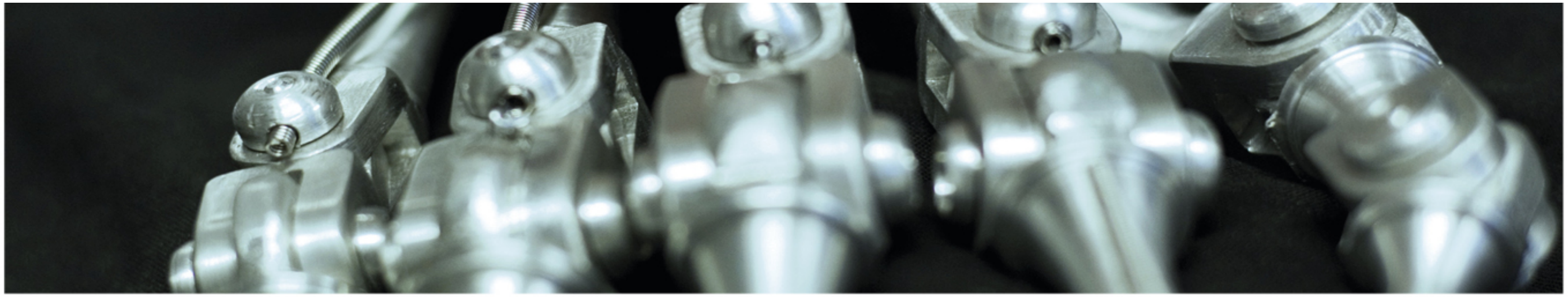


UiO : Department of Informatics
University of Oslo

INF 5860 Machine learning for image classification

23.1.18

Background in image convolution and filtering

Anne Solberg

Today

- Image filtering
- 2D convolution
- Edge detection filters

- Implementing convolution in python

- PURPOSE: give a short background to those without a background in image analysis
- Convolutional nets use convolution as the basic operation.

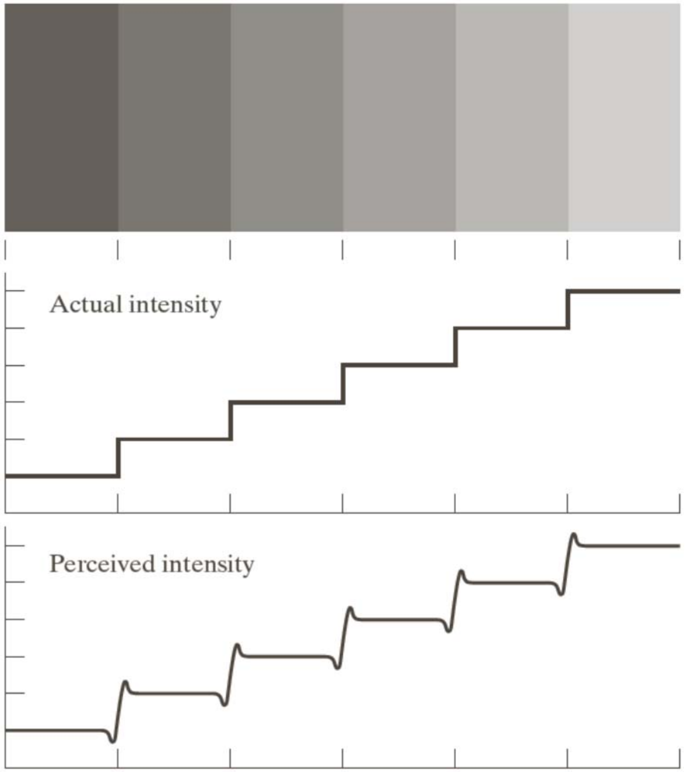
Properties of the human visual system

- We can see light intensities over a broad range
  - The largest is about **10^{10}** times higher than the lowest we can sense.

- We can only see a certain number of levels simultaneously,
  - About 50 different gray levels, but many more colors.

- When we focus on a different area, the eye adapts and we see local intensity differences.

