IMAGE FILTRATION II

Ole-Johan Skrede

01.03.2017

INF2310 - Digital Image Processing

Department of Informatics
The Faculty of Mathematics and Natural Sciences
University of Oslo

After original slides by Fritz Albregtsen

TODAY'S LECTURE

- High-pass filtering
- Image improvement and edge detection
- Gradient filters
- The Laplace and the LoG filter
- Canny's edge detection
- Sections in Gonzales & Woods:
  - 3.6: Sharpening Spatial Filters
  - 10.2: Point, Line, and Edge Detection
    - 10.2.1: Background
    - 10.2.2: Detection of Isolated Points
    - 10.2.3: Line Detection
    - 10.2.4: Edge Models
    - 10.2.5: Basic Edge Detection
    - 10.2.6: More Advanced Techniques for Edge Detection

HIGH PASS FILTERS

- Filters that let through high frequencies, and damps or block low frequencies.
- Effect:
  - Damps slow variations, e.g. background
  - Emphasizes sharp edges, lines, and details
- Is typically used to ”improve” the sharpness, and to detect edges.
- What happens with noise?
- Noise is amplified.

HIGH PASS FILTERING WITH CONVOLUTION

- The sum of the weights in the convolution kernel is typically 0.
- This means that the sum of the pixel values in the convolved image is also 0 (if we convolve in `full` mode).
- Which again mean that the convolved image contain both positive and negative values.
- If you need an unsigned out-image, add some value and scale the image to the desired range.

ZERO-SUM RESULT

Let w be some $P \times Q$ (with odd P and Q) filter such that

$$\sum_{s=-S}^{S} \sum_{t=-T}^{T} w[s,t] = 0,$$

where $S = (P-1)/2$ and $T = (Q-1)/2$.
Furthermore, let f be some $M \times N$ image. Then, if we convolve using a `full` mode (see lecture slides for week 6), we see that the sum of all elements in the result g is zero.

$$\begin{aligned}
\sum_{x=-S}^{M-1+S} \sum_{y=-T}^{N-1+T} (f * w)[x,y] &= \sum_{x=-S}^{M-1+S} \sum_{y=-T}^{N-1+T} \sum_{s=-S}^{S} \sum_{t=-T}^{T} f[x-s, y-t] w[s,t] \\
&= \sum_{s=-S}^{S} \sum_{t=-T}^{T} \sum_{x=-S}^{M-1+S} \sum_{y=-T}^{N-1+T} f[x-s, y-t] w[s,t] \\
&= \sum_{s=-S}^{S} \sum_{t=-T}^{T} \sum_{x=s}^{M-1+s} \sum_{y=t}^{N-1+t} f[x-s, y-t] w[s,t] \\
&= \sum_{s=-S}^{S} \sum_{t=-T}^{T} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n] w[s,t] \\
&= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left[f[m,n] \left(\sum_{s=-S}^{S} \sum_{t=-T}^{T} w[s,t] \right) \right] \\
&= 0.
\end{aligned}$$

ZERO SUM IMPLEMENTATION

Implementation in python

```python
import numpy as np
from scipy import signal

M = 4
N = 5

f = np.random.randint(low=0, high=255,
    size=(M, N))
w = np.array([[-1, -1, -1],
              [-1, 8, -1],
              [-1, -1, -1]])
g = signal.convolve2d(f, w, mode='full')

print('f = \n', f)
print('w = \n', w)
print('g = \n', g)
print('sum g = ', np.sum(g))

```

Print of result

```
f =
 [[104 162 238  20  78]
 [ 44 106  36 138 244]
 [102 110  17 189   8]
 [245  90  93  21 156]]
w =
 [[-1 -1 -1]
 [-1  8 -1]
 [-1 -1 -1]]
g =
 [[-104 -266 -504 -420 -336  -98  -78]
 [-148  520  768 1442 -574  222 -322]
 [-250 -232   35 -692  274 1519 -330]
 [-391  221  147 -647  799 -684 -408]
 [-347 1658  153  317 -295 1030 -164]
 [-245 -335 -428 -204 -270 -177 -156]]
sum g =  0

```

- An example of a high-pass filter

$$w = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- This filter can for instance be used to detect isolated points in an image f:
  - Compute the convolution $g = f * w$.
  - Isolated points will be highlighted as they yield a high absolute response.
  - For some threshold $T > 0$, we can detect points (x, y) that satisfy $|g(x, y)| > T$.

EXAMPLE — POINT DETECTION

1	1	1
1	−8	1
1	1	1

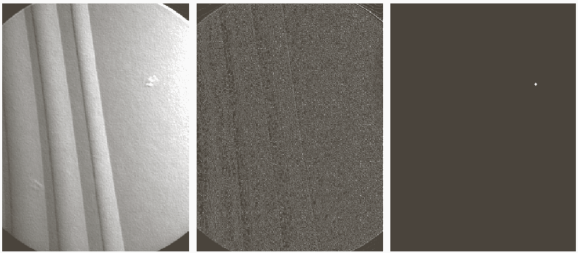


Figure 1: Point detection with the top mask. Left: X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. Middle: Result of convolving the mask with the image. Right: Result after thresholding the convolved image.

EXAMPLE — SHIP DETECTION

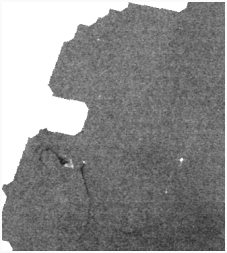


Figure 2: Radar image of an area of the sea.

- The small bright points are ships
- The previously shown filter will result in high response for the ships, but almost as high response at edges and corners.
- Better with a larger filter of the same type.

$$w = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 4 & 4 & 4 & -1 & -1 \\ -1 & -1 & 4 & 8 & 4 & -1 & -1 \\ -1 & -1 & 4 & 4 & 4 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

IMAGE IMPROVEMENT WITH HIGH-PASS FILTERING

- This filter can also be used for image improvement

$$w = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- Idea:
  - The filtration will detect the beginning and end of edges
  - Other areas will give responses close to zero
  - We can then add the filtered image to the original to make edges more visible, which makes the image seems sharper.

IMAGE IMPROVEMENT WITH HIGH-PASS FILTERING

- The convolution mask we have investigated can be rewritten as ”high-pass = k * (original - low-pass)”, where $k = 9$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = 9 \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

- With this, our image improvement strategy can be viewed as:
  1. Low pass filtering
  2. Subtract the low-pass filtered image from the original
  3. Add k times the difference from point 2. to the original image.
- The described method is called *highboost-filtering*.

