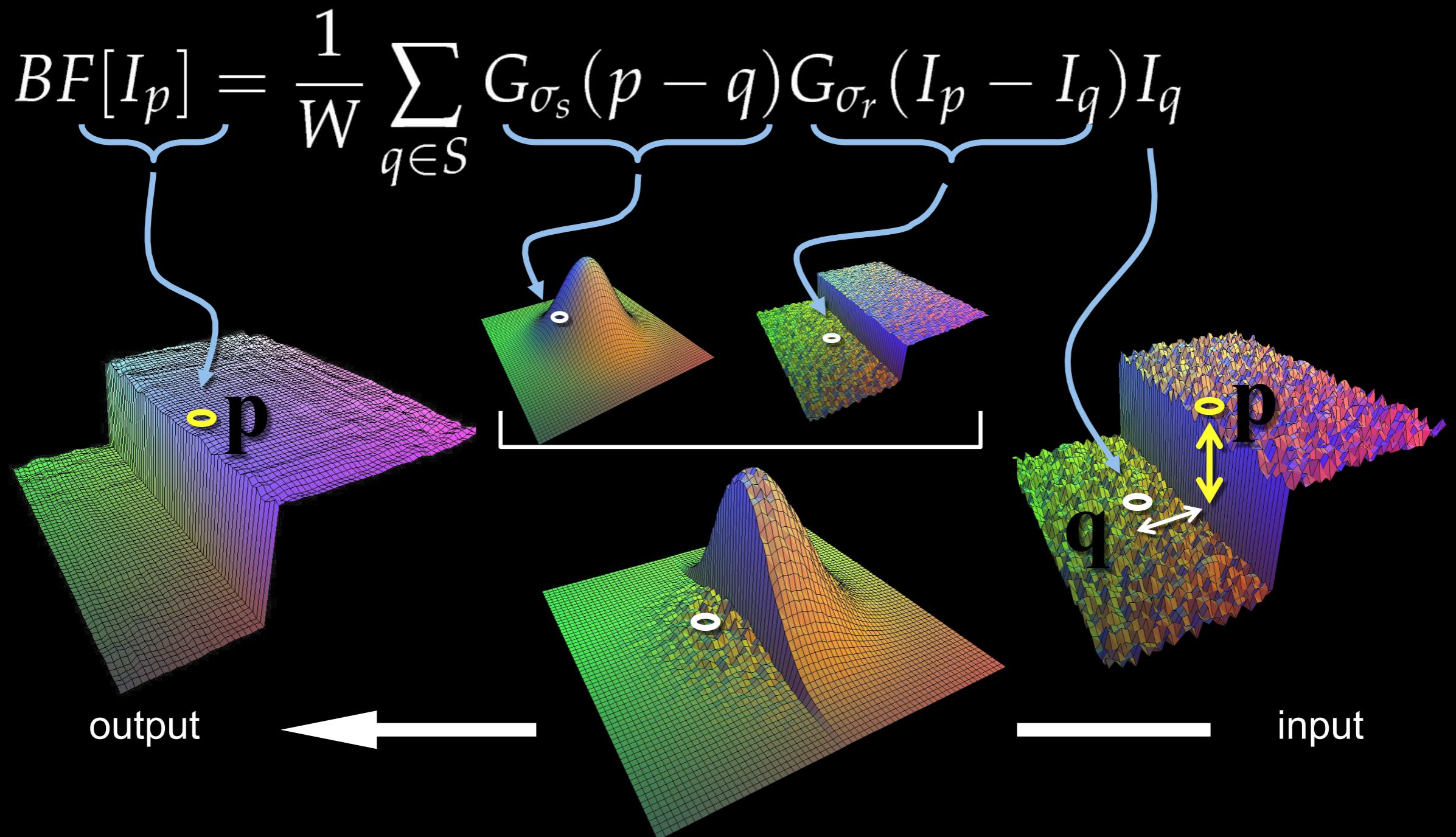


**Bilateral filter**



**space**      **range**

# Bilateral filter on a height field



# Space and range 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  $\sigma_s$ : spatial extent of the kernel, size of the considered neighbourhood.

