Image preprocessing in spatial domain
convolution, convolution theorem, cross-correlation

Revision: 1.5, dated: May 18, 2006

Tomáš Svoboda

Czech Technical University, Faculty of Electrical Engineering
Center for Machine Perception, Prague, Czech Republic

svoboda@cmp.felk.cvut.cz

http://cmp.felk.cvut.cz/~svoboda
Spatial processing—idea

Replace a value of the image function (pixel) by a new one computed from the immediate neighbourhood.
Spatial processing—idea

Replace a value of the image function (pixel) by a new one computed from the immediate neighbourhood.

What is it good for?

- spatial relationships are important in images
- may be faster than a frequency filter
- more natural formulation in some problems
- robust statistics may be applied
Noise in images

- deterioration of analog signal
- CCD/CMOS chips are not perfect
- typically, the smaller active surface, the more noise
Noise in images

- deterioration of analog signal
- CCD/CMOS chips are not perfect
- typically, the smaller active surface, the more noise

How to suppress noise?

- digital only, ie. no A/D and D/A conversion. → OK
- larger chips → EXPENSIVE, EXPENSIVE LENSES
- cooled cameras (astronomy) → SLOW, EXPENSIVE
- (local) image preprocessing
Example scene

Sample video\(^1\) from a static camera

\(^1\)http://cmp.felk.cvut.cz/cmp/courses/EZS/Demos/noise_in_camera.avi
Statistical point of view

Suppose we can acquire $N$ images of the same scene. For each pixel we obtain $N$ results $x_i, i = 1 \ldots N$. Assume:

- observations independent
- each $x_i$ has $\mathbb{E}\{x_i\} = \mu$ and $\text{var}\{x_i\} = \sigma^2$
Suppose we can acquire $N$ images of the same scene. For each pixels we obtain $N$ results $x_i, i = 1 \ldots N$. Assume:

- observations independent
- each $x_i$ has $E\{x_i\} = \mu$ and $\text{var}\{x_i\} = \sigma^2$

Properties of the average value $s_N = \frac{1}{N} \sum_{1}^{N} x_i$
Statistical point of view

Suppose we can acquire $N$ images of the same scene. For each pixels we obtain $N$ results $x_i, i = 1 \ldots N$. Assume:

- observations independent
- each $x_i$ has $\mathbb{E}\{x_i\} = \mu$ and $\text{var}\{x_i\} = \sigma^2$

Properties of the average value $s_N = \frac{1}{N} \sum_{1}^{N} x_i$

- Expectation: $\mathbb{E}\{s_N\} = \frac{1}{N} \sum_{1}^{N} \mathbb{E}\{x_i\} = \mu$
Statistical point of view

Suppose we can acquire $N$ images of the same scene. For each pixels we obtain $N$ results $x_i, i = 1 \ldots N$. Assume:

- observations independent
- each $x_i$ has $E\{x_i\} = \mu$ and $\text{var}\{x_i\} = \sigma^2$

Properties of the average value $s_N = \frac{1}{N} \sum_1^N x_i$

- Expectation: $E\{s_N\} = \frac{1}{N} \sum_1^N E\{x_i\} = \mu$
- Variance: We know that $\text{var}\{x_i/N\} = \text{var}\{x_i\}/N^2$, thus

$$\text{var}\{s_N\} = \frac{\text{var}\{x_1\}}{N^2} + \frac{\text{var}\{x_2\}}{N^2} + \cdots + \frac{\text{var}\{x_N\}}{N^2} = \frac{\sigma^2}{N}.$$ 

which means that standard deviation of $s_N$ decreases as $\frac{1}{\sqrt{N}}$. 
Example

a noisy image

average from $\approx 60$ observations.
Example — equalized

a noisy image

average from $\approx 60$ observations.
Standard deviations in pixels for images:

Lossy compression is generally not a good choice for machine vision!
Problem: noise suppression from just one image

- redundancy in images
- neighbouring pixels have mostly the same or similar value
- correction of the pixel value based on an analysis of its neighbourhood
- leads to image blurring
Problem: noise suppression from just one image

- redundancy in images
- neighbouring pixels have mostly the same or similar value
- correction of the pixel value based on an analysis of its neighbourhood
- leads to image blurring

spatial filtering
Spatial filtering — informally

Idea: Output is a function of a pixel value and those of its neighbours.