Neural processes in the retina

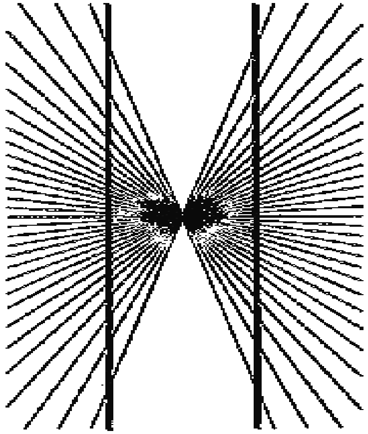


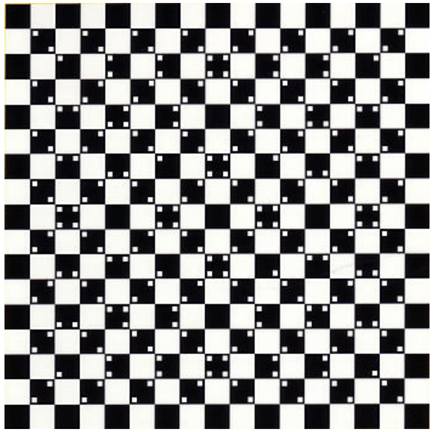
- Amplifies edges.
- Stimulating one part suspends other parts.
  - See one orientation at a time
- Increased contrast in edges between uniform regions
  - Called Mach band

Optical illusions

- Illusional contours

Straight or curved lines





- Multistable images

Simultaneous contrast

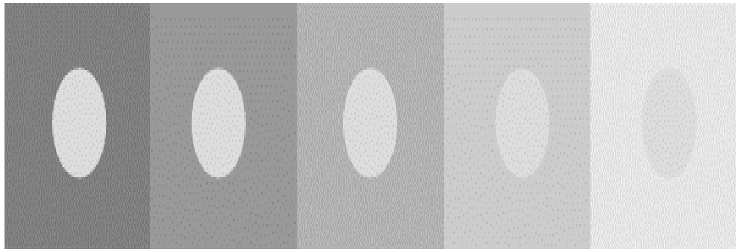


Image filtering

- One of the most used operations on images
- A filter kernel is applied either in the image domain or 2D Fourier domain.
- Applications:
  - Image enhancement
  - Image restoration
  - Image analysis – preprocessing
- Used to:
  - Reduce noise
  - Enhance sharpness
  - Detect edges and other structures
  - Detect objects
  - Extract texture information

Spatial filtering

- A filter is given as a matrix:

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

- The size of the matrix and the coefficients decides the result.

2-D convolution

- Output image: g. Input image f.

$$g(x,y) = \sum_{j=x-m2}^{x+m2} \sum_{k=y-n2}^{y+n2} w(x-j, y-k) f(j,k)$$

- w is a $m \times n$ filter of size $m=2m_2+1$, $n=2n_2+1$, $m_2=m//2$, $n_2=n//2$
- *m* and *n* usually odd.
- We will use square filters m=n
- Output image: weighted sum of input image pixels surrounding pixel (x,y). Weights:*w(j,k).*
- This operation is done for every pixel in the image

Step 1: convolution: rotate the image 180 degrees

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

In this case, nothing changes

Rotation in Python

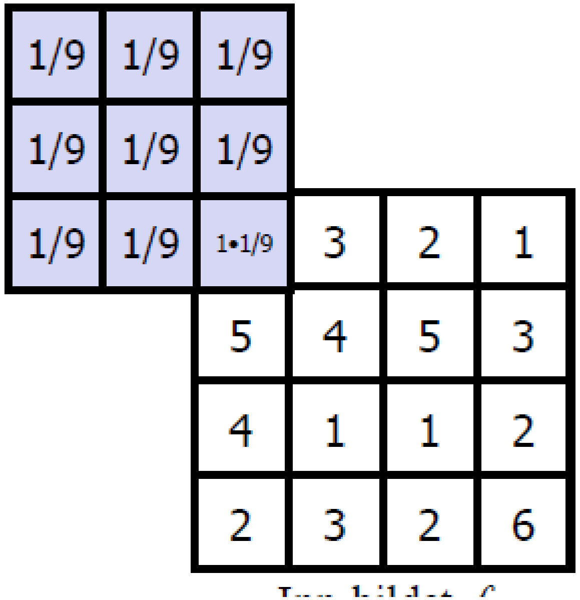
In [9]:
```
import numpy as np
kernel = [[1,2,1], [0,0,0],[-1,-2,-1]]
kernel = np.array(kernel)
print(kernel)
```

```
[[ 1  2  1]
 [ 0  0  0]
 [-1 -2 -1]]
```