range  $\sigma_r$ : “minimum” amplitude of an edge

# 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:  $G_{\sigma_s}(p - q) G_{\sigma_r}(I_p - I_q)$

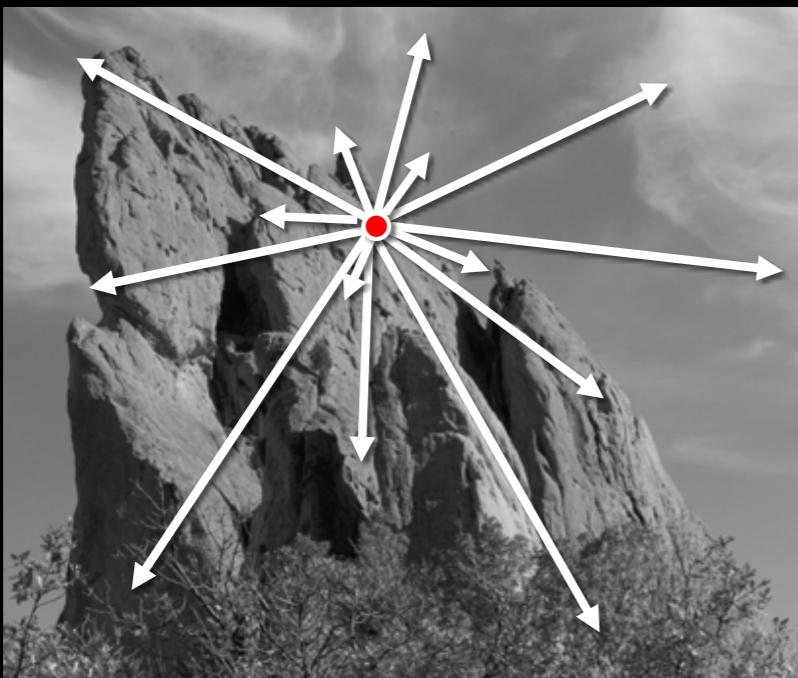
**VERY SLOW!!**

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

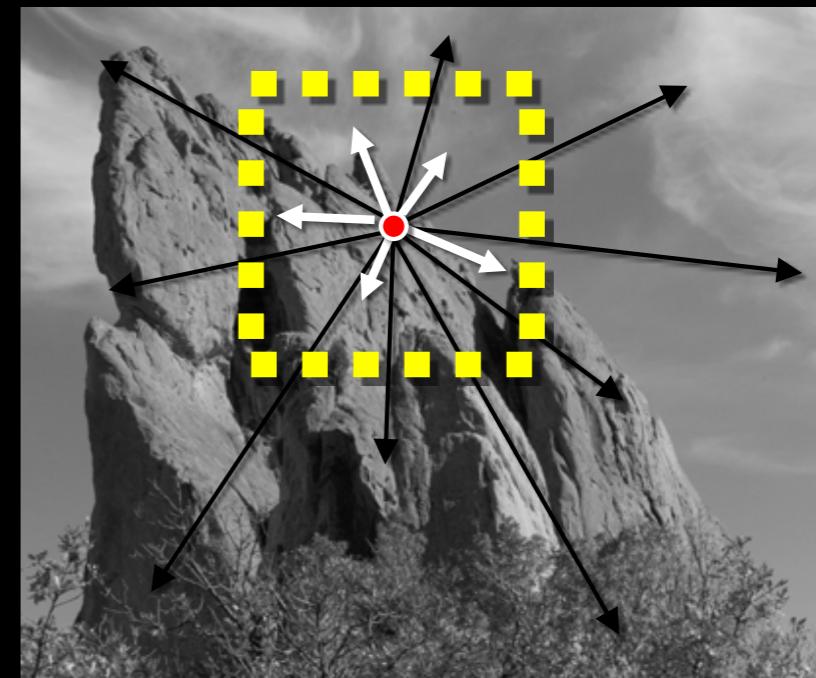
# Better bilateral filter

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

