

LECTURE NOTES

IMAGE PROCESSING

EUGENE AGEENKO

1999



UNIVERSITY OF KUOPIO

TABLE OF CONTENTS

<u>CHAPTER 1. IMAGE ACQUISITION AND REPRESENTATION.....</u>	<u>5</u>
1.1. INTRODUCTION: LIGHT, COLOR AND OUR EYES	5
1.2. DIGITAL IMAGE	5
1.3. IMAGE TYPES	6
1.4. DIGITIZATION PARAMETERS - IMAGE RESOLUTION AND QUANTIZATION.	12
<u>CHAPTER 2. IMAGE ANALYSIS AND ENHANCEMENT</u>	<u>14</u>
2.1. IMAGE ANALYSIS.....	14
2.2. IMAGE ENHANCEMENT	17
<u>CHAPTER 3. IMAGE RESIZING AND RESAMPLING</u>	<u>20</u>
3.1. RESIZING.....	20
3.2. RESAMPLING - QUANTIZATION	22
3.3. RESAMPLING – HALFTONING.....	24
<u>CHAPTER 4. NOISE AND IMAGE FILTERING</u>	<u>27</u>
4.1. NOISE	27
4.2. IMAGE FILTERING.....	29
<u>CHAPTER 5. IMAGE SEGMENTATION.....</u>	<u>31</u>
5.1. DETECTION OF DISCONTINUITIES	32
5.2. REGION GROWING.....	34
5.3. THRESHOLDING.....	35
<u>CHAPTER 6. BINARY IMAGE PROCESSING.....</u>	<u>38</u>
6.1. RESOLUTION REDUCTION	38
6.2. IMAGE FILTERING	39
<u>CHAPTER 7. GLOBAL METHODS – HIGH LEVEL IMAGE PROCESSING.....</u>	<u>40</u>
7.1. STATISTICAL CONTEXT-BASED FILTERING	40
7.2. FEATURE-BASED FILTERING.....	43
<u>APPENDIX: TERMINOLOGY OVERVIEW.</u>	<u>47</u>

Chapter 1. Image Acquisition and Representation

1.1. Introduction: Light, Color and Our Eyes

Light is the flow of particles with the weight of electron – photons. Due to its duality (matter-energy), each photon can be considered as the short piece of electromagnetic wave, and therefore light is electromagnetic radiation of varying frequencies. Visible light is in the range (approximately) 400-700 nm.

The frequency (or mix of frequencies) of the light determines the color.

The amount of light - quantity of photons - relates to its energy and defines light intensity. Light having a dominant frequency (or set of frequencies) is called *chromatic*. Light without such dominant frequencies (white light) is called *achromatic*.

Our eyes have two types of receptors:

- **Cones** are sensitive to colored light, but they are not very sensitive to dim light. Cones give us the information about light color.
- **Rods** are sensitive to achromatic light only. Rods provide us with light intensity and work even in dim light (therefore we lose the color information at night).

We perceive colors using three different types of cones: red, green, blue (The receptors actually respond in opponent pairs: red/green and blue/yellow, but it is a reasonable way of looking at it). Thus, color is variable combination of the three wavelengths – called *primary colors*.

1.2. Digital Image

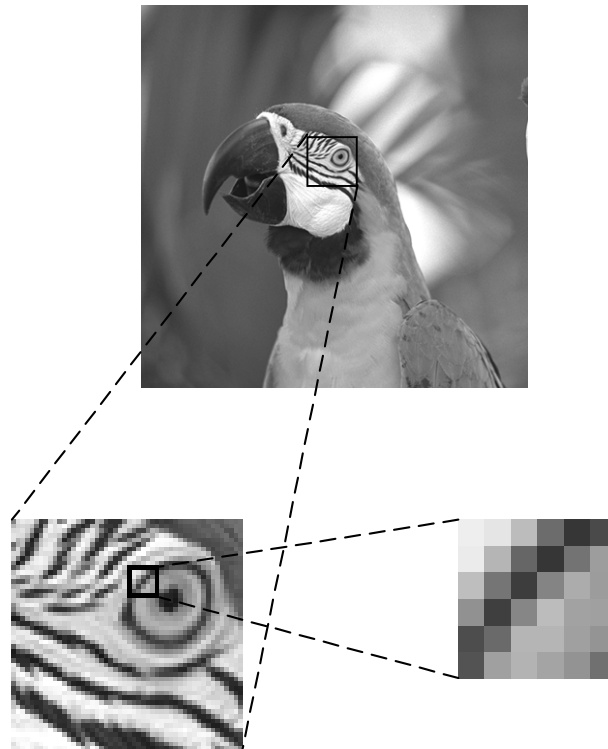
Function of spatial coordinates

Image is a function of the spatial coordinates: $f(x,y)$, where x,y denotes a position in a projected plane space. Two dimensions are most common because it is most natural for us to see two-dimensional projections of the world – photographs. Higher dimensions are also possible, since the image does not need to have any visual sense.

Array of discrete samples

Digital image is an array of discrete samples of continuous function representing the observable object. The sampling pattern is usually a rectangular grid, and samples are called *picture elements* or *pixels*, each pixel having its own discrete value in a finite range. The pixel values may represent the amount of visible or infra red light, absorption of x-rays, electrons, or any other measurable value such as ultrasonic wave impulses. The images may be obtained from a digital camera, scanner, radar, electron microscope, ultrasound stethoscope, or any other optical or non-optical sensor, or may be synthetic image as well.

We will deal here with digital images as with matrix of $N \times M$ pixels, each represented by k bits (eg. $k=8$). A pixel can thus have 2^k (256) different values typically illustrated using different shades of gray, and is considered as integer varying from 0 (black pixel) to 2^k-1 (white pixel).



1.3. Image types

There exist various categories of images. These can be categorized regarding of the source, the image is originated from:

1. Digital camera images
2. Satellite images
3. Radar images
4. Medical images
5. Binary images (fax and digitized document images)
6. 3D images
7. Synthetic images

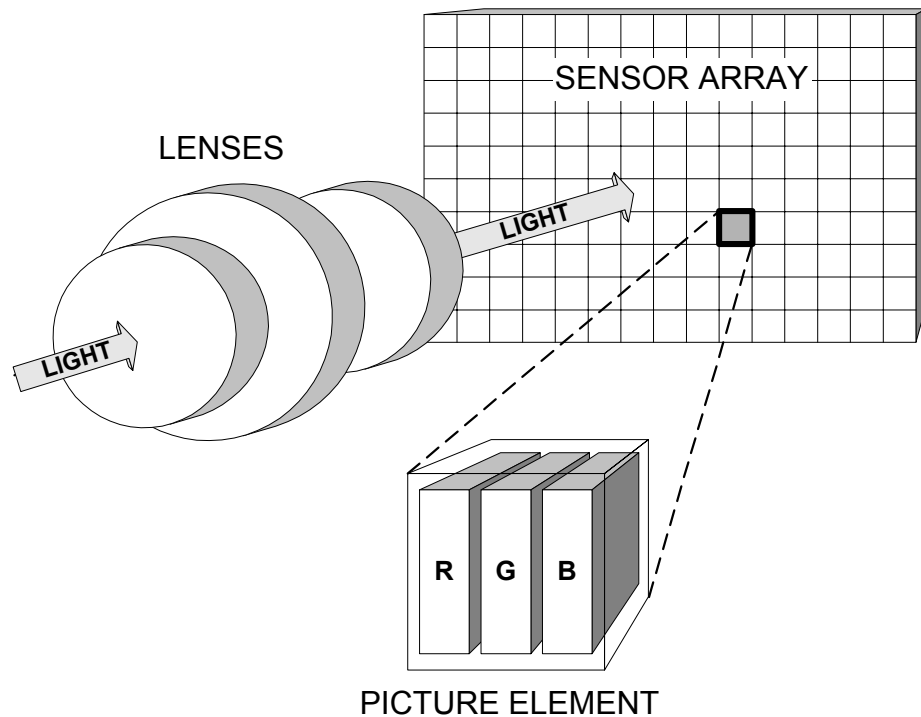
Digital Camera

The most common types of images are from cameras. Photographs are the most similar to what we see out of our eyes: a two-dimensional projection of a three-dimensional world. Scanner is also camera, by dealing with already flat two-dimensional images (eg. photographs). Scanner has linear sensor passing through the image and producing the digital (sampled) copy of the image.