UNSHARP MASKING AND HIGHBOOST-FILTERING

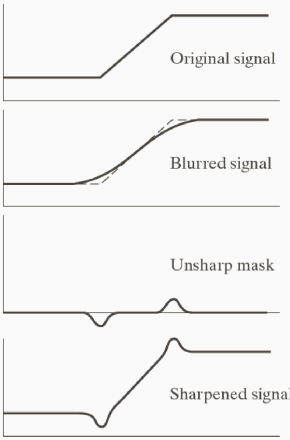
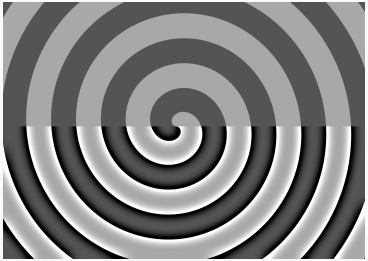


Figure 3: Schematic illustration of image sharpening on a 1D ramp.

- Given an image.
- Blur the image with a low-pass filter.
- Compute the difference $original - blurred$
- $sharpened = original + k(original - blurred)$
- k is a positive constant.
  - $k = 1$: Unsharp masking
  - $k > 1$: Highboost-filtering

EXAMPLES



EDGE-DETECTION

MOTIVATION

- There exists much image information at the edges of objects in an image.
- Biological visual systems are based on edge-detection.
- Such systems often works in parallel and/or sequential:
  - Local areas is treated independently
  - Local results can be dependent on earlier results

IDEAL EDGES AND LINES

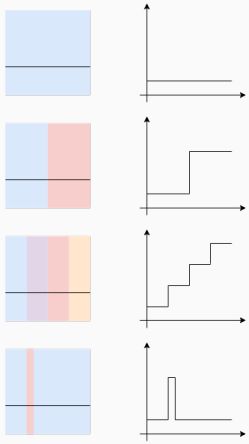


Figure 4: Edge illustrations and cross sections.

Homogeneous plane An area where all pixel values are equal.

Edge The transition between two areas with different color value

- The division can be between two pixels
- But often the first pixel on one of the sides

Line A line is two consecutive edges in the opposite ”direction”

SOME EDGE TYPES

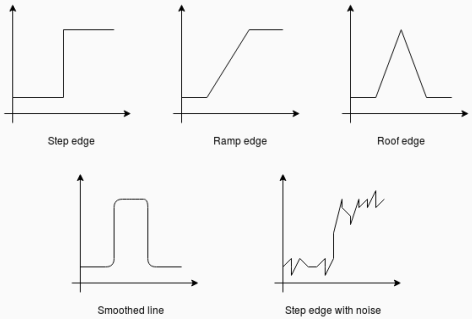


Figure 5: Some edge types.

DISCLAIMER

In order to arrive at some of the results in this lecture, we need to use some math. Depending on your previous experience, it may seem a bit heavy. We will touch upon

- Polar coordinates
- Calculus:
  - Derivatives in one and two variables
  - Chain rule in differentiation
  - Finding critical points (minima and maxima)

I will assume that this is known, and since this is not a math course, I will not spend so much time explaining it here. I strongly encourage you to read up on these topics if you do not understand the next couple of slides. Also feel free to ask me, and I can try to explain.

- An edge is characterized with a change in intensity
- For a function f, the slope of the tangent line at some point a on the x-axis can be approximated by the fraction

$$\frac{f(a+h) - f(a)}{(a+h) - a}$$

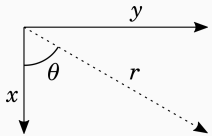
- The limit when the distance h approaches zero is defined to be the *derivative* of f at a

$$\frac{\mathrm{d}f}{\mathrm{d}x}(a) := \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}.$$

- If this limit exists, f is said to be *differentiable* at a.
- The derivative is in general *not defined everywhere* for discrete functions. We can, however, approximate it with $h \geq 1$ in the definition.

- A digital image is a bivariate discrete function
- A continuous function f in two variables x and y can be differentiated wrt. both
- We denote the so-called *partial derivatives* of f wrt. x and y as $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$, respectively.
- A vector of a function's partial derivatives is called the *gradient*, and we denote it with the *nabla* symbol

$$\nabla f := \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$



A function f will have values $f(x, y)$ in terms of variables (x, y) in a cartesian coordinate system. We can also find the values $f(r, \theta)$ in a *polar coordinate* system. Here

$$x = r\cos\theta, \text{ and } y = r\sin\theta.$$

We can therefore find the derivative of f wrt. r (called the directional derivative) using the chain rule

$$\begin{aligned}\frac{\partial f}{\partial r} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} \\ &= g_x\frac{\partial x}{\partial r} + g_y\frac{\partial y}{\partial r} \\ &= g_x\cos\theta + g_y\sin\theta.\end{aligned}$$

- We want to find the angle θ_g of steepest ascent.
- This is the angle where the directional derivative is largest.
- This is again the angle where the derivative of the directional derivative wrt. θ is equal to zero (a stationary or critical point), and the second order derivative of the directional derivative wrt. θ is smaller than zero. That is

$$\frac{\partial}{\partial\theta}\left(\frac{\partial f}{\partial r}\right)\bigg|_{\theta=\theta_g} = 0$$

and

$$\frac{\partial^2}{\partial\theta^2}\left(\frac{\partial f}{\partial r}\right)\bigg|_{\theta=\theta_g} < 0.$$

Remember our earlier result

$$\frac{\partial f}{\partial r} = g_x \cos\theta + g_y \sin\theta,$$

Therefore

$$\frac{\partial}{\partial \theta}\left(\frac{\partial f}{\partial r}\right) = -g_x \sin\theta + g_y \cos\theta.$$

Defining θ_g to be the angle θ where this is equal to zero yields

$$-g_x \sin\theta_g + g_y \cos\theta_g = 0$$

or

$$\theta_g = \arctan\left(\frac{g_y}{g_x}\right).$$

The expression of the second order derivative wrt theta is

$$\frac{\partial^2}{\partial\theta^2}\left(\frac{\partial f}{\partial r}\right) = -g_x\cos\theta - g_y\sin\theta.$$

For the next derivation, we will use the result from trigonometry

$$\sin\left(\arctan(x)\right) = \frac{x}{\sqrt{1+x^2}}, \text{ and } \cos\left(\arctan(x)\right) = \frac{1}{\sqrt{1+x^2}}.$$

Using this, and evaluating the second derivative at the stationary angle θ_g, we get

$$\begin{aligned}\frac{\partial^2}{\partial\theta^2}\left(\frac{\partial f}{\partial r}\right)(\theta_g) &= -g_x\frac{1}{\sqrt{1+\left(\frac{g_y}{g_x}\right)^2}} - g_y\frac{\left(\frac{g_y}{g_x}\right)}{\sqrt{1+\left(\frac{g_y}{g_x}\right)^2}} \\ &= -\sqrt{g_x^2+g_y^2}\end{aligned}$$

which is negative. This means that our stationary point θ_g, is a maximum.