looking at all pixels



looking at neighbors only



# Separable filter

Idea: filter the rows then the columns



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

# Applications: denoising

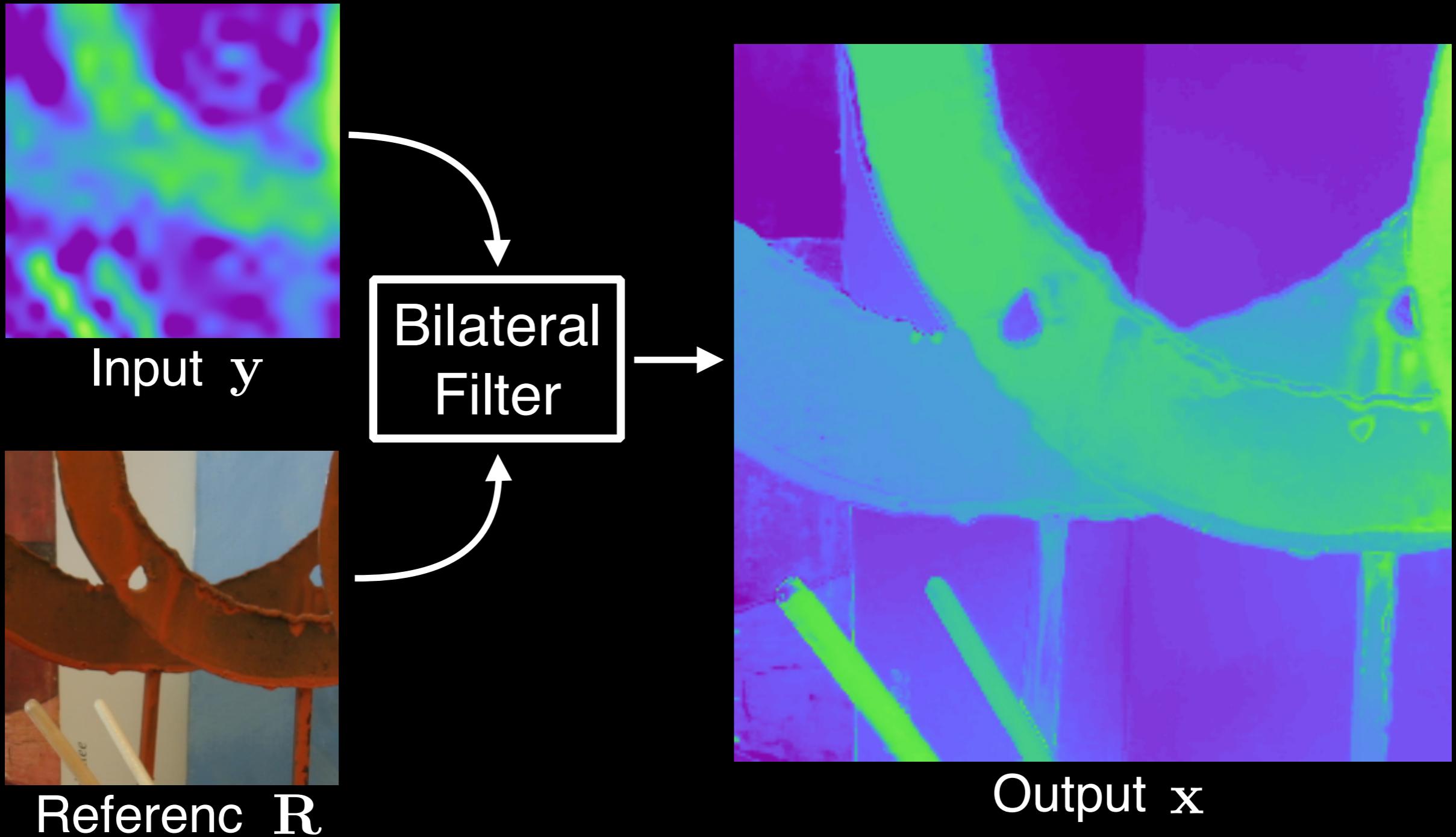
Noisy input



Bilateral filter 7x7 window

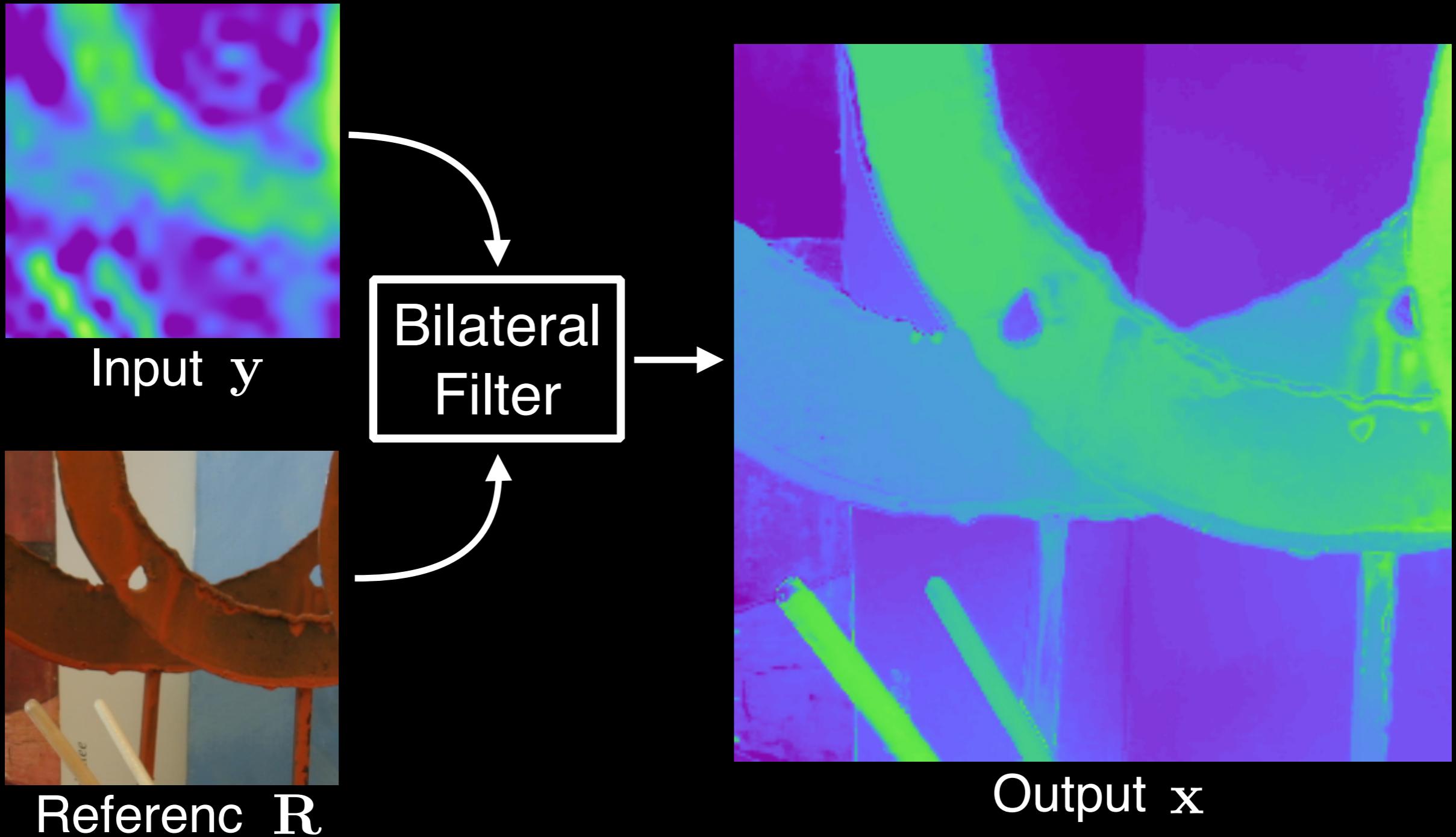


# Semantic segmentation



Blur the input while respecting edges in the reference image.

