Digital Image Processing

Ming Jiang

Revision : 1.1. June 5, 2009

School of Mathematical Sciences, Peking University

Contents

Chapter 1

Introduction and Course Overview

1.1 Digital image processing: What, Why and How

1.1.1 What is and why we need digital image processing

- Image is better than any other information form for our human being to perceive. Vision allows humans to perceive and understand the world surrounding us.

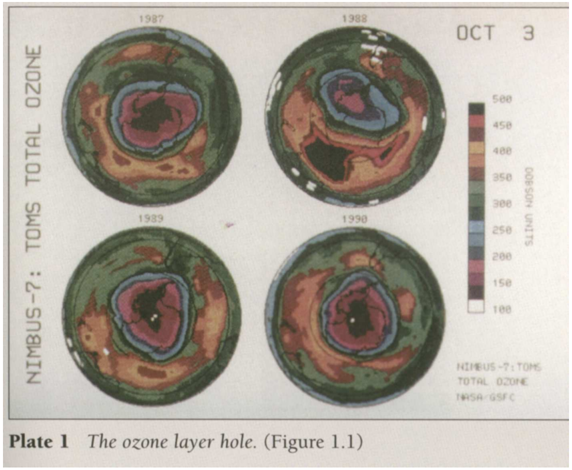
- Human are primarily visual creatures. Not all animals depend on their eyes, as we do, for 99% ([Russ, 1995]) (some says it is 90% [hua Zhao and hua Zhong, 1982]) or more of the information received about the world.

- Giving computers the ability to see is not an easy task — we live in a three dimensional (3D) world, and when computers try to analyze objects in 3D space, available visual sensors (e.g., TV cameras) usually give two dimensional (2D) images, and this projection to a lower number of dimensions incurs an enormous loss of information. Dynamic scenes such as those to which we are accustomed, with moving objects or a moving camera, make this task even more complicated.

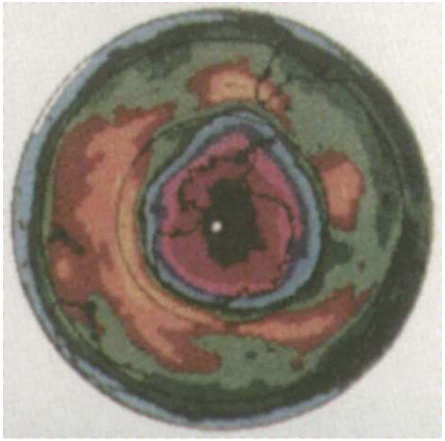
- Image understanding, image analysis, and computer vision aim to duplicate the effect of human vision by electronically ($=$ digitally, in the present context) perceiving and understanding image(s).

Let's first look at an example in § 1.1.2, which illustrate a few common issues.

1.1.2 Example: Detection of Ozone Layer Hole



(a) An ozone layer hole over Antarctica.



(b) An ozone layer image in 1990.

The image in Fig. 1.1(a) shows the ozone () layer hole over Antarctica, as captured by apparatus on board NASA's Nimbus 7 satellite over a period of years: 'TOMS' here stands for Total Ozone Mapping Spectrometer, and Dobson units are standard units (multiple of molecules per cubic centimeter) used used in ozone mapping — the normal value is about 300 Dobson units and so depletion () is obvious in these pictures.

It is natural to develop methods by which pictures like this may be analyzed **automatically** — the **qualitative** conclusion that there is a trend toward ozone depletion should be available from the changes in colors between the successive pictures, and we might also hope for some **quantitative** conclusions saying exactly (can we?) how much change is occurring. In fact, though, these pictures contain **formidable** quantities of information that make them far from trivial to analyze.

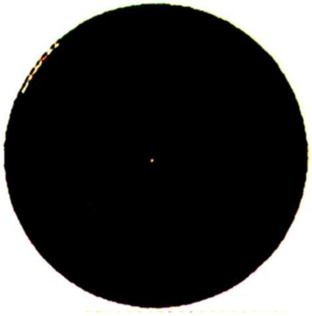
In the picture presented, a great deal of computer processing has already been applied: intensity normalization (normalization of the differences due to different scene environment conditions), geometric transformation, and a number of other filters which will be described in the course.

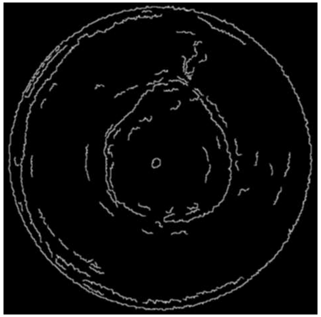
Example 1.1.1. *Delimit accurately (can we?) the ozone regions of different Dobson units in the picture and then draw conclusions about overall concentrations and trends.*

Let us look at the most latest picture at 1990 in Fig. 1.1(b).

The Khoros workspace for this example is here ozone example.

The result by thresholding is in Fig. 1.1(c). The result by using "Optimal Difference Recursive Filter for Edge Detection" is in Fig. 1.1(d).


(c) Image obtained by thresholding.


(d) Image obtained by using the "Optimal Difference Recursive Filter for Edge Detection".

We see that

- *While some of the clear evidence in the original is still visible, not all of it. There are a* **significant amount** *of further work to do on the early processing before being able to proceed.*
- *Several of the region boundaries are incomplete, while some significant information has disappeared altogether.*

These are problems that will be discussed and solutions presented in this course.

Following this stage would come **higher-level** *analysis in which values derived from the scale on the right of Fig. 1.1(a) would be attached to the region of Fig. 1.1(d). We would make reference to the color detected with the region located (the spectrometer information about its spectrum), and would probably deploy high-level ("domain") knowledge, such as expecting ozone concentration to vary continuously across the globe.*

This sequence of operations — image capture, early processing, region extraction, region labeling, high-level identification, qualitative/quantitative conclusion — is characteristics of image understanding and computer vision problems.

1.2 General Procedures

- There are philosophically two approaches: bionics () and engineering (that is project attempt coordinated), approaches. The bionics approach has not been so successful, since we do have a through understanding about the biological vision system.

- In order to simplify the tasks, two levels are generally, usually and universally recognized and distinguished: low-level image processing and high-level image understanding or computer vision. The goal is to obtain similar results to those provided by biological systems.

- Low-level methods usually use very little knowledge about the content or semantics of images.

- High-level processing is based on knowledge, goals, and plans of how to achieve those goals. High-level computer vision tries to imitate human cognition and the ability to make decisions according to the information contained in the image.

- Low-level image processing and high-level computer vision differ in the data used, Low-level data are comprised of original images represented by **matrices** composed of brightness values, while high-level data originates in images as well, but only those data which are relevant to high-level goal are extracted, reducing the data quantity considerably. High-level data represent knowledge about the image content — for example, object size, shape and mutual relations among objects in the images. High-level data are usually represented in symbolic form.

- This course deals almost exclusively with low-level image processing.

1.2.1 Low-level image processing

Low-level image processing techniques overlap almost completely with digital image processing, which has been practiced for decades. Most current low-level image processing methods were proposed in the 1970s or earlier. Recent research is trying to find more efficient and more general algorithms and implement them on more technologically sophisticated equipment — in particular, parallel machines are being used to ease the enormous computational load of operations conducted on image date sets.

A complicated and so far **unsolved problem** is how to order low-level steps to solve a specific task, and the aim of **automating** this problem has not yet been achieved. It is not surprising that in 1980s many projects focused on this problem using expert systems, but expert systems do not solve the problem by themselves.

The following sequence of processing steps is commonly recognized:

- Image Acquisition: An image is captured by a sensor (such as a TV camera) and digitized. Image may come in many **formats** and ways.
- Preprocessing: Image reconstruction or restoration, denoising and enhancement. E.g., computer tomography.
- Image coding and compression: this is important for transferring images.
- Image segmentation: computer tries to separate objects from the image background.
- Object description and classification in a totally segmented image is also understood as part of low-level image processing.

1.2.2 High-level image processing

High-level Processing is based on knowledge, goals and plans of how to achieve those goals and artificial intelligence methods are widely applicable. High-level computer vision tries to imitate human cognition and the ability to make decisions according to the information contained in the image.

In the ozone example above, high-level knowledge would be the continuity of the concentration figures and the fact that the area of similar concentration appear as (distorted) annuli centered at the polar area.

High-level vision begins with some form of formal **model** of the world, and then the **reality** perceived in the form of digitized images is compared to the model. A match is attempted, and when difference emerge, partial matches (or sub-goals) are sought that overcome the mismatch; the computer switches to low-level image processing to find information needed to update the model. This process is the repeated iteratively, and **understanding** an image thereby becomes a co-operation between top-down and bottom-up processes. A **feedback** loop is introduced in which high-level partial results and knowledge create task for low-level image processing, and the iterative image understanding process should eventually converge to the global goal.

Unsurprising this process is very complicated and computation intensive.

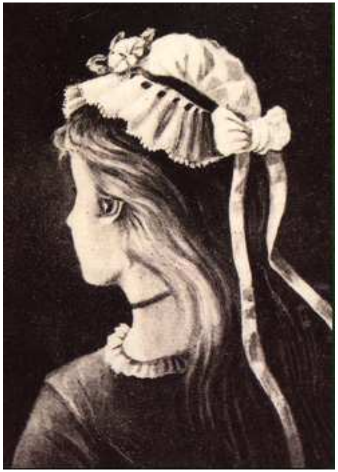
1.3 Digital Image Processing and Other Disciplines

Many techniques of image processing, image understanding and analysis, and computer vision use the results and methods of mathematics, pattern recognition, artificial intelligence, psychophysiology (), physiopsychology (), cognitive science, computer science, electronics and other scientific disciplines.

1.4 What Are the Difficulties

1.4.1 Poor understanding of the human vision system

Example 1.4.1. *How the human perceive process and store the visual information?*



Explanation and information from Home of Vision Illusion.

We do not have a clear understanding how the human perceive, process and store the visual information. We do not even know how the human measures internally the image visual quality and discrimination.

Do you see an old woman or a young woman in this illustration? They are both present, but you will not be able to see both of them simultaneously. Once you perceive both figures, see if you can get them to fluctuate back and forth between the two interpretations.

This type of reversible figure concerns the meaningful content of what is interpreted by your brain from the same static image. Your perception of each figure tends to remain stable until you attend to different regions or contours. Certain regions and contours tend to favor one perception, others the alternative.

Your visual system tends to group like or related regions together. It does not present you with some odd mixture of the two alternatives.

Attending to different regions or contours does tend to initiate a change of perception.

If this image is looked at with a steady eye, it will still change, though less often. Researchers have stabilized the image directly onto the retina to eliminate any effects that may arise from eye movements. Even under these conditions, a perceptual reversal may occur. This indicates that higher cortical processing occurs that strives to make meaning out of a stable image presented to the retina. This illustrates once more that vision is an active process that attempts to make sense of incoming information. As the late David Marr said, “Perception is the construction of a description.”

History of this illustration

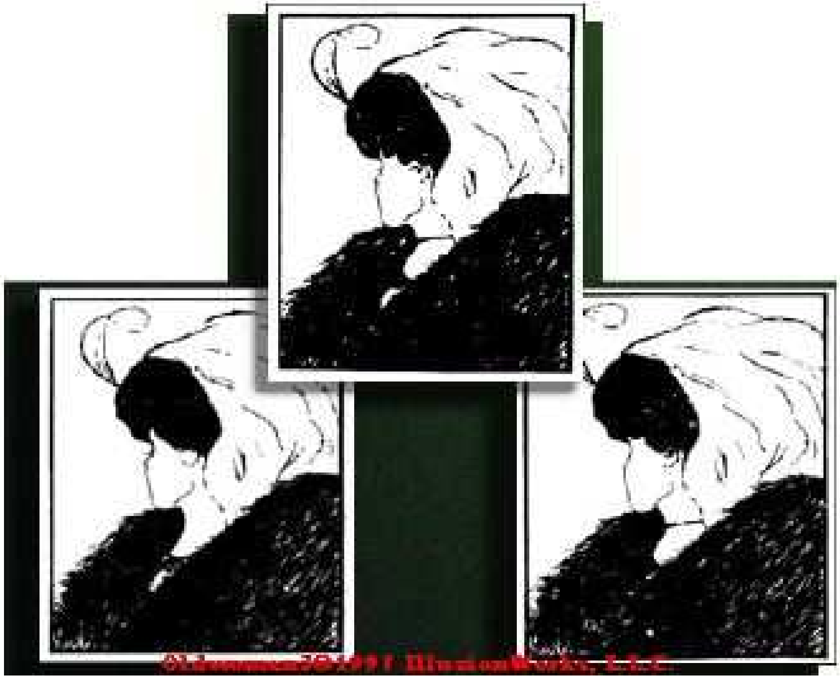
For many years the creator of this famous figure was thought to be British cartoonist W. E. Hill, who published it in 1915. Hill almost certainly adapted the figure from an original concept that was popular throughout the world on trading and puzzle cards.

This anonymous dated German postcard (shown at the top of the page) from 1888 depicts the image in its earliest known form.

The 1890 example on the left shows quite clearly its association as "My Wife and Mother-in-Law." Both of these examples predate the Punch cartoon that was previously thought to serve as the figure's inspiration.

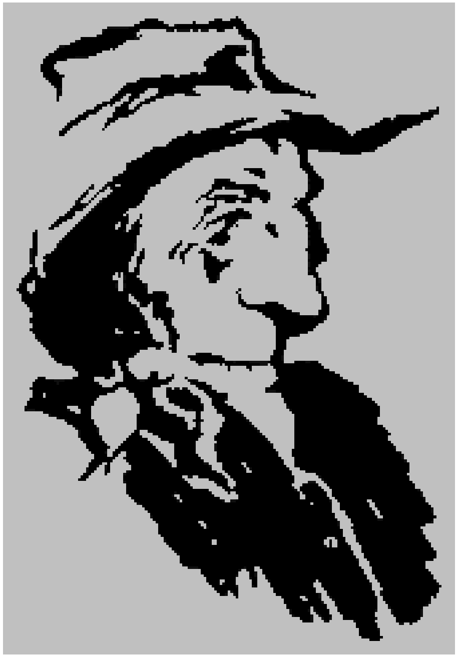
The figure was later altered and adapted by others, including the two psychologists, R. W. Leeper and E. G. Boring who described the figure and made it famous within psychological circles in 1930. It has often been referred to as the "Boring figure."

Versions of the figure proved to be popular and the image was frequently reprinted; however, perceptual biases started to occur in the image, unbeknownst to the plagiarizing artists and psychologists who were reprinting the images. Variations have appeared in the literature that unintentionally are biased to favor one interpretation or another, which defeats its original purpose as a truly ambiguous figure.



In the three versions shown above, can you tell which one is biased toward the young girl, the old woman?

In 1961, J, Botwinick redesigned this figure once again, and entitled it, "Husband and Father-in-Law."



1.4.2 Internal representation is not directly understandable

Images are usually represented as a two dimensional function. Digitized images are usually represented by two dimensional array. However, those representations are not suitable for machine understanding, while the computer is able to process those representations. General knowledge, domain-specific knowledge, and information extracted from the image will be essential in attempting to **understanding** those arrays of numbers.

Example 1.4.2. *Internal image representations for computer are not directly understandable.*

Given an matrix, what is the perceived information by human?

Do the following exercise:

- *Read and display a image file as a two dimensional function. The example matlab script file is here matlab display example.*

Both presentation contain exactly the same information, but for a human observer it is very difficult to find a correspondence between, and without the second, it is unlikely that one would recognize the child. The point is that a lot of a priori knowledge is used by humans to interpret the images; the machine only begins with an array of numbers and so will be attempting to us are more like the first display then the second display.

1.5 Course Overview

1.5.1 Course syllabus

Digital image processing, image analysis, image understanding are related branches of computer vision. This course is about digital image processing.

1.5.2 Textbooks and references

The main textbook is

- Milan Sonka, V. Hlavac, R. Boyle: Image Processing, Analysis and Machine Vision. Brooks/Cole Publishing Company, 1999.

Two main reference books are:

A. Kenneth R. Castleman: Digital Image Processing. Prentice-Hall International, Inc. 1996. Or Tsinghua University Press, 1998.

B. Rongchun Zhao: Digital Image Processing (in Chinese). Northwestern Polytechnical University Press, 1996.

Chapter 2

The Digitized Image and its Properties

2.1 Basic concepts

- Fundamental concepts and mathematical tools are introduced in this chapter which will be used throughout the course.
- Mathematical models are often used to describe images and other signals.
- A signal is a function depending on some variable with physical meaning. Signals can be
  - one-dimensional (e.g., audio signal dependent on time);
  - two-dimensional (e.g., images dependent on two co-ordinates in a plane);
  - three-dimensional (e.g., describing an object in space or video signal);
  - or higher-dimensional.
- A scalar function may be sufficient to describe a monochromatic image, while vector functions are to represent, for example, color images consisting of three component colors.
- Functions we shall work with may be categorized as continuous, discrete and digital. A continuous function has continuous domain and range; if the domain set is discrete, then we get a discrete function; if the range set is also discrete, then we have a digital function.

2.1.1 Image functions

- The image can be modeled by a continuous function of two or three variables; arguments are co-ordinates x and y in a plane, while if images change in time a third variable t might be added.
- The image function values correspond to the brightness at image points.
- The function value can express other physical quantities as well (temperature, pressure distribution, distance from the observer, etc.).
- The brightness integrates different optical quantities — using brightness as a basic quantity allows us to avoid the description of the very complicated process of image formation.

- The image on the human eye retina or on a TV camera sensor is intrinsically 2D. We shall call such a 2D image bearing information about brightness points an **intensity image**.
- The real world which surrounds us is intrinsically 3D.
- The 2D intensity image is the result of a perspective projection of the 3D scene.
- When 3D objects are mapped into the camera plane by perspective projection a lot of information disappears as such a transformation is not one-to-one.
- Recognizing or reconstructing objects in a 3D scene from one image is an ill-posed problem.
- Recovering information lost by perspective projection is only one, mainly geometric, problem of computer vision. The aim is to recover a full 3D representation such as may be used in computer graphics.
- The second problem is how to understand image brightness. The only information available in an intensity image is brightness of the appropriate pixel, which is dependent on a number of independent factors such as object surface reflectance properties (given by the surface material, micro-structure and

marking), illumination properties, and object surface orientation with respect to a viewer and light source. This is a non-trivial and again ill-posed problem.

- Some scientific and technical disciplines work with 2D images directly; for example,
  - an image of the flat specimen viewed by a microscope with transparent illumination;
  - a character drawn on a sheet of paper;
  - the image of a fingerprint, etc.
- Many basic and useful methods used in digital image analysis do not depend on whether the object was originally 2D or 3D (e.g, FFT).

- The image formation process is described in [Horn, 1986, Wang and Wu, 1991].
- Related disciplines are photometry which is concerned with brightness measurement, and colorimetry which studies light reflectance or emission depending on wavelength.
- A light source energy distribution $C(x, y, t, \lambda)$ depends in general on image co-ordinates (x, y), time t, and wavelength λ.
- For the human eye and most technical image sensors (e.g., TV cameras), the "brightness" f depends on the light source energy distribution C and the spectral sensitivity of the sensor, $S(\lambda)$ (dependent on the wavelength)

  $$f(x, y, t) = \int_0^\infty C(x, y, t, \lambda) S(\lambda)\, d\lambda \tag{2.1}$$

  An intensity image $f(x, y, t)$ provides the brightness distribution.
- In a color or multi-spectral image, the image is represented by a real vector function f

  $$f(x, y, t) = (f_1(x, y, t), f_2(x, y, t), \cdots, f_n(x, y, t)) \tag{2.2}$$

  where, for example, there may be red, green and blue, three components.

- Image processing often deals with static images, in which time t is constant.
- A monochromatic static image is represented by a continuous image function f(x,y) whose arguments are two co-ordinates in the plane.
- Most methods introduced in this course is primarily for intensity static image. It is often the case that the extension of the techniques to the multi-spectral case is obvious.

- Computerized image processing uses digital image functions which are usually represented by matrices, so co-ordinates are integer numbers. The domain of the image function is a region R in the plane

$$R = \{(x, y) : 1 \leq x \leq x_m, 1 \leq y \leq y_n\} \tag{2.3}$$

  where x_m and y_n represent maximal image co-ordinates.

- The image function has a limited domain — infinite summation or integration limits can be used, as it is assumed that the image function is zero outside the domain.

- The customary orientation of co-ordinates in an image is in the normal Cartesian fashion (horizontal x axis, vertical y axis), although the (row, column) orientation used in matrices is also quite often used in digital image processing.

- The range of image function values is also limited; by convention, in intensity images the lowest value corresponds to black and the highest to white.

- Brightness values bounded by these limits are gray levels.

- The gray level range is $0, 1, \cdots, 255$, represented by 8 bits, the data type used is **unsigned char**. In some applications, 14 bits or more is used, e.g, for medical images.

- The usual computer display supports 8 bit gray level.

Homework 2.1.1. *How to display a 16 bit gray level image? Generate an image of 16 bit and try to display it with your computer.*

Homework 2.1.2. *If a discrete image is of continuous range, the image matrix is of type* **float** *or* **double***. How to display it? Generate an image of* **float** *or* **double** *type and try to display it with your computer.*

- The quality of a digital image grows in proportion to the spatial, spectral, radiometric, and time resolution.
- The spatial resolution is given by the proximity of image samples in the image plane.
- The spectral resolution is given by the bandwidth of the light frequencies captured by the sensor.
- The radiometric (contrast, or density) resolution corresponds to the number of distinguishable gray levels.
- The time resolution is given by the interval between time samples at which images are captured.

2.2 Image digitization

- An image captured by a sensor is expressed as a continuous function $f(x, y)$ of two co-ordinates in the plane.
- Image digitization means that the function $f(x, y)$ is sampled into a matrix with M rows and N columns.
- The image quantization assigns to each continuous sample an integer value. The continuous range of the image function $f(x, y)$ is split into K intervals.
- The finer the sampling (i.e., the larger M and N) and quantization (the larger K) the better the approximation of the continuous image function $f(x, y)$.
- Two questions should be answered in connection with image function sampling:
  - First, the sampling period should be determined – the distance between two neighboring sampling points in the image;
  - Second, the geometric arrangement of sampling points (sampling grid) should be set.

2.2.1 Sampling

- A continuous image function $f(x, y)$ can be sampled using a discrete grid of sampling points in the plane.

- The image is sampled at points $x = j\Delta x$, $y = k\Delta y$

- Two neighboring sampling points are separated by distances Δx along the x axis and Δy along the y axis. Distances Δx and Δy are called the sampling interval and the matrix of samples constitutes the discrete image.

- The ideal sampling $s(x, y)$ in the regular grid can be represented using a collection of Dirac distributions

$$s(x, y) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \delta(x - j\Delta x, y - k\Delta y) \tag{2.4}$$

- The sampled image is the product of the continuous image $f(x, y)$ and the sampling function $s(x, y)$

$$f_s(x, y) = s(x, y) f(x, y) \tag{2.5}$$

- The collection of Dirac distributions in equation (2.4) can be regarded as periodic with period Δx, Δy and expanded into a Fourier series (assuming for a moment that the sampling grid covers the whole plane (infinite limits))

$$s = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} a_{mn} \mathbf{e}^{2\pi i \left(\frac{mx}{\Delta x} + \frac{ny}{\Delta y}\right)} \tag{2.6}$$

  where the coefficients of the Fourier expansion can be calculated as

$$a_{mn} = \frac{1}{\Delta x \Delta y} \int_{-\frac{\Delta x}{2}}^{\frac{\Delta x}{2}} \int_{-\frac{\Delta y}{2}}^{\frac{\Delta y}{2}} \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \delta(x - j\Delta x, y - k\Delta y) \mathbf{e}^{2\pi i \left(\frac{mx}{\Delta x} + \frac{ny}{\Delta y}\right)} \, dxdy \tag{2.7}$$

- Noting that only the term for $j = 0$ and $k = 0$ in the sum is nonzero in the range of integration (for other j and k, the center of the Delta function is outside the integral interval), the coefficients are

$$a_{mn} = \frac{1}{\Delta x \Delta y} \int_{-\frac{\Delta x}{2}}^{\frac{\Delta x}{2}} \int_{-\frac{\Delta y}{2}}^{\frac{\Delta y}{2}} \delta(x, y) \mathbf{e}^{2\pi i \left(\frac{mx}{\Delta x} + \frac{ny}{\Delta y}\right)} \, dxdy = \frac{1}{\Delta x \Delta y} \tag{2.8}$$

- Then, (2.5) can be rewritten as

$$f_s(x, y) = f(x, y) \frac{1}{\Delta x \Delta y} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \mathbf{e}^{2\pi i \left(\frac{mx}{\Delta x} + \frac{ny}{\Delta y}\right)} \tag{2.9}$$

- In frequency domain then

$$F_s(u,v) = \frac{1}{\Delta x \Delta y} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F(u - \frac{m}{\Delta x}, v - \frac{n}{\Delta y}) \tag{2.10}$$

  where F and F_s are the Fourier transform of f and f_s respectively.

- Recall the Fourier transform is

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \mathbf{e}^{-2\pi i (ux+vy)}\, dxdy \tag{2.11}$$

- Thus the Fourier transform of the sampled image is the sum of periodically repeated Fourier transforms $F(u,v)$ of the origin image.

- Periodic repetition of the Fourier transform result $F(u,v)$ may under certain conditions cause distortion of the image which is called aliasing; this happens when individual digitized components $F(u,v)$ overlap.

- There is no aliasing if the image function $f(x, y)$ has a band limited spectrum, its Fourier transform $F(u, v) = 0$ outside a certain interval of frequencies $|u| > U$ and $|v| > V$.

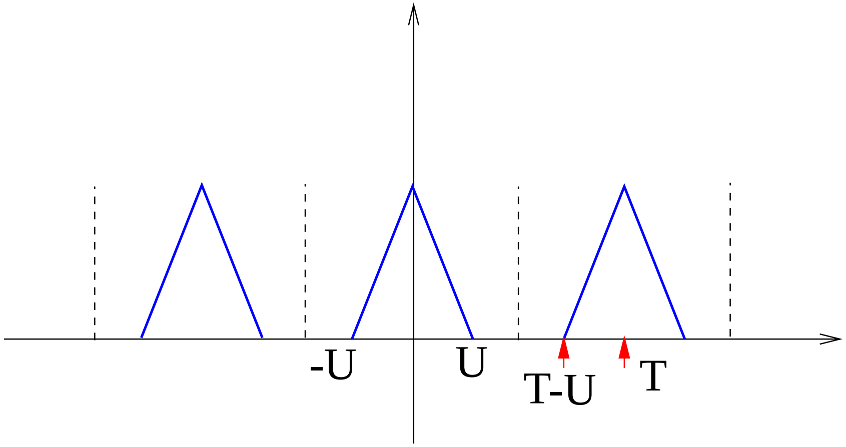


Figure 2.1: Where $T = \frac{1}{\Delta x}$.

- As you know from general sampling theory [Oppenheim et al., 1997], overlapping of the periodically repeated results of the Fourier transform $F(u, v)$ of an image with band limited spectrum can be prevented if the sampling interval is chosen according to

$$\Delta x \leq \frac{1}{2U}, \qquad \Delta x \leq \frac{1}{2V}. \tag{2.12}$$

- This is the Shannon sampling theorem that has a simple physical interpretation in image analysis: The sampling interval should be chosen in size such that it is less than or equal to half of the smallest interesting detail in the image.

- The sampling function is not the Dirac distribution in real digitizers – narrow impulses with limited amplitude are used instead.

- Assume a rectangular sampling grid which consists of $M \times N$ such equal and non-overlapping impulses $h_s(x, y)$ with sampling period Δx and Δy. Ideally, $h_s(x, y) = \delta(x, y)$. The function $h_s(x, y)$ simulates realistically the real image sensors. Outside the sensitive area of the sensor, the sampling element $h_s(x, y) = 0$. The sampled image is then given by the convolution computed in direct co-ordinates $x = j\Delta x$, $y = k\Delta y$,

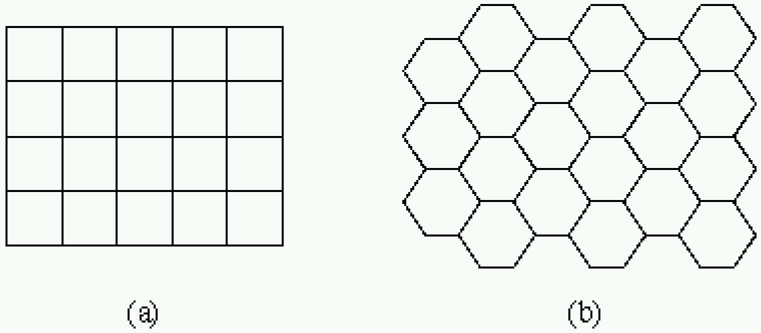
$$f_s(x, y) = f(x, y) \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} h_s(x - j\Delta x, y - k\Delta y) \tag{2.13}$$

- The sampled image f_s is distorted by the convolution of the original image f and the limited impulse h_s. The distortion of the frequency spectrum of the function F_s can be expressed using the Fourier transform

$$F_s(u, v) = \frac{1}{\Delta x \Delta y} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F(u - \frac{m}{\Delta x}, v - \frac{n}{\Delta y}) H_s(\frac{m}{\Delta x}, \frac{n}{\Delta y}). \tag{2.14}$$

Homework 2.2.1. *Prove (2.14) from (2.13).*

- There are other sampling schemes.
- These sampling points are ordered in the plane and their geometric relation is called the grid. Grids used in practice are mainly square or hexagonal



(a) Square grid, (b) hexagonal grid.

- One infinitely small sampling point in the grid corresponds to one picture element (pixel) in the digital image. The set of pixels together covers the entire image.

- Pixels captured by a real digitization device have finite sizes.
- The pixel is a unit which is not further divisible, sometimes pixels are also called points.

2.2.2 Quantization

- A magnitude of the sampled image is expressed as a digital value in image processing.
- The transition between continuous values of the image function (brightness) and its digital equivalent is called quantization.
- The number of quantization levels should be high enough for human perception of fine shading details in the image.
- The occurrence of false contours is the main problem in image which have been quantized with insufficient brightness levels. This effect arises when the number of brightness levels is lower than that which humans can easily distinguish.
- This number is dependent on many factors — for example, the average local brightness — but displays which avoids this effect will normally provide a range of at least 100 intensity levels.
- This problem can be reduced when quantization into intervals of unequal length is used; the size of intervals corresponding to less probable brightnesses in the image is enlarged. These gray-scale transformation techniques are considered in later sections.

- Most digital image processing devices use quantization into k equal intervals.
- If b bits are used ... the number of brightness levels is $k = 2^b$.
- Eight bits per pixel are commonly used, specialized measuring devices use 12 and more bits per pixel.

  Example 2.2.2. *Do quantization experiment. The Khoros workspace for this example is here quantization example.*

  Do you observe the false contours when the quantization levels is decreasing?

2.2.3 Color Images

- Color is a property of enormous importance to human visual perception.
- This is due to the biological structure of the human retina.
- But historically it has not been particularly used in digital image processing. The reason for this has been the cost of suitable hardware.
- Since 1980's, this has changed significantly. Color images are now routinely accessible via TV cameras or scanners. Color display is the default in most computer systems.
- Color is connected with the ability of objects to reflect electromagnetic waves of different wavelengths.
- The visible chromatic spectrum spans the electromagnetic spectrum from 400 nm to 700 nm.
- Human detect colors as combination of the **primary colors** red, green and blue, whose wavelength for the purpose of standardization have been defined as 700 nm, 546.1 nm and 435.8 nm, respectively, [Pratt, 1978].
- Hardware will generally deliver or display color via an RGB model.

- A particular pixel may have associate with it a three dimensional vector (r, g, b) which provides the respective color intensities.
- $(0, 0, 0)$ is black, (k, k, k) is white, $(k, 0, 0)$ is pure red, where k is the quantization level. Usually $k = 256 = 2^8$.
- Most image sensors will provide data according this model. The image can be captured by several sensors, each of which is sensitive to a rather narrow band of wavelengths, (2.1).
- Each spectral band is digitized independently and is represented by an individual digital image function as if it were a monochromatic image.
- Some images are delivered using a similar approach, but with a different complements — e.g., the LANDSAT 4 satellite transmits digital images in five spectral bands from near-ultraviolet to infrared.
- For our purpose this is useful, since a monochromatic image may not contain enough information for many applications, while color or **multi-spectral images** can often help.

Example 2.2.3. *Multi-spectral image captured by Land-sat TM-5. The example uses the images from its 3rd, 4th and 5th spectral bands. The Khoros workspace for this example is here Multi-spectral image example.*

- Other color models turn out to be equally important, if less intuitive.
- There are many of them: CMY, CMYK, YIQ, HSI, ...

Remark 2.2.4. *Be careful for formulas in various books.*

2.3 Digital image properties

A digital image has several properties, both metric and topological, which are somewhat different from those of continuous two-dimensional functions we are familiar with.

2.3.1 Metric and topological properties of digital images

- A digital image consists of picture elements of finite size.
- Usually pixels are arranged in a rectangular grid.
- A digital image is represented by a two-dimensional matrix whose elements are integer numbers corresponding to the quantization levels in the brightness scale.
- Some intuitively clear properties of continuous images have no straightforward analogy in the domain of digital images.

Metric properties of digital images

- Distance is an important example.
- The distance between two pixels in a digital image is a significant quantitative measure.
- The distance between points with co-ordinates (i,j) and (h,k) may be defined in several different ways:
- The Euclidean distance D_E is defined by

  $$D_E[(i,j),h,k] = \sqrt{(i-h)^2 + (j-k)^2} \qquad (2.15)$$

  The advantage of the Euclidean distance is the fact that it is intuitively obvious. The disadvantages are costly calculation due to the square root, and its not-integer value.
- The distance between two points can also expressed as the minimum number of elementary steps in the digital grid which are needed to move from the starting point to the end point.
- If only horizontal and vertical moves are allowed, the distance D_4 or city block distance is obtained:

  $$D_4[(i,j),h,k] = |i-h| + |j-k| \qquad (2.16)$$

which is the analogy with the distance between two locations in a city with a rectangular grid of streets and closed blocks of buildings.

- If moves in diagonal directions are allowed in addition, the distance D_8 or the chess-board distance is obtained:

$$D_8[(i,j),h,k] = \max\{|i-h|,|j-k|\} \tag{2.17}$$

Topological properties of digital images

- Pixel adjacency is another important concept in digital images.
- Any two pixels are called 4-neighbors if they have distance $D_4 = 1$ from each other. Analogously, 8-neighbors are two pixels with $D_8 = 1$.
- 4-neighbors and 8-neighbors are illustrated in Figure 2.3.1.

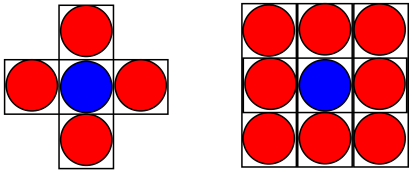


Figure 2.2: Pixel neighborhoods.

- It will become necessary to consider important sets consisting of several adjacent pixels — regions.
- **Region** is a contiguous (touching, neighboring, near to) set.

- A **path** from pixel P to pixel Q as a sequence of points $A_1, A_2, \cdots, A_n$, where $A_1 = P$ and $A_n = Q$, and A_{i+1} is a neighbor of A_i, $i = 1, \cdots, n-1$.

- A **region** is a set of pixels in which there is a path between any pair of its pixels, all of whose pixels also belong to the set.

- If there is a path between two pixels in the image, these pixels are called **contiguous**.

- The relation to be contiguous is reflexive, symmetric and transitive and therefore defines a decomposition of the set (in our case image) into equivalence classes (regions).

- Assume that R_i are disjoint regions in the image and that these regions do not touch the image boundary (to avoid special cases). Let R be the union of all regions R_i. Let R^C be the complement of R with respect to the image.

- The subset of R^C, which is contiguous with the image boundary, is called **background**, and the rest of the complement R^C is called **holes**.

- If there are no holes in a region we call it a **simply contiguous** region.

- A region with holes is called **multiply contiguous**.

- Note that the concept of region uses only the property to be contiguous. Secondary properties can be attached to regions which originate in image data interpretation. It is common to call some regions in the image **objects**.

- A process which determines which regions in an image correspond to objects in the world is part of **image segmentation**().

- E.g., the brightness of a pixel is a very simple property which can be used to find objects in some images.

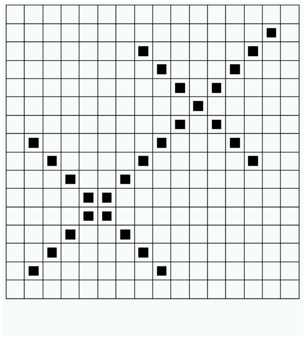
  If a pixel is darker than some other predefined values (threshold), then it belongs to some object. All such points which are also contiguous constitute one object. A hole consists of points which do not belong to the object and surrounded by the object, and all other points constitute the background.

- An example is the black printed text on the white paper, in which individual letters are objects. White areas surrounded by the letter are holes, e.g., the area inside a letter 'O'. Other which parts of the paper are background.

Contiguity paradoxes

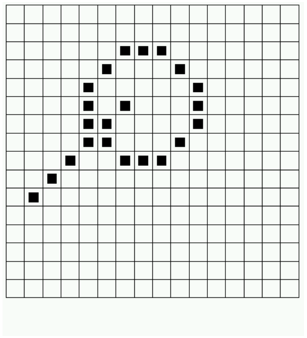
These neighborhood and contiguity definitions on the square grid create some paradoxes.

Example 2.3.1. *The following figure shows three digital lines with 45^o and -45^o slope.*



- *If 4-connectivity is used, the lines are not contiguous at each of their points.*
- *An even worse conflict with intuitive understanding of line properties is: two perpendicular lines do intersect in one case (upper right intersection) and do not intersect in another case (lower left), as they do not have any common point.*

Example 2.3.2. *The following figure shows another paradox. It is known from*



Euclidean geometry that each closed curve divides the plane into two non-contiguous regions. If image are digitized in a square grid using 8-connectivity, we can draw a line from the inner part of a closed curve into the outer part which does not intersect the curve. This implies that the inner and outer parts of the curve constitute only one contiguous region.

Example 2.3.3. *(Connectivity paradox).*

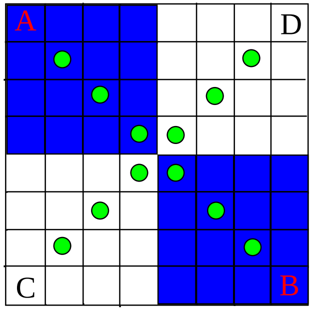


Figure 2.3: Connectivity paradox on a discrete grid.

- *If we assume 4-connectivity, the figure contains four separate contiguous regions A, B, C and D. $A \cup B$ are disconnected, as well as $C \cup D$. A topological contradiction. Intuitively, $C \cup D$ should be connected if $A \cup B$ are disconnected.*
- *If we assume 8-connectivity, there are two regions, $A \cup B$ and $C \cup D$. Both sets contain paths AB and CD entirely within themselves, but which also intersect!*

- One possible solution to contiguity paradox is to treat objects using 4-neighborhoods and background using 8-neighborhoods (or vice versa).
- More exact treatment of digital contiguity paradox and their solution for binary images and images with more brightness levels can be found in [Pavlidis, 1977].
- These problems are typical on square grids — a hexagonal grid (2.2.1) solves many of them. However, a grid of this type has also a number of disadvantages, [Pavlidis, 1977], p. 60.
- For reasons of simplicity and ease of processing, most digitizing devices use a square grid despite the stated drawbacks.
- And for the same reason, we do not pursue further into this topic in this course, but use the simple approach, although there are some paradoxes.

- An alternative approach to the connectivity problems is to use discrete topology based on CW complex theory in topology, which is called cell complex in [Kovalevski, 1989].
- This approach develops a complete strand of image encoding and segmentation.
- The idea, first proposed by Riemann in the nineteenth century, considers families of sets of different dimensions:
  - points, which are 0-dimensional, may then be assigned to sets containing higher dimensional structures (such as pixel array), which permits the removal of the paradoxes we have seen.
  - line segments, which are 1-dimensional, gives precise definition of edge and border.

Other topological and geometrical properties

- **Border** of a region is another important concept in image analysis.
- The border of a region R is the set of pixels within the region that have one or more neighbors outside R.
- This definition of border is sometimes referred to as **inner border**, to distinguish it from the **outer border**, which is the border of the background (i.e., the complement of) of the region.
- **Edge** is a local property of a pixel and its immediate neighborhood — it is a vector given by a magnitude and direction.
- The edge direction is perpendicular to the gradient direction which points in the direction of image function growth.

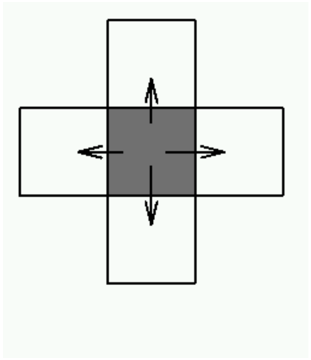
Remark 2.3.4. *Note the difference between border and edge.*

- *The border is a global concept related to a region, while edge expresses local properties of an image function.*
- *The border and edge are related as well.*

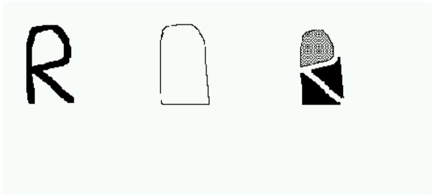
- *One possibility for finding boundaries is chaining the significant edges (points with high gradient of the image function).*

- The edge property is attached to one pixel and its neighborhood — sometimes it is of advantage to assess properties between pairs of neighboring pixels.
- The concept of the **crack edge** comes from this idea.
- Four crack edges are attached to each pixel, which are defined by its relation to its 4-neighbors.

  The direction of the crack edge is that of increasing brightness, and is a multiple of 90 degrees, while its magnitude is the absolute difference between the brightness of the relevant pair of pixels.



- Convex hull is used to describe geometrical properties of objects.
- The convex hull is the smallest region which contains the object, such that any two points of the region can be connected by a straight line, all points of which belong to the region.
- An object can be represented by a collection of its topological components. The sets inside the convex hull which does not belong to an object is called the deficit of convexity.
- This can be split into two subsets. First, **lakes** are fully surrounded by the objects; and second, **bays** are contiguous with the border of the convex hull of the object.



- The convex hull, lakes and bays are sometimes used for object description.

2.3.2 Histogram

- Brightness histogram provides the frequency of the brightness value in the image.
- The brightness histogram $h_f(z)$ is a function showing, for each brightness value z, the number of pixels in the image f that have that brightness value z.
- The histogram of an image with L gray levels is represented by a one-dimensional array with L elements.

- For a digital image of brightness value ranging in $[0, L-1]$, the following algorithm produces the brightness histogram:

  Algorithm 2.3.5. *Computing brightness histogram*

  ```
  1. Assign zero values to all element of the array $h_f$;
  2. For all pixels $(x, y)$ of the image $f$, increment $h_f[f(x, y)]$ by $1$.
  ```

- The histogram is often displayed as a bar graph.
- Readers familiar with statistics will recognize that Computing brightness histogram is similar to generating the histogram of a random variable from a given group of samples of the random variable.
- In the above algorithm, we take the starting value as 0, bin-width as 1, and bin number as L, to generate the histogram.
- This algorithm can be modified to generate brightness histogram of arbitrary bin-width and bin number.
- For multi-spectral band images, histogram of each individual band can be generated in a similar way.

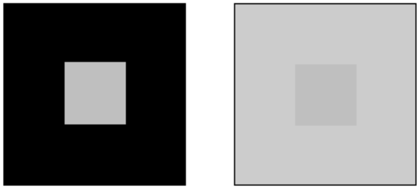
Example 2.3.6. *Histogram example. The Khoros workspace for this example is here Histogram Example.*

2.3.3 Visual perception of the image

- Anyone who creates or uses algorithms or devices for digital image processing should take into account the principle of human visual perception.
- There are psycho-physical parameters such as contrast, border, shape, texture, color, etc.
- Human perception of image provokes many illusions, the understanding of which provides valuable clues about visual mechanisms.
- The topic is covered exhaustively from the point of view of computer vision in [Frisby, 1979].
- This is a difficult and complicated field. We will only touch briefly on it in this course.

Contrast

- Contrast is the local change in brightness and is defined as the ratio between average brightness of an object and the background brightness.
- The human eye is logarithmically sensitive to brightness. That is why the gamma correction in most computer monitors is needed.
- Apparent brightness depends very much on the brightness of the local background; this effect is called conditional contrast.
- The following figure illustrates this effect with two small squares of the same brightness on a dark and a light background.



- Human perceive the brightness of the small squares as different.

Acuity

- Acuity is the ability to detect details in image.
- The human eye is less sensitive to slow and fast changes in brightness in the image plane but is more sensitive to intermediate changes.
- Acuity also decreases with increasing distance from the optical axis.
- Resolution in an image is firmly bounded by the resolution ability of the human eye; there is no sense in representing visual information with higher resolution than that of the viewer.
- Resolution in optics is defined as the inverse value of a maximum viewing angle between the viewer and two proximate points which human cannot distinguish, and so fuse together.

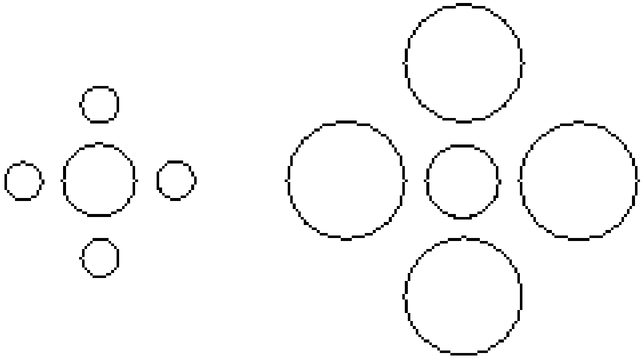
- Human vision has the best resolution for objects which are at a distance of about 25cm from an eye under illumination of about 500lux.
- This illumination is provided by a 60W from a distance 40cm. Under this conditions the distance between two distinguishable points is approximately 0.16mm.
- Another report says that the minimal distinguishable distance is 0.47mm, [Kutter, 1999].

Quiz 2.3.7. *Given the above two minimal distinguishable distance, what is the resolution in DPI needed for a printer to produce perfect output?*

DPI means "Dots Per Inch" (1in = 2.54cm).

Object border

- Object borders carry a lot of information, [Marr, 1982].
- Boundaries of objects and simple patterns such as blobs or lines enable adaption effects similar to conditional contrast.
- The Ebbinghaus illusion is a well known example — two circles of the same diameter in the center of images appear to have different size.



2.3.4 Image quality

- An image might be degraded during capture, transmission, or processing, and measures of image quality can be used to assess the degree of degradation.
- The quality required naturally depends on the purpose for which an image is used.
- Methods for assessing image quality can be divided into two categories: subjective and objective.

- The quality of the image $f(x, y)$ is usually estimated by comparison with a known reference image $g(x, y)$.
- A synthesized image $g(x, y)$ is often used for this purpose.
- One class of methods uses simple measures such as the mean quadratic difference (or mean squared error, MSE)

$$\sum_{x,y} (g(x, y) - f(x, y))^2 \tag{2.18}$$

- The problem here is that it is not possible to distinguish a few big differences from a lot of small differences.
- Instead of the mean quadratic difference, the mean absolute difference or simply maximal absolute difference may be used.
- Correlation between images f and g is another alternative.

- Signal to noise ratio SNR is also used as a image degradation measure. Let $f(x,y)$ be the original image and $f'(x,y)$ be the degraded image, the degree of degradation is measured by

$$SNR(f',f) = 10\log_{10}\frac{\sum_{x,y} f(x,y)^2}{\sum_{x,y}(f(x,y)-f'(x,y))^2} \quad \text{(db)} \tag{2.19}$$

- Peak signal to noise ratio PSNR is another measure in this class. It is defined as

$$PSNR(f',f) = 10\log_{10}\frac{N \max_{x,y} f(x,y)^2}{\sum_{x,y}(f(x,y)-f'(x,y))^2} \quad \text{(db)} \tag{2.20}$$

  where N is number of pixels.

  Experimentally, a PSNR larger than 32db means invisible visual degradation.

- Measures of image similarity are becoming more important since they may be used in image retrieval from multimedia databases.

- There are many other measures of image similarity based on distance functions, [D. R. Wilson and T. R. Martinez, 1997].

