Lecture 04
Image Filtering

Davide Scaramuzza
http://rpg.ifi.uzh.ch
No exercise this afternoon

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Description of the lecture/exercise</th>
<th>Lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.09.2018</td>
<td>10:15 - 12:00</td>
<td>01 - Introduction</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td>27.09.2018</td>
<td>10:15 - 12:00</td>
<td>02 - Image Formation 1: perspective projection and camera models</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise 1: Augmented reality wireframe cube</td>
<td>Titus Cieslewski &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>04.10.2018</td>
<td>10:15 - 12:00</td>
<td>03 - Image Formation 2: camera calibration algorithms</td>
<td>Guillermo Gallego</td>
</tr>
<tr>
<td>1.10.2018</td>
<td>10:15 - 12:00</td>
<td>04 - Filtering &amp; Edge detection</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td>18.10.2018</td>
<td>10:15 - 12:00</td>
<td>05 - Point Feature Detectors 1: Harris detector</td>
<td>Guillermo Gallego</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise 3: Harris detector + descriptor + matching</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>25.10.2018</td>
<td>10:15 - 12:00</td>
<td>06 - Point Feature Detectors 2: SIFT, BRIEF, BRISK</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td>01.11.2018</td>
<td>10:15 - 12:00</td>
<td>07 - Multiple-view geometry 1</td>
<td>Guillermo Gallego</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise 4: Stereo vision: rectification, epipolar matching, disparity, triangulation</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>08.11.2018</td>
<td>10:15 - 12:00</td>
<td>08 - Multiple-view geometry 2</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise 5: Eight-Point algorithm</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>15.11.2018</td>
<td>10:15 - 12:00</td>
<td>09 - Multiple-view geometry 3</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise 6: P3P algorithm and RANSAC</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>22.11.2018</td>
<td>10:15 - 12:00</td>
<td>10 - Dense 3D Reconstruction (Multi-view Stereo)</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise session: Intermediate VO Integration</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>29.11.2018</td>
<td>10:15 - 12:00</td>
<td>11 - Optical Flow and Tracking (Lucas-Kanade)</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise 7: Lucas-Kanade tracker</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>06.12.2018</td>
<td>10:15 - 12:00</td>
<td>12 – Place recognition</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise session: Deep Learning Tutorial</td>
<td>Antonio Loquercio</td>
</tr>
<tr>
<td>13.12.2018</td>
<td>10:15 - 12:00</td>
<td>13 – Visual inertial fusion</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>13:15 – 15:00</td>
<td>Exercise 8: Bundle adjustment</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
<tr>
<td>20.12.2018</td>
<td>10:15 - 12:00</td>
<td>14 - Event based vision</td>
<td>Davide Scaramuzza</td>
</tr>
<tr>
<td></td>
<td>12:30 – 13:30</td>
<td><strong>Scaramuzza’s lab visit and live demonstrations</strong>: Andreasstrasse 15, 2.11, 8050</td>
<td>Davide Scaramuzza &amp; his lab</td>
</tr>
<tr>
<td></td>
<td>14:00 – 16:00</td>
<td>Exercise session: final VO integration (it will take place close to Scaramuzza’s lab)</td>
<td>Antonio Loquercio &amp; Mathias Gehrig</td>
</tr>
</tbody>
</table>
Image filtering

- The word *filter* comes from frequency-domain processing, where “filtering” refers to the process of accepting or rejecting certain frequency components.
- We distinguish between low-pass and high-pass filtering:
  - A **low-pass filter** smooths an image (retains low-frequency components).
  - A **high-pass filter** retains the contours (also called edges) of an image (high frequency).
Low-pass filtering
Low-pass filtering
Motivation: noise reduction

- **Salt and pepper noise**: random occurrences of black and white pixels

- **Impulse noise**: random occurrences of white pixels

- **Gaussian noise**: variations in intensity drawn from a Gaussian distribution

Source: S. Seitz
Gaussian noise

How could we reduce the noise to try to recover the “ideal image”?
Moving average

- Replaces each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise process to be independent from pixel to pixel
Moving average

- Replaces each pixel with an average of all the values in its neighborhood
- Moving average in 1D:
Weighted Moving Average

- Can add weights to our moving average
- *Weights* $[1, 1, 1, 1, 1] / 5$
Weighted Moving Average

- Non-uniform weights $[1, 4, 6, 4, 1] / 16$
This operation is called convolution

Example of convolution of two sequences (or “signals”)
- One of the sequences is flipped (right to left) before sliding over the other
- Notation: $a \ast b$
- Nice properties: linearity, associativity, commutativity, etc.
This operation is called *convolution*

Example of convolution of two sequences (or “signals”)

- One of the sequences is flipped (right to left) before sliding over the other
- Notation: $a \ast b$
- Nice properties: linearity, associativity, commutativity, etc.
2D Filtering

• Convolution:
  – Flip the filter in both dimensions (bottom to top, right to left) (=180 deg turn)
  – Then slide the filter over the image and compute sum of products

\[ G[x, y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u, v] H[x - u, y - v] \]

\[ G = F \ast H \]