In [10]:
```
kernel = kernel[::-1, ::-1]
print(kernel)
```

```
[[-1 -2 -1]
 [ 0  0  0]
 [ 1  2  1]]
```

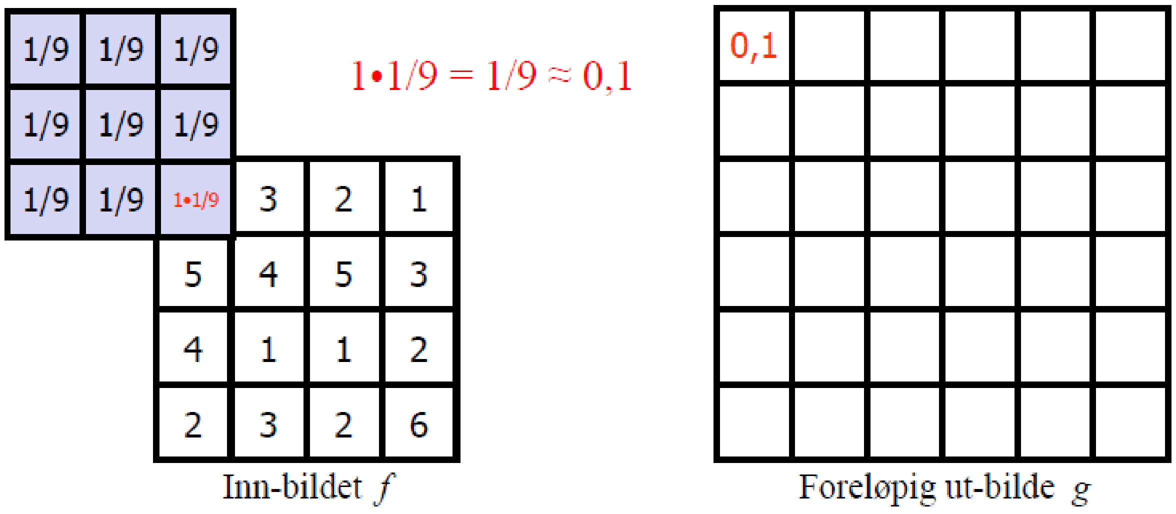
Step 2: Iterate over all locations where the filter overlaps the image



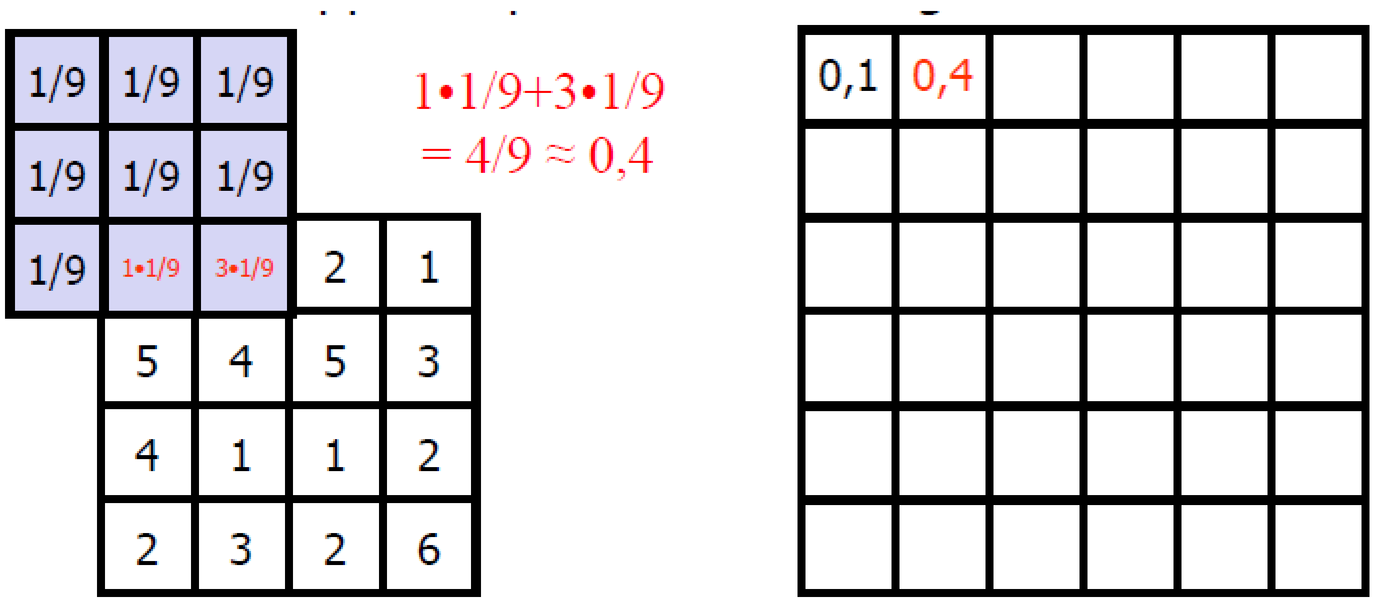
Multiply the image and the mask
Compute the result for location (x,y)