By evaluating the directional derivative at our stationary point, we get

$$\begin{aligned}\frac{\partial f}{\partial r}(\theta_g) &= g_x \cos\theta_g + g_y \sin\theta_g \\ &= g_x \frac{1}{\sqrt{1+\left(\frac{g_y}{g_x}\right)^2}} + g_y \frac{\left(\frac{g_y}{g_x}\right)}{\sqrt{1+\left(\frac{g_y}{g_x}\right)^2}} \\ &= \sqrt{g_x^2 + g_y^2}\end{aligned}$$

SUMMARY — GRADIENT ANGLE AND GRADIENT MAGNITUDE

- The gradient points in the direction where the function ascends the most. We call this the *gradient angle*

$$\theta_g = \arctan\left(\frac{g_y}{g_x}\right)$$

- The ascent magnitude at this point is called the *gradient magnitude* and is

$$\begin{aligned} |\nabla f| &= \frac{\partial f}{\partial r}(\theta_g) \\ &= \sqrt{g_x^2 + g_y^2}. \end{aligned}$$

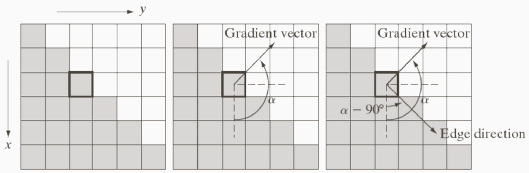
- Where

$$g_x = \frac{\partial f}{\partial x}, \text{ and } g_y = \frac{\partial f}{\partial y}$$

are the components of the gradient of f, ∇f.

EDGE AND GRADIENT DIRECTION

The gradient direction is the direction where the function increases the most. This is also the direction orthogonal to the edge direction.



DIGITAL GRADIENT APPROXIMATIONS

- We will approximate the gradient at some pixel (i, j) by the difference between pixels in the neighbourhood.
- We can do this by using two convolution filters (h_x, and h_y) that approximates the two components of the gradient operator.
- The convolution result $g_x[i, j]$ and $g_y[i, j]$ will then approximate the gradient at (i, j) in the vertical and horizontal direction, respectively.
- In the next slides, we will introduce several different gradient operators. Note that we write convolution filters, while G&W use correlation filters, and they are therefore rotated 180 degrees compared to ours.

1D ASYMMETRICAL GRADIENT OPERATORS

- With $g_x = f * h_x$ and $g_y = f * h_y$, where

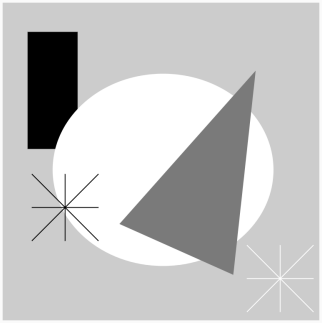
$$h_x = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \quad h_y = \begin{bmatrix} 1 & -1 & 0 \end{bmatrix},$$

  we get

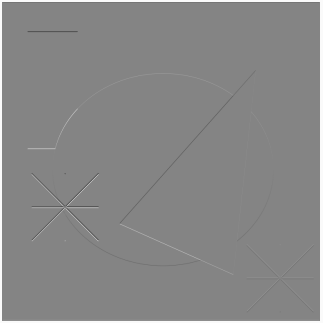
$$g_x[i,j] = f[i+1,j] - f[i,j]$$
$$g_y[i,j] = f[i,j+1] - f[i,j]$$

- The definition is such that the gradient components are positive for an edge where the intensity increases downwards and from the left to the right in the image.
- One problem with this operator is that the gradient-approximations refers to a point in between two pixels, and a different point for the x- and y-component.

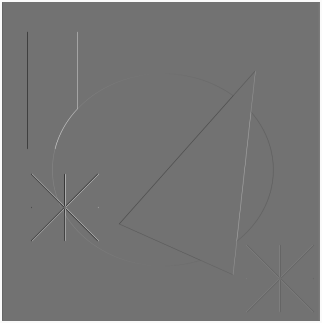
1D ASYMMETRICAL GRADIENT OPERATORS — DIRECTIONS



(a) Shapes



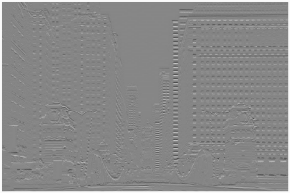
(b) g_x



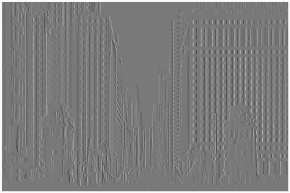
(c) g_y

1D ASYMMETRICAL GRADIENT OPERATORS — LARGER EXAMPLE

(a) Chicago buildings



(b) g_x



(c) g_y

1D SYMMETRICAL GRADIENT OPERATORS

- With $g_x = f * h_x$ and $g_y = f * h_y$, where

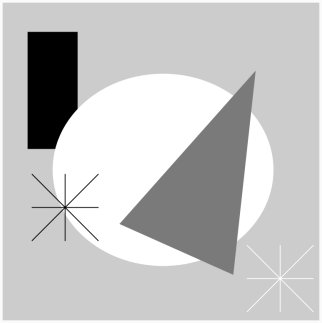
$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \quad h_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix},$$

  we get

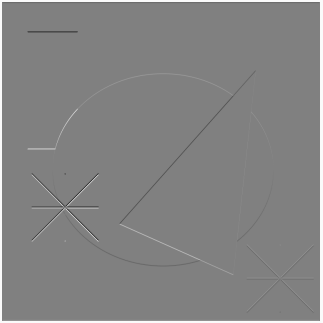
$$g_x[i,j] = f[i+1,j] - f[i-1,j]$$
$$g_y[i,j] = f[i,j+1] - f[i,j-1]$$

- This gives the derivative at a the pixel (i,j), for both the x-component and the y-component.
- One problem is that the operators are sensitive to noise.

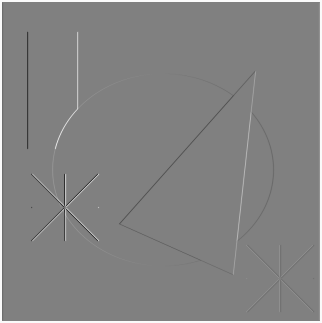
1D SYMMETRICAL GRADIENT OPERATORS — DIRECTIONS



(a) Shapes



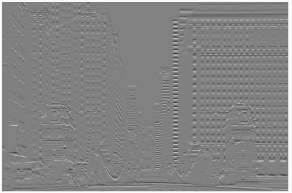
(b) g_x



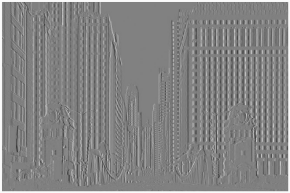
(c) g_y

SYMMETRICAL GRADIENT OPERATORS — LARGER EXAMPLE

(a) Chicago buildings



(b) g_x



(c) g_y

Increasing noise robustness: 2D gradient operators

- If we first apply a smoothing filter to reduce the noise.
- Then apply the desired gradient filter.
- We can utilize the separability of filters, and create new more noise-robust filters.

