THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Image Filtering

Image filtering is used to:

- ➢ Remove noise
- ➢ Sharpen contrast
- ➢ Highlight contours
- ➢ Detect edges
- ➢ Other uses?

Image filters can be classified as linear or nonlinear.

Linear filters are also know as convolution filters as they can be represented using a matrix multiplication.

Thresholding and image equalisation are examples of nonlinear operations, as is the median filter.

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Median Filtering

Median filtering is a nonlinear method used to remove noise from images.

It is widely used as it is very effective at removing noise while preserving edges.

It is particularly effective at removing ‘salt and pepper’ type noise.

The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighbouring pixels.

The pattern of neighbours is called the "window", which slides, pixel by pixel, over the entire image.

The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Median Filtering example

The following example shows the application of a median filter to a simple one dimensional signal.

A window size of three is used, with one entry immediately preceding and following each entry.

Window for x[6]→y[6]

x = 3

3	9	4	52	3	8	6	2	2	9

9

y[0] = median[3 3 9] = 3	y[5] = median[3 6 8] = 6
y[1] = median[3 4 9] = 4	y[6] = median[2 6 8] = 6
y[2] = median[4 9 52] = 9	y[7] = median[2 2 6] = 2
y[3] = median[3 4 52] = 4	y[8] = median[2 2 9] = 2
y[4] = median[3 8 52] = 8	y[9] = median[2 9 9] = 9

y =

3	4	9	4	8	6	6	2	2	9

For y[1] and y[9], extend the left-most or right most value outside the boundaries of the image

same as leaving left-most or right most value unchanged after 1-D median

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Median Filtering

In the previous example, because there is no entry preceding the first value, the first value is repeated (as is the last value) to obtain enough entries to fill the window.

What effect does this have on the boundary values?

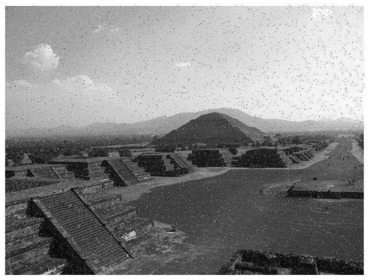
There are other approaches that have different properties that might be preferred in particular circumstances:

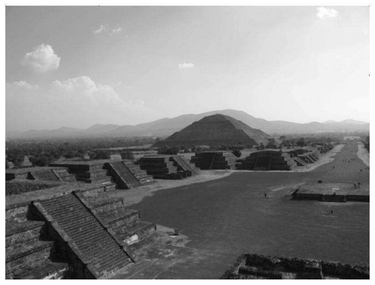
- Avoid processing the boundaries, with or without cropping the signal or image boundary afterwards.
- Fetching entries from other places in the signal. With images for example, entries from the far horizontal or vertical boundary might be selected.
- Shrinking the window near the boundaries, so that every window is full.

What effects might these approaches have on the boundary values?

Median Filtering

On the left is an image containing a significant amount of salt and pepper noise. On the right is the same image after processing with a median filter.





Notice the well preserved edges in the image.

There is some remaining noise on the boundary of the image. Why is this?

Median Filtering example 2

2D Median filtering example using a 3 x 3 sampling window:

Keeping border values unchanged

Sorted: 0,0,1,1,1,2,2,4,4

Input

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

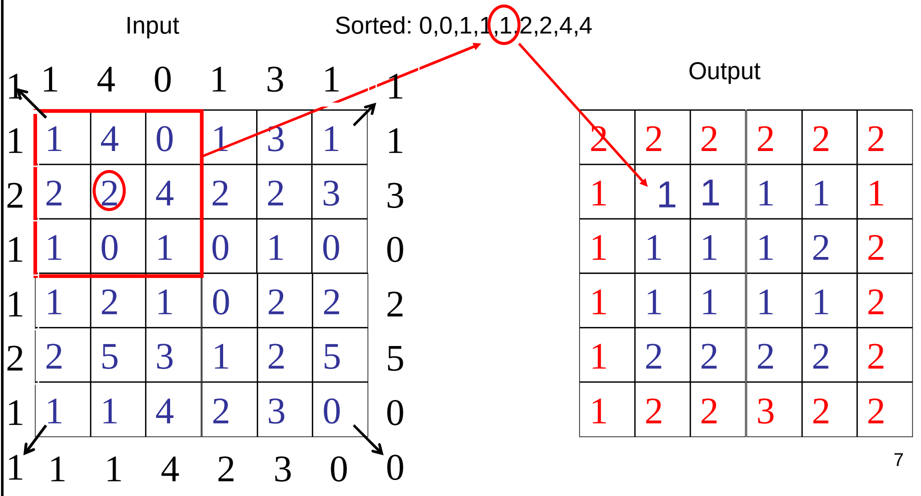
Output

1	4	0	1	3	1
2	1	1	1	1	3
1	1	1	1	2	0
1	1	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Median Filtering - Boundaries

2D Median filtering example using a 3 x 3 sampling window:

Extending border values outside with values at boundary



Input

1	1	4	0	1	3	1	1
1	1	4	0	1	3	1	1
2	2	2	4	2	2	3	3
1	1	0	1	0	1	0	0
1	1	2	1	0	2	2	2
2	2	5	3	1	2	5	5
1	1	1	4	2	3	0	0
1	1	1	4	2	3	0	0