Step 3

Multiply the filter and the image coefficients, and sum to get the result for ONE pixel in the output image



Repeat for next output pixel



Inn-bildet f Foreløpig ut-bilde g

Repeat for next output pixel

$$3 \cdot 1/9 + 2 \cdot 1/9 + 1 \cdot 1/9 + 4 \cdot 1/9 + 5 \cdot 1/9 + 3 \cdot 1/9 + 1 \cdot 1/9 + 1 \cdot 1/9 + 2 \cdot 1/9 = 22/9 \approx 2{,}4$$

1	3•1/9	2•1/9	1•1/9
5	4•1/9	5•1/9	3•1/9
4	1•1/9	1•1/9	2•1/9
2	3	2	6

Inn-bildet f

0,1	0,4	0,7	0,7	0,3	0,1
0,7	1,4	2,2	2,0	1,2	0,4
1,1	2,0	2,9	2,4		

Foreløpig ut-bilde g

The solution is

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

3x3-middelverdifilteret

*

1	3	2	1
5	4	5	3
4	1	1	2
2	3	2	6

Inn-bildet f

=

0,1	0,4	0,7	0,7	0,3	0,1
0,7	1,4	2,2	2,0	1,2	0,4
1,1	2,0	2,9	2,4	1,6	0,7
1,2	2,1	3,0	3,0	2,1	1,2
0,7	1,1	1,4	1,7	1,2	0,9
0,2	0,6	0,8	1,2	0,9	0,7

Ut-bildet g

- To get the result for one pixel in the output image, we compute

$$g(x,y) = \sum_{j=x-m2}^{x+m2} \sum_{k=y-n2}^{y+n2} w(j,k) f(x-j, y-k)$$

- Straightforward implementation: use two for-loops over j and k to compute g(x,y)
- To get the result for ALL pixels in the image with size (X,Y), we would need two additional for-loops over all pixels:

for x in range(n2: X-n2):

for y in range(m2:Y-n2):

IN this case we compute the result only where the filter completely overlaps the image

$$g(x,y) = \sum_{j=x-m2}^{x+m2} \sum_{k=y-n2}^{y+n2} w(j,k) f(x-j, y-k)$$

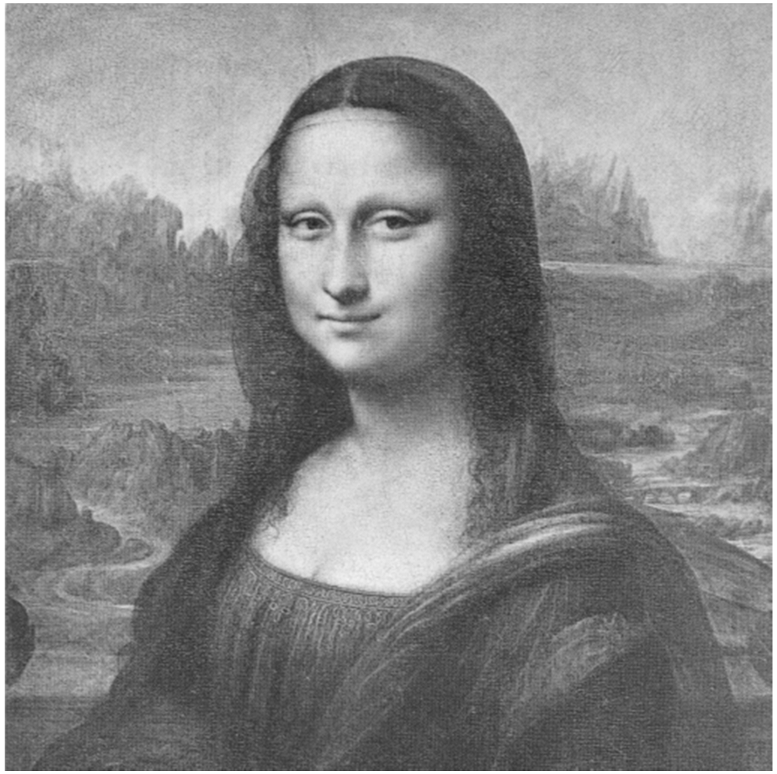
The inner sum (formula above) can be more efficiently implemented using np.sum.

