

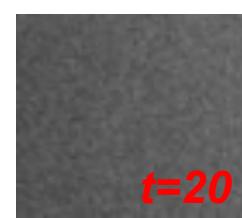
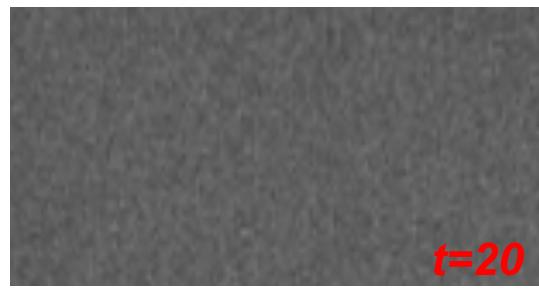
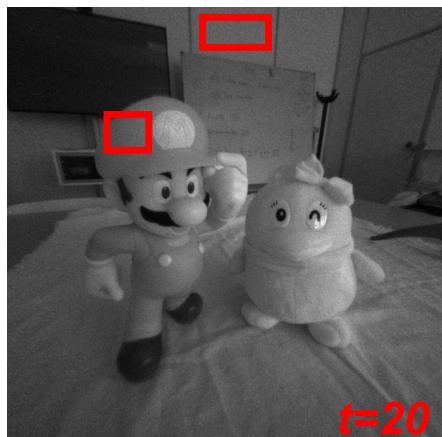
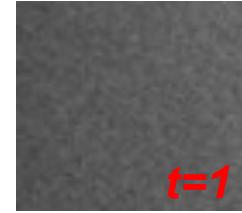
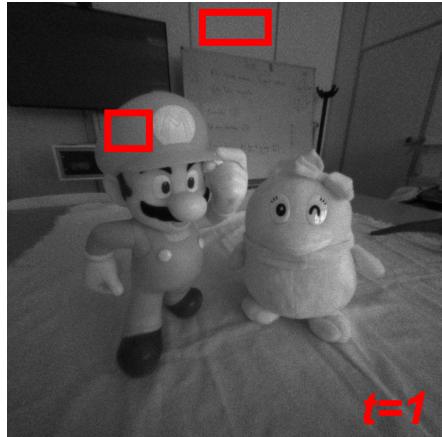
Image Filtering

Prof. Giuseppe Lisanti

giuseppe.lisanti@unibo.it

Visualizing and Understanding Noise

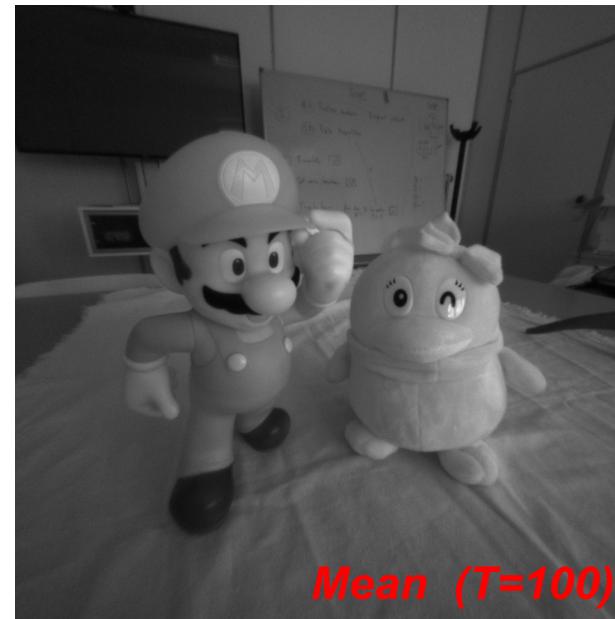
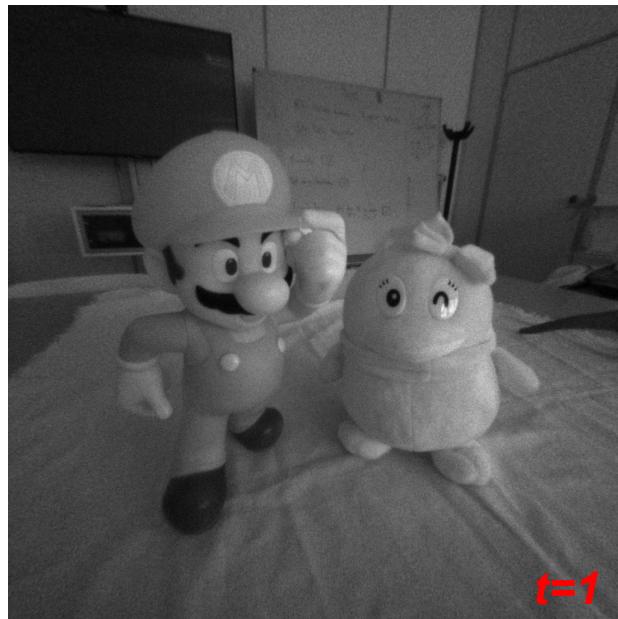
t_k



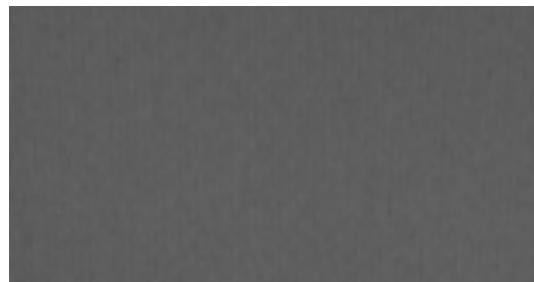
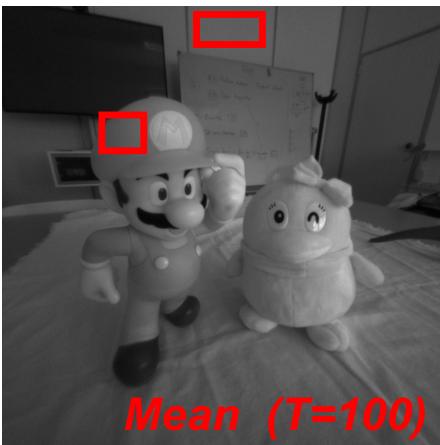
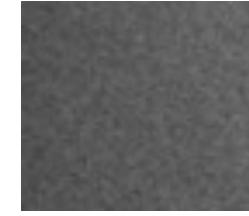
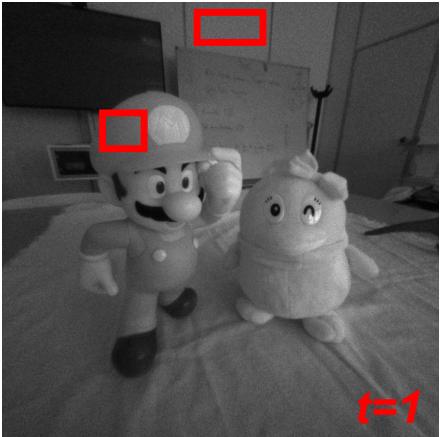
$$I_t(p) = \tilde{I}(p) + n_t(p) \quad \text{with } n_t(p) \text{ i.i.d. and } n_t(p) \sim N(0, \sigma)$$