PREWITT OPERATOR

$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix},$$

$$h_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}.$$

$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},$$

$$h_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}.$$

$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & \sqrt{2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix},$$

$$h_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}.$$

Gradient operators can be made more robust to noise by adding more low-pass filtering. One example is the 5×5 Sobel operators (remember to utilize the separability when implementing them).

$$h_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 2 & 8 & 12 & 8 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix},$$

$$h_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 & -2 & -1 \\ 4 & 8 & 0 & -8 & -4 \\ 6 & 12 & 0 & -12 & -6 \\ 4 & 8 & 0 & -8 & -4 \\ 1 & 2 & 0 & -2 & -1 \end{bmatrix}.$$

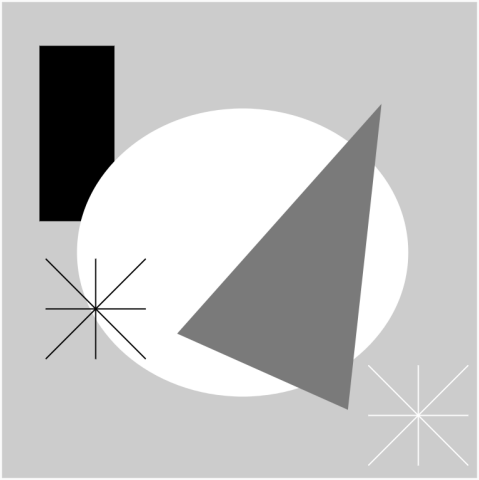
GRADIENT OPERATOR TRAITS

- We can make the gradient operators less sensitive to noise by low-pass filtering in one direction and approximate the derivative in the orthogonal direction.
- The Prewitt operator is more sensitive to horizontal and vertical edges than to diagonal edges.
- The Frei-Chen operator has equal sensitivity.
- The Sobel operator is more sensitive to diagonal edges than to horizontal and vertical edges.

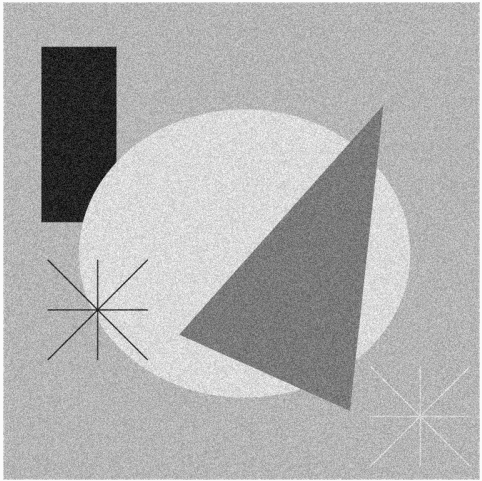
GRADIENT IMAGE IMPLEMENTATION

- For an image f, a vertical gradient filter h_x and a horizontal gradient filter h_y, we can compute:
- The vertical gradient $g_x[i,j] = (f * h_x)[i,j]$.
- The horizontal gradient $g_y[i,j] = (f * h_y)[i,j]$.
- The gradient direction $\theta[i,j] = \arctan\left(\frac{g_y[i,j]}{g_x[i,j]}\right)$.
- The gradient magnitude $|\nabla f| = \sqrt{g_x^2[i,j] + g_y^2[i,j]}$.
- Note that the gradients can contain negative values, so consider scaling them before displaying them.
- Note also that the gradient direction is in radians.