For color images: add an outer for loop over the bands (c=1:3 for RGB images)

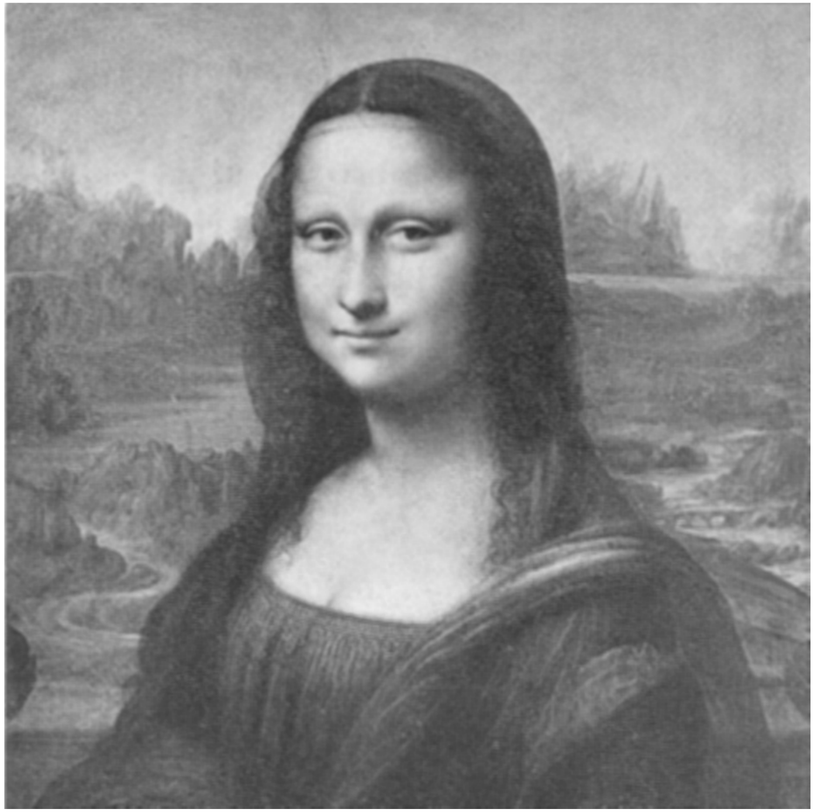
Mean filters

- 3×3: $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$
- 5×5: $\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$
- 7×7: $\frac{1}{49}\begin{bmatrix}1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\end{bmatrix}$

- Scale the result by the sum of the coefficients to keep the range of the input image



Original



Filtered 3x3

9x9

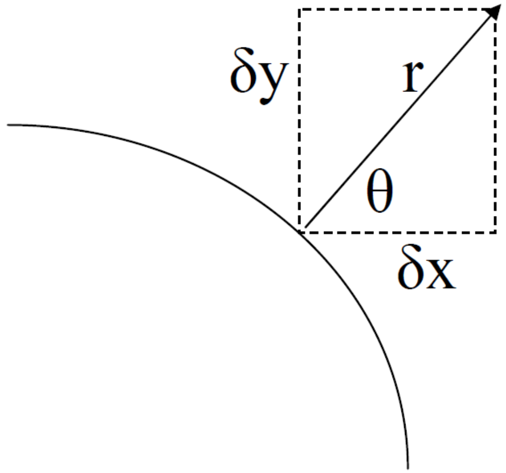
25x25

Gradients

- Gradient of *F* along r in direction θ

$$\frac{\partial F}{\partial r} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial r}$$

$$\frac{\partial F}{\partial r} = \frac{\partial F}{\partial x}\cos\theta + \frac{\partial F}{\partial y}\sin\theta$$