Denoising by taking a mean across time

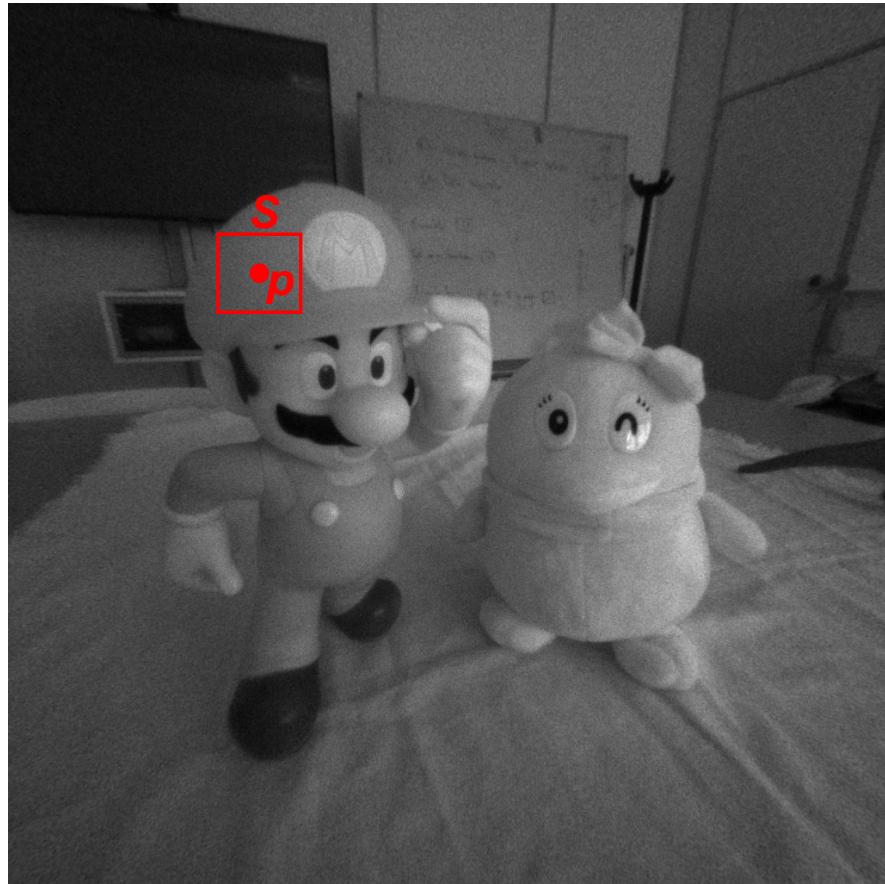
$$O(p) = \frac{1}{T} \sum_{t=1}^T I_k(p) = \frac{1}{T} \sum_{t=1}^T (\tilde{I}(p) + n_t(p)) = \frac{1}{T} \sum_{t=1}^T \tilde{I}(p) + \frac{1}{T} \sum_{t=1}^T n_t(p) \underset{\text{Red circles}}{\approx} \tilde{I}(p)$$



The mean image is far less noisy



What if we are given a single image ?



- We may compute a mean across neighbouring pixels, i.e. a spatial rather than temporal mean

$$o(p) = \frac{1}{|S|} \sum_{q \in S} I(q) = \frac{1}{|S|} \sum_{q \in S} (\tilde{I}(q) + n(q)) =$$

$$\frac{1}{|S|} \sum_{q \in S} \tilde{I}(q) + \frac{1}{|S|} \sum_{q \in S} n(q) \cong \tilde{I}(q)$$

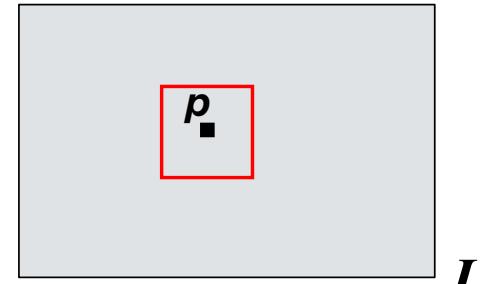


Denoising Filters

- How large should K be?
 - It is a trade-off!

Image Filters

- **Image Filters** are image processing operators that compute the new intensity (colour) of a pixel, p , based on the intensities (colours) of those belonging to a neighbourhood (aka *support*) of p .
- They accomplish a variety of useful image processing functions, such as e.g. *denoising* and *sharpening* (edge enhancement).
- An important sub-class of filters is given by *Linear* and *Translation-Equivariant (LTE)* operators
- *Signal theory*: their application in image processing consist in a *2D convolution* between the input image and the *impulse response function* (point spread function or kernel) of the LTE operator.
- LTE operators are used as *feature extractors* in CNNs (Convolutional Neural Networks).



I

$$o(p) = f(S(p))$$

LTE Operators and Convolution

- Given an input 2D signal $i(x, y)$, a 2D operator, $T\{\cdot\}$: $o(x, y) = T\{i(x, y)\}$, is said to be **linear iff**:

$$T\{\alpha i_1(x, y) + \beta i_2(x, y)\} = \alpha o_1(x, y) + \beta o_2(x, y) \quad \text{with} \quad o_1 = T\{i_1\} \quad \text{and} \quad o_2 = T\{i_2\}$$

and α, β are two constants.