Some (simpler) cameras only detect brightness – the color component is ignored – and resulting image is shown as *black-n-white* image. The simplest types of images are those that are truly black and white – *binary images*, whose pixels represented by a single bit. Example of such kind of images is document image or facsimile, where one bit per pixel is usually sufficient.

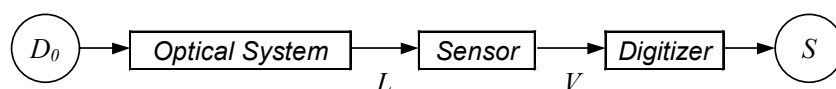
More often black-n-white images have varying shades of gray – *grayscale images*. The most common type uses one byte per pixel and stores pixel values in the range 0..255 or 0..64.

Color cameras use usually three type of sensors, each is sensitive to the one of the primary colors: Red, Green, Blue and resulting color is defined as their linear combination. Each pixel is described as three (r,g,b) , where values are often varied in diapason 0..255 (24 bit per pixel). And image can be thus represented as three grayscale images, representing the red, green and blue components, respectively.



The principles behind the digitization are following:

1. Being the scanning light reflected from the object – original document pass throw the *optical system*, usually lenses and form the image radiant energy distribution L .
2. This light intensity is transformed to the electrical current V by the *sensor*.
3. Finally, the electrical current is digitized by *digitizer*. Digitizer contains an Analog to Digital Converter (ADC), which transform analog image to its digital form S – matrix of the $N \times M$ pixels, where $N \times M$ is the image size.



Satellite images

Satellite images are two-dimensional projections of the real, three-dimensional planetary surface usually taken from far above the surface. Although these are mostly flat looking

images, 3-D data still can be acquired by analyzing the changing in the brightness caused by shadows. Several images from the single place during the various time of the day must be taken to perform such analysis.

Radar images

Radar images are mostly binary images representing surrounding environment. The specific of these images is that only contours of surrounding objects that lie in direct eye-of-sight can be seen on the picture.

Medical Images

Medical Images generally fall into two categories: transmission (projection) and emission.

Transmission (Projection)

Transmission or projection images use a radioactive source placed *behind* the target to project radiation through the target. The structures of the target cast shadows on the other side, where the image is then taken and depict body anatomy. The most common form of projection image is the *radiograph*, called often as X-ray.

By taking X-ray images at a range of angles around the body, one can create a three-dimensional reconstruction of the anatomy. This process is known as *Computed Tomography* (shortened from *Computed Axial Tomography*).

Ultrasound is a form of projection imaging using high-frequency sound waves, but rather than seeing shadows, it sees reflections (echoes).

Emission

Emission images use a radioactive source placed *inside* the target. Radioactive isotopes are usually bound with pharmaceuticals that distribute themselves in certain ways according to body function. The radiation emitted by the target is directly imaged (unless anatomy of the patient casts shadows) and gives information about distribution of the radiation source within the body.

A recent form of emission imaging is *Magnetic Resonance Imaging* or MRI, which causes certain molecules of the body to resonate and give off low-level radiation in the FM radio range. This signal can be detected outside of the body and the distribution of the resonating molecules can be formed into an image. MRI is more sensitive to soft tissues than X-ray imaging.

Binary and document images

Binary images are mostly document images. *Document image* is the exact digitized replica of an original paper document. Images are superior to paper documents because they can be economically stored, efficiently searched and browsed, copied without loss of quality, and quickly transmitted. Moreover, document image is a media, satisfying with the legal requirements and library preservation standards. The current industry is oriented to produce and reproduce paper documents, and it does it faster than the papers became digitized. Document Imaging aims at stopping (or at least slowing down) the growth of the paper piles and substitutes for paper in storing and accessing information. DI provides easier access to the electronic replicas of documents, and minimizes storage cost, comparing with other document storage solutions, such as paper or microfilm.

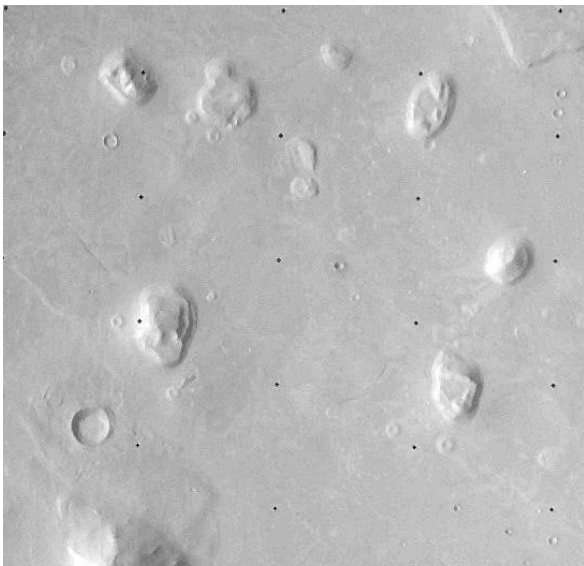
3D images

These images are either two-dimensional projection of 3D object given by a model or purely three-dimensional data array of pixels representing the 3D structure. In the first case, the image consists usually of 2D polygon projections, whose brightness varied depending on the orientation of the polygon the light source. Example of the later case is MRI or CT image consisting of 2D slices taken along the primary axis. Using computer simulation, one can produce useful 3D-model from these samples.

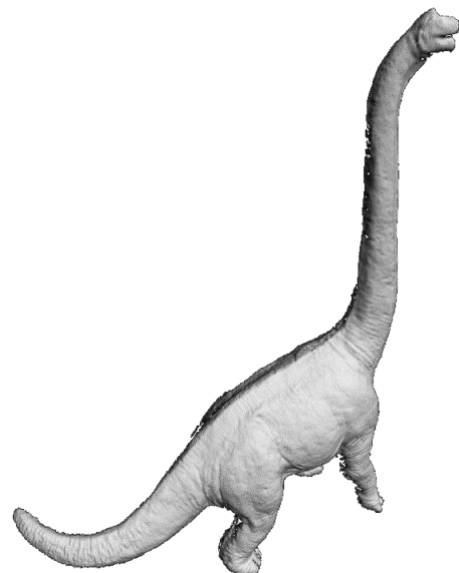
Synthetic images

Synthetic image represents the object having no analogs in visible world. It can be picture of micro-life or artificially colored radiograph. In principle any two dimensional signal can be visualized as the image.

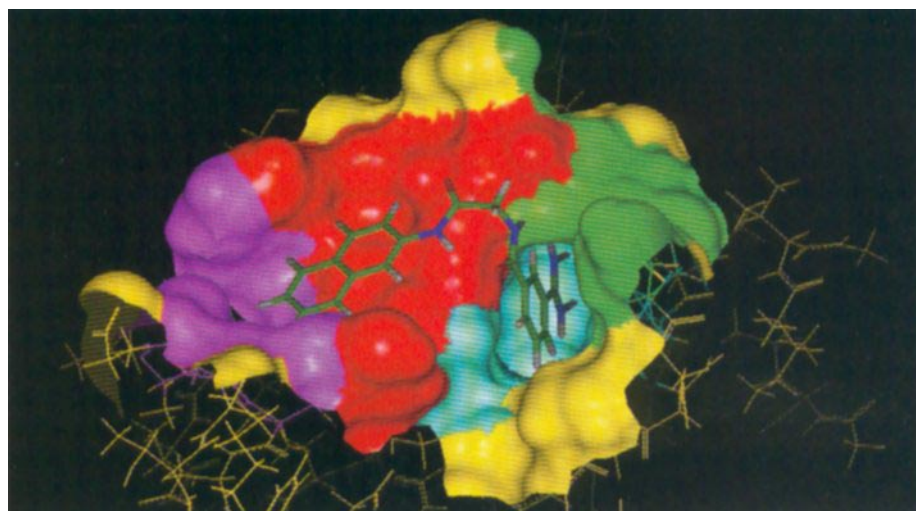
Image examples:



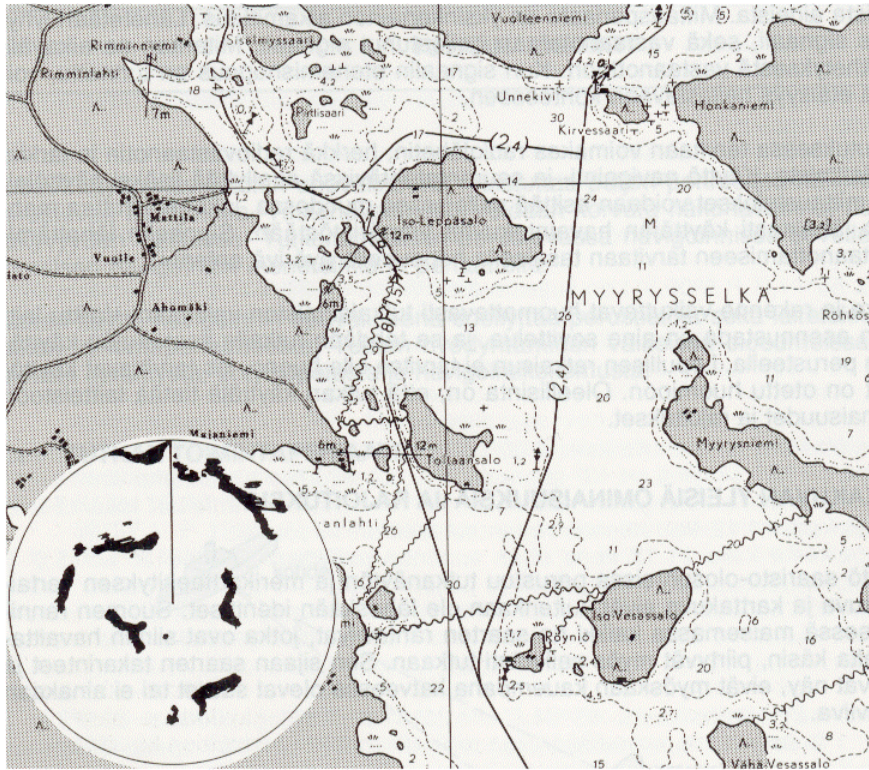
Satellite image



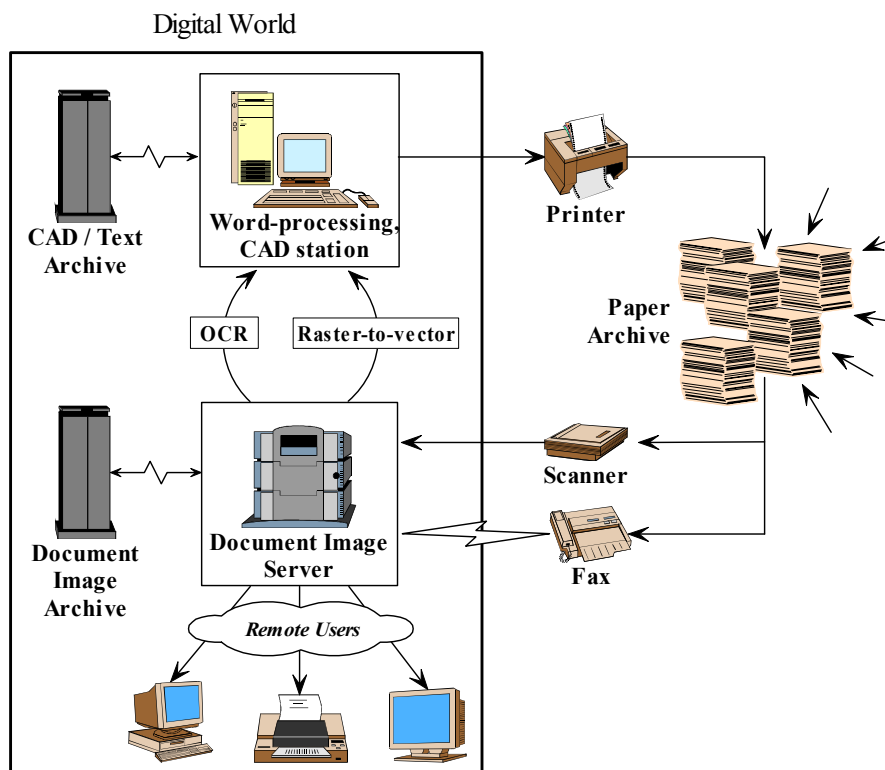
3D image



Molecular simulation

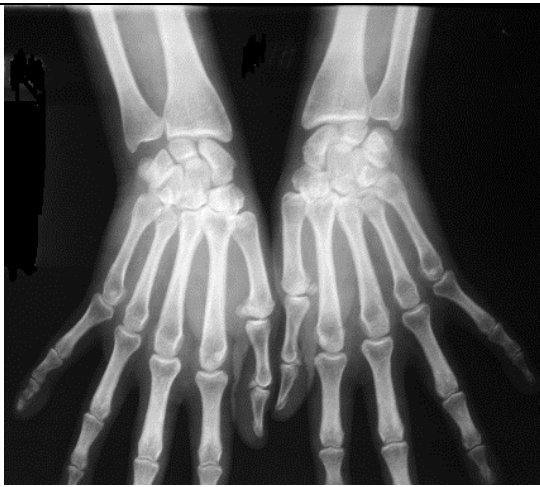

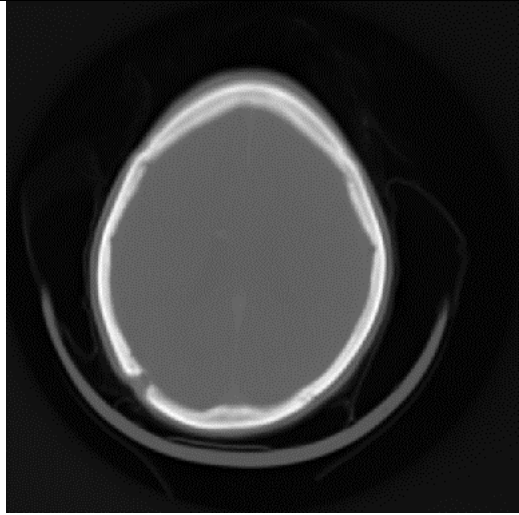
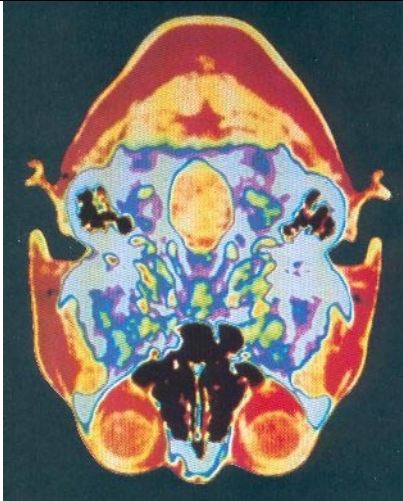
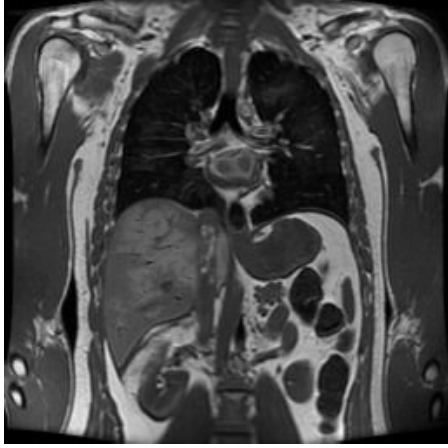



Radar image



Document imaging

Medical image examples:

	
<p>X-ray image</p>	<p>Enhanced X-ray image</p>
	
<p>CAT-image</p>	<p>Enhanced CAT-image</p>
	
<p>MRI-image</p>	<p>Ultrasonic image</p>

1.4. Digitization parameters - Image Resolution and Quantization.

As we know already digital image is the sampled version of the scene over the two-dimensional sampling grid. The two main parameters of the digitization process are:

- *Image resolution*: the number of samples in the grid per spatial unit.
- *Quantization steps*: how many bits are used per sample.

Both parameters have a direct effect on the image quality and the image file size. In general, image quality increase as the resolution and quantization level increase. But, in an image with a very high resolution only a few of quantization steps are required. For example, photograph consist of only the three types of crystals (cyan, magenta and yellow) each having only one of the two states: activated or not. From this point of view photograph is like the digital image – high resolution and only 3 bit per pixel quantization; but we use to consider it as the analog source, when digitizing in the scanner. Finally, everything became discrete on a higher level of magnification.

