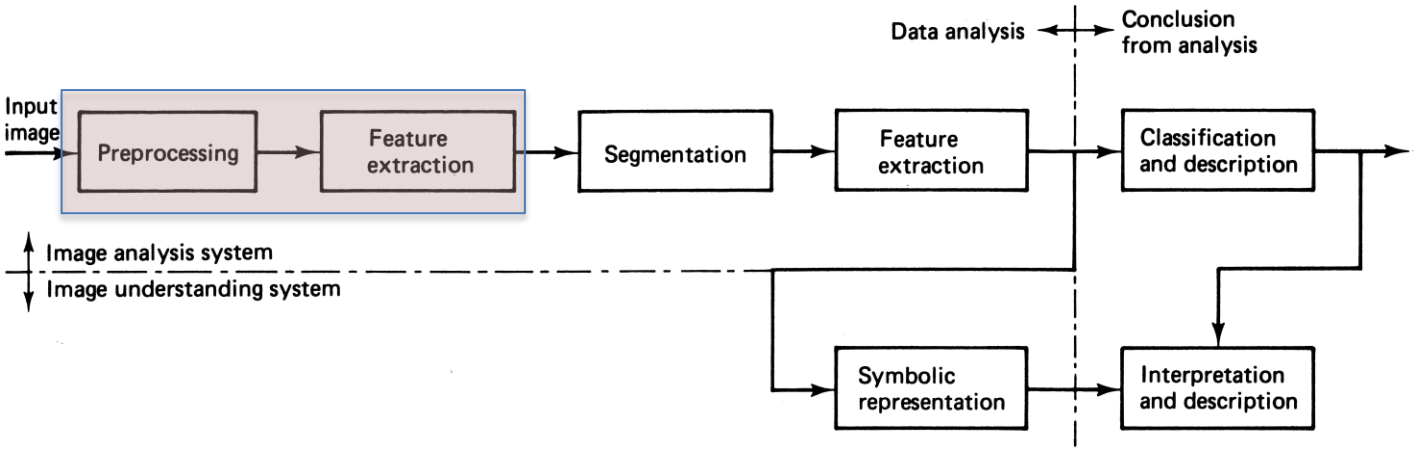
Introduction to programming
Image processing

Guido Klingbeil
Centre for Mathematical Biology
19.10.2012

Image processing

Image processing is any form of signal processing on images, such as a photograph or video frame. However, the output may be either an image or a set of characteristics or parameters related to the image.

An image processing system may consist of several steps:



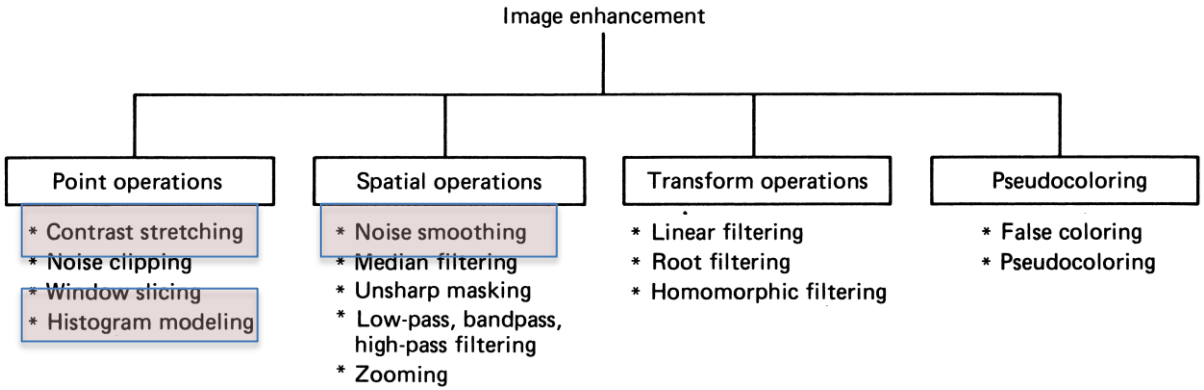
What we will cover

- Image enhancement:
  - Contrast stretching
  - Histogram equalisation
- Image filtering
- Feature extraction:
  - Edge detection
  - Detecting straight lines

Image enhancement

Image enhancement is the accentuation, or sharpening, of image features such as edges, boundaries, or contrast to make the image more accessible to a subsequent analysis.