- The operator is said to be **translation-equivariant iff**:

$$T\{i(x - x_0, y - y_0)\} = o(x - x_0, y - y_0)$$

LTE Operators and Convolution

- Given an input 2D signal $i(x, y)$, a 2D operator, $T\{\cdot\}: o(x, y) = T\{i(x, y)\}$, is said to be **linear iff**:

$$T\{\alpha i_1(x, y) + \beta i_2(x, y)\} = \alpha o_1(x, y) + \beta o_2(x, y) \quad \text{with} \quad o_1 = T\{i_1\} \quad \text{and} \quad o_2 = T\{i_2\}$$

and α, β are two constants.

- The operator is said to be **translation-equivariant iff**:

$$T\{i(x - x_0, y - y_0)\} = o(x - x_0, y - y_0)$$

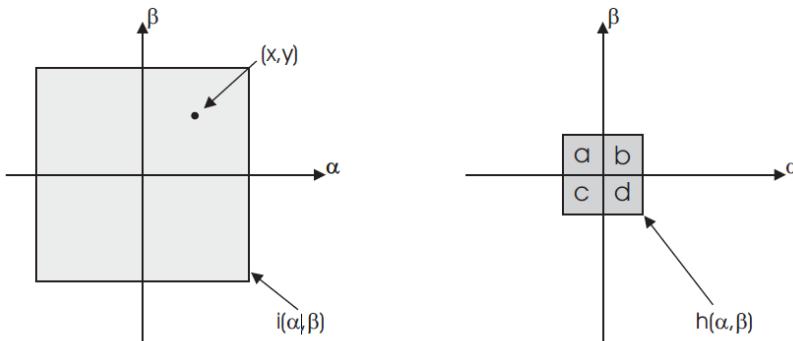
- If the operator is LTE, the output signal is given by the **convolution** between the input signal and the *impulse response (point spread function)*, $h(x, y) = T\{\delta(x, y)\}$, of the operator:

Unit impulse (Dirac delta function)

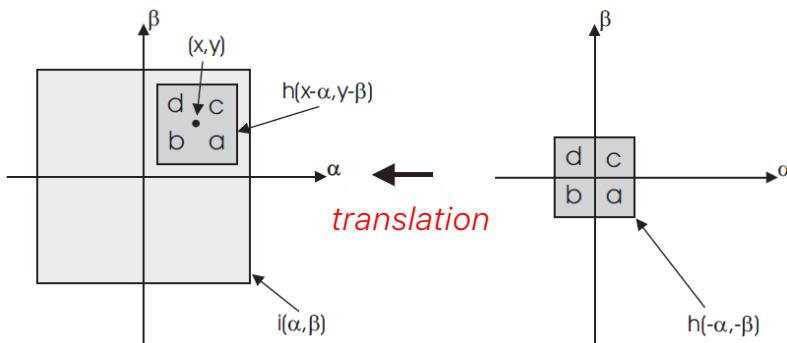
$$o(x, y) = T\{i(x, y)\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta) h(x - \alpha, y - \beta) d\alpha d\beta$$

A Graphical View of Convolution

$$o(x, y) = T\{i(x, y)\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta) h(x - \alpha, y - \beta) d\alpha d\beta$$



reflection about the origin



Properties of Convolution

- The convolution operation is often denoted using the symbol "*", e.g.

$$o(x, y) = i(x, y) * h(x, y)$$

- Some useful properties of convolution are as follows:

- Associative Property

$$f * (g * h) = (f * g) * h$$

- Commutative Property

$$f * g = g * f$$

- Distributive Property wrt the Sum

$$f * (g + h) = f * g + f * h$$

- Convolution Commutes with Differentiation

$$(f * g)' = f' * g = f * g'$$

Correlation

- The correlation of signal $i(x,y)$ wrt signal $h(x,y)$ is defined as:

$$i(x,y) \circ h(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta) h(x + \alpha, y + \beta) d\alpha d\beta$$

- Accordingly, the correlation of $h(x,y)$ wrt $i(x,y)$ is given by:

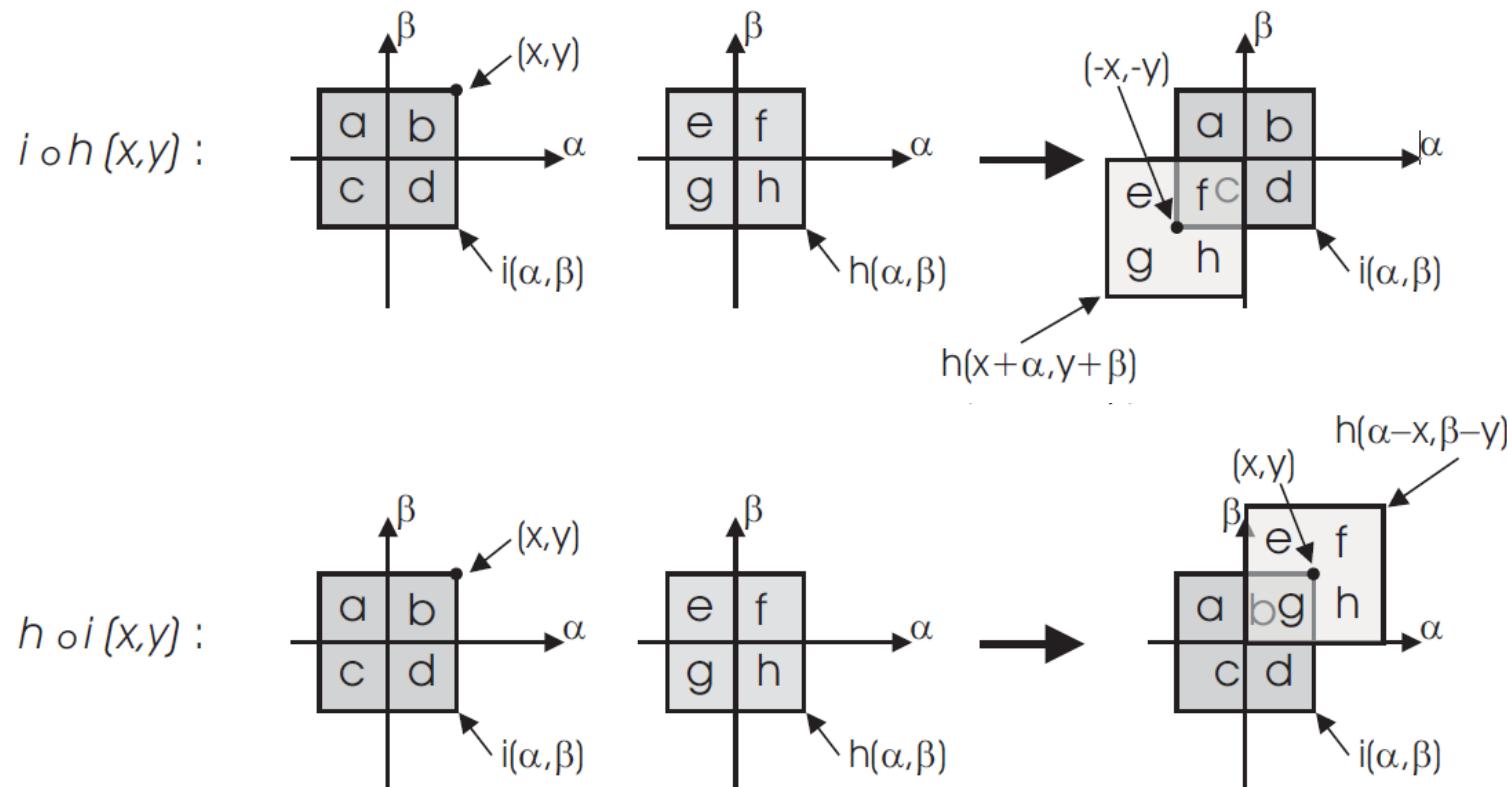
$$h(x,y) \circ i(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(\alpha, \beta) i(x + \alpha, y + \beta) d\alpha d\beta$$