The resolution and quantization may be the tradeoff parameters when storage size is making sense, and their choice depends on the application.

Sampling and Image Resolution

Resolution is often measured in pixels per spatial unit: inch or mm in linear dimension (DPI - *dot-per-inch*). Do not mix the resolution with the image *size* – number of pixels on each image.

The requirements on the spatial resolution depend either on the usage of the image and the image content. If the target printing (or display) size of the image is known, the digitizing resolution can be chosen accordingly.

In practical applications resolution has its minimal limit, related with sharpness of the optical system, used in camera, and defining the smallest size of features can be seen in an image. Resolution is usually defined in terms of how many distinct alternating black and white lines can be identified within some spatial range.

Quantization

In addition to discrete spatial sampling, images also use discrete *quantization* of the value of the image function. Images can use as few as two discrete levels (one bit) to represent an image as black and white or as many as 256 (8 bit) to represent it as a grayscale.

The properties of human eye imply some upper limits: it is known that we can observe at most one thousand different gray levels, but in practical situations 256 gray levels (or even 64) is usually enough. For application where the visual quality is not the primary concern (computer analysis) this parameter may have higher value.

Image pixels are not necessary have to be scalar values, but can be vector valued as well. The typical example is the RGB color system, where each pixel represented as vector in three-dimensional color space. So popular 24 bit-per-pixel quantization allocates 8 bit per each color axis. In contrary in YUV color system (or YIQ used in TV) the color information (U and V) is separated from the intensity channel (Y). The fact, that contrast of the intensity channel is noticeable higher than in case of color channels, allow to distribute more discrete levels for describing intensity and less for the color, and is widely used in color image compression.

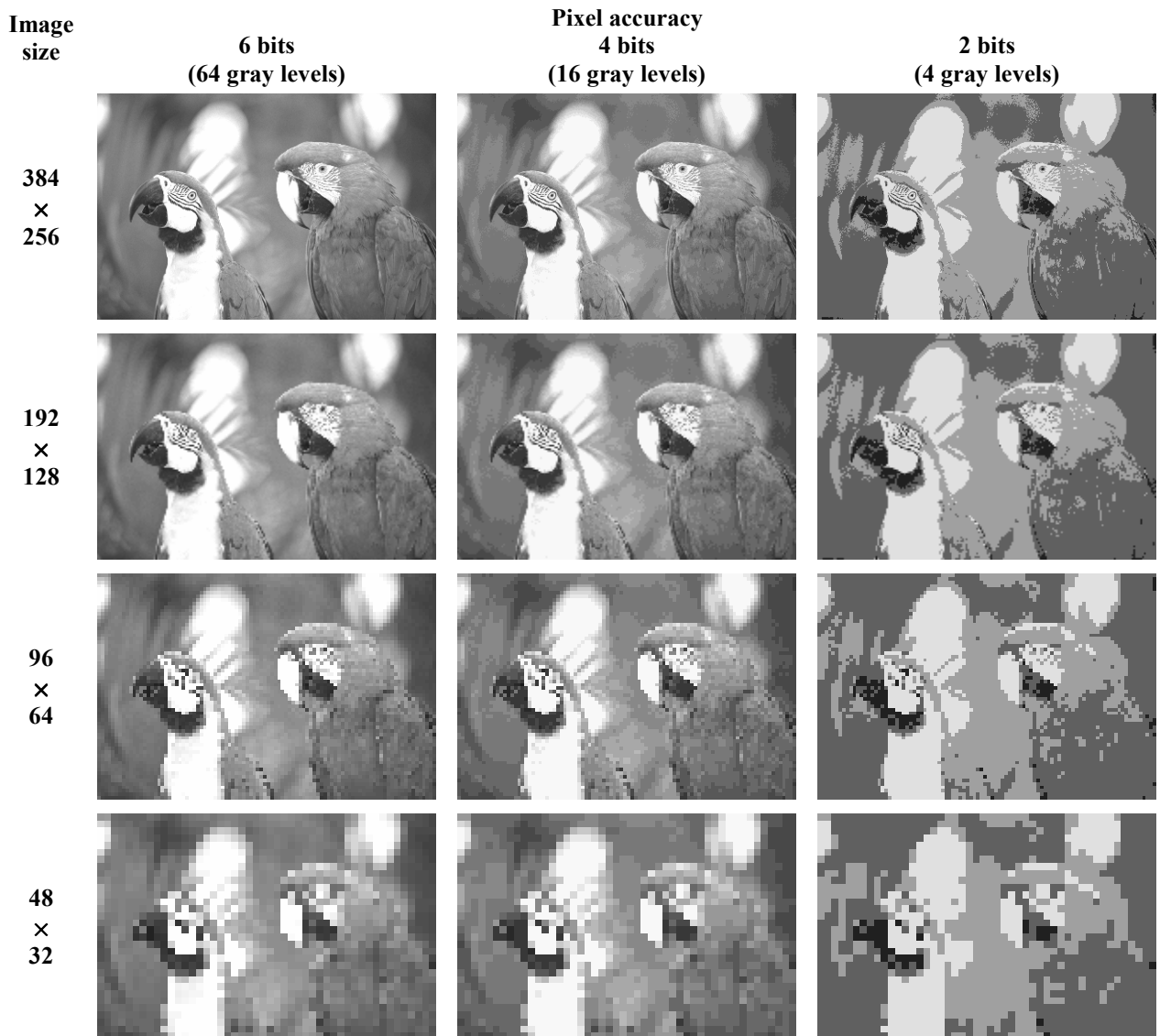


Table: Memory consumption (bytes) with the different parameter values.

Resolution	Bits per pixel:				
	1	2	4	6	8
32 × 32	128	256	512	768	1,024
64 × 64	512	1,024	2,048	3,072	4,096
128 × 128	2,048	4,096	8,192	12,288	16,384
256 × 256	8,192	16,384	32,768	49,152	65,536
512 × 512	32,768	65,536	131,072	196,608	262,144
1024 × 1024	131,072	262,144	524,288	786,432	1,048,576

Chapter 2. Image Analysis and Enhancement

2.1. Image Analysis.

The color.

Color is the perceptual result of light in the visible region of the spectrum, (400 nm .. 700 nm), incident upon the retina.

The eye has three types of color photoreceptor *cone* cells, which respond to incident radiation with somewhat different response.

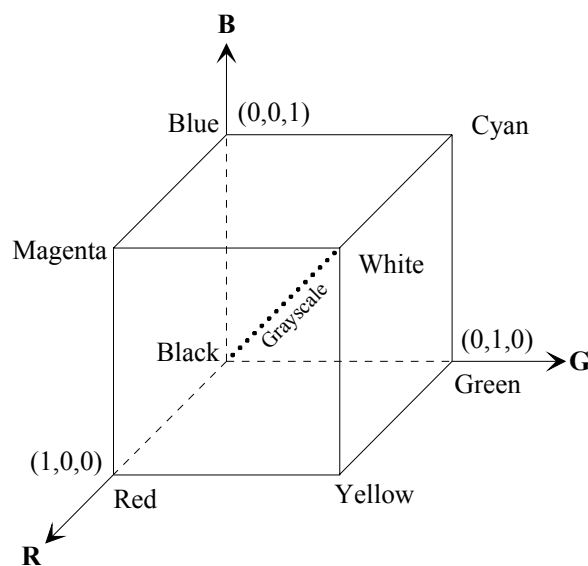
All colors are seen as variable combination of three wavelengths:

- 435.8 nm BLUE;
- 546.1 nm GREEN;
- 700.0 nm RED.

Note: Spectral power distribution (SPD) exists in real world, but color exists only in the eye and the brain!

Additive color model – RGB color space

$$\text{Color} = (R, G, B)$$



White color.

In additive image reproduction, the *white color* is the color reproduced by equal red, green and blue components.

Subtractive color.

CMY color model is based on three colors: CYAN, MAGENTA, YELLOW, which mixed together, will result in BLACK color. The model designed for color printing (monitor is black, paper is white).

$$C = 1 - R$$

$$M = 1 - G$$

$$Y = 1 - B$$

The intensity.

Intensity is a measure over some interval of the electromagnetic spectrum of the flow of power that is radiated from/on a surface (i.e. the total light across all frequencies).

Intensity is a *linear-light* measure, expressed in units such as watts per square meter.

