Lecture 5 Group Operators

COMP6223 Computer Vision (MSc)

How do we combine points to make a new point in a new image?

Book
pp
84- 114

Department of Electronics and Computer Science

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Content

1. How can we collect points as a group?
2. How can we apply processes to that group?

3×3 template and its inverse

Original template

w_{00}	w_{01}	w_{02}
w_{10}	w_{11}	w_{12}
w_{20}	w_{21}	w_{22}

Its inverse

w_{22}	w_{21}	w_{20}
w_{12}	w_{11}	w_{10}
w_{02}	w_{01}	w_{00}

Flip corresponding to
both directions

Template convolution

$$\mathbf{I} * \mathbf{T}(i,j) = \sum_{(x,y)\in W} \mathbf{I}_{x,y}\mathbf{T}_{i-x,j-y}$$

Template is actually
flipped around both axes

This does not matter for
symmetric templates
(i.e. the deep learning ones!)

In continuous domain:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

$$= \int_{-\infty}^{+\infty} f(t-\tau)g(\tau)d\tau$$

Two functions

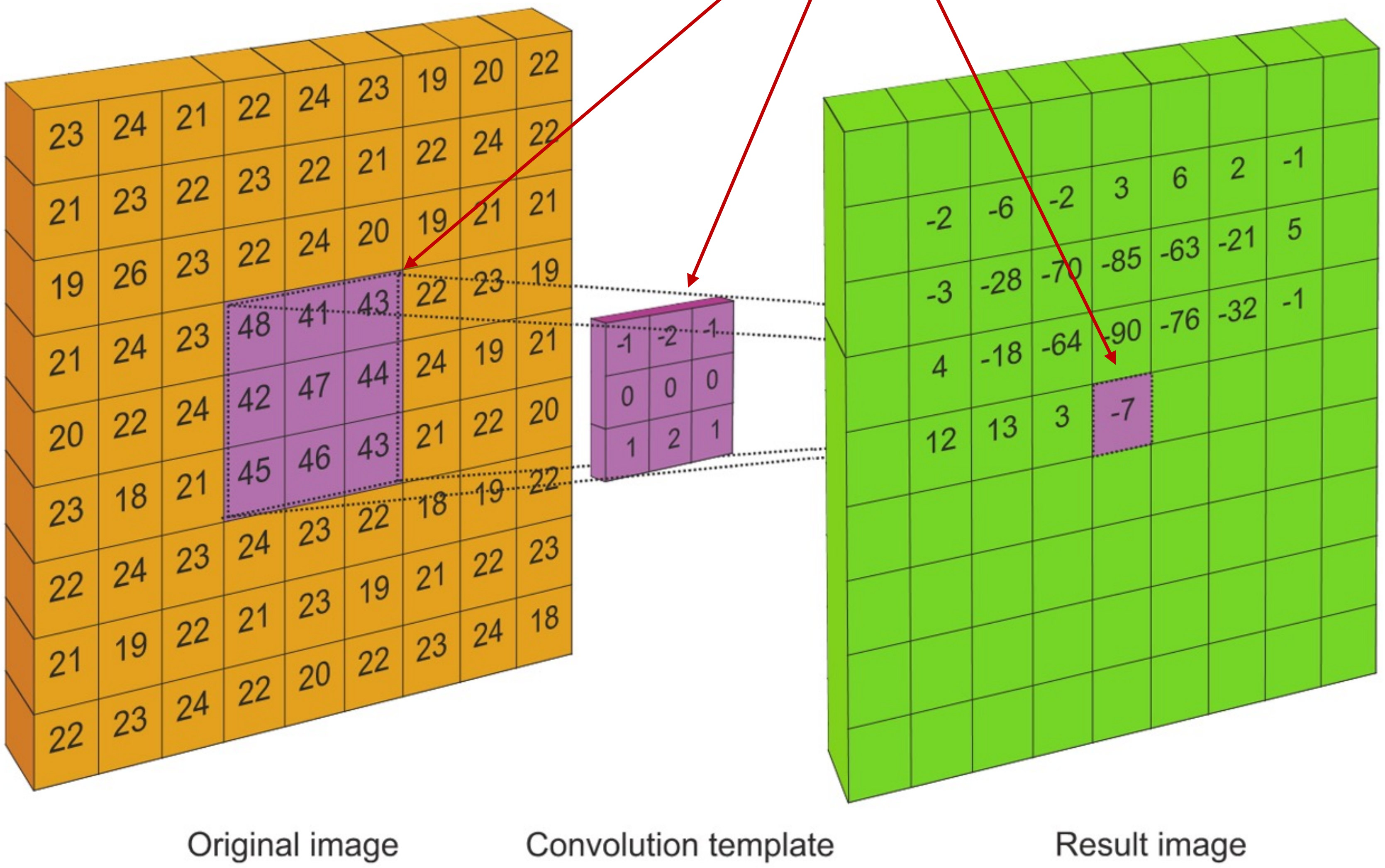
Template convolution

Calculate a new image from the original

Template is inverted for convolution

Template is convolved in a raster fashion

$48 \times 1 + 41 \times 2 + 43 \times 1 + 42 \times 0 + \ldots$



Template convolution

Image

100	100	200	200	200
100	100	200	200	200
100	100	200	200	200