It does not add any inherent information but increses the dynamic range of the existing information



Point operators

The image enhancement techniques we are looking at are point operators. They take the grey level of a pixel where is the maximum grey level and transform it:

( )

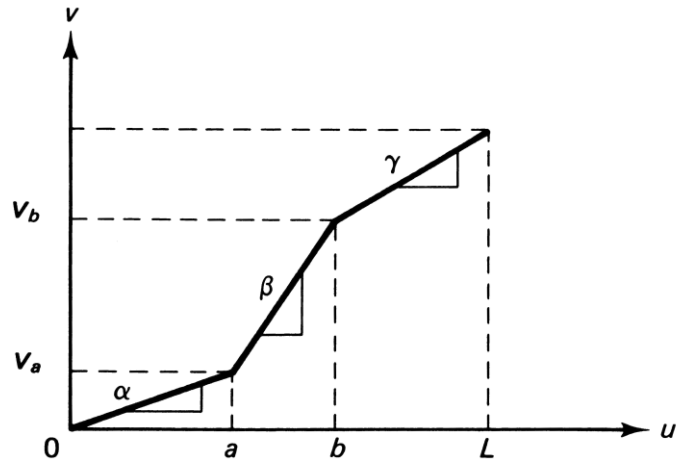
```
void point_operator(image* image_in) {
    int i = 0;
    int j = 0;

    for(i = 0; i < image_in->uint_yres; i ++) {
        for(j = 0; j < image_in->uint_xres; j ++) {
            /* v = f(u) */
        }
    }
    return;
}
```

Contrast stretching

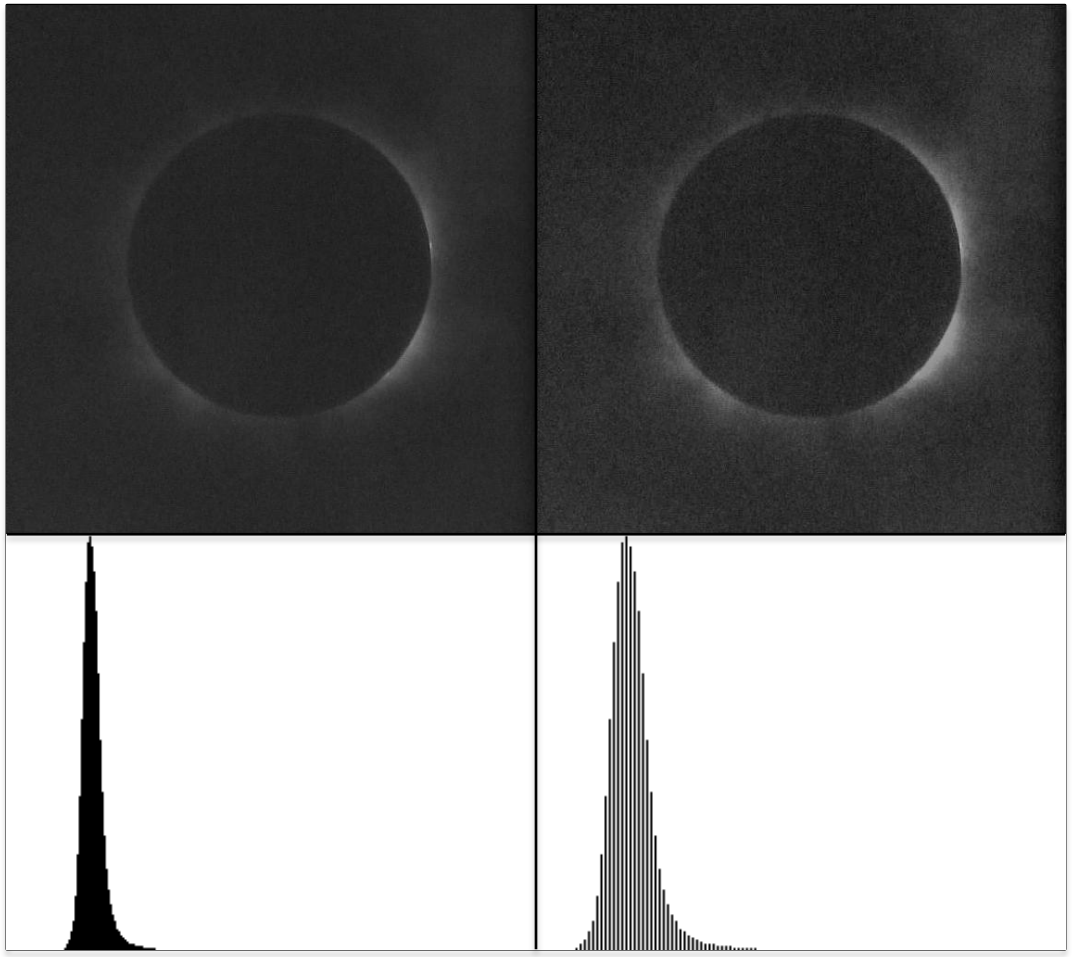
Also called normalisation.

The purpose is to bring the image into a range that is more familiar or normal to the senses, hence the term normalization.