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter \( H \) is also called “kernel” or “mask”.
Review: Convolution vs. Cross-correlation

Convolution

\[ G[x, y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u, v]H[x - u, y - v] \]

\[ G = F \ast H \]

Properties: linearity, associativity, commutativity

Cross-correlation

\[ G[x, y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u, v]H[x + u, y + v] \]

\[ G = F \otimes H \]

Properties: linearity, but not associativity and commutativity

For a Gaussian or box filter, how will the outputs differ?
Example: Moving Average in 2D

**Input image**

$F[x, y]$

**Filtered image**

$G[x, y]$

"box filter"
Example: Moving Average in 2D

Input image

\[ F[x, y] \]

Filtered image

\[ G[x, y] \]
Example: Moving Average in 2D

Input image

\[ F[x, y] \]

Filtered image

\[ G[x, y] \]
Example: Moving Average in 2D

Input image

\[ F[x, y] \]

Filtered image

\[ G[x, y] \]
Example: Moving Average in 2D

Input image

\[ F[x, y] \]

Filtered image

\[ G[x, y] \]
Example: Moving Average in 2D

Input image

\[ F(x, y) \]

Filtered image

\[ G(x, y) \]
Example: Moving Average in 2D

Box filter:
white = high value, black = low value

original

filtered
Gaussian filter

- What if we want the closest pixels to have higher influence on the output?

This kernel is the approximation of a Gaussian function:

$$H[u, v] = \frac{1}{2\pi\sigma^2}e^{-\frac{u^2 + v^2}{2\sigma^2}}$$
Smoothing with a Gaussian
Compare the result with a box filter

This effect is called aliasing
Gaussian filters

- What parameters matter?
- **Size** of the kernel
  - NB: a Gaussian function has **infinite support**, but discrete filters use finite kernels

\[
\sigma = 5 \text{ pixels} \\
\text{with 10 x 10 pixel kernel}
\]

\[
\sigma = 5 \text{ pixels} \\
\text{with 30 x 30 pixel kernel}
\]

Which one is better (left or right)?
Gaussian filters

• What parameters matter?
• Variance of Gaussian: control the amount of smoothing

Recall: standard deviation = \( \sigma \) [pixels], variance = \( \sigma^2 \) [pixels\(^2\)]
Smoothing with a Gaussian

σ is called “scale” or “width” or “spread” of the Gaussian kernel, and controls the amount of smoothing.
Sample Matlab code

```matlab
>> hsize = 20;
>> sigma = 5;
>> h = fspecial('gaussian', hsize, sigma);

>> mesh(h);

>> imagesc(h);

>> im = imread('panda.jpg');
>> outim = imfilter(im, h);
>> imshow(outim);
```
Boundary issues

• What about near the image edges?
  – the filter window falls off the edges of the image
  – need to pad the image borders
  – methods:
    • zero padding (black)
    • wrap around
    • copy edge
    • reflect across edge
Summary on (linear) smoothing filters

• **Smoothing filter**
  – has positive values (also called coefficients)
  – sums to 1 → preserve brightness of constant regions
  – removes “high-frequency” components; “low-pass” filter
Non-linear filtering
Effect of smoothing filters

Linear smoothing filters do not alleviate salt and pepper noise!
Median filter

- It is a non-linear filter
- Removes spikes: good for “impulse noise” and “salt & pepper noise”

**Input image**

- 10 15 20
- 23 90 27
- 33 31 30

**Element to be replaced**

**Sort**

10 15 20 23 27 30 31 33 90

**Median value**

**Output image**

- 10 15 20
- 23 27 27
- 33 31 30

**Replace element**
Median filter

Salt and pepper noise → 

Median filtered

Plots of a row of the image
Median filter

- Median filter preserves sharp transitions (i.e., edges),

... but it removes small brightness variations.
High-pass filtering
(edge detection)
Edge detection

- Ultimate goal of edge detection: an idealized line drawing.
- Edge contours in the image correspond to important scene contours.
Edges are sharp intensity changes
Images as functions $F(x, y)$

Edges look like steep cliffs
Derivatives and edges

An edge is a place of rapid change in the image intensity function.

The diagram shows:
- An image with a horizontal scanline.
- The intensity function along this scanline, which shows a smooth transition.
- The first derivative of the intensity function, highlighting the points of local extrema corresponding to the edges in the image.
Differentiation and convolution

For a 2D function $F(x, y)$ the partial derivative along $x$ is:

$$\frac{\partial F(x, y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{F(x + \varepsilon, y) - F(x, y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

$$\frac{\partial F(x, y)}{\partial x} \approx \frac{F(x + 1, y) - F(x, y)}{1}$$

What would be the respective filters along $x$ and $y$ to implement the partial derivatives as a convolution?
Partial derivatives of an image

\( \frac{\partial F(x, y)}{\partial x} \)

\( \frac{\partial F(x, y)}{\partial y} \)