200	200	400	400	400
300	300	400	400	400

G_y

0	0	0	0	0
0	400	400	0	0
0	500	500	0	0
0	600	600	0	0
0	0	0	0	0

Result

0	0	0	0	0
0	707	400	0	0
0	640	860	800	0
0	1000	1000	800	0
0	0	0	0	0

G_x

0	0	0	0	0
0	0	0	0	0
0	-500	-700	-800	0
0	-800	-800	-800	0
0	0	0	0	0

3×3 template and weighting coefficients

w_{00}	w_{01}	w_{02}
w_{10}	w_{11}	w_{12}
w_{20}	w_{21}	w_{22}

weights

$$\mathbf{N}_{x,y} = \sum_{i \in \text{template}} \sum_{j \in \text{template}} w_{i,j} \times \mathbf{O}_{x(i),y(j)}$$

Result calculated for centre point

The position of the point that matches the weighting coefficient position

Averaging operator

Window size 3 × 3, w_{ij} = 1/9,
Window size 5 × 5, w_{ij} = 1/25,
Window size 7 × 7, w_{ij} = 1/49 …

$$\mathbf{N}_{x,y} = \frac{1}{9}\sum_{i\in 3}\sum_{j\in 3}\mathbf{O}_{x(i),y(j)}$$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Before

After

Illustrating the effect of window size

3×3

5×5

7×7

Larger operators remove more noise, but lose more details

2D Gaussian function

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

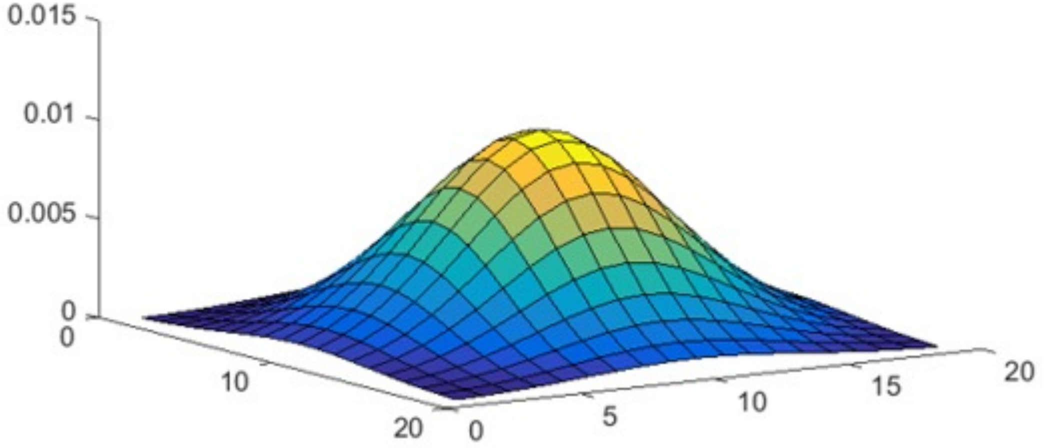
Used to calculate template values

Note compromise between variance σ^2 and window size

Common choices:

Window size: 5×5; σ = 1.0;
Window size: 7×7; σ = 1.2;
Window size: 9×9; σ = 1.4.