Example for 8-connected region.

\[
g(x, y) = \text{Op} \begin{bmatrix} f(x-1, y-1) & f(x, y-1) & f(x+1, y-1) \\ f(x-1, y) & f(x, y) & f(x+1, y) \\ f(x-1, y+1) & f(x, y+1) & f(x+1, y+1) \end{bmatrix}
\]

Possible operations: sum, average, weighted sum, min, max, median . . .
Spatial filtering by masks

- Very common neighbour operation is per-element multiplication with a set of weights and sum together.

- Set of weights is often called *mask* or *kernel*.

<table>
<thead>
<tr>
<th>Local neighbourhood</th>
<th>mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x-1,y-1) )</td>
<td>( w(-1,-1) )</td>
</tr>
<tr>
<td>( f(x,y-1) )</td>
<td>( w(0,-1) )</td>
</tr>
<tr>
<td>( f(x+1,y-1) )</td>
<td>( w(+1,-1) )</td>
</tr>
<tr>
<td>( f(x-1,y) )</td>
<td>( w(-1,0) )</td>
</tr>
<tr>
<td>( f(x,y) )</td>
<td>( w(0,0) )</td>
</tr>
<tr>
<td>( f(x+1,y) )</td>
<td>( w(+1,0) )</td>
</tr>
<tr>
<td>( f(x-1,y+1) )</td>
<td>( w(-1,+1) )</td>
</tr>
<tr>
<td>( f(x,y+1) )</td>
<td>( w(0,+1) )</td>
</tr>
<tr>
<td>( f(x+1,y+1) )</td>
<td>( w(+1,+1) )</td>
</tr>
</tbody>
</table>

\[
g(x, y) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} w(k, l) f(x + k, y + l)
\]
2D convolution

- Spatial filtering is often referred to as **convolution**.
- We say, we **convolve** the image by a kernel or mask.
- Though, it is not the same. Convolution uses a flipped kernel.

<table>
<thead>
<tr>
<th>Local neighbourhood</th>
<th>mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>f(x-1,y-1)</td>
<td>w(+1,+1)</td>
</tr>
<tr>
<td>f(x,y-1)</td>
<td>w(0,+1)</td>
</tr>
<tr>
<td>f(x+1,y-1)</td>
<td>w(-1,+1)</td>
</tr>
<tr>
<td>f(x-1,y)</td>
<td>w(+1,0)</td>
</tr>
<tr>
<td>f(x,y)</td>
<td>w(0,0)</td>
</tr>
<tr>
<td>f(x+1,y)</td>
<td>w(-1,0)</td>
</tr>
<tr>
<td>f(x-1,y+1)</td>
<td>w(+1,-1)</td>
</tr>
<tr>
<td>f(x,y+1)</td>
<td>w(0,-1)</td>
</tr>
<tr>
<td>f(x+1,y+1)</td>
<td>w(-1,-1)</td>
</tr>
</tbody>
</table>

\[ g(x, y) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} w(k, l) f(x - k, y - l) \]
2D Convolution — Why is it important?

- Input and output signals need not to be related through convolution, but if they are (and only if) the system is linear and time invariant.

\[ f(x) \xrightarrow{h(x)} g(x) = h(x) \ast f(x) \]
2D Convolution — Why is it important?

- Input and output signals need not to be related through convolution, but if they are (and only if) the system is linear and time invariant.

\[ g(x) = h(x) \ast f(x) \]

- 2D convolution describes well the formation of images.
2D Convolution — Why is it important?

- Input and output signals need not to be related through convolution, but if they are (and only if) the system is linear and time invariant.

\[ g(x) = h(x) \ast f(x) \]

- 2D convolution describes well the formation of images.
- Many image distortions made by imperfect acquisition may be modelled by 2D convolution, too.
2D Convolution — Why is it important?