Example 2.3.8. *Image quality/similarity example. The Khoros workspace for this example is here Quality/SimilarityExample.*

2.3.5 Noise in images

Image Noise

- Images are often degraded by random noise.
- Noise can occur during image capture, transmission or processing, and may be dependent on or independent of image content.
- Noise is usually described by its probabilistic characteristics.
- White noise — constant power spectrum (its intensity does not decrease with increasing frequency); it is frequently applied as a crude approximation of image noise in most cases. Its auto-correlation is the delta function. So it is un-correlated at two different instances.
- The advantage is that it simplifies the calculations.
- A special case of noise is Gaussian noise.
  - Gaussian noise is a very good approximation of noise that occurs in many practical cases.
  - Probability density of the random variable is given by the Gaussian function.

– 1D Gaussian noise — μ is the mean and σ is the standard deviation of the random variable.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\mathbf{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2.21}$$

Noise Type

Noise may be

- **additive** noise ν and image signal g are independent

$$f(x, y) = g(x, y) + \nu(x, y). \tag{2.22}$$

  During image transmission, noise is usually independent of the image signal occurs. The degradation can be modeled as additive noise.

- **multiplicative** noise is a function of signal magnitude

$$f(x, y) = g(x, y) + \nu(x, y)g(x, y) = g(x, y)(1 + \nu(x, y)) = g(x, y)n(x, y). \tag{2.23}$$

- **impulse** noise means that an image is corrupted with individual noisy pixels whose brightness differs significantly from that of the neighborhood.

- The term “salt and pepper noise” is used to describe saturated impulsive noise — an image corrupted with white and/or black pixel is an example.

The problem of suppressing noise in images is addressed in subsequent lectures of this course.

Simulation of noise

We first consider the simulation of Gaussian noise.

Theorem 2.3.9. *(Box-Muller Method) Let U_1 and U_2 be i.i.d (independent identical distributed) uniformly distributed random variables on $(0, 1)$. Then the random variables*

$$N_1 = \sqrt{-2\log U_1}\cos(2\pi U_2) \tag{2.24}$$

$$N_2 = \sqrt{-2\log U_1}\sin(2\pi U_2) \tag{2.25}$$

are independent standard Gaussian.

To prove it we need the following

Theorem 2.3.10. *(Transformation Theorem for Densities) Let Z_1, Z_2 and U_1, U_2 be random variables. Assume*

- (U_1, U_2) *takes values in the open set G' of $\mathbf{R}^2$ and has density f on G';*
- (Z_1, Z_2) *takes values in the open set G of $\mathbf{R}^2$;*
- $\varphi : G \longmapsto G'$ *is a continuously differentiable bijection with continuously differentiable inverse $\varphi^{-1} : G' = \varphi(G) \longmapsto G$.*

Given

$$\begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \varphi \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix}. \tag{2.26}$$

the random vector (Z_1, Z_2) on G has the density

$$g(z) = f \circ \varphi(z) |J_\varphi(z)| \tag{2.27}$$

where $J_\varphi(z) = \frac{\partial(\varphi_1, \varphi_2)}{\partial(z_1, z_2)}$ is the Jacobian of φ.

Proof of Theorem 2.3.9: Let us first determine the map φ from the last theorem. We have

$$N_1^2 = -2\log U_1 \cos^2(2\pi U_2) \tag{2.28}$$
$$N_2^2 = -2\log U_1 \sin^2(2\pi U_2). \tag{2.29}$$

Hence

$$N_1^2 + N_2^2 = -2\log U_1 \tag{2.30}$$

and

$$U_1 = \mathbf{e}^{-\frac{N_1^2+N_2^2}{2}}. \tag{2.31}$$

Moreover, by (2.24),

$$\frac{N_2}{N_1} = \tan(2\pi U_2), \tag{2.32}$$

i.e.,

$$U_2 = \frac{1}{2\pi}\arctan\left(\frac{N_2}{N_1}\right). \tag{2.33}$$

Hence

$$\varphi(z_1, z_2) = \begin{pmatrix} \mathbf{e}^{-\frac{z_1^2+z_2^2}{2}} \\ \frac{1}{2\pi}\arctan(\frac{z_2}{z_1}) \end{pmatrix} \tag{2.34}$$

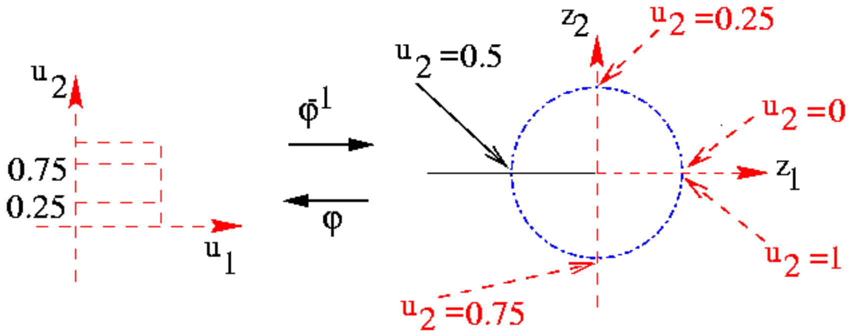
The transform domains are

$$G = \mathbf{R}^2 \setminus [\{z_1 = 0\} \cup \{z_2 = 0 \text{ and } z_1 > 0\}] \tag{2.35}$$

and

$$G' = (0,1) \times (0,1) \setminus \{u_2 = \frac{1}{4} \text{ or } u_2 = \frac{3}{4}\} \tag{2.36}$$

The partial derivatives of φ are



$$\frac{\partial \varphi_1}{\partial z_1}(z) = -z_1 \mathbf{e}^{-\frac{z_1^2+z_2^2}{2}}, \qquad \frac{\partial \varphi_1}{\partial z_2}(z) = -z_2 \mathbf{e}^{-\frac{z_1^2+z_2^2}{2}} \tag{2.37}$$

$$\frac{\partial \varphi_1}{\partial z_1}(z) = \frac{1}{2\pi}\frac{-z_2}{z_1^2+z_2^2}, \qquad \frac{\partial \varphi_1}{\partial z_2}(z) = \frac{1}{2\pi}\frac{z_1}{z_1^2+z_2^2}. \tag{2.38}$$

which implies

$$J_\varphi(z) = \frac{1}{2\pi}\mathbf{e}^{-\frac{z_1^2+z_2^2}{2}} = \frac{1}{\sqrt{2\pi}}\mathbf{e}^{-\frac{z_1^2}{2}} \cdot \frac{1}{\sqrt{2\pi}}\mathbf{e}^{-\frac{z_2^2}{2}} \tag{2.39}$$

Since (U_1, U_2) has density $\chi_{(0,1)\times(0,1)}$, which is identically 1 on $\chi_{(0,1)\times(0,1)}$, (N_1, N_2) has density

$$\frac{1}{\sqrt{2\pi}}\mathbf{e}^{-\frac{z_1^2}{2}} \cdot \frac{1}{\sqrt{2\pi}}\mathbf{e}^{-\frac{z_2^2}{2}} \tag{2.40}$$

on G. Therefore they are independent Gaussian variables. $\square$

The rejection method

- This method uses an auxiliary density for generation of random quantities from distributions not amenable to analytic treatment.
- Consider the generation of samples from a density π.
- Consider also an auxiliary density q for which we know how to generate samples.

- The method is general enough to generate samples from π without knowing the complete expression for π.
- It is extremely common in statistics to encounter such situations where the kernel of π is known but the constant ensuring it integrals to 1 cannot be obtained analytically.
- The idea is to use q to make samples from π.
- The only mathematical restriction over q is that there must exist a constant A such that
$$\pi(x) \leq Aq(x) \tag{2.41}$$
for every possible value of x.
- The method consists of independently drawing X from q and $U \sim U[0,1]$ and accepting X as a sample generated from π if
$$AUq(X) \leq \pi(X). \tag{2.42}$$
Otherwise X is not accepted as a sample from π and the process must be reinitialized until a value X is accepted.
- Hence the name of the method, which is also known as the acceptance/rejection method.

- The samples to be generated can have any form: scalar, vector or matrix.
- In each case, the rejection step is based on a comparison of densities with aid of a scalar uniform random variable.

Proof of **rejection method**: To prove the rejection procedure effectively generates samples from π, we need to show that the density function of X is a constant multiple of π, when conditioned by the acceptance condition, (2.42).

The joint density of (X, U) is $f(x,u) = q(x)\chi_{[0,1]}(u)$, by the independence between X and U and the uniformity of U.

Then the conditional probability, by Bayes' theorem,

$$\begin{aligned}
\mathbf{Pr}(X \le t | AUq(X) \le \pi(X)) &= \mathbf{Pr}(X \le t | U \le \frac{\pi(X)}{Aq(X)}) && (2.43)\\
&= \frac{\mathbf{Pr}(X \le t, U \le \frac{\pi(X)}{Aq(X)})}{\mathbf{Pr}(U \le \frac{\pi(X)}{Aq(X)})} && (2.44)\\
&= \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} q(x)\chi_{[0,1]}(u)\chi_{\left\{x\le t, u \le \frac{\pi(x)}{Aq(x)}\right\}}(x,u)\,dxdu}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} q(x)\chi_{[0,1]}(u)\chi_{\left\{u \le \frac{\pi(x)}{Aq(x)}\right\}}(x,u)\,dxdu} && (2.45)\\
&= \frac{\int_{-\infty}^{t} dx \int_0^{\frac{\pi(x)}{Aq(x)}} q(x)\chi_{[0,1]}(u)\,du}{\int_{-\infty}^{\infty} dx \int_0^{\frac{\pi(x)}{Aq(x)}} q(x)\chi_{[0,1]}(u)\,du} && (2.46)
\end{aligned}$$

$$= \frac{\int_{-\infty}^{t} \frac{\pi(x)}{Aq(x)} q(x)\, dx}{\int_{-\infty}^{\infty} \frac{\pi(x)}{Aq(x)} q(x)\, dx} \tag{2.47}$$

$$= \frac{\int_{-\infty}^{t} \pi(x)\, dx}{\int_{-\infty}^{\infty} \pi(x)\, dx} \tag{2.48}$$

as required. □

- The required density is given by the normalized version of π.
- When π is already the complete expression of a density, the normalization is not needed, as $\int_{-\infty}^{\infty} \pi(x)\,dx = 1$.
- q should be a density that is easy to draw samples from.
- The overall acceptance probability is

$$\mathbf{Pr}(U \leq \frac{\pi(X)}{Aq(X)}) = \frac{1}{A}\int_{-\infty}^{\infty} \pi(x)\,dx \qquad (2.49)$$

  Hence, A must be chosen as close as possible to $\int_{-\infty}^{\infty} \pi(x)\,dx$.

- An extreme case is that $\pi = q$. Then

$$\int_{-\infty}^{\infty} \pi(x)\,dx = \int_{-\infty}^{\infty} q(x)\,dx = 1 \qquad (2.50)$$

  as q is a normalized density. We can choose $A = 1$ to satisfy the mathematical requirement (2.41) and get the maximal acceptance probability — the acceptance condition (2.42) is satisfied for all x and u. This is obvious, since each sample from q is also a sample from π.

- A special case of rejection sampling is given for truncated distributions.
- Let q be any density and π be its truncation to the region C, i.e., $\pi = q\chi_C \leq q$.
- Taking $A = 1$, the acceptance condition is

$$Uq(X) \leq \pi(X) = q\chi_C \tag{2.51}$$

  which is satisfied, almost surely, if and only if $X \in C$.
- Hence, to generate sample from q restricted to C, one simply has to draw sample from q and accept it if and only if it is in C.

The Polar method

- This a variant of the Box-Muller method to generate standard normal deviates. It is due to G. Marsaglia.
- It is substantially faster than the Box-Muller method since it avoids the calculation of the trigonometric functions (but still slower than other methods, [Knuth, 1981]) and it has essential perfect accuracy.

- The Box-Muller method may be rephrased as follows:
  - given (W, Θ) uniformly distributed on $[0,1] \times [0, 2\pi]$;
  - the variables

$$N_1 = \sqrt{-2 \log W} \cos(\Theta) \tag{2.52}$$

$$N_2 = \sqrt{-2 \log W} \sin(\Theta) \tag{2.53}$$

    are independent standard Gaussian.

- The rejection method allows us to sample directly from $\sqrt{W}\cos\Theta$ and $\sqrt{W}\sin\Theta$, thus avoiding to calculate the sines and cosines:
  - Given (Z_1, Z_2) uniformly distributed on the unit disk;
  - $Z_1 = R\cos\Theta$ and $Z_2 = R\sin\Theta$ in polar coordinates R, Θ;
  - Then $W = R^2$ and Θ have joint density

$$\frac{1}{\pi}\chi_{\{|r|\leq 1\}}(\sqrt{W}\cos\Theta, \sqrt{W}\sin\Theta)|\frac{\partial Z_1, Z_2}{\partial W, \Theta}| \tag{2.54}$$

$$= \frac{1}{\pi}\chi_{(0,1]\times[0,2\pi)}\left|\begin{pmatrix} \frac{\cos\Theta}{2\sqrt{W}} & \frac{\sin\Theta}{2\sqrt{W}} \\ -\sqrt{W}\sin\Theta & \sqrt{W}\cos\Theta \end{pmatrix}\right| \tag{2.55}$$

$$= \frac{1}{2\pi}\chi_{[0,1]\times[0,2\pi)} \tag{2.56}$$

    on $(0, 1] \times [0, 2\pi)$;
  - Hence (W, Θ) constructed above is uniform and independent on $(0, 1] \times [0, 2\pi)$;
  - Clearly $W = Z_1^2 + Z_2^2$ and $\cos\Theta = \frac{Z_1}{\sqrt{W}}$, and $\sin\Theta = \frac{Z_2}{\sqrt{W}}$;
  - Then the Box-Muller transform can be written as

$$N_1 = \sqrt{\frac{-2\log W}{W}}Z_1 \tag{2.57}$$

$$N_2 = \sqrt{\frac{-2\log W}{W}} Z_2 \tag{2.58}$$
$$W = Z_1^2 + Z_2^2 \tag{2.59}$$

- To sample from the unit disk, we adopt the truncated rejection method:
  - sample (V_1, V_2) uniformly from $[-1, 1] \times [-1, 1]$;
  - until $0 < V_1^2 + V_2^2 \leq 1$, set $(Z_1, Z_2) = (V_1, V_2)$.

Algorithm 2.3.11. *C code for generating Gaussian deviates by the polar method:*

```
#include <math.h>

/*
 * your_rand (): the function that generate sample
 * from uniform distribution on [-1,1]
 */
float your_rand (void);

float gasdev(void)
{
```

```
static int iset = 0;
static float gset;
float fac, rsq, v1, v2;

if  (iset == 0)
  {
    do
      {
        v1  = 2.0 * your_rand () - 1.0;
        v2  = 2.0 * your_rand () - 1.0;
        rsq = v1 * v1 + v2 * v2;
      } while (rsq >= 1.0 || rsq == 0.0);
    fac = sqrt (-2.0 * log (rsq) / rsq);
    gset = v1 * fac;
    iset = 1;
    return v2 * fac;
  }
else
  {
    iset = 0;
    return gset;
    }
}

/* modified from Numerical Recipes */
/* (C) Copr. 1986-92 Numerical Recipes Software )#40. */
```

Reference for this section is [Press et al., 1992, Wrinkler, 1995, Gamerman, 1997].

Example 2.3.12. *Noise example. The Khoros workspace for this example is here Noise Example.*

Chapter 3

Data structures for image analysis

- Data and an algorithm are the two basic parts of any program.
- Computer program $=$ data $+$ algorithm.
- Data organization can considerably affect the simplicity of the selection and the implementation of an algorithm. The choice of data structures is fundamental when writing a program.
- The difficulties in image processing, image analysis and computer vision come from the bad representation or organization of the data involved. In fact, the

visual information representation and organization inside human brain is not well understood at present.

- Although we are to discuss some representations used so far, none of them are appropriate for a general purpose processing target.

3.1 Levels of representation

- The aim of computer visual perception is to find a relation between an input image and the models of the real world.
- During the transition from the raw image to the model, semantic knowledge about the interpretation of image data is used more and more.
- Several levels of visual information representation are defined on the way between the input image the model.
- Computer vision then comprises a design of the
  - Intermediate representations (data structures).
  - Algorithms used for the creation of representation and introduction of relations between them.

- The representation can be stratified in four levels.
- However, there are no strict borders between them and a more detailed classification of the representational levels may be used in some applications. For some specific uses, some representations can be omitted.
- These four representational levels are ordered from signals at low level of abstraction to the description that a human can understand.
- The information flow between the levels may be bi-directional.
- The four levels are
  - Iconic images
  - Segmented images
  - Geometric representations
  - Relational models

- Iconic images — consists of images containing original data; integer matrices with data about pixel brightness.
- E.g., outputs of pre-processing operations (e.g., filtration or edge sharpening) used for highlighting some aspects of the image important for further treatment.

- Segmented images — parts of the image are joined into groups that probably belong to the same objects.
- E.g., the output of the segmentation of a scene with polyhedrons is either line segments coinciding with borders or two-dimensional regions corresponding with faces of bodies.
- It is useful to know something about the application domain while doing image segmentation; it is then easier to deal with noise and other problems associated with erroneous image data.

- Geometric representations — hold knowledge about 2D and 3D shapes.
- The quantification of a shape is very difficult but very important.
- It is the inverse problem of computer graphics.

- Relational models - give the ability to treat data more efficiently and at a higher level of abstraction.
- A priori knowledge about the case being solved is usually used in processing of this kind.
- Example - counting planes standing at an airport using satellite images
  - position of the airport (e.g., from a map).
  - relations to other objects in the image (e.g., to roads, lakes, urban areas).
  - geometric models of planes for which we are searching.
  - etc.
- AI techniques are often explored.
- Information gained from the image may be represented by semantic nets or frames.

3.2 Traditional image data structures

- Traditional image data structures, such as
  - matrices
  - chains
  - graphs
  - lists of object properties
  - relational databases
  - etc.

  are important not only for the direct representation of image information, but also a basis of more complex hierarchical methods of image representation.

3.2.1 Matrices

- Most common data structure for low level image representation
- Elements of the matrix are integer, real or complex numbers, corresponding to brightness, or to another property of the corresponding pixel of the sampling grid.
- Image data of this kind are usually the direct output of the image capturing device, e.g., a scanner.
- Pixels of both rectangular and hexagonal sampling grids((2.2.1)) can be represented by a matrix.
- The correspondence between data and matrix elements is obvious for a rectangular grid; with a hexagonal grid every row in the image is shifted half a pixel to the right or left.

STOP

- Image information is the matrix is accessible through the co-ordinates of a pixel that correspond with row and column indexes.

- The matrix is a full representation of the image, independent of the contents of the image data.
- A representation of a segmented image by a matrix usually saves memory than an explicit list of all spatial relations between all objects, although sometimes we need to record other relations among objects.

- Some special images that are represented by matrices are:
  - Binary images (an image with two brightness levels only) is represented by a matrix containing zeros and ones.
  - Several matrices can contain information about one multi-spectral image. Each of these matrices contains one image corresponding to one spectral band.
  - Matrices of different resolution are used to obtain hierarchical image data structures. This hierarchical representation of the image can be very convenient for parallel computers with the "processor array" architecture.

- Most programming language use a standard array data structure to represent a matrix.
- In C, an image of 256 gray-levels with dimension $M \times N$ can be stored in a two dimensional array

```
unsigned char image [M][N]
```

- The above stored array may be dis-continuous in memory. Then another way to represent the image is by a one-dimensional array, which is continuous in the memory

```
unsigned char image [M * N]
```

- We need to know the image dimension M and N usually.

- Another method is to represent an image by a structure type,

  ```
  typedef struct _image
    {
     int rows;
     int columns;
     char *values;
     <other useful information>;
    } Image;
  ```

  or use a pointer to the above structure

  ```
  typedef Image * ImageX;
  ```

  or declared as

  ```
  typedef struct _image * ImageX;
  ```

  When initialized,

```
value = (char *) malloc (sizeof(char) * rows * columns);
```

- To access one element of the matrix at pixel (m, n), use the following macro

  ```
  #define Pixel(IX, m, n) \
          (IX->values[m*(IX->columns)+n])
  ```

  where $0 \leq m \leq M - 1$ and $0 \leq n \leq N - 1$.

- You may use other type such as

  ```
  float *values;
  ```

  or

  ```
  double *values;
  ```

  to gain much precise computation.

3.2.2 Chains

- Chains are used for the representation of object borders in computer vision.
- One element of the chain is a basic symbol, which corresponds to some kind of primitives in the image.
- Chains are appropriate for data that can be arranged as a sequence of symbols.
- The neighboring symbols in a chain usually correspond to the neighboring of primitives in the image.

- **Chain codes** or **Freeman codes** [Freeman, 1961] are often used for the description of object borders, or other one-pixel-wide primitives in images.

- The border is defined by the co-ordinates of its reference pixel and the sequence of symbols corresponding to the line of the unit length in several predefined orientations.

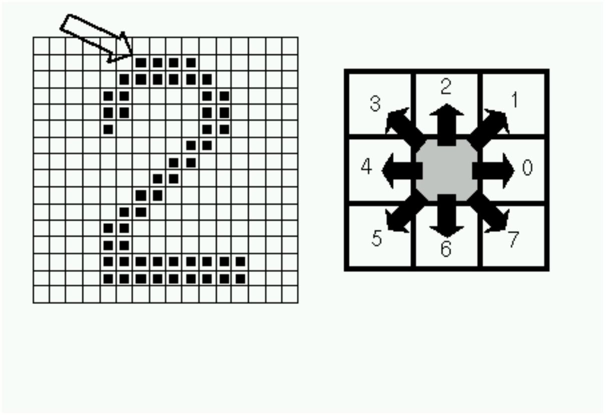
- Notice that a chain code is of a relative nature; data are expressed with respect to some reference point.

- Chain codes describe an object by a sequence of unit-size length line segments with a given orientation.

- The first element of such a sequence must bear information about its position to permit the region to be reconstructed.

- The result is a sequence of numbers, indicating the orientation.

- An example of a chain code is shown in the following figure, where 8-neighborhoods are used — it is possible to define chain code using 4-neighborhoods as well. The reference pixel is marked by an arrow. The chain code is



00077665555556600000006444444442221111112234445652211

- To exploit the position invariance of chain codes, e.g., when used for matching, the first element, which contains the position information, should be omitted. One need a method to normalize the chain codes.
- A chain code is very sensitive to noise and arbitrary changes in scale and rotation may cause problems if used fro recognition.
- The description of an image by chains is appropriate for syntactic pattern recognition that is based on formal language theory approaches.
- Chains can be represented using static data structures (e.g., 1D arrays); their size is the longest length of the chain expected.
- Dynamic data structures are more advantageous to save memory.

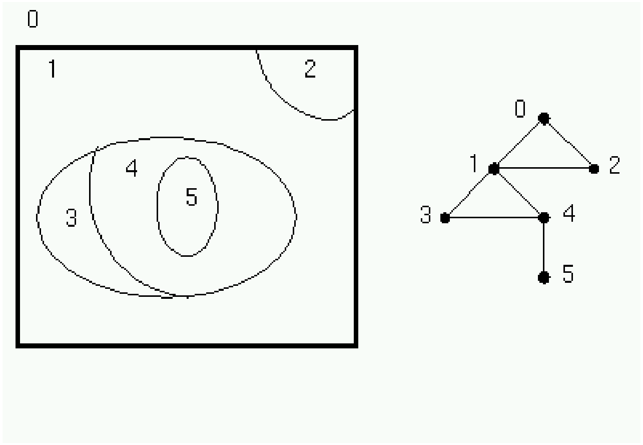
3.2.3 Topological data structures

- Topological data structures describe the image as a set of elements and their relations.
- These relations are often represented using graphs.
- A graph $G = (V, E)$ is an algebraic structure which consists of a set of nodes
$$V = \{v_1, v_2, \cdots, v_n\}$$
and a set of arcs
$$E = \{e_1, e_2, \cdots, e_m\}$$
- Each arc e_k is naturally connected with an unordered pair of nodes $\{v_i, v_j\}$ which are not necessarily distinct.
- The degree of the node is equal to the number of incident arcs of the node.

- An evaluated graph is a graph in which values are assigned to arcs, to nodes or to both — these values may, e.g., represent weights, or costs.

- The **region adjacency graph** is typical of this class of data structures.
- Nodes correspond to regions and neighboring regions are connected by an arc.
- The segmented image consists of regions of pixels with similar properties (brightness, texture, color, ...) that correspond to some entities in the scene, and the neighborhood relation is fulfilled when the regions have some common border.

Example 3.2.1. *An example of an image with regions labeled by numbers and corresponding region adjacency graph is shown in the following figure The label 0*



is denotes pixels out of the image. This value is used to indicate regions that touch borders of the image in the region adjacency graph.

- The region adjacency graph has several attractive features.
- If a region enclose other regions, then the part of the graph corresponding with the areas inside can be separated by a cut in the graph.
- Nodes of degree 1 represent simple holes.
- The region adjacency graph can be used for matching with a stored pattern for recognition purpose.

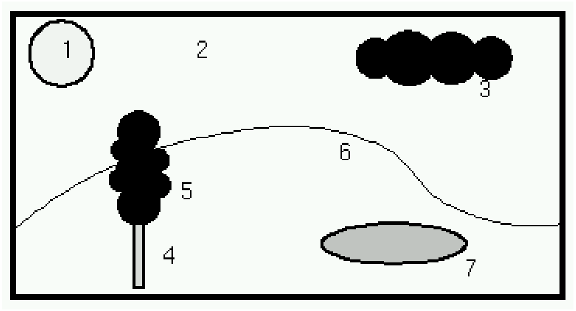
STOP

- The region adjacency graph is usually created from the **region map**, which is a matrix of the same dimension as the original image matrix whose elements are identification labels of the regions.
- To created the region adjacency graph, borders of all regions in the image are traced, and labels of all neighboring regions are stored.
- The region adjacency graph can easily be created from an image represented by a quadtree (3.3.2) as well.

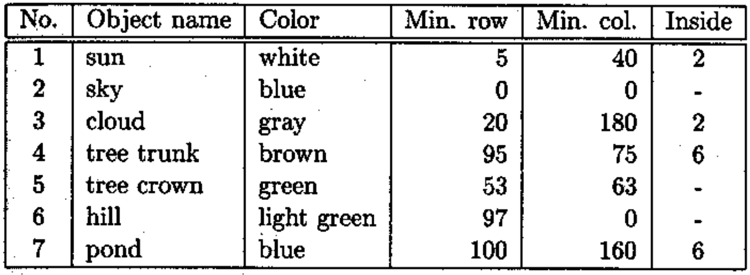
3.2.4 Relational structures

- Relational databases can also be used for representation of information from an image.
- All the information is then concentrated in relations between semantically important parts of the image — objects — that are the result of segmentation.
- Relations are recorded in the form of tables.
- Individual objects are associated with their names and other features, e.g., the top-left pixel of the corresponding region in the image.
- Relations between objects are expressed in the relational table as well.
- Description by means of relational structures is appropriate for higher level image understanding.
- Search using keys, similar to database searches, can be used to speed up the whole process.

Example 3.2.2. *An example of such a representation is shown in the following figure and table.*



(a) Image



No.	Object name	Color	Min. row	Min. col.	Inside
1	sun	white	5	40	2
2	sky	blue	0	0	-
3	cloud	gray	20	180	2
4	tree trunk	brown	95	75	6
5	tree crown	green	53	63	-
6	hill	light green	97	0	-
7	pond	blue	100	160	6

(b) Database

3.3 Hierarchical data structures

- Computer vision is by its nature very computationally expensive, if for no other reason than the large amount of data to be processed.
- Systems which we might call sophisticated must process considerable quantities of image data — hundreds of kilobytes to tens of megabytes.
- The visual information perceived by the two human eyes is about 3000MB/s (add reference here!!!).
- One of the solutions is using parallel computers (in other words brute force).
- Unfortunately many computer vision problems are difficult to divide among processors, or decompose in any way.

- Hierarchical data structures make it possible to use algorithms which decide a strategy for processing on the basis of relatively small quantities of data.
- They work at the finest resolution only with those parts of the image for which it is necessary, using knowledge instead of brute force to ease and speed up the processing.
- We are going to introduce two typical hierarchical structures, pyramids and quadtrees.

3.3.1 Pyramids

- Pyramids are among the simplest hierarchical data structures.
- We distinguish between **M-pyramids** (matrix pyramids) and **T-pyramids** (tree pyramids).

- A M-pyramid is a sequence $\{M_L, M_{L-1}, \cdots, M_0\}$ of images.
- M_L has the same dimensions and elements as the original image.
- M_{i-1} is derived from the M_i by reducing the resolution by one half.
- When creating pyramids, it is customary to work with square matrices with dimensions equal to powers of 2.
- For images with arbitrary dimensions, a resampling procedure is needed in creating the pyramids.
- M_0 corresponds to one pixel only.

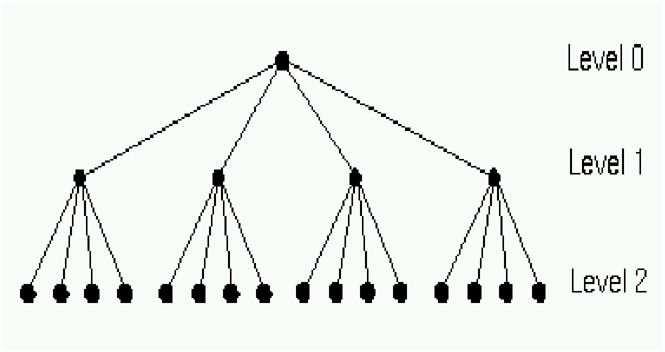
- The number of image pixels used by an M-pyramid for storing all matrices is

$$N^2 \left(1 + \frac{1}{4} + \frac{1}{16} + \cdots \right) = 1.33N^2$$

- M-pyramids are used when it is necessary to work with an image at different resolutions simultaneously.
- An image having one degree smaller resolution in a pyramid contains four times less data, so that it is processed approximately four times as quickly.

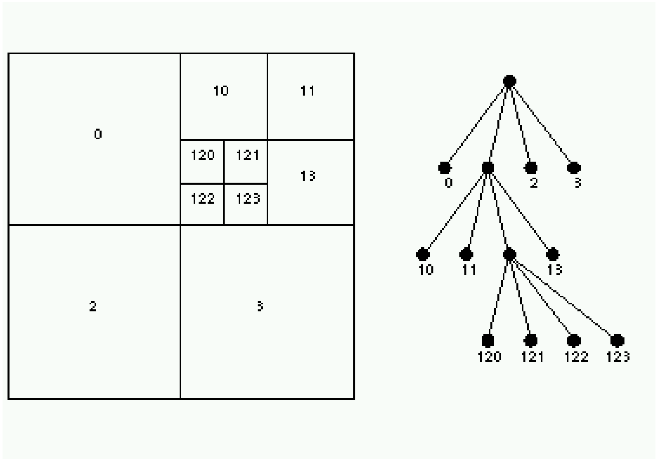
Example 3.3.1. *M-pyramid created by shrinking the image dimensions. The Khoros workspace for this example is here M-pyramid example.*

- Often it is advantageous to use several images of the same resolution simultaneously rather than to create just one image at a resolution in the M-pyramid.
- E.g., we use images at a resolution, containing additional information at this resolution, texture, orientation and segmentation properties, etc.
- Such images can be represented using tree pyramids — T-pyramids.
- The following figure is a example T-pyramid tree. Every node of the T-pyramid has 4 child nodes.



3.3.2 Quadtrees

- Quadtrees are modifications of T-pyramids.
- Every node of the tree except the leaves has four children (NW: north-western, NE: north-eastern, SW: south-western, SE: south-eastern).
- the image is divided into four quadrants at each hierarchical level, however it is not necessary to keep nodes at all levels.
- If a parent node has four children of the same (e.g., brightness) value, (which is often characterized by the differences among the pixel brightness and the average with a given threshold value), it is not necessary to record them.


0
10
11
120
121
122
123
13
2
3
0
2
3
10
11
13
120
121
122
123

- Quadtrees are usually represented by recording the whole tree as a list of its individual nodes, every node being a record with several items characterizing it.
- An example is given as following

```
node = {
          node_type,
          pointer_to_NW_son,
          pointer_to_NE_son,
          pointer_to_SW_son,
          pointer_to_SE_son,
          pointer_to_Father,
          other_data
       }
```

  - In the item "node_type", there is information about whether the node is a leaf or inside the tree.
  - "other_data" can be the level of the node in the tree, position in the picture, brightness for this node, etc.

- This kind of representation is redundant and expansive in memory. Its advantage is easy access to any node.

Example 3.3.2. *Example from matlab. Launch matlab, run the demo "qtdemo". matlab uses sparse matrix to store the quadtree decomposition, without the brightness value information for each node or block.*

Use the demo data from "help qtdecomp" to see the result numerically.

- Problems associated with hierarchical image representation:
  - Dependence on the position, orientation and relative size of objects.
  - Two similar images with just very small differences can have very different pyramid or quadtree representations.
  - Even two images depicting the same, slightly shifted scene, can have entirely different representations.

Chapter 4

Image Pre-processing

- Pre-processing is a common name for operations with images at the lowest level of abstraction — both input and output are intensity images.

- These iconic images are of the same kind as the original data captured by the sensor, with an intensity image usually represented by a matrix of image function values (brightness).

- The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

- Four categories of image pre-processing methods according to the size of the pixel neighborhood that is used for the calculation of a new pixel brightness:
  - pixel brightness transformations.
  - geometric transformations.
  - pre-processing methods that use a local neighborhood of the processed pixel.
  - image restoration that requires knowledge about the entire image.
- Other classifications of image pre-processing methods exist.

STOP

- Image pre-processing methods use the considerable redundancy in images.
- Neighboring pixels corresponding to one object in real images have essentially the same or similar brightness value.
- Thus, distorted pixel can often be restored as an average value of neighboring pixels.

Quiz 4.0.1. *Do you remember the example of filtering impulse noise? (2.3.12)*

STOP

- If pre-processing aims to correct some degradation in the image, the nature of a priori information is important and is used to different extent:
  - no knowledge about the nature of the degradation is used; only very general properties of the degradation are assumed.
  - using knowledge about the properties of the image acquisition device, and conditions under which the image was obtained. The nature of noise (usually its spectral characteristics) is sometimes known.
  - using knowledge about objects that are searched for in the image, which may simplify the pre-processing quite considerably.

- If knowledge about objects is not available in advance it can be estimated during the processing.
- The following strategy is possible.
  - First the image is coarsely processed to reduce data quantity and to find image objects.
  - The image information derived is used to create a hypothesis about image object properties and this hypothesis is then verified in the image at finer resolution.
  - Such an iterative process cab be repeated until the presence of knowledge is verified or rejected.
  - This feedback may span more than pre-processing, since segmentation also yields semantic knowledge about objects — thus feedback can be initiated after the object segmentation.

4.1 Pixel brightness transformations

- Brightness transformations modify pixel brightness — the transformation depends on the properties of a pixel itself.
- There are two brightness transformations:
- Brightness corrections
  - consider the original brightness
  - and pixel position in the image.
- Gray scale transformation
  - change brightness without regard to position in the image.

4.1.1 Position dependent brightness correction

- Ideally, the sensitivity of image acquisition and digitization devices should not depend on position in the image, but this assumption is not valid in practice.
- Sources of degradation:
  - non-homogeneous property of optical system;

    The lens attenuates light more if it passes farther from the optical axis.
  - non-homogeneous sensitivity of light sensors;

    The photo sensitive part of the sensor (vacuum-tube camera, CCD camera elements) is not of identical sensitivity.
  - non-homogeneous object illumination.
- Systematic degradation can be suppressed by brightness correction.
- Let a multiplicative error coefficient $e(i, j)$ describe the change from the ideal identity transfer function
  - $g(i, j)$ is the original undegraded image (or desired image);
  - $f(i, j)$ is the image containing degradation.

$$f(i, j) = e(i, j)g(i, j) \tag{4.1}$$

- If a reference image $g_c(i,j)$ is known (e.g., constant brightness c)
  - the degraded result is $f_c(i,j)$
  - systematic brightness errors can be suppressed:

$$g(i,j) = \frac{f(i,j)}{e(i,j)} = \frac{g_c(i,j) f(i,j)}{f_c(i,j)} = \frac{c}{f_c(i,j)} f(i,j) \tag{4.2}$$

- This method can be used only if the image degradation process is stable.

  If we wish to suppress this kind of degradation in the image capture process, we should calibrate the device from time to time (find error coefficients $e(i,j)$)

- This method implicitly assumes linearity of the transformation, which is not true in reality as the brightness scale is limited into some interval.
  - overflow is possible in (4.2). Then the limits of the brightness scale are used instead in (4.2).
  - the best reference image should have brightness that is far enough from both limits.

- If the gray scale has 256 brightnesses the ideal image has constant brightness value 128.

– Most TV cameras have automatic control of the gain which allows them to operate under changing illumination conditions. If systematic errors are suppressed using error coefficients, this automatic gain control should be switched off first.

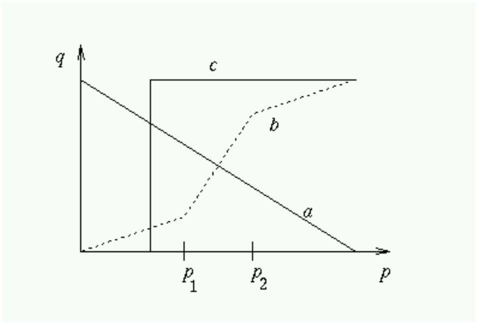
Example 4.1.1. *Brightness correction example. The Khoros workspace for this example is here Brightness correction Example.*

4.1.2 Grey scale transformation

- Grey scale transformations do not depend on the position of the pixel in the image.

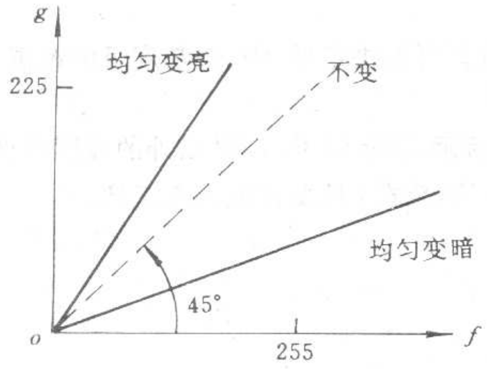
- Brightness transform is a monotonic function:

$$q = T(p) \tag{4.3}$$

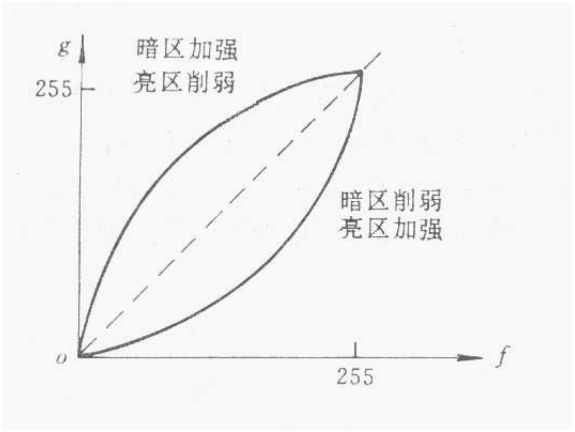


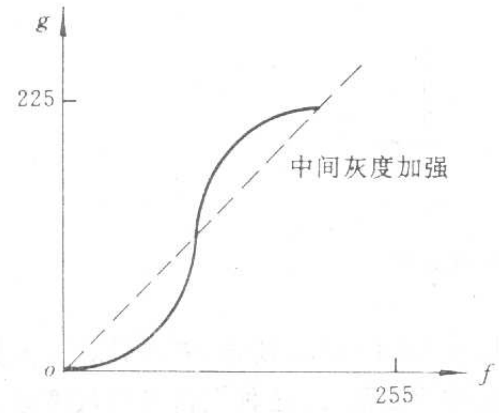
  - a - Negative transformation
  - b - contrast enhancement (between p_1 and p_2)
  - c - Brightness thresholding

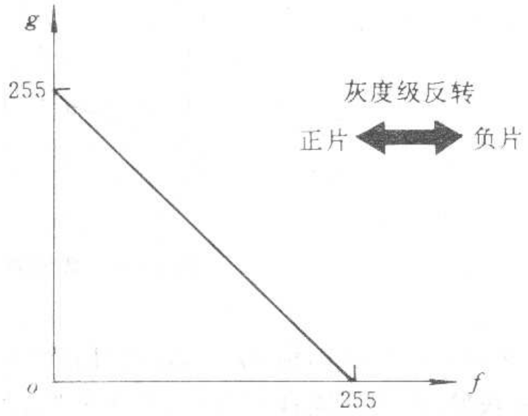
- Some other examples:

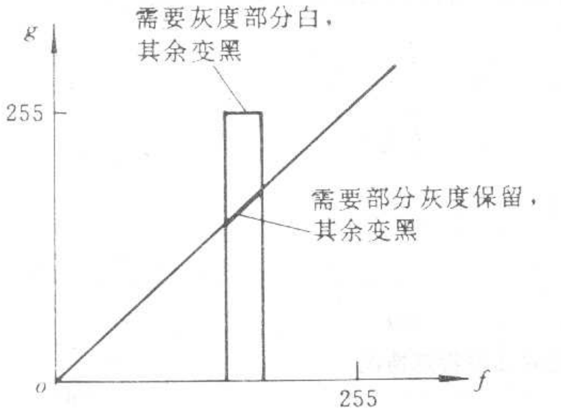


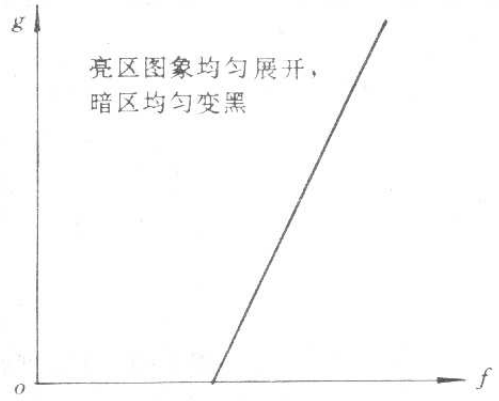
- Grey scale transformations can be performed using look-up tables.
- Grey scale transformations are mostly used if the result is viewed by a human.


g
255
暗区加强
亮区削弱
暗区削弱
亮区加强
o
255
f


g
225
中间灰度加强
o
255
f


g
255
灰度级反转
正片
负片
o
255
f


g
需要灰度部分白，
其余变黑
255
需要部分灰度保留，
其余变黑
o
255
f


g
亮区图象均匀展开，
暗区均匀变黑
o
f

Windows and level

- An interactive contrast enhancement tool normally available in image processing software is called Window and Level.
- It is an expansion of the contrast of the pixels within a given window range.
- Two parameters define the range: the middle point Level, and the width of the range Window.
- Another name for this operation is Intensity of Interest (IOI).
- A graphic visualization of this process is shown below.

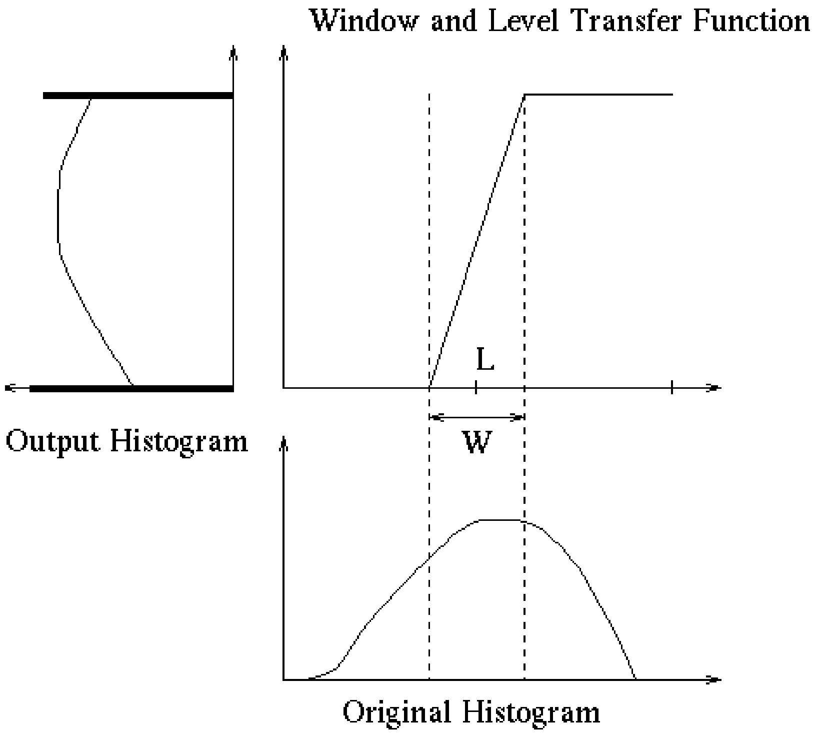


Figure 4.1: A graphic visualization of this process.

Example 4.1.2. *Window and Level example. The Khoros workspace for this example is here Window Level Example.*

First find the minimum and maximum value, decide the start value, bin-width and number of bins for computing the histogram. From the histogram, chose the lower and upper cutoff value.

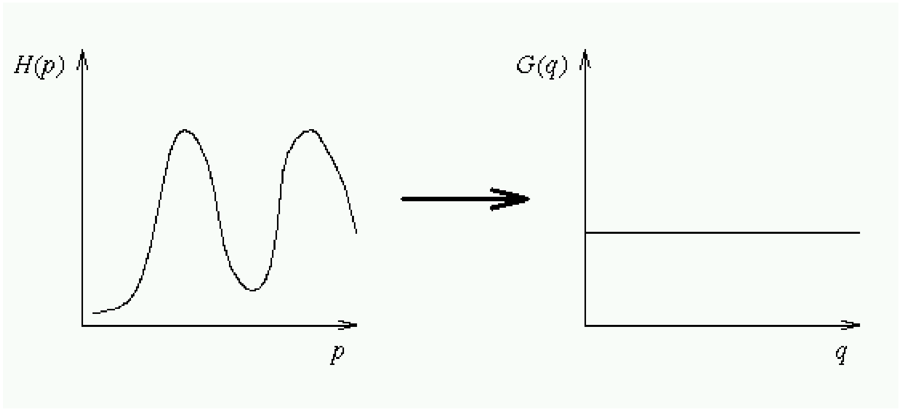
Histogram stretching

- Histogram stretching can be seen as a Window and Level contrast enhancement technique where the window ranges from the minimum to the maximum pixel values of the image.
- This normalization or histogram stretching operation is automatically performed in many display operators.
- To fully illustrate the difference between the two images above (original and stretched) a gray level scale was superimposed in both images to guarantee that the display operator will use the same gray-level scale.

Example 4.1.3. *Another Window and Level example. The Khoros workspace for this example is here Window Level Example.*

Histogram equalization

- Histogram equalization is another typical gray level transform.
- The aim is *to produce an image with equally distributed brightness levels over the whole brightness scale*.



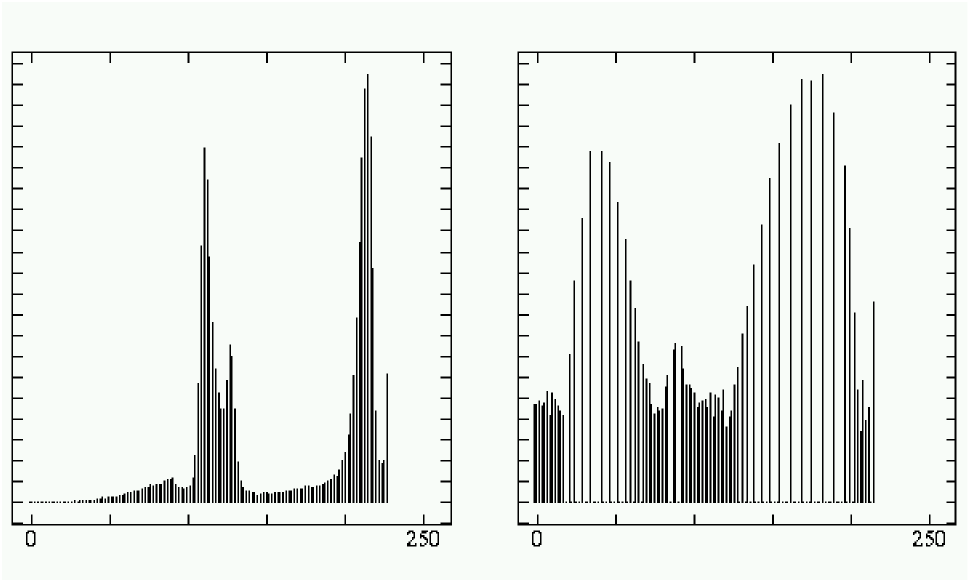
- Let $H(p)$ be the input histogram and that the input gray-scale is $[p_0, p_k]$.