Sorted: 0,0,1,1,1,2,2,4,4

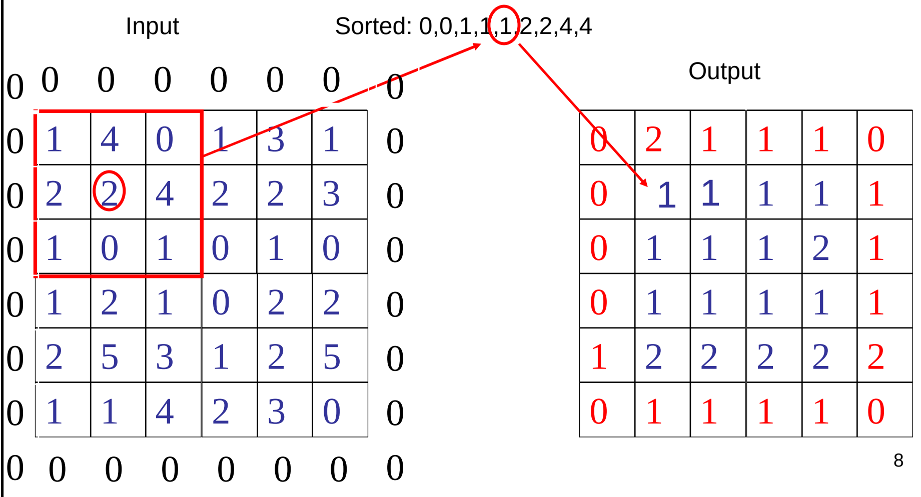
Output

2	2	2	2	2	2
1	1	1	1	1	1
1	1	1	1	2	2
1	1	1	1	1	2
1	2	2	2	2	2
1	2	2	3	2	2

Median Filtering - Boundaries

2D Median filtering example using a 3 x 3 sampling window:

Extending border values outside with 0s



Input

0	0	0	0	0	0	0	0
0	1	4	0	1	3	1	0
0	2	2	4	2	2	3	0
0	1	0	1	0	1	0	0
0	1	2	1	0	2	2	0
0	2	5	3	1	2	5	0
0	1	1	4	2	3	0	0
0	0	0	0	0	0	0	0

Sorted: 0,0,1,1,1,2,2,4,4

Output

0	2	1	1	1	0
0	1	1	1	1	1
0	1	1	1	2	1
0	1	1	1	1	1
1	2	2	2	2	2
0	1	1	1	1	0

Average Filtering

Average (or mean) filtering is a method of ‘smoothing’ images by reducing the amount of intensity variation between neighbouring pixels.

The average filter works by moving through the image pixel by pixel, replacing each value with the average value of neighbouring pixels, including itself.

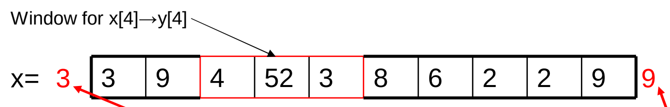
There are some potential problems:

- ➢ A single pixel with a very unrepresentative value can significantly affect the average value of all the pixels in its neighbourhood.
- ➢ When the filter neighbourhood straddles an edge, the filter will interpolate new values for pixels on the edge and so will blur that edge. This may be a problem if sharp edges are required in the output.

Average Filtering

The following example shows the application of an average filter to a simple one dimensional signal.

A window size of three is used, with one entry immediately preceding and following each entry.



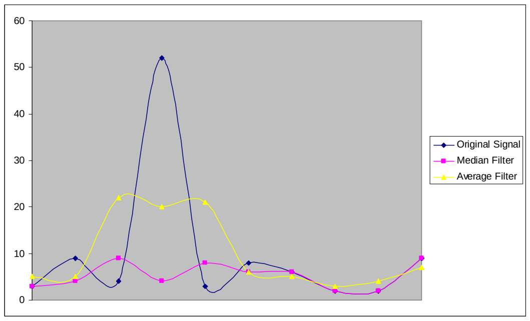
y[0] = round((3+3+9)/3)= 5	y[5] = round((3+8+6)/3)= 6
y[1] = round((3+9+4)/3)= 5	y[6] = round((8+6+2)/3)= 5
y[2] = round((9+4+52)/3)= 22	y[7] = round((6+2+2)/3)= 3
y[3] = round((4+52+3)/3)= 20	y[8] = round((2+2+9)/3)= 4
y[4] = round((52+3+8)/3)= 21	y[9] = round((2+9+9)/3)= 7

y=	5	5	22	20	21	6	5	3	4	7

For y[1] and y[9], extend the left-most or right most value outside the boundaries of the image

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Filter Comparison



The graph above shows the 1D signals from the median and average filter examples.

What are the differences in the way the filters have modified the original signal?

3 by 3 Average filtering

- Consider the following 3 by 3 average filter:
- We can write it mathematically as:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$I\_new(x,y) = \sum_{j=-1}^{1}\sum_{i=-1}^{1} 1 \times I\_old(x+i, y+j)$$

$$I\_new\_normalized(x,y) = \frac{1}{\sum_{j=-1}^{1}\sum_{i=-1}^{1} 1} \sum_{j=-1}^{1}\sum_{i=-1}^{1} 1 \times I\_old(x+i, y+j)$$

- Why normalizing is important ?
  - To keep the image pixel values between 0 and 255

Average Filtering example 2

2D Average filtering example using a 3 x 3 sampling window:

Keeping border values unchanged

Average = round(1+4+0+2+2+4+1+0+1)/9 = 2

Input

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output

1	4	0	1	3	1
2	2	2	2	1	3
1	2	1	1	1	0
1	2	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Average Filtering - Boundaries

2D Average filtering example using a 3 x 3 sampling window:

Extending border values outside with values at boundary

Input

Average = round(1+4+0+1+4+0+2+2+4)/9 = 2

1	1	4	0	1	3	1	1
1	1	4	0	1	3	1	1
2	2	2	4	2	2	3	3
1	1	0	1	0	1	0	0
1	1	2	1	0	2	2	2
2	2	5	3	1	2	5	5
1	1	1	4	2	3	0	0
1	1	1	4	2	3	0	0

Output

2	2	2	2	2	2
2	2	2	2	1	2
1	2	1	1	1	2
2	2	1	1	1	2
2	2	2	2	2	2
2	2	3	3	2	2

Average Filtering - Boundaries

2D Median filtering example using a 3 x 3 sampling window:

Extending border values outside with 0s (Zero-padding)

Input

Average = round(2+5+0+3+0+0+0+0+0)/9 = 1

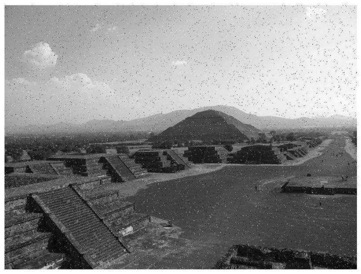
0	0	0	0	0	0	0	0
0	1	4	0	1	3	1	0
0	2	2	4	2	2	3	0
0	1	0	1	0	1	0	0
0	1	2	1	0	2	2	0
0	2	5	3	1	2	5	0
0	1	1	4	2	3	0	0
0	0	0	0	0	0	0	0

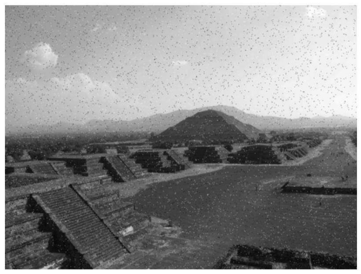
Output

1	1	1	1	1	1
1	2	2	2	1	1
1	2	1	1	1	1
1	2	1	1	1	1
1	2	2	2	2	2
1	2	2	2	1	1

Average Filtering

On the left is an image containing a significant amount of salt and pepper noise. On the right is the same image after processing with an Average filter.





What are the differences in the result compared with the Median filter?

Is this a linear (convolution) or nonlinear filter?

Gaussian Filtering

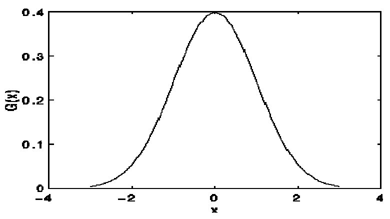
Gaussian filtering is used to blur images and remove noise and detail.

In one dimension, the Gaussian function is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

Where σ is the standard deviation of the distribution. The distribution is assumed to have a mean of 0.

Shown graphically, we see the familiar bell shaped Gaussian distribution.



Gaussian distribution with mean 0 and σ = 1

Gaussian filtering

- Significant values

x	0	1	2	3	4
$\sigma * G(x) / 0.399$	1	$e^{-0.5/\sigma^2}$	e^{-2/σ^2}	$e^{-9/4\sigma^2}$	e^{-8/σ^2}
$G(x) / G(0)$	1	$e^{-0.5/\sigma^2}$	e^{-2/σ^2}	$e^{-9/4\sigma^2}$	e^{-8/σ^2}

For σ=1:

x	0	1	2
$G(x)$	0.399	0.242	0.05
$G(x) / G(0)$	1	0.6	0.125

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

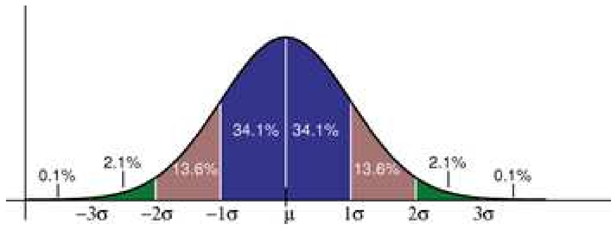
Gaussian Filtering

Standard Deviation

The Standard deviation of the Gaussian function plays an important role in its behaviour.

The values located between +/- σ account for 68% of the set, while two standard deviations from the mean (blue and brown) account for 95%, and three standard deviations (blue, brown and green) account for 99.7%.

This is very important when designing a Gaussian kernel of fixed length.



Distribution of the Gaussian function values (Wikipedia)

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Gaussian Filtering

The Gaussian function is used in numerous research areas:

- It defines a probability distribution for noise or data.
- It is a smoothing operator.
- It is used in mathematics.

The Gaussian function has important properties which are verified with respect to its integral:

$$I = \int_{-\infty}^{\infty} \exp\left(-\mathrm{x}^2\right) dx = \sqrt{\pi}$$

In probabilistic terms, it describes 100% of the possible values of any given space when varying from negative to positive values

Gauss function is never equal to zero.

It is a symmetric function.

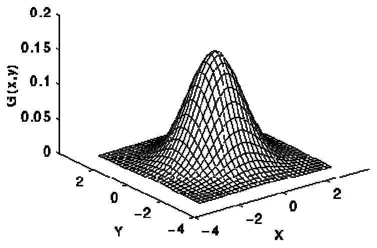
Gaussian Filtering

When working with images we need to use the two dimensional Gaussian function.

This is simply the product of two 1D Gaussian functions (one for each direction) and is given by:

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

A graphical representation of the 2D Gaussian distribution with mean(0,0) and σ = 1 is shown to the right.



Gaussian Filtering

The Gaussian filter works by using the 2D distribution as a point-spread function.

This is achieved by convolving the 2D Gaussian distribution function with the image.

We need to produce a discrete approximation to the Gaussian function.

This theoretically requires an infinitely large convolution kernel, as the Gaussian distribution is non-zero everywhere.

Fortunately the distribution has approached very close to zero at about three standard deviations from the mean. 99% of the distribution falls within 3 standard deviations.

This means we can normally limit the kernel size to contain only values within three standard deviations of the mean.

Gaussian Filtering

Gaussian kernel coefficients are sampled from the 2D Gaussian function.

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

Where σ is the standard deviation of the distribution.

The distribution is assumed to have a mean of zero.

We need to discretize the continuous Gaussian functions to store it as discrete pixels.

An integer valued 5 by 5 convolution kernel approximating a Gaussian with a σ of 1 is shown to the right,

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Gaussian Filtering

The Gaussian filter is a non-uniform low pass filter.

The kernel coefficients diminish with increasing distance from the kernel’s centre.

Central pixels have a higher weighting than those on the periphery.