The brightness.

Brightness is defined as *the attribute of a visual sensation according to which an area appears to emit more or less light*. Brightness is a perceptual attribute, it has no firm objective measure.

In image processing software, brightness is defined as the component that determines the amount of black in a color. It is represented usually as integer value from 0 to 100.

The luminance.

Luminance, denoted Y , is radiant power weighted by a spectral sensitivity function that is characteristic of vision (according to *CIE*).

$$Y = 0.3 \cdot R + 0.6 \cdot G + 0.1 \cdot B$$

- The magnitude of luminance is proportional to physical power. But the spectral composition of luminance is related to the brightness sensitivity of human vision.

The chrominance.

Chrominance components U and V represent the color information and form, together with Luminance, the **YUV color model**.

$$U = B - Y$$

$$V = R - Y$$

$$Color = (Y, U, V)$$

YIQ color model.

A color model used in television broadcast systems (North American video standard - NTSC). Colors are split into a luminance value (Y) and two chromaticity values (I and Q). On a color monitor, all three components are visible; on a monochrome monitor, only the Y component is visible.

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

$$I = 0.596 \cdot R - 0.275 \cdot G - 0.321 \cdot B$$

$$Q = 0.212 \cdot R - 0.523 \cdot G + 0.311 \cdot B$$

The hue.

Hue is the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors, i.e. the dominant wavelength in spectrum.

The saturation.

Saturation is the colorfulness of an area judged in proportion to its brightness. Saturation runs from neutral gray through pastel to saturated colors. I.e. the more spectrum distribution is concentrated at one wavelength, the more saturated will be the color.

One can desaturate a color by adding light that contains power at all wavelengths.

HSB and HSI color models.

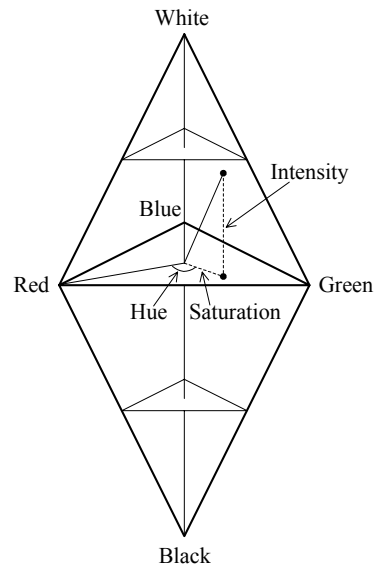
Hue, Saturation and Luminance form the HSL color model:

$$Color = (H, S, L)$$

Note: in most computer system the L is often replaced by I (intensity), defined as:

$$I = \frac{Red + Green + Blue}{3}$$

or by B (brightness) determining the amount of black in a color. This computation conflicts with the properties of *Human Vision* and is made only to simplify expressions for H and S.



The lightness.

Human vision has a nonlinear perceptual response to brightness:

- a source having a luminance only 18% of a reference luminance appears about half as bright.

The perceptual response to luminance is called *Lightness* and is defined by the CIE as a modified cube root of luminance:

$$L^* = 116 \cdot \left(\frac{Y}{Y_n} \right)^{\frac{1}{3}} - 16$$

where Y_n is the luminance of the white reference

Stated differently, lightness perception is roughly logarithmic.

Lab color model.

The color model developed by Commission internationale de l'eclairage (CIE) based on three parameters: lightness (**L**), and two chromaticity ranges: **a** (green to red) and **b** (blue to yellow). The Lab square, two-dimensional visual selector defines the a and b coordinates from -60 to 60; the vertical visual selector defines the L value from 0 to 100. This model is device-independent, and encompasses the color gamut of both the CMYK and the RGB color models.

Color model illustrations

See Appendix

Histogram

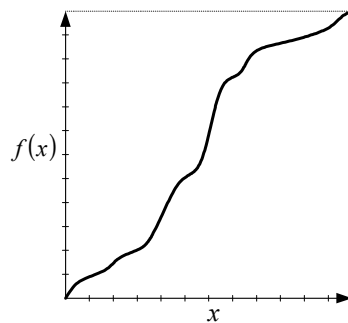
The **histogram** is a frequency distribution function of the pixels in an image. It is usually depicted as a horizontal bar chart indicating *how many pixels are at each brightness level*. The histogram is commonly used in analysis of tonal problems in the image and decision how to deal with them.

Every image's histogram will have a certain amount of hills and valleys, but you will probably need to tonally correct an image with obvious spikes (probably, because some images legitimately contain large amounts of black or white). If the pixels are obviously weighted at either end, you may need to compress the tonal range or redistribute the pixels along the tonal range. If there are large gaps between the bars, posterization has probably occurred.

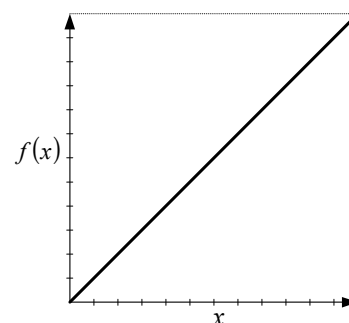
2.2. Image Enhancement

Gray level (intensity) transform

The gray-level transform (GLT) is a function (usually non-decreasing) that takes current pixel brightness values as input and outputs them at different values. It allows performing global tonal and color corrections, and offers precise, local control over any individual level of values in relation to all other levels of values. The GLT is characterized by its response graph, known as *tonal curve*.



Arbitrary Transform

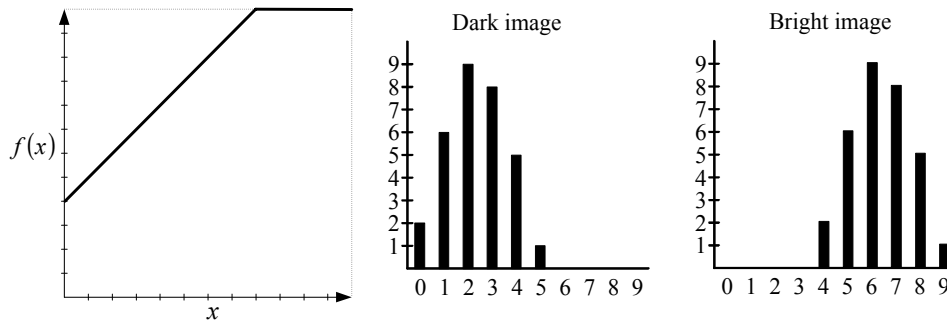


Identity Transform

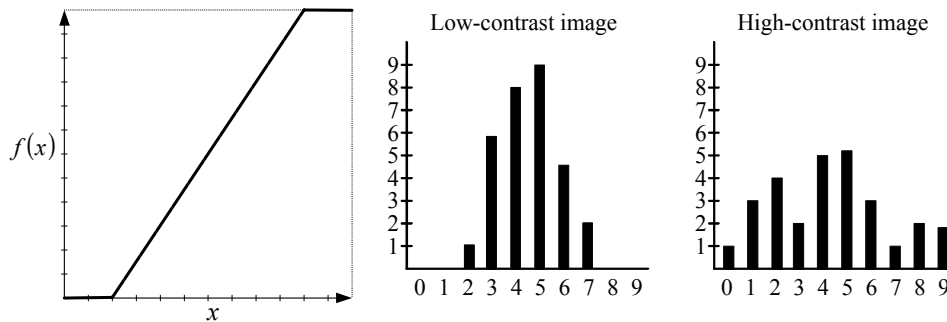
Tonal corrections allow you to control the relationship between the shadows, midtones, and highlights in your image, as well as to adjust the brightness, contrast, lightness, and darkness of your colors. It is widely used to restore detail that is lost in shadows or highlights, to correct under or over-exposure, and to generally improve the tonal quality of the image.

Typical Gray Level transforms

Constant addition, $f(x) = x \pm const$, controls image brightness by shifting all pixel values up or down the tonal range. When you adjust the brightness, you are lightening or darkening all colors equally.



The *Contrast* control adjusts the distance between the lightest and darkest pixels in the image.