- The intention is to find a monotonic pixel brightness $q = T(p)$ such that the output histogram $G(q)$ is uniform over the whole output brightness scale $[q_0, q_k]$.
- The histogram is a discrete probability density function.
- The monotonic property of the transform T implies

$$\sum_{i=0}^{j} G(q_i) = \sum_{i=0}^{j} H(p_i) \tag{4.4}$$

  where $q_i = T(p_i)$.
- The sum in the above equation can be interpreted as discrete distribution function.

- The equalized histogram can be obtained precisely only for the "idealized" continuous probability density.



- Assume that the image has M rows and N columns. The total number of pixels is MN.
- The equalized histogram corresponding to the uniform probability density function f whose function value satisfies

$$f(q)(q_k - q_0) = MN. \tag{4.5}$$

- The continuous version of (4.4) is

$$\int_{q_0}^{q} f(r)\, dr = \int_{p_0}^{p} H(s)\, ds, \quad q = T(p). \tag{4.6}$$

- Therefore, we obtain,

$$MN \int_{q_0}^{q} \frac{1}{q_k - q_0}\, dr = \frac{MN(q - q_0)}{q_k - q_0} = \int_{p_0}^{p} H(s)\, ds. \tag{4.7}$$

- The desired pixel brightness transformation T can then be derived as

$$q = T(p) = \frac{q_k - q_0}{MN} \int_{p_0}^{p} H(s)\, ds + q_0. \tag{4.8}$$

- The integral in the above equation is called the cumulative histogram, which is approximated by a sum for digital images, for example, as follows,

$$F[p] = \int_{p_0}^{p} H(s)\,ds = \sum_{i=0}^{j} H(p_i), \quad p = \text{round}(p_j) \tag{4.9}$$

- So the resulting histogram is not equalized ideally.

- The discrete approximation of the continuous pixel brightness transformation form the above equation is

$$q_j = T(p_j) = \frac{q_k - q_0}{MN} \sum_{i=0}^{j} H(p_i) + q_0 \tag{4.10}$$

- The algorithm to perform equalization is as follows

Algorithm 4.1.4. *Histogram equalization*

1. *For an $N \times M$ image of G gray-levels (often 256), create two arrays H and T of length G initialized with* 0 *values.*

2. *Form the image histogram: scan every pixel and increment the relevant member of H — if pixel X has intensity p, perform*

$$H[p] = H[p] + 1 \tag{4.11}$$

3. *Form the cumulative image histogram H_c. We may use the same array H to stored the result.*

$$\begin{aligned} H[0] &= H[0] \\ H[p] &= H[p-1] + H[p] \end{aligned}$$

   for $p = 1, \cdots, G-1$.

4. *Set*

$$T[p] = round\left[\frac{G-1}{MN} H[p]\right]. \tag{4.12}$$

   Note the new gray-scale is assumed the same as the input image, i.e., $q_k = G-1$ and $q_0 = 0$.

5. *Rescan the image and write an output image with gray-levels q, setting*

$$q = T[p]. \tag{4.13}$$

Example 4.1.5. *Histogram equalization example. The Khoros workspace for this example is here Histogram equalization Example.*

Example 4.1.6. *Histogram equalization example. The matlab script from* **vision-book** *is is here Histogram equalization Example.*

Remark 4.1.7. *([Klette and Zamperoni, 1996], p. 148)*

- *The gray value range equalization may be used for improving the image quality if the original image covers only a part of the full gray scale.*
- *An insufficient exploitation of the full gray scale is mostly due to image acquisition circumstances, as e.g., low scene illumination or automatic gain control of the camera.*
- *In case of good gray value dynamics of the input image, an equalization can lead even to quality losses in form of unsharp edges.*

Example 4.1.8. *Another histogram equalization example. The Khoros workspace for this example is here Histogram equalization Example.*

Some regions/edges disappear after equalization.

Histogram matching

- The generalization of histogram equalization is called histogram matching.
- The aim is to produce an image with desired distributed brightness levels over the whole brightness scale.
- Assume the desired probability density function is $G(q)$.
- Let the desired pixel brightness transform be T.
- Similarly we have

  $$F[p] = \int_{q_0}^{q=T[p]} G[s]\, ds \tag{4.14}$$

  where $F[p]$ is the cumulative histogram of the input image. Assumed that the cumulative histogram is normalized in $[0, 1]$.
- From the above equation, it is possible to find the transformation T.
- E.g., if G is the exponential distribution,

  $$G[q] = \alpha \mathbf{e}^{-\alpha(q-q_0)} \tag{4.15}$$

  for $q \geq q_0$.

- We have
$$F[p] = 1 - \mathbf{e}^{-\alpha(T[p]-q_0)} \tag{4.16}$$
- Then we find the transformation
$$T[p] = q_0 - \frac{1}{\alpha}\log(1 - F[p]) \tag{4.17}$$
- In the discrete case, the transformation can be implemented by building look-up tables.

Homework 4.1.9. *Find in the reference book B, (1.5.2), p. 81, some other probability density functions and the related transforms. Implementation of those transforms are left as homework. The homework includes:*

1. *Simulation of the brightness correction;*
2. *Grey scale transformations with given functions such as exponent, logarithm, user defined (by some control points for a piece wise linear function), etc.;*
3. *Histogram stretching to a given range;*
4. *Histogram equalization;*
5. *Histogram matching to given density functions;*
6. ★ *Histogram matching to the histogram of a given image.*
7. *Histogram should be plotted when histogram is involved.*
8. *Your final score will also depend on the interface of your program. Be sure to give us a friendly interface.*
9. *Be sure to choose illustrative images.*

4.2 Geometric transformations

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed.

- An example is an attempt to match remotely sensed images of the same area taken after one year, when the more recent image was probably not taken from precisely the same position.

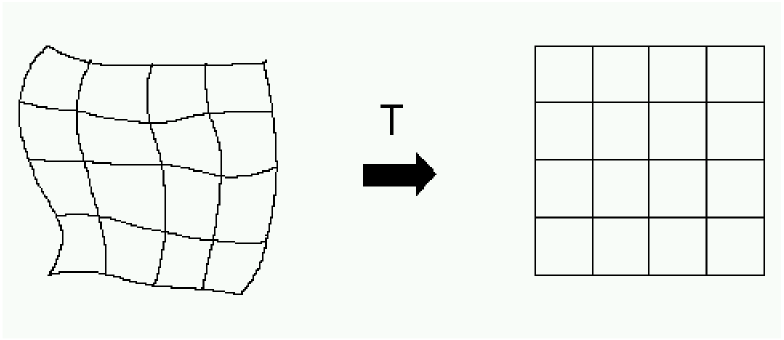
- To inspect changes over the year, it is necessary first to execute a geometric transformation, and then subtract one image from the other, to determine if more geometric calibration is needed.

- Another example, commonly encountered in document image processing applications, is correcting for document skew, which occurs when an image with an obvious orientation (e.g., a printed page) is scanned, or otherwise captured, at a different orientation.

- This orientation difference may be very small, but can be critical if the orientation is exploited in subsequent processing — this is usually the case in optical character recognition (OCR).

- A geometric transform is a vector function T that maps the pixel (x, y) to a new position (x', y'),

$$x' = T_x(x, y) \quad y' = T_y(x, y) \tag{4.18}$$



- The transformation equations are
  - either known in advance,
  - or can be determined from known original and transformed images,
  - or can be estimated from known obvious orientations (e.g., OCR applications).
  - Several pixels in both images with known correspondence are used to derive the unknown transformation.

- A geometric transform consists of two basic steps:

  1. determining the pixel co-ordinate transformation
     - mapping of the co-ordinates of the input image pixel to the point in the output image.
     - the output point co-ordinates should be computed as continuous values (real numbers) as the position does not necessarily match the digital grid after the transform.
  2. determining the brightness of the points in the digital grid.

- The brightness values are usually computed as an interpolation of the brightnesses of several points in the neighborhood.
- This idea enables the classification of geometric transformation among other pre-processing techniques, the criterion being that only the neighborhood of a processed pixel is needed for the calculation.
- Geometric transformations are on the boundary between point and local operations.

4.2.1 Pixel co-ordinate transformations

Polynomial approximation

- General case of finding the co-ordinates of a point in the output image after a geometric transform.
    - usually approximated by a polynomial equation (of degree m)

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk} x^r y^k \tag{4.19}$$

$$y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk} x^r y^k. \tag{4.20}$$

- This transform is linear with respect to the coefficients a_{rk} and b_{rk}.
- If pairs of corresponding points (x, y), (x', y') in both images are known, it is possible to determine a_{rk} and b_{rk} by solving a set of linear equations.
- More points than coefficients are usually used to get robustness. The mean square method (least squares fitting) is often used.

- If the geometric transform does not change rapidly depending on position in the image, low order approximating polynomials, $m = 2$ or $m = 3$, are used, needing at least 6 or 10 pairs of corresponding points.
- The corresponding points should be distributed in the image in a way that can express the geometric transformation — usually they are spread uniformly.
- The higher the degree of the approximating polynomial, the more sensitive to the distribution of the pairs of corresponding points the geometric transform.

- A geometric transform applied to the whole image may change the co-ordinate system, and a Jacobean J provides information about how the co-ordinate system changes
$$J(x, y) = \frac{\partial(x', y')}{\partial(x, y)} \tag{4.21}$$
- The area of the image is invariant if and only if $|J| = 1$.

Bilinear transform

- In practice, the geometric transform is often approximated by the bilinear transformation:

$$x' = a_0 + a_1 x + a_2 y + a_3 xy, \tag{4.22}$$
$$y' = b_0 + b_1 x + b_2 y + b_3 xy. \tag{4.23}$$

- 4 pairs of corresponding points are sufficient to find transformation coefficients.

- Even simpler is the affine transformation for which three pairs of corresponding points are sufficient to find the coefficients:

$$x' = a_0 + a_1 x + a_2 y \tag{4.24}$$
$$y' = b_0 + b_1 x + b_2 y \tag{4.25}$$

- The affine transformation includes typical geometric transformations such as rotation, translation, scaling and skewing (shear).

Important transformations

- Rotation by the angle ϕ

$$x' = x\cos\phi + y\sin\phi \tag{4.26}$$
$$y' = -x\sin\phi + y\cos\phi \tag{4.27}$$
$$J = 1 \tag{4.28}$$

- Change of scale a in the x-axis and b in the y-axis

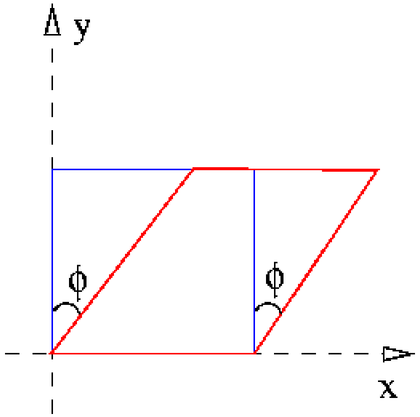
$$x' = ax \tag{4.29}$$
$$y' = by \tag{4.30}$$
$$J = ab \tag{4.31}$$

- Skew by the angel ϕ

$$x' = x + y \tan\phi \tag{4.32}$$

$$y' = y \tag{4.33}$$

$$J = 1 \tag{4.34}$$



- It is possible to approximate complex geometric transformations (distortion) by partitioning an image into smaller rectangular sub-images.
- For each sub-image, a simple geometric transformation, such as the affine, is estimated using pairs of corresponding pixels.
- The geometric transformation (distortion) is then performed separately in each sub-image.

4.2.2 Brightness interpolation

- Assume that the planar transformation has been accomplished, and new point co-ordinates (x', y') were obtained.
- The position of the pixel transformed does not in general fit the discrete grid of the output image.
- Values on the integer grid are needed.
- Each pixel value in the output image can be obtained by brightness interpolation of some neighboring non-integer samples, transformed from the input image.
- The brightness interpolation problem is usually expressed in a dual way (by determining the brightness of the original point in the input image that corresponds to the point in the output image lying on the discrete raster).
- Assume that we wish to compute the brightness value of the pixel (x', y') in the output image where x' and y' lie on the discrete grid.
- The co-ordinates of the point (x, y) in the original image can be obtained by inverting the transformation

$$(x, y) = T^{-1}(x', y'). \tag{4.35}$$

- In general the real co-ordinates (x, y) after inverse transformation do not fit the input image discrete grid, and so brightness is not known.

- To get the brightness value of the point (x, y) the input image is re-sampled or interpolated:

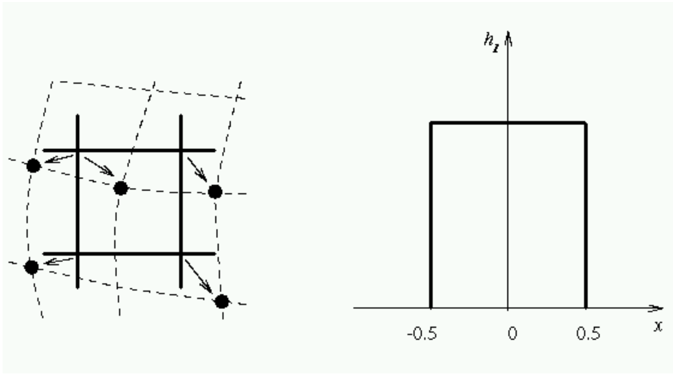
$$f_n(x, y) = \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} g_s(l\Delta x, k\Delta y) h_n(x - l\Delta x, y - k\Delta y) \tag{4.36}$$

  where $f_n(x, y)$ is the result of interpolation and h_n is the interpolation kernel. n distinguishes different interpolation methods.

- Usually, a small neighborhood is used, outside which h_n is zero.

Nearest neighbor interpolation

- Assign to the point (x, y) the brightness value of the nearest point g in the discrete raster. The right side of the above figure shows how the new bright-



  ness is assigned. Dashed lines show how the inverse planar transformation maps the grids of the output image into the input image — solid lines show the grids of the input image.

- Nearest neighbor interpolation is given by

$$f_1(x, y) = g_s(\text{round}(x), \text{round}(y)). \tag{4.37}$$

Remark 4.2.1. *The interpolation kernel h_1 as in (4.36) is*

$$h_1(x, y) = h_1^1(x)h_1^1(y), \tag{4.38}$$

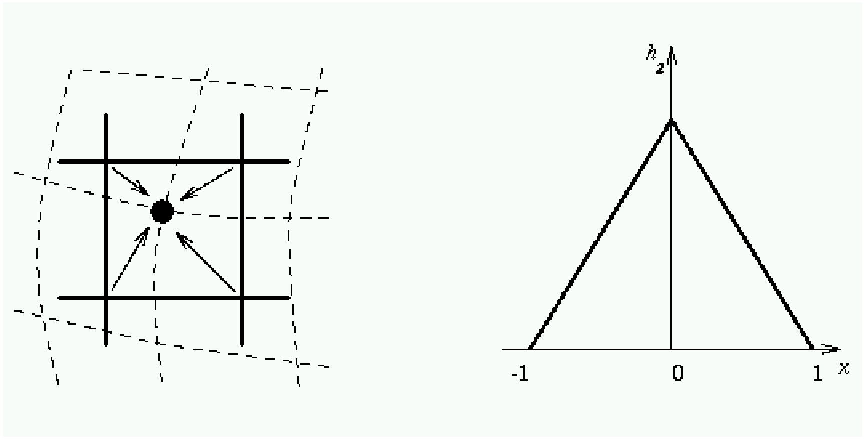
where,

$$h_1^1(t) = \begin{cases} 1, & \textit{if } t \in [-0.5, 0.5], \\ 0, & \textit{otherwise}. \end{cases} \tag{4.39}$$

- The position error of the nearest neighborhood interpolation is at most half a pixel.
- This error is perceptible on objects with straight line boundaries that may appear step-like after the transformation.

Bilinear interpolation

- Bilinear interpolation explores four points neighboring the point (x, y), and assumes that the brightness function is bilinear in this neighborhood.



- Bilinear interpolation is given by

$$\begin{aligned} f_2(x,y) =&(1-a)(1-b)g_s(l,k)+a(1-b)g_s(l+1,k)\\ &(1-a)bg_s(l,k+1)+abg_s(l+1,k+1)\\ =&g_s(l,k)\\ &+(g_s(l+1,k)-g_s(l,k))a\\ &+(g_s(l,k+1)-g_s(l,k))b\\ &+(g_s(l,k)+g_s(l+1,k+1)-g_s(l+1,k)-g_s(l,k+1))ab, \end{aligned}$$

where

$$l=\mathrm{floor}(x), \quad a=x-l, \tag{4.40}$$

$$k=\mathrm{floor}(y), \quad b=y-k. \tag{4.41}$$

Proof. Note by construction

$$x = a \cdot (l+1) + (1-a) \cdot l \tag{4.42}$$
$$y = b \cdot (k+1) + (1-b) \cdot k \tag{4.43}$$

Since f_2 is bilinear,

$$f_2(x,k) = (1-a)g_s(l,k) + ag_s(l+1,k) \tag{4.44}$$
$$f_2(x,k+1) = (1-a)g_s(l,k+1) + ag_s(l+1,k+1). \tag{4.45}$$

Then

$$\begin{aligned} f_2(x,y) =& bf_2(x,k+1) + (1-b)f_2(x,k) \\ =& b(1-a)g_s(l,k+1) + bag_s(l+1,k+1) \\ &+ (1-b)(1-a)g_s(l,k) + (1-b)ag_s(l+1,k) \end{aligned}$$

□

Remark 4.2.2. *The interpolation kernel h_2 is*

$$h_2(x, y) = h_2^1(x)h_2^1(y), \tag{4.46}$$

where

$$h_2^1(t) = h_1^1 * h_1^1(t) = \begin{cases} 1 - t, & \text{if } t \in [0, 1], \\ t + 1, & \text{if } t \in [-1, 0], \\ 0, & \text{otherwise.} \end{cases} \tag{4.47}$$

- Linear interpolation can cause a small decrease in resolution and blurring due to its averaging nature.
- The problem of step like straight boundaries with the nearest neighborhood interpolation is reduced.

Bi-cubic interpolation

- Bi-cubic interpolation improves the model of the brightness function by approximating it locally by a bicubic polynomial surface; 16 neighboring points are used for interpolation.

- interpolation kernel ('Mexican hat') is defined via

$$h_3^1(t) = \begin{cases} 1 - 2|t|^2 + |t|^3, & \text{if } |t| < 1 \\ 4 - 8|t| + 5|t|^2 - |t|^3, & \text{if } 1 \le |t| < 2 \\ 0, & \text{otherwise} \end{cases} \tag{4.48}$$

  by

$$h_3(x, y) = h_3^1(x) h_3^1(y). \tag{4.49}$$

- Bicubic interpolation does not suffer from the step-like boundary problem of nearest neighborhood interpolation, and copes with linear interpolation blurring as well.

- Bicubic interpolation is often used in raster displays that enable zooming to an an arbitrary scale.

- Bicubic interpolation preserves fine details in the image very well.

Example 4.2.3. *Geometric transform example. The Khoros workspace for this example is here Geometric transform Example*

Example 4.2.4. *Geometric transform example. The matlab script from* **visionbook** *is is here Geometric transform Example.*

4.3 Local pre-processing

- Pre-processing methods use a small neighborhood of a pixel in an input image to get a new brightness value in the output image.
- Such pre-processing operations are called also filtration (or filtering) if signal processing terminology is used.
- Local pre-processing methods can be divided into the two groups according to the goal of the processing:
  - First, `smoothing` aims to suppress noise or other small fluctuations in the image.
  - Smoothing is equivalent to the suppression of high frequencies in the frequency domain.
  - Unfortunately, smoothing also blurs all sharp edges that bear important information about the image.
  - If objects are rather large, an image can be enhanced by smoothing of small degradations.
  - Smoothing operators will benefit if some general knowledge about image degradation is available; this might, e.g., be statistical parameters of the noise.

- Second, `gradient operators` are based on local derivatives of the image function.
- Derivatives are bigger at locations of the image where the image function undergoes rapid changes.
- The aim of gradient operators is to indicate such locations in the image.
- Gradient operators have a similar effect as suppressing low frequencies in the frequency domain.
- Noise is often high frequency in nature; unfortunately, if a gradient operator is applied to an image, the noise level increases simultaneously.

- Clearly, smoothing and gradient operators have conflicting aims.
- Some pre-processing algorithms solve this problem and permit smoothing and edge enhancement simultaneously.

- Another classification of local pre-processing methods is according to the transformation properties.

- Linear and nonlinear transformations can be distinguished.

- Linear operations calculate the resulting value in the output image pixel $g(i, j)$ as a linear combination of brightnesses in a local neighborhood of the pixel $f(i, j)$ in the input image.

- The contribution of the pixels in the neighborhood is weighted by coefficients h
$$f(i, j) = \sum_{(m,n)\in\mathcal{O}}\sum h(i - m, j - n)g(m, n) \tag{4.50}$$

- The above equation is equivalent to discrete convolution with the kernel h, that is called a `convolution mask`.

- Rectangular neighborhoods $\mathcal{O}$ are often used with an odd number of pixels in rows and columns, enabling the specification of the central pixel of the neighborhood.

- The choice of the local transformation, size, and shape of the neighborhood $\mathcal{O}$ depends strongly on the size of objects in the processed image.

- Convolution-based operators (filters) can be used for smoothing, gradient operators, and line detectors.

- There are methods that enable the speed-up of calculations to ease implementation in hardware — examples are recursive filters or separable filters.

STOP

- Local pre-processing methods typically use very little *a priori* knowledge about the image contents.

- It is very difficult to infer this knowledge while an image is being processed, as the known neighborhood $\mathcal{O}$ of the processed pixel is small.

4.3.1 Image smoothing

- Image smoothing is the set of local pre-processing methods whose predominant use is the suppression image noise — it uses redundancy in the image data.

- Calculation of the new value is based on averaging of brightness values in some neighborhood $\mathcal{O}$.

- Smoothing poses the problem of blurring sharp edges in the image, and so we shall study smoothing methods which are edge preserving.

- They are based on the general idea that the average is computed only from those points in the neighborhood which have similar properties to the processed point.

- Local image smoothing can effectively eliminate impulsive noise or degradations appearing as thin stripes, but does not work if degradations are large blobs or thick stripes.

- The solution for complicated degradations may be to use image restoration techniques, described in section § 4.4.

Averaging

- Assume that the noise value ν at each pixel is an independent random variable with zero mean and standard deviation σ.
- We can obtain such an image by capturing the same static scene several times.
- The result of smoothing is an average of the same n points in these images $g_1, \cdots, g_n$ with noise values $1, \cdots, n$.

$$\frac{g_1 + \cdots + g_n}{n} + \frac{\nu_1 + \cdots + \nu_n}{n} \tag{4.51}$$

- The second term here describes the effect of the noise, which is again a random value with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$. The standard deviation is decreased by a factor $\sqrt{n}$.
- Thus if n images of the same scene are available, the smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j) \tag{4.52}$$

- In many cases only one image with noise is available, and averaging is then realized in a local neighborhood.
- Results are acceptable if the noise is smaller in size than the smallest objects of interest in the image, but blurring of edges is a serious disadvantage.
- In the case of smoothing within a single image, one has to assume that there are no changes in the gray levels of the underlying image data.
- This assumption is clearly violated at locations of image edges, and edge blurring is a direct consequence of violating the assumption.
- Averaging is a special case of discrete convolution. For a 3×3 neighborhood the convolution mask h is

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.53}$$

- The significance of the central pixel may be increased, as it approximates the properties of noise with a Gaussian probability distribution.

$$h = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.54}$$

- Larger convolution masks for averaging are created analogously.

Example 4.3.1. *Noise suppression. An image corrupted with different additive noise is used to demonstrate the effect of averaging. The Khoros workspace for this example is here Average Example.*

Averaging with data validity

- Methods that average with limited data validity try to avoid blurring by averaging only those pixels which satisfy some criterion, the aim being to prevent involving pixels that are part of a separate feature.

- A very simple criterion is to use only pixels in the original image with brightness in a predefined interval [min, max].

- Considering the point (m, n) in the image, the convolution mask is calculated in the neighborhood $\mathcal{O}$ from the nonlinear formula

$$h(i,j) = \begin{cases} 1, & \text{for } g(m+i, n+j) \in [\text{min}, \text{max}] \\ 0, & \text{otherwise.} \end{cases} \tag{4.55}$$

  Note that in the equation above, the interval [min, max] represents valid data.

- The second method performs the averaging only if the computed brightness change of a pixel is in some predefined interval.

- This method permits repair to large-area errors resulting from slowly changing brightness of the background without affecting the rest of the image.

- The third method uses edge strength (i.e., magnitude of a gradient) as a criterion.
- The magnitude of some gradient operator is first computed for the entire image, and only pixels in the input image with a gradient magnitude smaller than a predefined threshold are used in averaging.

Averaging according to inverse gradient

- The convolution mask is calculated at each pixel according to the inverse gradient.
- The idea is that brightness change within a region is usually smaller than between neighboring regions.
- Let (i, j) be the central pixel of a convolution mask with odd size; the inverse gradient at the point (m, n) with respect to (i, j) is then

$$\delta(i, j, m, n) = \begin{cases} \frac{1}{|g(m,n) - g(i,j)|}, & \text{if } g(m, n) \neq g(i, j); \\ 2 & \text{if } g(m, n) = g(i, j). \end{cases} \tag{4.56}$$

- The inverse gradient is then in the interval $(0, 2]$, and is smaller on the edge than in the interior of a homogeneous region.
- Weight coefficients in the convolution mask h are normalized by the inverse gradient,

$$h(i, j, m, n) = \frac{\delta(i, j, m, n)}{\sum_{(m',n') \in \mathcal{O}} \delta(i, j, m', n')}, \quad \text{if } (m, n) \in \mathcal{O}. \tag{4.57}$$

- The above method assumes sharp edges.
- Isolated noise points within homogeneous regions have small values of the inverse gradient; points from the neighborhood take part in averaging and the noise is removed.
- When the convolution mask is close to an edge, pixels from the region have larger coefficients than pixels near the edge, and it is not blurred.

Averaging using a rotating mask

- This method avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood.
- The resulting image is in fact sharpened.
- Brightness average is calculated only within the homogeneous region.
- A brightness dispersion σ^2 is used as the region homogeneity measure.
- Let n be the number of pixels in a region R and $g(i,j)$ be the input image. Dispersion σ^2 is calculated as

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left[g(i,j) - \frac{1}{n} \sum_{(i',j')\in R} g(i',j') \right]^2 \tag{4.58}$$

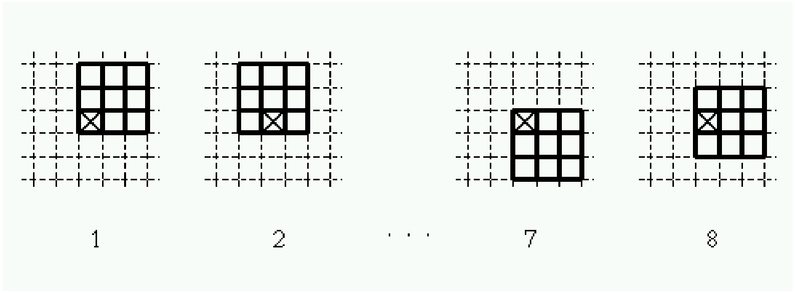
- The computational complexity (number of multiplications) of the dispersion calculation can be reduced if expressed as follows

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left\{ g(i,j)^2 - 2g(i,j) \frac{\sum_{(i',j')\in R} g(i',j')}{n} + \left[\frac{\sum_{(i',j')\in R} g(i',j')}{n} \right]^2 \right\}$$

$$= \frac{1}{n} \left\{ \sum_{(i,j)\in R} g(i,j)^2 - 2\frac{\left[\sum_{(i',j')\in R} g(i',j')\right]^2}{n} + n\left[\frac{\sum_{(i',j')\in R} g(i',j')}{n}\right]^2 \right\}$$

$$= \frac{1}{n} \left\{ \sum_{(i,j)\in R} g(i,j)^2 - \frac{\left[\sum_{(i,j)\in R} g(i,j)\right]^2}{n} \right\}$$

- Having computed region homogeneity, we consider its shape and size.
- The eight possible 3×3 masks that cover a 5×5 neighborhood of a current pixel (marked by small cross in the following figure) are shown in the following figure The ninth mask is the 3×3 neighborhood of the current pixel itself.

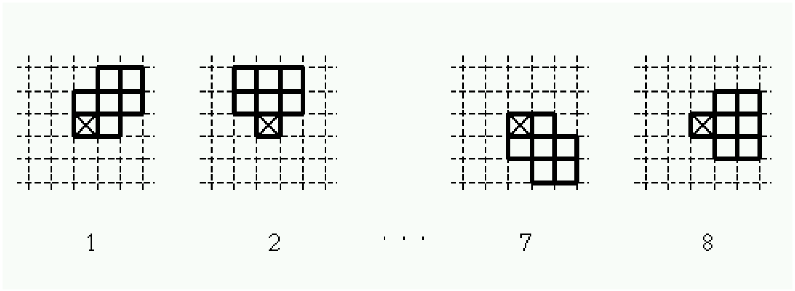


- Image smoothing using the rotating mask technique uses the following algorithm.

Algorithm 4.3.2. Smoothing using a rotating mask

1. *Consider each image pixel (i, j). Calculate dispersion in the mask for all possible mask rotations about pixel (i, j).*
2. *Choose the mask with minimum dispersion.*
3. *Assign to the pixel (i, j) in the output image the average brightness in the chosen mask.*

- Other mask shapes can also be used.
- The following figure shows another set of eight masks covering a 5×5 neighborhood of the current pixel Again the ninth mask is the 3×3 neighborhood



of the current pixel itself.

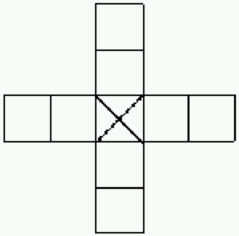
- Another possibility is to rotate a small 2×1 mask to cover the 3×3 neighborhood of the current pixel.
- This algorithm can be used iteratively. (What about other algorithms?)
- The iterative process converges quite quickly to stable state (that is, the image does not change any more).

Example 4.3.3. *Rotating mask example. The matlab script from* **visionbook** *is is here Rotating mask Example.*

Median filtering

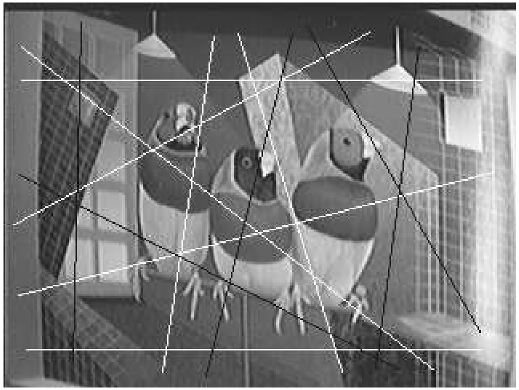
- In a set of ordered values, the median is the central value.
- Median filtering assigns the output pixel brightness value to be the median value of the brightness values in a neighborhood of the pixel.
- Median filtering reduces blurring of edges.
- The idea is to replace the current point in the image by the median of the brightness in its neighborhood.
- The median of the brightness in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulsive noise quite well.
- As median filtering does not blur edges much, it can be applied iteratively.

- The main disadvantage of median filtering in a rectangular neighborhood is its damaging of thin lines and sharp corners in the image — this can be avoided if another shape of neighborhood is used.



- Variants of median filtering is to choose the maximum and minimum values in the neighborhood. This leads to the dilation and erosion operators in mathematical morphology.

Example 4.3.4. *The original image Birds.gif that was corrupted with black and white lines can be found in the data directory. This example is to remove the lines*



from the image using median filtering. The Khoros workspace for this example is here Median Example.

Example 4.3.5. *Median Filter example. The matlab script from* **visionbook** *is is here Median Filter Example.*

Non-linear mean filtering

- The non-linear mean filter is another generalization of averaging techniques.
- It is defined by

$$f(i,j) = u^{-1}\left(\frac{\sum_{(m,n)\in\mathcal{O}} a(m,n)u[g(m,n)]}{\sum_{(m,n)\in\mathcal{O}} a(m,n)}\right) \tag{4.59}$$

  where $f(i,j)$ is the result of the filtering, $g(m,n)$ is the pixel in the input image, and $\mathcal{O}$ us a local neighborhood of the current pixel (i,j).

- The function u of one variable has an inverse function u^{-1}. The $a(m,n)$ are weight coefficients.
- If the coefficients $a(i,j)$ are constants, the filter is called homomorphic.
- Some homomorphic filters used in image processing are
  - Arithmetic mean, $u(g) = g$.
  - Harmonic mean, $u(g) = \frac{1}{g}$.
  - Geometric mean, $u(g) = \log g$.

Example 4.3.6. *Non-linear mean filtering. The matlab script to run this example is here Non-linear mean filtering Example.*

Variational properties of some local smoothing operators

Theorem 4.3.7. *Given $x_1 \le x_2 \le \cdots \le x_N$, then*

1. $\mathbf{arg\,min}_a \sum_{i=1}^N |x_i - a|^2$ *is the arithmetic mean of* $x_1, x_2, \cdots, x_N$*;*
2. $\mathbf{arg\,min}_a \sum_{i=1}^N |x_i - a|$ *is the median of* $x_1, x_2, \cdots, x_N$*;*
3. $\mathbf{arg\,min}_a \max_{1 \le i \le N} |x_i - a|$ *is* $\frac{x_1 + x_N}{2}$.

Proof. (1) Let

$$g(a) = \sum_{i=1}^{N} |x_i - a|^2. \tag{4.60}$$

The result follows immediately by calculus.

(2) Let

$$g(a) = \sum_{i=1}^{N} |x_i - a|. \tag{4.61}$$

Let a_0 and a_1 be arbitrary number such that

$$a_0 < x_1 \leq x_N < a_1 \tag{4.62}$$

Then

$$g(a_0) = \sum_{i=1}^{N} x_i - Na_0 \tag{4.63}$$

and

$$g(a_1) = Na_1 - \sum_{i=1}^{N} x_i. \tag{4.64}$$

If $a \in [x_1, x_N]$, assume that

$$x_k \leq a < x_{k+1}. \tag{4.65}$$

Then

$$g(a) = \sum_{i=1}^{k}(a - x_i) + \sum_{i=k+1}^{N}(x_i - a) = \sum_{i=k+1}^{N} x_i - \sum_{i=1}^{k} x_i + (2k - N)a. \tag{4.66}$$

If $2k \leq N$,

$$\begin{aligned} g(a_0) - g(a) &= 2\sum_{i=1}^{k} x_i - Na_0 + (N - 2k)a \\ &= 2\sum_{i=1}^{k}(x_i - a_0) + (N - 2k)(a - a_0) \geq 0. \end{aligned}$$

If $2k \geq N$,

$$\begin{aligned} g(a_1) - g(a) &= Na_1 - 2\sum_{i=k+1}^{N} x_i + (N - 2k)a \\ &= 2\sum_{i=k+1}^{N}(a_1 - x_i) + (N - 2(N - k))a_1 + (N - 2k)a \\ &= 2\sum_{i=k+1}^{N}(a_1 - x_i) + (2k - N)(a_1 - a) \geq 0. \end{aligned}$$

Therefore, the minimum of $g(a)$ is attained in $[x_1, x_N]$. Note that $g(a)$ is continuous, piece wise linear function on $[x_1, x_N]$. $g(a)$ is decreasing if $2k \leq N$ and increasing if $2k \geq N$.

If $N = 2m$ is even, the minimum of $g(a)$ is attained in the central sub-interval $[x_m, x_{m+1}]$. Since $g(a)$ is constant on this sub-interval, any value of $[x_m, x_{m+1}]$ is a minimizer of $g(a)$. We choose $a = \frac{x_m + x_{m+1}}{2}$ the median as the minimizer.

If $N = 2m+1$, $g(a)$ is decreasing in one of the central sub-interval $[x_m, x_{m+1}]$ and increasing in another central sub-interval $[x_{m+1}, x_{m+2}]$. Then x_{m+1} is the minimizer, which is the median in this case.

(3) Let

$$g(a) = \max_{1 \leq i \leq N} |x_i - a|. \tag{4.67}$$

Then it is easy to verify that

$$g(a) = \begin{cases} x_N - a, & \text{if } a < x_1; \\ x_N - a, & \text{if } x_1 \leq a < \frac{x_1 + x_N}{2}; \\ a - x_1, & \text{if } \frac{x_1 + x_N}{2} \leq a \leq x_N; \\ a - x_1, & \text{if } x_N < a. \end{cases} \tag{4.68}$$

The conclusion follows immediately. $\square$

4.3.2 Edge detectors

Edge: what is it?

- Edge detectors are a collection of very important local image pre-processing methods used to locate (sharp) changes in the intensity function.
- Edges are pixels where the brightness function changes abruptly.

STOP

- Neurological and psychophysical research suggests that locations in the image in which the function value changes abruptly are important for image perception.
- Edges are to a certain degree invariant to changes of illumination and viewpoint.
- If only edge elements with strong magnitude (edgels) are considered, such information often suffices for image understanding.
- The positive effect of such a process is that it leads to significant reduction of image data.

- Nevertheless such a data reduction does not undermine understanding the content of the image (interpretation) in many cases.

- Edge detection provides appropriate generalization of the image data.

- For instance, painters of line drawings perform such a generalization.

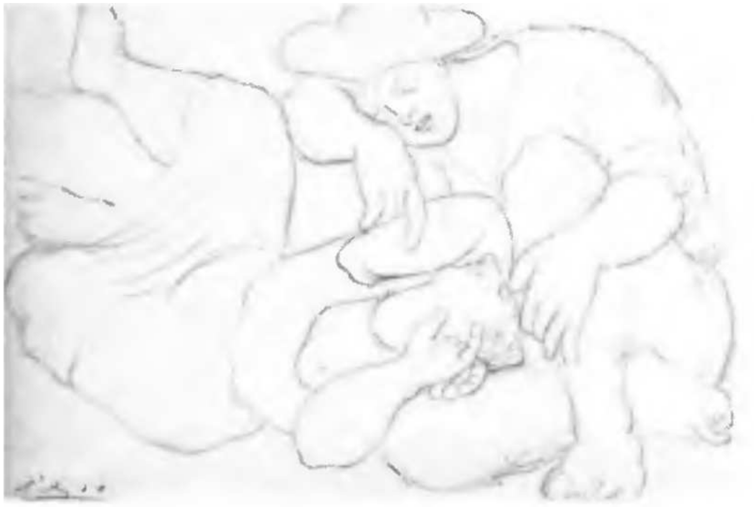
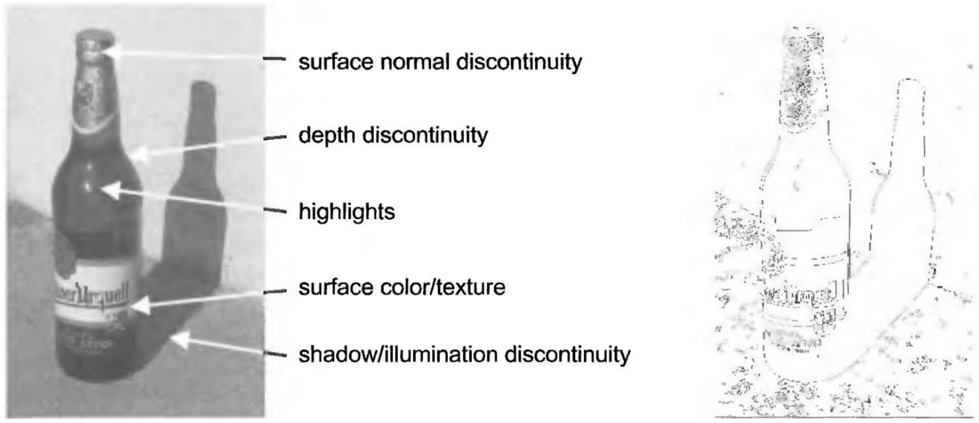


Figure 4.2: Siesta by Pablo Picasso, 1919

.

- We shall consider which physical phenomena in the image formation process lead to abrupt changes in image values.

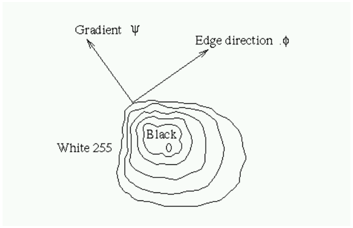

surface normal discontinuity
depth discontinuity
highlights
surface color/texture
shadow/illumination discontinuity

- Calculus describes changes of continuous functions using derivatives.

- An image function depends on two variables — co-ordinates in the image plane — so operators describing edges are expressed using partial derivatives.

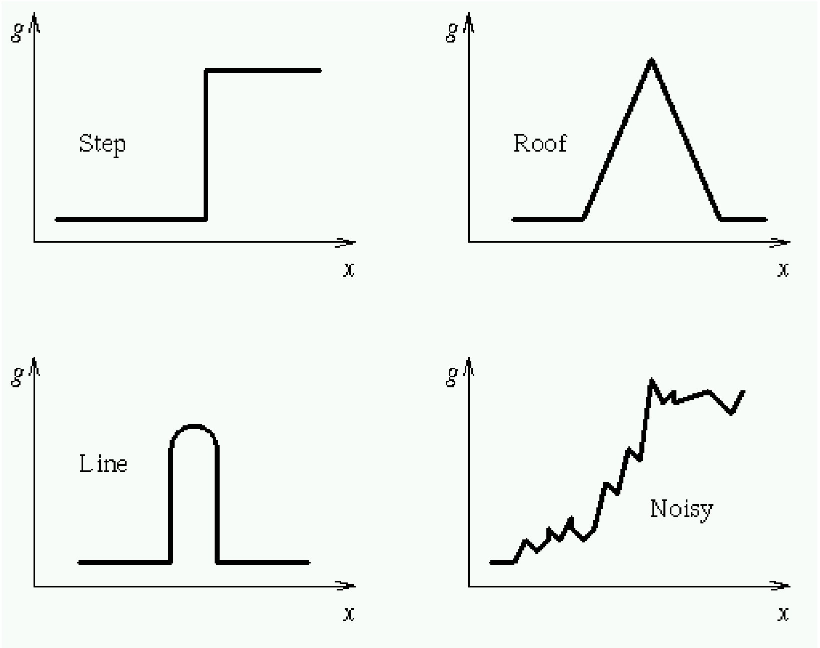
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.

- An edge is a (local) property attached to an individual pixel and is calculated from the image function in a neighborhood of the pixel.
- It is a **vector variable** with two components
  - magnitude of the gradient;
  - and direction ϕ is rotated with respect to the gradient direction ψ by -90^o.
- The gradient direction gives the direction of maximal growth of the function, e.g., from black ($g(i,j) = 0$) to white ($f(i,j) = 255$).
- This is illustrated below; closed contour lines are lines of the same brightness; the orientation 0^o points East.



- Edges are often used in image analysis for finding region boundaries.
- Boundary is at the pixels where the image function varies and consists of pixels with high(?) edge magnitude.
- Boundary and its parts (edges) are perpendicular to the direction of the gradient.

- The following figure shows several typical standard edge profiles.



Typical edge profiles.

- Roof edges are typical for objects corresponding to thin lines in the image.
- Edge detectors are usually tuned for some type of edge profile.

- Sometimes we are interested only in changing magnitude without regard to the changing orientation.
- A linear differential operator called the Laplacian may be used.
- The Laplacian has the same properties in all directions and is therefore invariant to rotation in the image.

$$\Delta g(x, y) = \frac{\partial^2 g(x, y)}{\partial x^2} + \frac{\partial^2 g(x, y)}{\partial y^2} \tag{4.69}$$

Finite Gradient

- The gradient magnitude and gradient direction are image functions,

$$|\text{grad}g(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \tag{4.70}$$

$$\psi = \text{arg}(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}) \tag{4.71}$$

  where $\text{arg}(u,v) = \text{arctan}(\frac{v}{u})$ is the angle (in radians) from the x-axis to the point (u,v).

- In practice, for fast computation, the magnitude is approximated by

$$|\text{grad}g(x,y)| = |\frac{\partial g}{\partial x}| + |\frac{\partial g}{\partial y}| \tag{4.72}$$

  or

$$|\text{grad}g(x,y)| = \max\{|\frac{\partial g}{\partial x}|, |\frac{\partial g}{\partial y}|\} \tag{4.73}$$

- A digital image is discrete in nature. So derivatives must be approximated by finite differences.

- The first order differences of the image g in the vertical direction (for fixed i) and in the horizontal direction (for fixed j) are given by backward difference

$$\Delta_i g(i,j) = \frac{g(i,j) - g(i-n,j)}{n} \tag{4.74}$$

$$\Delta_j g(i,j) = \frac{g(i,j) - g(i,j-n)}{n} \tag{4.75}$$

or by forward difference

$$\Delta_i g(i,j) = \frac{g(i+n,j) - g(i,j)}{n} \tag{4.76}$$

$$\Delta_j g(i,j) = \frac{g(i,j+n) - g(i,j)}{n} \tag{4.77}$$

- n is a small integer, usually 1.

- The value n should be chosen small enough to provide a good approximation to the derivative, but large enough to neglect unimportant changes in the image function.

- Symmetric expressions for the difference, i.e., central differences, are not usually used because they neglect the impact of the pixel (i,j) itself.

$$\Delta_i g(i,j) = \frac{g(i+n,j) - g(i-n,j)}{2n} \tag{4.78}$$

$$\Delta_j g(i,j) = \frac{g(i,j+n) - g(i,j-n)}{2n} \tag{4.79}$$

- Individual gradient operators that examine small local neighborhoods are in fact linear space-invariant operators, hence are equivalent to convolutions, cf. (4.50), and can be expressed by convolution masks.
- Each convolution mask corresponds to a derivative in one certain direction, if the masks induces edge orientation information.

Example 4.3.8. *Example: computing the gradient and edge magnitude and direction by finite difference. The Khoros workspace for this example is here Finite Difference Example.*

More on gradient operators

- Gradient operators as a measure of edge sheerness can be divided into three categories

  1. Operators approximating derivatives of the image function using finite differences (introduced above):
     - Some of them are rotationally invariant (e.g., the Laplacian) and direction independent and thus need one convolution mask only.
     - Others approximate first derivatives using several masks. The direction of the gradient is given by the mask giving maximal response. The absolute value of the response on that mask is the magnitude?
  2. Operators based on the zero crossings of the second derivatives of the image function (e.g., Marr-Hildreth or Canny edge detector).
  3. Operators which attempt to match an image function to a parametric model of edges.

- The remainder of this section will consider some of the many operators which fall into the first category, and the next section will consider the second category.
- The last category is briefly outlined in (4.3.6). Parametric models describe edges more precisely than simple edge magnitude and direction and are much more computationally intensive.
- This is an area of active research.
- It may be difficult to select the optimal edge detection strategy.

Roberts operator

- The Roberts operator is one of the oldest operators.
- It is very easy to compute as it uses only a 2×2 neighborhood of the current pixel.
- Its convolution masks are

$$h_1 = \begin{bmatrix} h_1(0,0) & h_1(1,0) \\ h_1(0,1) & h_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{4.80}$$

- The magnitude of the edge is computed as

$$|g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)| \tag{4.81}$$

- The primary disadvantage of the Roberts operator is its high sensitivity to noise, because very few pixels are used to approximate the gradient.

Discrete Laplacian

- The Laplace operator is a very popular operator approximating the second derivative which gives the gradient magnitude only.
- The Laplacian (4.69) is approximated in digital images by a convolution sum.
- A 3×3 mask h_4 is often used, for 4-neighborhoods and it is defined as

$$h_{4,1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \text{or} \qquad h_{4,2} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.82}$$

and for 8-neighborhoods it is defined as

$$h_{8,1} = \frac{h_{4,1} + 2h_{4,2}}{3} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.83}$$

or

$$h_{8,2} = \frac{4h_{4,2} - h_{4,1}}{3} = \frac{1}{3} \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \tag{4.84}$$

or

$$h_{8,3} = 3h_{4,1} - 2h_{4,2} = \begin{bmatrix} -1 & 3 & -1 \\ 3 & -8 & 3 \\ -1 & 3 & -1 \end{bmatrix} \tag{4.85}$$

- A Laplacian operator with stressed significance of the the central pixel or its neighborhood is sometimes used.