- Unlike convolution, correlation is **not commutative**:

$$\begin{aligned} h(x,y) \circ i(x,y) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(\alpha, \beta) i(x + \alpha, y + \beta) d\alpha d\beta \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\xi, \eta) h(\xi - x, \eta - y) d\xi d\eta \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta) h(\alpha - x, \beta - y) d\alpha d\beta \\ &\neq i(x,y) \circ h(x,y) \end{aligned}$$

$$\begin{aligned} \xi &= x + \alpha \Rightarrow \alpha = \xi - x \\ \eta &= y + \beta \Rightarrow \beta = \eta - y \end{aligned}$$

A Graphical View of Correlation



Convolution and Correlation

- The correlation of h wrt i is similar to convolution: the product of the two signals is integrated after translating h without reflection.
 - Hence, if h is an even function ($h(x, y) = h(-x, -y)$), the convolution between i and h , ($i * h = h * i$), is the same as the correlation of h wrt i :

$$\begin{aligned} i(x, y) * h(x, y) &= h(x, y) * i(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta) h(x - \alpha, y - \beta) d\alpha d\beta \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta) h(\alpha - x, \beta - y) d\alpha d\beta \\ &= h(x, y) \circ i(x, y) \end{aligned}$$

we can swap

- It is worth observing that correlation is never commutative, even if h is an even function. To recap:

- Convolution is commutative:

$$i * h = h * i$$

- Correlation is not commutative:

$$i \circ h \neq h \circ i$$

- If h is an even function:

$$i * h = h * i = h \circ i$$

Discrete Convolution

$$o(x, y) = T\{i(x, y)\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta) h(x - \alpha, y - \beta) d\alpha d\beta$$

$$O(i, j) = T\{I(i, j)\} = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} I(m, n) H(i - m, j - n)$$

- where $I(i, j)$ and $O(i, j)$ are the discrete 2D input and output signals, respectively, and $H(i, j) = T\{\delta(i, j)\}$ is the kernel of the discrete LTE operator, i.e. the response to the 2D discrete unit impulse (Kronecker delta function), $\delta(i, j)$.
- As for continuous signals, discrete convolution consists in summing the product of the two signals where one has been reflected about the origin and translated.
- The previously highlighted four major convolution properties hold for discrete convolution too.

Practical Implementation

- In image processing both the input image and the kernel are stored into matrices of given finite sizes, with the image being much larger than the *kernel*. One would cycle through the kernel:

$$\begin{pmatrix} & \vdots \\ & \vdots \\ I(i-k, j-k) & & I(i-k, j+k) \\ \vdots & & \vdots \\ \dots & I(i, j) & \dots \\ I(i+k, j-k) & I(i+k, j) & I(i+k, j+k) \\ \vdots & & \vdots \\ & \vdots \end{pmatrix} * \begin{pmatrix} K(-k, -k) & \dots & K(-k, 0) & \dots & K(-k, k) \\ \vdots & \dots & \vdots & \dots & \vdots \\ K(0, -k) & \dots & K(0, 0) & \dots & K(0, k) \\ \vdots & \dots & \vdots & \dots & \vdots \\ K(k, -k) & \dots & K(k, 0) & \dots & K(k, k) \\ \vdots & \dots & \vdots & \dots & \vdots \end{pmatrix}$$

$(2k+1) \times (2k+1)$

$$O(i, j) = \sum_{m=-k}^k \sum_{n=-k}^k K(m, n)I(i - m, j - n)$$

Conceptually, to obtain the output image we need to slide the kernel across the whole input image and compute the convolution at each pixel **(do not overwrite the input matrix!)**



Practical Implementation

- In image processing both the input image and the kernel are stored into matrixes of given finite sizes, with the image being much larger than the *kernel*. One would cycle through the kernel:

$$\begin{pmatrix} \vdots & \\ \vdots & \\ \vdots & \\ \dots & I(i,j) & \dots \\ \vdots & \\ \vdots & \\ \vdots & \end{pmatrix} * \begin{pmatrix} K(-k, -k) & \dots & K(-k, 0) & \dots & K(-k, k) & \\ \vdots & & \vdots & & \vdots & \\ K(0, -k) & \dots & K(0, 0) & \dots & K(0, k) & \\ \vdots & & \vdots & & \vdots & \\ K(k, -k) & \dots & K(k, 0) & \dots & K(k, k) & \\ \vdots & & & & & \vdots \end{pmatrix}$$

$(2k+1) \times (2k+1)$

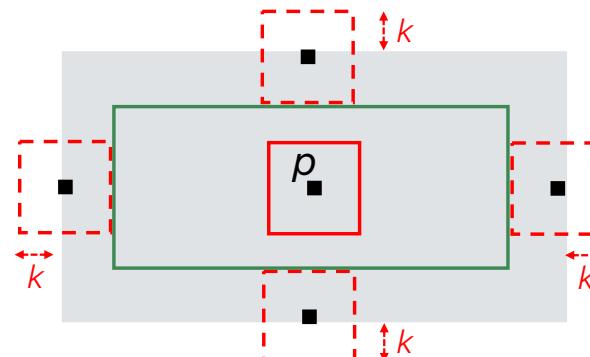
$$O(i,j) = \sum_{m=-k}^k \sum_{n=-k}^k K(m,n)I(i-m, j-n)$$