2D Gaussian function and template



0.002	0.013	0.022	0.013	0. 002
0.013	0.060	0. 098	0. 060	0.013
0.022	0. 098	0.162	0. 098	0.022
0.013	0. 060	0.098	0. 060	0.013
0. 002	0.013	0.022	0.013	0. 002

Gaussian Averaging Template:
window size 5$\times$5; σ = 1.0

Applying Gaussian averaging

(a) 3 × 3 **(b)** 5 × 5 **(c)** 7 × 7

Comparison

Direct averaging

(a) 5×5 (b) 7×7 (c) 9×9

Which one is better?

Gaussian averaging

(a) 3×3 (b) 5×5 (c) 7×7

Border?

Options:

- Set border to black

- Padding the original image
  1. constant
  2. wrap-around (periodic or cyclic)
  3. symmetric

- Make template smaller near edges

Normally we assume object of interest is near centre so set border to black

Template convolution via the Fourier transform

Convolution theorem allows for **fast** computation via FFT for template size ≥ 7×7

Fourier transform of the picture, $\Im(\mathbf{P})$

Template convolution $*$

Fourier transform of the template, $\Im(\mathbf{T})$

$$\mathbf{P} * \mathbf{T} = \Im^{-1}\left(\Im(\mathbf{P}) .\times \Im(\mathbf{T})\right)$$