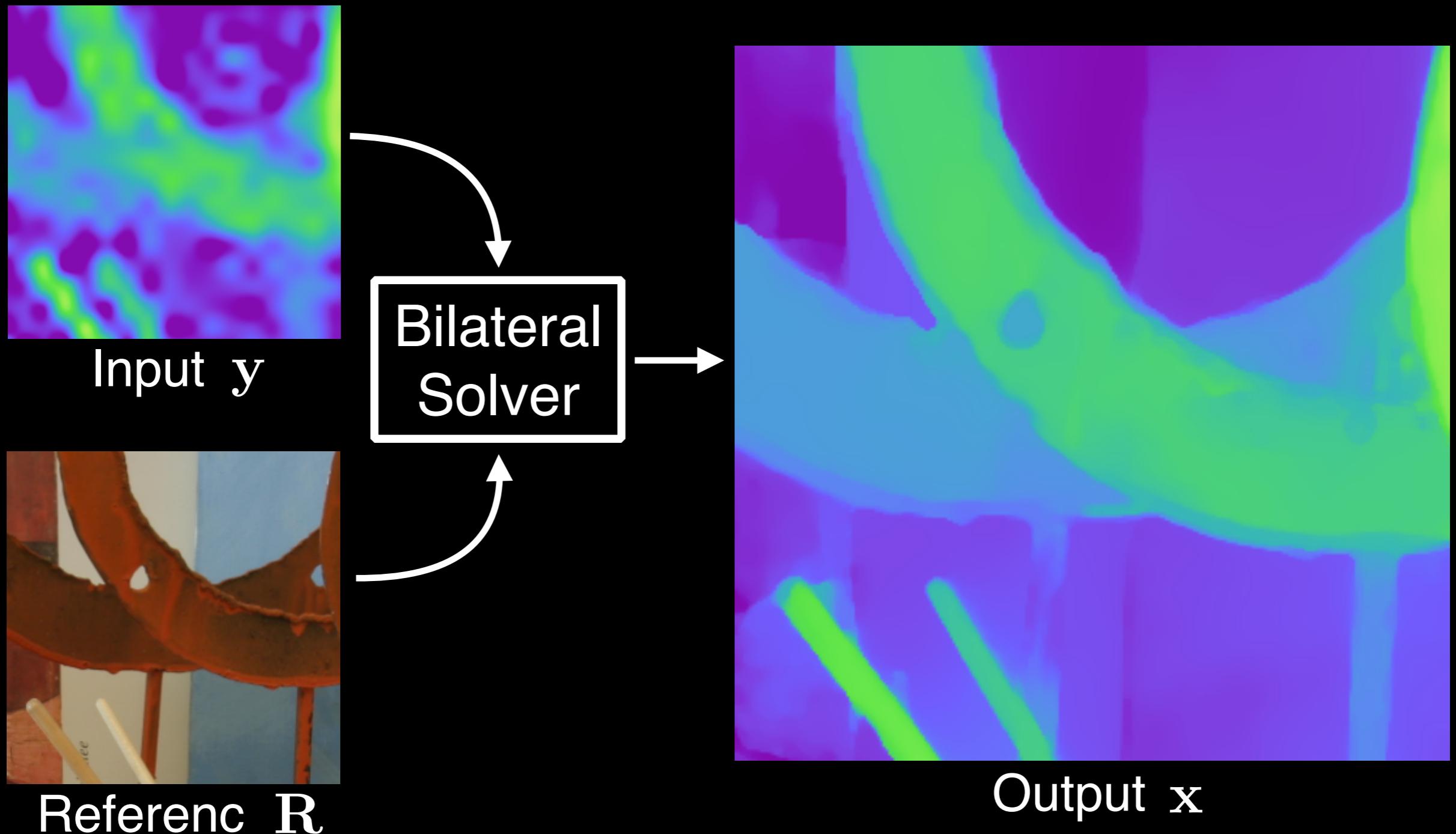
# Semantic segmentation



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

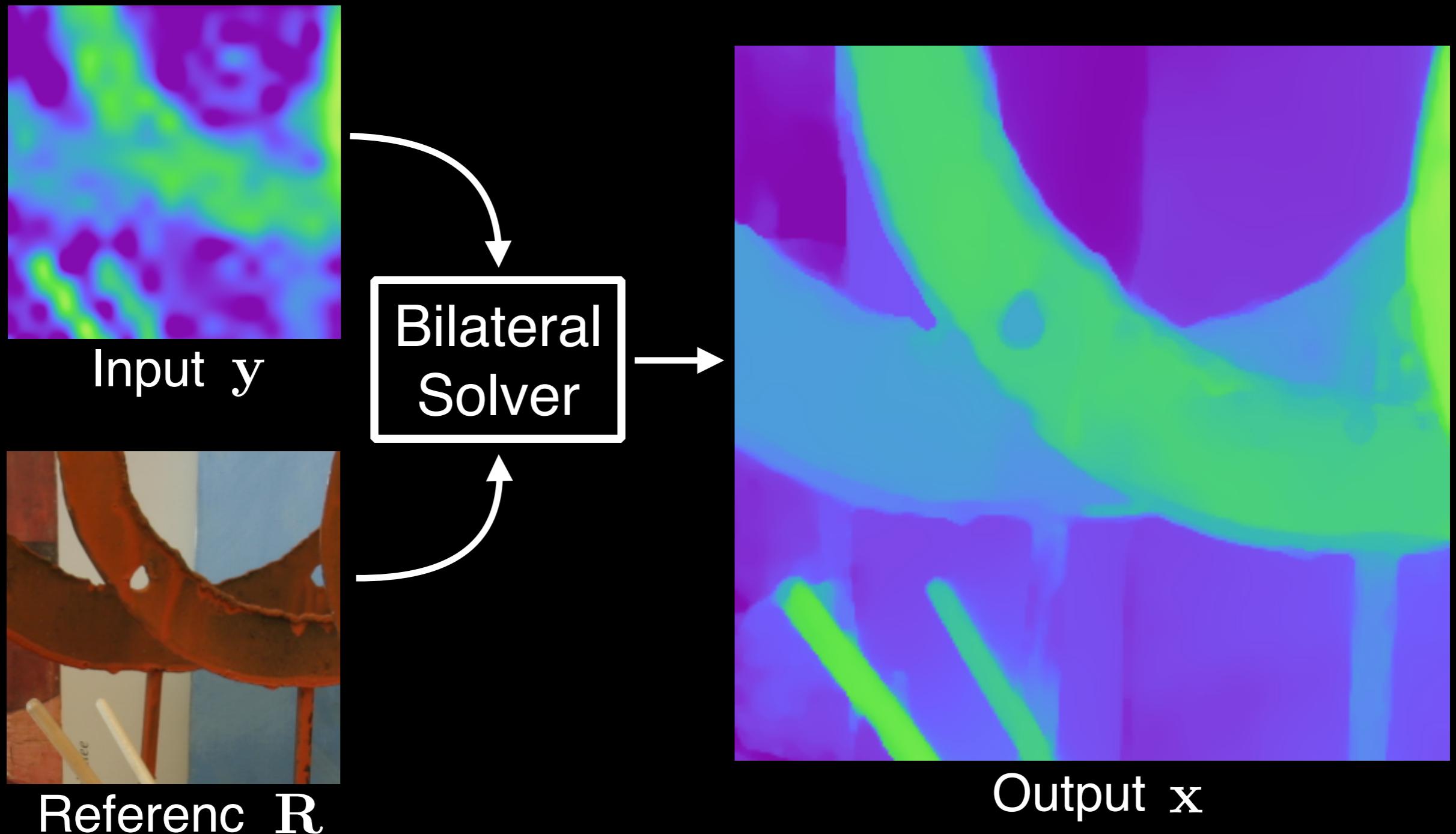
$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma_d^2}\right)$$

# Bilateral Solver



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



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

# Non-local means

Bilateral filter: Average neighbours with similar intensities

Non-local means: Average neighbours with similar neighbourhoods!

for every pixel  $p$



# Non-local means

Bilateral filter: Average neighbours with similar intensities

Non-local means: Average neighbours with similar neighbourhoods!