Larger values of σ produce a wider peak (greater blurring).

Kernel size must increase with increasing σ to maintain the Gaussian nature of the filter.

Gaussian kernel coefficients depend on the value of σ.

At the edge of the mask, coefficients must be close to 0.

The kernel is rotationally symmetric with no directional bias.

Gaussian kernel is separable, which allows fast computation.

Gaussian filters might not preserve image brightness.

Gaussian Filtering examples

- Is the kernel

1	6	1

a 1D Gaussian kernel?
- Give a suitable integer-value 5 by 5 convolution mask that approximates a Gaussian function with a σ of 1.4.
- How many standard deviations from the mean are required for a Gaussian function to fall to 5%, or 1% of its peak value?
- What is the value of σ for which the value of the Gaussian function is halved at +/-1 x.
- Compute the horizontal Gaussian kernel with mean=0 and σ=1, σ=5.

Gaussian Filtering examples

Apply the Gaussian filter to the image:
Borders: keep border values as they are

15	20	25	25	15	10
20	15	50	30	20	15
20	50	55	60	30	20
20	15	65	30	15	30
15	20	30	20	25	30
20	25	15	20	10	15

Original image

¼*

1	2	1

15	20	24	23	16	10
20	25	36	33	21	15
20	44	55	51	35	20
20	29	44	35	22	30
15	21	25	24	25	30
20	21	19	16	14	15

¼*

1
2
1

15	20	24	23	16	10
19	28	38	35	23	15
20	35	48	43	28	21
19	31	42	36	26	28
18	23	28	25	22	21
20	21	19	16	14	15

Or:

1	2	1
2	4	2
1	2	1

*1/16

Gaussian Filtering examples

Convolve the Gaussian filter (μ=0, σ=1, 0 padding) to the image:

15	20	25	25	15	10
20	15	50	30	20	15
20	50	55	60	30	20
20	15	65	30	15	30
15	20	30	20	25	30
20	25	15	20	10	15

Original image

σ=1 means the Gaussian function is at 0 at +/-3σ (g(3)=0.004 and g(0)=0.399, g(1)=0.242, g(2)=0.054). We can approximate a Gaussian function with a kernel of width 5:

1	5	8	5	1

/20

1
5
8
5
1

/20

12	14	25	21	13	9
18	24	43	35	20	15
20	30	54	41	23	21
19	26	51	35	21	26
17	21	35	24	18	24
13	16	17	15	11	15

10	16	20	19	14	8
15	27	34	32	23	13
18	33	42	38	27	16
17	30	38	35	26	17
14	23	27	25	21	15
10	15	16	15	13	10

Gaussian Filtering examples

Convolve the Gaussian filter (μ=0, σ=0.2) to the image:

15	20	25	25	15	10
20	15	50	30	20	15
20	50	55	60	30	20
20	15	65	30	15	30
15	20	30	20	25	30
20	25	15	20	10	15

Original image

σ=0.2 means the Gaussian function is at 0 at +/-3σ (g(0.6)= 0.02 and g(0)= 1.99, g(1)=0.0000074). We can approximate a Gaussian function with a kernel of width 1:

0	1	0

0
1
0

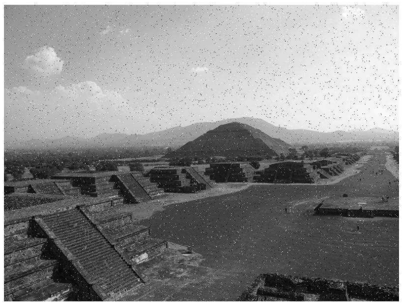
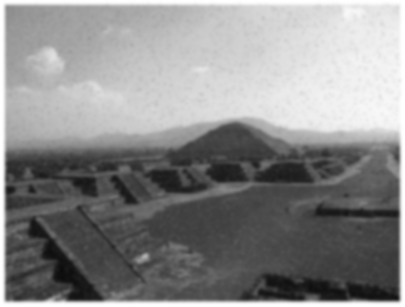
15	20	25	25	15	10
20	15	50	30	20	15
20	50	55	60	30	20
20	15	65	30	15	30
15	20	30	20	25	30
20	25	15	20	10	15

15	20	25	25	15	10
20	15	50	30	20	15
20	50	55	60	30	20
20	15	65	30	15	30
15	20	30	20	25	30
20	25	15	20	10	15

Gaussian Filtering

Gaussian filtering is used to remove noise and detail. It is not particularly effective at removing salt and pepper noise.

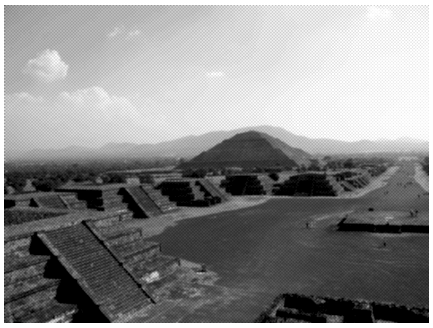
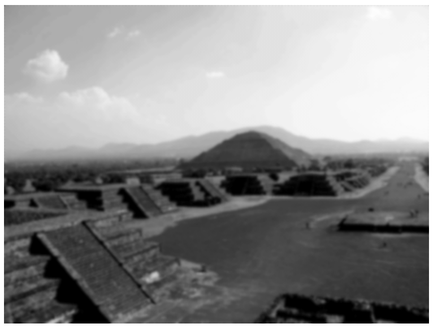
Compare the results below with those achieved by the median filter.

Gaussian Filtering

Gaussian filtering is more effective at smoothing images. It has its basis in the human visual perception system. It has been found that neurons create a similar filter when processing visual images.

The halftone image at left has been smoothed with a Gaussian filter and is displayed to the right.