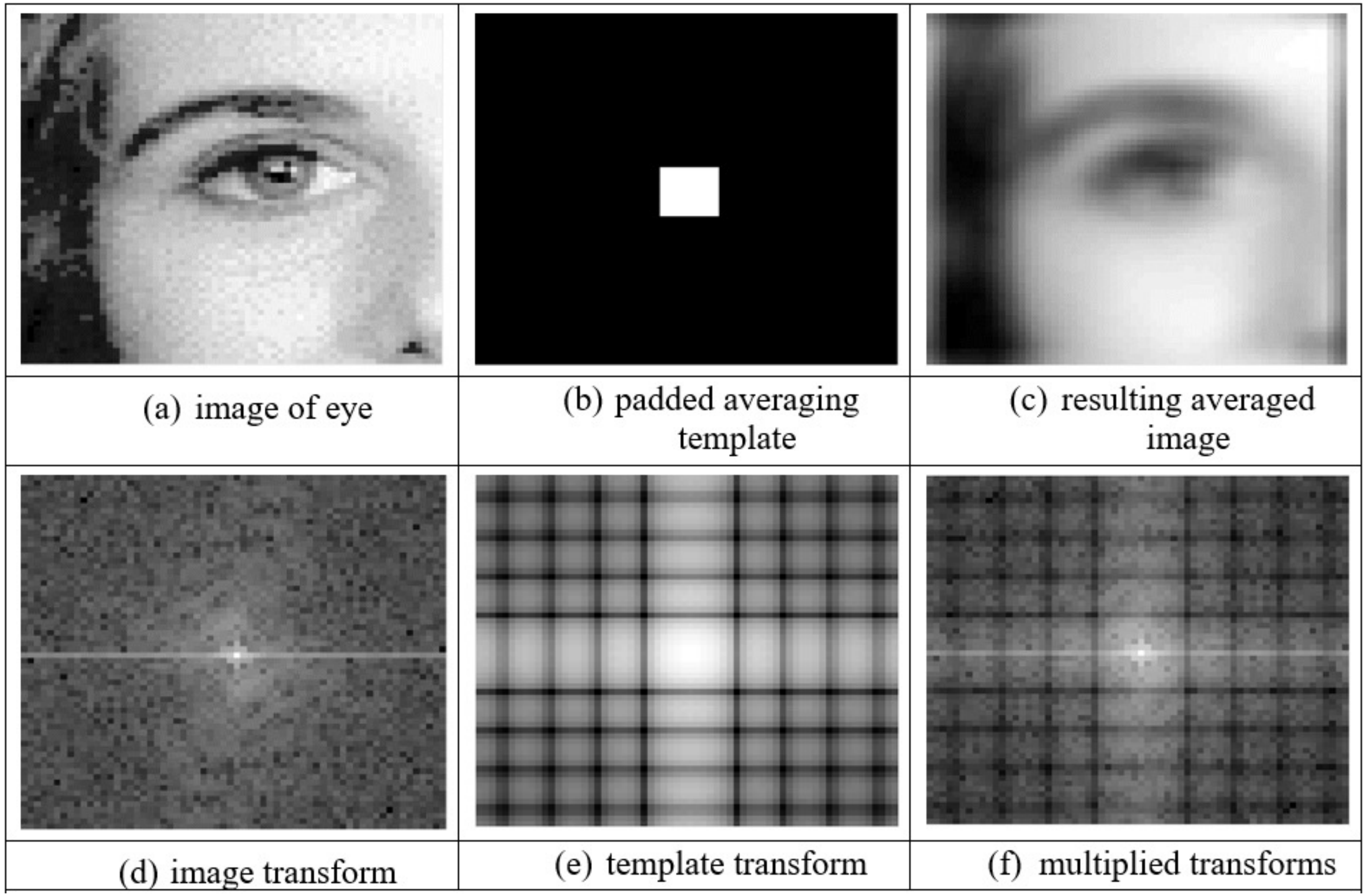
Picture and the template

Inverse Fourier

Point by point multiplication

The inversion is implicit in Fourier
The theory is at end, for information only

Template Convolution via the Fourier Transform



(a) image of eye

(b) padded averaging template

(c) resulting averaged image

(d) image transform

(e) template transform

(f) multiplied transforms

Finding the median from a 3×3 template

2	8	7
4	0	6
3	5	7