TEST IMAGE

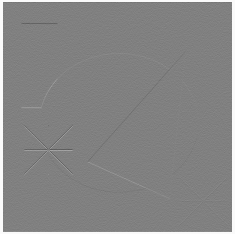


(a) Shapes

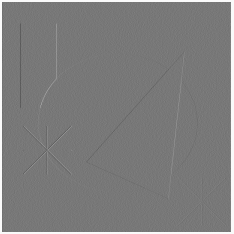


(b) Shapes and added white noise

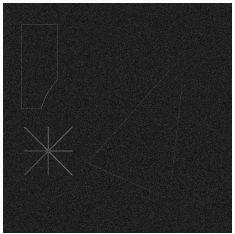
1D ASYMMETRICAL GRADIENT OPERATORS



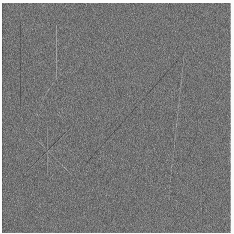
(a) g_x



(b) g_y

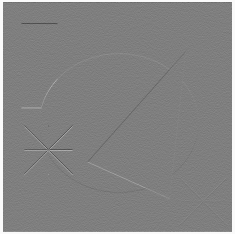


(c) $|\nabla f|$

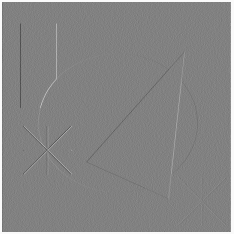


(d) θ

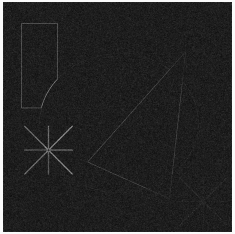
1D SYMMETRICAL GRADIENT OPERATORS



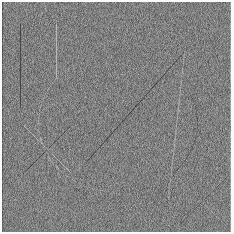
(a) g_x



(b) g_y

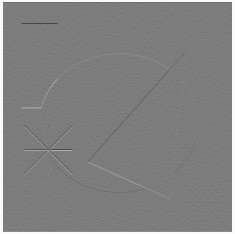


(c) $|\nabla f|$

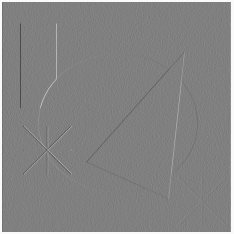


(d) θ

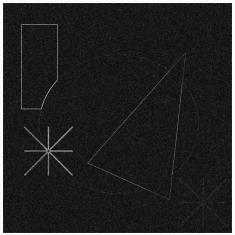
PREWITT GRADIENT OPERATORS



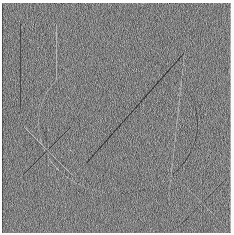
(a) g_x



(b) g_y

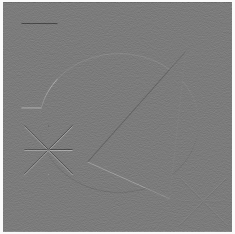


(c) $|\nabla f|$

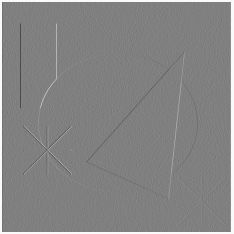


(d) θ

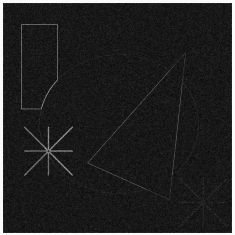
SOBEL GRADIENT OPERATORS



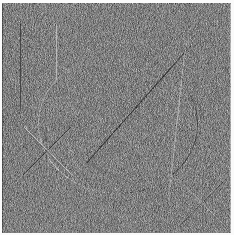
(a) g_x



(b) g_y

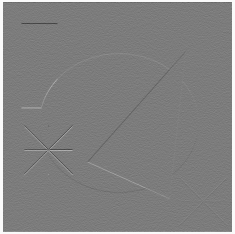


(c) $|\nabla f|$

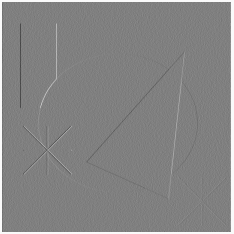


(d) θ

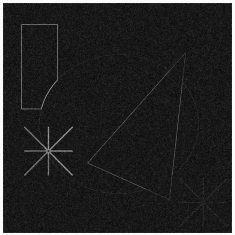
FREI-CHEN GRADIENT OPERATORS



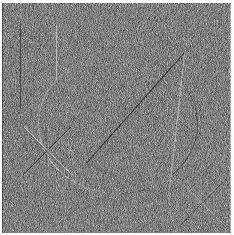
(a) g_x



(b) g_y

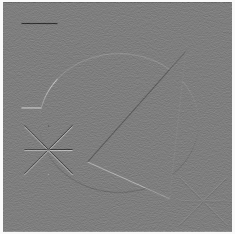


(c) $|\nabla f|$

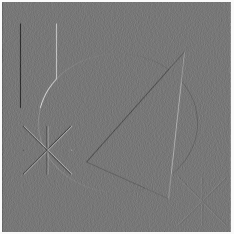


(d) θ

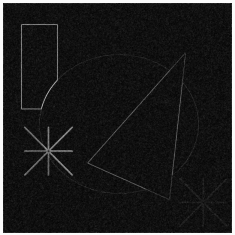
5x5 Sobel gradient operators



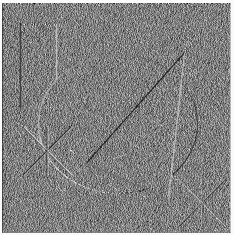
(a) g_x



(b) g_y



(c) $|\nabla f|$

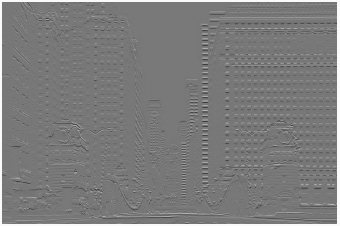


(d) θ

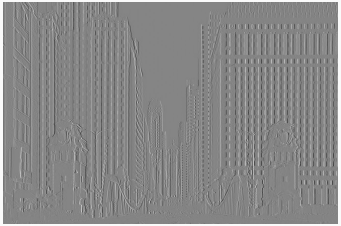
TEST IMAGE

Figure 17: Buildings

1D ASYMMETRICAL GRADIENT OPERATORS

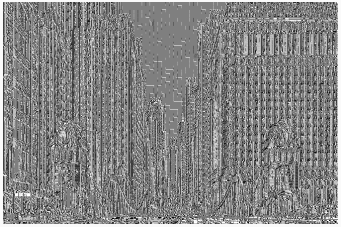


(a) g_x



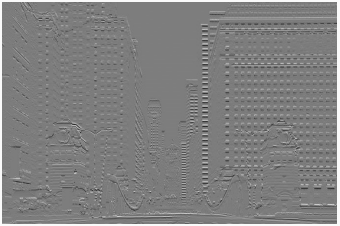
(b) g_y

(c) $|\nabla f|$

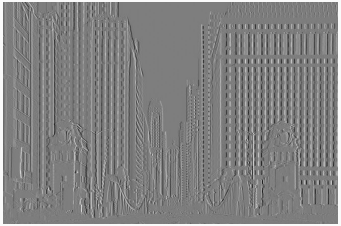


(d) θ

1D SYMMETRICAL GRADIENT OPERATORS

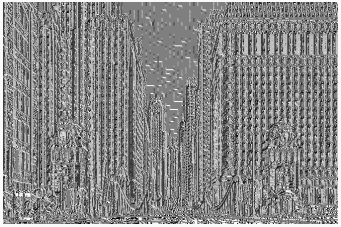


(a) g_x



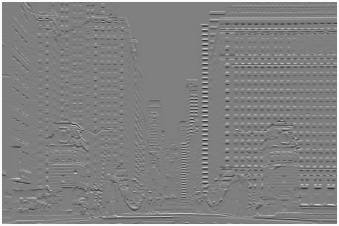
(b) g_y

(c) $|\nabla f|$

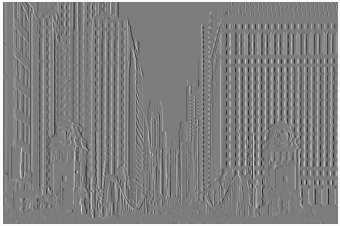


(d) θ

PREWITT GRADIENT OPERATORS

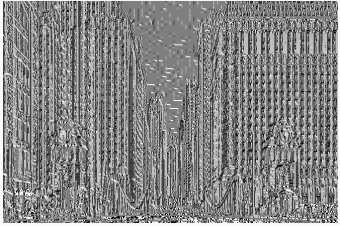


(a) g_x



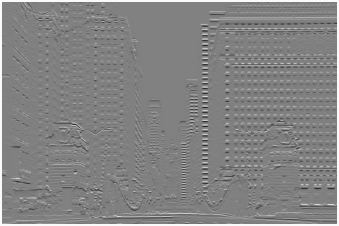
(b) g_y

(c) $|\nabla f|$

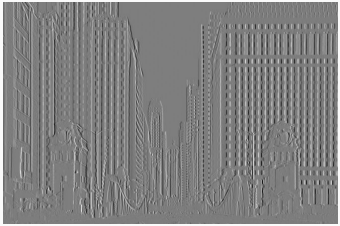


(d) θ

SOBEL GRADIENT OPERATORS

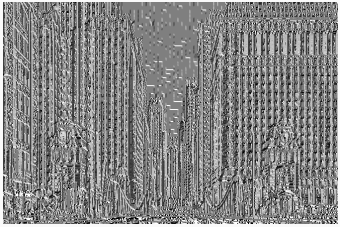


(a) g_x



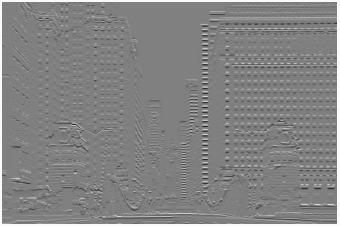
(b) g_y

(c) $|\nabla f|$

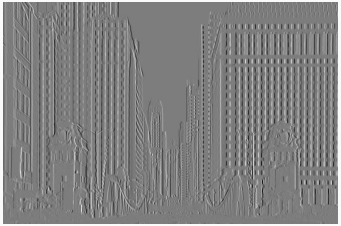


(d) θ

FREI-CHEN GRADIENT OPERATORS

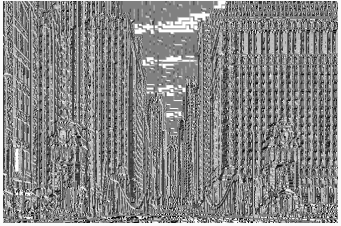


(a) g_x



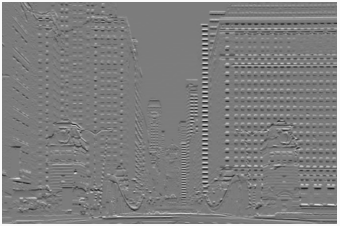
(b) g_y

(c) $|\nabla f|$

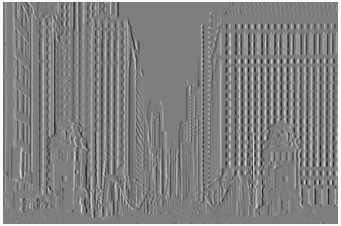


(d) θ

5x5 Sobel gradient operators

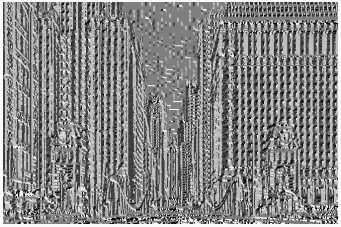


(a) g_x



(b) g_y

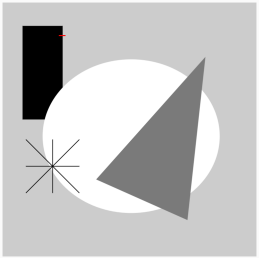
(c) $|\nabla f|$

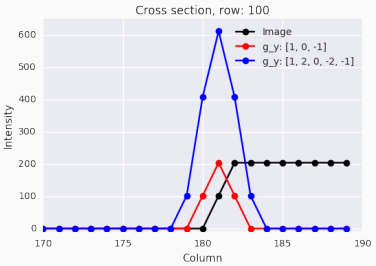


(d) θ

LOCATION OF EDGE

- The gradient magnitude has a ”wide response”, but we want a exact, thin edge.
- The maximum response seems reasonable.