Histogram Stretching or Level Equalization

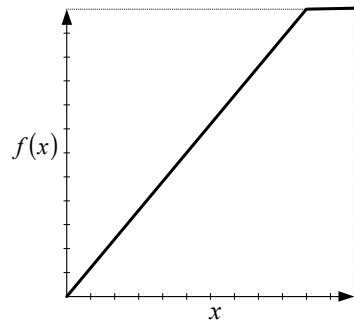
The Level Equalization enables you to adjust the shadow and highlight areas of your image or redistribute the pixel values throughout the entire tonal range. It stretches or shrinks the histogram between the selected points of the tonal range, and redistributes shades from the darkest to the lightest, starting and ending with the values you set.

In an image processing software, one can select the highlight point (brightness levels that should be the lightest in the image) and shadow point (the darkest) directly from the image by pointing to the appropriate pixels. Image processing software usually have the highlight and shadow slider-controls that will move the right (bright) and left (dark) point of the histogram in both direction stretching or shrinking the histogram, respectively.

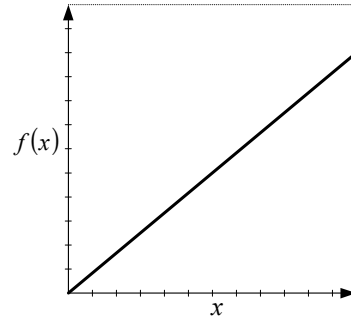
You can use this tool to artificially create color gradations when posterization has happened unintentionally, to lighten or darken any combination of the shadows, mid-tones, or highlights, to compress brightness values to printable limits, or expand it throughout the entire tonal range.

The histogram stretching can be applied to luminance channel alone to control the brightness and contrast of the image, or to each color channel separately, to improve color tone distribution and eliminated any color casts in the image.

Here (below) are examples of the histogram stretching function $f(x) = x \cdot const$. It controls the highlight point (most-right corner of the histogram) by moving it left/right within the tonal range: left if $const < 1$, and right if $const > 1$.



Histogram stretching (right)



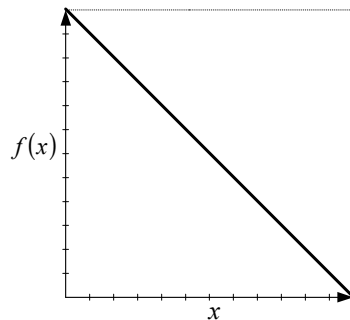
Histogram stretching (left)

Negation

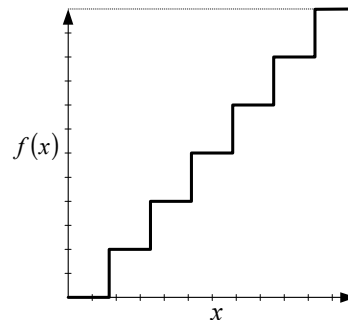
Negation, $f(x) = white - x$, is used for displaying the negative medical images and photographs on the positive screen.

Quantization

Image quantization is a technique where the number of gray-levels is reduced. For example, the gray-scale image with 8 bits per pixel has 256 different gray levels. The image can be quantized to 64 levels simply by taking the 6 most significant bits of the pixel values. This operation performs a *uniform quantization* or *posterization*, where the range of each gray-level value is equal. The applications of quantization can be found in image compression (eg. downsampling of chromacity channels in compression of video images).



Negation

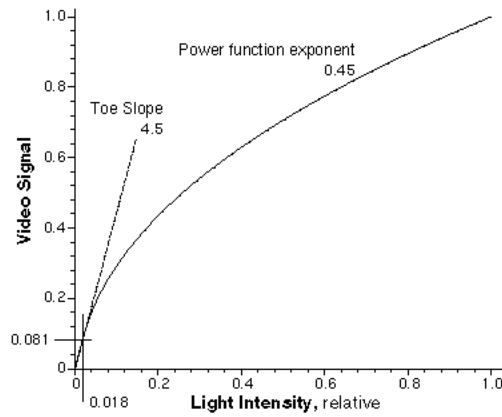


Quantization

Gamma correction

Gamma correction is a method of tonal correction that takes the human eye's into account. It lets you pick up detail in a low contrast image without significantly affecting the shadows or highlights. The resulting image has a higher contrast in the shadows and lower in the highlights.

Video systems also approximate the human perceptual response to luminance (lightness) using RGB signals that are each subject to a 0.45 power function.



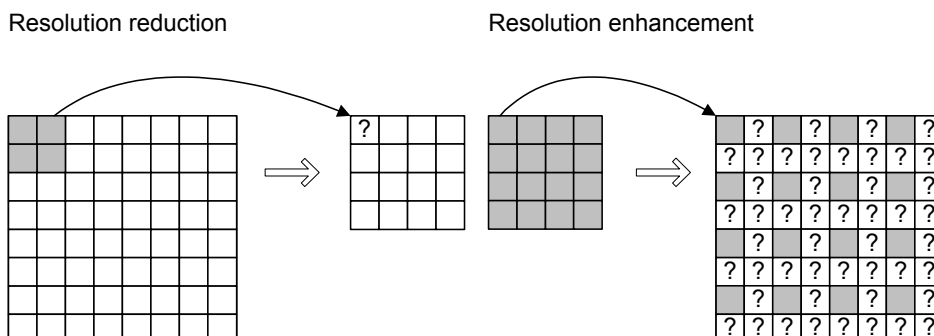
Chapter 3. Image Resizing and Resampling

3.1. Resizing

Resolution reduction

A reduced resolution version of a given image is sometimes needed for a preview purpose, for example. A preview image (or *thumbnail*) must be small enough to allow fast access but also with sufficient quality so that the original image is still recognizable. A smaller resolution copy is also needed if the image is embedded into another application (or printed) using smaller size as the full resolution would allow. Sometimes the image resolution may be reduced just for saving memory.

Resolution reduction is formally defined as follows: given an image of $N \times M$ pixels, generate an image of size $N/c \times M/c$ pixels (where c is a zooming factor) so that the visual content of the image is preserved as well as possible. There are two alternative strategies: (1) *sub-sampling*, (2) *averaging*. In both cases, the input image is divided into blocks of $c \times c$ pixels. For each block, one representative pixel is generated to the output image. In sub-sampling, any of the input pixels is chosen, e.g. the upper leftmost pixel in the block. In averaging, the pixel depends on the values of all input pixels in the block. It could be chosen as the average, weighted average, or the median of the pixels. Averaging results in smoother image whereas sub-sampling preserves more details. Depending on the application, various constraints may apply when resizing the image. These help to preserve the edges in the image – high frequency components that can be lost due to resolution reduction. The sub-sampling method is known to preserve these feature better than averaging, although averaging results in a smoother and more pleasant for viewing image.

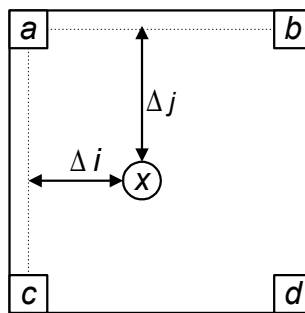


Resolution enhancement

The resolution of an image must sometimes be increased, e.g. when the image is zoomed for viewing. For each input pixels there are $c \times c$ output pixels to be generated. This is opposite to resolution reduction. A straightforward method simply takes copies of the input pixel but this may results in a jagged (blocky) image where the pixels are clearly visible. A more sophisticated method known as *bilinear interpolation* generates the unknown pixel values by taking the linear combination of the four nearest known pixel values. The value of a pixel $x_{i,j}$ at the location (i,j) can be calculated as:

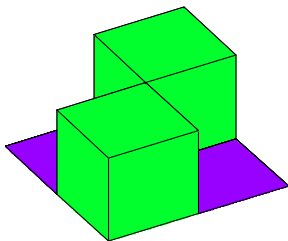
$$x_{ij} := a + \Delta_i \cdot (b - a) + \Delta_j \cdot (c - a) + \Delta_i \cdot \Delta_j \cdot (a - b - c + d)$$

where a, b, c, d are the nearest known pixel values to x . The result is a smoother image compared to straightforward copying of a single pixel value. Examples of resolution reduction and resolution increasing by interpolation are illustrated below.