Conceptually, to obtain the output image we need to slide the kernel across the whole input image and compute the convolution at each pixel **(do not overwrite the input matrix!)**

- Border Issue, two main options:

- CROP (common in image processing)
- PAD (preferred in CNNs)

- How to pad: zero-padding, replicate (aa|a....d|dd), reflect (cba|abc....dfg|gfd), reflect_101 (dcb|abcd.....efgh|gfe),



Mean Filter

- Mean filtering is the simplest (and fastest) way to denoise an image. It consists in replacing each pixel intensity by the average intensity over a chosen neighbourhood (e.g. 3x3, 5x5, 7x7...).
- The Mean Filter is an LTE operator as it can be expressed as a convolution with a kernel.

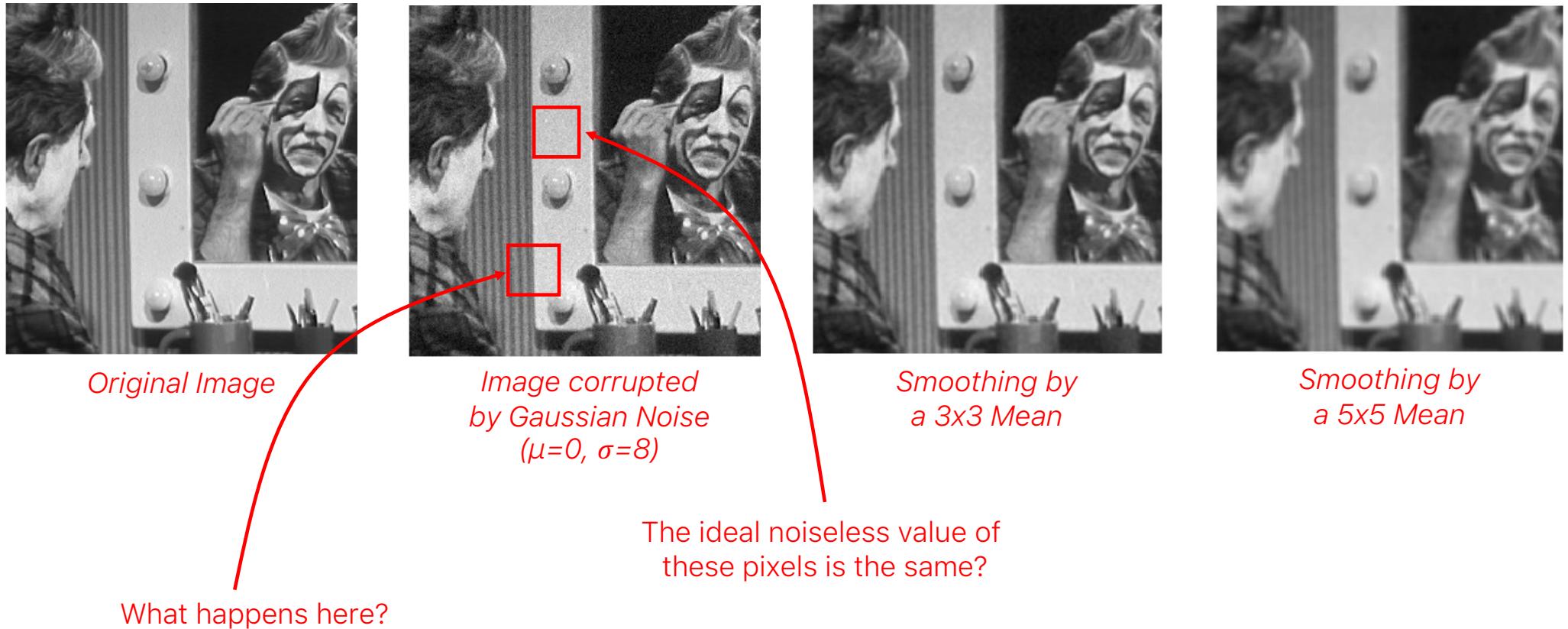
Below, the kernels for a 3x3 and 5x5 mean filter:

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

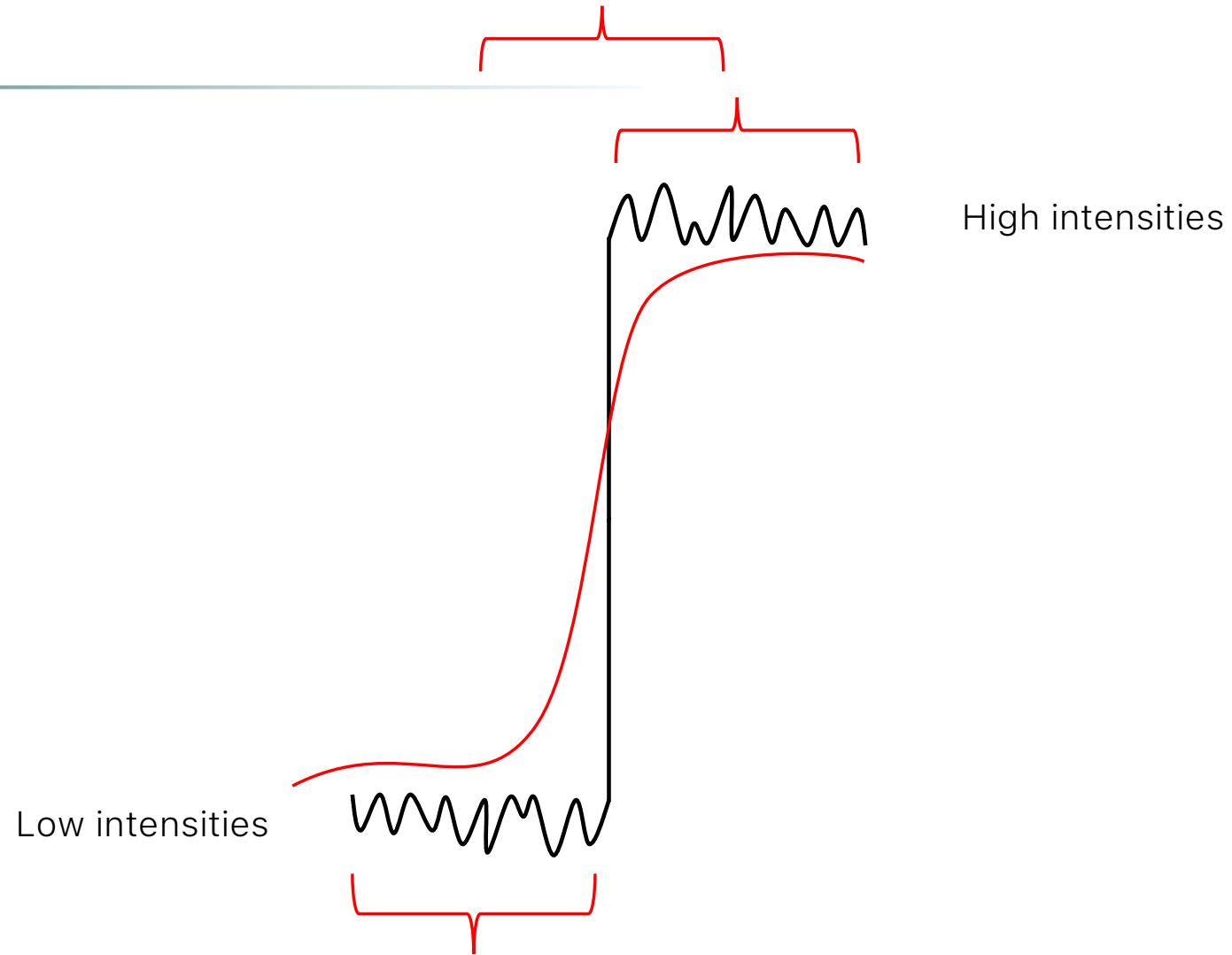
$$\begin{bmatrix} 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \end{bmatrix} = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- According to signal processing theory, the Mean Filter carries out a *low-pass filtering* operation, which in image processing is also referred to as *image smoothing*.
- Smoothing is often aimed at image denoising, though sometimes the purpose is to cancel out small-size unwanted details that might hinder the image analysis task.
- Mean filtering is inherently fast because multiplications are not needed.

Example: Gaussian Noise



Linear filtering does reduce noise but blurs the image

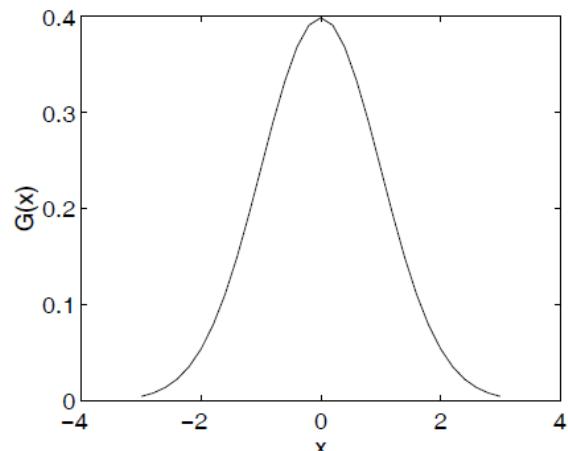


Gaussian Filter

- LTE operator whose impulse response is a 2D Gaussian function (with zero mean and constant diagonal covariance matrix).

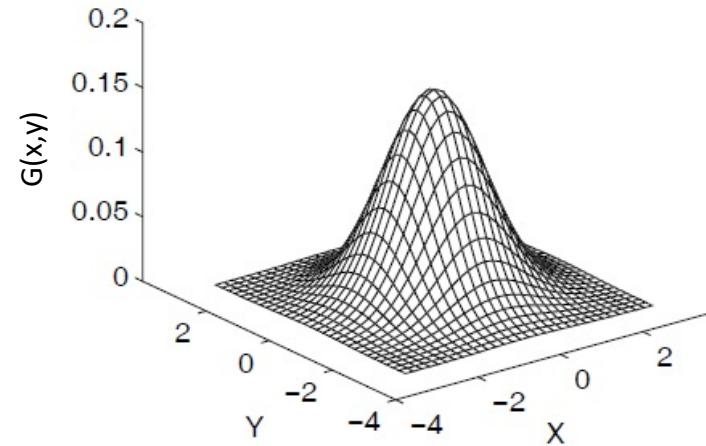
1D Gaussian

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$



2D Gaussian

$$G(x, y) = G(x)G(y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Circularly Symmetric

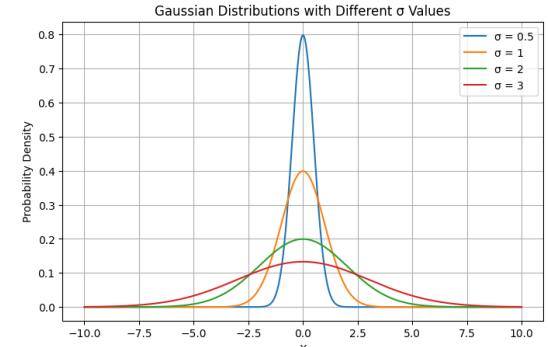
Practical Implementation

- The discrete Gaussian kernel can be obtained by sampling the corresponding continuous function, which is however of infinite extent. A finite size must therefore be properly chosen.
- To this purpose, we can observe that:
 - The larger is the size, the more accurate turns out the discrete approximation of the ideal continuous filter.
 - The computational cost grows with filter size.
 - The Gaussian gets smaller and smaller as we move away from the origin.
- We should use larger sizes for filters with high σ , smaller sizes whenever σ is smaller.
- Rule-of-thumb to choose the size of the filter given σ :
 - as the interval $[-3\sigma, +3\sigma]$ captures 99% of the area ("energy") of the Gaussian function, a typical rule-of-thumb dictates taking a $(2k+1) \times (2k+1)$ kernel with $k = [3\sigma]$

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

e.g.