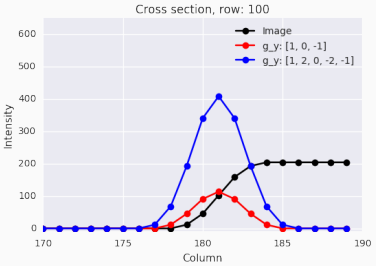


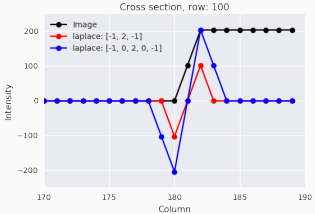
Figure 24: The images to the right show the image and gradient responses (normal on top, smoothed at the bottom) at the cross section indicated by red in the image above.

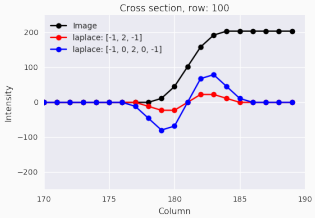
THE LAPLACE OPERATOR

- For a continuous bivariate function f, the *Laplace operator* is given by

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Function curvature information:
  - It is positive for decreasing gradient,
  - negative for increasing gradient,
  - zero at critical points in the gradient.
- Has two extrema for each edge, which is why we used it for image improvement earlier.
- If we use the location of the zero crossing, we can get a thin, well defined location of the edge.





Continuous	Discrete (3 neighbourhood)	Discrete (5 neighbourhood)
$f(x)$	$f[x]$	$f[x]$
$f'(x)$	$f[x+1/2] - f[x-1/2]$	$f[x+1] - f[x-1]$
$f''(x)$	$f[x+1] - 2f[x] + f[x-1]$	$f[x+2] - 2f[x] - f[x-2]$

If we apply the mapping from the first row to the second row on the second row, we arrive at the third row.
As is convention, we usually change the sign of the second derivative to arrive to the Laplace operator

2D LAPLACE OPERATOR

- Use the 1D Laplace approximation in both directions

$$
\begin{aligned}
-\nabla^2 f &= -\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \\
&\approx -(f[x+1,y] - 2f[x,y] + f[x-1,y]) - (f[x,y+1] - 2f[x,y] + f[x,y-1]) \\
&= -f[x+1,y] - f[x,y+1] + 4f[x,y] + f[x-1,y] + f[x,y+1]
\end{aligned}
$$

- This can be computed by convolving the image with the following filter

$$
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 4 & -1 \\
0 & -1 & 0
\end{bmatrix}
$$

FULL 3X3 LAPLACE OPERATOR

- If we, in addition, use the 1D Laplace approximation on the diagonal directions, we get the following filter

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- This convolution filter is the same as we used earlier in this slides for point detection and image sharpness enhancement.

- Let an image be modeled as a second order polynomial, then

$$
\begin{aligned}
f(x,y) &= k_1 + k_2 x + k_3 y + k_4 x^2 + k_5 xy + k_6 y^2 \\
\frac{\partial f}{\partial x} &= k_2 + 2k_4 x + k_5 y \\
\frac{\partial^2 f}{\partial x^2} &= 2k_4 \\
\frac{\partial f}{\partial y} &= k_3 + k_5 x + 2k_6 y \\
\frac{\partial^2 f}{\partial y^2} &= 2k_6
\end{aligned}
$$

- The discrete version of this polynomial, with $(x = 0, y = 0)$ in the center, is

$k_1 - k_2 - k_3 + k_4 + k_5 + k_6$	$k_1 - k_2 + k_4$	$k_1 - k_2 + k_3 + k_4 - k_5 + k_6$
$k_1 - k_3 + k_6$	k_1	$k_1 + k_3 + k_6$
$k_1 + k_2 - k_3 + k_4 - k_5 + k_6$	$k_1 + k_2 + k_4$	$k_1 + k_2 + k_3 + k_4 + k_5 + k_6$

- The negative Laplacian in the continuous case is then

$$-\nabla^2 f = -2(k_4 + k_6)$$

- This is the same result that Laplace operators

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \text{ and } \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

yield in the discrete case.