( ) ( ) ( )



Contrast stretching



Histogram equalisation

The probability that a pixes has a certain grey level is given by:

( ) ———— ,

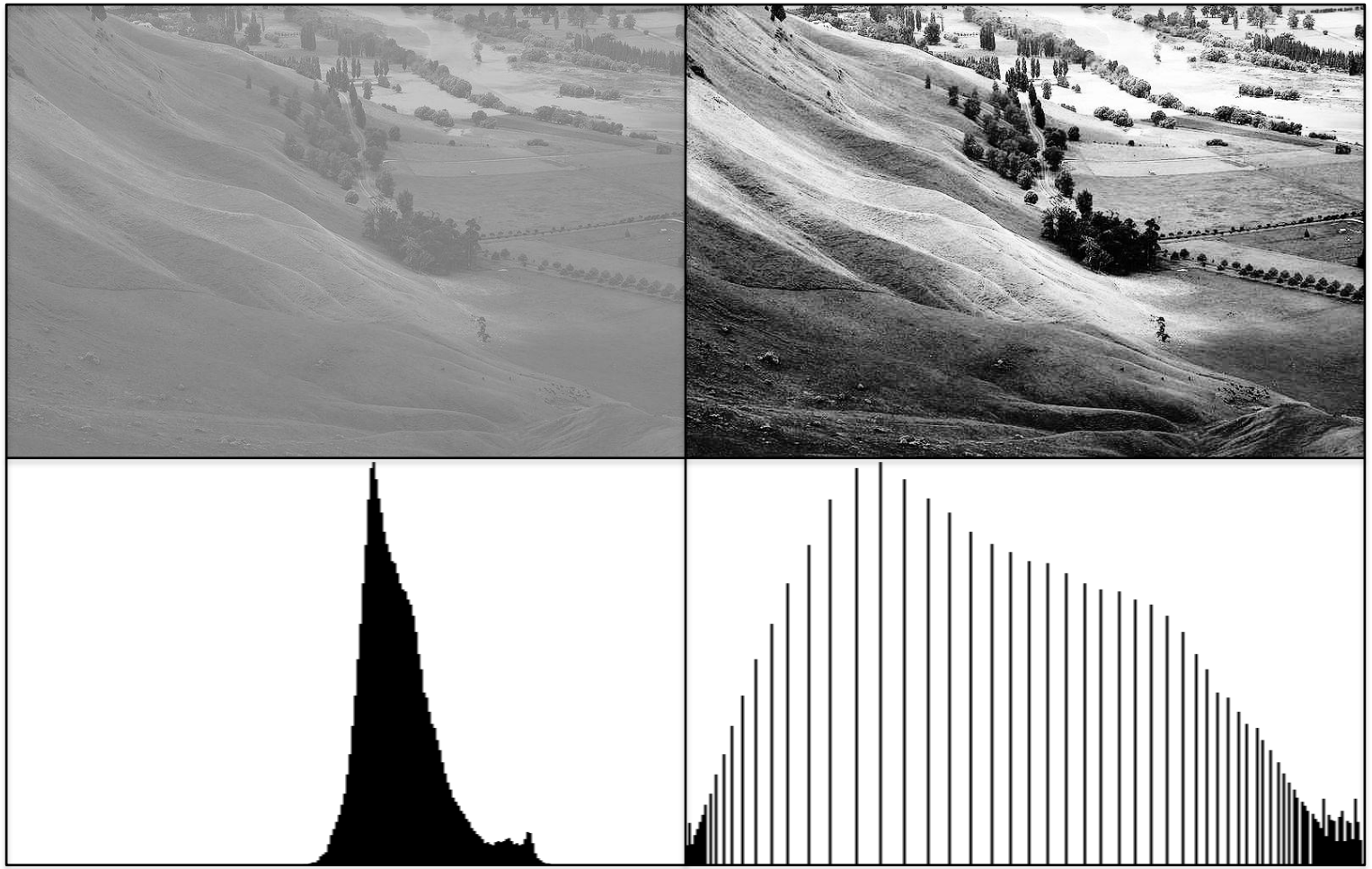
We want to re-assign a new grey level such that they span the entire interval [ ]:

,

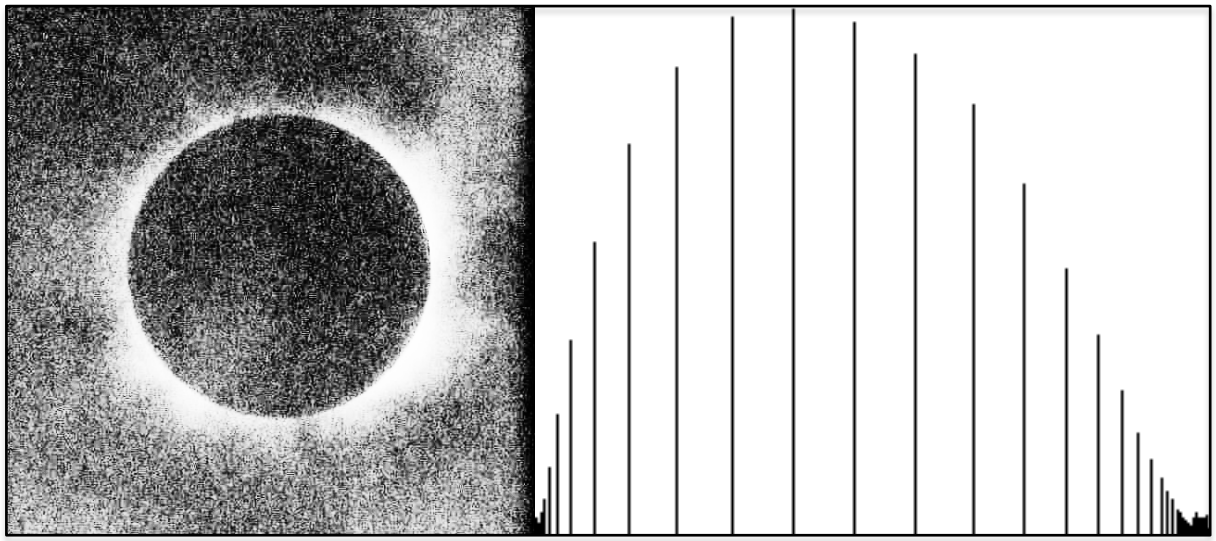
———( ) ,

Where is the smallest grey level present in the image

Histogram equalisation

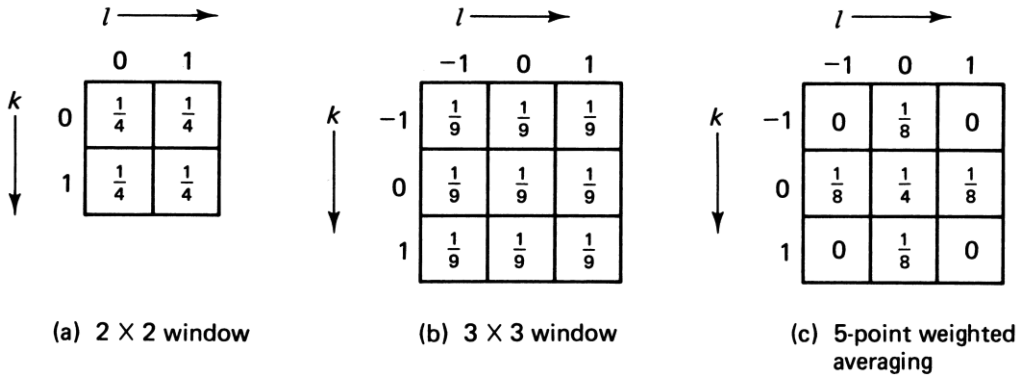


Histogram equalisation – gone wrong



Spatial operators or filtering

- Spatial operators transform a pixel based on its local neighbourhood.
- Mathematically speaking, the image is convoled with a finite impulse response filter the spatial mask .

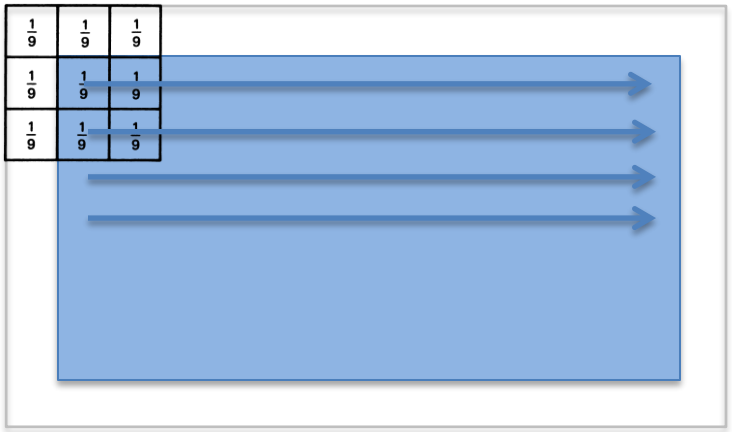


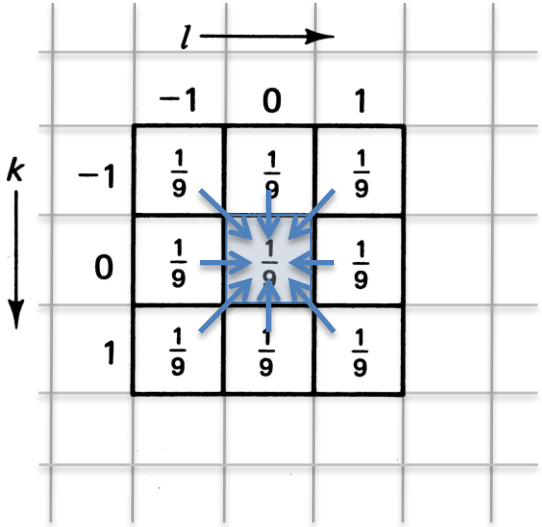
(a) 2 × 2 window (b) 3 × 3 window (c) 5-point weighted averaging