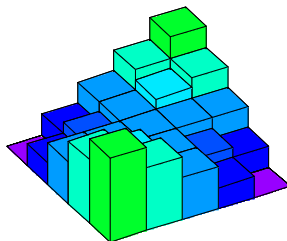
Bilinear interpolation of pixel x using the four known pixels a, b, c, d :

Original 2x2



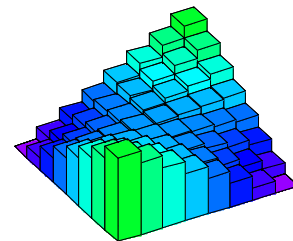
X

Bilinear interpolation 5x5



B

Bilinear interpolation 8x8



Y(X, 8, 8)

Example of bi-linear interpolation



Example of resolution reduction $384 \times 256 \rightarrow 48 \times 32$:
original image (left), resolution reduced by averaging (middle), and by sub-sampling (right).



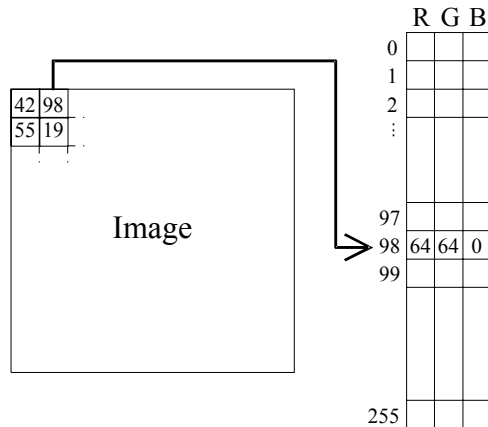
Bilinear interpolation using two 48×32 images from above:
original image (left), obtained using averaging (middle), sub-sampling (right)

3.2. Resampling - quantization

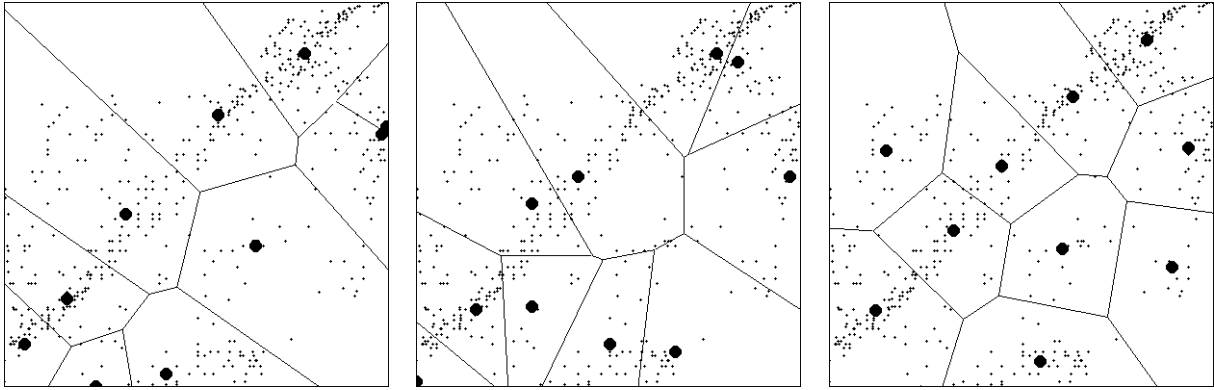
Image Quantization stands for reducing the number of colors in the image. It has applications in image compression and image analysis.

Palette images

Palette images are used to minimize the memory requirements (eg. 8-bpp VGA/SVGA graphics or GIF images). A *color palette* of 256 specially chosen colors (basis colors) is generated to approximate the image. Each original pixel is mapped to its nearest palette color and the index of this color is stored instead of the *rgb*-triple. Along with the image data, a dictionary (*codebook*) is stored in the file to identify the color values for basis colors.



An important application of color image quantization is the conversion of true color images to color-palette images. Here the RGB color space is quantized to K colors. Gray-scale pixels are often quantized by using very simple algorithms, eg. cutting off the least significant bits. The color images, on the other hand, consist of three components, and if these were separately quantized the result would not be satisfactory.

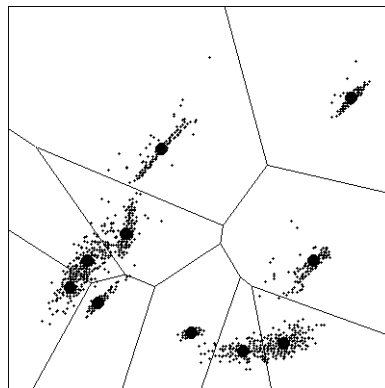


Examples of three various palettes for two-color space model

The following algorithms are widely used for generation the color palette.

The popularity algorithm

This algorithm is creates color palette by finding the densest regions in the color distribution of the original image. It simply chooses the K colors from the histogram with the highest frequencies, and uses these for the color-map. This can be done using a simple *sort*. The algorithm works well with most of the image, but performs poorly on images with a wide range of colors.



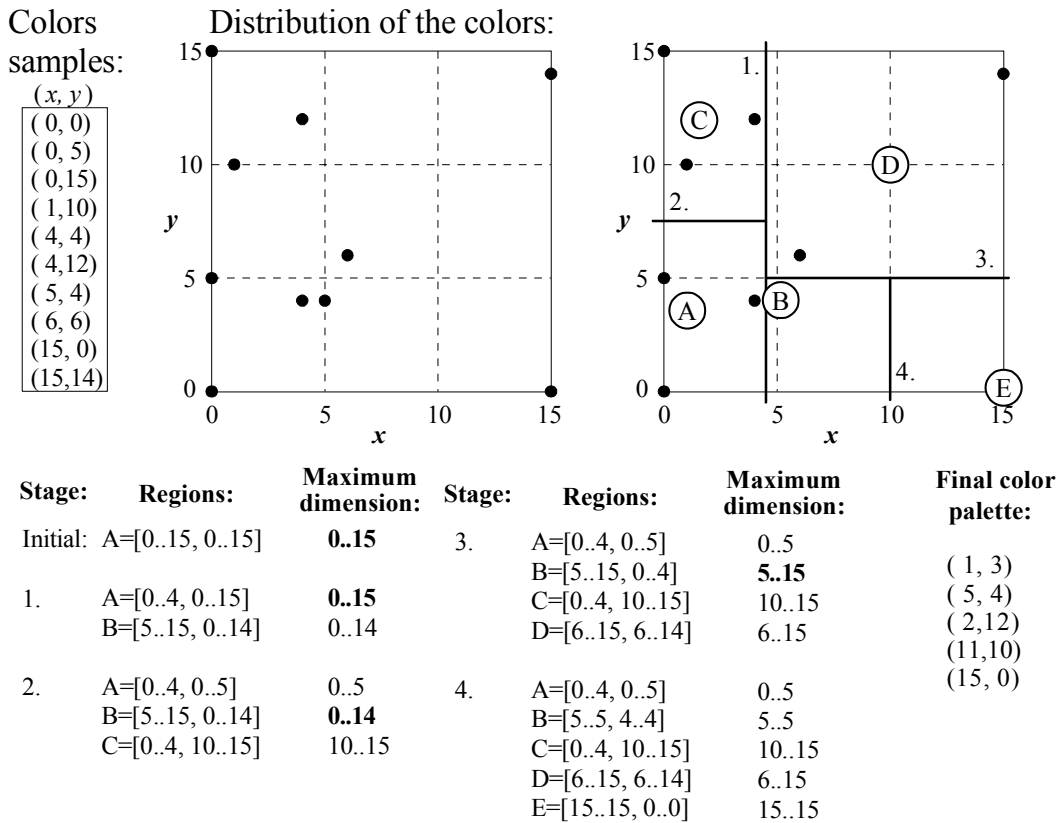
Example of the popularity algorithm

The median cut algorithm:

The median cut algorithm splits the color space iteratively into rectangular sub-regions until it consists of K regions only. It begins with only one region that is the full color space. At each iteration, region dimensions are calculated by finding minimum and maximum values of each of the color coordinates. The enclosed points are sorted along the longest dimension of the region (the values of the points are projected to the axis corresponding to the longest region dimension and then these projections are sorted). This region with the maximum dimension is further split into two halves at the medial point. Approximately equal numbers of points will fall on each side of the cutting plane. The step is recursively applied until K regions are generated. The representative for each region is computed by averaging the colors contained in each.