Prewitt operator

- The Prewitt operator, similarly to the Sobel, Kirsch, Robinson (as discussed later) and some other operators, approximates the first derivative.
- The gradient is estimated in eight (for a 3×3 convolution mask) possible directions.
- The convolution result of the greatest magnitude indicates the gradient magnitude.
- The convolution mask of greatest magnitude indicates the gradient direction.
- Larger masks are possible.
- Operators approximating first derivative of an image function are sometimes called **compass operators** because of the ability to determine gradient direction.
- We present only the first three 3×3 masks for each operator; the others can be created by simple repeated clockwise 45^o rotation.

- Prewitt operator

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \qquad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.86}$$

Sobel operator

- $$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \qquad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.87}$$

- The Sobel operator is often used as a simple detector of horizontality and verticality of edges. In this case only masks h_1 and h_3 are used.

- If the h_1 response is y and the h_3 response x, we might then derive edge strength (magnitude) as

$$\sqrt{x^2 + y^2} \qquad \text{or} |x| + |y| \tag{4.88}$$

and direction as

$$\arctan(\frac{y}{x}) \tag{4.89}$$

Robinson operator

- $$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix} \qquad h_3 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \tag{4.90}$$

Kirsch operator

- $$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix} \qquad h_3 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix} \tag{4.91}$$

Examples

Example 4.3.9. *Example: Roberts operator. The Khoros workspace for this example is here Roberts operator Example.*

Example 4.3.10. *Example: Laplacian operator. The Khoros workspace for this example is here Laplacian operator Example.*

Example 4.3.11. *Example: Prewitt operator. The Khoros workspace for this example is here Prewitt operator Example.*

Example 4.3.12. *Example: Sobel operator. The Khoros workspace for this example is here Sobel operator Example.*

Example 4.3.13. *Example: Robinson operator. The Khoros workspace for this example is here Robinson operator Example.*

Example 4.3.14. *Example: Kirsch operator. The Khoros workspace for this example is here Kirsch operator Example.*

Remark 4.3.15. 1. *Visually, the edge images produced by the foregoing edge operators (Roberts, Prewitt, Sobel, Kirsch, Robinson operators) appears rather similar.*

2. *The Roberts operator, being two by two, responds best on sharp transitions in low-noise images.*

3. *The other operators, being three by three, handle more gradual transition and noisier images better.*

Image sharpening

- Image sharpening makes edges steeper — the sharpened image is intended to be observed by a human.
- The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - C S(i,j) \tag{4.92}$$

- C is a positive coefficient which gives the strength of sharpening and $S(i,j)$ is a measure of the image function sheerness that is calculated using a gradient operator.
- The Laplacian is very often used to estimate $S(i,j)$.

- Image sharpening/edge detection can be interpreted in the frequency domain as well.

- The result of the Fourier transform is a combination of harmonic functions.

- The derivative of the harmonic function $\sin(nx)$ is $n\cos(nx)$; thus the higher the frequency, the higher the magnitude of its derivative. This is another explanation of why gradient operators enhance edges.

Example 4.3.16. *Example: Image sharpen by Laplacian. The Khoros workspace for this example is here Image sharpen Example.*

Example 4.3.17. *Image sharpen by Laplacian. The matlab script from* **visionbook** *is is here Image sharpen Example.*

Unsharp masking

- Unsharp masking is often used in printing industry applications — another image sharpening approach. (Note the conflicting name.)
- A signal proportional to an unsharp image (e.g., blurred by some smoothing operator) is subtracted from the original image, again a parameter C may be used to control the weight of the subtraction.

Example 4.3.18. *Example: Unsharp masking. The Khoros workspace for this example is here Unsharpen masking.*

4.3.3 Zero-crossings of the second derivative

- In the 1970's, Marr's theory conclude from neurophysiological experiments that object boundaries are the most important cues that link an intensity image with its interpretation.

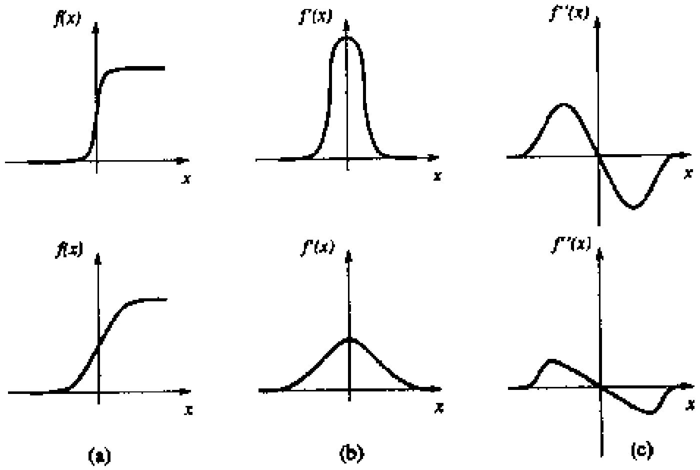
- Edge detection techniques at that time like the Kirsch, Sobel, Prewitt operators are based on convolution in very small neighborhoods and work well for specific images only.

- The main disadvantage of these edge detectors is their dependence on the size of objects and sensitivity to noise.

- An edge detection technique, based on the **zero crossings** of the second derivative (in its original form, the Marr-Hildreth edge detector) explores the fact that a step edge corresponds to an abrupt change in the image function.

- The first derivative of the image function should have an extreme at the position corresponding to the edge in the image, and so the second derivative should be zero at the same position.

- It is much easier and more precise to find a zero crossing position than an extreme - see the follwoing figure.



- The crucial question is how to compute the the 2nd derivative robustly.
- One possibility is to smooth an image first (to reduce noise) and then compute second derivatives.

- When choosing a smoothing filter, there are two criteria that should be fulfilled, [Marr and Hildreth, 1980].

   1. The filter should be smooth and roughly band limited in the frequency domain to reduce the possible number of frequencies at which function changes can take place.
   2. The constraint of spatial localization requires the response of a filter to be from nearby points in the image.

- These two criteria are conflicting — Heisenberg uncertainty principle.

- *A nonzero function and its Fourier transform cannot both be sharply localized*. [Folland and Sitaram, 1997, p. 207]

- ([Folland and Sitaram, 1997, Theorem 1.1]) For any $f \in L^2(\mathbf{R})$ and any $a \in \mathbf{R}$ and $b \in \mathbf{R}$,

$$\int (x-a)^2 |f(x)|^2 \, dx \int (\xi - b)^2 |\hat{f}(\xi)|^2 \, d\xi \geq \frac{\|f\|_2^4}{16\pi^2} \tag{4.93}$$

  Equality holds if and only if $f = C\mathbf{e}^{2\pi ibx - \gamma(x-a)^2}$ for some $C \in \mathbf{C}$ and $\gamma > 0$.

- [Folland and Sitaram, 1997, Theorem 7.6] For a, $b > 0$, let $E(a,b)$ be the space of all measurable functions f on $\mathbf{R}$ such that

$$|f(x)| \leq c\mathbf{e}^{-a\pi x^2}, |\hat{f}(\xi)| \qquad \leq c\mathbf{e}^{-b\pi\xi^2}, \tag{4.94}$$

  for some $c > 0$. Then

  (1) If $ab < 1$, $\dim E(a,b) = \infty$;

  (2) If $ab = 1$, $\dim E(a,b) = \mathbf{C}\mathbf{e}^{-a\pi x^2}$;

  (3) If $ab > 1$, $\dim E(a,b) = \{0\}$;

- There is a well known joke: "Heisenberg is pulled over by a policeman whilst driving down a motorway, the policeman gets out of his car, walks towards Heisenberg's window and motions with his hand for Heisenberg to wind the window down, which he does. The policeman then says 'Do you know what speed you were driving at sir?', to which Heisenberg responds 'No, but I knew exactly where I was."'[1]

[1] This is copy-edited from http:en.wikipedia.orgwikiUncertainty_principle.

- But they can be optimized simultaneously using a Gaussian distribution.
- The 2D Gaussian smoothing operator $G(x, y)$

$$G(x, y) = \frac{1}{2\pi\sigma^2}\mathbf{e}^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.95}$$

  where x and y are the image co-ordinates and σ is the standard deviation of the associated probability distribution.
- The standard deviation σ is the only parameter of the Gaussian filter — it is proportional to the size of neighborhood on which the filter operates.
- Pixels more distant from the center of the operator have smaller influence, and pixels further than 3σ from the center have negligible influence.

STOP

- Our goal is to get a second derivative of a smoothed 2D function $f(x, y)$.
- We have seen that the Laplacian operator gives the second derivative, and is non-directional (isotropic).

- Consider then the Laplacian of an image $f(x, y)$ smoothed by a Gaussian.
- This operator is abbreviated by some authors as **LoG**, from **Laplacian of Gaussian**:
$$\Delta[G(x, y) * f(x, y)] \tag{4.96}$$
- The order of differentiation and convolution can be interchanged due to linearity of the operations:
$$[\Delta G(x, y)] * f(x, y) \tag{4.97}$$
- The derivative of the Gaussian filter is independent of the image under consideration and can be precomputed analytically reducing the complexity of the composite operation.
$$\Delta G(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6}\mathbf{e}^{-\frac{r^2}{2\sigma^2}} \tag{4.98}$$
- Because its shape, the LoG operator is commonly called a Mexican hat.

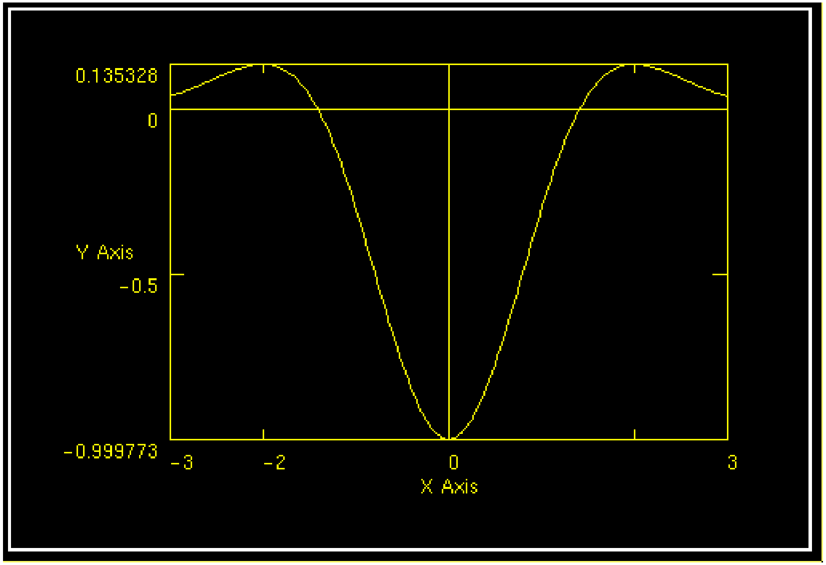


Figure 4.3: LoG filter.

- Finding second derivatives in this way is very robust(?).
- Gaussian smoothing effectively suppresses the influence of the pixels that are up to a distance 3σ from the current pixel; then the Laplace operator is an efficient and stable measure of changes in the image.

- The location in the LoG image where the zero level is crossed corresponds to the position of the edges.
- The advantage of this approach compared to classical edge operators of small size is that a larger area surrounding the current pixel is taken into account; the influence of more distant points decreases according to the variance σ of the Gaussian.

- Convolution masks become large for larger.
- Fortunately, there is a separable decomposition of the ΔG operator that can

speed up computation considerable.

- The practical implication of Gaussian smoothing is that edges are found reliably.
- If only globally significant edges are required, the standard deviation of the Gaussian smoothing filter may be increased, having the effect of suppressing less significant evidence.

- The LoG operator can be very effectively approximated by convolution with a mask that is the difference of two Gaussian averaging masks with substantially different — this method is called the Difference of Gaussians — DoG.
- Even coarser approximations to LoG are sometimes used — the image is filtered twice by an averaging operator with smoothing masks of different size and the difference image is produced.

- When implementing a zero-crossing edge detector, trying to detect zeros in the LoG or DoG image will inevitably fail, while naive approaches of thresholding the LoG/DoG image and defining the zero-crossings in some interval of values close to zero give piecewise disconnected edges at best.

- Many other approaches improving zero-crossing performance can be found in the literature.

- The traditional second-derivative zero-crossing has disadvantages as well:
  - it smooths the shape too much; for example, sharp corners are lost.
  - it tends to create closed loops of edges (nicknamed the 'plate of spaghetti' effect).

- Neurophysiological experiments provide evidence that the human retina operation on image can be described analytically as the convolution of the image with the ΔG operator.

Example 4.3.19. *Marr-Hildreth edge detector. The Khoros workspace for this example is here Marr-Hildreth edge detector.*

4.3.4 Scale in image processing

- Many image processing techniques work locally, theoretically at the level of individual pixels — edge detection methods are examples.
- The essential problem in such computation is **scale**.
- Edges correspond to the gradient of the image function that is computed as a difference among pixels in some neighborhood.
- There is seldom a sound reason for choosing a particular size of neighborhood:
  - The ‘right’ size depends on the size of the objects under investigation.
  - To know what the objects are assumes that it is clear how to interpret an image and this is not in general known at the pre-processing stage.

- The phenomenon under investigation is expressed at different resolutions of the description, and a formal model is created at each resolution.
- Then the qualitative behavior of the model is studied under changing resolution of the description.
- Such a methodology enables the deduction of meta-knowledge about the phenomenon that is not seen at the individual description.

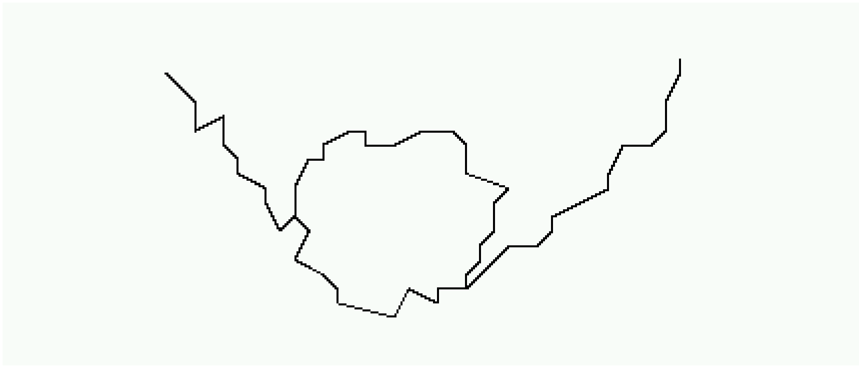
- Different description levels are easily interpreted as different scales in the domain of digital images.
- The idea of scale is fundamental to Marr's edge detection technique, introduced in the last sub-section, where different scales are provided by different sizes of Gaussian filter masks.
- The aim was not only to eliminate fine scale noise but also to separate events at different scales arising from distinct physical processes, [Marr, 1982].

- Assume that a signal has been smoothed with several masks of variable sizes.
- Every setting of the scale parameters implies a different description, but it is not known which one is correct.
- For many tasks, no one scale is categorically correct.
- If the ambiguity introduced by the scale is inescapable, the goal of scale-independent description is to reduce this ambiguity as much as possible.

- Many publications tackle scale-space problems. Here we shall consider just three examples of the application of multiple scale description to image analysis.
- There are other approaches involving non-linear partial differential equations to generate non-linear scale-space descriptions.

- The first approach aimed to process planar noisy curves at a range of scales — the segment of curve that represents the underlying structure of the scene needs to be found.

- The problem is illustrated in by an example of two noisy curves in the following figure.



- One of these may be interpreted as a closed (perhaps circular) curve, while the other could be described as two intersecting straight lines.

- Local tangent direction and curvature of the curve are significant only with some scales after the curve is smoothed by Gaussian filter with varying standard deviations.

- After smoothing using the Gaussian filter with varying standard deviations, the significant segments of the original curve can be found.

- The second approach, called **scale space filtering**, tried to describe signals qualitatively with respect to scale.
- The problem was formulated for 1D signals $f(x)$, but it could easily be generalized to 2D functions as images.
- The original 1D signal f(x) is smoothed by convolution with a 1D Gaussian

$$f(x, \sigma) = G(x, \sigma) * f(x) \tag{4.99}$$

- If the standard deviation σ is slowly changed the function $f(x, \sigma)$ represents a surface on the (x, σ) plane that is called the scale-space image.
- Inflection points of the curve $f(x, \sigma_0)$ for a distinct value σ_0

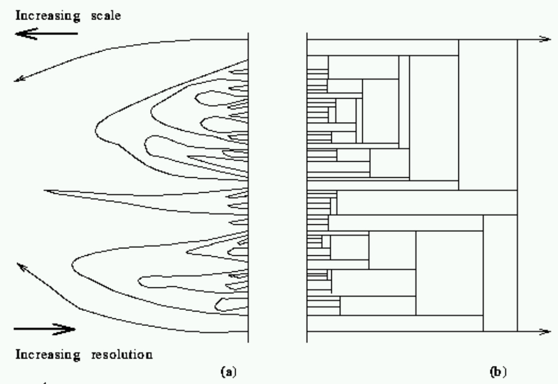
$$\frac{\partial^2 f(x, \sigma_0)}{\partial x^2} = 0, \qquad \frac{\partial^3 f(x, \sigma_0)}{\partial x^3} \neq 0. \tag{4.100}$$

describe the curve $f(x)$ qualitatively.

- The positions of inflection points can be drawn as a set of curves

$$\Sigma(x, \sigma_0) \tag{4.101}$$

in (x, σ) co-ordinates



Scale-space image:

(a) Varying number and locations of curve segmentation points as a function of scale,

(b) curve representation by an interval tree.

- Coarse to fine analysis of the curves corresponding to inflection points, i.e., in the direction of the decreasing value of the σ, localizes events at different scales.
- The qualitative information contained in the scale-space image can be transformed into a simple **interval tree** that expresses the structure of the signal $f(x)$ over all (observed) scales.
- The interval tree is built from the root that corresponds to the largest scale.
- Then the scale-space image is searched in the direction of decreasing σ.
- The interval tree branches at those points where new curves corresponding inflection points appears.

- The third example of the application of scale — Canny edge detector, discussed in the next sub-section.

4.3.5 Canny edge detection

- Canny edge detector is optimal for step edges corrupted by white noise.
- The optimality of it is related to three criteria:
  - The **detection** criterion expresses that fact that important edges should not be missed, and that there should be no spurious responses.
  - The **localization** criterion says that the distance between the actual and located position of the edge should be minimal.
  - The **one response** criterion minimizes multiple responses to a single edge (also partly covered by the first criterion, since when there are two responses to a single edge one of them should be considered as false).

- Canny's edge detector is based on several ideas:

  1. The edge detector was expressed for a 1D signal and the first two optimality criteria. A closed form solution was found using the calculus of variations.
  2. If the third criterion (multiple responses) is added, the best solution may be found by numerical optimization. The resulting filter can be approximated effectively by the first derivative of a Gaussian smoothing filter with standard deviation σ; the reason for doing this is the existence of an effective implementation.
     - There is a strong similarity here to the Marr-Hildreth edge detector (Laplacian of a Gaussian)
  3. The detector is then generalized to two dimension. A step edge is given by its position, orientation, and possibly magnitude (strength).
     - It can be shown that convolving an image with a symmetric 2D Gaussian and then differentiating in the direction of the gradient (perpendicular to the edge direction) forms a simple and effective directional operator.
     - Recall that the Marr-Hildreth zero crossing operator does not give information about edge direction as it uses Laplacian filter.

- Suppose G is a 2D Gaussian (4.95) and assume we wish to convolute the image with an operator $G_{\mathbf{n}}$ which is a first derivative of G in the direction $\mathbf{n}$.

$$G_{\mathbf{n}} = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G \tag{4.102}$$

- The direction $\mathbf{n}$ should be oriented perpendicular to the edge
  - this direction is not known in advance
  - however, a robust estimate of it based on the smoothed gradient direction is available
  - if g is the image, the normal to the edge is estimated as

$$\mathbf{n} = \frac{\nabla(G * g)}{|\nabla(G * g)|} \tag{4.103}$$

- The edge location is then at the local maximum in the direction $\mathbf{n}$ of the operator $G_{\mathbf{n}}$ convoluted with the image g

$$\frac{\partial}{\partial \mathbf{n}} G_{\mathbf{n}} * g = 0. \tag{4.104}$$

- Substituting in equation (4.104) for $G_{\mathbf{n}}$ from equation (4.102), we get

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * g = 0. \tag{4.105}$$

- This equation (4.105) shows how to find local maxima in the direction perpendicular to the edge; this operation is often referred to as **non-maximal suppression**.
- As the convolution and derivative are associative operations in equation (4.105)
  - ∗ first convolute an image g with a symmetric Gaussian G;
  - ∗ then compute the directional second derivative using an estimate of the direction **n** computed according to equation (4.103);
  - ∗ strength of the edge (magnitude of the gradient of the image intensity function g) is measured as

$$|G_{\mathbf{n}} * g| = |\nabla(G * g)|. \tag{4.106}$$

4. Spurious responses to the single edge caused by noise usually create a so called 'streaking' problem that is very common in edge detection in general.
   - Output of an edge detector is usually thresholded to decide which edges are significant.
   - Streaking means breaking up of the edge contour caused by the operator fluctuating above and below the threshold.

- Streaking can be eliminated by **thresholding with hysteresis**.
  - If any edge response is above a high threshold, those pixels constitute definite output of the edge detector for a particular scale.
  - Individual weak responses usually correspond to noise, but if these points are connected to any of the pixels with strong responses they are more likely to be actual edges in the image.
  - Such connected pixels are treated as edge pixels if their response is above a low threshold.
  - The low and high thresholds are set according to an estimated signal to noise ratio. Please refer to Canny's original paper for detailed discussions. Canny discussed the one dimensional case in §VI in his paper.

5. The correct scale for the operator depends on the objects contained in the image.
   - The solution to this unknown is to use multiple scales and aggregate information from them.
   - Different scale for the Canny detector is represented by different standard deviations σ of the Gaussians.
   - There may be several scales of operators that give significant re-

sponses to edges (i.e., signal to noise ratio above the threshold); in this case the operator with the smallest scale is chosen as it gives the best localization of the edge.

- Canny proposed a **Feature synthesis** approach.
- All significant edges from the operator with the smallest scale are marked first.
- Edges of a hypothetical operator with larger σ are synthesized from them (i.e., a prediction is made of how the larger σ should perform on the evidence gleaned from the smaller σ).
- Then the synthesized edge response is compared with the actual edge response for larger σ.
- Additional edges are marked only if they have significantly stronger response than that predicted from synthetic output.
- This procedure may be repeated for a sequence of scales, a cumulative edge map is built by adding those edges that were not identified at smaller scales.

Algorithm 4.3.20. *Canny edge detector*

1. *Repeat steps (2) till (6) for ascending values of the standard deviation.*
2. *Convolve an image g with a Gaussian of scale σ.*
3. *Estimate local edge normal directions* $\mathbf{n}$ *using equation (4.103) for each pixel in the image.*
4. *Find the location of the edges using equation (4.105) (non-maximal suppression).*
5. *Compute the magnitude of the edge using equation (4.106).*
6. *Threshold edges in the image with hysteresis to eliminate spurious responses.*
7. *Aggregate the final information about edges at multiple scale using the "feature synthesis" approach.*

- Canny's detector represents a complicated but major contribution to edge detection.
- Its full implementation is unusual, it being common to find implementations that omit feature synthesis — that is, just steps 2 — 6 of algorithm.
- Reference for this section is Canny's paper [Canny, 1986].

Example 4.3.21. *Canny edge detector. The example matlab script file is here Canny edge detector example.*

(You need to start matlab.)

Compare the result with the edge detector with other filters. Note the improvement at the top part of the hat.

Study the matlab code. Note the "thin" code.

Homework 4.3.22. *This is an important homework. The total score is 20. The homework includes:*

1. *(4) Implement median filter with variable window size and shape;*
2. *(4) Implement another local smoothing filter you know with possible variable parameters;*
3. *(8) Implement Canny edge detector; what is your extension to step 7? (Without step 7, you can only get at most 4 scores).*
4. *(4) Implement another edge detector you know; compare it with Canny detector.*

4.3.6 Parametric edge models

- Parametric models are based on the idea that the discrete image intensity function can be considered a sampled and noisy approximation of the underlying continuous or piecewise continuous image intensity function.

- While the continuous image function is not known, it can be estimated from the available discrete image intensity function and image properties can be determined from this estimate, possibly with sub-pixel precision.

- Piecewise continuous function estimate called `facets` are used to represent (a neighborhood) image pixel.

- Such image representation is called `facet model`.

- The intensity function in a pixel neighborhood can be estimated using models of different complexity.
- The simplest one is the `flat facet model` that uses piecewise constants and each pixel neighborhood is represented by a flat function of constant intensity.
- The sloped model uses piecewise linear functions forming a sloped plane fitted to the image intensities in the pixel neighborhood.
- Quadratic and bi-cubic facet models employ correspondingly more complex functions.
- A thorough treatment of facet models and their modifications for peak noise removal, segmentation into constant-gray-level regions, determination of statistically significant edges, gradient edge detection, directional second-derivative zero-crossing detection, and line and corner detection is given in [Haralic and Shapiro,

- An edge detection example is given in the following.
- Consider a bi-cubic facet model

  $$g(x, y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2 + c_7x^3 + c_8x^2y + c_9xy^2 + c_{10}y^3 \tag{4.107}$$

  The parameter of which are estimated from a pixel neighborhood(the co-ordinate of the central pixel is $(0, 0)$).
- To determine the model parameters, a least-squares method with singular-value decomposition may be used.
- Once the facet model parameters are available for each image pixel, edges can be detected as extrema of the first directional derivative and/or zero-crossings of the the second directional derivative of the local continuous facet model functions.
- Edge detectors based on parametric models describe edges more precisely than convolution based edge detectors.
- Additionally, they carry the potential for sub-pixel edge localization (?).
- However, their computational requirements are much higher.

4.3.7 Edges in multi-spectral images

- There are several possibilities for the detection of edges in multi-spectral images.
- The first is to detect edges separately in individual image spectral components using the ordinary local gradient operators mentioned in Section (4.3.2).
  - Individual images of edges can be combined to get the resulting edge image, with the value corresponding to edge magnitude and direction being the maximal edge value from all spectral components.
  - A linear combination of edge spectral components can also be used, and other combination techniques are possible.
- A second possibility is to use the brightness difference of the same pixel in two different spectral components.
  - The ratio instead of the difference can also be used as well, although it is necessary to assume that pixel values are not zero in this case.
- A third possibility is to create a multi-spectral edge detector which uses brightness information from all n spectral bands.

- An edge detector of this kind was proposed in [Cervenka and Charvat, 1987].
- The neighborhood used has size $2 \times 2 \times n$ pixels, where the 2×2 neighborhood is similar to that of the Roberts gradient, (4.80).
- The coefficients weighting the influence of the component pixels are similar to the correlation correlation coefficients.
- Let $\bar{f}(i,j)$ denote the arithmetic mean of the brightness corresponding to the pixel with the same co-ordinates (i,j) in all n spectral component images and f_r be the brightness of the r^{th} spectral component.
- The edge detector result in pixel (i,j) is given as the minimum of the following expressions:

$$\frac{\sum_{r=1}^{n}\left[f_r(i,j)-\bar{f}(i,j)\right]\left[f_r(i+1,j+1)-\bar{f}(i+1,j+1)\right]}{\sqrt{\sum_{r=1}^{n}\left[f_r(i,j)-\bar{f}(i,j)\right]^2\sum_{r=1}^{n}\left[f_r(i+1,j+1)-\bar{f}(i+1,j+1)\right]^2}}$$

$$\cdot\frac{\sum_{r=1}^{n}\left[f_r(i+1,j)-\bar{f}(i+1,j)\right]\left[f_r(i,j+1)-\bar{f}(i,j+1)\right]}{\sqrt{\sum_{r=1}^{n}\left[f_r(i+1,j)-\bar{f}(i+1,j)\right]^2\sum_{r=1}^{n}\left[f_r(i,j+1)-\bar{f}(i,j+1)\right]^2}}$$

- This multi-spectral edge detector gives very good (?) results on remotely sensed images.

4.3.8 Line detection by local pre-processing operators

- Several other local operations exists which do not belong to the taxonomy in this Section (4.3), as they are used for different purposes.
- `Line finding`, `line thinning`, and `line filling` operators are among them.
- The second group of operators finds `interest points` or `locations of interest` in the image.
- There is yet another class of local nonlinear operators, `mathematical morphology techniques`.

- Recall that one of the reasons why edges are being detected is that they bear a lot of information about underlying objects in the scene.
- Taking just edge elements instead of all pixels reduces the amount of data which has to be processed.
- The edge detector is a general tool which does not depend on the content of the particular image.
- The detected edges are to some degree robust as they do not depend much on small changes in illumination, viewpoint change, etc.

- It is interesting to seek yet richer features which can be reliably detected in the image and which can outperform simple edge detectors in some classes of applications.
- Line detectors and corner detectors are some such.

Line detection

- `Line finding operators` aim to find very thin curves in the image, e.g., roads in satellite images.
- It is assumed that curves do not bend sharply. Such curves and straight lines are called **lines** for the purpose of describing this technique.
- Lines are modeled by a roof profile among edges, Fig. 4.3.2.
- We assume that the width of the lines is approximately one or two pixels.
- Lines in the image can be detected by a number of local convolution operators h_k, which serve as line patterns [Cervenka and Charvat, 1987].
- The output value of the line finding detector in pixel (i,j) is given by

$$f(i,j) = \max\left\{0, \max_k (f * h_k)\right\} \tag{4.108}$$

where $f * h_k$ denotes the convolution of the k-th mask with the neighborhood of a pixel (i,j) in the input image.

- The simplest collection of four such patterns of size 3 x 3 is able to detect lines rotated modulo the angle 45^o:

$$h_1 = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}, \quad h_3 = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

- A similar principle can be applied to bigger masks.
- The following is a set of convolution masks of size 5×5. There are 14 possible orientation of the line finding convolution mask of this size; only the first eight are shown, as the others are obvious by rotation.
- Such line detectors sometimes produce more lines than needed, and other non-linear constraints may be added to reduce this number.

$$h_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$h_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad h_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Line thinning

- In the edge detection methods introduced so far, edges are usually found by simple thresholding of the edge magnitude.
- Such edge thresholding does not provide ideal contiguous boundaries that are one pixel wide.
- Sophisticated segmentation techniques are discussed in the next chapter.
- Here, much simpler edge thinning and filling methods are described.
- These techniques are based on knowledge of small local neighborhoods and are very similar to other local pre-processing techniques.

- One line thinning method uses knowledge about orientation and in this case edges are thinned before thresholding.
- Edge magnitude and directions provided by some gradient operator are used as input, and the edge magnitude of two neighboring pixels perpendicular to the edge direction are examined for each pixel in the image.
- If at least one of these pixels has edge magnitude higher than the edge magnitude of the examined pixel, then the edge magnitude of the examined pixel is assigned a zero value.
- The technique is called **non-maximal suppression** and is similar to the idea mentioned in conjunction with the Canny edge detector.

STOP

- There are many line thinning methods which we do not present here.
- In most cases the best line thinning is achieved using mathematical morphology methods (?).

Edge filling

- Edge points after thresholding do not create contiguous boundaries and the edge filling method tries to recover edge pixels on the potential object boundary which are missing.
- We present here a very simple local edge filling technique; more complicated methods based on edge relaxation are discussed in the next chapter.
- The local edge filling procedure checks the 3×3 neighborhood of the current pixel matches one of the following masks If so, the central pixel of the mask

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

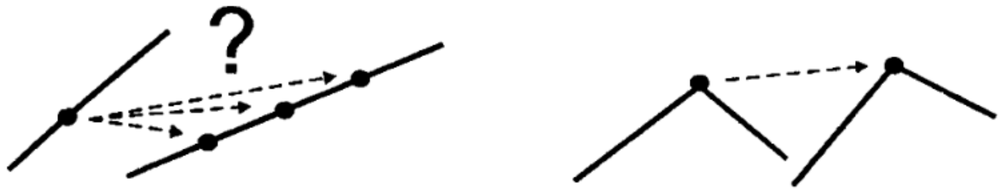
is changed from zero to one.

- These simple methods for edge thinning and filling do not guarantee that the width of the lines will be equal to one and the contiguity of the edges are is not certain either.
- Note that local thinning and filling operators can be treated as special cases of mathematical morphology operators.

4.3.9 Corners and interesting points

- In many cases it is of advantage to find pairs of `corresponding points` in two similar images.
- E.g., when finding geometric transforms, knowing the position of corresponding points enables the estimation of geometric transforms from live data.
- Finding corresponding points is also a core problem in the analysis of moving images and fir recovering depth information form pairs of stereo images.
- In general, all possible pairs of points should be examined to solve this `correspondence problem`, and this is very computation expensive.
- If two images have n pixels each, the complexity is $O(n^2)$.
- This process might be simplified if the correspondence is examined among a much smaller number of points, called `interest points`.
- An interest point should have some typical local property.
- E.g., if square objects are present in the image, then `corners` are very good interest points.

- Corners serve better than lines when the correspondence problem is to be solved.
- This is due to the `aperture problem`.
- Assume a moving line is seen through a small aperture. In such a case, only the motion vector perpendicular to the line can be observed.
- The component collinear with the line remains invisible.
- The situation is better with corners. They provide ground for unique matching, cf. the following figure for illustration.



- Edge detectors themselves are not stable at corners.
- This is natural as the gradient at the tip of the corner is ambiguous.
- Near the corner there is a discontinuity in the gradient direction.
- This observation is used in corner detectors.

- Corners in image can be located using local detectors.
- Input to the corner detector is the gray-level image and output is the image $f(i,j)$ in which values are proportional to the likelihood that the pixel is a corner.
- Interest points are obtained by thresholding the result of the corner detector.

- The corner in the image can be defined as a pixel in whose small neighborhood there are two dominant and different edge directions.

- This definition is not precise as an isolated point of local intensity maximum or minimum, line endings, or an abrupt change in the curvature of a curve with a response similar to a corner.

- Nevertheless, such detectors are named corner detectors in the literature and are widely used.

- If corners have to be detected then some additional constraints have to be applied.

- Corner detectors are not usually very robust.

- This deficiency is overcome either by manual expert supervision or large redundancies introduced to prevent the effect of individual errors from dominating the task.

- The latter means that many more corners are detected in two or more images than necessary for estimating a transformation sought between these images.

- The simplest corner detector is the **Moravec detector** which is maximal in pixels with high contrast.
- These points are on corners and sharp edges.
- The Moravec operator MO is given by

$$\mathrm{MO}(i,j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} |g(k,l) - g(i,j)|. \tag{4.109}$$

- Better results are produced by computationally more expensive corner operators based on the facet model, (4.107).
- The image function g is approximated in the neighborhood of the pixel (i,j) by a bi-cubic polynomial with coefficients c_k as in (4.107).

$$g(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3 \tag{4.110}$$

- The Zuniga-Haralick operator ZH is given by

$$\mathrm{ZH}(i,j) = \frac{-\left(c_2^2 c_6 - 2c_2 c_3 c_5 + c_3^2 c_4\right)}{\left(c_2^2 + c_3^2\right)^{\frac{3}{2}}} \tag{4.111}$$

  which is the curvature of the plane curve $g(x, y) = \text{const}$.

- The Kitchen-Rosenfeld KR is given by

$$\mathrm{KR}(i,j) = \frac{-\left(c_2^2 c_6 - 2c_2 c_3 c_5 + c_3^2 c_4\right)}{c_2^2 + c_3^2} \tag{4.112}$$

  which is the second order derivative along the direction of edge.

- The ZH operator has been shown to outperform the KR detector in test images.

- The Harris corner detector improved upon Moravec's by considering the differential of the corner score (sum of square differences).

- Consider a 2D gray-scale image f.

- An image patch W in f is taken and is shifted by Δx and Δy.
- The sum of square differences S between values of the image f given by the patch W and its shifted variant by Δx and Δy is given by:

$$S_W(\Delta x, \Delta y) = \sum_{(x_i,y_i)\in W} \left(f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y)\right)^2 . \tag{4.113}$$

- A corner point not suffering from the aperture problem must have a high response of $S_W(\Delta x, \Delta y)$ for all Δx and Δy.

- If the shifted image patch is approximated by the first-order Taylor expansion

$$f(x_i - \Delta x, y_i - \Delta y) = f(x_i, y_i) - \frac{\partial f}{\partial x}(x_i, y_i)\Delta x - \frac{\partial f}{\partial y}(x_i, y_i)\Delta y, \tag{4.114}$$

then the minimum of $S_W(\Delta x, \Delta y)$ can be obtained analytically.

- Substituting (4.114) into (4.113),

$$S_W(\Delta x, \Delta y) = \sum_{(x_i,y_i)\in W} \left(\frac{\partial f}{\partial x}(x_i, y_i)\Delta x + \frac{\partial f}{\partial y}(x_i, y_i)\Delta y, \right)^2$$

$$= \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} A_W(x,y) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

where the Harris matrix

$$A_W(x,y) = \sum_{(x_i,y_i)\in W} \begin{pmatrix} \left(\frac{\partial f}{\partial x}(x_i,y_i)\right)^2 & \frac{\partial f}{\partial x}(x_i,y_i)\frac{\partial f}{\partial y}(x_i,y_i) \\ \frac{\partial f}{\partial x}(x_i,y_i)\frac{\partial f}{\partial y}(x_i,y_i) & \left(\frac{\partial f}{\partial y}(x_i,y_i)\right)^2 \end{pmatrix} . \qquad (4.115)$$

- Usually an isotropic window W is used, such as a Gaussian.
- The response will be isotropic too.

- The local structure matrix A_W represents the neighborhood — the Harris matrix A_W is symmetric and positive semi-definite.
- Its main modes of variation correspond to partial derivatives in orthogonal directions and are reflected in eigenvalues λ_1 and λ_2 of the matrix A_W.

- Three distinct cases can appear:

1. Both eigenvalues are small. This means that image f is flat in the examined pixel. There are no edges or corners in this location.
2. One eigenvalue is small and the second one large. The local neighborhood is ridge-shaped. Significant change of image f occurs if a small movement is made perpendicularly to the ridge.
3. Both eigenvalues are rather large. A small shift in any direction causes significant change of the image f. A corner is found.

- Harris suggested that exact eigenvalue computation can be avoided by calculating the response function $R(A) = \det(A) - \kappa \operatorname{trace}^2(A)$, where κ is a tunable parameter where values from 0.04 to 0.15 were reported in literature as appropriate.

Algorithm 4.3.23. *Harris corner detector*

1. *Filter the image with a Gaussian.*
2. *Estimate intensity gradient in two perpendicular directions for each pixel. This is performed by twice using a ID convolution with the kernel approximating the derivative.*

3. *For each pixel and a given neighborhood window:*
    - *Calculate the local structure matrix A;*
    - *Evaluate the response function $R(A)$.*
4. *Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.*

- The Harris corner detector has been very popular.
- Its advantages are insensitivity to 2D shift and rotation, to small illumination variations, to small viewpoint change, and its low computational requirements.
- On the other hand, it is not invariant to larger scale change, viewpoint changes and significant changes in contrast.
- Many more corner-like detectors exist, and the reader is referred to the overview papers.

Example 4.3.24. *Corner detection example with Harris corner detector. The matlab script from* **visionbook** *is is here Corner Detection Example.*

4.3.10 Adaptive neighborhood pre-processing

- The majority of pre-processing operators work in neighborhoods of fixed sizes in the whole image, of which square windows (3×3, 5×5, or 7×7) are most common.
- Pre-processing operators of variable sizes and shapes exist and bring improved pre-processing results.
- They are based on detection of the most homogeneous neighborhood of each pixel.
- However they are not widely used, mostly because of computational demands and the non-existence of a unifying approach.

- A recent approach to image pre-processing introduces the concept of an adaptive neighborhood which is determined for each image pixel.
- The neighborhood size and shape are dependent on characteristics of image data and on parameters which define measures of homogeneity of a pixel neighborhood.
- A significant property of the neighborhood for each pixel is the ability to self tune to contextual details in the image.

Neighborhood

- An adaptive neighborhood is constructed for each pixel, this pixel being called a **seed pixel** of the neighborhood.
- The adaptive neighborhood consists of all the 8-connected ((2.3.1)) pixels which satisfy a property of similarity with the seed pixel.
- The pixel property may represent a gray-level, or some more complex image properties such as texture, local motion parameters, etc.
- Let's consider gray-level as a basic pixel property — the adaptive neighborhood for gray-level image pre-processing is based on as additive or multiplicative tolerance interval.

- All the pixels which are 8-connected with the seed pixel and which have their gray-levels in a tolerance interval become members of the adaptive neighborhood.

- Let (i,j) represent the seed pixel, and (k,l) represent pixels 8-connected to the seed pixel.

- The adaptive neighborhood of the pixel (i,j) is defined as a set of pixels (k,l) 8-connected to the seed pixel and tither satisfying the additive tolerance property
$$|f(k,l) - f(i,j)| \leq T_1 \tag{4.116}$$
or satisfying a multiplicative property
$$\frac{|f(k,l) - f(i,j)|}{f(i,j)} \leq T_2 \tag{4.117}$$
where T_1 and T_2 are parameters of the adaptive neighborhood and represent the maximum allowed dissimilarity of a neighborhood pixel from the seed pixel.

- The above procedure defines the first layer of the adaptive neighborhood (called th foreground layer) which is used in all adaptive neighborhood pre-processing

- Sometimes not only the foreground layer but also a background layer must be used to represent more diverse contextual information. The second (background) layer is molded to the outline of the first layer and has a thickness of s pixels, s being a parameter for the adaptive neighborhood.

- Although a neighborhood is constructed for each pixel, all pixels in the given foreground layer that have the same gray-level as the seed pixel produce the same adaptive neighborhood. These pixels are called **redundant seed pixels**.

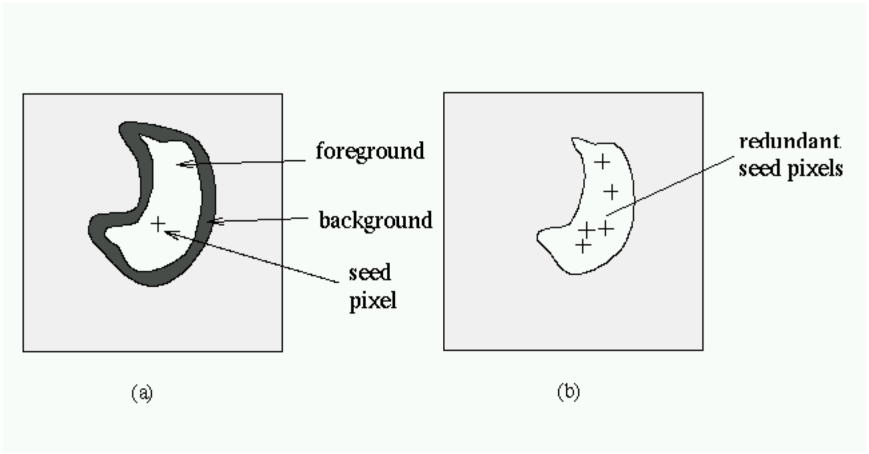
- Many fixed-neighborhood pre-processing may be implemented by applying the adaptive neighborhood concept.

Noise suppression

- An adaptive neighborhood is not formed across region boundaries.

- Therefore, noise suppression will not blur image edges as often happens with other techniques.

- If noise suppression is the goal, only the foreground neighborhood layer is used.


foreground
background
seed pixel
redundant seed pixels
(a)
(b)

- Having constructed the adaptive neighborhood for each pixel, the rest is straightforward: Each seed pixel is assigned a new value computed as a mean, median, etc., of all the pixels in the adaptive neighborhood.

- If the noise is additive, the additive criterion for neighborhood construction should be applied; if the noise is multiplicative, the multiplicative tolerance criterion is appropriate.

- Adaptive neighborhood noise suppression may be applied several times in a sequence with good results and the edges will not be blurred.

- Adaptive neighborhood smoothing does not work well for impulse noise because large gray-level difference between the noise pixel and other pixels in the neighborhood cause the adaptive neighborhood to consist of only the noise pixel.

- A solution may be to apply a fixed-size averaging or median filtering preprocessing step prior to the adaptive neighborhood operations.

- Show Fig. 4.25.

Histogram modification

- The main disadvantage of full-frame histogram equalization is that the global image properties may not be appropriate under a local context.
- Local area histogram equalization computes a new gray-level for each pixel based on the equalization of a histogram acquired in a local fixed-size neighborhood.
- Adaptive neighborhood histogram modification is based on the same principle — the local histogram is computed from a neighborhood which reflects local contextual image properties.
- Show Fig. 4.26.

Contrast enhancement

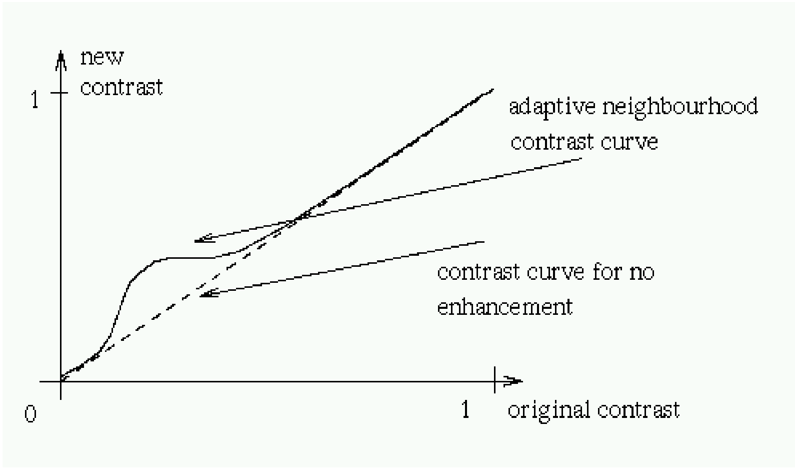
- Contrast is a property based on human abilities.
- An approximate definition of contrast is

$$c = \frac{F - B}{F + B} \tag{4.118}$$

where F and B are the mean gray-levels of two regions whose contrast is evaluated.

- Standard contrast enhancement techniques such as sharpening (4.3.2) do not enhance the contrast of regions, only local edge perception.
- The larger the contrast between image parts, the larger is the enhancement.
- In other words, the most serious enhancement is achieved where the contrast is sufficient anyway.
- The adaptive neighborhood is associated with objects and therefore it is feasible to enhance contrast in regions by modifying gray-levels in regions and not only along their borders.
- Further, the contrast enhancement may be non-linear:
  - No enhancement for very small gray-level difference between neighborhoods (caused probably by quantization noise or very small gray-level variance);
  - Moderate to strong enhancement applied if the contrast between regions is small but outside the range of quantization contrast;

– No contrast enhancement is applied if the contrast is already sufficient.



- For contrast enhancement, both foreground and background adaptive neighborhood layers are used, the background size being comparable in size to the foreground size.
- For each seed pixel and corresponding adaptive neighborhood, the original

contrast c is computed using (4.118).

- The new desired contrast c' is obtained from the applied contrast curve (4.3.10).
- The new gray value $f'(i,j)$ to be assigned to the seed pixel (i,j) is computed as

$$f'(i,j) = B\frac{1+c'}{1-c'} \tag{4.119}$$

  where B is the original mean gray-level of the background adaptive neighborhood layer.
- The contrast between the seed pixel and the background layer is c'.
- Show Fig. 4.28.

Remark 4.3.25.

- *The principle of adaptive neighborhood pre-processing gives significantly better results, but larger computational load is the price to pay for this improvement.*

- *Nevertheless, taking advantage of redundant seed pixels decreases the computational demands, also, feasibility of implementing these methods in parallel may soon make these methods as standard as fixed neighborhood methods are today.*

4.4 Image restoration

- Pre-processing methods that aim to suppressing image degradation using knowledge about its nature are called **image restoration**.
- Most image restoration methods are based on convolution applied globally to the whole image.

- Degradation of images can have many causes
  - defects of optical lenses;
  - nonlinearity of the electro-optical sensor;
  - graininess of the film material;
  - relative motion between an object and camera
  - wrong focus,
  - atmospheric turbulence in remote sensing or astronomy,
  - etc.
- The objective of image restoration is to reconstruct the original image from its degraded version.