	-3	-2	-1	0	1	2	3
k	0.0044	0.054	0.242	0.3989	0.242	0.054	0.0044



$$\begin{aligned} \sigma = 1 &\Rightarrow 7 \times 7 \\ \sigma = 1.5 &\Rightarrow 11 \times 11 \\ \sigma = 2 &\Rightarrow 13 \times 13 \\ \sigma = 3 &\Rightarrow 19 \times 19 \end{aligned}$$

Deploying Separability

- To further speed-up the filtering operation, one can deploy the separability property: due to the 2D Gaussian being the product of two 1D Gaussians, the original 2D convolution can be split into the chain of two 1D convolutions, i.e. either along x first and then along y, or viceversa.

$$I(x, y) * G(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\alpha, \beta) G(x - \alpha, y - \beta) d\alpha d\beta$$

$$G(x, y) = G(x)G(y)$$

$$I(x, y) * G(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\alpha, \beta) G(x - \alpha) G(y - \beta) d\alpha d\beta$$

$$I(x, y) * G(x, y) = \int_{-\infty}^{+\infty} G(y - \beta) \left(\int_{-\infty}^{+\infty} I(\alpha, \beta) G(x - \alpha) d\alpha \right) d\beta$$

$$I(x, y) * G(x, y) = (I(x, y) * G(x)) * G(y) = (I(x, y) * G(y)) * G(x)$$

Accordingly, the speed-up (S) brought in by the separability property can be expressed as:

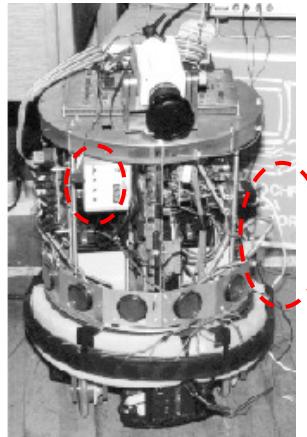
$$\begin{cases} \text{2D Filter: } N_{OPS} = 2 \cdot (2k + 1)^2 \\ \text{1D Filter: } N_{OPS} = 2 \cdot 2 \cdot (2k + 1) \end{cases}$$



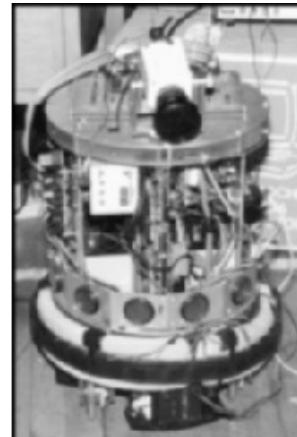
$$S = \frac{2 \cdot (2k + 1)^2}{2 \cdot 2 \cdot (2k + 1)} = k + \frac{1}{2}$$

Parameter σ sets the amount of smoothing

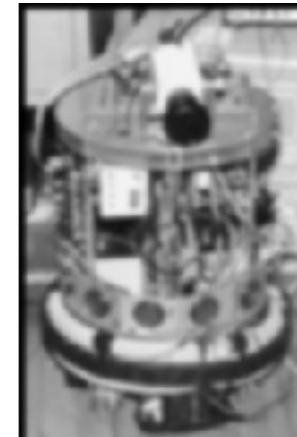
- The higher σ , the stronger the smoothing caused by the filter. This can be understood, e.g., by observing that as σ increases, the weights of closer points get smaller while those of farther points get larger.



Original Image



Smoothing by a Gaussian Filter with $\sigma = 1$



Smoothing by a Gaussian Filter with $\sigma = 2$



Smoothing by a Gaussian Filter with $\sigma = 4$

- As σ gets larger, small details disappear and the image content deals with larger size structures. Thus, filtering with a chosen σ can be thought of as setting the "scale" of interest to analyse image content.

Example: Impulse Noise



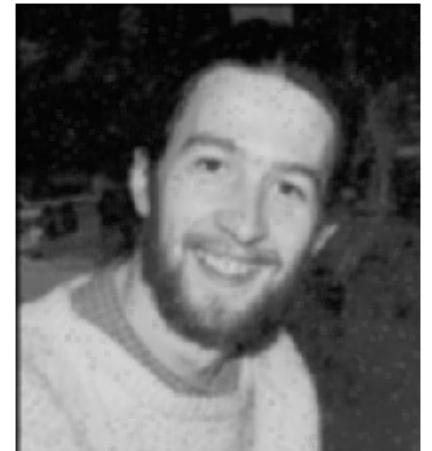
Original Image



*Image corrupted by Impulse Noise
(aka Salt-and-Pepper Noise)*



*Smoothing by
a 3x3 Mean*



*Smoothing by
a 5x5 Mean*

Linear filtering is ineffective toward impulse noise (and blurs the image)

Median Filter

- **Non-linear** filter whereby each pixel intensity is replaced by the median over a given neighbourhood, the median being the value falling half-way in the sorted set of intensities.

$$\text{median } [A(x) + B(x)] \neq \text{median } [A(x)] + \text{median } [B(x)]$$

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:
115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

- Median filtering counteracts impulse noise effectively, as outliers (i.e. noisy pixels) tend to fall at either the top or bottom end of the sorted intensities.
- Median filtering tends to keep sharper edges than linear filters such as the Mean or Gaussian:

$$\begin{aligned} \dots & 10 \ 10 \ 40 \ 40 \ \dots \Rightarrow \dots & 10 \ 20 \ 30 \ 40 \ \dots & (\text{Mean}) \\ & \Rightarrow \dots & 10 \ 10 \ 40 \ 40 \ \dots & (\text{Median}) \end{aligned}$$

Example



Original Image



Image Corrupted by impulse noise (+100, 5% of randomly picked pixels)



Filtering by a 3x3 Median

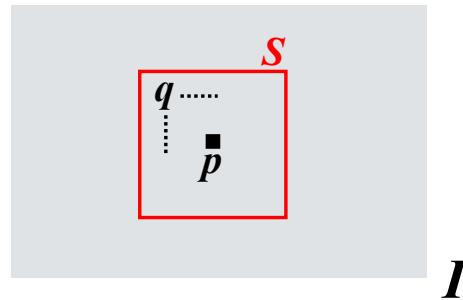


Filtering twice by a 3x3 Median

- When dealing with impulse noise, the Median Filter can effectively denoise the image without introducing significant blur.
- Yet, **Gaussian-like noise**, such as sensor noise, cannot be dealt with by the Median, as this would require computing new noiseless intensities.
 - Purposely, the Median may be followed by linear filtering.