- Largest gradient when $\frac{\partial}{\partial \theta}\left(\frac{\partial F}{\partial r}\right) = 0$

- This is the angle θ_g where

$$-\frac{\partial F}{\partial x}\sin\theta_g + \frac{\partial F}{\partial y}\cos\theta_g = 0 \Leftrightarrow \frac{\partial F}{\partial y}\cos\theta_g = \frac{\partial F}{\partial x}\sin\theta_g$$

- $g_x = \delta F/\delta x$ and $g_y = \delta F/\delta x$ are the horisontal and vertical components of the gradient.
- Gradient points in the direction where the function has the largest increase.

Gradient magnitude and direction

- Gradient direction:

$$\frac{g_y}{g_x} = \frac{\sin\theta_g}{\cos\theta_g} = \tan\theta_g \qquad \theta_g = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

- Gradient magnitude

$$\left(\frac{\partial F}{\partial r}\right)_{\max} = \left[g_x^2 + g_y^2\right]^{1/2}$$

Gradient-filters

These filter do gradient computation in one direction, and smoothing in the other direction

- Prewitt-operator

$$H_x(i,j)=\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix} H_y(i,j)=\begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix}$$

- Sobel-operator

$$H_x(i,j)=\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} H_y(i,j)=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

- On this slide, the filters are denoted h, not w

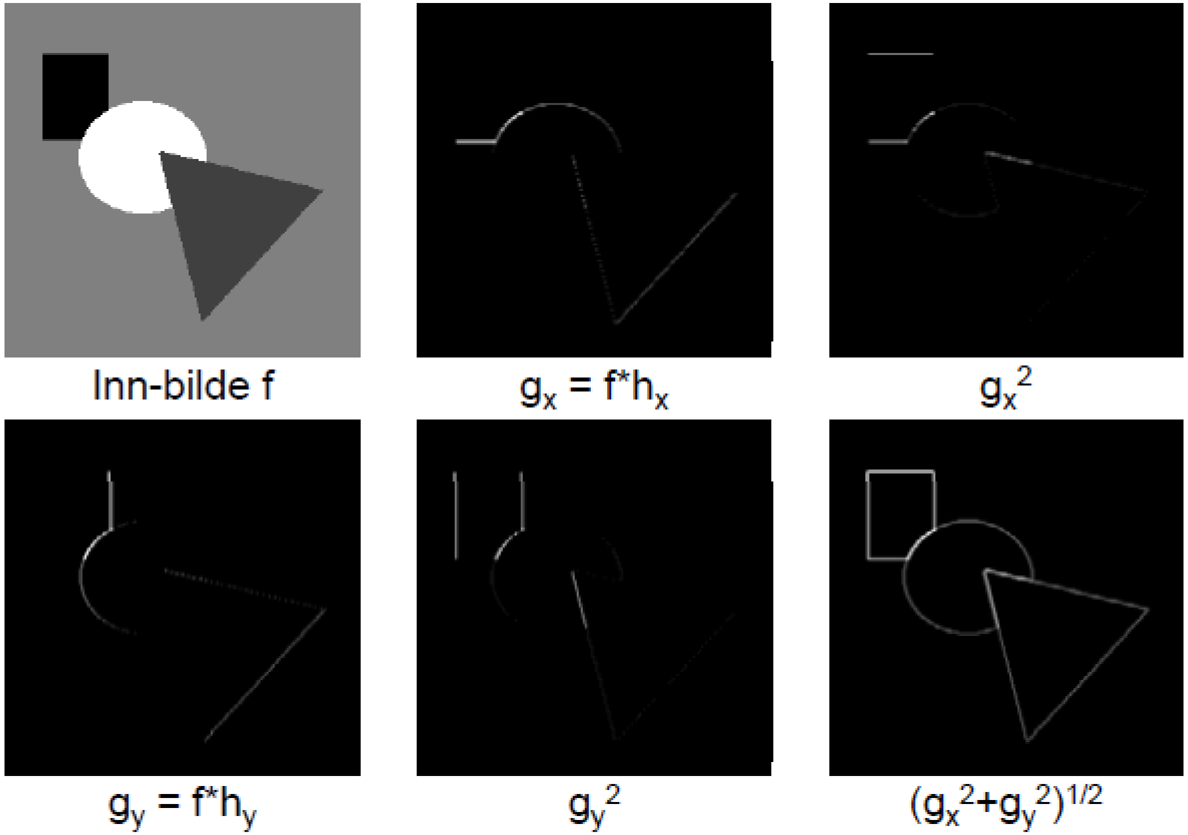
Computing the gradients

- Find the horisontal edges by convolving with hx:
  - Compute $g_x = h_x * f$
- Find the verdical edges by convolving with hy:
  - Compute: $g_y = h_y * f$
- Compute gradient magniture and direction:

$$M(i,j) = \sqrt{g_x^2(i,j) + g_y^2(i,j)}$$ **Gradient-magnitude**