- Image restoration techniques can be classified into two groups:
  - **Deterministic methods** are applicable to images with little noise and a known degradation function.
  - The original image is obtained from the degraded one by a transformation inverse to the degradation.
  - **Stochastic techniques** try to find the best restoration according to particular stochastic criterion, e.g., a least squares method.
  - In some cases the degradation transformation must be estimated first.

- It is advantageous to know the degradation function explicitly.
- The better this knowledge is, the better are the results of the restoration.
- There are three typical degradations with a simple function:
  - Relative constant speed movement of the object with respect to the camera,
  - wrong lens focus,
  - and atmospheric turbulence.

- In most practical cases, there is insufficient knowledge about the degradation, and it must be estimated and modeled.
- The estimation can be classified into two groups according to the information available **a priori** and a posteriori.
- If degradation type and/or parameters need to be estimated, this step is the most crucial one, being responsible for image restoration success or failure.
- It is also the most difficult part of image restoration.

- **A priori knowledge**
  - A priori knowledge about degradation is either known in advance or can be obtained before restoration.
  - E.g., if it is clear in advance that the image was degraded by relative motion of an object with respect to the sensor then the modeling only involves the speed and direction of the motion.
  - E.g., parameters of a capturing device such as a TV camera or digitizer, whose degradation remains unchanged over a period of time and can be modeled by studying a known sample image and its degraded version.
- **A posteriori knowledge** is obtained by analyzing the degraded image.
  - A posteriori knowledge is obtained by analyzing the degraded image.
  - A typical example is to find some interest points in the image (e.g. corners, straight lines) and guess how they looked before degradation.
  - Another possibility is to use spectral characteristics of the regions in the image that are relatively homogeneous.

- Only the basics of the restoration and three typical degradations are considered here.

- A degraded image g can arise from the original image f by a process which can be expressed as

$$g(i,j) = s\left(\int_{(a,b)\in\mathcal{O}} f(a,b)h(a,b,i,j)\,da\,db\right) + \nu(i,j) \tag{4.120}$$

  where s is some nonlinear function and ν describes the noise.

- The degradation is very often simplified by
  - neglecting the nonlinearity;
  - assuming that the function h is invariant with respect to position in the image.

- Degradation can be then expressed as convolution

$$g(i,j) = (f * h)(i,j) + \nu(i,j) \tag{4.121}$$

- If the degradation is given by equation (4.121) and the noise is not significant then image restoration equates to inverse convolution (also called deconvolution).

- If noise is not negligible then the inverse convolution is solved as an undetermined system of linear equations.

- Methods based on minimization of the least square error such as Wiener filtering or Kalman filtering are examples.

4.4.1 Degradation that are easy to restore

- Some degradations can be easily expressed mathematically (convolution) and also restored simply in images.
- These degradation can be expressed by convolution, equation (4.121).
- Let F, G and H be the Fourier transform of the undegraded image f, the degraded image g and the convolution kernel h, respectively.
- In the absence of noise, the relationship is

$$G = HF \tag{4.122}$$

- Therefore, not considering image noise ν, knowledge of the degradation function fully determines image restoration by inverse filtration, (4.4.2), discussed in the next section.
- We first discuss several degradation functions.

Relative motion of the camera and the object

- Assume an image is acquired with a camera with a mechanical shutter.
- Relative motion of the camera and the photographed object during the shutter open time T causes smoothing of the object in the image.
- Suppose V is the constant speed in the direction of the x axis; the Fourier transform $H(u, v)$ of the degradation caused in time T is given by

$$H(u, v) = \frac{\sin(\pi VTu)}{\pi Vu} \tag{4.123}$$

Wrong lens focus

- Image smoothing caused by imperfect focus of a thin lens can be described by the following function

$$H(u, v) = \frac{J_1(ar)}{ar} \tag{4.124}$$

where J_1 is the Bessel function of the first order, $r^2 = u^2 + v^2$, and a is the displacement.

Atmospheric turbulence

- Atmospheric turbulence is degradation that needs to be restored in remote sensing and astronomy.
- It is caused by temperature non-homogeneity in the atmosphere that deviates passing light rays.
- The mathematical model

$$H(u,v) = \mathbf{e}^{-c\left(u^2+v^2\right)^{\frac{5}{6}}} \tag{4.125}$$

  where c is a constant that depends on the type of turbulence which is usually found experimentally.
- The power $5/6$ is sometimes replaced by 1.

4.4.2 Inverse filtration

- An obvious approach to image restoration is based on the properties of the Fourier transform.
- Inverse filtration uses the assumption that degradation was caused by a linear convolution kernel $h(i,j)$.

- the additive noise ν is another source of degradation.
- It is further assumed that ν is independent of the signal.
- Applying the Fourier transform to equation (4.121)

$$G(u, v) = H(u, v)F(u, v) + N(u, v) \tag{4.126}$$

- The degradation can be eliminated if the restoration filter has a transfer function that is inverse to the degradation h.
- The Fourier transform of the inverse filter is then expressed as $H^{-1}(u, v)$.
- The undegraded image F is derived from its degraded version G.

$$F(u, v) = G(u, v)H^{-1}(u, v) - N(u, v)H^{-1}(u, v) \tag{4.127}$$

- This equation shows that inverse filtration works well for images that are not corrupted by noise, not considering possible computational problems if $H(u, v)$ gets close to zero at some location of the u, v space — fortunately, such locations can be neglected without perceivable effect on the restoration result.

- If noise is present, some problems arise:
  - its influence is significant for frequencies where $H(u, v)$ has small magnitude.
  - These usually correspond to high frequencies u, v and thus fine details are blurred in the image.
  - In reality, $H(u, v)$ usually decreases in magnitude much more rapidly than $N(u, v)$ and thus the noise effect may dominate the entire restoration result.
  - Limiting the restoration to a small neighborhood of the u, v origin in which $H(u, v)$ is sufficiently large overcomes this problem and the results usually quite acceptable.
  - The second problem deals with the spectrum of the noise itself — we usually do not have enough information about the noise to determine $N(u, v)$ sufficiently well.

4.4.3 Wiener filtration

Discrete Fourier transform, convolution and correlation

- Let $f(m,n)$ be a $M \times N$ two dimensional periodic discrete signals, the discrete Fourier transform of it is defined as

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \mathbf{e}^{-j2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)} \tag{4.128}$$

  for $u = 0, \cdots, M-1$ and $v = 0, \cdots, N-1$.

- The Fourier transform is still a periodic signal of the same periodicity.

- The inverse transform is defined as

$$f(m,n) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \mathbf{e}^{j2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)} \tag{4.129}$$

- If f and g are periodic signals of the same periodicity, the circulant convolution of f and g is defined as

$$f \odot g(r,s) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) g(r-m, s-n) \tag{4.130}$$

which is still a periodic signal of the same periodicity.

- Note that the circulant convolution is different from the ordinary convolution when f and g are treated as two discrete non-periodic signals.
- Recall the (ordinary) convolution is defined for two discrete signals f and g of $M \times N$ and and $A \times B$ samples, respectively, as

$$f * g(r,s) = \sum_{m=0}^{r} \sum_{n=0}^{s} f(m,n) g(r-m, s-n). \tag{4.131}$$

The size of $f * g$ is $(M + A - 1) \times (N + B - 1)$.

- Let

$$F(z_1, z_2) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) z_1^{-m} z_2^{-n}$$
$$G(z_1, z_2) = \sum_{m=0}^{A-1} \sum_{n=0}^{B-1} g(m,n) z_1^{-m} z_2^{-n}$$

be the z-transform of f and g respectively. Then $f * g(r,s)$ is the coefficients of the mono term $z_1^{-r} z_2^{-s}$ in the product $F(z_1, z_2) G(z_1, z_2)$.

- We use $\mathcal{F}$ to denote the Fourier transform.
- With circulant convolution, we have the convolution theorem for Fourier transform
$$\mathcal{F}[f \odot g] = \mathcal{F}[f] \cdot \mathcal{F}[g] \tag{4.132}$$
- This property does not hold for the (ordinary) convolution.

- To calculate the (ordinary) convolution by Fourier transform, we procedure as following.

  1. Extending f and g by zero-padding to size $K \times L$, where $K \geq M + A - 1$ and $L \geq N + B - 1$,

  $$f_e(m,n) = \begin{cases} f(m,n), & 0 \leq m < M \text{ and } 0 \leq n < N; \\ 0, & \text{otherwise.} \end{cases}$$

  $$g_e(m,n) = \begin{cases} g(m,n), & 0 \leq m < A \text{ and } 0 \leq n < B; \\ 0, & \text{otherwise.} \end{cases}$$

  2. Calculating

  $$\mathcal{F}^{-1}[\mathcal{F}[f] \cdot \mathcal{F}[g]] \tag{4.133}$$

  and convolution result result is obtained by taking those part of size $(M + A - 1) \times (N + B - 1)$ from the beginning.

- If f and g are periodic signals of the same periodicity, the correlation of f and g is defined as

$$R_{fg}(r,s) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m+r, n+s) g(m,n) \tag{4.134}$$

which is still a periodic signal of the same periodicity.

- Similarly to the circulant convolution, we have the discrete correlation property

$$\mathcal{F}[R_{fg}] = \mathcal{F}[f] \cdot \mathcal{F}[g]^* \tag{4.135}$$

  where a^* denotes the complex conjugate of a.

- R_{ff} is called the auto-correlation of f and its Fourier transform is called the power spectrum of f, denoted by S_f.

- Recall that white noise is defined with constant power spectrum.

Discrete representation of the image degradation model

- The discrete image degradation (4.121), after zero-padding, could be written as circulant convolution

$$g(i,j) = (f \odot h)(i,j) + \nu(i,j) \tag{4.136}$$

- If the original image f is of size $M \times N$, one may be only interested in those part of size $M \times N$ from the beginning of (4.133) or only those part is available in a practical image acquiring device, i.e., we assume that the original and the observed images are of the same resolution.

- To simplify notations, we denote f, g and ν by a specified one dimensional index i or j (e.g., by the dictionary order) and write (4.136) in the following matrix form
$$g = hf + \nu \tag{4.137}$$
where f and g are vector in $\mathbf{R}^N$ and h is a $N \times N$ matrix.

- Note that the resolutions of f and g may not be the same. Then the matrix h may not be a square matrix.

- However, one should keep in mind that the matrix multiplication hf is circulant convolution in the following.

Ill-posed Problem

- Given the basic formulation of image restoration problem above in (4.137).

- One of the most essential problems is the fact that image restoration is an ill-conditioned problem at best and a singular problem at worst.

- These terms will be left undefined for the moment, their specific meaning is to be calrified.

- In the mathematical sense, the problem of image restoration corresponds to the existence and uniqueness of an inverse transformation.

- In the best case, the noise is zero.
- If the inverse transform h^{-1} does not exist, then there is no mathematical basis for asserting that g can be exactly recovered from f.
- This is called *singular*.
- Singularity is common in image restoration.

- In the worst case, the noise is not zero.
- Every possible function in the ensemble of the random process ν is potentially the function added to g.
- There is an infinite family of object distribution.
- Even if h^{-1} exists, it may be *ill-conditioned*.
- I.e., a trivial(small) perturbation in g can produce non-trivial perturbations in the restored image.
- Examples can be constructed by Riemann-Lebesgue lemma, [Andrews and Hunt, 1977

- Another ill-conditioned source is when the smallest eigenvalue approaches zero.
- In the presence of noise(whether sensor noise or computational noise in the solution process), neat singularity is associated with very small eigenvalues and the solution of the resulting linear equations become difficult.
- Thus, solution of the digital restoration problem is thus tied to the solution of ill-conditioned (near-singular) systems of linear equations and *computational methods* become important.

- There is no *unique* solution in view of noise and ill-conditioning, and some meaningful *rationale* must be employed to pick a better solution.
- The selection of a specific solution from the family must be guided by some criterion or a set of criteria.
- Virtually, all image restoration schemes can be posed as the solution of some particular optimization problem.
- The restoration schemes will differ in the choice of criteria functions and/or the specific side conditions included as problem constraints.

Unconstrained least square restoration

- In the total absence of any knowledge about the noise term ν, the philosophy might be adopted that a criterion for solution would be based on an estimate of the restored object distribution, $\hat{f}$, that would satisfy (4.137) with the noise term ν being minimal in some sense.
- Picking the measure of ν as the norm of the vector, then the criterion stated in the previous sentence is equivalent to finding a solution $\hat{f}$ such that

$$< g - hf, g - hf >=< n, n >= n^{\text{tr}} n \tag{4.138}$$

is minimum.

- This is the so called *least squares* solution philosophy.
- Let

$$L(f) =< g - hf, g - hf >= \sum_i \left(g_i - \sum_j h_{ij} f_j \right)^2 . \tag{4.139}$$

- Differentiating with respect to each f_j(using the one-dimensional index introduced above),

$$\frac{\partial L(f)}{\partial f_j} = -2 \sum_i h_{ij} \left(g_i - \sum_{j'} h_{ij'} f_{j'} \right)$$

i.e.,

$$\frac{\partial L(f)}{\partial f} = -2h^{\mathrm{tr}}(g - hf).$$

- By the minimum criterion,

$$\frac{\partial L(f)}{\partial f} = -2h^{\mathrm{tr}}(g - h\hat{f}) = 0.$$

- Therefore
$$h^{\mathrm{tr}} h \hat{f} = h^{\mathrm{tr}} g.$$

- Then
$$\hat{f} = \left(h^{\mathrm{tr}} h\right)^{-1} h^{\mathrm{tr}} g$$

- $\hat{f}$ is equivalent to the inverse filtration if h is square and invertible,
$$\hat{f} = h^{-1} g$$

- If $H(u, v)$ has zeros, h is singular and neither h^{-1} nor $\left(h^{\mathrm{tr}} h\right)^{-1}$ exists.

Wiener Filtration

- Rather than seeking a solution consistent with minimum contamination by noise, we wish to attack the restoration problem directly and propose a criterion that explicitly evaluates how close the restoration is to the original image.

- Assume an estimate of the original image f. Call this estimate $\hat{f}$.

- A number of criteria measuring the estimation error can be posed. We use the minimum mean-square error (MMSE) here.

- The error is defined as

$$\epsilon = f - \hat{f}. \tag{4.140}$$

- The MMSE criterion requires the total error of estimation to be a minimum over entire ensemble of all possible images.

- The error as defined above can fluctuate both positive and negative.

- Hence, we consider the positive quantity

$$|\epsilon|^2 =< \epsilon, \epsilon > \tag{4.141}$$

- Since the transform h is linear, pragmatism indicates a linear estimate.

- I.e., an estimate of $\hat{f}$ derived by a linear operation on the observed image,

$$\hat{f} = lg \tag{4.142}$$

  We also assume that the above multiplication by l is also a two dimensional circulant convolution. Let the Fourier transform of l be L.

- L is to be derived such that (4.141) is minimized.

- Note

$$|\epsilon|^2 = ||F - \hat{F}||^2 = ||F - LG||^2$$
$$= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(L(u,v)G(u,v) - F(u,v))\,(L(u,v)^*G(u,v)^* - F(u,v)^*)$$

- Differentiating with respect to $L(u,v)^*$ we have

$$G(u,v)^*\,(L(u,v)G(u,v) - F(u,v)) = 0$$

for $u = 0, \cdots, M-1$ and $v = 0, \cdots, N-1$.

- Then

$$L(u,v) = \frac{G(u,v)^*F(u,v)}{|G(u,v)^2} \tag{4.143}$$
$$= \frac{(H(u,v)^*F(u,v)^* + N(u,v)^*)F(u,v)}{|H(u,v)|^2|F(u,v)|^2 + 2\mathrm{Re}\,[H(u,v)F(u,v)N(u,v)^*] + |N(u,v)|^2} \tag{4.144}$$

- By the discrete correlation property (4.135),

$$|F|^2 = S_f \quad \text{and} \quad |N|^2 = S_n$$

being the power spectra of f and n respectively.

- We assume the the signal f and the noise n are uncorrelated, i.e.,

$$R_{fn} = 0.$$

- By the discrete correlation property (4.135) again,

$$\mathcal{F}[R_{fn}] = FN^* = 0$$

- Then, by (4.143)

$$\begin{aligned} L(u,v) &= \frac{H(u,v)^* S_f(u,v)}{|H(u,v)|^2 S_f(u,v) + S_n(u,v)} \\ &= \frac{H(u,v)^*}{|H(u,v)|^2 + \frac{S_n(u,v)}{S_f(u,v)}} \end{aligned}$$

- If the noise is not present, $S_n = 0$, Wiener filtration reduces to inverse filtration.

- Usually the power spectra of the original image and the noise is not known *a priori*, a simplification to Wiener filtration is use a constant K to denote the ration of both of the spectra, assuming both are of constant power spectra.
- Then we have the following simplified Wiener filtration

$$L(u,v) = \frac{H(u,v)^*}{|H(u,v)|^2 + K}$$

- Note that

$$\frac{1}{K} = \frac{S_f(u,v)}{S_n(u,v)}$$

  is the signal-to-noise ratio.

Remark 4.4.1. 1. *The estimate by Wiener filtration is equal to the theoretical optimum only if the stochastic processes describing images f, g and the noise ν are stationary Gaussian. These conditions are not usually fulfilled for real life images.*

2. *The criterion of optimality is based on minimum mean square error and weights all errors equally, a mathematically fully acceptable criterion that unfortunately does not perform well if an image is restored for human viewing. The reason is that humans perceive the restoration errors more seriously in constant-gray-level area and in brightness regions and are much less sensitive to errors located in dark regions and in high gradient areas.*

3. *Spatially variant degradations cannot be restored using the standard Wiener filtration approach.*

4. *There are many other restoration techniques. This is an old but active research field.*

Example 4.4.2. *Restoration example by Wiener filtration. The Khoros workspace for this example is here Wiener filtration.*

Chapter 5

Segmentation

- Image segmentation is one of the most important steps leading to the analysis of processed image data.
- Its main goal is to divide an image into parts that have a strong correlation with objects or areas of the real world contained in the image.

- Two kinds of segmentation
- `Complete segmentation`:
  - which results in set of disjoint regions uniquely corresponding with objects in the input image.
  - Cooperation with higher processing levels which use specific knowledge of the problem domain is necessary.

- `Partial segmentation`:
  - in which regions do not correspond directly with image objects.
  - Image is divided into separate regions that are homogeneous with respect to a chosen property such as brightness, color, reflectivity, texture, etc.
  - In a complex scene, a set of possibly overlapping homogeneous regions may result. The partially segmented image must then be subjected to further processing, and the final image segmentation may be found with the help of higher level information.

- However, there is a whole class of segmentation problems that can be solved successfully using low-level processing only.
- In this case, the image commonly consists of contrasted objects on a uniform background — simple assembly tasks, blood cells, printed characters, etc.
- Here, a simple global approach can be used and the complete segmentation of an image into objects and background can be obtained.
- Such processing is context independent — no object-related model is used and no knowledge about expected segmentation results contributes to the final segmentation.
- Example: thresholding....

- Image data ambiguity is one of the main segmentation problems, often accompanied by information noise.

- Totally correct and complete segmentation of complex scenes usually cannot be achieved in this (low-level) processing phase.
- A reasonable aim is to use partial segmentation as an input to higher level processing.

- Segmentation methods can be divided into three groups according to the dominant features they employ
  - First is the `global knowledge segmentation` about an image or its part; the knowledge is usually represented by a histogram of image features.
  - The `Edge-based segmentation` form the second group;
  - The `region-based segmentation` is the third.

- Many different characteristics used in edge detection or region growing
  - brightness
  - texture
  - velocity field
  - etc.

🛑

- Edge-based and region-bases segmentation approaches solve a dual problem ... border $\times$ region.
- Each region can be represented by its closed boundary and each closed boundary describes a region.

- Because of the different natures of the various edge- and region-based algorithms, they may be expected to give somewhat different results and consequently different information.

- The segmentation results of these two approaches can therefore be combined in a single description structure.

- A common example of this is a region adjacent graph, in which regions are represented by nodes and graph arcs represent adjacent relations based on detected region borders, (3.2.3).

5.1 Thresholding

- Gray-level thresholding is the simplest segmentation process.
- Many objects or image regions are characterized by constant reflectivity or light absorption of their surface; a brightness constant or `threshold` can be determined to segment objects and background.
- Thresholding is computationally inexpensive and fast — it is the oldest segmentation method and is still widely used in simple applications.
- Thresholding can easily be done in real time using specialized hardware.

- A complete segmentation of an image R is a finite set of regions $R_1, \cdots, R_S$,

$$R = \bigcup_{i=1}^{S} R_i, \qquad R_i \bigcap R_j = \emptyset \quad \text{if } i \neq j. \tag{5.1}$$

- Complete segmentation can result from thresholding in simple scenes.

- Thresholding is the transformation of an input image f to an output (segmented) binary image g as follows

$$g(i,j) = \begin{cases} 1, & \text{for } f(i,j) \geq T; \\ 0, & \text{for } f(i,j) < T \end{cases} \tag{5.2}$$

  where T is the `threshold`, $g(i,j) = 1$ for image elements of objects and $g(i,j) = 0$ for image elements of the background (or vice versa).

🛑

Algorithm 5.1.1. *Basic thresholding*

Search all the pixels (i,j) of the image f. A pixel (i,j) is an object pixel if $f(i,j) \geq T$, and is a background pixel otherwise.

- If objects do not touch each other, and if their gray-levels are clearly distinct from background gray-levels, thresholding is a suitable segmentation method.
- Correct threshold selection is crucial for successful threshold segmentation.
- Threshold selection can be interactive or it can be the result of some threshold detection method that will be discussed in the next sub-section.

  Example 5.1.2. *Basic thresholding example. The Khoros workspace for this example is here Basic Thresholding.*

  - *Show students the results with different thresholding values.*
  - *Change the input image to* `airplane.kdf`*. Can we find a suitable threshold?*

- Only under very unusual circumstances can thresholding be successful using a single threshold for the whole image (global thresholding), since in very simple images there are likely gray-level variations in objects and background;

- This variation may be due to non-uniform lighting, non-uniform input device parameters or a number of other factors.

- Segmentation using variable thresholding (also called **adaptive thresholding**), in which the threshold value varies over the image as a function of local image characteristics, can produce the solution in these cases.

  – A global threshold is determined from the whole image f

  $$T = T(f). \tag{5.3}$$

  – On the other hand, local thresholds are position dependent

  $$T = T(f, f_c) \tag{5.4}$$

  where f_c is the image part in which the threshold is determined.

- One option is
  - dividing the image f into sub-images f_c and determining a threshold independently in each sub-image;
  - if a threshold cannot be determined in some sub-image, it can be interpolated from thresholds determined in neighboring sub-images;
  - each sub-image is then processed with respect to its local threshold.

- Basic thresholding as defined by equation (5.2) has many modifications.
  - `Band-thresholding`
    - Segment an image into regions of pixels with gray levels from a set D and into background otherwise

$$g(i,j) = \begin{cases} 1, & \text{for } f(i,j) \in D; \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

    - Thid thresholding can be useful, e.g., in micrroscopic blood cell segmentations.
    - This thresholding definition can serve as a border detector as well.
    - If the gray-level set D is chosen to contain just these object-border gray-levels, and if thresholding according to (5.5) is used, object borders can be found.
    - Isolines of gray can be found using this appropriate gray-level set D.

- There are many modification that use `multi-thresholding`, after which the resulting image is no longer binary, but rather an image consisting of a very limited set of gray-levels

$$g(i,j) = \begin{cases} 1, & \text{for } f(i,j) \in D_1; \\ 2, & \text{for } f(i,j) \in D_2; \\ \cdots & \\ n, & \text{for } f(i,j) \in D_n; \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

  where each D_i is a specified subset of gray-levels.

- Another special method of thresholding defines `semi-thresholding`
  - which is sometimes used to make human-assisted analysis easier

$$g(i,j) = \begin{cases} f(i,j), & \text{for } f(i,j) \geq T; \\ 0, & \text{for } f(i,j) < T \end{cases} \tag{5.7}$$

  - This process aims to mask out the image background, leaving gray level information present in the objects

- Thresholding has been presented relying only on gray-level image properties. Note that this is just one of many possibilities;
- Thresholding can also be applied if the values $f(i,j)$ do not represent gray-levels, but instead represent gradient, a local texture property or the value of any other image decomposition criterion.

5.1.1 Threshold detection methods

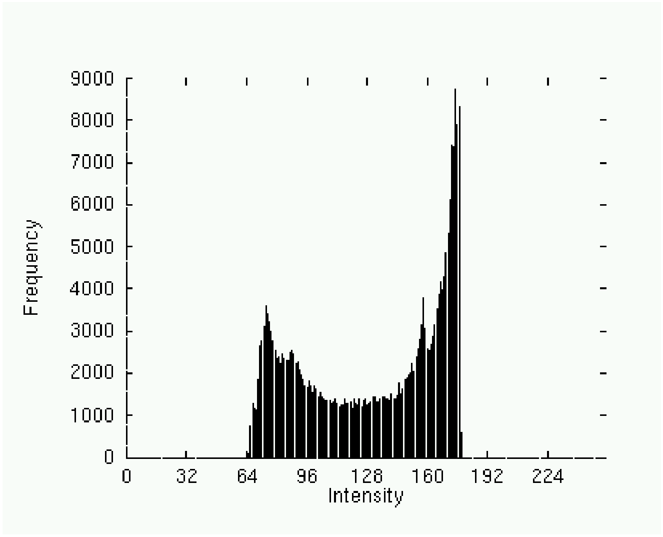
- If some property of an image after segmentation is known a priori, the task of threshold selection is simplified, since the threshold is chosen to ensure this property is satisfied.

- A printed text sheet may be an example if we know that characters of the text cover $1/p$ of the sheet area.

- Using this prior information about the ratio between the sheet area and character area, it is very easy to choose a threshold T (based on the image histogram), such that $1/p$ of the image area has gray values less than T and the rest has gray values larger than T.

- This method is called p-`tile-thresholding`.

- Most complex methods of threshold detection are based on histogram shape analysis.
- If an image consists of objects of approximately the same gray level that differs from the gray level of the background, the resulting histogram is bi-modal.



- Pixels of objects form one of its peaks, while pixels of the background form the second peak.
- The histogram shape illustrates the fact that the gray values between the two peaks are not common in the image, and *probably* result from border pixels between objects and background.
- The chosen threshold must meet minimum segmentation error requirements.
- It makes intuitive sense to determine the threshold as the gray-level that has a minimum histogram value between the two mentioned maxima.
- If the histogram is multi-modal, more thresholds may be determined at minima between any two peaks.
- Each threshold gives different segmentation results, of course.
- Multi-thresholding is another option.

- To decide if a histogram is bimodal or multi-modal may not be so simple in reality.
- It is often impossible to interpret the significance of local histogram maxima.

- Bi-modal histogram threshold detection algorithms usually find the highest local maxima first and detect the threshold as a minimum between them.
- This technique is called the `mode method`.

- To avoid detection of two local maxima belonging to the same global maximum, a minimum distance in gray levels between these maxima is usually required or techniques to smooth histograms are applied.
- Note that histogram bi-modality itself does not guarantee correct threshold segmentation — even if the histogram is bi-modal, correct segmentation may not occur with objects located on a background of different gray-levels.

- A two-part image with one half white and the second half black actually has the same histogram as an image with randomly spread white and black pixels.
- This is one example showing the need to check threshold segmentaion results whenever the threshold has been determined from a histogram only, using no other image characteristics.

STOP

- Thresholding is a very popular tool in image segmentation, and a large variety of thresholding detection techniques exist in addition to the main techniques discussed above.
- Real-time threshold detection is a current research effort.

5.1.2 Optimal thresholding

- This method is based on approximation of the histogram of an image using a weighted sum of two or more probability densities with normal distributions.
- The threshold is set as the closest gray level corresponding to the probability between the maxima of two or more normal distributions, which results in minimum error segmentation, i.e., the number of mis-segmented pixels is smallest,

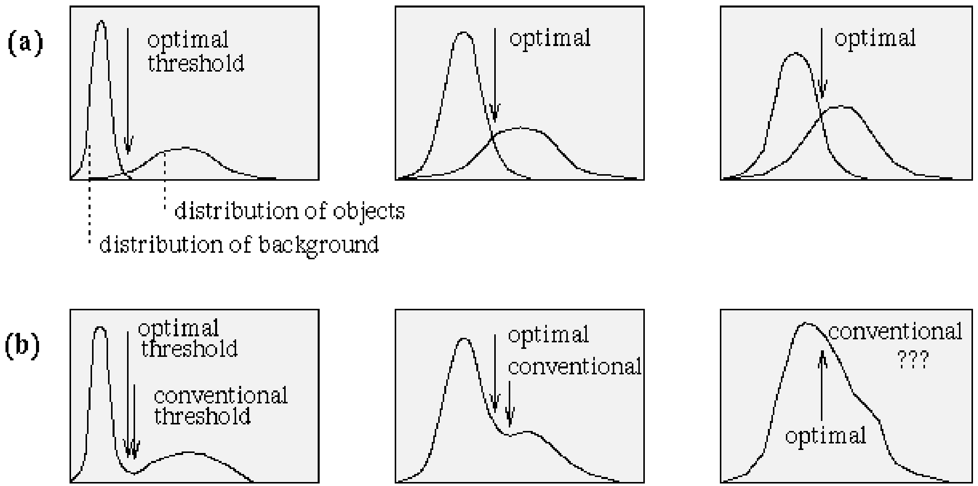


Figure 5.4 *Grey level histograms approximated by two normal distributions; the threshold is set to give minimum probability of segmentation error: (a) Probability distributions of background and objects, (b) corresponding histograms and optimal threshold.*

- Let's consider a simple case in the following to illustrate the idea of this method.
- Assume the image consists of the objects and background, where the objects occupies θ percent of the pixels.
- Assume the objects are subject to a normal distribution $\mathcal{N}(m_o, \sigma_o)$ and that the background is subject to a normal distribution $\mathcal{N}(m_b, \sigma_b)$, as shown in the above figure.

- By the total probability rule, the image is with the following density function

$$d(z) = \theta\mathcal{N}(m_o, \sigma_o) + (1-\theta)\mathcal{N}(m_b, \sigma_b) \tag{5.8}$$

Proof. Let f be the random variable representing the image. Let o and b be the random variables representing the object and background, respectively. By the total probability rule,

$$\mathbf{Pr}(A) = \sum_i \mathbf{Pr}(H_i)\mathbf{Pr}(A|H_i) \tag{5.9}$$

where H_i are events such that $\bigcup_i H_i$ is the total space and $H_i \bigcap H_j = \emptyset$ if $i \neq j$.

Then

$$\begin{aligned}\mathbf{Pr}(f \leq z) &= \mathbf{Pr}(f = o)\mathbf{Pr}(f \leq z|f = o) + \mathbf{Pr}(f = b)\mathbf{Pr}(f \leq z|f = b)\\ &= \theta\mathbf{Pr}(o \leq z) + (1-\theta)\mathbf{Pr}(b \leq z)\end{aligned}$$

□

- Now let t be the threshold. Then the mis-segmentation takes place in the following two cases.

- **Background pixels are mis-classified into object pixels**: the error probability (or the number of errors) is

$$\int_t^{\infty} \mathcal{N}(m_b, \sigma_b)(z)dz = 1 - \int_{-\infty}^{t} \mathcal{N}(m_b, \sigma_b)(z)dz. \tag{5.10}$$

- **Object pixels are mis-classified into background pixels**: the error probability (or the number of errors) is

$$\int_{-\infty}^{t} \mathcal{N}(m_o, \sigma_o)(z)dz. \tag{5.11}$$

- So the total error of mis-segmentation is, again by the total probability rule

$$E(t) = \theta \int_{-\infty}^{t} \mathcal{N}(m_o, \sigma_o)(z)dz + (1-\theta) \left[1 - \int_{-\infty}^{t} \mathcal{N}(m_b, \sigma_b)(z)dz\right] \tag{5.12}$$

- The optimal threshold is

$$\hat{t} = \mathop{\mathbf{arg\,min}}_{t} E(t) \tag{5.13}$$

- Differentiating $E(t)$, we find $\hat{t}$ should satisfy

$$\frac{\partial E(t)}{\partial t} = \theta \mathcal{N}(m_o, \sigma_o)(t) - (1-\theta)\mathcal{N}(m_b, \sigma_b)(t) = 0 \tag{5.14}$$

- I.e.,

$$\theta \mathcal{N}(m_o, \sigma_o)(t) = (1-\theta)\mathcal{N}(m_b, \sigma_b)(t) \tag{5.15}$$

- Substituting the formulas for the Gaussians into the above equation, we obtain

$$\frac{\theta}{\sqrt{2\pi\sigma_o^2}} \mathbf{e}^{-\frac{(t-\mu_o)^2}{2\sigma_o^2}} = \frac{1-\theta}{\sqrt{2\pi\sigma_b^2}} \mathbf{e}^{-\frac{(t-\mu_b)^2}{2\sigma_b^2}} \tag{5.16}$$

- Or

$$\frac{(t-\mu_o)^2}{2\sigma_o^2} - \frac{(t-\mu_b)^2}{2\sigma_b^2} = \log \frac{\theta\sigma_b}{(1-\theta)\sigma_o} \tag{5.17}$$

- Two specific examples are as following.

- If $\theta = \frac{1}{2}$ and $\sigma_o = \sigma_b$, the optimal threshold is

$$t = \frac{m_o + m_b}{2}. \tag{5.18}$$

- If $\sigma_o = \sigma_b$, the optimal threshold is

$$t = \frac{m_o + m_b}{2} + \frac{\sigma_o}{m_b - m_o} \log \frac{\theta}{1-\theta}. \tag{5.19}$$

- The above approach can be easily extended to the case using the combination of several Gaussians to approximate the the histogram of the image and to multi-thresholding.
- The difficulty with this method is in estimating normal distribution parameters together with the uncertainty that the distribution may be considered normal.
- Alternatively, other minimization approaches may be used.

- The following algorithm represents a simpler version that shows a rational for this approach and works well even if the image histogram is not bi-modal.
- It assumes that regions of two main gray-levels are present in image, thresholding of printed text being an example.

- The algorithm is iterative, four to ten iterations usually being sufficient.

Algorithm 5.1.3. *Iterative optimal threshold selective algorithm*

1. *Assuming no knowledge about the exact location of objects, consider as a first approximation that the four corners of the image contain background pixels only and the remainder contains object pixels;*
2. *At step n, compute μ_B^n and μ_O^n as the mean background and object gray-level, respectively, where segmentation into background and objects at step n is defined by the threshold T^n determined in the previous step;*
3. *Set*
$$T^{(n+1)} = \frac{\mu_B^n + \mu_O^n}{2}. \tag{5.20}$$
$T^{(n+1)}$ *now provides an updated background-object distinction.*
4. *If $T^{(n+1)} = T^n$, halt; otherwise return to step 2.*

Example 5.1.4. *Threshold a grayscale image using the terative optimal threshold selective algorithm. The matlab script from* **visionbook** *is here Histogram equalization Example.*

5.2 Edge-based segmentation

- Edge-based segmentation represents a large group of methods based on information about edges in the image; it is one of the earliest segmentation approaches and still remains very important.

- Edge-based segmentations rely on edges found in an image by edge detecting operators — these edges mark image locations of discontinuities in gray level, color, texture, etc.

- But the image resulting from edge detection cannot be used as a segmentation result.

- Supplementary processing steps must follow to combine edges into edge chains that correspond better with borders in the image.

- The final aim is to reach at least a partial segmentation — that is, to group local edges into an image where only edge chains with a correspondence to existing objects or image parts are present.

- We will discuss several edge-based segmentation methods which differ in strategies leading to final border construction, and also differ in the amount of prior information that can be incorporated into the method.

- The more prior information that is available to the segmentation process, the better the segmentation results that can be obtained.

STOP

- The most common problems of edge-based segmentation, caused by image noise or unsuitable information in an image, are an edge presence in locations where there is no border, and no edge presence where a real border exists.

- First we will discuss simple edge-based methods requiring minimum prior information and the necessity for prior knowledge will increase during the section. Construction of regions from edge-based partial segmentations is discussed at the end of the section.

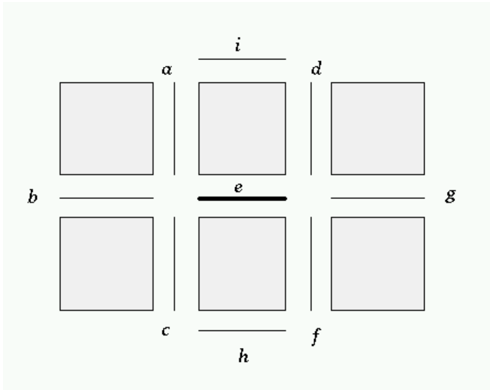
5.2.1 Edge image thresholding

- Almost no zero-value pixels are present in an edge image, but small edge values correspond to non-significant gray level changes resulting from, e.g., quantization noise, small lighting irregularities, etc.
- Simple thresholding of an edge image can be applied to remove these small values.
- The approach is based on an image of edge magnitude processed by an appropriate threshold.
- Selection of an appropriate global threshold is often difficult and sometimes impossible; p-`tile thresholding` can be applied to define a threshold.
- Alternatively, non-maximal suppression (4.3.8) and hysteresis thresholding (4.3.5) can be used as was introduced in the Canny edge detector.

5.2.2 Edge relaxation

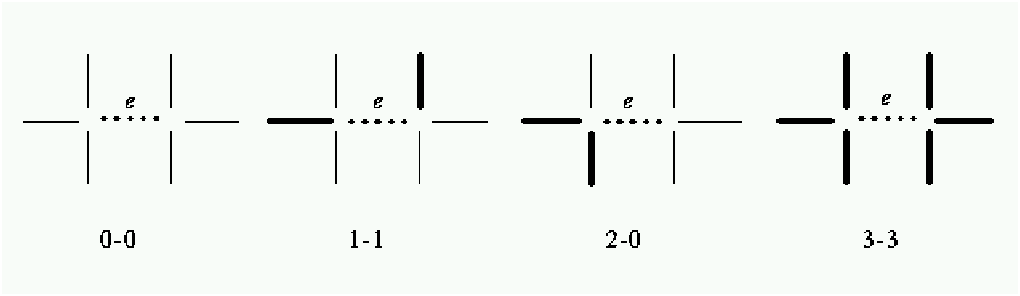
- Borders resulting from the previous method are strongly affected by image noise, often with important parts missing.
- Considering edge properties in the context of their mutual neighbors can increase the quality of the resulting image.
- All the image properties, including those of further edge existence, are iteratively evaluated with more precision until the edge context is totally clear — based on the strength of edges in a specified local neighborhood, the confidence of each edge is either increased or decreased.
- A weak edge positioned between two strong edges provides an example of context; it is highly probable that this inter-positioned weak edge should be a part of a resulting boundary.
- If, on the other hand, an edge (even a strong one) is positioned by itself with no supporting context, it is probably not a part of any border.

- A method we are going to discuss here is a classical example of edge context evaluation.
- This method uses crack edges (edges located between pixels), (2.3.1).
- Edge context is considered at both ends of an edge, giving the minimal edge neighborhood.



- The central edge **e** has a vertex at each of its ends and three possible border continuations can be found from both of these vertexes.

- All three possible edge positions at the end of the edge **e** must be included to cover all the possible ways the border can continue from both ends of **e**.
- Edge relaxation aims for continuous border construction, so we discuss the edge patterns that can be found in the local neighborhood.
- Let each vertex be evaluated according to the number of edges emanating from the vertex, not counting the edge **e**. Call this number the vertex type.



- The type of edge **e** can then be represented using a number pair $i-j$ describing edge patterns at each vertex, where i and j are the vertex types of the edge **e**.

- By symmetry, we only list the cases where $i \leq j$. The following context situations are possible:
  - $0 - 0$: isolated edge — negative influence on the edge confidence;
  - $0 - 2$, $0 - 3$: dead end — negative influence on edge confidence;
  - $0 - 1$: uncertain — weak positive, or no influence on edge confidence;
  - $1 - 1$: continuation — strong positive influence on edge confidence;
  - $1 - 2$, $1 - 3$: continuation to border intersection — medium positive influence on edge confidence;
  - $2 - 2$, $2 - 3$, $3 - 3$: bridge between borders — not necessary for segmentation, no influence on edge confidence.

- Edge relaxation is an iterative method, with edge confidences converging either to zero (edge termination) or one (the edge forms a border).
- The confidence of each edge **e** in the first iteration can be defined as a normalized magnitude of the crack edge, with normalization based on
  - either the global maximum of crack edges in the whole image;
  - or on a local maximum in some large neighborhood of the edge, thereby decreasing the influence of a few very high values of edge magnitude in the image.

Algorithm 5.2.1. *Edge Relaxation*

1. *Evaluate a confidence $c^{(1)}(\mathbf{e})$ for all crack edges $\mathbf{e}$ in the image;*
2. *Find the edge type of each edge based on edge confidences in its neighborhood;*
3. *Update the confidence $c^{(k+1)}(\mathbf{e})$ according to its type and its previous confidence $c^{(k)}(\mathbf{e})$;*
4. *Stop if all edge confidence have converged either to 0 or 1. Repeat steps (2) and (3) otherwise.*

- The main steps of the above algorithm are evaluation of vertex types followed by evaluation of edge types, and the manner in which the edge confidences are modified.

- A vertex is considered to be of type i if

$$\text{type}(i) = \operatorname*{\mathbf{arg\,max}}_{k=0,1,2,3} \text{type}(k) \tag{5.21}$$

where

$$\text{type}(0) = (m-a)(m-b)(m-c) \tag{5.22}$$
$$\text{type}(1) = a(m-b)(m-c) \tag{5.23}$$
$$\text{type}(2) = ab(m-c) \tag{5.24}$$
$$\text{type}(3) = abc \tag{5.25}$$

where a, b, c are the normalized values of the other incident crack edges, $a \geq b \geq c$,

$$m = \max\{a, b, c, q\}. \tag{5.26}$$

The introduction of the quantity q ensures that type(0), is non-zero for small values of a and is comprable with respect to type(1), type(2), and type(3).

- For example, consider the case $q = 0.5$, and a, b, c take the values 0 or 1. Equation gives the correct vertex type as we discussed above.

- Another non-trivial example, choosing $q = 0.1$, a vertex $(a, b, c) = (0.5, 0.05, 0.05)$ is a type 1 vertex, while a vertex $(0.3, 0.2, 0.2)$ is a type 3 vertex.

- Similar results can be obtained by simply counting the number of edges emanating from the vertex above a threshold value. This coincides with the above example where a, b, c only take the 0 and 1 values., (5.2.2).

- Edge confidences are modified as follows, increased or decrease according to the influence on the edge confidence, (5.21):

$$\text{confidence increase:} \qquad c^{(k+1)}(\mathbf{e}) = \min\{1, c^{(k)}(\mathbf{e}) + \delta\} \tag{5.27}$$

$$\text{confidence decrease:} \qquad c^{(k+1)}(\mathbf{e}) = \max\{0, c^{(k)}(\mathbf{e}) - \delta\} \tag{5.28}$$

$$\tag{5.29}$$

STOP

- Edge relaxation, as described above, rapidly improves the initial edge labeling in the first few iterations.

- Unfortunately, it often slowly drifts, giving worse results than expected after larger numbers of iterations.
- The reason for this strange behavior is in searching for the global maximum of the edge consistency criterion over all the image, which may not give locally optimal results.
- A theoretical explanation, convergence proof, and practical solutions are given in the reference in the text book.
- A solution is found in setting edge confidences to zero under a certain threshold, and to one over another threshold which increases the influence of original image data.
- Therefore, one additional step must be added to the edge confidence computation (5.27) and (5.28)

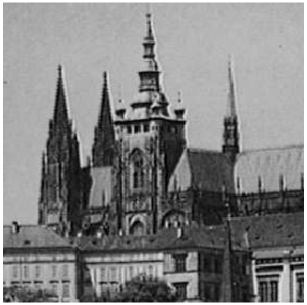
$$\text{if } c^{(k+1)}(\mathbf{e}) > T_1 \text{ then assign } c^{(k+1)}(\mathbf{e}) = 1 \tag{5.30}$$

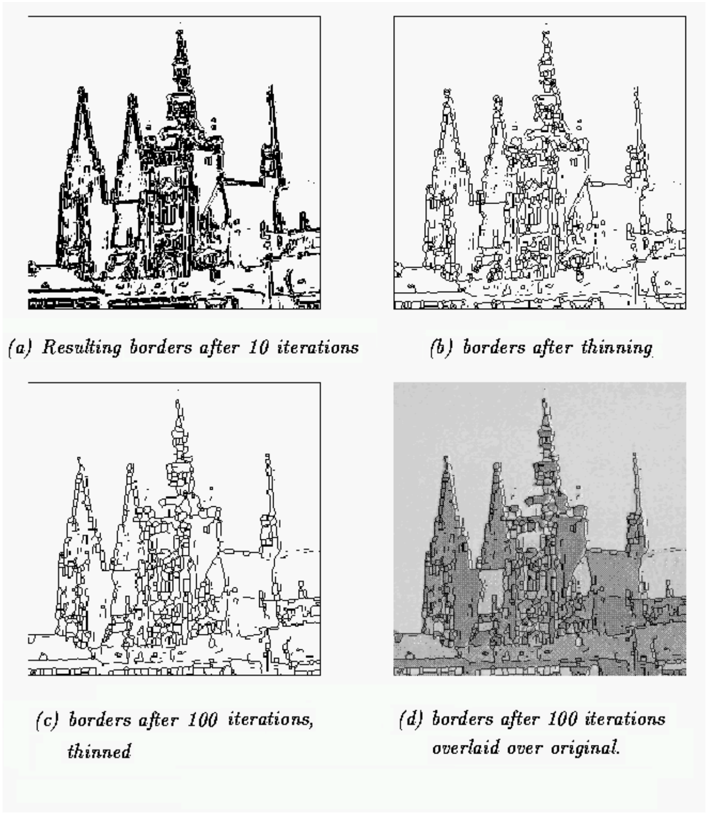
$$\text{if } c^{(k+1)}(\mathbf{e}) < T_2 \text{ then assign } c^{(k+1)}(\mathbf{e}) = 0 \tag{5.31}$$

$$\tag{5.32}$$

where T_1 and T_2 are parameters controlling the edge relaxation convergence speed and resulting border accuracy.

Example 5.2.2. *Edge detection by edge relaxation.*


(a) Original image


(a) Resulting borders after 10 iterations
(b) borders after thinning
(c) borders after 100 iterations, thinned
(d) borders after 100 iterations overlaid over original.
(b) Edge detection by edge relaxation

- More recent approaches to edge relaxation use edge and border information derived from image data.
- A method to determine probabilistic distribution of possible edge neighborhood is given in recent literature.

5.2.3 Border tracing

Simple border tracing

- If a region border is not known but regions have been defined in the image, borders can be uniquely detected.
- First, let us assume that the image with regions is either binary or that regions have been labeled.
- An inner region border is a subset of the region while an outer border is not a subset of the region, § 2.3.1.
- The first goal is to determine inner region borders.

- The following algorithm covers inner boundary tracing in both 4-connectivity and 8-connectivity.