- Input and output signals need not to be related through convolution, but if they are (and only if) the system is linear and time invariant.

\[ g(x) = h(x) * f(x) \]

- 2D convolution describes well the formation of images.
- Many image distortions made by imperfect acquisition may be modelled by 2D convolution, too.
- It is a powerful thinking tool.
2D convolution — definition

Convolution integral

\[ g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x - k, y - l)h(k, l)dkdl \]
2D convolution — definition

Convolution integral

\[ g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x - k, y - l) h(k, l) dk dl \]

Symbolic abbreviation

\[ g(x, y) = f(x, y) * h(x, y) \]
Discrete 2D convolution

\[ g(x, y) = f(x, y) \ast h(x, y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f(x - k, y - l) h(k, l) \]

What with missing values \( f(x - k, y - l) \)?

Zero-padding: add zeros where needed.

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 2 & 1 \\
\end{bmatrix}
\ast
\begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
\end{bmatrix}
= 
\]
Discrete 2D convolution

\[ g(x, y) = f(x, y) * h(x, y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f(x - k, y - l) h(k, l) \]

What with missing values \( f(x - k, y - l) \)?

Zero-padding: add zeros where needed.

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 2 & 1
\end{bmatrix}
\ast
\begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix}
= 
\begin{bmatrix}
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 2 & 2 & 1 \\
1 & 2 & 3 & 3 & 1 \\
1 & 2 & 3 & 1 & 0 \\
1 & 2 & 1 & 0 & 0
\end{bmatrix}
\]

The result is zero elsewhere. The concept is somehow contra-intuitive, practice with a pencil and paper.
Thinking about convolution

\[ g(x) = f(x) \ast h(x) = \sum_k f(k)h(x-k) \]
Thinking about convolution

\[ g(x) = f(x) \ast h(x) = \sum_{k} f(k)h(x - k) \]

Blurring \( f \):
Thinking about convolution

\[ g(x) = f(x) * h(x) = \sum_k f(k)h(x - k) \]

Blurring \( f \):

- break the \( f \) into each discrete sample
Thinking about convolution

\[ g(x) = f(x) \ast h(x) = \sum_{k} f(k)h(x - k) \]

Blurring $f$:

- break the $f$ into each discrete sample
- send each one individually through $h$ to produce blurred points
Thinking about convolution

$$g(x) = f(x) * h(x) = \sum_k f(k)h(x - k)$$

Blurring $f$:

- break the $f$ into each discrete sample
- send each one individually through $h$ to produce blurred points
- sum up the blurred points
Thinking about convolution

\[ g(x) = f(x) \ast h(x) = \sum_{k} f(k)h(x - k) \]

Blurring \( f \):

- break the \( f \) into each discrete sample
- send each one individually through \( h \) to produce blurred points
- sum up the blurred points

Shifting \( h \):
Thinking about convolution

\[ g(x) = f(x) \ast h(x) = \sum_k f(k)h(x - k) \]

**Blurring** \( f \):
- break the \( f \) into each discrete sample
- send each one individually through \( h \) to produce blurred points
- sum up the blurred points

**Shifting** \( h \):
- shift a copy of \( h \) to each position \( k \)
Thinking about convolution

\[ g(x) = f(x) \ast h(x) = \sum_{k} f(k)h(x - k) \]

Blurring \( f \):
- break the \( f \) into each discrete sample
- send each one individually through \( h \) to produce blurred points
- sum up the blurred points

Shifting \( h \):
- shift a copy of \( h \) to each position \( k \)
- multiply by the value at that position \( f(k) \)
Thinking about convolution

\[ g(x) = f(x) \ast h(x) = \sum_k f(k)h(x - k) \]

Blurring \( f \):
- break the \( f \) into each discrete sample
- send each one individually through \( h \) to produce blurred points
- sum up the blurred points

Shifting \( h \):
- shift a copy of \( h \) to each position \( k \)
- multiply by the value at that position \( f(k) \)
- add shifted, multiplied copies for all \( k \)
Thinking about convolution II

\[ g(x) = f(x) \ast h(x) = \sum_{k} f(x - k)h(k) \]

Mask filtering:

- flip the function \( h \) around zero
Thinking about convolution II

\[ g(x) = f(x) \ast h(x) = \sum_k f(x - k)h(k) \]

Mask filtering:

- flip the function \( h \) around zero
- shift to output position \( x \)
Thinking about convolution II

\[ g(x) = f(x) \ast h(x) = \sum_k f(x - k)h(k) \]

Mask filtering:

- flip the function \( h \) around zero
- shift to output position \( x \)
- point-wise multiply for each position \( k \) value \( f(x - k) \) and the shifted flipped copy of \( h \).
Thinking about convolution II

\[ g(x) = f(x) \ast h(x) = \sum_k f(x - k)h(k) \]

Mask filtering:

- flip the function \( h \) around zero
- shift to output position \( x \)
- point-wise multiply for each position \( k \) value \( f(x - k) \) and the shifted flipped copy of \( h \).
- sum for all \( k \) and write that value at position \( x \)
Motion blur modelled by convolution

\[ g(x) = \sum_k f(x - k)h(k) \]

- \( g(x) \) is the image we get
- \( f(x) \) say to be the (true) 2D function
- \( g \) does not depend only on \( f(x) \) but also on all \( k \) previous values of \( f \)
- \( \#k \) measures the amount of the motion
- if the motion is steady then \( h(k) = 1/(\#k) \)

\( h \) is impulse response of the system (camera), we will come to that later

Camera moves along \( x \) axis during acquisition.
Spatial filtering vs. convolution — Flipping kernel

Why not \( g(x) = \sum_k f(x + k)h(k) \) as in spatial filtering but
\( g(x) = \sum_k f(x - k)h(-k) \)?
Spatial filtering vs. convolution — Flipping kernel

Why not \( g(x) = \sum_k f(x + k)h(k) \) as in spatial filtering but
\( g(x) = \sum_k f(x - k)h(-k) \)?

Causality!
Spatial filtering vs. convolution — Flipping kernel

Why not \( g(x) = \sum_k f(x + k)h(k) \) as in spatial filtering but
\( g(x) = \sum_k f(x - k)h(-k) \)?

Causality!

In \( g(x) = \sum_k f(x + k)h(k) \) we are asking for values of input function \( f \) that are yet to come!
Spatial filtering vs. convolution — Flipping kernel

Why not \( g(x) = \sum_k f(x + k)h(k) \) as in spatial filtering but
\( g(x) = \sum_k f(x - k)h(-k) \)?

Causality!

In \( g(x) = \sum_k f(x + k)h(k) \) we are asking for values of input function \( f \) that are yet to come!

Solution: \( h(-k) \)
Convolution theorem

The Fourier transform of a convolution is the product of the Fourier transforms.

\[ \mathcal{F}\{f(x, y) \ast h(x, y)\} = F(u, v)H(u, v) \]
Convolution theorem

The Fourier transform of a convolution is the product of the Fourier transforms.

\[ \mathcal{F}\{ f(x, y) \ast h(x, y) \} = F(u, v)H(u, v) \]

The Fourier transform of a product is the convolution of the Fourier transforms.

\[ \mathcal{F}\{ f(x, y)h(x, y) \} = F(u, v) \ast H(u, v) \]
Convolution theorem — proof

\[ F\{ f(x, y) \ast h(x, y) \} = F(u, v)H(u, v) \]

\[ F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) \exp\left(-i2\pi ux/M\right) \text{ and } g(x) = \sum_{k=0}^{M-1} f(k)h(x - k) \]

\[ F\{g(x)\} = \ldots \]

\[ \frac{1}{M} \sum_{x=0}^{M-1} \sum_{k=0}^{M-1} f(k)h(x - k)e^{-i2\pi ux/M} \]

\[ \text{introduce new (dummy) variable } w = x - k \]

\[ \frac{1}{M} \sum_{k=0}^{M-1} f(k) \sum_{w=-k}^{(M-1)-k} h(w)e^{-i2\pi u(w+k)/M} \]

\[ \text{remember that all functions } g, h, f \text{ are assumed to be periodic with period } M \]

\[ \frac{1}{M} \sum_{k=0}^{M-1} f(k)e^{-i2\pi uk/M} \sum_{w=0}^{M-1} h(w)e^{-i2\pi uw/M} \]

\[ \text{which is indeed } F(u)H(u) \]
Convolution theorem — what is it good for?

- Direct relationship between filtering in spatial and frequency domain. See few slides later.
Convolution theorem — what is it good for?

✦ Direct relationship between filtering in spatial and frequency domain. See few slides later.

✦ Image restoration, sometimes called deconvolution
Convolution theorem — what is it good for?