$$\theta(i,j) = \tan^{-1}\left(\frac{g_y(i,j)}{g_x(i,j)}\right)$$ **Gradient-retning**

Gradient example



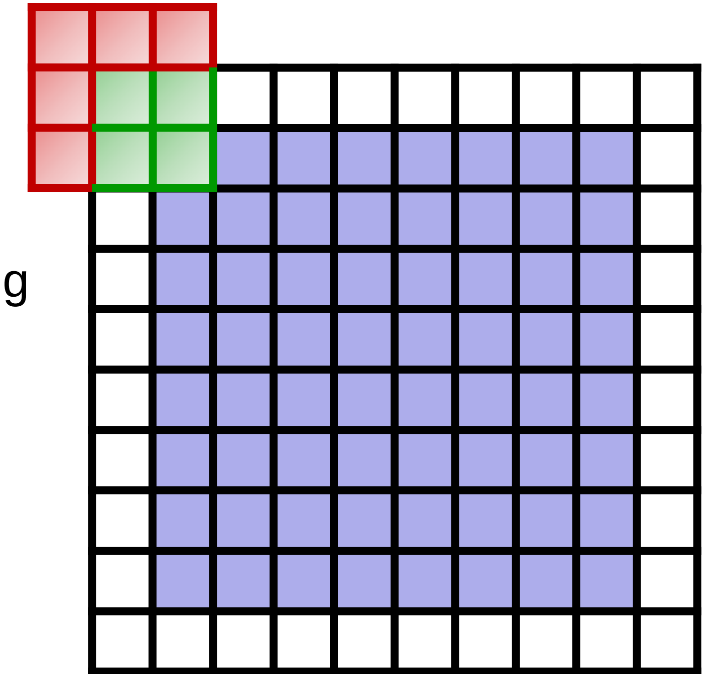
Size of the resulting image

Alt.1. Compute the result only where the filter fits inside the input image.

- Assume the image has X x Y pixels and that the filter has size m x n (m,n odd numbers)
- Output size:
  (M-m+1)x(N-n+1)
  - 3x3-filter: (M-2)x(N-2)
  - 5x5-filter: (M-4)x(N-4)

Alt.2. Keep the size of the input image

- Will normally be used for deep learning
- Common for image filtering also
- Special considerations at the border
  - Zero-pad the input image
  - Or, change the filter size at the boundary



Filtrering II: Correlation

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}h(s,t)f(x+s,y+t)$$

- Difference from convolution: **Pluss replaces minus**.
  - No rotation!

- Normally applied in convolutional nets as the net itself will estimate the filter coefficients (and they could be estimated with the rotation included)

Like to learn more:

- See lecture notes from INF2310 (notes in English)
  - http://www.uio.no/studier/emner/matnat/ifi/INF2310/v17/undervisningsmateriale/slides_inf2310_s17_week06.pdf
  - http://www.uio.no/studier/emner/matnat/ifi/INF2310/v17/undervisningsmateriale/slides_inf2310_s17_week07.pdf

This weeks exercise

- Task: complete the notebooks:
  - math_operations.ipynb
  - indexing.ipynb
  - convolution.ipynb

- See lecture notes on how to start jupyter notebooks
- https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/

Useful links:

- http://cs231n.github.io/python-numpy-tutorial/
- http://cs231n.github.io/ipython-tutorial/