(a) 3 × 3 region

2	4	3	8	0	5	7	6	7

(b) unsorted vector

0	2	3	4	5	6	7	7	8

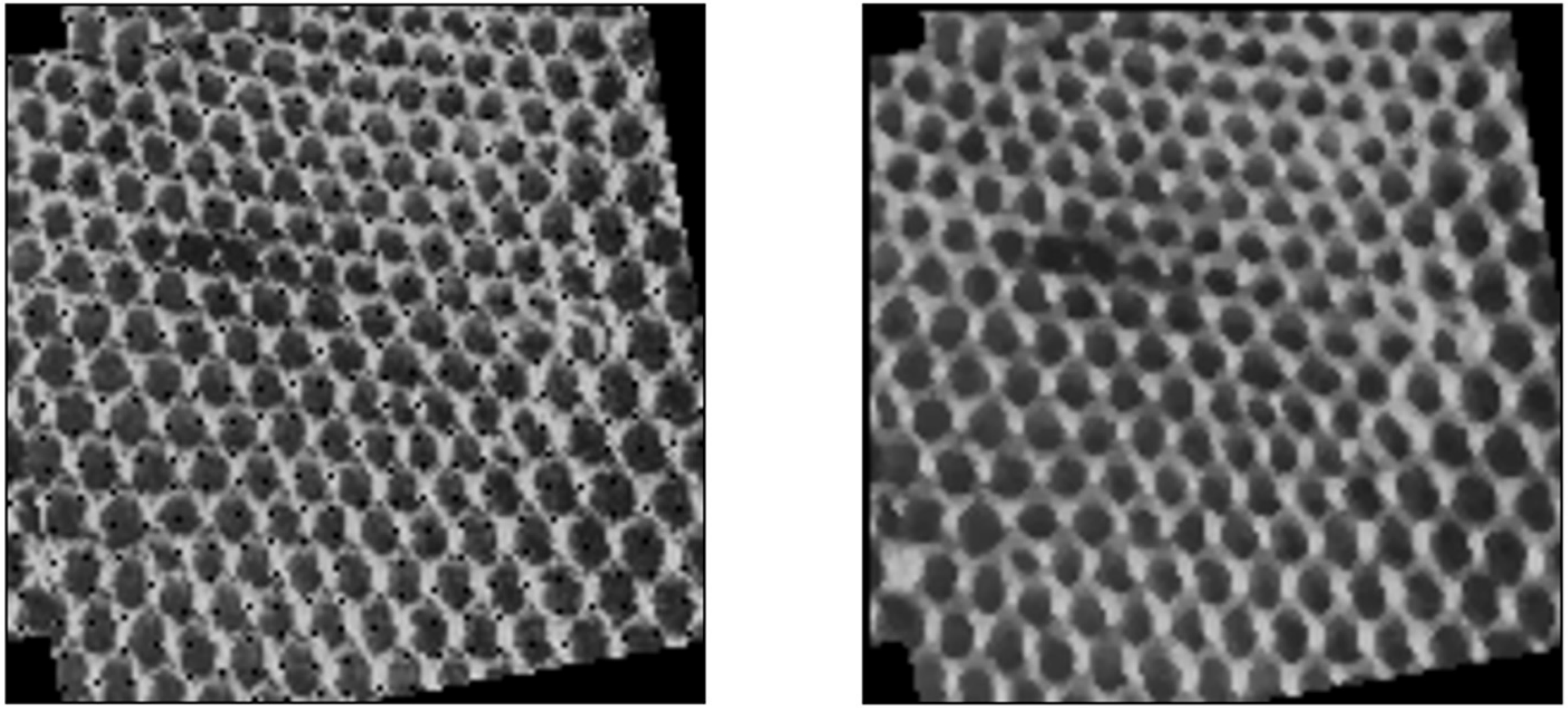
↑ median

(c) sorted vector, giving median

The median is the centre element of a rank-ordered set of template points

Finding the median from a 3×3 template

Pros:
- Preserves edges;
- Removes salt and pepper noise



(a) rotated fence

(b) median filtered

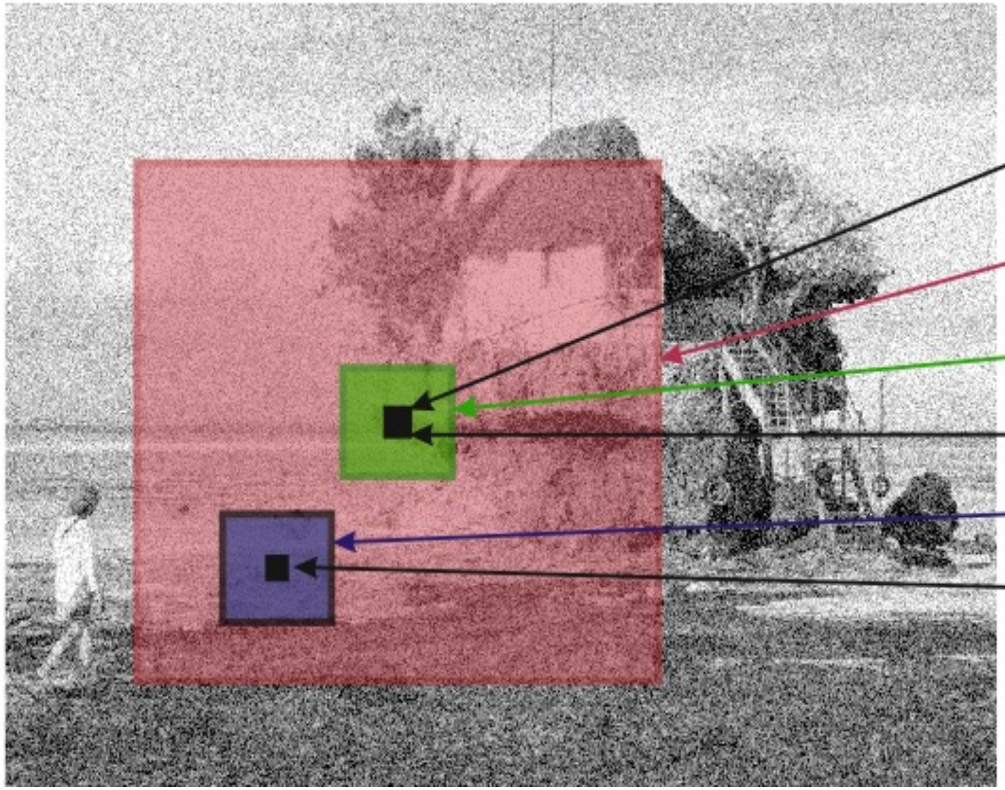
Non-local means

Averaging which preserves regions

$$\mathbf{N}(p) = \frac{1}{C(p)} \sum_{q \in \boldsymbol{\Omega}} \mathbf{O}(q)\, w(p,q)$$

Normalising factor

Gaussian weighting function



Current point x,y

$M \times M$ search region : say $\boldsymbol{\Omega}$

$N \times N$ template

p : local mean $\bar{\mathbf{O}}(p)$

$N \times N$ template

q : local mean $\bar{\mathbf{O}}(q)$

$$\frac{1}{2\pi\sigma^2} e^{-\frac{(\bar{\mathbf{O}}(p) - \bar{\mathbf{O}}(q))^2}{2\sigma^2}}$$