Algorithm 5.2.3. *Inner boundary tracing*

1. *Search the image from top left until a pixel of a new region is found; This pixel P_0 then has the minimum column value of all pixels of that region having the minimum row value. Pixel P_0 is a starting pixel of the region border.*

   Define a variable dir which stores the direction of the previous move along the border from the previous border element to the current border element. Assign

   (a) *$dir = 3$ if the border is detected in 4-connectivity;*

   (b) *$dir = 7$ if the border is detected in 8-connectivity.*

2. *Search the 3×3 neighborhood of the current pixel in an anti-clockwise direction, beginning the neighborhood search in the pixel positioned in the direction*

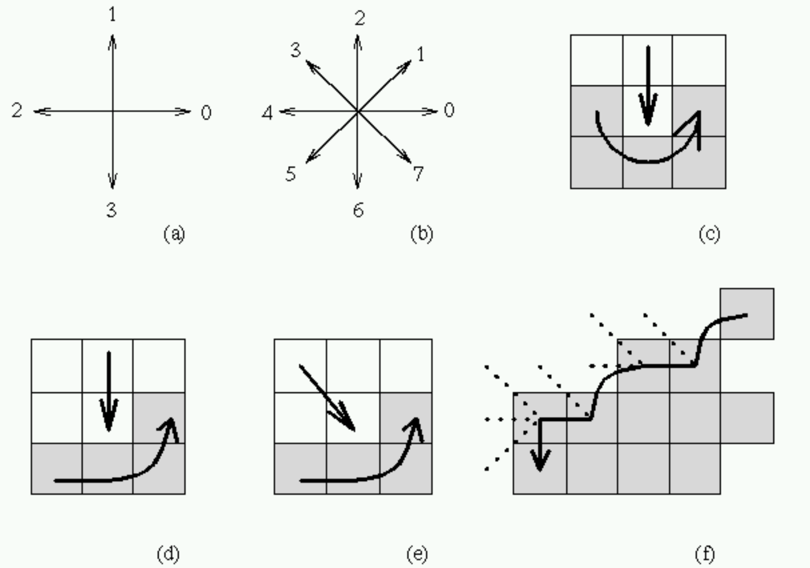
   (a) $(dir + 3) \mod 4$ *if the border is detected in 4-connectivity;*

   (b) $(dir + 7) \mod 8$ *if dir is even;*
   $(dir + 6) \mod 8$ *if dir is odd.*

The first pixel found with the same value as the current pixel is a new boundary element P_n.

Update the dir value.

3. *If the current boundary element P_n is equal to the second border element P_1, and if the previous border element P_{n-1} is equal to P_0, stop. Otherwise repeat step 2.*
4. *The detected inner border is represented by pixels $P_0 \cdots P_{n-2}$.*



Inner boundary tracing: (a) Direction notation, 4-connectivity, (b) 8-connectivity, (c) pixel neighborhood search sequence in 4-connectivity, (d),(e) search sequence in 8-connectivity, (f) boundary tracing in 8-connectivity (dashed lines show pixels tested during the border tracing).

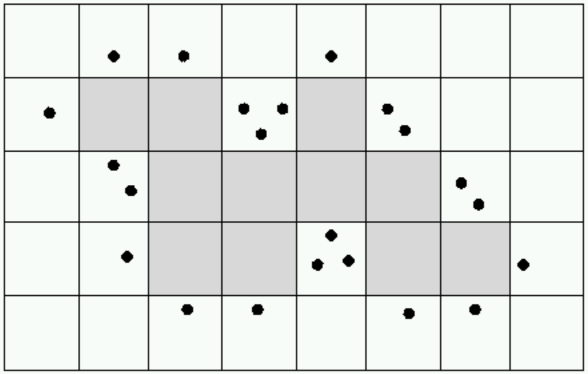
- The above algorithm works for all regions larger than one pixel.
- This algorithm is able to find region borders but does not find borders of region holes.
- To search for hole borders as well, the border must be traced starting in each region or hole border element if this element has never been a member of any border previously traced.
- The search for border elements always starts after a currently traced border is closed, and the search for 'unused' border elements can continue in the same way as the search for the first border element was done.
- Note that if objects are of unit width, more conditions must be added.

- If the goal is to detect an outer region border, the given algorithm may still be used based on 4-connectivity.

Algorithm 5.2.4. *Outer boundary tracing*

1. *Trace the inner region boundary in 4-connectivity until done;*
2. *The outer boundary consists of all non-region pixels that were tested during the search process; if some pixels were tested more than once, they are listed more than once in the outer boundary list.*

- Note that some border elements may be repeated in the outer border up to three times. See the following figure.

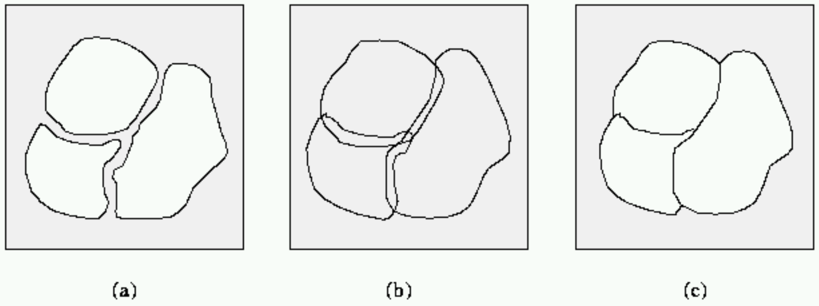


Outer boundary tracing; ● denotes outer border elements. Note that some pixels may be listed several times.

- The outer region border is useful for deriving properties such as perimeter, compactness, etc.

Extended border tracing

- The inner border is always part of a region but the outer border never is.
- Therefore, if two regions are adjacent, they never have a common border, which causes difficulties in higher processing levels with region description, region merging, etc.
- The inter-pixel boundary extracted, for instance, from crack edges is common to adjacent borders; nevertheless, its position cannot be specified in pixel co-ordinates.
- Boundary properties better than those of outer borders may be found in `extended borders`, [Pavlidis, 1977].



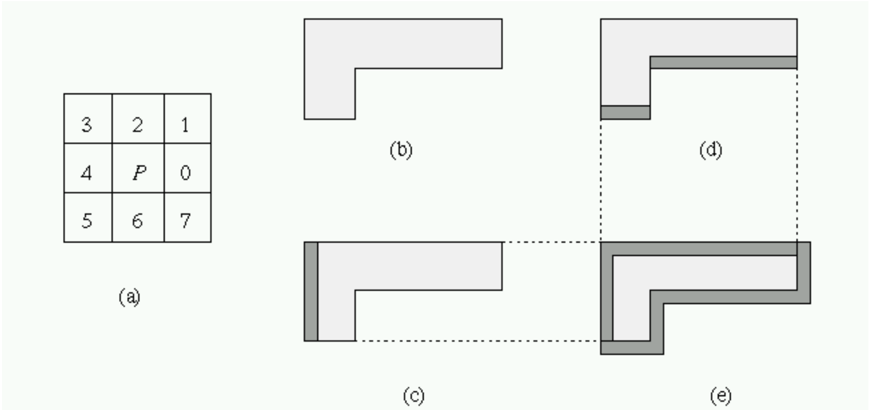
Boundary locations for inner, outer, and extended boundary definition: (a) Inner, (b) outer, (c) extended.

- The advantages of the extended boundary definition are:
  - it defines a single common border between adjacent regions;
  - it may be specified using standard pixel co-ordinates;
  - all the useful properties of the outer border still remain, for deriving properties such as perimeter, compactness, etc.;
  - the boundary shape is exactly equal to the inter-pixel shape but is shifted one half-pixel down and one half-pixel right

- The existence of a common border between regions makes it possible to incorporate into the boundary tracing a boundary description process.

- An `evaluated graph` consisting of border segments and vertexes may result directly from the boundary tracing process; also the border between adjacent regions may be traced only once and not twice as in conventional approaches.

- The extended boundary is defined using 8-neighborhoods and the pixels are coded according to the following figure:



Extended boundary definition: (a) Pixel coding scheme, (b) region R, (c) LEFT(R), (d) LOWER(R), (e) extended boundary.

- E.g., $P_4(P)$ denotes the pixel immediately to the left of pixel P.

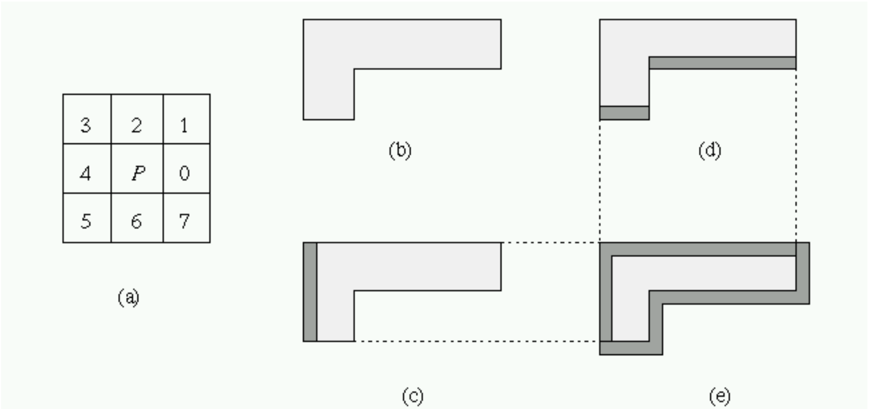
- Four kinds of inner boundary pixels of a region R are defined; if Q denotes the pixels outside the region R, then a pixel $P \in R$ is

$$\begin{array}{l}\text{a LEFT pixel of } R \text{ if } P_4(P) \in Q \\ \text{a RIGHT pixel of } R \text{ if } P_0(P) \in Q \\ \text{an UPPER pixel of } R \text{ if } P_2(P) \in Q \\ \text{a LOWER pixel of } R \text{ if } P_6(P) \in Q.\end{array}$$

- Let LEFT(R), RIGHT(R), UPPER(R), LOWER(R) represent the corresponding subsets of R.

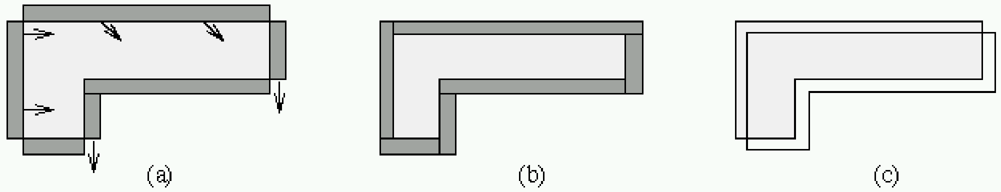
- The extended boundary EB is defined as the following:

$$
\begin{aligned}
\mathrm{EB} = & \{P : P \in \mathrm{LEFT}(R)\} \cup \{P : P \in \mathrm{UPPER}(R)\} \\
& \cup \{P_6(P) : P \in \mathrm{LOWER}(R)\} \cup \{P_0(P) : P \in \mathrm{RIGHT}(R)\} \\
& \cup \{P_7(P) : P \in \mathrm{RIGHT}(R)\}.
\end{aligned}
$$



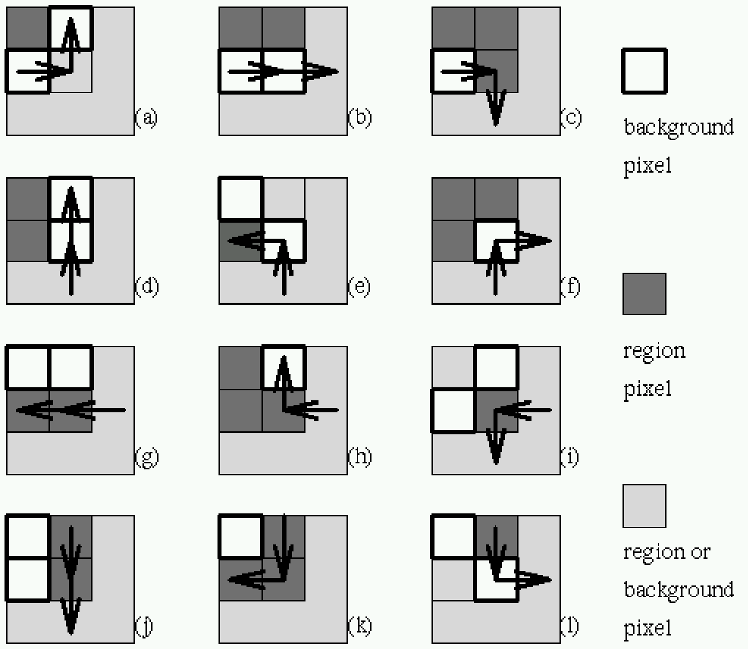
Extended boundary definition: (a) Pixel coding scheme, (b) region R, (c) LEFT(R), (d) LOWER(R), (e) extended boundary.

- The extended boundary can easily be constructed from the outer boundary.
- Using an intuitive definition of RIGHT, LEFT, UPPER, and LOWER outer boundary points, the extended boundary may be obtained by shifting all the UPPER outer boundary points one pixel down and right, shifting all the LEFT outer boundary points one pixel to the right, and shifting all the RIGHT outer boundary points one pixel down. The LOWER outer boundary point positions remain unchanged.
- The following figure illustrates the construction.



Constructing the extended boundary from outer boundary: (a) Outer boundary, (b) extended boundary construction, (c) extended boundary has the same shape and size as the natural object boundary.

- A more sophisticated approach for extended boundary tracing is based on detecting common boundary segments between adjacent regions and vertex points in boundary segment connections.
- The detection process is based on a look-up table, which defines all 12 possible situations of the local configuration of 2×2 pixel windows, depending on the previous detected direction of boundary, and on the status of window pixels which can be inside or outside a region.



Look-up table defining all 12 possible situations that can appear during extended border tracing. Current position is in the central pixel. The direction of the next move depends on the local configuration of background and region points, and on the direction of approach to the current pixel.

Algorithm 5.2.5. *Extended boundary tracing via look-up table.*

1. *Define a starting pixel of an extended boundary in a standard way (the first region pixel found in a left-to-right and top-to-bottom line-by-line image search).*

2. *The first move along the traced boundary from the starting pixel is in direction $dir = 6$ (down), corresponding to the situation (i) in Fig. 5.1.*

3. *Trace the extended boundary using the look-up table in the above figure until a closed extended border results.*

- Note that there is no hole-border tracing included in the algorithm.
- The holes are considered separate regions and therefore the borders between the region and its hole are traced as a border of the hole.

- The look-up table approach makes the tracing more efficient than conventional methods and makes parallel implementation possible.

- A pseudo-code description of algorithmic details is given in [Liow, 1991], where a solution to the problem of tracing all the borders in an image in an efficient way is given.

- In addition to extended boundary tracing, it provides a description of each boundary segment in chain code form together with information about vertices's.

- This method is very suitable for representing borders in higher level segmentation approaches including methods that integrate edge-based and region-based segmentation results.

- Moreover, in the conventional approaches, each border between two regions must be traced twice. The algorithm can trace each boundary segment only once storing the information about what has already been done in double-linked lists.

Border tracing in gray-level images

- A more difficult situation is encountered if the borders are traced in gray level images where regions have not yet been defined.
- The border is represented by a simple path of high-gradient pixels in the image.
- Border tracing should be started in a pixel with a high probability of being a border element, and then border construction is based on the idea of adding the next elements which are in the most probable direction.
- To find the following border elements, edge gradient magnitudes and directions are usually computed in pixels of probable border continuation.

5.2.4 Border detection as graph searching

- Whenever additional knowledge is available for boundary detection, it should be used.
- One example of prior knowledge is a known starting point and a ending point of the border, even if the precise border location is not known.
- Even some relatively weak additional requirements such as smoothness, low curvature, etc., may be included as prior knowledge.
- If this kind of supporting information is available in the border detection task, general problem solving methods used in AI can be applied.

- A graph is a general structure consisting of a set of nodes n_i and arcs between the nodes (n_i, n_j), § 3.2.3. We consider oriented and numerically weighted arcs, these weights being called costs.

- The border detection process is transformed into a search for the optimal path in the weighted graph, the aim being to find the best path that connects two specified nodes, the starting and ending nodes.

- While cost minimization is considered throughout this section, the approach works equally well if a maximum cost path is sought.

STOP

- Assume that both edge magnitude $s(x)$ and edge direction $\phi(x)$ information is available in an edge image.

- Each image pixel corresponds to a graph node weighted by a values $s(x)$.

- Two nodes n_i and n_j corresponding to two 8-connected adjacent pixels x_i and x_j are connected by an arc if the edge directions $\phi(x_i)$ and $\phi(x_j)$ match the local border direction in a sense to be defined below.

- We can apply the following rules to construct the graph: To connect a node n_i representing the pixel x_i with a node n_j representing the pixel x_j,

  1. pixel x_j must be one of three existing neighbors of x_i in the direction $d \in [\phi(x_i) - \pi/4, \phi(x_i) + \pi/4]$;
  2. $s(x_i)$ and $s(x_j)$ must be greater than T, where T is some preset threshold of edge significance.

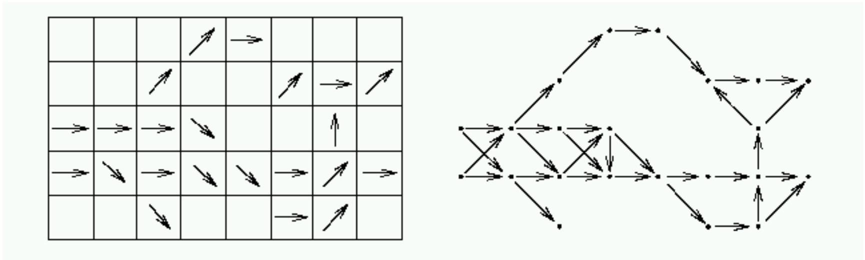
STOP

- Another common requirement is to connect two nodes only if the difference of their edge direction is less than $\pi/2$.

STOP

- These conditions can be modified in specific edge detection problems.

- The following figure shows an image of edge directions, with only significant edges according their magnitudes listed, and the oriented graph constructed in accordance with the edge image.



Graph representation of an edge image: (a) Edge directions, (b) corresponding graph.

- The application of graph search to edge detection was first published in [Martelli, 1972].

- Let x_A be the starting border element, and x_B be the end border element.

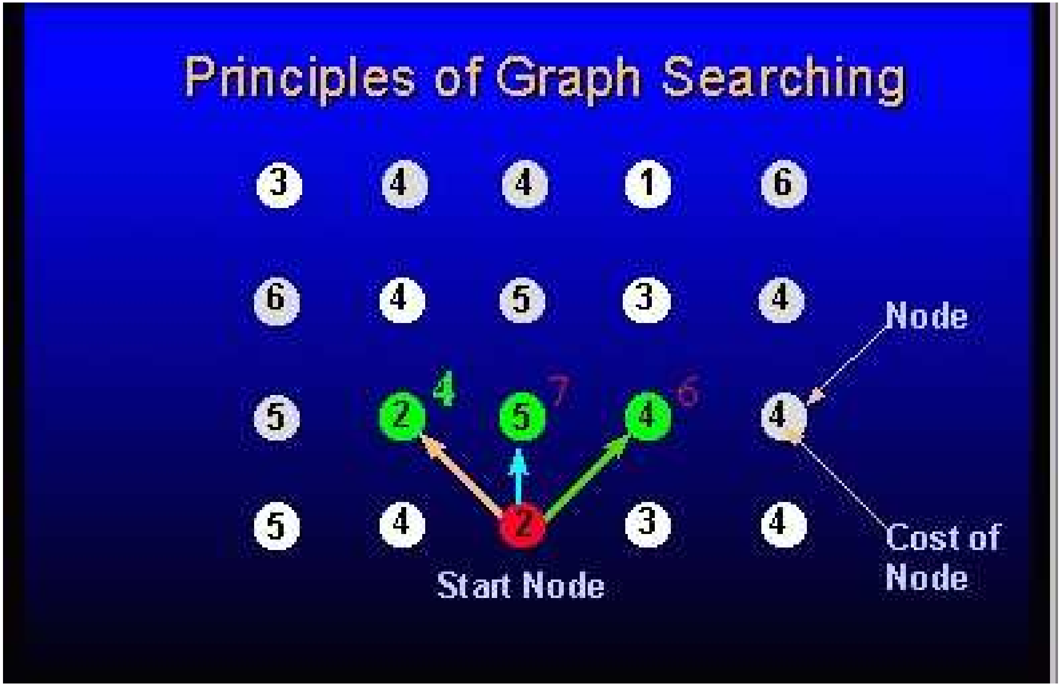
- To use graph search for region border detection, a method of oriented weighted-graph expansion must first be defined (one method is just described.)
- A cost function $f(x_i)$ must also be defined that is a cost estimate of the path between nodes n_A and n_B (pixels x_A and x_B) which goes through an intermediate node n_i (pixel x_i).
- The cost function $f(x_i)$ typically consists of two components: an estimate $\tilde{g}(x_i)$ of the minimum path cost between the starting element x_A and x_i, and an estimate $\tilde{h}(x_i)$ of the minimum path cost between x_i and the end border element x_B.
- The cost $\tilde{g}(x_i)$ of the path from the starting element x_A to the node x_i is usually a sum of costs associated with the arcs or nodes that are in the path.
- The cost function must be separable and monotonic with respect to the path length, and therefore the local costs associated with arcs are required to be non-negative.
- A simple example of $\tilde{g}(x_i)$ satisfying the given conditions is to consider the path length from x_A to x_i.
- An estimate $\tilde{h}(x_i)$ may be the length of the border from x_i to x_B, it making sense to prefer shorter borders between x_A and x_B as the path with lower cost.

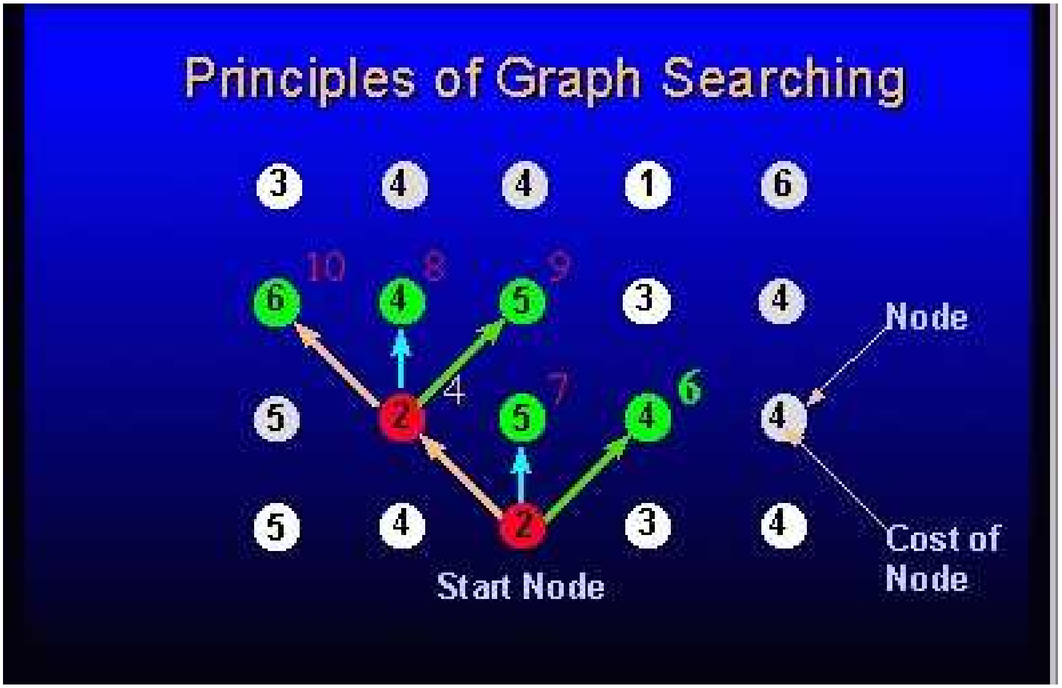
- The following graph search algorithm (Nilsson's A-algorithm from AI) can be applied to the border detection.

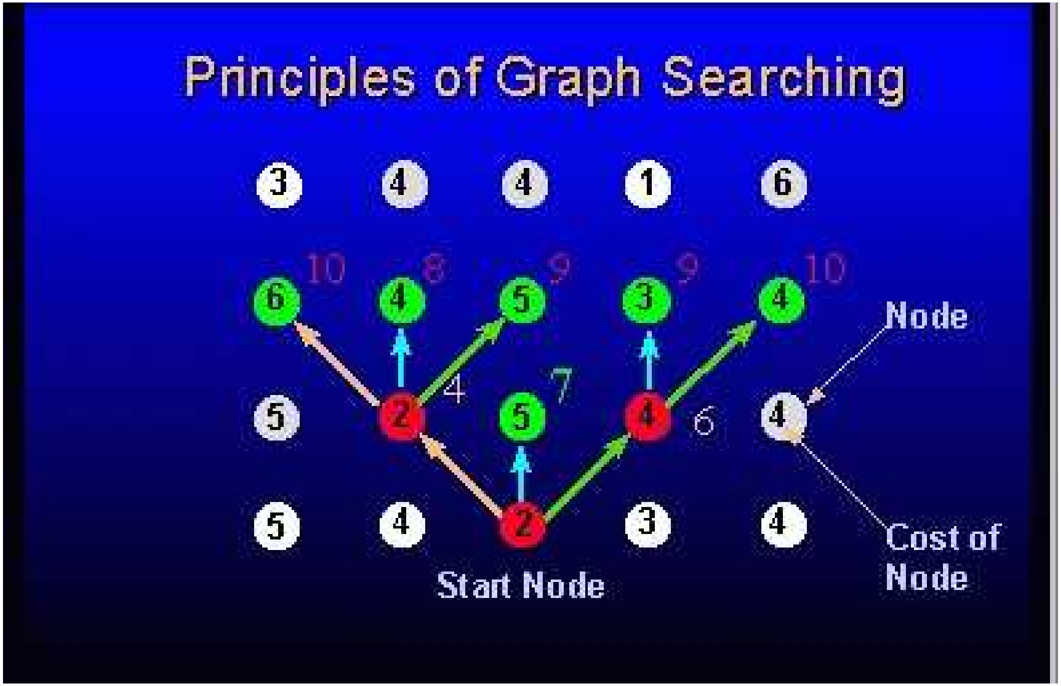
Algorithm 5.2.6. *A-algorithm graph search*

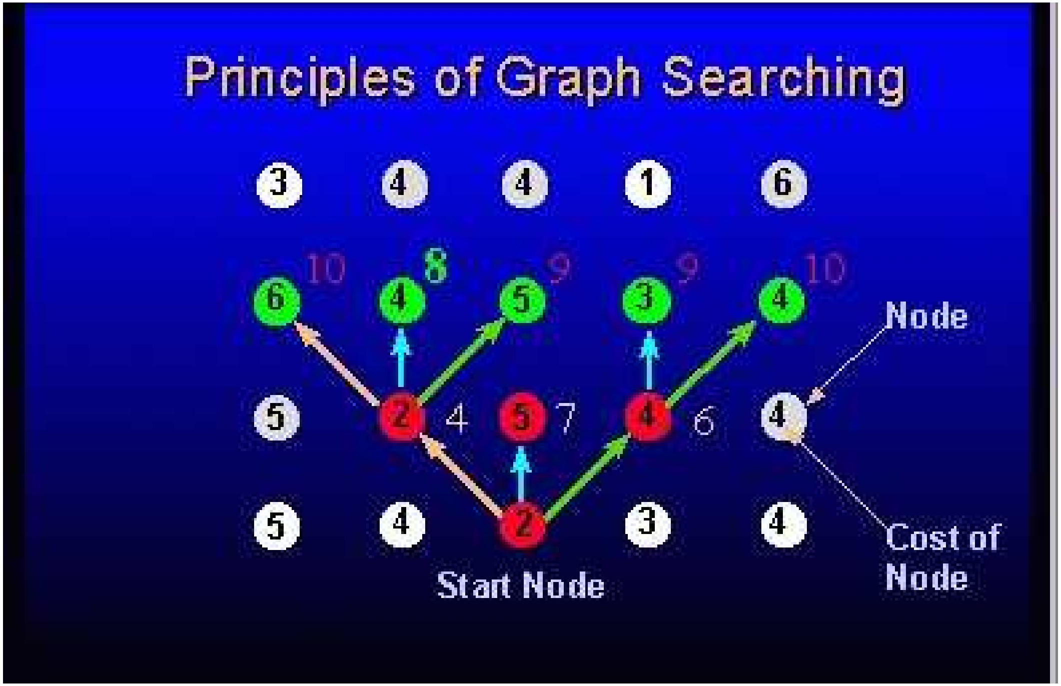
1. *Expand the starting node n_A and put all its successors into an OPEN list with pointers back to the starting node n_A. Evaluate the cost function for expanded node;*
2. *If the OPEN list is empty, fail; Determine the node n_i from the OPEN list with the lowest associated cost $f(n_i)$ and remove it. If $n_i = n_B$, then track back through the pointers to find the optimum path and stop;*
3. *If the option to stop was not taken in step 2, expand the specified node n_i, and put its successors on the OPEN list with pointers back to n_i. Compute their costs f. Go to step 2.*

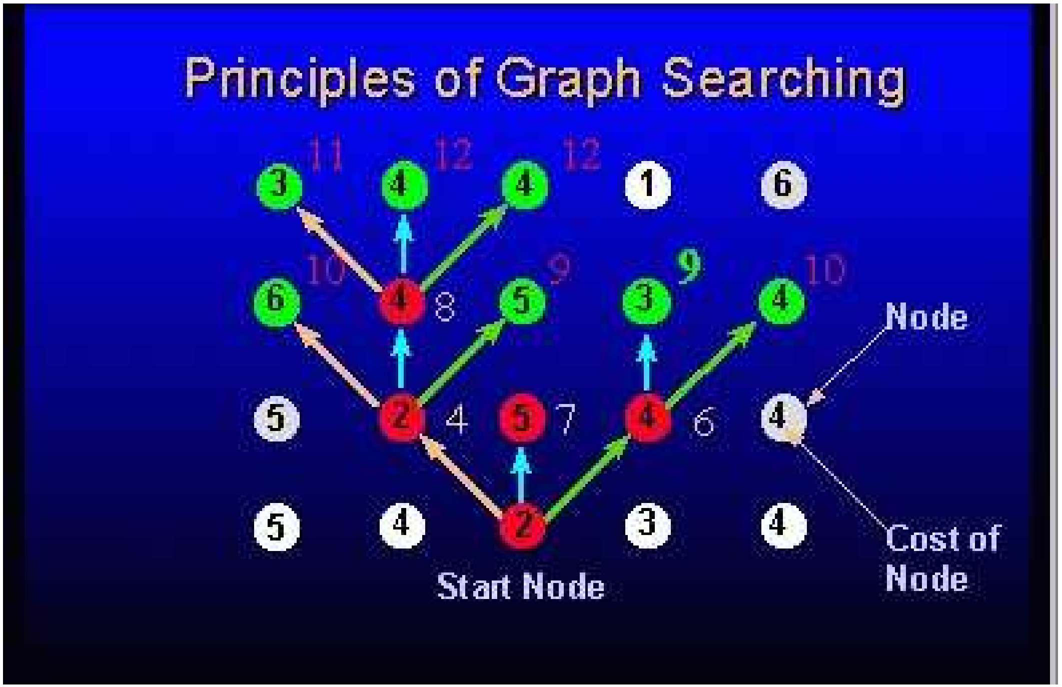
Example 5.2.7. *Example of a graph searching using the A-algorithm, based on (weighted) path length.*

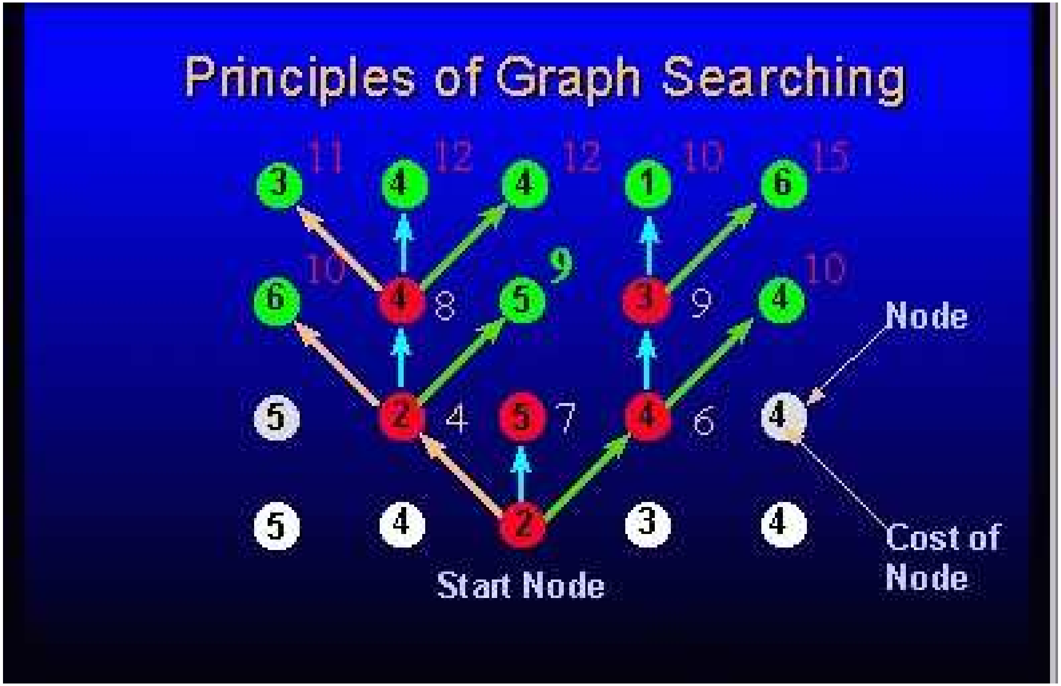

Principles of Graph Searching
3 4 4 1 6
6 4 5 3 4
5 2 5 4 4
4 7 6
5 4 2 3 4
Node
Cost of Node
Start Node

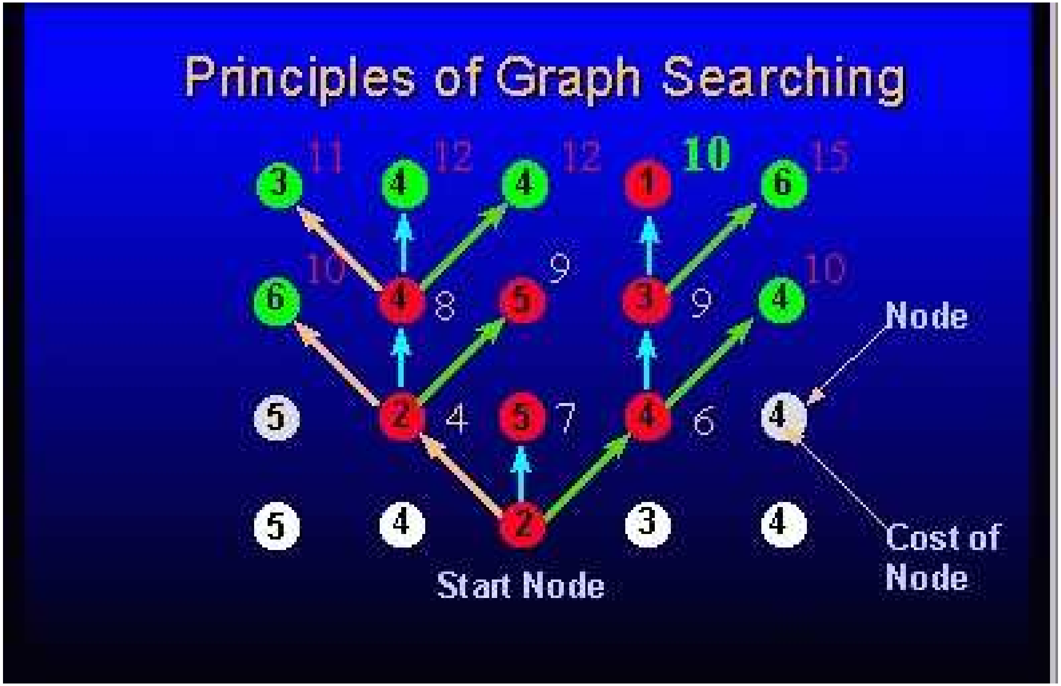

Principles of Graph Searching
3
4
4
1
6
10
8
9
6
4
5
3
4
Node
4
7
6
5
2
5
4
4
5
4
2
3
4
Start Node
Cost of Node

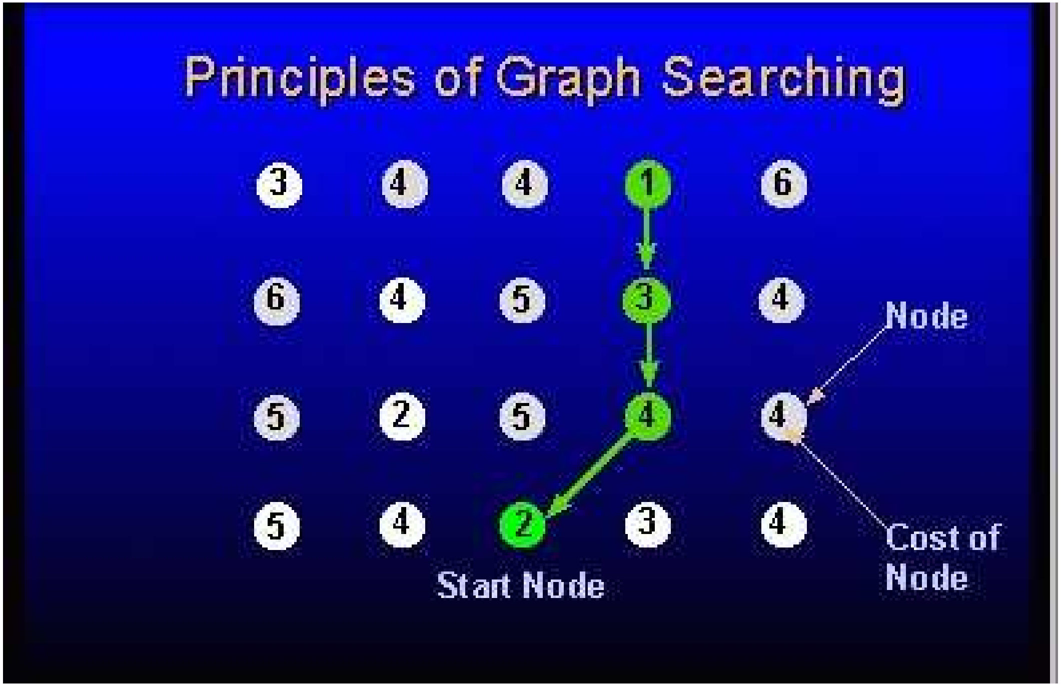

Principles of Graph Searching
Node
Cost of Node
Start Node


Principles of Graph Searching
3
4
4
1
6
10
8
9
9
10
6
4
5
3
4
Node
5
2
4
5
7
4
6
4
5
4
2
3
4
Cost of Node
Start Node


Principles of Graph Searching
11
12
12
3
4
4
1
6
10
9
9
10
6
4
8
5
3
4
Node
5
2
4
5
7
4
6
4
5
4
2
3
4
Start Node
Cost of Node

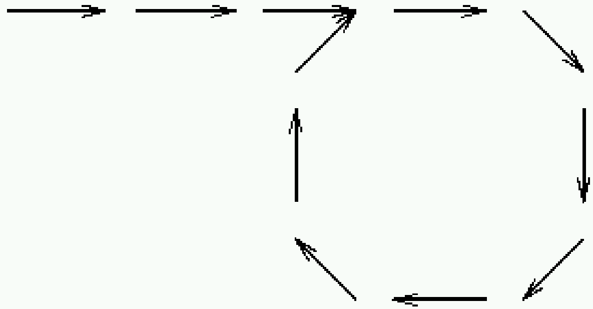

Principles of Graph Searching
11
12
12
10
15
3
4
4
1
6
10
9
10
6
4
8
5
3
9
4
Node
5
2
4
5
7
4
6
4
5
4
2
3
4
Start Node
Cost of Node


Principles of Graph Searching
11
12
12
10
15
10
9
8
10
9
Node
4
7
6
Start Node
Cost of Node



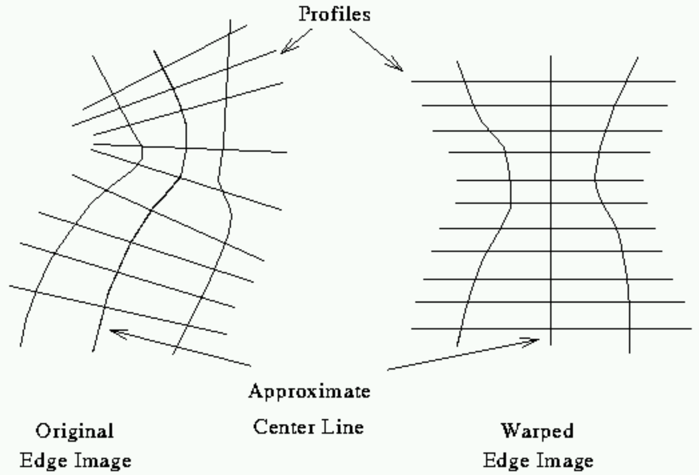
Principles of Graph Searching

- If no additional requirements are set on the graph construction and search, this process can easily result in an infinite loop.



Example of following a closed loop in image data.

- To prevent this behavior, no node expansion is allowed that puts a node on the OPEN list if this node has already been visited, and put on the OPEN list in the past.
- On the other hand, prohibiting the backward search may limit the shapes of borders that can be successfully identified.
- A simple solution is not to allow searching in a backward direction.
- It may be possible to straighten the processed image (and the *graph*).



Geometric warping produces a straightened image: The graph constructed requires (and allows) searches in one main direction only (e.g. top-down).

- The estimate of the cost of the path from the current node n_i to the end node n_B has a substantial influence on the search behavior.
- If this estimate $\tilde{h}(n_i)$ of the true cost $h(n_i)$ is not considered, so $\tilde{h}(n_i) = 0$, no heuristic is included in the algorithm and a breadth-first search is done.
- Hence, the detected path will always be optimal according to the criterion used, the minimum cost path will always be found.
- However, applying heuristics, the detected cost does not always have to be optimal but the search can often be much faster.

☞

- The closer the estimate $\tilde{h}(n_i)$ is to $h(n_i)$, the lower the number of nodes expanded in the search.
- The problem is that the exact cost of the path from the node n_i to the end node n_B is not known beforehand.
- In some applications, it may be more important to get a quick rather than an optimal solution.
- Optimality is not guaranteed but the number of expanded nodes will typically be smaller because the search can be stopped before the optimum is found.

- A crucial question is how to choose the evaluation cost functions for graph-search border detection.
- A good cost function should have elements common to most edge detection problems and also specific terms related to the particular application.
- Some generally applicable cost functions are:

  1. Strength of edges forming a border:

     The heuristic "the stronger the edges that form the border, the higher the probability of the border" is very natural and almost always gives good results.

     If a border consists of strong edges, the cost of that border is small.

     The cost of adding another node to the border will be

$$\left(\max_{x_k \in \text{image}} s(x_k) \right) - s(x_i). \tag{5.33}$$

2. Border curvature:

   Sometimes, borders with a small curvature are preferred.

   If this is the case, the total border curvature can be evaluated as monotonic function of local curvature increments:

   $$\operatorname{diff}\left[\phi(x_i) - \phi(x_j)\right], \tag{5.34}$$

   where diff is some suitable function evaluating the difference in edge directions in two consecutive border elements.

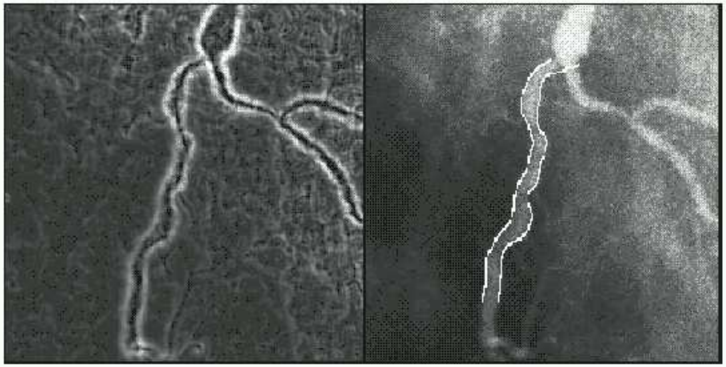
3. Estimates of the distance to the goal (end point):

   If a border is reasonably straight, it is natural to support expansion of those nodes that are located closer to the goal node than other nodes:

   $$\tilde{h}(x_i) = \operatorname{dist}(x_i, x_B). \tag{5.35}$$

- Since the range of border detection applications is quite wide, the cost functions described may need some modification to be relevant to a particular task.

Example 5.2.8. *Border detection by graph searching.*



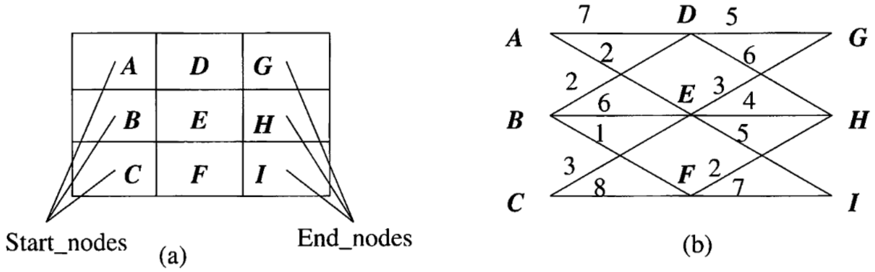
Graph search applied to coronary vessel border detection:
(a) Edge image (b) determined vessel borders.

- Graph searching techniques offer a convenient way to ensure global optimality of the detected contour.
- Searching for all the borders in the image without knowledge of the start and end-points is more complex.
- A good overview of recent detection and edge linking methods can be found in [van der Heijden, 1995].

5.2.5 Border detection as dynamic programming

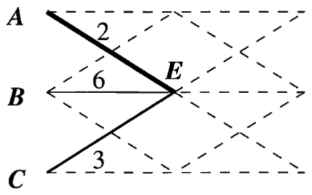
- Dynamic Programming is an optimization method, which searches for optima of functions in which not all variables are simultaneously interrelated.

- Consider the following simple boundary-tracing problem,

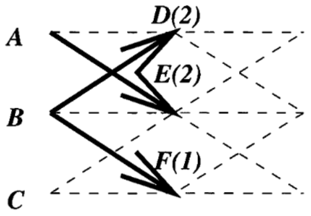


- The aim is to find the best path (minimum cost) between one of the possible start points A, B, C and one of the possible ending points G, H, I. The boundary must be contiguous in 8-connectivity. The graph representing the problem, together with assigned partial cost is shown.

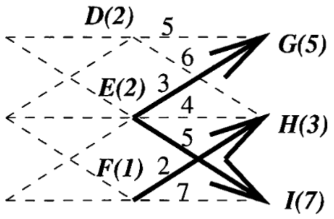
- The main idea of the principle of optimality is: *Whatever the path to the node E was, there exists an optimal path between E and the end point*. In other words, *if the optimal path [start point-endpoint] goes through E, then both its parts [start point-E] and [E-end point] are also optimal, respectively*.
- Therefore, the optimal path can be found by following only the partial optimal path, sucessively layer-by-layer.



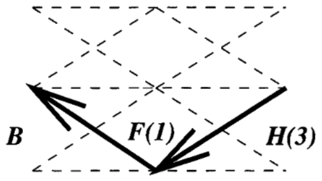
(c)



(d)



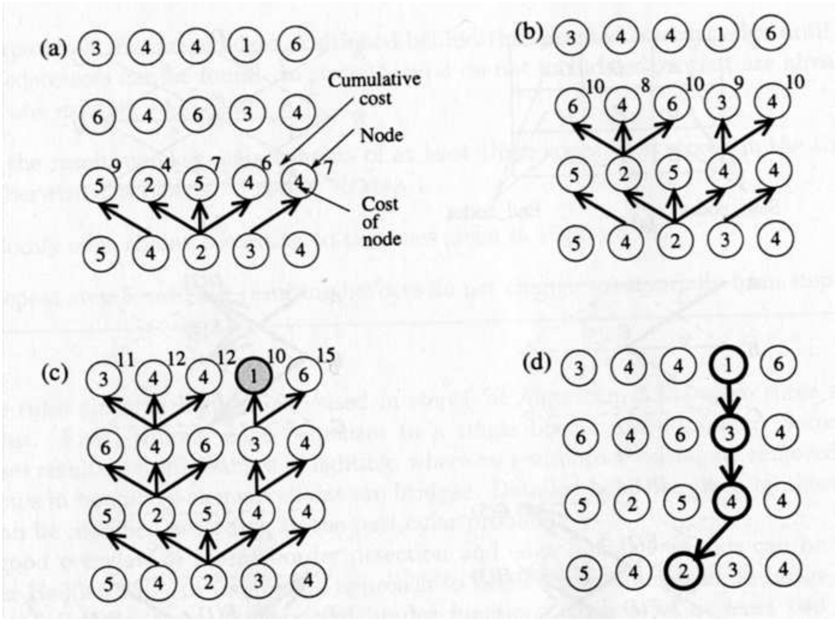
(e)



(f)

- The end point with the minimum path cost represents the optimum path; node H is therefore the optimal boundary end point, and the optimal boundary is $B - F - H$ (Fig. (f)).