LARGER LAPLACE OPERATOR

Rationale: Approximate the laplace

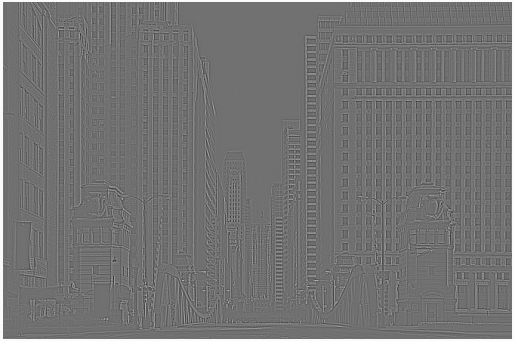
$$-\nabla^2 f = -\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2}$$

with a sum of sequential convolutions with some gradient operator.
Example using the Sobel operator

$$
\begin{aligned}
-\nabla^2_{5\times 5} &= -\left(\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}\right) \\
&= -\left(\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & -2 & 0 & 1 \\ 4 & 0 & -8 & 0 & 4 \\ 6 & 0 & -12 & 0 & 6 \\ 4 & 0 & -8 & 0 & 4 \\ 1 & 0 & -2 & 0 & 1 \end{bmatrix}\right) \\
&= \begin{bmatrix} -2 & -4 & -4 & -4 & -2 \\ -4 & 0 & 8 & 0 & -4 \\ -4 & 8 & 24 & 8 & -4 \\ -4 & 0 & 8 & 0 & -4 \\ -2 & -4 & -4 & -4 & -2 \end{bmatrix}
\end{aligned}
$$

3x3 Sobel gradient magnitude vs 3x3 Laplace

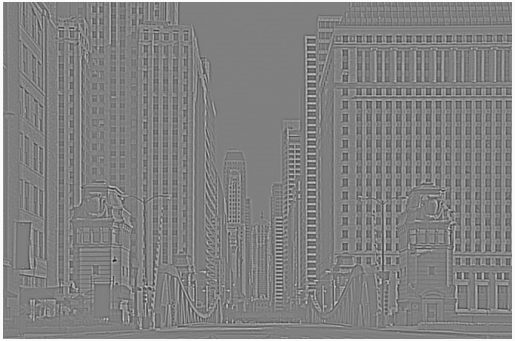
(a) Sobel gradient magnitude



(b) Laplace

5x5 Sobel gradient magnitude vs 5x5 Laplace

(a) Sobel gradient magnitude



(b) Laplace

NOISE SENSITIVITY IN LAPLACE OPERATOR

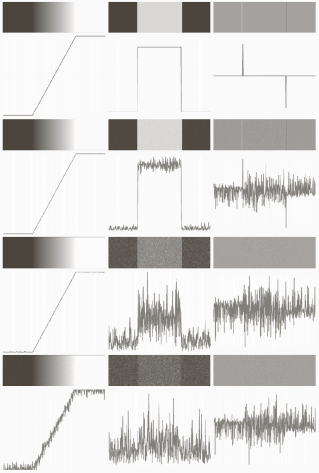


Figure 29: First column: Images and intensity profiles of a ramp edge corrupted by Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

LAPLACE OF GAUSSIAN (LOG) OPERATOR

- We made the gradient operators more robust to noise by building in a low-pass filtering.
- We can do the same sith the Laplace operator.
- It is common to build in a Gaussian filter (G)

$$-\nabla^2 * (G * f) = (-\nabla^2 * G) * f := LoG * f$$

- We call the operator $LoG = -\nabla^2 * G$ the *Laplace of Gaussian operator*

If we convolve a 3×3 (unnormalized) Gaussian filter G with a 3×3 Laplacian filter $-\nabla^2$

$$G = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad -\nabla^2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

we get a 5×5 LoG filter

$$LoG = \begin{bmatrix} -1 & -3 & -4 & -3 & -1 \\ -3 & 0 & 6 & 0 & -3 \\ -4 & 6 & 20 & 6 & -4 \\ -3 & 0 & 6 & 0 & -3 \\ -1 & -3 & -4 & -3 & -1 \end{bmatrix}.$$

EXAMPLE — 7X7 LOG

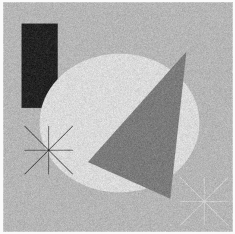
If we convolve a 3×3 (unnormalized) Gaussian filter G with a 5×5 Laplacian filter $-\nabla^2$

$$G = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad -\nabla^2 = \begin{bmatrix} -2 & -4 & -4 & -4 & -2 \\ -4 & 0 & 8 & 0 & -4 \\ -4 & 8 & 24 & 8 & -4 \\ -4 & 0 & 8 & 0 & -4 \\ -2 & -4 & -4 & -4 & -2 \end{bmatrix}$$

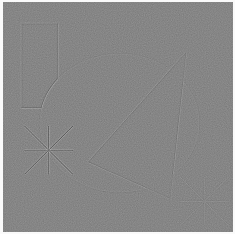
we get a 7×7 LoG filter

$$LoG = \begin{bmatrix} -2 & -8 & -14 & -16 & -14 & -8 & -2 \\ -8 & -24 & -24 & -16 & -24 & -24 & -8 \\ -14 & -24 & 30 & 80 & 30 & -24 & -14 \\ -16 & -16 & 80 & 160 & 80 & -16 & -16 \\ -14 & -24 & 30 & 80 & 30 & -24 & -14 \\ -8 & -24 & -24 & -16 & -24 & -24 & -8 \\ -2 & -8 & -14 & -16 & -14 & -8 & -2 \end{bmatrix}.$$

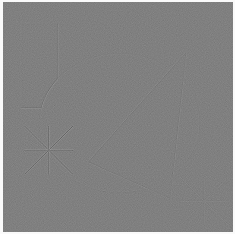
COMPARISON LAPLACE AND LOG



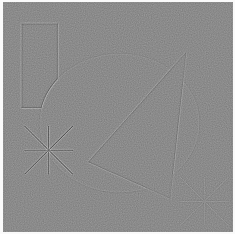
(a) Noisy shapes



(b) 5x5 Laplace



(c) 5x5 LoG



(d) 7x7 LoG

TWO WAYS TO CREATE THE LOG OPERATOR

- **Via convolution** As we have seen earlier, we can construct a LoG operator by convolving a Gaussian and a Laplacian filter.
- **Via continuous function** Another way is to compute the analytic, continuous LoG function, and sample from it. The continuous function results from taking the Laplacian on a Gaussian function.
- These two versions will in general not produce completely equal filters, but they are nevertheless both called LoG filters.