Applying non-local means

Original image

Gaussian averaging

Nonlocal means

(a) Original

(b) (a) with added Gaussian noise

(c) Averaged

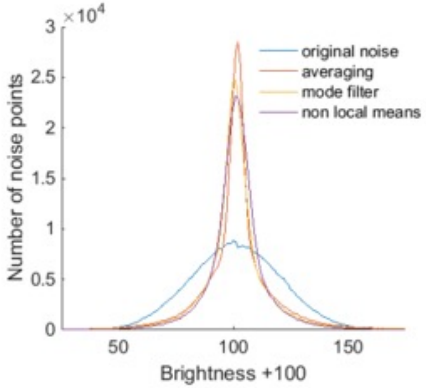
(d) Gaussian smoothed

(e) Median

(f) Truncated Median

(g) Anisotropic diffusion

(h) Non-local-means



(i) Effect of filtering on noise

Comparison of Filtering Operators

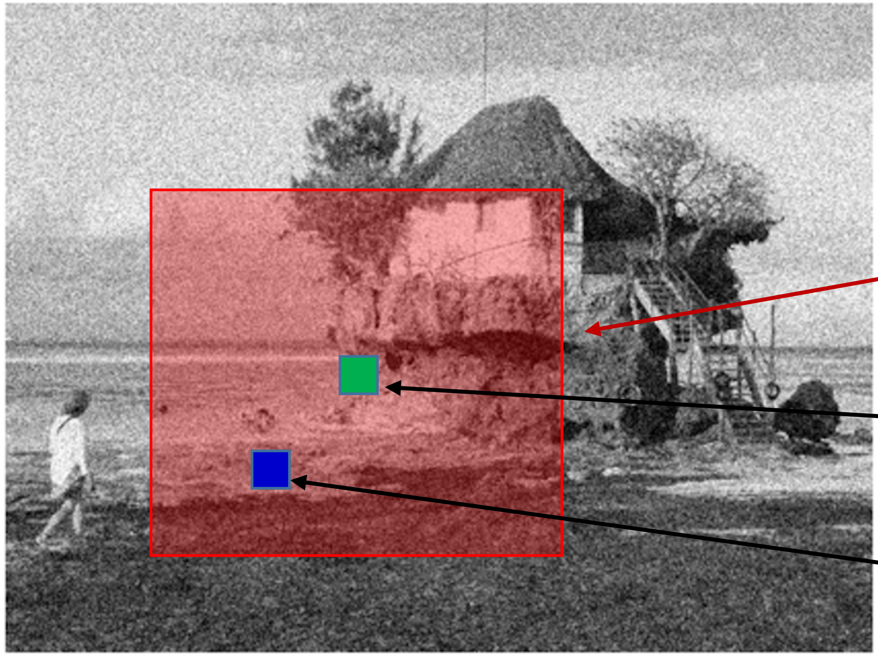
Bilateral filtering

Averaging which preserves regions

$$\mathbf{N}(p) = \frac{1}{C(p)} \sum_{q \in \boldsymbol{\Omega}} \mathbf{O}(q)\, w(p,q)$$

Normalising factor

Weighting function



Region $\boldsymbol{\Omega}$

Pixel p

Pixel q

$$e^{-\frac{(\mathbf{O}(p)-\mathbf{O}(q))^2}{2\sigma_1{}^2}} \times e^{-\frac{(p-q)^2}{2\sigma_2{}^2}}$$

https://en.wikipedia.org/wiki/Bilateral_filter

Main points so far

1 – collection of points is called a template

2 – application to an image is called template convolution

3 – can use Fourier to improve speed

4 – averaging reduces noise

How do we find features? That’s edge detection, coming next

Convolution theorem, for completeness only!

1-D convolution is defined as: $\mathbf{p} * \mathbf{q} = \sum_{i=0}^{N-1} p_i \, q_{m-i}$

by the DFT, for component u:

$$\begin{aligned}\mathcal{F}(\mathbf{p} * \mathbf{q})_u &= \sum_{m=0}^{N-1}\left(\sum_{i=0}^{N-1} p_i \, q_{m-i}\right) e^{-j\frac{2\pi}{N}mu} \\ &= \sum_{i=0}^{N-1} p_i \sum_{m=0}^{N-1} q_{m-i} \, e^{-j\frac{2\pi}{N}mu} \\ &= \sum_{i=0}^{N-1} p_i \sum_{m=0}^{N-1} q_m \, e^{-j\frac{2\pi}{N}mu} e^{-j\frac{2\pi}{N}iu} \\ &= \sum_{i=0}^{N-1} p_i e^{-j\frac{2\pi}{N}iu} \sum_{m=0}^{N-1} q_i \, e^{-j\frac{2\pi}{N}mu} \\ &= \left(\mathcal{F}(\mathbf{p}) \times \mathcal{F}(\mathbf{q})\right)_u\end{aligned}$$

By this, the implementation of discrete convolution using the DFT is achieved by multiplication. For two sampled signals each with N points we have

$$\mathcal{F}(\mathbf{p} * \mathbf{q}) = \mathcal{F}(\mathbf{p}) .\times \mathcal{F}(\mathbf{q})$$

So convolution is the point-wise multiplication of the two transforms, and the template does not need to be inverted. The inversion is implicit in the use of the Fourier transform.

$$F_{f*g}(w) = F_f(w) \times F_g(w)$$ F : Fourier transform

Proof: $$F_{f*g}(w) = \int_{-\infty}^{+\infty} f*g(t)\, e^{-j2\pi wt}\, dt$$

$$= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(\tau)\, g(t-\tau)\, d\tau\; e^{-j2\pi wt}\, dt$$

$$= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} g(t-\tau)\, e^{-j2\pi w(t-\tau)}\, dt \right) f(\tau) \cdot e^{-j2\pi w\tau}\, d\tau$$

$$= \int_{-\infty}^{+\infty} F_g(w) \cdot f(\tau) \cdot e^{-j2\pi w\tau}\, d\tau$$

$$= F_f(w) \times F_g(w) \quad \square$$