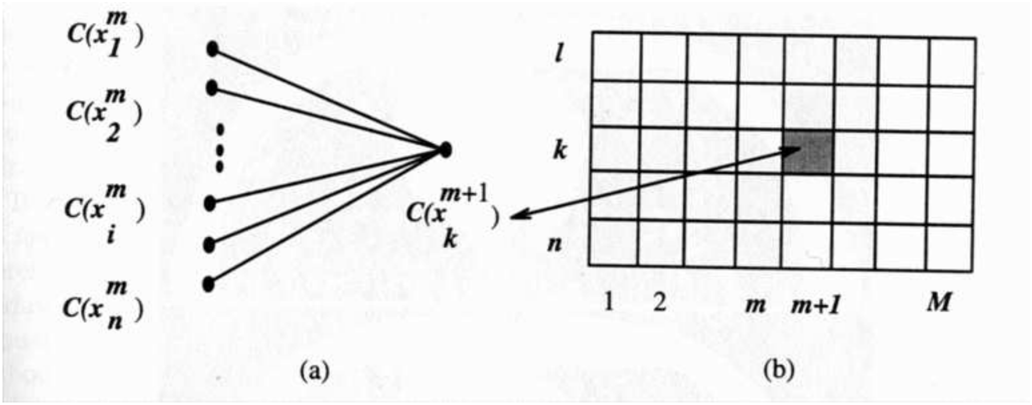
- The following is an example in which node costs are used (not arc cost as in the above example). Note that the graph, the cost function and resulting path are identical to those used in the previous section.



- If the graph has more layers, the process is repeated until one of the end points is reached. Each repetition consists of a simpler optimization as shown in the following figure:

$$C(x_k^{m+1}) = \min_i \left[C(x_i^m) + g^m(i,k) \right], \tag{5.36}$$

where $C(x_k^{m+1})$ is the new cost assigned to the node x_k^{m+1}, and $g^m(i,k)$ is the partial path cost between nodes x_i^m and x_k^{m+1}.



- For the complete optimization problem,

$$\min\left[C(x^1, x^2, \cdots, x^M)\right] = \min_k \left[C(x_k^M)\right], \tag{5.37}$$

  where

  - x_k^M are the end point nodes,
  - M is the number of graph layers between start points and end points,
  - $C(x^1, x^2, \cdots, x^M)$ denotes the cost of a path between the first and the last (M^{th}) graph layer.

- The final optimal path results from back-tacking through the searched graph.

- Requiring an 8-connected border and assuming n nodes in each graph layer m,
  - $3n$ cost combinations must be computed for each layer,
  - $3n(M-1)+n$ is the total number of cost combination computations.
- Compared to the brute-force enumerative search, where $n3^{M-1}$ combinations must be computed, the improvement is obvious.

- Note that the number of neighbors depends on the definition of contiguity and definition of the searched graph and is not limited to three.

Algorithm 5.2.9. *Boundary tracing as dynamic programming*

1. *Specify initial costs* $C(x_i^1)$ *in the first graph layer,* $i = 1, \cdots, n$ *and partial path costs* $g^m(i, k)$*,* $m = 1, \cdots, M - 1$*.*

2. *Repeat step 3 for all* $m = 1, \cdots, M - 1$*.*

3. *Repeat step 4 for all nodes* $k = 1, \cdots, n$ *in the graph layer* m*.*

4. *Let*
$$C(x_k^{m+1}) = \min_{i=-1,0,1} \left[C(x_{k+i}^m) + g^m(i, k) \right] . \tag{5.38}$$
Set pointer from node x_k^{m+1} *back to node* x_i^{m*} *where* $*$ *denotes the optimal predecessor.*

5. *Find an optimal node* x_i^{M*} *in the last graph layer* M *and obtain and optimal path by back-tracking through the pointers from* x_i^{M*} *to* x_i^{1*}*.*

- It has been shown that heuristic search may be more efficient than dynamic programming for finding a path between two nodes in a graph.
- Further, an A-algorithm-based graph search does not require explicit definition of the graph (only the rule for node expansion).
- However, dynamic programming presents an efficient way of simultaneously searching for optimal paths from multiple starting and ending points.
- If these points are not known, dynamic programming is probably a better choice, especially if computation of the partial costs $g^m(i,k)$ is simple.
- Nevertheless, which approach is more efficient for a particular problem depends on evaluation functions and on the quality of heuristics for an A-algorithm.
- Please refer to the text book for further discussions.

- Tracing borders of elongated objects such as roads and rivers in aerial photographs and vessels in medical images, represents typical applications of dynamic programming in image segmentation.

Example 5.2.10. *"Find a minimum-cost connected path between the first and last row of an image matrix. A connected path is only allowed to move by at most one pixel horizontally for each vertical step. The advantage of dynamic programming is that the algorithm is fast and exact (not heuristic). While the formulation may seem restrictive, a surprisingly large number of segmentation and boundary finding problems may be coerced into it." (from dpboundary.m)*

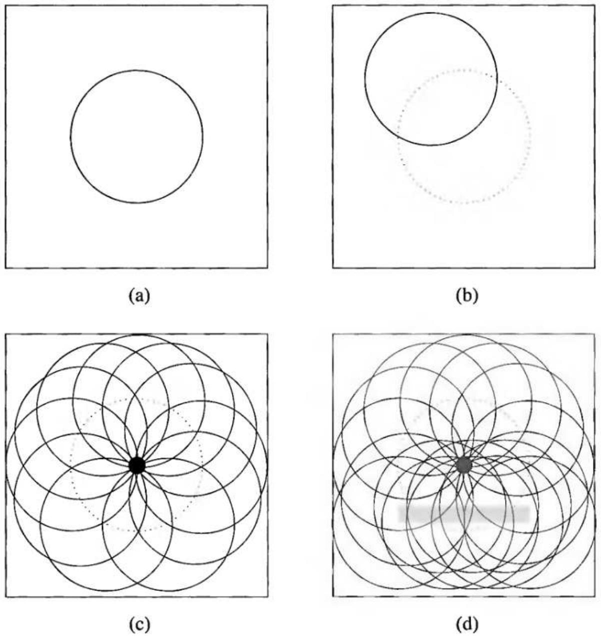
"We demonstrate dpboundary on the task of tracing a blood vessel on part of an MRI image of a lower limb (leg). Since the vessel is bright, we take a negative value of the brightness as the cost function." (from dpboundary_demo.m)

The matlab script from **visionbook** *is here Border detection by dynamic programming.*

5.2.6 Hough transform

- If an image consists of objects with known shape and size. segmentation can be viewed as a problem of finding this object within an image.
- Typical tasks are to locate circular pads in printed circuit boards. or to find objects of specific shapes in aerial or satellite data, etc.
- One of many possible ways to solve these problems is to move a mask with an appropriate shape and size along the image and look for correlation between the image and the mask.
- Unfortunately, the specified mask often differs too much from the object's representation in the processed data, because of shape distortions, rotation, zoom, etc.
- One very effective method that can solve this problem is the *Hough transform*, which can even be used successfully in segmentation of overlapping or semi-occluded objects.

- To introduce the main concepts of the Hough transform, consider an example of circle detection.



(a) (b) (c) (d)

- Let the task be to detect a dark circle of a known radius r in an image with a uniform bright background.
- The method starts with a search for dark image pixels; after such a pixel is found, a locus of potential center points of the circle associated with it can be determined.
- Such a locus of potential center points forms a circle with the radius r as demonstrated in the figure (b).
- If the loci of potential circle centers are constructed for all dark pixels identified in the original image, the frequency can be determined with which each pixel of the image occurs as an element of the circle-center loci.
- As seen from the figure (c), the true center of the circle being sought is represented by the pixel with the highest frequency of occurrence in the circle-center loci.
- Thus, the center of the searched circle is determined.
- With the known circle radius, the image segmentation is completed.

- The original Hough transform was designed to detect straight lines and curves and this original method can be used if analytic equations of object borderlines are known — no prior knowledge of region position is necessary.
- An advantage of this approach is its robustness of segmentation results; that is, segmentation is not too sensitive to imperfect data or noise.
- It is often impossible to get analytic expressions describing border.
- There is a generalized Hough transform designed to find objects even if an analytic expression of the border is not known.

- The basic idea of the method can be seen from the simple problem of detecting a straight line in an image.
- A straight line is defined by two points $A = (x_1, y_1)$ and $B = (x_2, y_2)$, shown in the following figure.

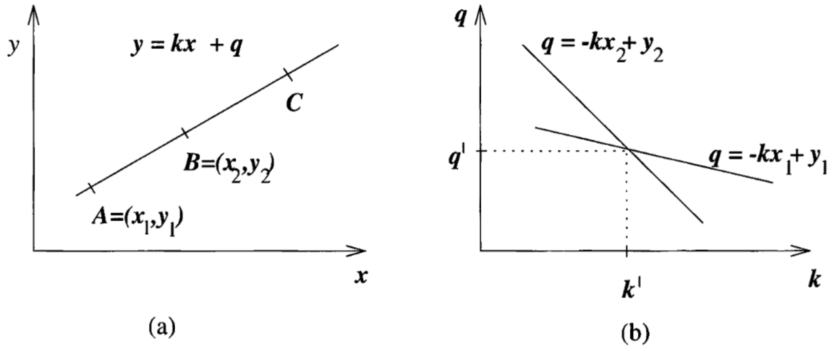


Figure 5.2: Hough transform principle: (a) image space; (b) k, q parameter space.

- All straight lines going through the point A are given by the expression $y_1 = kx_1 + q$ for some values of k and q.
- This means that the same equation can be interpreted as an equation in the parameter space k, q.
- All the straight lines going through the point A are then represented by the equation
$$q = -kx_1 + y_1 \tag{5.39}$$
in the parameter space k, q.
- Straight lines going through the point B can likewise be represented as $q = -kx_2 + y_2$.
- The only common point of both straight lines in the k, q parameter space is the point which in the original image space represents the only existing straight line connecting points A and B.

- This means that any straight line in the image is represented by a single point in the k, q parameter space and any part of this straight line is transformed into the same point.
- The main idea of this method is to determine all the possible line pixels in the image, by transforming all lines that can go through the pixels into corresponding points in the parameter space and detecting the points k, q in the parameter space that frequently resulted from the Hough transform of lines, $q = -kx_2 + y_2$, in the image.
- The main steps will be descrined in details in the following.

- Detection of all possible line pixels in the image may be achieved by applying an edge detector to the image; then, all pixels with edge magnitude exceeding some threshold can be considered possible line pixels (referred to as edge pixels below).

- In the general case, nothing is known about lines in the image and therefore lines of any direction may go through any of the edge pixels.

- In reality, the number of these lines is infinite; however, for practical purposes, only a limited number of line directions may be considered.
- The possible directions of lines define a discretization of the parameter k.
- Similarly, the parameter q is sampled into a limited number of values.
- The parameter space is not continuous any more, but rather is represented by a rectangular structure of cells.
- This array of cells is called the `accumulator array` A, whose elements are accumulator cells $A(k, q)$.

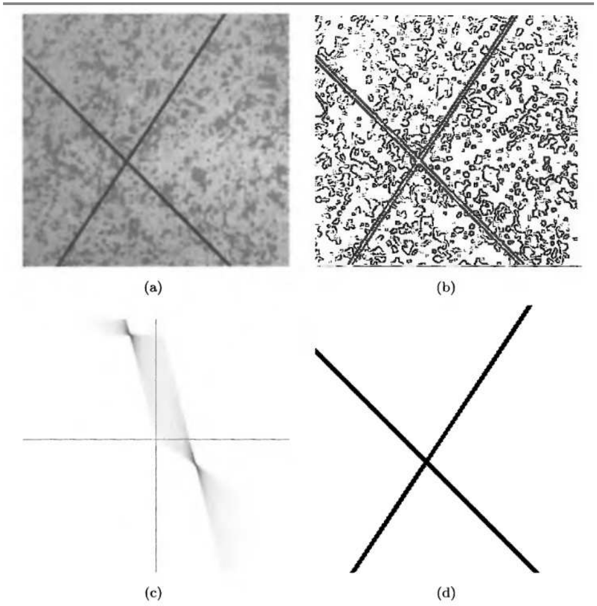
- For each edge pixel, parameters k, q are determined which represent lines of allowed direction going through this pixel.
- For each such line, the values of line parameters k, q are used to increase the value of the accumulator cell $A(k, q)$.
- If a line represented by an equation $y = kx + q$ is present in the image, the value of the accumulator cell $A(k, q)$ will be increased many times — as many

times as the line $y = kx + q$ us detected as a line possibly going through any of the edge pixels.

- For any pixel P, lines going through it may have any direction k (from the set of allowed directions), but the second parameter q is constrained by the image co-ordinates of the pixel P and the direction k.
- Therefore, lines existing in the image will cause large values of the appropriate accumulator cells in the image, while other possibly going through edge pixels, which do not correspond to lines existing in the image, have different k, q parameters for each edge pixel, and therefore the corresponding accumulator cells are increased rarely.
- In other words, lines existing in the image may be detected as high-valued accumulator cells in the accumulator array, and the parameters of the detected line are specified by the accumulator array co-ordinates.
- As a result, line detection in the image is transformed to detection of local maxima in the accumulator space.

- It has been noted that an important property of the Hough transform is its insensitivity to mission part of lines, to image noise, and to other non-line structures co-existing in the image.
- This is caused by the robustness of transformation from the image space to the accumulator space — a missing part of the line will cause only a lower local maximum because a smaller number of edge pixels contributes to the corresponding accumulator cell.

- Insensitivity to data imprecision and noise can be seen in the following figure.



- Note that the parametric equation of the line $y = kx + q$ is appropriate only for explanation of the Hough transform principles — it causes difficulties in vertical line detection ($k \to \infty$) and in nonlinear discretization of the parameter k.

- If a line is represented as

$$x \cos\theta + y \sin\theta = s, \tag{5.40}$$

  the Hough transform does not suffer from those limitations. All the properties of the Hough transform still hold for this parametrization.

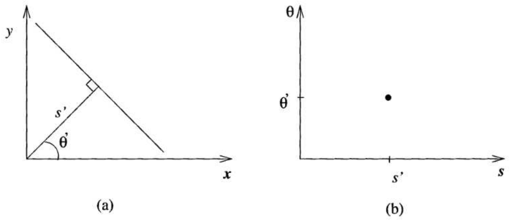
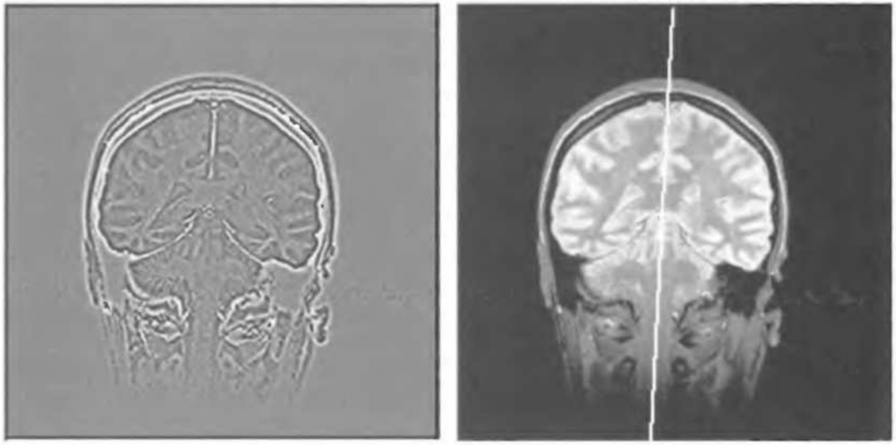


Figure 5.3: Hough transform in s, θ space. (a) Straightline in image space. (b) s, θ parameter space.

- A practical example showing the segmentation of an MR image of the brain into the left and right hemispheres is given in the following figure.



(a) (b)

- Discretization of the parameter space is an important part of this approach; also detecting the local maxima in the accumulator array is a non-trivial problem.
- In reality, the resulting discrete parameter space usually has more than one local maximum per line existing in the image, and smoothing the discrete parameter space may be a solution.
- The above remarks remain valid if more complex curves are sought in the image using the Hough transform, the only difference being the dimensionality of the accumulator array.

- Generalization to more complex curves that can be described by an analytic equation is straightforward.
- Consider an arbitrary curve represented by an equation

$$f(x, a) = 0, \tag{5.41}$$

where a is the vector of curve parameters.

Algorithm 5.2.11. *Curve detection using the Hough transform*

1. *Quantize parameter space within the limits of parameters a. The dimensionality n of the parameter space is given by the number of parameters of the vector a.*

2. *Form an n-dimensional accumulator array $A(a)$ with structure matching the quantization of the parameter space; set all elements to zero.*

3. *For each image point (x_1, x_2) with non-zero pixel value in the appropriately thresholded gradient image, increase all accumulator cells $A(a)$ if $f(x, a) = 0$,*

$$A(a) = A(a) + \Delta A \tag{5.42}$$

   for all a inside the limits used in step 1.

4. *Local maxima in the accumulator array $A(a)$ correspond to realizations of curves $f(x, a)$ that are present in the original image.*

- If we are looking for circles, the analytic expression $f(x, a)$ of the desired curve is
$$(x_1 - a)^2 + (x_2 - b)^2 = r^2 \tag{5.43}$$
where the circle has center (a, b) and radius r.

- Therefore, the accumulator data structure must be 3-dimensional.

- For each pixel x whose edge magnitude exceeds a given threshold, all accumulator cells corresponding to potential circle centers (a, b) are incremented in step 3 of the given algorithm.

- The accumulator cell $A(a, b, r)$ is incremented if the point (a, b) is at distance r from point x.

- If some potential center (a, b) of a circle of radius r is frequently found in the parameter space, it is highly probable that a circle with center (a, b) and radius r really exist in the processed data.

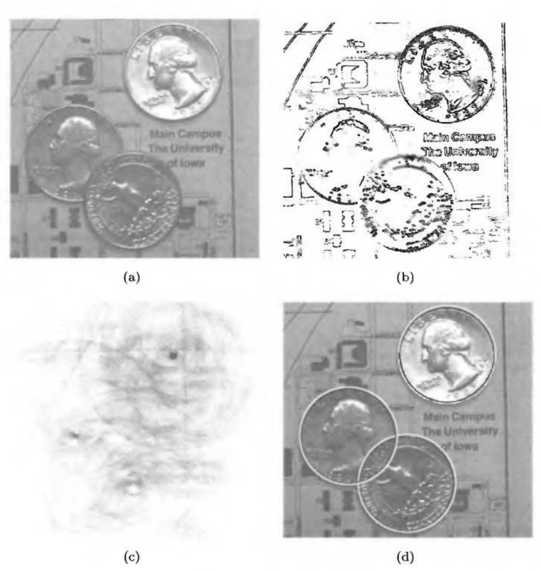


Figure 5.4: Hough transform — circle detection: (a) original image; (b) edge image (note that the edge informaiton is far from perfect; (c) parameter space; (d) detected circles.

- The processing results in a set of parameters of desired curves $f(x, a) = 0$ that correspond to local maxima of accumulator cells in the parameter space; these maxima best match the desired curves and processed data.
- Parameters may represent unbounded analytic curves (e.g., line, parabola, etc.), but to look for finite parts of these curves, the end points must be explicitly defined and other conditions must be incorporated into the algorithm.
- Even though the Hough transform is a very powerful technique for curve detection, exponential growth of the accumulator data structure with the increase of the number of curve parameters restricts its practical usability to curves with few parameters.

- If prior information about edge directions is used, computational demands can be decreased significantly.
- Please refer to the textbook for discussions.

- The randomized Hough transform offers a different approach to achieve increased efficiency.
- It randomly selects n pixels from the edge image and determines n parameters of the detected curve followed by incrementing a single accumulator cell only.
- Please refer to the textbook for discussions.

Example 5.2.12. *Line detection in chessboard.The matlab script from* **visionbook** *is here Line detection example.*

5.3 Region-based segmentation

- The aim of segmentation methods described in the previous section was to find borders between regions; the following methods construct regions directly.
- It is easy to construct regions from their borders, and it is easy to detect borders of existing regions.
- However, segmentations resulting from edge-based methods and region-growing methods are not usually exactly the same, and a combination of results may often be a good idea.
- Region growing techniques are generally better in noisy images, where borders are extremely difficult to detect.
- Homogeneity is an important property of regions and is used as the main segmentation criterion in region growing, whose basic idea is to divide an image into zones of maximum homogeneity.

- The homogeneity criteria affect the region-based methods.
- General and specific heuristics for homogeneity may also be incorporated.
- The criteria for homogeneity can be based on gray-level, color, texture, shape, model (using semantic information), etc.
- The simplest homogeneity criterion uses an average gray-level of the region, its color properties, simple texture properties, or an m-dimensional vector of average gray values for multi-spectral images.
- Properties chosen to describe regions influence the form, complexity, and amount of prior information in the specific region-growing segmentation method.
- Methods that specifically address region-growing segmentation of color images are reported in the literature.

- Regions have already been defined before.
- A complete segmentation of an image R is a finite set of regions $R_1, \cdots, R_S$,

$$R = \bigcup_{i=1}^{S} R_i, \qquad R_i \bigcap R_j = \emptyset \quad \text{if } i \neq j \tag{5.44}$$

  where S is the total number of regions in the image.

- Further assumptions needed in this section are that regions must satisfy the following conditions:

$$H(R_i) = \text{TRUE}, \quad \text{for } i = 1, \cdots, S \tag{5.45}$$

  and

$$H(R_i \bigcup R_j) = \text{FALSE}, \quad \text{for } i \neq j \text{ and } R_i \text{ is adjacent to } R_j. \tag{5.46}$$

  where $H(R_i)$ is a binary homogeneity evaluation of the region R_i.

- Resulting regions of the segmented image must be both homogeneous and maximal, where by 'maximal' we mean that the homogeneity criterion would not be true after merging a region with any adjacent region.

- While the region growing methods discussed below deal with two-dimensional images, three-dimensional implementations are often possible.
- Considering three-dimensional connectivity constraints, homogeneous regions (volumes) of a three-dimensional image can be determined using three-dimensional region growing.

5.3.1 Region merging

- The most natural method of region growing is to begin the growth in the raw image data, each pixel representing a single region.
- These regions almost certainly *do not* satisfy the condition of equation (5.46), and so regions will be merged as long as equation (5.45) remains satisfied.

Algorithm 5.3.1. *Region merging (outline)*

1. *Define some starting method to segment the image into many small regions satisfying condition (5.45).*
2. *Define a criterion for merging two adjacent regions.*
3. *Merge all adjacent regions satisfying the merging criterion. If no two regions can be merged maintaining condition (5.45), stop.*

- This algorithm represents a general approach to region merging segmentation.
- Specific methods differ in the definition of the starting segmentation and in the criterion for merging.
- In the descriptions that follow, regions are those parts of the image that can be sequentially merged into larger regions satisfying equation (5.45) and (5.46).

STOP

- The result of region merging usually depends on the order in which regions are merged, meaning that segmentation results will probably differ if segmentation begins, for instance, in the upper left or lower right corner.
- This is because the merging order can cause two similar adjacent regions R_1 and R_2 not to be merged, since an earlier merge used R_1 and its new characteristics no longer allow it to merged with region R_2. If the merging process used a different order, this merge may have been realized.

- The simplest methods begin merging by starting the segmentation using regions of 2×2, 4×4 or 8×8. Region descriptions are then based on their statistical gray-level properties — a region gray-level histogram is a good example.

- A `region description` is compared with the description of an adjacent region; if they match, they are merged into a larger region and a new region description is computed.

- Otherwise, regions are marked as non-matching.

- Merging of adjacent regions continues between all neighbors, including newly formed ones.

- If a region cannot be merged with any of its neighbors, it is marked 'final'.

- The merging process stops when all image regions are so marked.

- Another approach uses the supergrid.
- The `super-grid` carries all the necessary information for easy region merging in 4-adjacency, see the following figure:

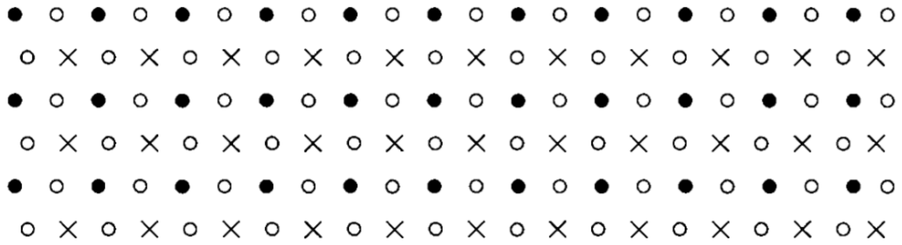


Figure 5.5: Super-grid data structure: $\times$ – image data, $\circ$ – crack edges; $\bullet$ – unused.

- Starting regions are formed by pixels of the same gray-level.
- Region merging is based on the crack edge computation, where local boundaries between regions are evaluated by the strength of crack edges along their common border.

- This region merging uses the following two heuristics:
    1. Two adjacent regions are merged if a significant part of their common boundary consists of weak edges (*significance* can be based on the region with the shorter perimeter; the ratio of the number of weak common edges to the total length of the region perimeter).
    2. Two adjacent region are also merged if a significant part of their common boundary consists of weak edges, but in this case not considering the total length of the region borders.

- Of the two heuristics, the first is more general and the second cannot be used alone because it does not consider the influence of different region sizes.

- Edge significance can be evaluated according to the formula

$$v_{ij} = \begin{cases} 0, & \text{if } s_{ij} < T_1; \\ 1, & \text{otherwise} \end{cases} \tag{5.47}$$

  where $v_{ij} = 1$ indicates a significant edge, and $v_{ij} = 0$ a weak edge, T_1 is a preset threshold, and s_{ij} is the crack edge value (e.g., $s_{ij} = |f(x_i) - f(x_j)|$).

- The algorithm is as follows.

Algorithm 5.3.2. *Region merging via boundary melting*

1. *Define a starting image segmentation into regions of constant gray-level. Construct a super-grid edge data structure in which to store the crack edge information.*

2. *Remove all weak crack edge data structure (using (5.47) and threshold T_1).*

3. *Recursively remove common boundaries of adjacent regions R_i and R_j, if*

$$\frac{W}{\min(L_i, L_j)} \geq T_2 \tag{5.48}$$

   where W is the number of weak edges on the common boundary, L_i and L_j are the perimeter lengths of regions R_i and R_j, and T_2 is another preset threshold.

4. *Recursively remove common boundaries of adjacent regions R_i and R_j, if*

$$\frac{W}{\min(L)} \geq T_3 \tag{5.49}$$

   or, using a weaker criterion

$$W \geq T_3 \tag{5.50}$$

   where L is the length of the common boundary and T_3 is a third threshold.

- Note that even if we have described a region growing method, the merging criterion is based on border properties and so the merging does not necessarily keep condition (5.45).
- Please refer to the textbook for discussions on using a more advanced data structure than the supergrid.

- The following figure gives a comparison of region merging methods.
- The original image is displayed in its pseudo-color representation

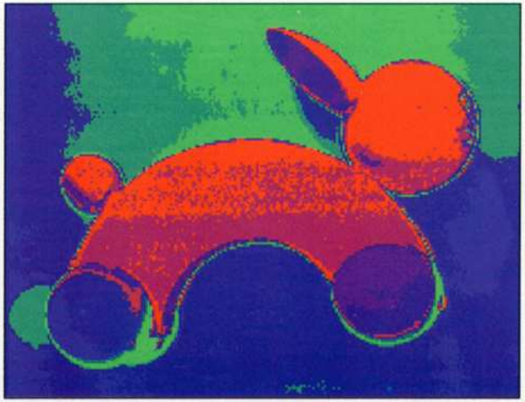


Plate 2 *Pseudocolor representation of the original image.* (Figure 5.38 b)

- The original image cannot be segmented by thresholding because of the significant and continuous gray-level in all regions.

- Results of a recursive region merging method — pixels are merged in the row-first fashion as long as they do not differ by more than a pre-specified parameter from the seed pixel, are shown as follows:

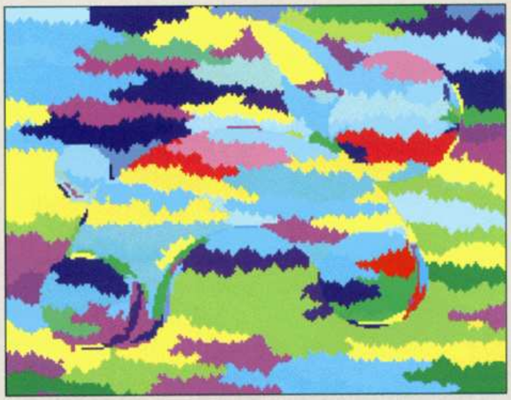


Plate 3 *Recursive region merging.*
(Figure 5.38 c)

- Note the resulting horizontally elongated regions corresponding to vertical changes of image gray-levels.

- If region merging via boundary melting is applied, the result is improved dramatically.

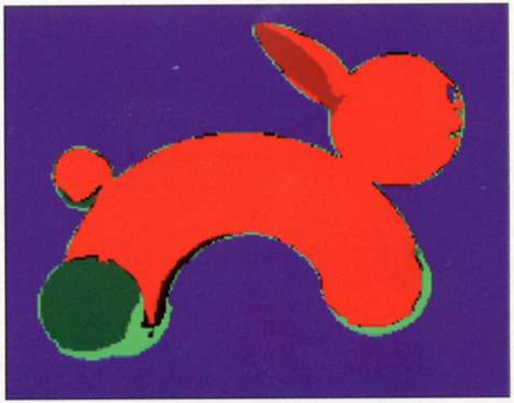


Plate 4 *Region merging via boundary melting.* (Figure 5.38 d)

Example 5.3.3. *Region merging with boundary melting. The matlab script from* **visionbook** *is here Region merging example.*

5.3.2 Region splitting

- Region splitting is the opposite of region merging, and begins with the whole image represented as a single region which does not usually satisfy the condition (5.45).
- Therefore, the existing image region are sequentially split to satisfy (5.44), (5.45) and (5.46).
- It generally uses similar criteria of homogeneity as region merging method and differs only in the direction of their application.
- Even if this approach seems to be dual to region merging, region splitting does not result in the same segmentation even if the same homogeneity criteria are used.
  - Some regions may be homogeneous during the splitting process and therefore are not split any more.
  - Considering the homogeneous regions created by region merging procedures, some may not be constructed because of the impossibility of merging smaller sub-regions earlier in the process.

- A fine black-and-white chessboard is an example: let the homogeneity criterion be based on variance of average gray-levels in the quadrants of the evaluated region in the next lower pyramid level.

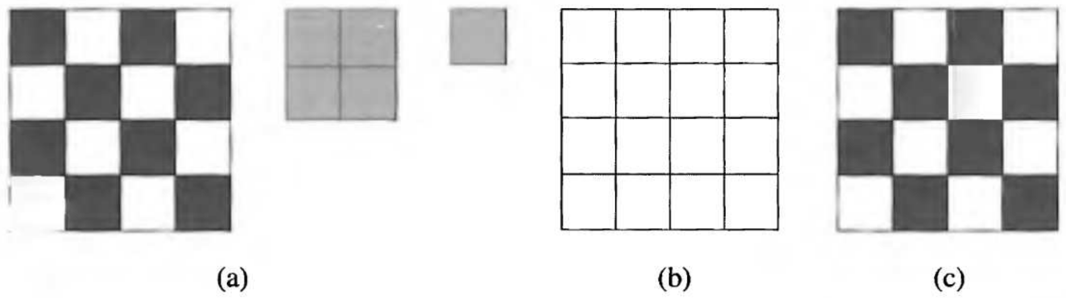
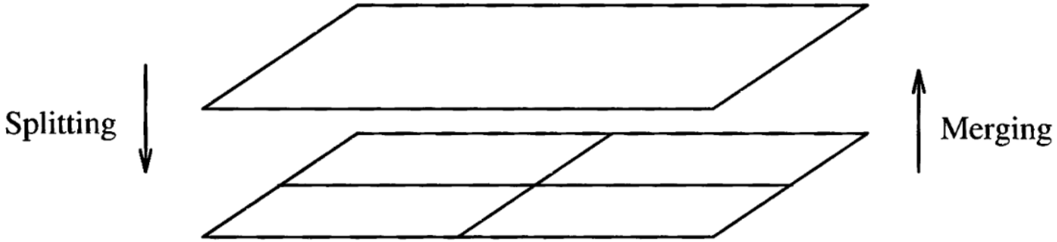


Figure 5.6: Different segmentations may result from region splitting and region merging approaches. (a) Chessboard imaging, corresponding pyramid. (b) Region splitting segmentation (upper pyramid level is homogeneous, no splitting possible). (c) Region merging segmentation (lowest pyramid level consists of regions that cannot be merged).

5.3.3 Splitting and merging

- A combination of splitting and merging may result in a method with the advantage of both approaches.

- Split-and-merge approaches work using pyramid image representations. Regions are square shaped and corresponding to elements of the appropriate pyramid level.



- If any region in any pyramid level is not homogeneous (excluding the lowest level), it is split into four sub-regions — these are elements of higher resolution at the level below.

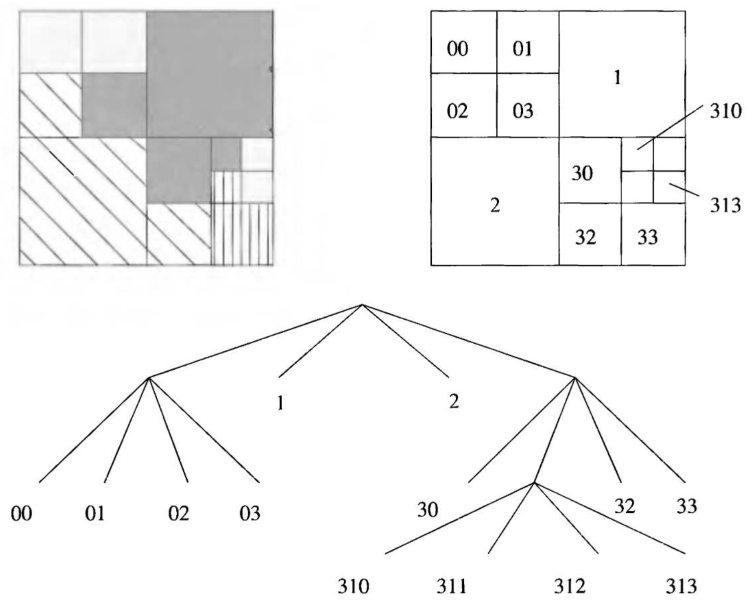
- If four regions with the same parent node exists at any pyramid level with approximately the same value of homogeneity measure, they are merged into a single region in an upper pyramid level.

- The segmentation process can be understood as the construction of a segmentation quadtree where each leaf node represents a homogeneous region — that is, an element of some pyramid level.

- Splitting and merging corresponds to removing or building parts of the segmentation quadtree — the number of leaf nodes of the tree corresponds to the number of segmented regions after the segmentation process is over.

- These approaches are sometimes called split-and-link methods if they use segmentation tress for storing information about adjacent regions. Split-and-merge methods usually store the adjacency information in region adjacency graphs (or similar data structures).

- An unpleasant drawback of segmentation quadtree is the square-region shape assumption, see the following figure:



- The homogeneity criterion plays a major role in split-and-merge algorithms, as it does in all other region growing methods.
- If the image being processed is reasonably simple, a split-and-merge approach can be based on local image properties.
- If the image is very complex, even elaborate criteria including semantic information may not give acceptable results.

5.3.4 Watershed segmentation

- The concepts of watersheds and catchment basins are well known in topography.
- Watershed lines divide individual catchment basins.
- The North American Continental Divide is a textbook example of a watershed line with catchment basins formed by the Atlantic and Pacific Oceans.
- Working the 2D function presentations, image data may be interpreted as a topographic surface where the (gradient) image gray-levels represent altitudes.
- Thus, region edges correspond to high watersheds and low-gradient region interiors correspond to catchment basins.
- According to equations (5.44), (5.45) and (5.46), the goal of region growing segmentation is to create homogeneous regions.
- In watershed segmentation, catchment basins of the topographic surface are homogeneous in the sense that all pixels belonging to the same catchment basin are connected with the basin's region of minimum altitude (gray-level) by a simple path of pixels that have monotonically decreasing altitude (gray-level) along the path.

- Such catchment basins then represent the regions of the segmented image.

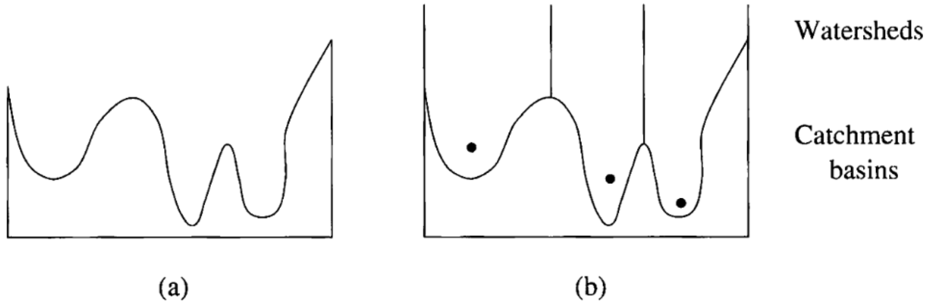


Figure 5.7: One dimensional example of watershed segmentation. (a) Gray-level profile of image data. (b) Watershed segementation — local minima of gray-level (altitude) yield catchment basins, local maxima define the watershed lines.

- While the concept of watersheds and catchment basins is quite straightforward, development of algorithms for watershed segmentation is a complex task, with many of the early methods resulting in either slow or inaccurate execution.

- There are two basic approaches to watershed image segmentation.
- The first one starts with finding a downstream path from each pixel of the image to a local minimum of image surface altitude.
- A catchment basin is then defined as the set of pixels for which their respective downstream paths all end up in the same altitude minimum.
- While the downstream paths are easy to determine for continuous altitude surfaces by calculating the local gradients, no rules exist to define the downstream paths uniquely for digital surfaces.

- The second watershed segmentation approach represented by a seminal paper [Vincent and Soille, 1992] makes the idea practical.
- This approach is essentially dual to the first one.
- Instead of identifying the downstream paths, the catchment basins fill from the bottom.
- As was explained earlier, each minimum represents one catchment basin, and the strategy is to start at the altitude minima.
- Imagine that there is a hole in each local minimum, and that the topographic surface is immersed on water.
- As a result, the water starts filling all catchment basins, minima of which are under the water level.
- If two catchment basins would merge as a result of further immersion, a dam is built all the way to the highest surface altitude and the dam represents the watershed line.
- An efficient algorithm for such watershed segmentation was presented in [Vincent and Soille, 1992].

- The algorithm is based on *sorting* the pixels in an increasing order of their gray values, followed by a *flooding* step consisting of a fast breadth-first scanning of all pixels in the order of their gray-levels.

- During the sorting step, a brightness histogram is computed.
- Simultaneously, a list of pointers to pixels of gray-level h is created and associated with each histogram gray-level to enable direct access to all pixels of any gray-level.
- Information about the image pixel sorting is used extensively in the flooding step.

- Suppose the flooding has been completed up to a level (gray-level, altitude) k.

  Then every pixel having gray-level less than or equal to k has already been assigned a unique catchment basin label.

- Next, pixels having gray-level $k+1$ will be processed; all such pixels can be found in the list prepared in the sorting step — all these pixels can be accessed directly.
- A pixel having gary-level $k+1$ may belong to a catchment basin labeled L if at least one of its neighbors already carries this label.
- Pixels that represent potential catchment basin members are put in a first-in first-out queue and await further processing.

- Geodesic influence zones are computed for all hitherto determined catchment basins.

  A geodesic influence zone of a catchment basin L_i is the locus of non-labeled image pixels of gray-level $k+1$ that are contiguous with the catchment basin L_i (contiguous within the region of pixels of gray-level $k+1$) for which their distance to L_i is smaller than their distance to any other catchment basin L_j.

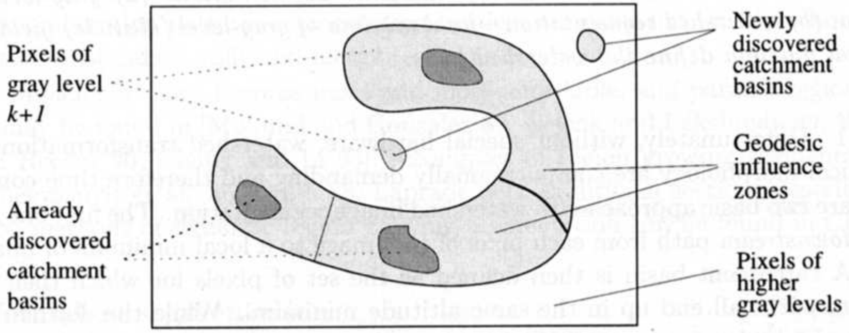

Pixels of gray level k+1
Already discovered catchment basins
Newly discovered catchment basins
Geodesic influence zones
Pixels of higher gray levels

Figure 5.50: *Geodesic influence zones of catchment basins.*

- All pixels with gray-level $k+1$ that belong to the influence zone of a catchment basin labeled L are also labeled with the label L, thus causing the catchment basin to grow.
- The pixels from the queue are processed sequentially, and all pixels from the queue that cannot be assigned an existing label represent newly discovered catchment basins and are marked with new and unique labels.

- The following is an example of watershed segmentation.

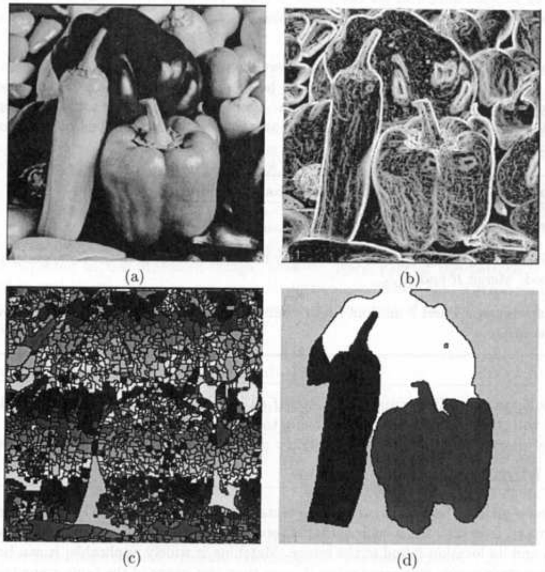


Figure 5.51: *Watershed segmentation: (a) original; (b) gradient image, 3×3 Sobel edge detection, histogram equalized; (c) raw watershed segmentation; (d) watershed segmentation using region markers to control oversegmentation. Courtesy W. Higgins, Penn State University.*

- Note that the raw watershed segmentation produces a severely over-segmented image with hundreds or thousands of catchment basins (c).
- To overcome this problem, region markers and other approaches have been suggested to generate good segmentation (d).

  STOP

- Please refer to the textbook for further discussions.

5.3.5 Region growing post-processing

- Images segmented by region growing methods often contain either too many regions (under-growing)or too few regions (over-growing) as a result of non-optimal parameter setting.
- To improve classification results, a variety of post-processors has been developed. Some of them combine segmentation information obtained from region growing and edge-based segmentation. See the references in the text book.

- Simpler post-processors are based on general heuristics and decrease the number of small regions in the segmented image that cannot be merged with any adjacent region according to the originally applied homogeneity criteria.
- These small regions are usually not significant in further processing and can be considered as segmentation noise. It is possible to remove them from the image as follows.

Algorithm 5.3.4.

1. *Search for the smallest image region R_{min}.*
2. *Find the adjacent region R most similar to R_{min}, according to the homogeneity criteria used. Merge R and R_{min}.*
3. *Repeat steps 1 and 2 until all regions smaller than a pre-selected size are removed from the image.*

- This algorithm will execute much faster if all regions smaller than a pre-selected size are merged with their neighbors without having to order them by size.

Example 5.3.5. *Small region removal remsmall algorithm. The matlab script from* **visionbook** *is here Small region removal example.*

5.4 Active contour models — snakes

- The development of active contour models results from the work [Kass et al., 1988].
- It is a energy-minimization approach to a variety of tasks in image analysis such as image segmentation, and is suitable for analysis of dynamic image data or 3D image data.
- The active contour model, or snake, is defined as an energy-minimizing spline — the snake's energy depends on its shape and location within the image.
- Local minima of this energy then correspond to desired image properties.

- The following figure illustrates how a snake is defomed from its initial position to the target.

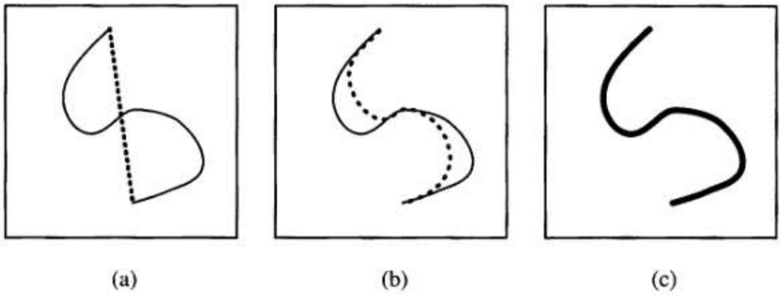


Figure 5.8: Active contour model — snake. (a) Initial snake position (dotted) defined interactively near the true contour. (b), (c) Iteration steps of snake energy minimization: the snake is pulled toward the true contour.

- Snakes may be understood as a special case of a more general technique of matching a deformable model to an image by means of energy minimization.
- Snakes do not solve the entire problem of finding contours in images; rather, they depend on other mechanisms such as interaction with a user, interaction with some higher-level image understanding process, or information from image data adjacent in time or space.
- This interaction must specify an approximate shape and starting position for the snake somewhere near the desired contour.
- A priori as well as image-based information are then used to push the snake toward an appropriate solution.
- Unlike most other image models, the snake is active, always minimizing its energy functional, therefore exhibiting dynamic behavior.

5.4.1 Traditional snakes and balloons

- The energy functional which is minimized is a weighted combination of internal and external forces.
- The internal forces emanate from the shape of the snake, while the external forces come from the image andor from higher-level image understanding processes.
- The snake is defined parametrically as $v(s) = [x(s), y(s)]$, where $x(s)$ and $y(s)$ are (x, y) co-ordinates along the contour and $s \in [0, 1]$.

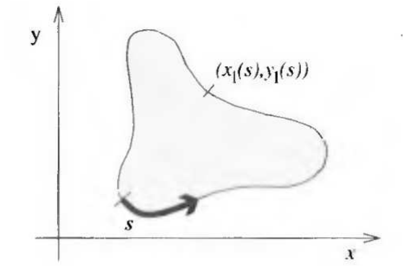


Figure 5.9: Active contour model — a parametric representation of the boundary.

- The energy functional to be minimized may be written as

$$E^*_{\text{snake}} = \int_0^1 E_{\text{snake}}(v(s))\,ds \tag{5.51}$$

$$= \int_0^1 \left\{ E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s)) \right\} ds. \tag{5.52}$$

  where

  – E_{int} represents the internal energy of the spline due to bending,

  – E_{image} denotes image forces,

  – E_{con} external constraint forces.

- Usually, $v(s)$ is approximated as a spline to ensure desirable properties of continuity.

- The internal spline energy can be written as

$$E_{\text{int}}(v(s)) = \alpha(s) \left| \frac{dv}{ds} \right|^2 + \beta(s) \left| \frac{d^2v}{ds^2} \right|^2 \tag{5.53}$$

  where $\alpha(s)$ and $\beta(s)$ specify the elasticity and stiffness of the snake.

- $\alpha(s)$ and $\beta(s)$ control how the snake shrinks.

- Note that setting $\beta(s_k) = 0$ at a point s_k allows the snake to become second-order discontinuous at that point, and develop a corner.

- The image force term E_{image} is derived from the image data over which the snake lies.
- As an example, a weighted combination of three different functionals is presented which attracts the snake to lines, edges, and terminations

$$E_{\text{image}} = w_{\text{line}} E_{\text{line}} + w_{\text{edge}} E_{\text{edge}} + w_{\text{term}} E_{\text{term}}. \tag{5.54}$$

- The line-based functional may be very simple

$$E_{\text{line}} = f(x, y), \tag{5.55}$$

or

$$E_{\text{line}} = G_\sigma * f(x, y), \tag{5.56}$$

where $f(x, y)$ denotes image gray-levels at image location (x, y).

Then depending on the sign of w_{line}, the snake will be attracted to light or dark lines.