Spatial operators or filtering

Each pixel is replaced by a weighted average of its neighbourhood pixels:

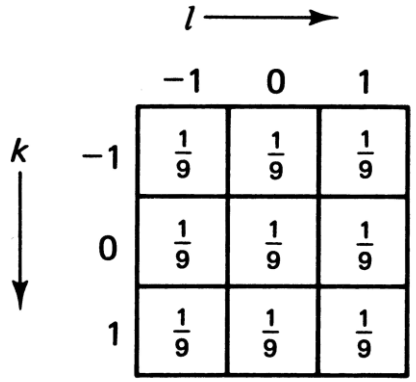
( ) ( )





Noise filtering or smoothing

Spatial averaging smoothes the noise of an image.
Suppose we have an image with some noise
( ) with zero mean and variance :
( ) ( ) ( ) .
The spatial average yields:

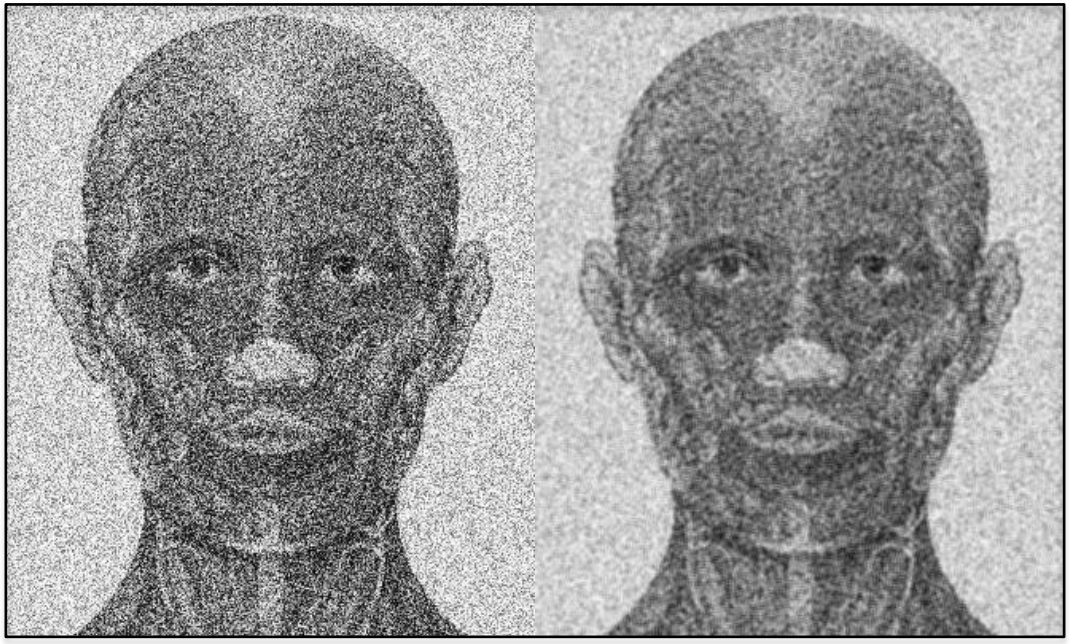


( ) — ( ) ( )

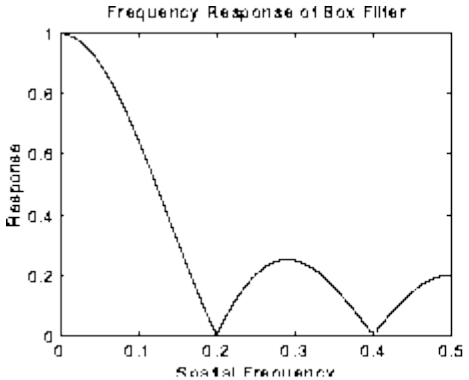
Where is the number of pixels of the filter kernel and is the resulting noise with zero mean and variance — .

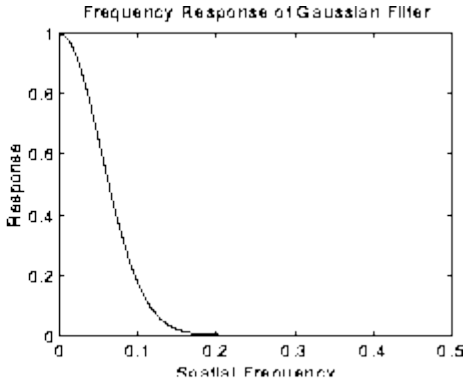
That is the noise power is reduced by the size of the filter kernel.