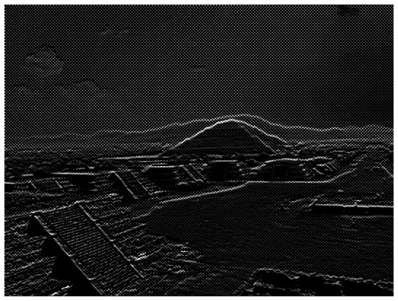



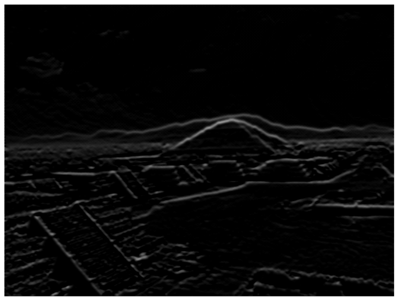
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Gaussian Filtering

This is a common first step in edge detection.

The images below have been processed with a Sobel filter commonly used in edge detection applications. The image to the right has had a Gaussian filter applied prior to processing.





THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Convolution

The convolution of two functions f and g is defined as:

$$(f * g)(x, y) = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} f(u, v) g(x-u, y-v)$$

Where $f(x,y)$ is a function that represents the image and $g(x,y)$ is a function that represents the kernel.

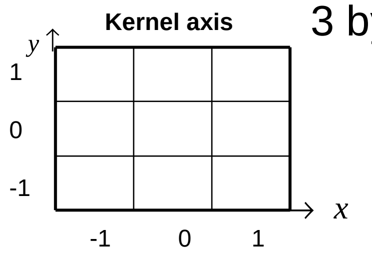
In practice, the kernel is only defined over a finite set of points, so we can modify the definition to:

$$(f * g)(x, y) = \sum_{v=y-h}^{y+h} \sum_{u=x-w}^{x+w} f(u, v) g(x-u, y-v)$$

Where $2w+1$ is the width of the kernel and $2h+1$ is the height of the kernel.

g is defined only over the points $[-w, w] \times [-h, h]$

3 by 3 convolution



$$\begin{pmatrix} \alpha_{-11} & \alpha_{01} & \alpha_{11} \\ \alpha_{-10} & \alpha_{00} & \alpha_{10} \\ \alpha_{-1-1} & \alpha_{0-1} & \alpha_{1-1} \end{pmatrix}$$

- Consider the above 3 by 3 kernel with weights α_{ij} .
- We can write the convolution Image I_old by the above kernel as:

$$I\_new(x,y) = \sum_{j=-1}^{1}\sum_{i=-1}^{1} \alpha_{ij} I\_old(x-i, y-j)$$

If all α_{ij} are positive we can normalise the kernel.

$$I\_new\_normalized(x,y) = \frac{1}{\sum_{j=-1}^{1}\sum_{i=-1}^{1} \alpha_{ij}} \sum_{j=-1}^{1}\sum_{i=-1}^{1} \alpha_{ij} I\_old(x-i, y-j)$$

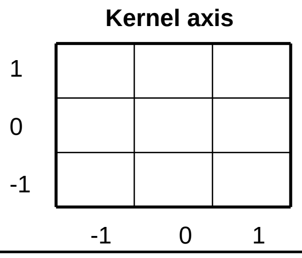
Why normalizing is important ?

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Convolution Pseudocode

Pseudocode for the convolution of an image f(x,y) with a kernel k(x,y) (2w+1 columns, 2h+1 lines) to produce a new image g(x,y):

```
for y = 0 to ImageHeight do
    for x = 0 to ImageWidth do
        sum = 0
        for i= -h to h do
            for j = -w to w do
                sum = sum + k(j,i) * f(x - j, y - i)
            end for
        end for
        g(x,y) = sum
    end for
end for
```



Convolution equation for a 3 by 3 kernel

The pixel value p(x,y) of image f after convolution with a 3 by 3 kernel k is:

$$p(x,y)=\sum_{i=-1}^{i=+1}\sum_{j=-1}^{j=+1}k(j,i)\,f(x-j,y-i)$$
$$=k(-1,-1)\,f(x+1,y+1)+$$
$$=k(0,-1)\,f(x,y+1)+$$
$$=k(1,-1)\,f(x-1,y+1)+$$
$$=k(-1,0)\,f(x+1,y)+$$
$$=k(0,0)\,f(x,y)+$$
$$=k(1,0)\,f(x-1,y)+$$
$$=k(-1,1)\,f(x+1,y-1)+$$
$$=k(0,1)\,f(x,y-1)+$$
$$=k(1,1)\,f(x-1,y-1)$$

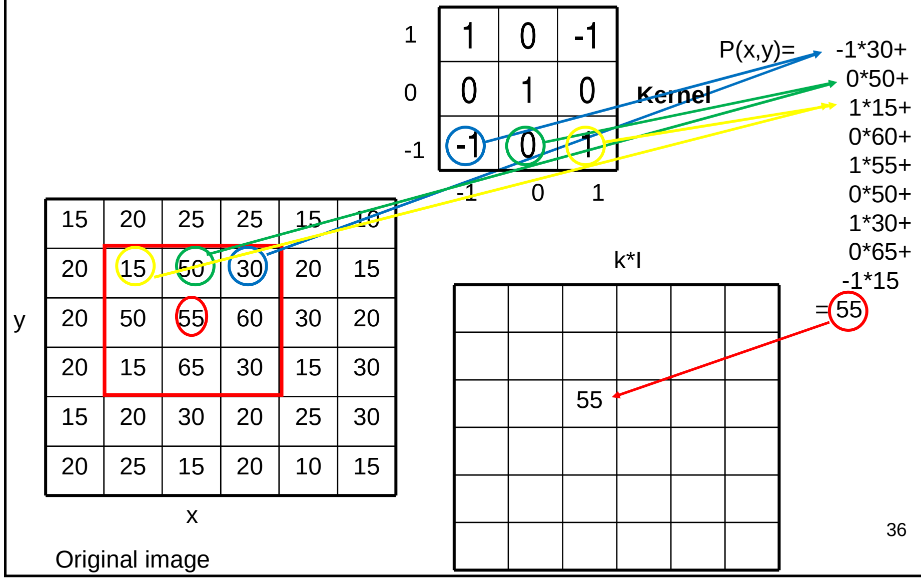
Kernel k

$$\begin{array}{c|ccc} y & & & \\ 1 & k(-1,1) & k(0,1) & k(1,1) \\ 0 & k(-1,0) & k(0,0) & k(1,0) \\ -1 & k(-1,-1) & k(0,-1) & k(1,-1) \\ \hline & -1 & 0 & 1 \; x \end{array}$$