Bilateral Filter

- Advanced **non-linear** filter to accomplish denoising of Gaussian-like noise without blurring the image (aka edge preserving smoothing).



$$O(p) = \sum_{q \in S} H(p, q) \cdot I_q$$
$$H(p, q) = \frac{1}{W(p)} G_{\sigma_s}(d_s(p, q)) G_{\sigma_r}(d_r(p, q))$$

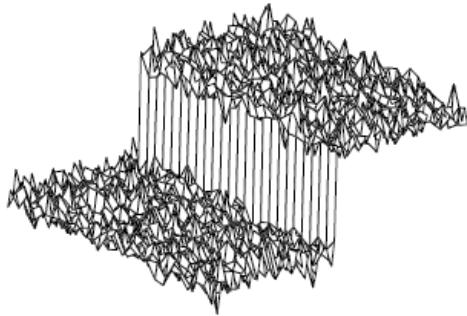
$$d_s(p, q) = \|p - q\|_2 = \sqrt{(u_p - u_q)^2 + (v_p - v_q)^2} \quad \longrightarrow \text{Spatial Distance}$$

$$d_r(I_p, I_q) = |I_p - I_q| \quad \longrightarrow \text{Range (Intensity) Distance}$$

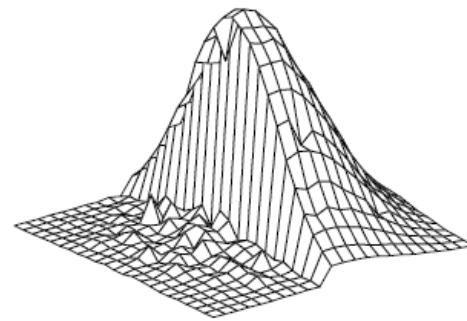
$$W(p) = \sum_{q \in S} G_{\sigma_s}(d_s(p, q)) G_{\sigma_r}(d_r(p, q)) \quad \longrightarrow \text{Normalization Factor (Unity Gain)}$$

Bilateral Filter

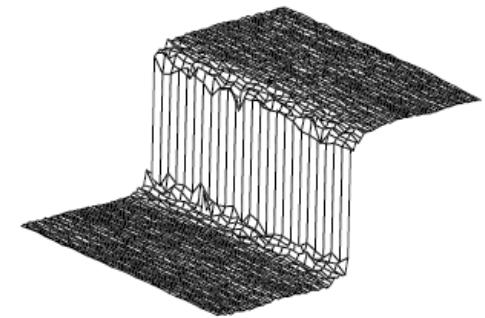
Step-edge as wide
as 100 gray-levels



$H(p, q)$ at a pixel just across
the edge in the brighter region



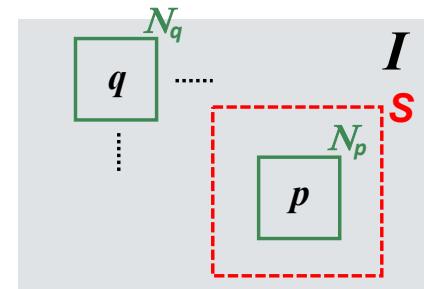
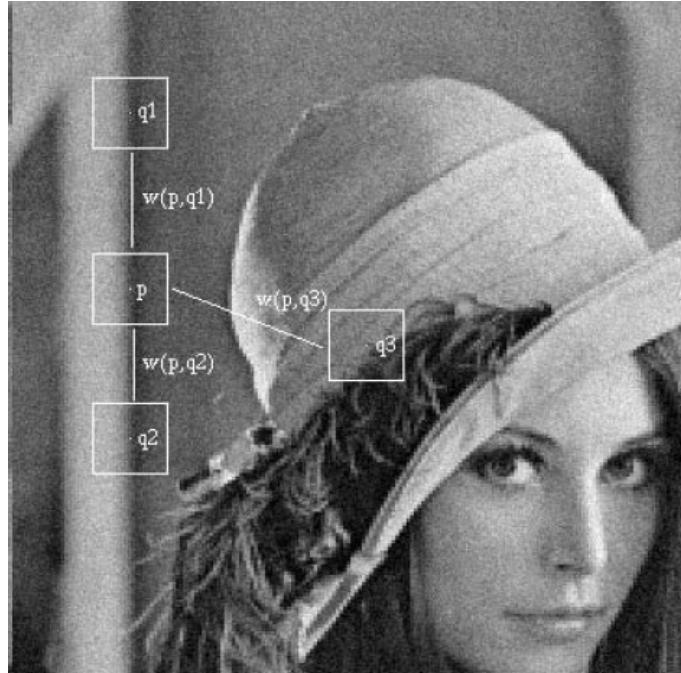
Output provided
by the filter $\sigma_s = 5, \sigma_r = 50$



- Given the supporting neighbourhood, neighbouring pixels take a larger weight as they are both closer and more similar to the central pixel.
- At a pixel nearby an edge, the neighbours falling on the other side of the edge look quite different and thus cannot contribute significantly to the output value due to their weights being small.

Non-local Means Filter

- Another well-known *non-linear* edge preserving smoothing filter. The key idea is that the similarity among patches spread over the image can be deployed to achieve denoising.



$$O(p) = \sum_{q \in S} w(p, q) I(q)$$

$$w(p, q) = \frac{1}{Z(p)} e^{-\frac{\|N_p - N_q\|_2^2}{h^2}}$$

$$Z(p) = \sum_{q \in I} e^{-\frac{\|N_p - N_q\|_2^2}{h^2}}$$

Non-local Means Filter



*Image Corrupted by
Gaussian Noise ($\sigma = 20$)*



Gaussian Filter



Non-local Means Filter
 $N = 7 \times 7$,
 $S = 21 \times 21$,
 $h = 10 \cdot \sigma$

References

- V. S. Nalwa, "A Guided Tour of Computer Vision", Addison-Wesley Publishing Company, 1993.
- R. Fisher, S. Perkins, A. Walker, E. Wolfart, "Hypermedia Image Processing Reference", Wiley, 1996 (<http://homepages.inf.ed.ac.uk/rbf/HIPR2/>)
- R. Schalkoff, "Digital Image Processing And Computer Vision, Wiley, 1989.
- C. Tomasi, R. Manduchi "Bilateral Filtering for Gray and Color Images", ICCV 1998.
- S.Paris, P. Kornprobst, J. Tumblin, F. Durand "A Gentle Introduction to Bilateral Filtering and its Applications" (SIGGRAPH 2008,CVPR 2008, SIGGRAPH 2007)
http://people.csail.mit.edu/sparis/siggraph07_course/
- A. Buades, B. Coll, J.M. Morel, "A non-local algorithm for image denoising", CVPR 2005.