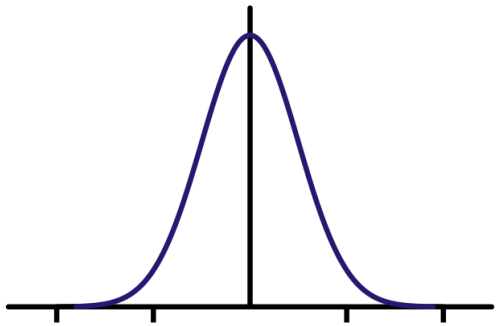
Noise filtering or smoothing

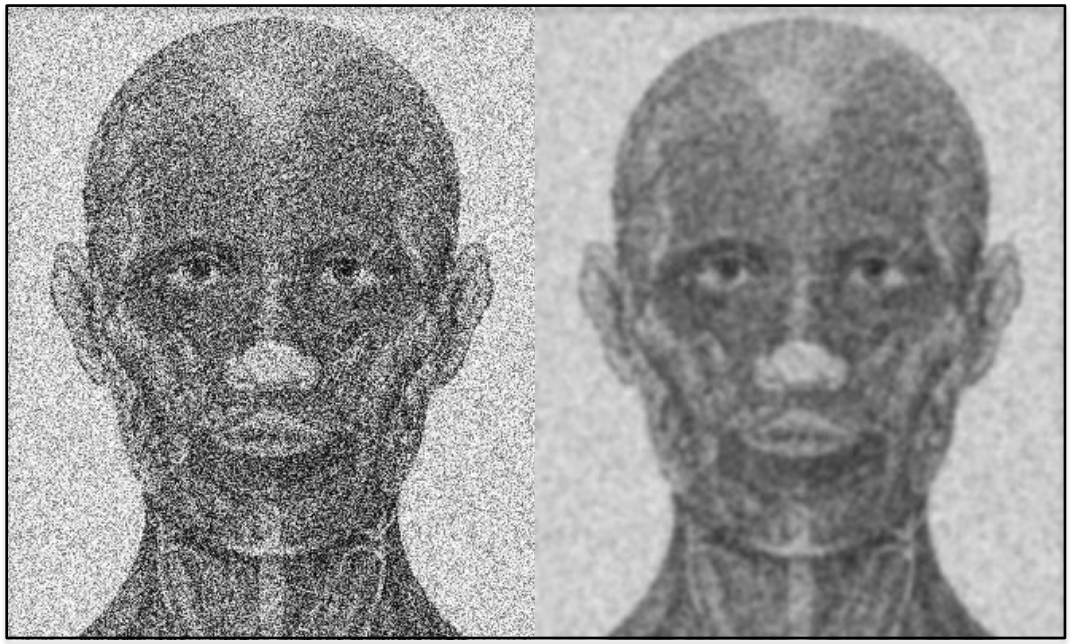


Gaussian noise filter









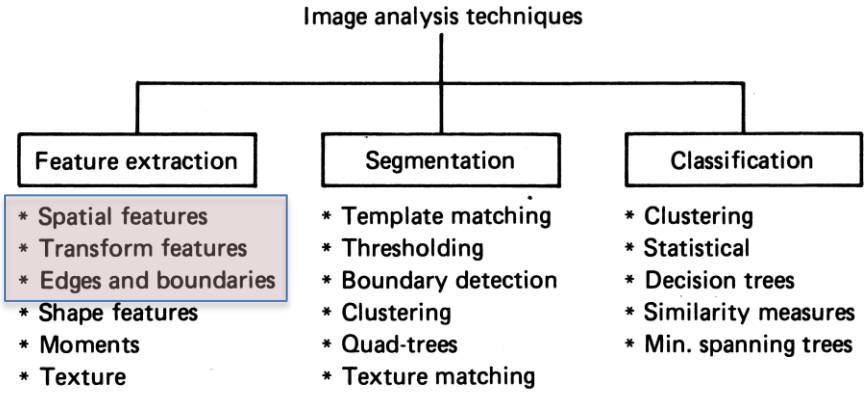
$$\frac{1}{273}$$

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

Image analysis

Image analysis reaches beyond image enhancement and tries to extract features such as object boundaries and their shapes from images.

We will try to detect the edges of objects in an image.



Edge detection

Edges characterise boundaries of objects in a scene. Finding object boundaries is fundamental for registration, and identification of objects.

We can think of an edge as an abrupt change in the grey level of neighbouring pixels and we can use the gradient of it to detect edges:

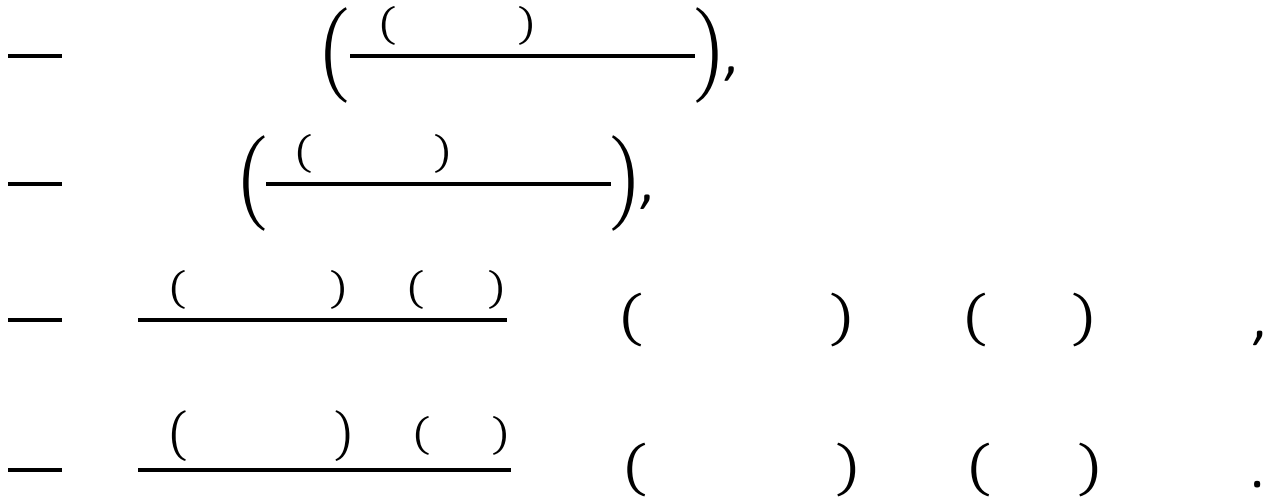
$$\left(\begin{array}{c} \\ \\ \end{array}\right),$$

$$(\quad) \quad \sqrt{(-)\quad(-)},$$

$$(\quad) \qquad \left(\begin{array}{c} \\ \\ \\ \end{array}\right).$$

Edge detection

Since digital images are discrete, we need to approximate the gradient using finite differences or central differences:



Edge detection

The Roberts edge detection operator implements this but along the diagonal.

Other operators can be derived using central differences.

The Sobel operator has

two steps:

- Smoothes the image with triangular kernel 1,2,1,
- Approximates the gradient.

Commonly used gradient operator masks:

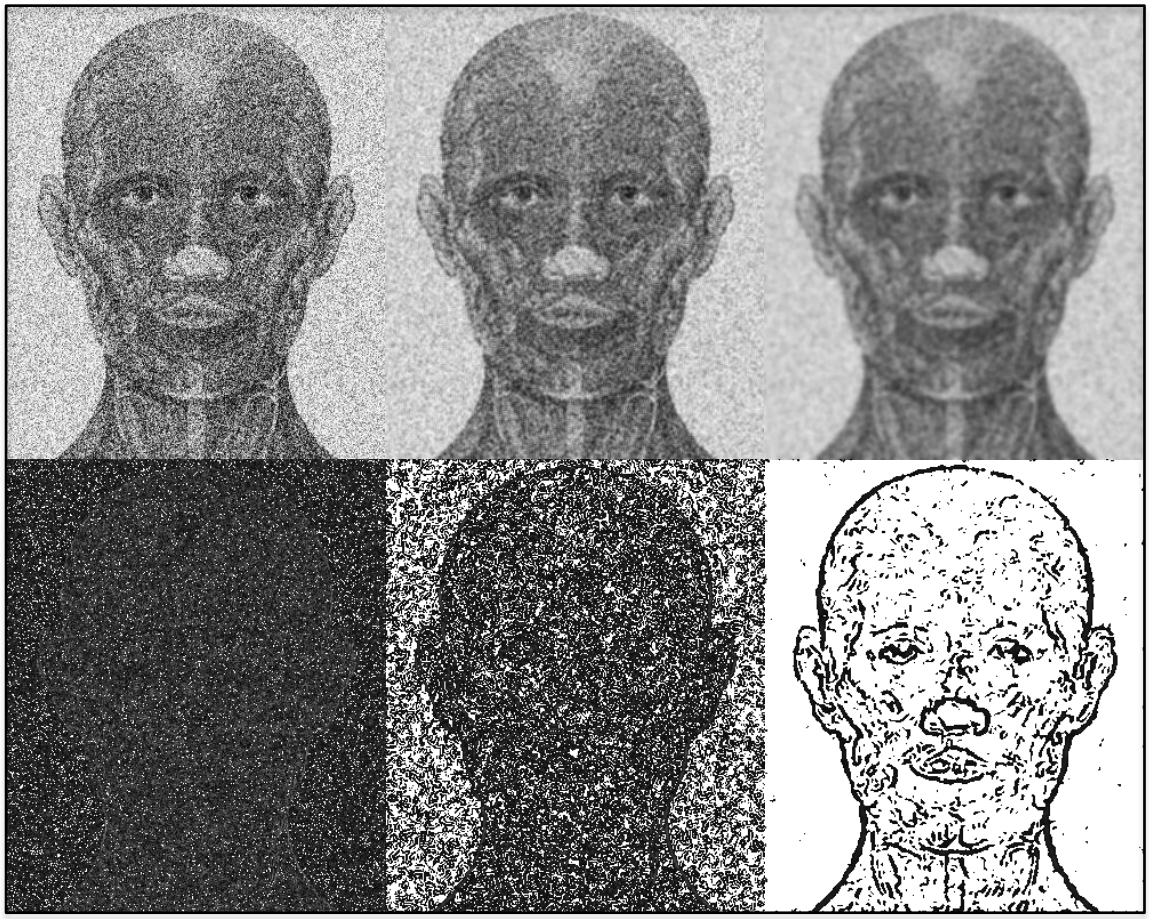
	H_1	H_2
Roberts	$\begin{bmatrix} \boxed{0} & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} \boxed{1} & 0 \\ 0 & -1 \end{bmatrix}$
Smoothed (Prewitt)	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & \boxed{0} & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & \boxed{0} & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Isotropic	$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & \boxed{0} & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & \boxed{0} & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