35

Convolution examples

Do the convolution of the following kernel k with the image I:



Kernel:

$$\begin{array}{c|ccc} 1 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 1 \\ \hline & -1 & 0 & 1 \end{array}$$

Original image:

15	20	25	25	15	10
20	15	50	30	20	15
20	50	55	60	30	20
20	15	65	30	15	30
15	20	30	20	25	30
20	25	15	20	10	15

P(x,y)= -1*30+ 0*50+ 1*15+ 0*60+ 1*55+ 0*50+ 1*30+ 0*65+ -1*15 =55

36

Convolution – Potential Problems

Summation over a neighbourhood might exceed the range and/or sign permitted in the image format:

- The data may need to be temporarily stored in a 16 – 32 bit integer representation.
- Then normalised back to the appropriate range (0-255 for an 8 bit image).

Another issue is how to deal with image borders:

- Convolution is not possible if part of the kernel lies outside the image.
- What is the size of image window which is processed normally when performing a Convolution of size m x n on an original image of size M x N ?

Convolution Border Issues

How to deal with convolution at image borders:

1) Extend image limits with 0s (Zero padding)
2) Extend image limits with own image values
3) Generate specific filters to take care of the borders

Find the corner and border specific kernel for:

1	1	1
1	1	1
1	1	1

Image top left corner filter:
Kernel center (in red)

🟥 1	1
1	1

-1	-1	-1
-1	8	-1
-1	-1	-1

Image left most column filter:
Kernel center (in red)

-1	-1
-1	5
-1	-1

-1	0	1
-1	0	1
-1	0	1

Top row filter:
Kernel center (in red)

-1	0	1

Edge Detection

Edges in images are areas with strong intensity contrasts; a jump in intensity from one pixel to the next.

The process of edge detection significantly reduces the amount of data and filters out unneeded information, while preserving the important structural properties of an image.

There are many different edge detection methods, the majority of which can be grouped into two categories:

- Gradient,
- and Laplacian.

The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.

The Laplacian method searches for zero crossings in the second derivative of the image .

We will look at two examples of the gradient method, Sobel and Prewitt.

39

Edge Detection

Edge detection is a major application for convolution.

What is an edge:

- A location in the image where is a sudden change in the intensity/colour of pixels.
- A transition between objects or object and background.
- From a human visual perception perspective it attracts attention.

Problem: Images contain noise, which also generates sudden transitions of pixel values.

Usually there are three steps in the edge detection process:

1) **Noise reduction**
   Suppress as much noise as possible without removing edges.
2) **Edge enhancement**
   Highlight edges and weaken elsewhere (high pass filter).
3) **Edge localisation**
   Look at possible edges (maxima of output from previous filter) and eliminate spurious edges (often noise related).

40

Edge Detection

Gradient Estimation

Estimation of the intensity gradient at a pixel in the x and y direction, for an image f, is given by:

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x-1, y)$$

$$\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y-1)$$

We can introduce noise smoothing by convoluting with a low pass filter (e.g. mean, Gaussian, etc)

The gradient calculation (g_x,g_y) can be expressed as:

$$g_x = h_x * f(x, y)$$

$$g_y = h_y * f(x, y)$$

Sobel Filter

The Sobel filter is used for edge detection.

It works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction.

The result shows how abruptly or smoothly the image changes at each pixel, and therefore how likely it is that that pixel represents an edge.

It also shows how that edge is likely to be oriented.

The result of applying the filter to a pixel in a region of constant intensity is a zero vector.

The result of applying it to a pixel on an edge is a vector that points across the edge from darker to brighter values.

Sobel Filter

The sobel filter uses two 3 x 3 kernels. One for changes in the horizontal direction, and one for changes in the vertical direction.

The two kernels are convolved with the original image to calculate the approximations of the derivatives.

If we define Gx and Gy as two images that contain the horizontal and vertical derivative approximations respectively, the computations are:

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * A \quad \text{and} \quad G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * A$$

Where A is the original source image.

The x coordinate is defined as increasing in the right-direction and the y coordinate is defined as increasing in the down-direction.

Sobel Filter

To compute G_x and G_y we move the appropriate kernel (window) over the input image, computing the value for one pixel and then shifting one pixel to the right. Once the end of the row is reached, we move down to the beginning of the next row.

The example below shows the calculation of a value of G_x:

a11	a12	a13	…	
a21	a22	a23	…	
a31	a32	a33	…	
…	…	…	…	

Input image

kernel =

1	0	-1
2	0	-2
1	0	-1

b11	b12	b13	…	
b21	b22	b23	…	
b31	b32	b33	…	
…	…	…	…	

Output image (G_x)

$b_{22} = a_{13} - a_{11} + 2a_{23} - 2a_{21} + a_{33} - a_{31}$

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Edge Detection

The kernels contain positive and negative coefficients.

This means the output image will contain positive and negative values.

For display purposes we can:

- map the gradient of zero onto a half-tone grey level.
  - This makes negative gradients appear darker, and positive gradients appear brighter.
- Use the absolute values of the gradient map (stretched between 0 and 255).
  - This makes very negative and very positive gradients appear brighter.

The kernels are sensitive to horizontal and vertical transitions.