\(-1\) \hspace{1em} 1

\(-1\) \hspace{1em} 1
Alternative Finite-difference filters

Prewitt filter

\[ G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \]

Sobel filter

\[ G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \]

Sample Matlab code

```matlab
>> im = imread('lion.jpg');
>> h = fspecial('sobel');
>> outim = imfilter(double(im), h);
>> imagesc(outim);
>> colormap gray;
```
Image gradient

The gradient of an image: \( \nabla F = \left[ \frac{\partial F}{\partial x}, \frac{\partial F}{\partial y} \right] \)

The gradient points in the direction of fastest intensity change

\[ \nabla F = \left[ \frac{\partial F}{\partial x}, 0 \right] \]

\[ \nabla F = \left[ 0, \frac{\partial F}{\partial y} \right] \]

The gradient direction (orientation of edge normal) is given by:

\[ \theta = \tan^{-1} \left( \frac{\partial F}{\partial y} / \frac{\partial F}{\partial x} \right) \]

The edge strength is given by the gradient magnitude

\[ \| \nabla F \| = \sqrt{\left( \frac{\partial F}{\partial x} \right)^2 + \left( \frac{\partial F}{\partial y} \right)^2} \]
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

\[ \frac{\partial F(x)}{\partial x} \]

Where is the edge?
Solution: smooth first

Look for peaks in $\frac{\partial}{\partial x} (F \ast H)$

Where is the edge?
Alternative: combine derivative and smoothing filter

\[
\frac{\partial}{\partial x} (F * H) = F * \frac{\partial H}{\partial x}
\]

Differentiation property of convolution.
Derivative of Gaussian filter (along x)

\[(I \ast G) \ast H = I \ast (G \ast H)\]
Derivative of Gaussian filters

$x$-direction

$y$-direction
Laplacian of Gaussian

Consider \[ \frac{\partial^2}{\partial x^2} (F \ast H) = F \ast \frac{\partial^2 H}{\partial x^2} \]

Where is the edge? Zero-crossings of bottom graph
2D edge detection filters

\[ G = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}} \]

\[ \frac{\partial G}{\partial x} \]

\[ \nabla^2 G \]

- \( \nabla^2 \) is the Laplacian operator:

\[ \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \]
Summary on (linear) filters

• **Smoothing filter:**
  – has positive values
  – sums to 1 → preserve brightness of constant regions
  – removes “high-frequency” components: “low-pass” filter

• **Derivative filter:**
  – has **opposite signs** used to get high response in regions of high contrast
  – sums to 0 → no response in constant regions
  – highlights “high-frequency” components: “high-pass” filter
The Canny edge-detection algorithm (1986)

- Compute gradient of smoothed image in both directions
- Discard pixels whose gradient magnitude is below a certain threshold
- **Non-maximal suppression**: identify local maxima along gradient direction
The Canny edge-detection algorithm (1986)

Take a grayscale image. If not grayscale (i.e., RGB), convert it into a grayscale by replacing each pixel by the mean value of its R, G, B components.
The Canny edge-detection algorithm (1986)

Take a grayscale image. If not grayscale (i.e., RGB), convert it into a grayscale by replacing each pixel by the mean value of its R, G, B components.
The Canny edge-detection algorithm (1986)

Convolve the image $F$ with $x$ and $y$ derivatives of Gaussian filter

$$\frac{\partial F}{\partial x} = F \ast \frac{\partial G}{\partial x}$$

$$\frac{\partial F}{\partial y} = F \ast \frac{\partial G}{\partial y}$$

$$\|\nabla F\| = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2} : \text{Edge strength}$$
The Canny edge-detection algorithm (1986)

Threshold it (i.e., set to 0 all pixels whose value is below a given threshold)
The Canny edge-detection algorithm (1986)

Take local maximum along gradient direction

Thinning: non-maxima suppression (local-maxima detection) along edge direction
Summary (things to remember)

- Image filtering (definition, motivation, applications)
- Moving average
- Linear filters and formulation: box filter, Gaussian filter
- Boundary issues
- Non-linear filters
  - Median filter and its applications
- Edge detection
  - Derivating filters (Prewitt, Sobel)
  - Combined derivative and smoothing filters (deriv. of Gaussian)
  - Laplacian of Gaussian
  - Canny edge detector
- Readings: Ch. 3.2, 4.2.1 of Szeliski book
Understanding Check

Are you able to:

• Explain the differences between convolution and correlation?
• Explain the differences between a box filter and a Gaussian filter?
• Explain why should one increase the size of the kernel of a Gaussian filter if it is large (i.e. close to the size of the filter kernel?)
• Explain when would we need a median filter?
• Explain how to handle boundary issues?
• Explain the working principle of edge detection with a 1D signal?
• Explain how noise does affect this procedure?
• Explain the differential property of convolution?
• Show how to compute the first derivative of an image intensity function along x and y?
• Explain why the Laplacian of Gaussian operator is useful?
• List the properties of smoothing and derivative filters?
• Illustrate the Canny edge detection algorithm?
• Explain what non-maxima suppression is and how it is implemented?