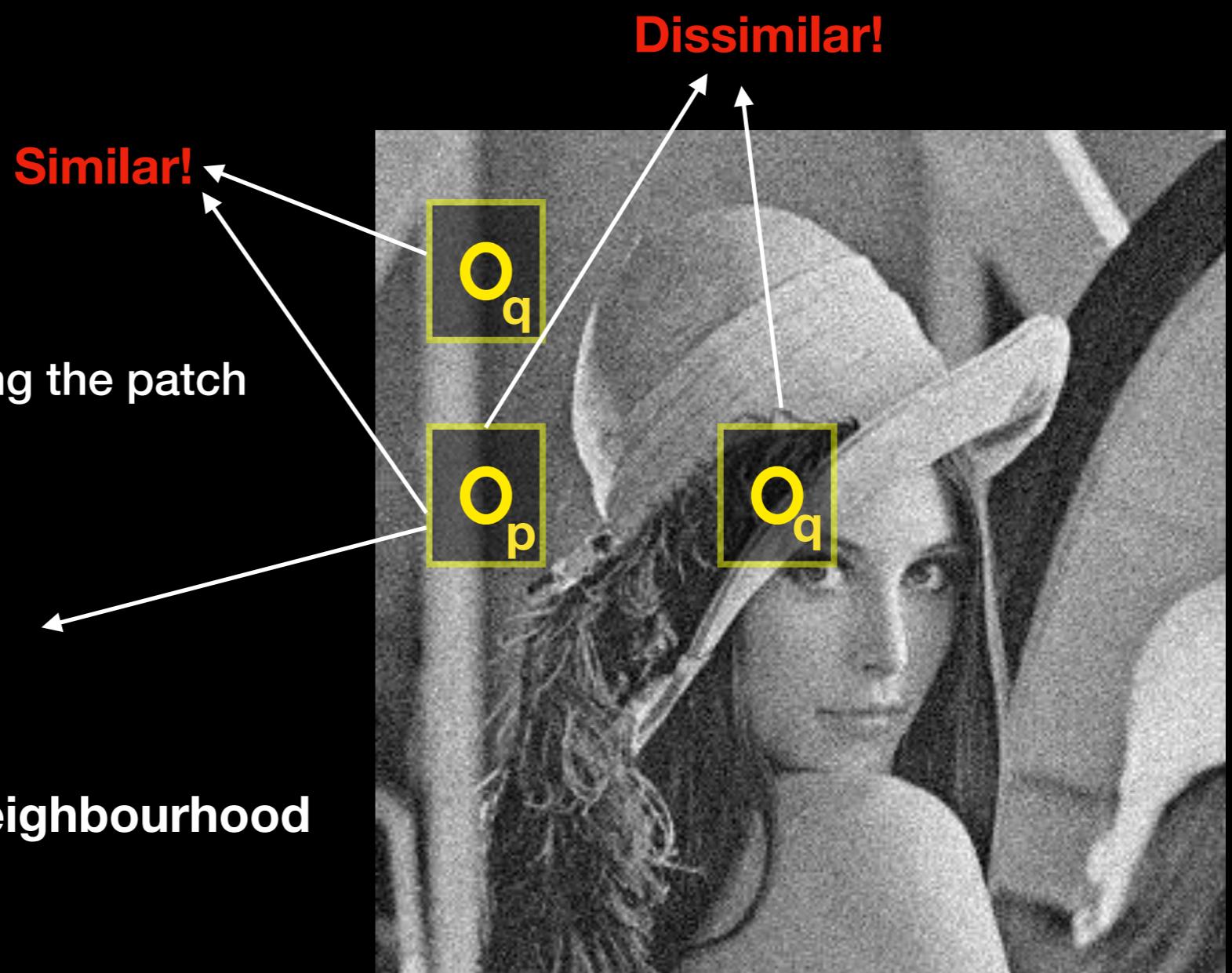
Filter with this!

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

$v_p, v_q$  are vectors describing the patch

for every pixel p

- define a small neighbourhood



# Non-local means

Filter with this!

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

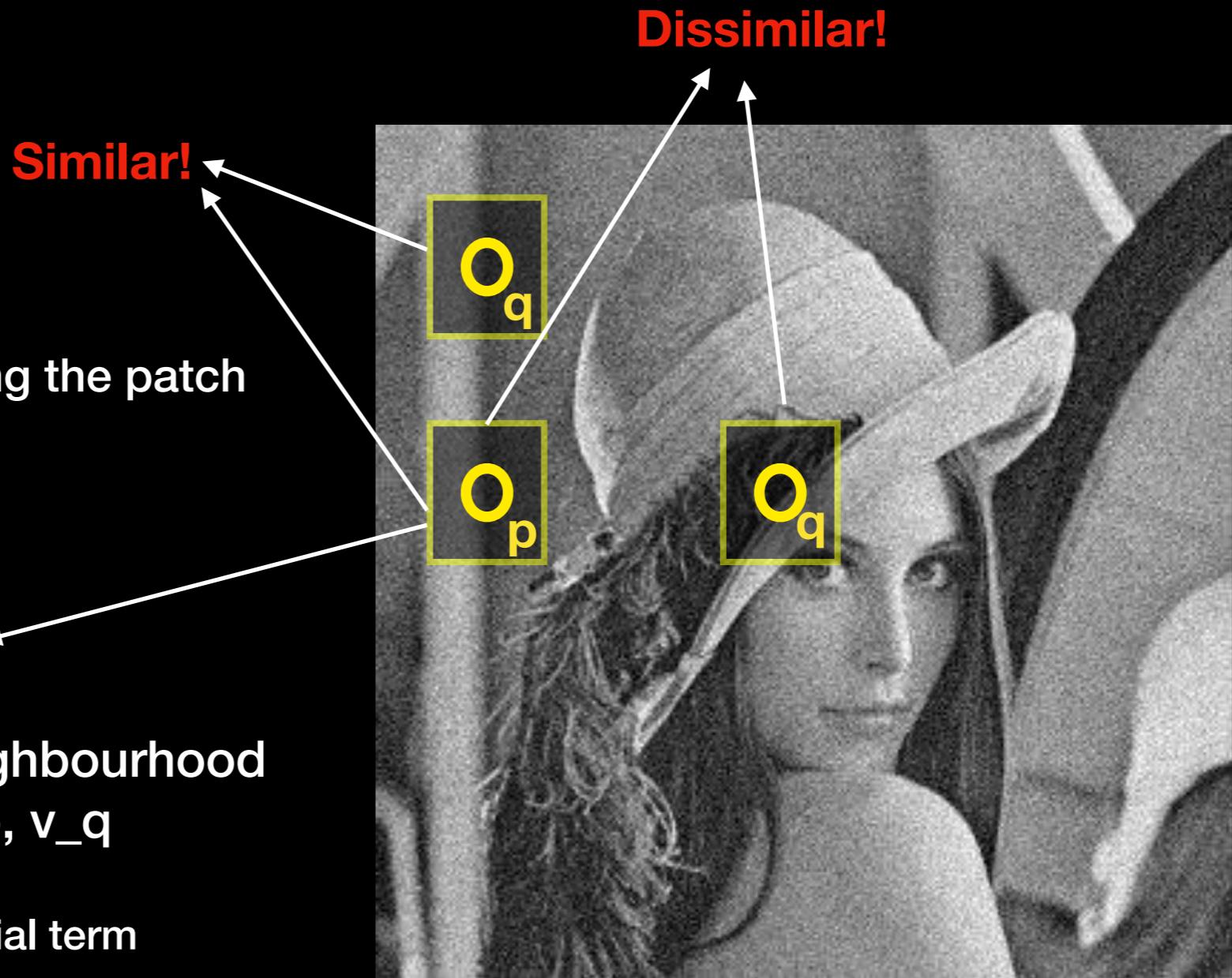
$v_p, v_q$  are vectors describing the patch