The measure of an edge is its amplitude and angle. These are readily calculated from G_x and G_y.

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Sobel Filter

At each pixel in the image, the gradient approximations given by G_x and G_y are combined to give the gradient magnitude, using:

$$G = \sqrt{G_x^2 + G_y^2}$$

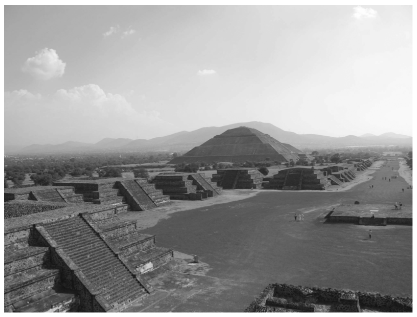
The gradient's direction is calculated using:

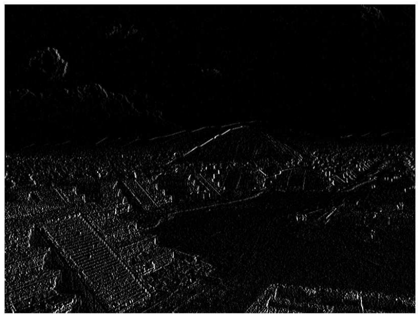
$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

A Θ value of 0 would indicate a vertical edge that is darker on the left side.

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Sobel Filter





The image to the right above is Gx, calculated as:

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * A$$

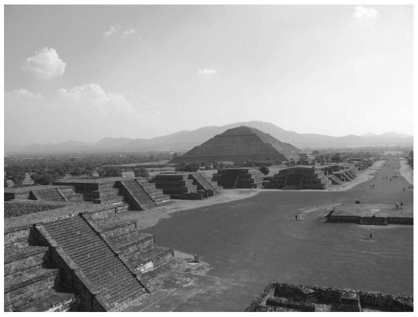
Where A is the original image to the left.

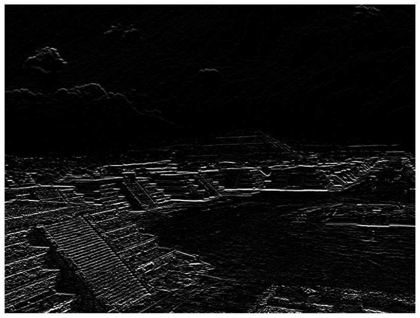
Notice the general orientation of the edges.

What would you expect to be different in Gy?

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Sobel Filter





The image to the right above is Gy, calculated as:

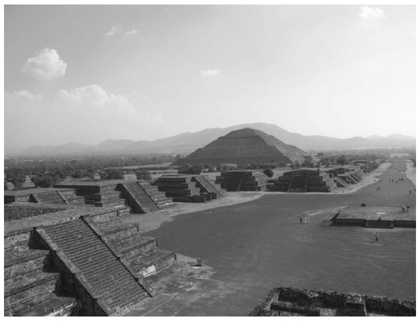
$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * A$$

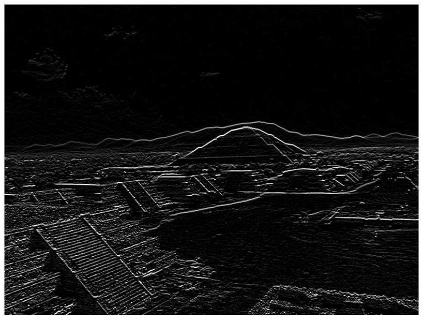
Where A is the original image to the left.

What do we expect from the combined image?

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Sobel Filter





The image to the right above is the result of combining the G$_x$ and G$_y$ derivative approximations calculated from image A on the left.

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Sobel Filter example

Convolve the Sobel kernels to the original image (use 0 padding)

10	50	10	50	10
10	55	10	55	10
10	65	10	65	10
10	50	10	50	10
10	55	10	55	10

Original image

1	0	-1
2	0	-2
1	0	-1

-1	-2	-1
0	0	0
1	2	1

155	0	0	0	-155
225	0	0	0	-225
235	0	0	0	-235
220	0	0	0	-220
160	0	0	0	-160

-75	-130	-130	-130	-75
-15	-30	-10	-30	-15
5	10	10	10	5
10	20	20	20	10
70	120	120	120	70

Sobel filter example

- Compute Gx and Gy, gradients of the image performing the convolution of Sobel kernels with the image
- Use zero-padding to extend the image

0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10

y

x

G_x

0	30	30	0	-30
0	40	40	0	-40
0	40	40	0	-40
0	40	40	0	-40
0	30	30	0	-30

G_y

-10	-30	-40	-30	-10
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
10	30	40	30	10

1	0	-1
2	0	-2
1	0	-1

h_x

-1	-2	-1
0	0	0
1	2	1

h_y

Sobel filter example

- Compute Gx and Gy, gradients of the image performing the convolution of Sobel kernels with the image
- Use border values to extend the image

0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10

y

x

G_x

0	40	40	0	0
0	40	40	0	0
0	40	40	0	0
0	40	40	0	0
0	40	40	0	0

G_y

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	0	-1
2	0	-2
1	0	-1

h_x

-1	-2	-1
0	0	0
1	2	1

h_y

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

	0	0		
	0	0		
	0	0		
	0	0		
	0	0		

Sobel filter example

- Compute Gx and Gy, gradients of the image performing the convolution of Sobel kernels with the image
- Use border values to extend the image

0	0	0	0	0
0	0	0	0	0
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10

y

x

G_x

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

G_y

0	0	0	0	0
-40	-40	-40	-40	-40
-40	-40	-40	-40	-40
0	0	0	0	0
0	0	0	0	0

1	0	-1
2	0	-2
1	0	-1

h_x

-1	-2	-1
0	0	0
1	2	1

h_y

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

-∞	-∞	-∞	-∞	-∞
-∞	-∞	-∞	-∞	-∞

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Prewitt Filter

The Prewitt filter is similar to the Sobel in that it uses two 3 x 3 kernels. One for changes in the horizontal direction, and one for changes in the vertical direction.