- Direct relationship between filtering in spatial and frequency domain. See few slides later.
- Image restoration, sometimes called deconvolution
- Speed of computation. Convolution has $O(M^2)$, Fast Fourier Transform (FFT) has $O(M \log_2 M)$
Convolution theorem — what is it good for?

- Direct relationship between filtering in spatial and frequency domain. See few slides later.
- Image restoration, sometimes called deconvolution
- Speed of computation. Convolution has $O(M^2)$, Fast Fourier Transform (FFT) has $O(M \log_2 M)$
- ... but, some frequency filters may be well approximated by a small spatial mask.
Convolution theorem — what is it good for?

- Direct relationship between filtering in spatial and frequency domain. See few slides later.

- Image restoration, sometimes called deconvolution

- Speed of computation. Convolution has $O(M^2)$, Fast Fourier Transform (FFT) has $O(M \log_2 M)$

- ... but, some frequency filters may be well approximated by a small spatial mask.

Enough theory for now. Go for examples ...
Spatial filtering

What is it good for?

- smoothing
- sharpening
- noise removal
- edge detection
- pattern matching
- ...

Smoothing

Output value is computed as an average of the input value and its neighbourhood.

- Advantage: less noise
Smoothing

Output value is computed as an average of the input value and its neighbourhood.

- Advantage: less noise
- Disadvantage: blurring
Smoothing

Output value is computed as an average of the input value and its neighbourhood.

- Advantage: less noise
- Disadvantage: blurring
- Any kernel with all positive weights causes smoothing or blurring
Smoothing

Output value is computed as an average of the input value and its neighbourhood.

- Advantage: less noise
- Disadvantage: blurring
- Any kernel with all positive weights causes smoothing or blurring
- They are called low-pass filters (We know them already!)
Smoothing

Output value is computed as an average of the input value and its neighbourhood.

- Advantage: less noise
- Disadvantage: blurring
- Any kernel with all positive weights causes smoothing or blurring
- They are called low-pass filters (We know them already!)

Averaging:

\[
g(x, y) = \frac{\sum_{k} \sum_{l} w(k, l) f(x + k, y + l)}{\sum_{k} \sum_{l} w(k, l)}
\]
Smoothing kernels

Can be of any size, any shape

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad h = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix},$$

$$h = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$
Averaging ones ($n \times n$) — increasing mask size

image 1024 x 768

7 x 7

11 x 11

15 x 15

29 x 29

43 x 43
Frequency analysis of the spatial convolution — Simple averaging

Original image

$21 \times 21$ constant mask

Filtered image
Frequency analysis of the spatial convolution — Gaussian smoothing

Original image

$21 \times 21$ Gauss. mask

filtered image
Simple averaging vs. Gaussian smoothing

Both images blurred but filtering by a constant mask still shows up some high frequencies!
Frequency analysis of the spatial convolution — Simple averaging

Original image

21 × 21 const. mask

filtered image
Frequency analysis of the spatial convolution — Gaussian smoothing
Simple averaging vs. Gaussian smoothing

Both images blurred but filtering by a constant mask still shows up some high frequencies!
Non-linear smoothing

Goal: reduce blurring of image edges during smoothing
Non-linear smoothing

**Goal**: reduce blurring of image edges during smoothing

**Homogeneous neighbourhood**: find a proper neighbourhood where the values have minimal variance.
Non-linear smoothing

**Goal**: reduce blurring of image edges during smoothing

**Homogeneous neighbourhood**: find a proper neighbourhood where the values have minimal variance.

**Robust statistics**: something better than the mean.
Rotation mask $3 \times 3$ seeks a homogeneous part at $5 \times 5$ neighbourhood.

Together 9 positions, 1 in the middle $+ 8$ on the image.

The mask with the lowest variance is selected as the proper neighbourhood.
Rotation mask—original image
Rotation mask—first filtration
Rotation mask—second filtration
Rotation mask—third filtration
Rotation mask—fourth filtration
Rotation mask—fifth filtration
Nonlinear smoothing — Robust statistics

Order-statistic filters
Nonlinear smoothing — Robust statistics

Order-statistic filters

- median
Nonlinear smoothing — Robust statistics

Order-statistic filters

- median
  - Sort values and select the middle one.
Nonlinear smoothing — Robust statistics

Order-statistic filters

- median
  - Sort values and select the middle one.
  - A method of edge-preserving smoothing.
  - Particularly useful for removing salt-and-pepper, or impulse noise.
Nonlinear smoothing — Robust statistics

Order-statistic filters

◆ median

- Sort values and select the middle one.
- A method of edge-preserving smoothing.
- Particularly useful for removing salt-and-pepper, or impulse noise.