for every pixel p

- define a small neighbourhood
- define vectors  $v_p, v_q$

No spatial term

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



# Non-local means

Filter with this!

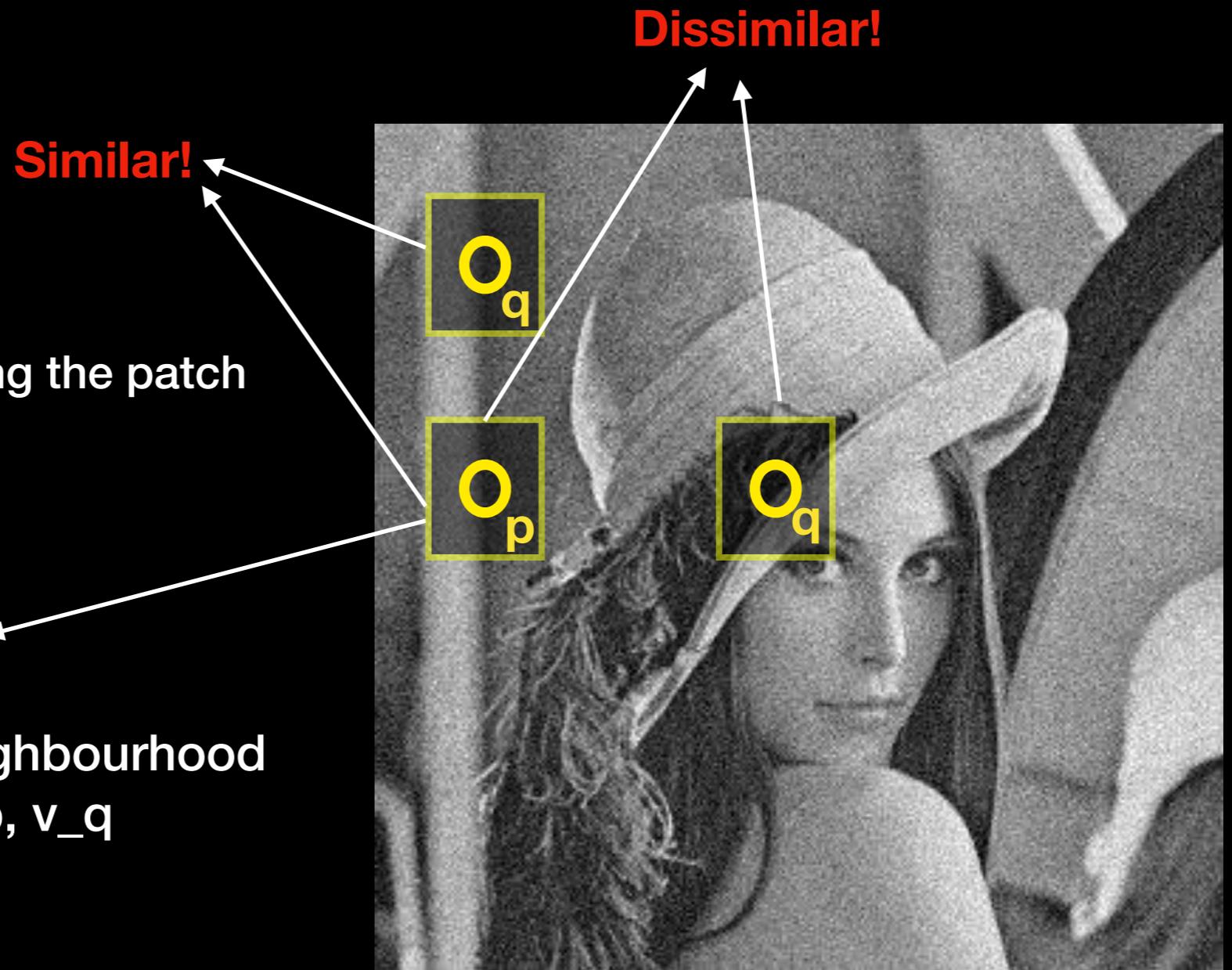
$$\|v_p - v_q\|^2$$

$v_p, v_q$  are vectors describing the patch