Assume you have a zero-mean 2D Gaussian with a diagonal covariance $\sigma^2 I$

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}.$$

It's first order derivatives are

$$\frac{\partial G}{\partial x}(x,y) = -\frac{x}{2\pi\sigma^4} \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}, \quad \frac{\partial G}{\partial y}(x,y) = -\frac{y}{2\pi\sigma^4} \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}.$$

It's second order derivatives are

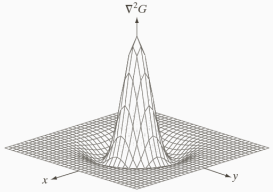
$$\frac{\partial^2 G}{\partial x^2}(x,y) = -\frac{1-\frac{x^2}{\sigma^2}}{2\pi\sigma^4} \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}, \quad \frac{\partial^2 G}{\partial y^2}(x,y) = -\frac{1-\frac{y^2}{\sigma^2}}{2\pi\sigma^4} \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}.$$

The LoG is then the Laplacian (the negative sum of the second order derivatives above)

$$-\nabla^2 G(x,y) = \frac{1}{2\pi\sigma^4}\left(2 - \frac{x^2+y^2}{\sigma^2}\right) \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}.$$

LOG FILTER FROM THE LOG FUNCTION

- We arrive at a LoG operator by sampling the LoG function at integer x ad y.
- We usually only sample from the area where the function is non-zero. Therefore, normally, the size of the operator is $\sim 3w \approx 8.5\sigma$, where $w = 2\sqrt{2}\sigma$ is the size of the LoG kernel (the positive peak in the middle).



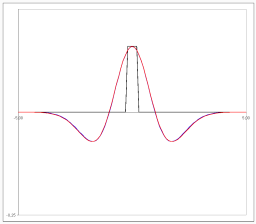
0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

Figure 31: LoG function. (a) 3D plot. (b) Image display. (c) Cross-section showing zero-crossings. (d) 5x5 mask approximation to the shape in (b).

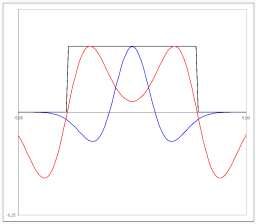
USAGE OF LOG OPERATORS

- The Laplace operator detects edges, but is sensitive to noise.
- One must often apply a low-pass filter prior to Laplace filtration.
- A LoG operator combines the low-pass and the Laplace filtering.

- Rule of thumb for structures: *The LoG-kernel needs to be smaller than the structure*
  - If the structure is smaller than half the LoG-kernel, the zero-crossing is outside the edge limits.
  - If the structure is larger then half the LoG-kernel, the zero-crossing is exactly at the edge limits.
  - If the structure is in between, it depends on the discretization and the LoG-approximation.



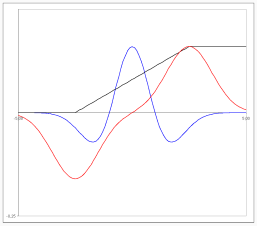
(a) Thin structure



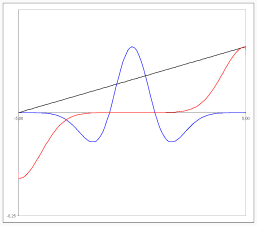
(b) Wide structure

Figure 32: LoG function (blue), rectangular structures (black), and the convolution respons (red).

- Rule of thumb for ramps: *The LoG filter must be larger than the ramp*
  - If the ramp is wider than the LoG-filter, the zero-crossing is not unique.
  - Otherwise the zero-crossing is at the center of the ramp.
  - Because of noise, we often need the LoG-kernel to be much larger than the width of the ramp.
- In general: *Choose the LoG-kernel and filter size thoughtfully.* The size should first be controlled by the standard deviation in the Gaussian.



(a) Thin ramp



(b) Wide ramp

Figure 33: LoG function (blue), ramp (black), and the convolution respons (red).

There is normally three steps in robust edge detection.

1. **Noise reduction.** Try to remove as much noise as possible withot smoothing the edges too much. Low-pass filtering.
2. **Edge-filtering.** Emphasize the edges. High-pass filtering, e.g. gradient operators.
3. **Edge-localization.** Post-processing of the result from the edge-filtering to find the exact location of the edges. Ideally the edge should be 1 pixel wide, and at the actual location of the edge in the image.

HALLMARKS OF A GOOD EDGE DETECTOR

- Finds *all* edges, and *only* the relevant edges.
- The position of the detected edge agrees with the actual location of the edge in the image.
- One edge respons for each edge.
- Robust against noise. Often need to compromise the robustness and location accuracy.

Canny's edge detection — idea

- Make an edge-detector that is optimal given these three criteria:
  - As good detection as possible (every edge, and only edges).
  - Good edge location.
  - One single respons for each edge.

1. Low-pass filtering using a Gaussian filter.
2. Compute the gradient magnitude and gradient direction (angle).
3. Thinning the gradient magnitude orthogonally on the edge.
4. Hysteresis thresholding. Given the thinned result g and two thresholds t_w and t_s:
   - 4.1 Tag all pixels (x, y) where $g[x, y] \geq t_s$.
   - 4.2 For all pixels where $t_w \leq g[x, y] < t_s$: If a neighbour (in a 4 or 8 neighbourhood) is tagged, then tag (x, y) as well.
   - 4.3 Repeat from (4.2) until convergence (no new tagged pixels).

EDGE DETECTION — COMPARISON OF METHODS

Figure 34: House

(a) 3x3 Sobel gradient magnitude, $T = 10\%$ of max

(b) 5x5 Laplace, $T = 10\%$ of max

(c) 7x7 LoG, $T = 10\%$ of max

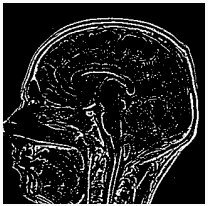
(d) Canny. Hysteresis threshold: 100 and 200

Figure 35: Edge detectors

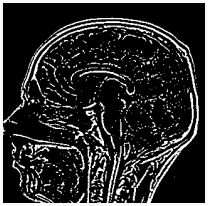
EDGE DETECTION — COMPARISON OF METHODS

Figure 36: CT scanning of head

(a) 3x3 Sobel gradient magnitude, $T = 20\%$ of max



(b) 5x5 Laplace, $T = 10\%$ of max



(c) 7x7 LoG, $T = 15\%$ of max

(d) Canny. Hysteresis threshold: 100 and 200

Figure 37: Edge detectors

SUMMARY

- We have derived simple edge-detectors.
- Gradient operators smooths in one direction and emphasizes edges in the other (orthogonal) direction.
- Can compute both edge magnitude and edge direction from gradient operators.
- The Laplace operator localizes the edge precisely, but emphasizes noise.
- The LoG-operator is a more noise tolerant version of the Laplace.
- Canny's edge detector compromizes noise reduction and edge localization.

Questions?