Edge detection

Main steps in edge detection using masks:

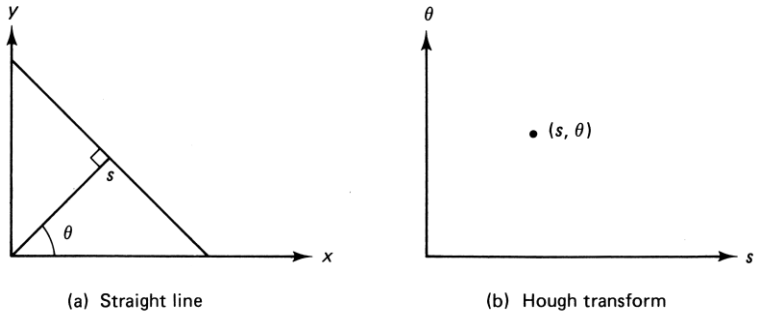
1. Approximate by computing the gradient in direction by filtering the image with the mask , and direction with .
2. Compute the magnitude of the gradient ( ),
3. Compute the direction of the gradient ( ),
4. Pixel is an edge point if ( ) where is a chosen threshold.

Effect of noise on edge detection



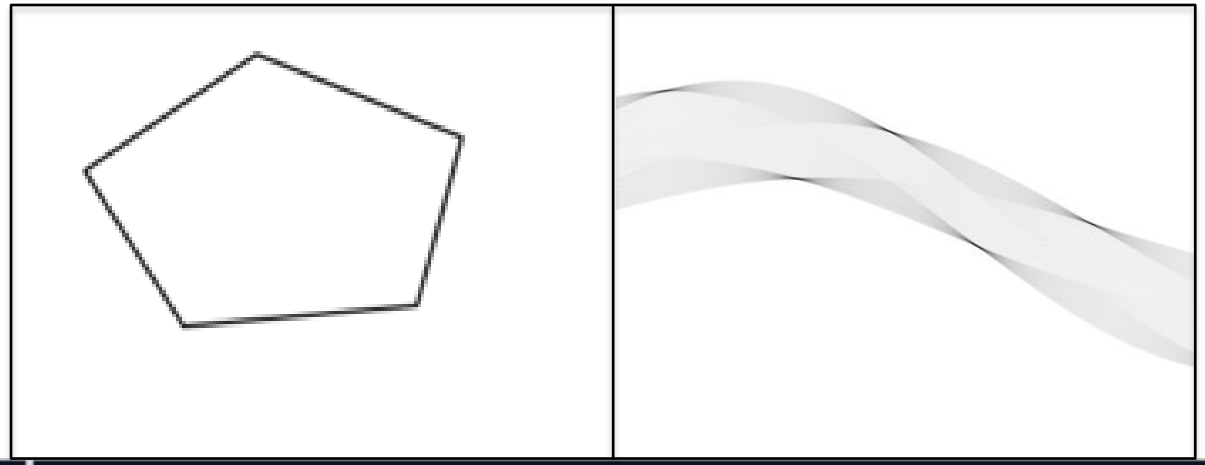
Hough transform

- A technique to extract straight lines form the detected edges.



(a) Straight line (b) Hough transform

Hough transform



Notes on the exercises

You will get working source code that is able to allocate, read, and write an ASCII encoded PGM image as well as code to transform a histogram into an image. The data types are:

```
typedef struct {
    unsigned int** int_image_data;
    unsigned int uint_xres;
    unsigned int uint_yres;
    unsigned int uint_max;
} image;

typedef struct {
    unsigned int uint_num_bins;
    unsigned int* uint_bins;
} histogram;
```

Notes on the exercises

The function prototypes are:

```
int read_image_p2(char* char_name, image* image_input);
int write_image_p2(char* char_name, image* image_output);
int allocate_image_p2(image* image_p2, unsigned int uint_xres,
    unsigned int uint_yres, unsigned int uint_greylevel);
void free_image_P2(image* image_p2);
void display_image_p2(image* image_p2);
void clone_image_p2(image* image_parent, image* image_child);
```

All functions are strictly, except for primitive data types, call by reference. So you have to make sure to allocate memory for any image or histogram!