for every pixel p

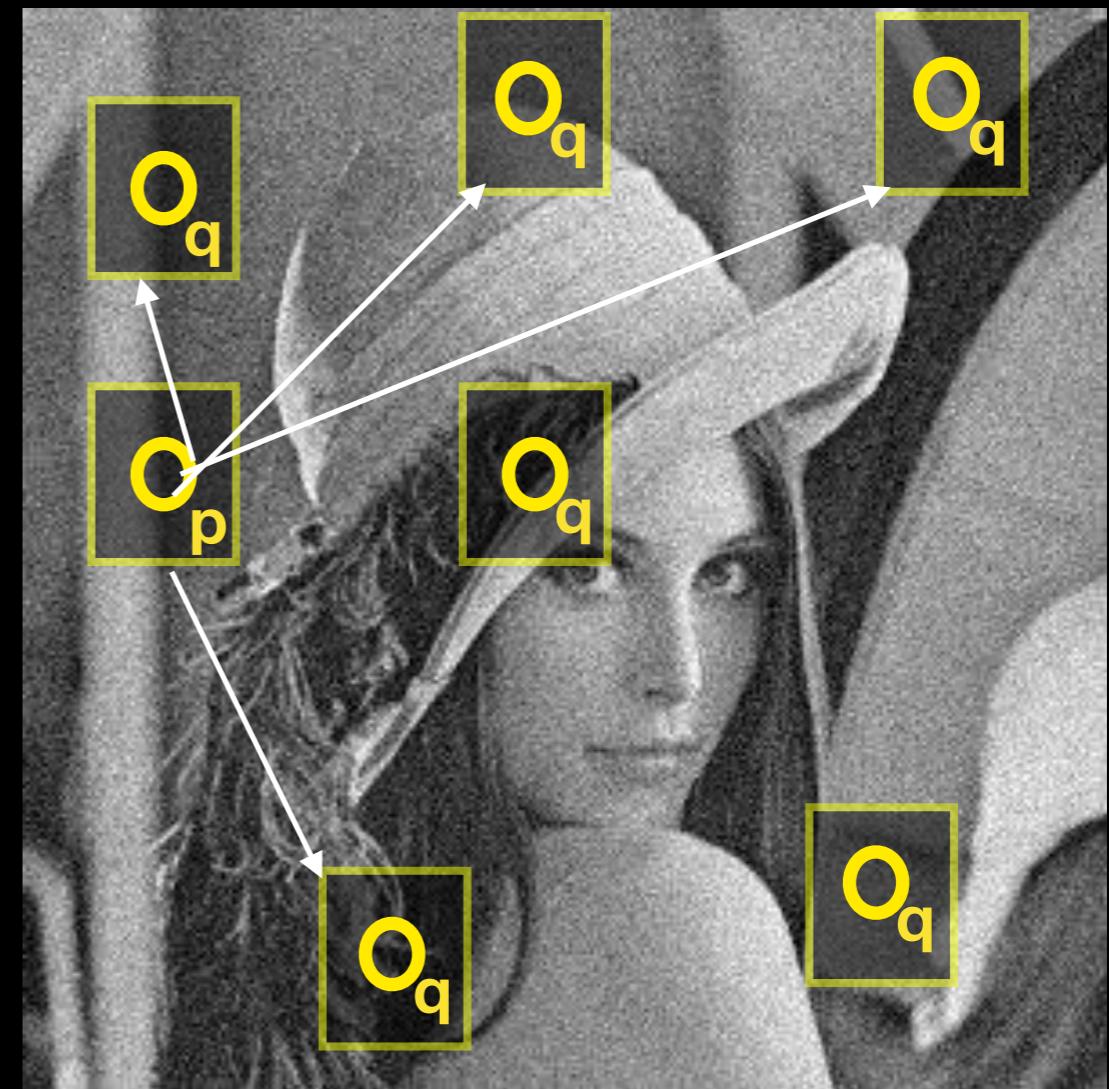
- define a small neighbourhood
- define vectors  $v_p, v_q$

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



# 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!

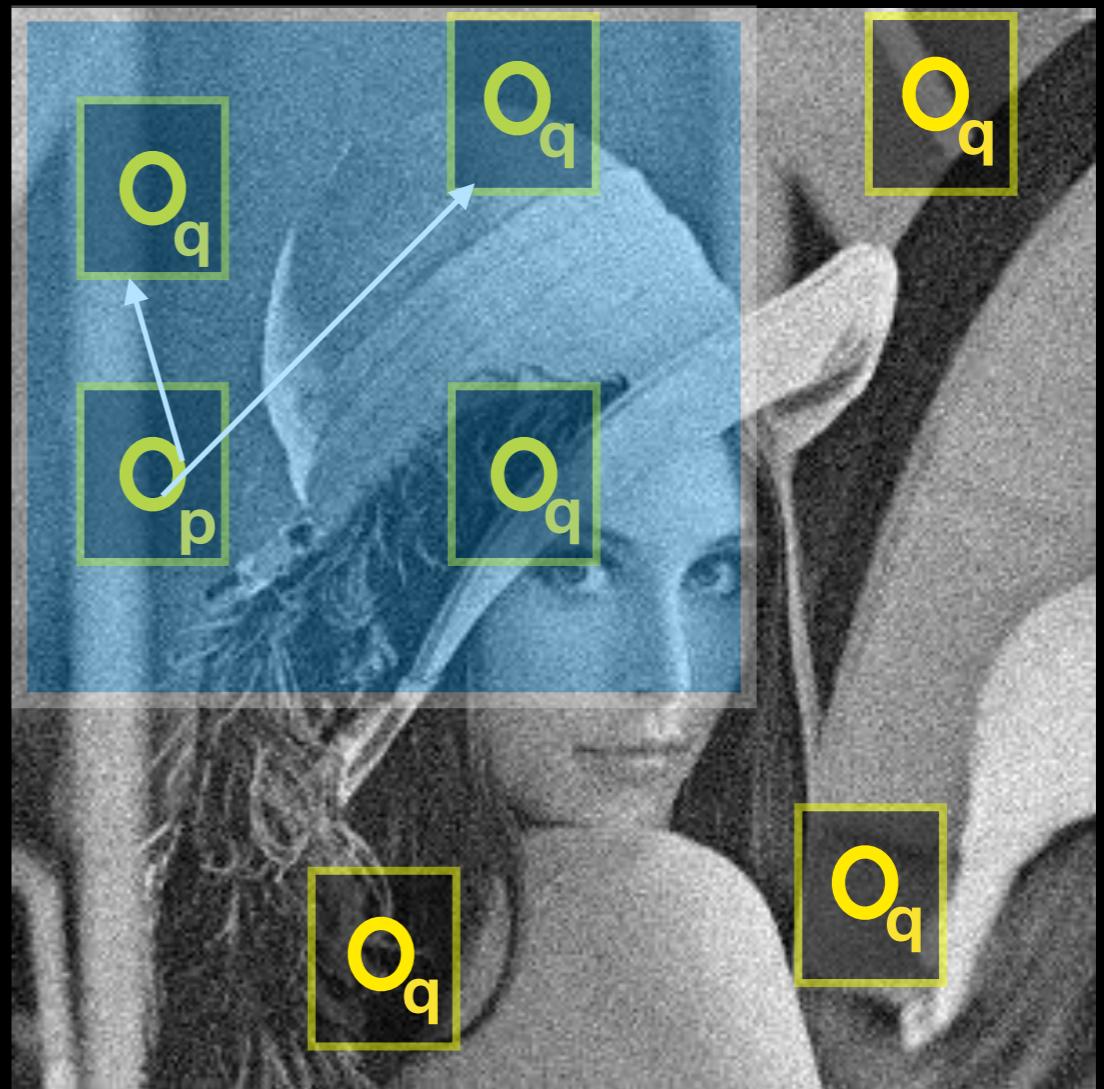
# Non-local means: complexity?