The two kernels are convolved with the original image to calculate the approximations of the derivatives.

If we define Gx and Gy as two images that contain the horizontal and vertical derivative approximations respectively, the computations are:

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} * A \quad \text{and} \quad G_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} * A$$

Where A is the original source image.

The x coordinate is defined as increasing in the right-direction and the y coordinate is defined as increasing in the down-direction.

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Prewitt Filter

To compute G_x and G_y we move the appropriate kernel (window) over the input image, computing the value for one pixel and then shifting one pixel to the right. Once the end of the row is reached, we move down to the beginning of the next row.

The example below shows the calculation of a value of G_x:

a11	a12	a13	…	
a21	a22	a23	…	
a31	a32	a33	…	
…	…	…	…	

Input image

kernel =

1	0	-1
1	0	-1
1	0	-1

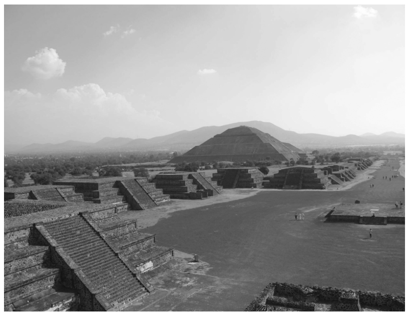
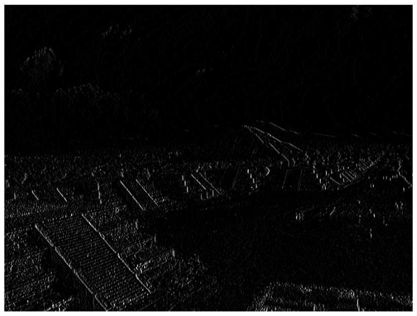
b11	b12	b13	…	
b21	b22	b23	…	
b31	b32	b33	…	
…	…	…	…	

Output image (G_x)

$b_{22} = -a_{11} + a_{13} - a_{21} + a_{23} - a_{31} + a_{33}$

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Prewitt Filter

The image to the right above is G_x, calculated as: $G_x = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} * A$

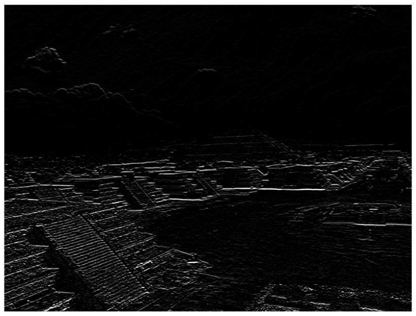
Where A is the original image to the left.

Notice the general orientation of the edges.

What would you expect to be different in G_y?

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Prewitt Filter



The image to the right above is Gy, calculated as:

$$G_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} * A$$

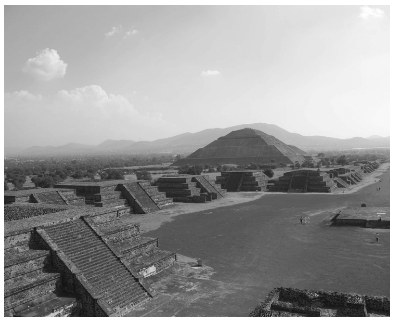
Where A is the original image to the left.

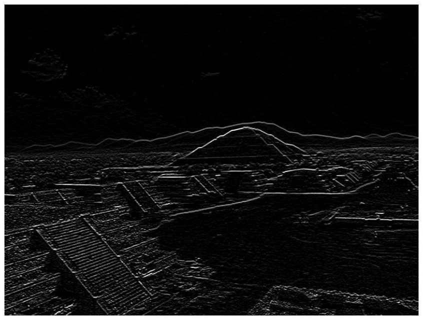
What do we expect from the combined image?

57

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Prewitt Filter





The image to the right above is the result of combining the G_x and G_y derivative approximations calculated from image A on the left.

58

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Prewitt Filter example

Convolve the Prewitt kernels to the original image (0 padding)

10	50	10	50	10
10	55	10	55	10
10	65	10	65	10
10	50	10	50	10
10	55	10	55	10

Original image

1	0	-1
1	0	-1
1	0	-1

-1	-1	-1
0	0	0
1	1	1

105	0	0	0	-105
170	0	0	0	-170
170	0	0	0	-170
170	0	0	0	-170
105	0	0	0	-105

-65	-75	-120	-75	-65
-15	-15	-30	-15	-15
5	5	10	5	5
10	10	20	10	10
50	70	110	75	65

Laplacian filter example

- Compute the convolution of the Laplacian kernels L_4 and L_8 with the image
- Use border values to extend the image

0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10

y

x

L_8

0	-30	30	0	0
0	-30	30	0	0
0	-30	30	0	0
0	-30	30	0	0
0	-30	30	0	0

L_4

0	-10	10	0	0
0	-10	10	0	0
0	-10	10	0	0
0	-10	10	0	0
0	-10	10	0	0

-1	-1	-1
-1	8	-1
-1	-1	-1

L_8

0	-1	0
-1	4	-1
0	-1	0

L_4

Laplacian filter example

- Compute the convolution of the Laplacian kernels L_4 and L_8 with the image
- Use zero-padding to extend the image

0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10

y

x

-1	-1	-1
-1	8	-1
-1	-1	-1

L_8

0	-1	0
-1	4	-1
0	-1	0

L_4

L_8

0	-20	50	50	50
0	-30	30	0	30
0	-30	30	0	30
0	-30	30	0	30
0	-20	50	50	50

L_4

0	-10	20	10	20
0	-10	10	0	10
0	-10	10	0	10
0	-10	10	0	10
0	-10	20	10	20