- The edge-based functional

$$E_{\text{edge}} = -\left|\nabla f(x,y)\right|^2 , \tag{5.57}$$

or

$$E_{\text{edge}} = -\left|\nabla \left[G_\sigma * f(x,y)\right]\right|^2 , \tag{5.58}$$

attracts the snake to contours with large image gradients — that is, to locations of strong edges, and is minimal at edges if defined as

- In order to find teminations and corners, we use the curvature of level lines in a slightly smoothed image.
- Let $C(x,y) = G_\sigma(x,y) * f(x,y)$ be a slightly smoothed version of the image.
- Then the curvature of the level contours in $C(x,y)$ can be written as

$$E_{\text{term}} = \frac{C_{yy} C_x^2 - 2 C_{xy} C_x C_y + C_{xx} C_y^2}{\sqrt{C_x^2 + C_y^2}^3} . \tag{5.59}$$

- The snake behavior may be controlled by adjusting the weights w_{line}, w_{edge} and w_{term}.
- A snake attracted to edges and terminations is shown in the follwing figure.

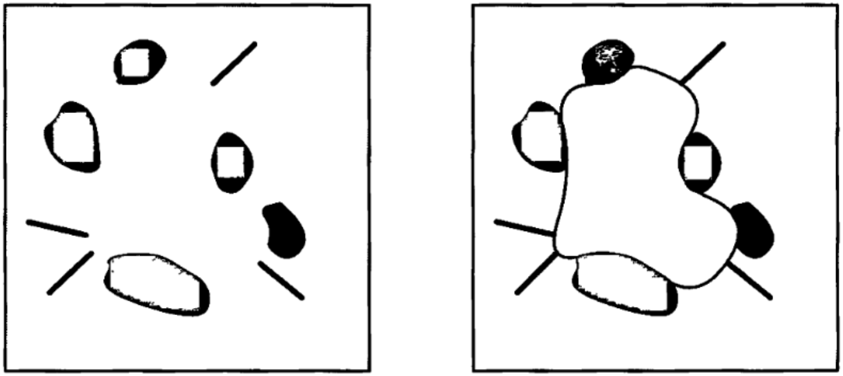


Figure 5.10: A snake attracted to edges arid terminations, (a) Contour illusion, (b) A snake attracted to the subjective contour. Adapted from [Kass et al., 1987].

- The third term of the integral (5.52) comes from external constraints imposed either by a user or some other higher-level process which may force the snake toward or away from particular features.
- If the snake is near to some desirable feature, the energy minimization will pull the snake the rest of the way.
- However, if the snake settles in a local energy minimum that a higher-level process determines as incorrect, an area of energy peak may be made at this location to force the snake away to a different local minimum.

- A contour is defined to lie in the position in which the snake reaches a local energy minimum.

- From (5.52) and the calculus of variations, the Euler-Lagrange condition states that the spline $v(s)$ which minimizes E^*_{snake} must satisfy

$$\frac{\delta E^*_{\text{snake}}}{\delta v} = -\frac{d}{ds}\left(\alpha(s)\frac{dv}{ds}\right) + \frac{d^2}{ds^2}\left(\beta(s)\frac{d^2 v}{ds^2}\right) + \nabla E_{\text{ext}}(v(s)) = 0, \tag{5.60}$$

  where $E_{\text{ext}} = E_{\text{image}} + E_{\text{con}}$.

- To solve the Euler-Lagrange equation, suppose an initial estimate of the solution is available. An evolution equation is formed:

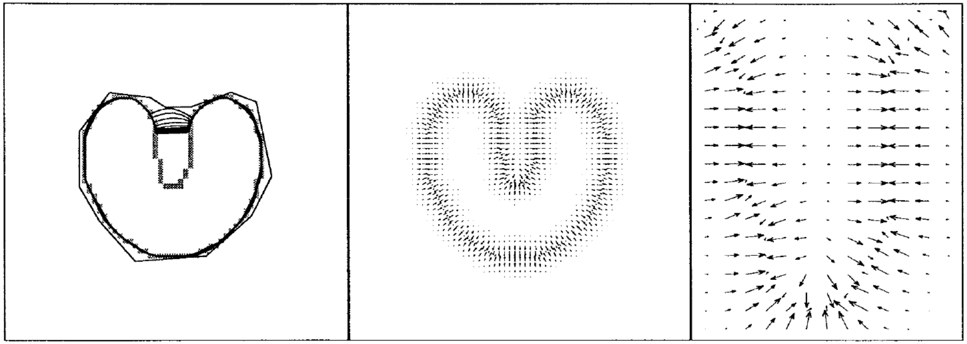
$$\frac{\partial v(s,t)}{\partial t} = -\frac{\delta E^*_{\text{snake}}}{\delta v} = \frac{d}{ds}\left(\alpha(s)\frac{dv}{ds}\right) - \frac{d^2}{ds^2}\left(\beta(s)\frac{d^2 v}{ds^2}\right) - \nabla E_{\text{ext}}(v(s)). \tag{5.61}$$

- The solution is found if $\frac{\partial v(s,t)}{\partial t} = 0$.

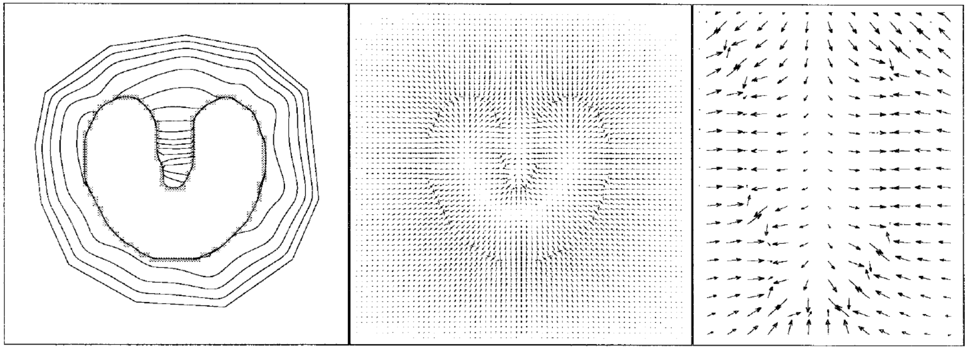
- Numerous parameters must be designed (weighting factors, iteration steps, etc.), a reasonable initialization must be available, and, moreover, the solution of the Euler-Lagrange equation suffers from numerical instability.
- Please refer to the textbook and recent literature for the development of the snake model and its numerical methods.

5.4.2 Gradient vector flow snakes

- Two main limitations common to these approaches are
  - the requirement of snake initialization being close to the desired solution,
  - difficulties in segmenting concave portions of the boundary.
- To overcome these problems, gradient vector flow (GVF) fields and their use in snake image segmentation were reported in [Xu and Prince, 1998].
- GVF field is an external force field that points toward the boundaries when in their proximity and varies smoothly over homogeneous image regions all the way to image borders.
- Consequently, it can drive a snake toward a border from a large distance and can segment object concavities.
- The GVF field is derived from an image by minimization of an energy functional by solving decoupled linear partial differential equations via diffusing the gradient vectors of the edge image.
- The GVF is then used as an external force in the snake equations (5.60) and (5.61) forming a GVF snake.



(a) Snake force



(b) GVF force

Figure 5.11: When adding distance-based forces, the snake segmentation fails in a

- The GVF field $g(x, y) = (u(x, y), v(x, y))$ minimizes the energy functional

$$E = \iint \mu \left[u_x^2 + u_y^2 + v_x^2 + v_y^2\right] + |\nabla f|^2 \, |g - \nabla f|^2 \; dxdy. \tag{5.62}$$

  where μ is a regularization parameter balancing the weight of the first and second terms (increasing μ with increased noise).

- The GVF can be obtained by solving the Euler equations

$$\mu \Delta u - (u - f_x)(f_x^2 + f_y^2) = 0, \tag{5.63}$$
$$\mu \Delta v - (v - f_y)(f_x^2 + f_y^2) = 0. \tag{5.64}$$

- The second term in (5.63) and (5.63) is zero in homogeneous regions since the derivatives f_x, f_y are zero.

- Consequently, the GVF behavior in the homogeneous regions is fully defined by the Laplace equation effectively diffusing the information from the boundaries to the homogeneous parts of the image.

- Solutions to (5.63) and (5.63) can be found by solving the following two decoupled equations for $t \to \infty$,

$$\frac{\partial u}{\partial t} = \mu \Delta u - (u - f_x)(f_x^2 + f_y^2) = 0, \tag{5.65}$$
$$\frac{\partial v}{\partial t} = \mu \Delta v - (v - f_y)(f_x^2 + f_y^2) = 0. \tag{5.66}$$

- These generalized diffusion equations can be solved as separate scalar partial differential equations in u and v.

- Once $g(x, y)$ is computed, (5.61) is modified using the GVF external force $g(x, y)$ yielding the GVF snake equation

$$\frac{\partial v(s,t)}{\partial t} = \frac{d}{ds}\left(\alpha(s)\frac{dv}{ds}\right) - \frac{d^2}{ds^2}\left(\beta(s)\frac{d^2v}{ds^2}\right) + g(v(s)). \tag{5.67}$$

- GVF can be generalized to higher dimensions defining the d-dimensional GVF field.

- Active contour models represent a recent approach to contour detection and image interpretation.
- They differ substantially from classical approaches, where features are extracted from an image and higher-level processes try to interpolate sparse data to find a representation that matches the original data.
- Active contour models start from an initial estimate based on higher-level knowledge, and an optimization method is used to refine the initial estimate.
- During the optimization, image data, an initial estimate, desired contour properties, and knowledge-based constraints are considered.
- Feature extraction and knowledge-based constrained grouping of these features are integrated into a single process, which seems to be the biggest advantage.
- Active contour models, however, search for local energy minima not attempting to achieve globally optimal solutions.
- Applications can be found in many areas of machine vision and medical image analysis.
- Please refer to the GVF page.

5.5 Geometric deformable models — level sets and geodesic active contours

- There are two main groups of deformable contour/surface models: the snakes belong to the `parametric model` family as borders are represented in a parametric form.
- While appropriate for many segmentation tasks, they may yield cusps or intersecting boundaries in some situations.
- They are not capable of handling topology changes of the evolving contours, with direct implementation [Sapiro, 2001, p. 143–144].
- The topology of the final curve will be as the initial one, unless special proceedures are implemented for possible splitting and merging [Sapiro, 2001, p. 143–144].
- This is a problem when an unknown number of objects must be simultaneously detected [Sapiro, 2001, p. 143–144].
- This approach is also non-intrinsic, as the energy depends on the parametrization of the curve and is not directly related to the object geometry [Sapiro, 2001, p. 143–144].

- The second family of deformable surfaces — `geometric deformable models` — overcome this problem by representing developing surfaces by partial differential equations.

- The geometric deformable model literature has grown extensively in the past years, in many cases reporting a variety of applications in which deformable model based segmentation can be used.

- Geometric deformable models were named `level set front propagation` and `geodesic active contour` segmentation approaches, respectively, in the literature.

- The main feature separating geometric deformable models from parametric ones is that curves are evolved using only geometric computations, independent of any parameterization: the process is implicit.

- Consequently, the curves and/or surfaces can be represented as level sets of higher dimensional functions yielding seamless treatment of topological changes.

- Hence, without resorting to dedicated contour tracking, unknown numbers of multiple objects can be detected simultaneously.

- Let a curve moving in time t be denoted by $\mathbf{X}(s.t) = [X(s,t), Y(s,t)]$ where s is the curve parameterization.
- Let $\mathbf{N}$ be the moving curve's inward normal, and c its curvature, and let the curve develop along its normal direction according to the partial differential equation

$$\frac{\partial \mathbf{X}}{\partial t} = V(c)\mathbf{N}, \tag{5.68}$$

where $V(c)$ is the speed function.

- The following figure demonstrates the concept of front evolution.

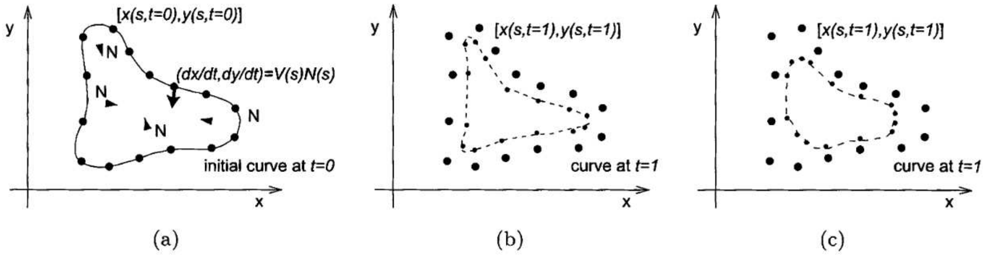


Figure 5.12: Concept of front evolution, (a) Initial curve at $t = 0$. (b) Curve at $t = 1$. Note that each curve point moved in direction of **N** by distance given by velocity V. (c) Curve at $t = 1$ assuming the velocity $V(c)$ is a function of curvature.

- As the curve is moving, it may need to be reparameterized to satisfy equation (5.68).

- If the curve evolution is driven by a curvature deformation equation, the partial differential equation describes a curve smoothing process which removes potential singularities and eventually shrinks the curve to a point

$$\frac{\partial \mathbf{X}}{\partial t} = \alpha c \mathbf{N}, \tag{5.69}$$

where α is a constant — similar to elastic internal forces used in snakes (5.53).

- The following figure shows deformation behavior using positive ($\alpha > 0$) and negative ($\alpha < 0$) curvature.

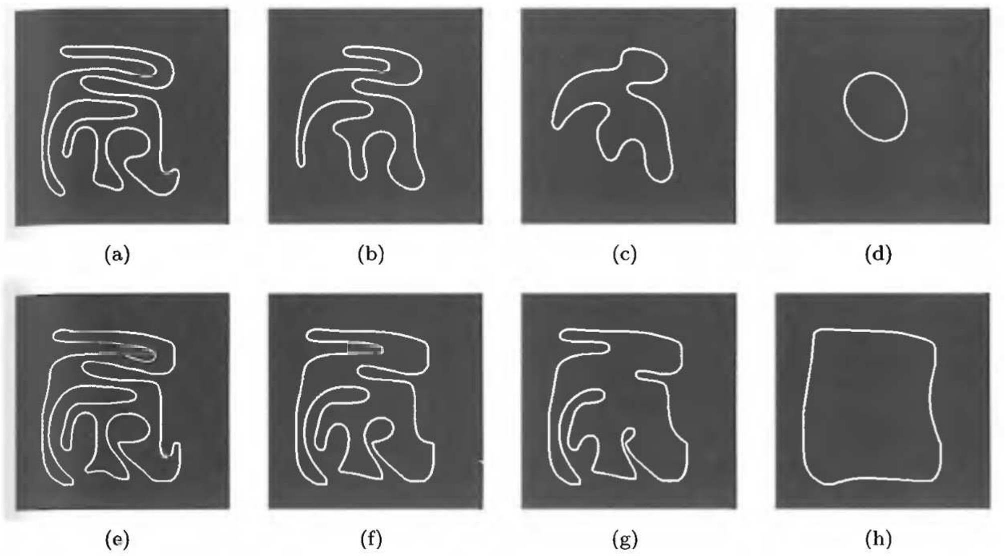


Figure 5.13: Evolution of a closed 2D curve using curvature deformation, (a-d) Using positive curvature, iterations 100, 2,000, 4,000, 17,000. (e-h) Using negative curvature, iterations 100, 2,000, 4,000, 17,000.

- Curve deformation driven by the constant deformation equation

$$\frac{\partial \mathbf{X}}{\partial t} = V_0 \mathbf{N}, \tag{5.70}$$

  is **complementary**, and is similar to the inflation balloon force discussed earlier and may introduce singularities like sharp corners.

- Geometric deformable models perform image segmentation by starting with an initial curve and evolving its shape using the speed equation (5.68).
- During the evolution process, curvature deformation and/or constant deformation are used and the speed of curve evolution is locally dependent on the image data — this represents the motivation for the approach.
- The ultimate goal of curve evolution is to yield desirable image segmentation for $t \rightarrow \infty$: in other words, curve evolution should stop at object boundaries.
- This evolution can be implemented using `level sets` and — similar to many general techniques — the exact behavior of the segmentation technique depends on the segmentation parameters.
- In this case, the segmentation behavior depends on the design of the speed function $V(c)$.

- An important question however remains — how to efficiently execute the curve evolution process.
- The idea that made geometric deformable models feasible is to represent the segmentation boundary/surface implicitly as a level set of a higher-dimensional function — the level set function ϕ — defined on the same image domain.

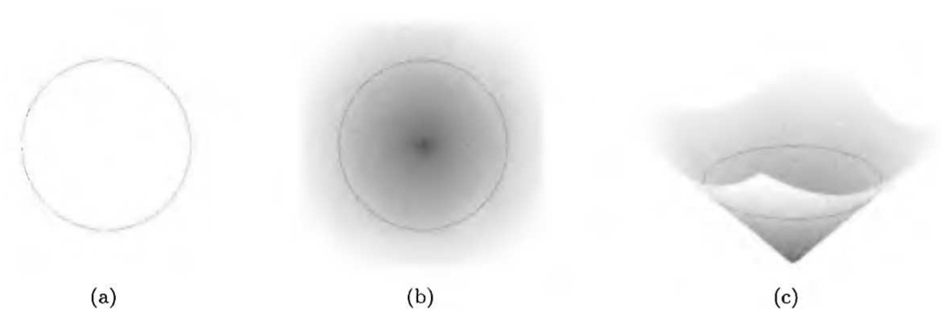


Figure 5.14: An example of embedding a curve as a level set. (a) A single curve, (b) The level set function where the curve is embedded as the zero level set $\phi(\mathbf{X}(s,t)) = 0$ (in black), (c) The height map of the level set function with its zero level set depicted in black.

- Using the level set representation of the curve allows its evolution by updating the level set function $\phi(t)$.
- Instances of curve evolution are obtained by determination of the zero-level set for individual time points $\phi(t) = 0$.
- The final solution is given by the zero-level set $\phi(t \rightarrow \infty) = 0$.

- Importantly, the level set approach allows topology changes, singularity development, etc.

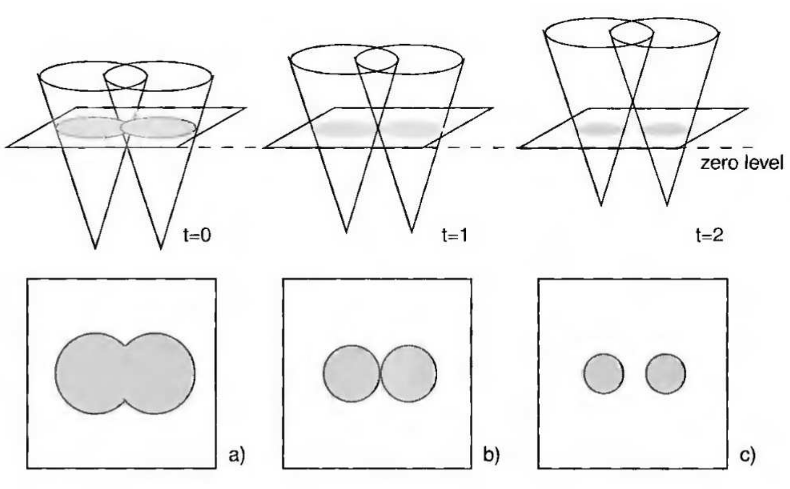


Figure 5.15: Topology change using level sets. As the level set function is updated for $t = 1, 2, 3$, the zero-level set changes topology, eventually providing a 2-object boundary.

- A more formal treatment to define a level set embedding of the curve evolution equation (5.68) is now appropriate.
- Having a level set function $\phi(x, y.t)$ with the contour $\mathbf{X}(s, t)$ as its zero-level set, it follows that

$$\phi(\mathbf{X}(s, t)) = 0. \tag{5.71}$$

- If this equation is differentiated with respect to t and the chain rule is used,

$$\frac{\partial \phi}{\partial t} + \left\langle \nabla \phi, \frac{\partial \mathbf{X}}{\partial t} \right\rangle = 0. \tag{5.72}$$

- Now assuming that ϕ is negative inside the zero-level set and positive outside, the inward unit normal to the level set curve is

$$\mathbf{N} = -\frac{\nabla \phi}{|\nabla \phi|}, \tag{5.73}$$

and from the speed equation (5.68)

$$\frac{\partial \mathbf{X}}{\partial t} = -V(c)\frac{\nabla \phi}{|\nabla \phi|}. \tag{5.74}$$

- Hence,

$$\frac{\partial \phi}{\partial t} - V(c)\left\langle \nabla\phi, \frac{\nabla\phi}{|\nabla\phi|} \right\rangle = 0. \tag{5.75}$$

- Therefore,

$$\frac{\partial \phi}{\partial t} = V(c)|\nabla\phi|. \tag{5.76}$$

- The curvature c at the zero-level set is

$$c = \nabla \frac{\nabla\phi}{|\nabla\phi|} = \frac{\phi_{yy}\phi_x^2 - 2\phi_{xy}\phi_x\phi_y + \phi_{xx}\phi_y^2}{\sqrt{\phi_x^2 + \phi_y^2}^3}. \tag{5.77}$$

- Equation (5.76) shows how to perform curve evolution specified by equation (5.68) using the level set method.

- To implement geometric deformable contours, an initial level set function $\phi(x, y, t = 0)$ must be defined, the speed function must be derived for the entire image domain, and evolution must be defined for locations in which normals do not exist due to the development of singularities.
- The initial level set function is frequently based on the signed distance $D(x.y)$ from each grid point to the zero-level set, $\phi(x, y, 0) = D(x, y)$.
- An efficient algorithm for construction of the signed distance function is called a `fast marching method` [Sethian, 1999].

- Note that the evolution equation (5.76) is only derived for the zero-level set.
- Consequently, the speed function $V(c)$ is undefined on other level sets and needs to be extended to all level sets.
- A number of extension approaches, including the frequently used `narrow band extension`, can be found in [Sethian, 1999].
- The level set equation can be solved iteratively using time step At. However, inherent time step requirements exist to ensure stability of the numerical scheme via the Courant- Friedrichs-Lewy (CFL) condition.

- Further numerical issues are discussed in [Sethian, 1999].

STOP

- A large variety of applications exist in which geometric deformable models were used for image segmentation.

STOP

- The frequently highlighted feature of geometric deformable model segmentations allowing topology changes is an important contribution to the image segmentation tool set.
- However, this behavior may be as detrimental as it may be useful.
- When applied to noisy data with boundary gaps, shapes may be generated which have topology inconsistent with that of the underlying objects.
- In such situations, segmentation topology constraints may be required and a choice of parametric deformable models or graph-based approaches may be more suitable.

Chapter 6

Scale-space Theory with PDE

- The main references for this lecture is [Perona and Malik, 1990, Alvarez et al., 1993, ter Haar Romeny, 1994, Ames, 1992]
- This lecture deals with a fundamental aspect of early image representation — the notion of `scale`.

- The problem of scale must be faced in any imaging situation.
- An inherent property of objects in the world and details in image is that they only exists as meaningful entities over certain range of scale.
  - A simple example of this is the concept of a branch of a tree, which makes sense only at a scale from, say, a few centimeters to at most a few meters.
  - It is meaningless to discuss the tree concept at the nanometer or the kilometer level. At those scales it is more relevant to talk about the molecules that form the leaves, or the forest in which the tree grows.
- Consequently, a `multi-scale representation` is of crucial importance if one aims at describing the structure of the world, or more specially the structure of projections of the three dimensional world onto two dimensional images.

- A vision system for handling objects of different sizes and at different distances needs a way to control the scale(s) at which the world is observed.
- Why should one represent a signal at multiple scales when all information is present in the original data anyway?
- The major reason for this is to explicitly represent the multi-scale aspect of real world images.
- Another aim is to simplify further processing by removing unnecessary and disturbing details.

6.1 Introduction

- We define a `multi-scale analysis` to be a family of transforms $(T_t)_{t \geq 0}$ which, when applied to the original picture $I_0(x)$, yield a sequence of pictures $I(t, x) = (T_t I_0)(x)$.
- We do not now define the scale t, its meaning will be clear from the mathematical formulation below.
- The image $I(t, x)$ is called the analysis of the image I_0 at scale t.
- The goal of this lecture is to review some fundamental results concerning a framework known as `scale-space` that has been developed by the computer vision community for controlling the scale of observation and representing the multi-scale nature of image data.
- We shall see that a few formal principles (or axioms) are enough to characterize and unify the theories and algorithms.

6.1.1 Transformation invariance

- The Euclidean nature of the world around us and the perspective mapping onto images impose natural constraints on the possible operations.
- Objects move rigidly, the illumination varies, the size of objects at the retina changes with the depth from the eye, and view directions may change etc.
- Hence, it is natural to require early visual operations to be unaffected by certain primitive transformations (e.g., translations, rotations, and gray-scale transformations).
- In other words, the visual system should extract properties that are `invariant` with respect to these transformations.

6.1.2 Causality

- A crucial requirement is that structures at coarse scales in the multi-scale representation should constitute simplifications of corresponding structures at finer scales — there should not be any accidental phenomena created by the smoothing method.
- Structures at coarse scales should be related to structures at finer scales in a well-behaved manner.
- Hummel [Hummel, 1986] made the important observation that the maximum principle from the theory of parabolic differential equations is equivalent to causality.
- Therefore, one would expect that a number of (possible nonlinear) differential equations would satisfy causality and possibly have useful applications in vision and image processing.

6.1.3 Morphological invariance

- Another requirement essential to the understanding of images is to take into account how arbitrary the `gray scale` of perceptual or digital image is.
- In the case of digital pictures, many electronic devices are applied successively to an image before its arrival at the human eye or at some automatic image analysis device.
- Since the gray scale of the resulting image has been changed by each device, the only sound assumption about the information preserving properties of the whole chain of captors or transmitters is that they might preserve the order of gray levels.
- In other terms, if some point or some region was brighter than an another in the original picture, this order should be preserved in the final picture.
- This means a "multi-scale" image analysis should not depend upon the image contrast, as the human vision system does not depend upon the image contrast, eps. the local image contrast.

6.2 Some classical approaches

- There exists a large number of possible ways to construct a one-parameter family of derived signals from a given signal.
- The terminology that will be adopted here is to refer a `multi-scale representatio` as any one-parameter family for which the parameter has a clear interpretation in terms of spatial scale.

6.2.1 Linear scale-space

- An earlier formalism for this problem is the idea of scale-space filtering.
- The essential idea of this approach is quite simple: embed the original image in a family of derived images $I(t, x, y)$ obtained by convolving the original image $I_0(x, y)$ with a Gaussian kernel $G(x, y, t)$ of variance t:

$$I(t, x, y) = I_0(x, y) * G(x, y, t). \tag{6.1}$$

- Larger values of t, the scale-space parameter, correspond to images at coarser resolutions.
- This one parameter family of derived images may equivalently be viewed as the solution of the heat conduct or diffusion equation (with heat conduct coefficient or diffusion coefficient $c = 1$)

$$\begin{cases} \dfrac{\partial I}{\partial t}(t, x, y) = \Delta I \\ I(0, x, y) = I_0(x, y). \end{cases} \tag{6.2}$$

- Koenderink [Koederink, 1984] motivates the diffusion equation formation by stating two criteria:

    1. *Causality*: Any feature at a coarse level is required to possess a (not necessarily unique) "cause" at a finer level of resolution although the reverse need not be true.

       In other words, no spurious details should be generated when the resolution is diminished;

    2. *Homogeneity and Isotropy*: the blurring is required to be space invariant.

6.2.2 Quadtrees

- One of the earliest types of multi-scale representations of image data is the quadtrees in § 3.3.2 introduced by Klinger [Klinger, 1971].

6.2.3 Pyramids

- Pyramids are another multi-scale representations, § 3.3.1.
- The main advantage of the pyramid representation are that they lead to a *rapidly decreasing image size*, which reduces the computational work both in the actual computation of the representation and in the subsequent processing.
- The main disadvantage concerning pyramid is that they correspond to quite a coarse quantization along the scale direction, which makes it algorithmically complicated to relate (match) structures across scales,
- Further, pyramids are not translationally invariant.

6.3 Axioms and basic principles

What basic principles must the scale space of image obey?

6.3.1 Causality

- Since the scale space is assumed to yield more and more global information about the image and its features, it is clear that when the scale increases, no new feature should be created by the multi-scale analysis.
- The image and edges at scale $t' > t$ must be simpler than the edges at scale t.
- The formalization has been discussed by many authors.
- The result of the discussion in the case of image processing is that causality must be formalized as `pyramidality` plus a `local comparison principle`.

- First we present the pyramidality assumption.
- We assume that the output at scale $t + h$ can be computed from the output at a scale t for very small h.
- This relationship is obtained by composition of `transition filters` which we call $T_{t+h,t}$.
- We have the following formal formulation

  Pyramidal Structure (Causality 1) $T_{t+h} = T_{t+h,t} \circ T_t,\ T_{t,t} = T_0 = Id$.
- A special case of the above axiom is when the transition filters $T_{t+h,t} = T_h$.
- We have the following semi-group property requirement for T_t,

  Semi-group $T_{t+h} = T_h \circ T_t,\ T_0 = Id$.

- Furthermore, the operator $T_{t+h,t}$ will always be assumed to act "*locally*", i.e., to look at a small part of the processed image.

- In other terms, $T_{t+h,t}(I)_{(x)}$ must essentially depend upon the values of $I(y)$ where y lies in a small neighborhood of x.

- Its physical interpretation is as follows: if the basic elements of the pyramid are assumed to be "neurons", this only means that a neuron is primarily influenced by its neighbors.

- A clear argument for that is based on *time*: only neurons which are close can influence without transmission delay.

- The local comparison principle is: if an image I is locally brighter than another J, then this order must be conserved sometime by the analysis:

  Local Comparison Principle (Causality 2) If $I(y) > J(y)$ for y in a neighborhood of x, then for h small enough,

$$T_{t+h,t}(I)_{(x)} \geq T_{t+h,t}(J)_{(x)}. \tag{6.3}$$

- We finally need some assumption stating that a very smooth image must evolve in a smooth way with the multi scale analysis.
- Somehow, this belongs to the "causality" galaxy — structures at coarse scales should be related to structures at finer scales in a well-behaved manner.
- But we prefer to call it *regularity* and it clearly corresponds to the assumption of the existence of an infinitesimal generator for the multi-scale analysis.
- Readers familiar with the semi-group theory of functional analysis will recognize that the following axiom is similar to the definition of the infinitesimal generator of a linear semi-group.
- However, we only need the axiom holds for quadratic forms.

  Regularity Let $I(y) = \frac{1}{2} < A \cdot (x-y), x-y > + < p, x-y > +c$ be a quadratic form of $\mathbf{R}^N$. There exists a function $F(A,p,x,c,t)$, continuous with respect to A, such that

$$\frac{(T_{t+h,t}(I) - I)_{(x)}}{h} \to F(A,p,x,c,t), \qquad \text{when } h \to 0+. \tag{6.4}$$

- In [Alvarez et al., 1993], the *regularity* formalization is more general than the formalization (6.4).
- The regularity assumptions in [Alvarez et al., 1993] are: the regularity of the operator $T_{t+h,t}$ on some specific function space when t and h are fixed; temporal regularity of the operator $T_{t+h,t}$ with respect to t and s; and some other invariance assumptions.
- However, with those invariance assumptions ([Translation invariance], [Grey-scale-shift invariance]), the regularity assumptions in [Alvarez et al., 1993] is equivalent to (6.4), under causality and conditions.

6.3.2 Morphological invariance

- The morphological invariance that image analysis should not depend upon the contrast change implies that if g is a nondecreasing continuous function:

  Morphological Invariance $g \circ T_{t+h,t} = T_{t+h,t} \circ g$, which means that change of contrast and multi-scale analysis can be applied in any order.

6.3.3 Transformation invariance

- For $z \in \mathbf{R}^2$, define
$$\tau_z(I)_{(x)} = I(x+z) \tag{6.5}$$
to be the translation operator. The translation invariance is

  Translation Invariance $\tau_z T_{t+h,t} = T_{t+h,t}\tau_z$.

- If A is an isometry, denote by Au the function $Au_{(x)} = u(Ax)$. The Euclidean invariance is

  Euclidean Invariance $AT_{t+h,t} = T_{t+h,t}A$.

6.4 PDEs for the scale-space theory

- There are therefore six main axioms and we shall see that they allow to completely classify and characterize the theories of multi-scale image analysis, to unify and to improve several of them.

- First we present the fundamental theorem in this framework.

Theorem 6.4.1. *If an image multi-scale analysis T_t is causal[1] and regular then $I(t,x) = T_t(I)_{(x)}$ is a viscosity solution of*

$$\frac{\partial I}{\partial t} = F(\nabla^2 I, \nabla I, I, x, t) \tag{6.6}$$

where the function F, defined in the regularity axiom (6.4), is nondecreasing with respect to its first argument $\nabla^2 I$.

Conversely, if I_0 is a bounded uniformly continuous image, then the equation (6.6) has a unique viscosity solution.

[1] It means that T_t satisfies both causalities 1 and 2.

- One particular case of this equation is that $F(A, p, c, x, t) = \operatorname{tr}(A)$. We get the classical heat equation

$$\frac{\partial I}{\partial t} = \Delta I \tag{6.7}$$

- As a consequence of this theorem, all multi-scale models can be classified and new, more invariant models can be proposed.

Proof We give a simplified proof by assuming that $I(t,x)$ is C^2. A completed and rigorous proof could be found in [Alvarez et al., 1993]. In a neighborhood of (t,x), we have

$$I(t,y) = I(t,x) + < \nabla I(x), y-x > + \frac{1}{2} < \nabla^2 I(x)(y-x), y-x > + o(|y-x|^2).$$

Let $\varepsilon > 0$ and Q_ε the quadratic form given by

$$Q_\varepsilon(y) = I(t,x) + < \nabla I(x), y-x > + \frac{1}{2} < \nabla^2 I(x) \cdot (y-x), y-x > + \varepsilon |y-x|^2.$$

Then, in a neighborhood of (t,x)

$$Q_{-\varepsilon} < I(t,y) < Q_\varepsilon(y) \qquad \text{for } y \neq x.$$

By using the causality principle "Local Comparison Principle", we obtain

$$T_{t+h,t}(Q_{-\varepsilon})_(x) \leq T_{t+h,t}(I(t))_(x) \leq T_{t+h,t}(Q_\varepsilon)_(x).$$

On the other hand, we also have

$$Q_{-\varepsilon}(x) = I(t,x) = Q_\varepsilon(x)$$

Therefore we deduce from the above relations, when $h \to 0+$,

$$\begin{aligned} F(\nabla^2 I - 2\varepsilon Id, \nabla I, I, x, t) &= \lim \frac{T_{t+h,t}(Q_{-\varepsilon})_{(x)} - Q_{-\varepsilon}(x)}{h} \\ \leq \liminf \frac{T_{t+h,t}(I(t))_{(x)} - I(t,x)}{h} &= \liminf \frac{I(t+h,x) - I(t,x)}{h} \\ \leq \limsup \frac{I(t+h,x) - I(t,x)}{h} &= \limsup \frac{T_{t+h,t}(I(t))_{(x)} - I(t,x)}{h} \\ \leq \lim \frac{T_{t+h,t}(Q_{\varepsilon})_{(x)} - Q_{\varepsilon}(x)}{h} &= F(\nabla^2 I + 2\varepsilon Id, \nabla I, I, x, t) \end{aligned}$$

By using the regularity principle and the continuity of the function F with respect to A, and taking $\varepsilon \to 0+$, we obtain that $I(t,x)$ satisfies equation (6.6).

Finally, in order to obtain that $F(A,p,c,x,t)$ is nondecreasing with respect to A, we notice that if $A \leq B$ then the quadratic forms

$$Q_A(y) = \frac{1}{2} < A \cdot (x-y), x-y > + < p, x-y > +c$$
$$Q_B(y) = \frac{1}{2} < B \cdot (x-y), x-y > + < p, x-y > +c$$

satisfy $Q_A(y) \leq Q_B(y)$ and $Q_A(x) \leq Q_B(x)$. By using an obvious adaptation of the above proof, we obtain

$$F(A, p, c, x, t) \leq F(B, p, c, x, t)$$

$\square$

- To simplify the exposition, we have showed that equation (6.6) is true in the case where I is a C^2 function.
- By using the viscosity solutions (see [Bardi et al., 1997]), it can be showns that equation (6.6) is true in the sense of viscosity solution for any $I(t, x)$ uniformly continuous satisfying the causality and regularity principles.
- For a proof that if $I_0(x)$ is a bounded uniformly continuous function, the equation (6.6) has a unique solution, see the introduction in [Bardi et al., 1997]) and the references therein.

6.4.1 The Marr-Hildreth-Koenderink-Witkin theory

- We now deduce from Theorem 6.4.1 a characterization of the heat equation $\frac{\partial I}{\partial t} = \Delta I$ as the unique linear and isometrically invariant multiscale model.

- Thus we get a formal justification of a theory based on many formal and heuristic arguments which has always pointed to the heat equation as the only possible multiscale analysis.

- We give here a proof of this intuition: the heat equation is the only linear isometrically invariant multiscale analysis.

- Thus for image models, *linearity and morphological invariance are incompatible*.

- We also obtain an explanation for the coexistence of (at least) two different schools in image processing: mathematical morphology on one side and classical multiscale analysis on the other.

- The classical model comes from Marr and Hildreth [Marr and Hildreth, 1980] and has been formalized by Witkin [Witkin, 1983] and Koenderink [Koederink, 1984]. Canny [Canny, 1986] proposed an efficient variant. The basic step of the multiscale analysis is the convolution of the original image with Gaussian of increasing variance. Koenderink [Koederink, 1984] noticed that the convolution of the signal with Gaussian at each scale is equivalent to the solution of the heat equation with the signal as initial datum.

- To state the result in our axiomatic approach, we need the following invariance axiom, instead of the morphological invariance,

  Grey-scale-shift Invariance $T_t(0) = 0$, $T_t(I + C) = T_t(I) + C$.

  Theorem 6.4.2. *If a multiscale analysis is causal, regular, translation invariant, Euclidean invariant, gray-scale-shift invariant, and linear, then up to a rescaling $t' = h(t)$, $I(t, x) = T_t(I_0)(x)$ is the solution of the heat equation*

$$\begin{cases} \frac{\partial I}{\partial t} & = \Delta I, \\ I\big|_{t=0} & = I_0. \end{cases} \tag{6.8}$$

  Proof For $z \in \mathbf{R}^2$, by translation invariant,

$$I_z(t, x) = I(t, x + z) = \tau_z(I(t))_{(x)} = \tau_z(T_t(I_0))_{(x)} = T_t(\tau_z(I_0))_{(x)}$$

  is the solution of $\frac{\partial I_z}{\partial t} = F(\nabla^2 I_z, \nabla I_z, I_z, x, t)$. We have

$$\frac{\partial I}{\partial t}(t, x + z) = F(\nabla^2 I(t, x + z), \nabla I(t, x + z), I(t, x + z), x, t).$$

  On the other hand $I_z(t, x)$ satisfies, by the equation of I,

$$\frac{\partial I}{\partial t}(t, x + z) = F(\nabla^2 I(t, x + z), \nabla I(t, x + z), I(t, x + z), x + z, t).$$

Therefore, by comparing the above two equation, we have

$$F(A, p, c, x, t) = F(A, p, c, x + z, t).$$

Then we conclude that $F(A, p, c, x, t)$ does not depend on x. We get

$$F(A, p, c, x, t) = F(A, p, c, t).$$

Similarly, by Grey-scale-shift invariance, we get

$$F(A, p, c, x, t) = F(A, p, t).$$

By linearity, $F(A, p, t)$ is a linear function of A and p for each fixed t. We may write

$$F(A, p, t) = F_2(A, t) + F_1(p, t).$$

Now consider the Euclidean invariance. Given an isometry R,

$$I_R(t, x) = I(t, R \cdot x) = R(I(t))_{(x)} = R(T_t(I_0))_{(x)} = T_t(R(I_0))_{(x)}$$

is the solution of $\frac{\partial I_R}{\partial t} = F(\nabla^2 I_R, \nabla I_R, t)$. We have

$$\frac{\partial I}{\partial t}(t, R \cdot x) = F(\nabla^2 I(t, R \cdot x), \nabla I(t, R \cdot x), t).$$

On the other hand $I_R(t, x)$ satisfies, by the equation of I,

$$\frac{\partial I}{\partial t}(t, R \cdot x) = F(R^{\text{tr}} \cdot \nabla^2 I(R \cdot x) \cdot R, R^{\text{tr}} \nabla I(R \cdot x), t).$$

Therefore, by comparing the above two equation, we have

$$F_2(R^{\text{tr}} A R, t) = F_2(A, t)$$
$$F_1(R^{\text{tr}} p, t) = F_1(p, t)$$

where A is any symmetric matrix.

Consider first F_1. For any $p \neq 0$, we can find an R such that

$$R^{\text{tr}} p = (1, 0).$$

Therefore F_1 takes at most two possible values $F_1(0)$ and $F_1(1, 0)$. Since F_1 is linear, $F_1 = 0$.

Now consider F_2. Since for any symmetric matrix A, there exists orthogonal matrix R such that $R^{\text{tr}} A R$ is a diagonal matrix, the elements on the diagonal line being the eigenvalues of A, λ_1, λ_2. We may write

$$F_2(A, t) = F_2(\lambda_1, \lambda_2, t)$$

Since F_2 does not depend on the order of the eigenvalues, it is a symmetric function of the eigenvalues. Now the only linear symmetric function is the sum. We conclude

$$F_2(A, t) = c(t)\mathrm{trace}(A).$$

In summary, the equation (6.6) becomes

$$\frac{\partial I}{\partial t} = c(t)\Delta I$$

Thus with the rescaling $\frac{\partial t'}{\partial t} = c(t)$, we obtain the heat equation. □

6.4.2 The general morphological multiscale analysis

Theorem 6.4.3. *Let $N = 2$. If a multiscale analysis is causal, regular, translation invariant, Euclidean invariant, morphological invariant, $I(t,x) = T_t(I_0)(x)$ is the solution of the heat equation*

$$\begin{cases} \frac{\partial I}{\partial t} & = |\nabla I| G(div(\frac{\nabla I}{|\nabla I|})), \\ I|_{t=0} & = I_0. \end{cases} \tag{6.9}$$

where G is a continuous function on $\mathbf{R} \times (\mathbf{R}^2 \setminus \{0\})$.

- This is equivalent to a curve evolution by

$$\frac{\partial X}{\partial t} = -G(\kappa)\overrightarrow{\mathbf{n}}, \tag{6.10}$$

by the level-set method (5.76).

Bibliography

[Alvarez et al., 1993] Alvarez, L., Guichard, F., Lions, P. L., and Morel, J. M. (1993). Axioms and fundamental equations of image processing. *Archive for Rational Mechanics and Analysis*, 123(3):199 – 257.

[Ames, 1992] Ames, W. F. (1992). *Numerical Methods for Partial Differential Equations*. Academic Press, San Diego.

[Andrews and Hunt, 1977] Andrews, H. C. and Hunt, B. R. (1977). *Digital Image Restoration*. Prentice–Hall, Englewood Cliffs, NJ.

[Bardi et al., 1997] Bardi, M. et al. (1997). *Viscosity Solutions and Applications*, volume 1660 of *Lecture Notes in Mathematics*. Springer–Verlag, Berlin–Heideberg–New York.

[Canny, 1986] Canny, A. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:769 – 698.

[Cervenka and Charvat, 1987] Cervenka, V. and Charvat, K. (1987). Survey of the image processing research applicable to the thematic mapping based on aerocosmic data. Technical report, Geodetic and Carthographic Institute, Prague, Czechoslovakia. Technical Report A 12-346-811.

[D. R. Wilson and T. R. Martinez, 1997] D. R. Wilson and T. R. Martinez (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*.

[Folland and Sitaram, 1997] Folland, G. B. and Sitaram, A. (1997). The uncertainty principle: A mathematical survey. *Journal of Fourier Analysis and Applications*, 3:207 – 238.

[Freeman, 1961] Freeman, H. (1961). On the envoding of arbitrary geometric confgiuration. *IRE Transactions on Electronic Computers*, EC-10(2):260 – 268.

[Frisby, 1979] Frisby, J. P. (1979). *Seeing — Illusion, Brain and Mind*. Oxford University Press, Oxford.

[Gamerman, 1997] Gamerman, D. (1997). *Markov Chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman & Hall.

[Haralic and Shapiro, 1992] Haralic, R. M. and Shapiro, L. G. (1992). *Computer and Robot Vision*. Reading: Addision Wesley.

[Horn, 1986] Horn, B. R. (1986). *Robot Vision*. M.I.T. Press, Cambridge.

[hua Zhao and hua Zhong, 1982] hua Zhao, K. and hua Zhong, X. (1982). *Optics*. Peking University Press, Beijing. 2 volumes, in Chinese.

[Hummel, 1986] Hummel, R. A. (1986). Representations based on zero crossings in scale space. In *Proceedings of the IEEE computer Vision and Pattern Recognition Conference*, pages 204 – 209.

[Kass et al., 1987] Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: active contour models. In *1st International Conference on Computer Vision*, pages 259 – 268, London, England.

[Kass et al., 1988] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331.

[Klette and Zamperoni, 1996] Klette, R. and Zamperoni, P. (1996). *Handbook of Image Processing Operators*. John Wiley & Sons, Inc, New York.

[Klinger, 1971] Klinger, A. (1971). Pattern and search statistics. In Rustagi, J. S., editor, *Optimizing Methods in Statistics*. Academic Press.

[Knuth, 1981] Knuth, D. E. (1981). *Seminumerical Algorithms*. Addison-Wesley, Reading, Massachusetts, 2nd edition. volume 2 of The art of Computer Programming.

[Koederink, 1984] Koederink, J. J. (1984). The structure of images. *Biological Cybernetics*, 50:363 – 370.

[Kovalevski, 1989] Kovalevski, V. A. (1989). Finite topology as applied to image analysis. *Computer Vision, Graphics and Image Processing*, 46:141 – 161.

[Kutter, 1999] Kutter, M. (1999). *Digital Image Watermarking: hiding information in images*. ph.d dissertation, Départment d'Électricité, Ècole Polytechnique Fédérale de Lausanne.

[Liow, 1991] Liow, Y. T. (1991). A contour tracing algorithm that preserves common boundaries between regions. *CVGIP – image understanding*, 53:313 – 321.

[Marr, 1982] Marr, D. (1982). *Vision*. Freeman.

[Marr and Hildreth, 1980] Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London. Series B*, 207:187 – 217.

[Martelli, 1972] Martelli, A. (1972). Edge detection using heuristic search methods. *Computer Vision, Graphics and Image Processing*, 1:169 – 182.

[Oppenheim et al., 1997] Oppenheim, A. V., Willsky, A. S., and Nawwab, S. H. (1997). *Signals and Systems*. Prentice Hall, Inc.

[Pavlidis, 1977] Pavlidis, T. (1977). *Structural Pattern Recognition*. Springer Verlag, Berlin, Heidelberg, New York.

[Perona and Malik, 1990] Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629 – 639.

[Pratt, 1978] Pratt, W. K. (1978). *Digital Image Processing*. John Wiley & Sons, Inc, New York.

[Press et al., 1992] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientifc Computing*. Cambridge University Press, 2nd edition.

[Russ, 1995] Russ, J. C. (1995). *The Image Processing Handbook, 2nd ed.* CRC Press.

[Sapiro, 2001] Sapiro, G. (2001). *Geometric partial differential equations in image analysis*. Cambridge University Press, New York, NY, USA.

[Sethian, 1999] Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press. Cambridge Monograph on Applied and Computational Mathematics.

[ter Haar Romeny, 1994] ter Haar Romeny, B. M., editor (1994). *Geometric-Driven Diffusion in Computer Vision*. Kluwer Academic Publishers, Dordrecht.

[van der Heijden, 1995] van der Heijden, F. (1995). Edge and kine feature extraction based on covariance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:69 – 77.

[Vincent and Soille, 1992] Vincent, L. and Soille, O. (1992). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583 – 598.

[Wang and Wu, 1991] Wang, Z. J. and Wu, S. D. (1991). *Imaging Optics*. Science Press, Beijing. in Chinese.

[Witkin, 1983] Witkin, A. P. (1983). Space-scale filtering. In *Proc. of IJCAI*, pages 1019 – 1021.

[Wrinkler, 1995] Wrinkler, G. (1995). *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*. Springer, Berlin–Heideberg.

[Xu and Prince, 1998] Xu, C. and Prince, J. L. (1998). Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359 – 369.