PyNotebook

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

Problem: Too many patches to search!

Simple remedy: confine the search to an extended neighbourhood.



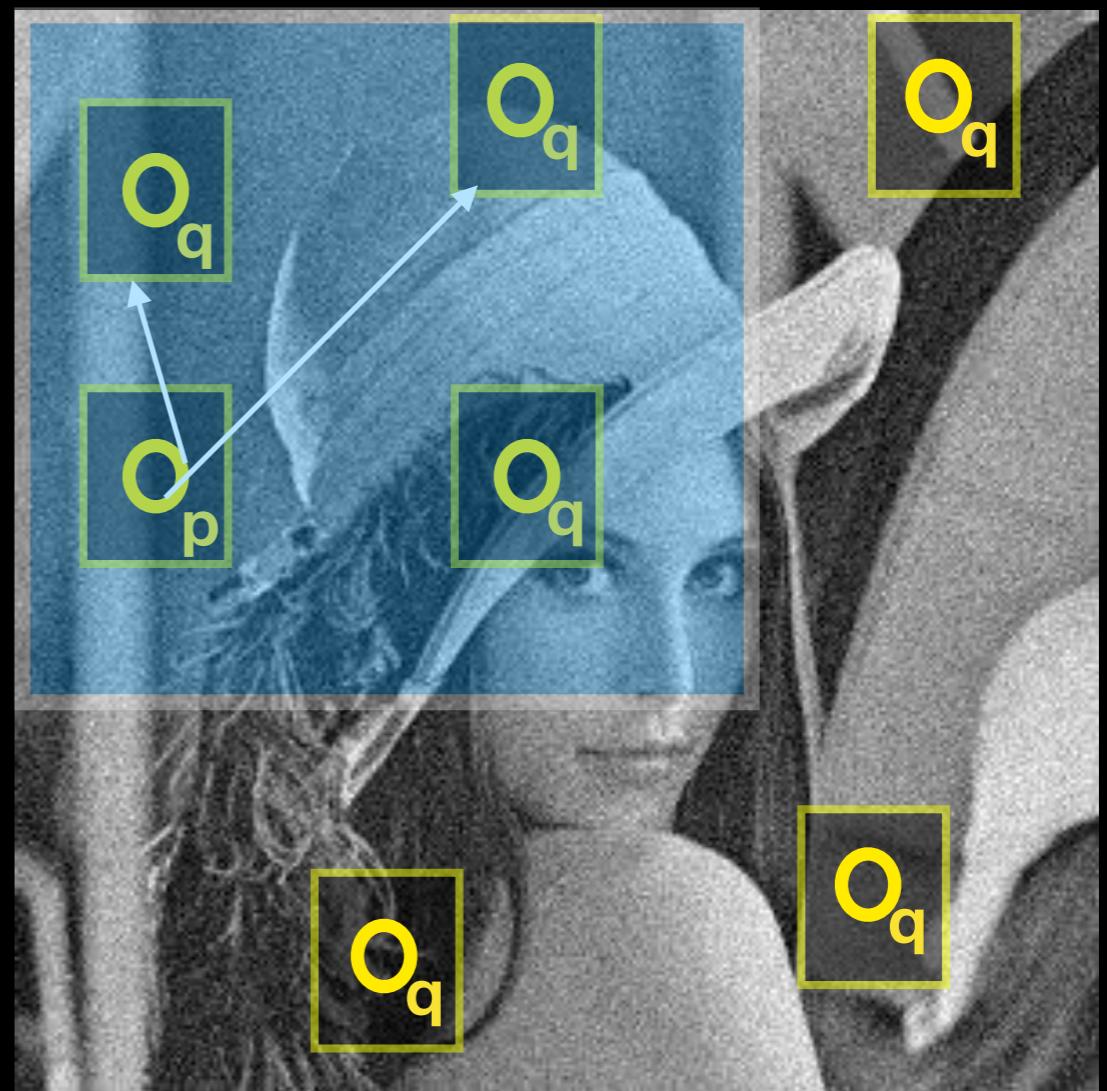
# Non-local means: complexity?

PyNotebook

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

Problem: Too many patches to search!

Simple remedy: confine the search to an extended neighbourhood.



# Non-local means: faster?

PyNotebook

- Get all the patches from the image
- Compute their PCA — know which shapes are more descriptive than others.
- Since low-importance components are mostly noise — we can discard them. We reduce the patch size from  $W^2$  to smaller number.
- Index all the reduced patch-vectors in a tree and query over it.

Office hours?  
Anytime is OK other than Tue 14:30-16:30