◆ trimmed mean

- Throw away outliers and average the rest.
- More robust to a non-Gaussian noise than a standard averaging.
Median filtering

<table>
<thead>
<tr>
<th></th>
<th>100</th>
<th>98</th>
<th>102</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>105</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>100</td>
<td>255</td>
<td></td>
</tr>
</tbody>
</table>
Median filtering

<table>
<thead>
<tr>
<th></th>
<th>100</th>
<th>98</th>
<th>102</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>105</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>100</td>
<td>255</td>
<td></td>
</tr>
</tbody>
</table>

Mean = 117.2
Median filtering

Mean = 117.2

median: 95 98 99 100 100 101 102 105 255

Very robust, up to 50% of values may be outliers.
Nonlinear smoothing examples

noisy image

averaging $3 \times 3$

noisy image

median $3 \times 3$

The median filtering damage corners and thin edges.
Cross-correlation

\[ g(x, y) = \sum_k \sum_l h(k, l) f(x + k, y + l) = h(x, y) \ast f(x, y) \]

Cross-correlation is not, unlike convolution, commutative

\[ h(x, y) \ast f(x, y) \neq f(x, y) \ast h(x, y) \]

When \( h(x, y) \ast f(x, y) \) we often say that \( h \) scans \( f \).

Cross-correlation is related to convolution through

\[ h(x, y) \ast f(x, y) = h(x, y) \ast f(-x, -y) \]

Cross-correlation is useful for pattern matching
Cross-correlation

This is perhaps not exactly what we expected and what we want. The result depend on the amplitudes. Do we have some normalisation?
Normalised cross-correlation

Sometimes called correlation coefficient

\[
c(x, y) = \frac{\sum_k \sum_l (h(k, l) - \bar{h})(f(x + k, y + l) - \bar{f}(x, y))}{\sqrt{\sum_k \sum_l (h(k, l) - \bar{h})^2 \sum_k \sum_l (f(x + k, y + l) - \bar{f}(x, y))^2}}
\]

\hspace{1cm} \bullet \bar{h} \text{ is the mean of } h

\hspace{1cm} \bullet \bar{f}(x, y) \text{ is the mean of the } k, l \text{ neighbourhood around } (x, y)

\hspace{1cm} \bullet \sum_k \sum_l (h(k, l) - \bar{h})^2 \text{ and } \sum_k \sum_l (f(x + k, y + l) - \bar{f}(x, y))^2 \text{ are indeed the variances. }

\hspace{1cm} \bullet -1 \leq c(x, y) \leq 1
Normalised cross-correlation

\[ h(x, y) \quad f(x, y) \quad g(x, y) \]

The \(-1\)'s are in fact undefined, \( NaN \). The maximum response is indeed where we expected.
Normalised cross-correlation — real images

\[ h(x, y) \quad f(x, y) \quad g(x, y) \]
Normalised cross-correlation — non-maxima suppression

Red rectangle denotes the pattern. The crosses are the 5 highest values of ncc after non-maxima suppression.
Normalised cross-correlation — non-maxima suppression

Red rectangle denotes the pattern. The crosses are the 10 highest values of ncc after non-maxima suppression.

We see the problem. The algorithm finds the cow in any position in the image. However, it does not scale.
Red rectangle denotes the **pattern**. The crosses are the 10 highest values of ncc after non-maxima suppression.

We see the problem. The algorithm finds the cow in any position in the image. However, it does not **scale**.

But we leave the problem for some advanced computer vision course.
Autocorrelation

\[ g(x, y) = f(x, y) \ast f(x, y) \]
References
Standard deviation in red channel

100  200  300  400  500  600
50 100 150 200 250 300 350 400 450

1 1.5 2 2.5 3 3.5 4 4.5
\[ f(x) \quad h(x) \quad g(x) = h(x) \ast f(x) \]