See below an example of the median cut algorithm. For simplicity let us consider a two color system (x, y) with the range of $[0, 15]$, (i.e. 256 colors at maximum). Suppose that we have ten color samples of the image, and that the size of the color palette is predefined to five. At the first phase the color space is split at $x = 4.5$. At the second phase the region A has the

maximum dimension and is therefore split at $y = 7.5$. At this point, we have three regions consisting of five (region A), three (B), and two (C) color samples. At the next stage, the region B is split at $y = 7$. Finally, the region A is further split at $x = 3$. The resulting five regions are then assigned by the average colors of the samples in each region. The colors of the regions (A, B, C, D, E) are $[(1,3), (5,4), (2,12), (11,10), (15,0)]$.



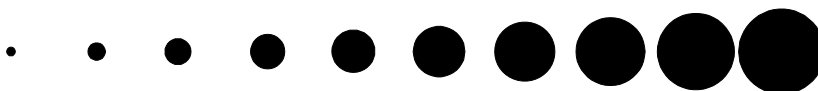
Example of the median cut algorithm.

3.3. Resampling – Halftoning

Halftoning represents a class of methods for converting gray-scale images to binary, black and white images. It is also used to convert color images to the images with reduced number of colors, for example to CMYK images with only cyan, magenta, yellow and black color, used for color printing. Halftoning can be trivially performed by global thresholding, or mapping to the nearest basis color. However, it is usually desired to retain intensity information in the image.

Rasterization

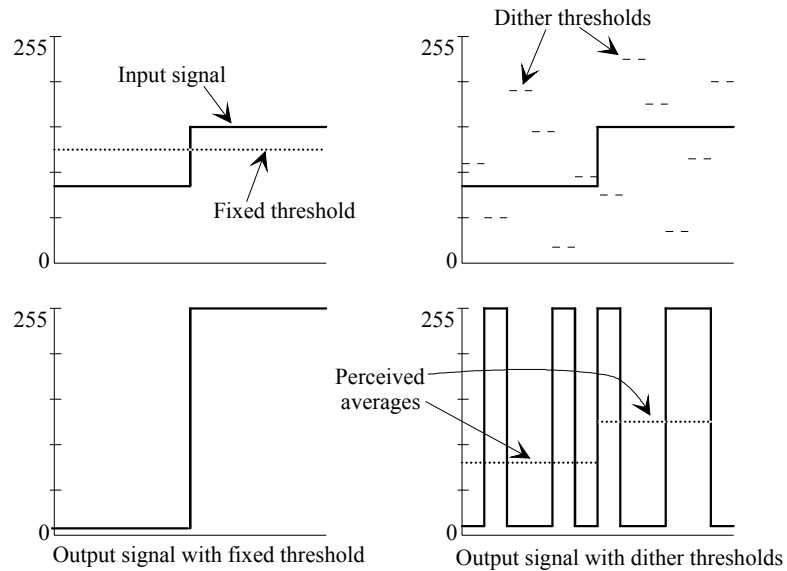
Rasterization is widely used in newspaper technology in the printing phase, where the digital image is transferred back to analog form. The gray-scale pixels are reproduced as different sized black dots whose diameter depends on the brightness of the pixel. The darker the pixel, the larger the dot and vice versa.



Rasterized gray-scale values from white to black.

Dithering

Dither coding (or *dithering*) methods convert gray-scale images to binary images while trying to retain the average grayness of the image without increasing the resolution. The main idea of dithering is to apply a variable threshold in the binarization, see Figure 4.13. The two most common dithering methods are *ordered dithering* and *error diffusion*.



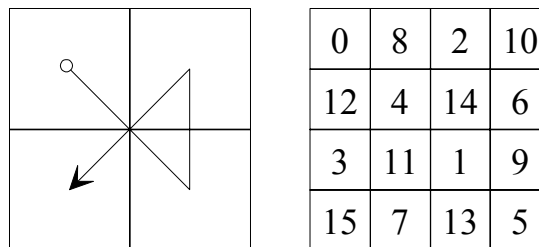
The principle of dithering.

Ordered dithering

In ordered dithering the image is processed by $n \times n$ pixel blocks. The thresholds of the pixels are given by the pseudo-random generator (see picture below). If the pixel values are in the range $[0, x_{max}]$, the threshold value $th_{i,j}$ is calculated as:

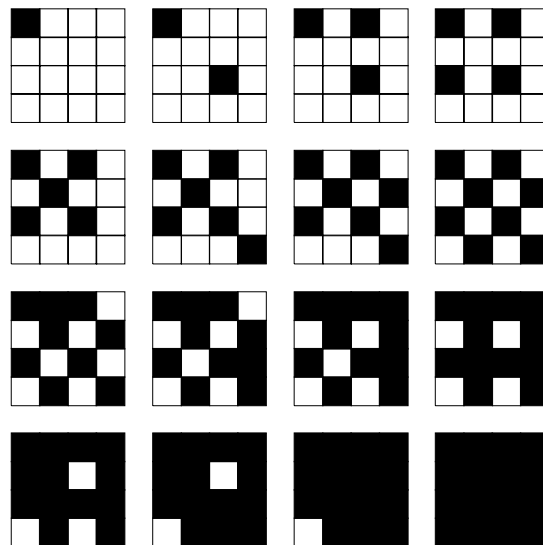
$$th_{i,j} = \frac{x_{max} + 1}{n^2} \cdot (k_{i,j} + 0.5)$$

where n is the size of the dither matrix, and $k_{i,j}$ is the corresponding pseudo-random value given by the matrix. If the pixel value $x_{i,j}$ exceeds the threshold $th_{i,j}$ then it is set to white, otherwise to black color (assuming that 0 is black).



General orientation of the ordered dither matrix (left), and threshold matrix for 4x4 ordered dithering (right).

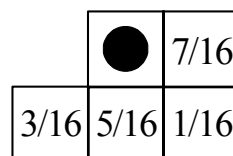
Consider a block of pixels with constant gray-value of 4 (in the scale [0, 15]). The resulting dithered block consists of 4 white and 12 black pixels having the average grey value of 4. The advantage of the ordered dither is that it can be applied in parallel, for each pixel at the same time.



Resulting patterns of ordered dithering for blocks with constant gray-level values.

Error diffusion

Error diffusion processes the image in row-major order (top-down and left-to-right). Each pixel is examined and rounded to black or white. For each pixel, the binarization error is calculated and further compensated for the following pixels. For example, if a pixel value 191 is rounded up to 255, the binarization error yields 64 (picture become 64 units too bright). This error is compensated by making the neighboring gray pixels little darker (on 64 units) so that the sum of the pixel gray values is unchanged. Specifically, the pixels neighboring to east, south, southeast and southwest are adjusted. The error is not evenly distributed to these pixels, but *weighted* using the *mask*. The best-known error diffusion method, *Floyd-Steinberg dithering*, applies the weighting mask given below.



Weighting mask of the Floyd-Steinberg error diffusion method.

Dithering examples:



Original image



Floyd-Steinberg dither



Ordered dither with 4x4 matrix



Ordered dither with 8x8 matrix

Chapter 4. Noise and Image filtering

4.1. Noise

Definition

Noise is somewhat hard to define, and many people argue about its exact definition. In general, though, we consider the *signal* to be the information-carrying part of the input and *noise* to be any additional information-less part. To be precise, we mean that it is information-less with respect to the task we're trying to do. So, for example, Mozart's music may be very rich in information, but it is noise when we're trying to listen to a conversation.

Some forms of noise are fixed and predictable: for example, a 60(50) Hz hum on a transmission wire introduced by cross-over from power lines.

In general, though, noise is a random process. As such, it is usually characterized using the language of statistics.

Quick Review of Statistics

Mean – the average or *expected value*

$$\mu = E\{x\} = \frac{\sum x}{No(x)}$$

Variance – the expected value of the squared error.

$$\sigma^2 = E\{(x - \mu)^2\} = E\{x^2\} - \mu^2$$

Standard deviation – the square root of the variance.

If we compare the strength of a signal or image (the mean of the ensemble) to the variance between individual acquired images we get a *signal-to-noise ratio*:

$$SNR = \frac{\mu}{\sigma}$$

A high signal-to-noise ratio indicates a relatively clean signal or image; a low signal-to-noise ratio indicates that the noise is great enough to impair our ability to discern the signal in it.

Additive Noise

Often, noise is additive, simply causing the resulting image to be pixel-by-pixel higher or lower than it should be. Such noise degradation can be modeled by

$$NewPixelValue = PixelValue + Noise(t)$$

(as function of the time)

Example: add random value in interval -50 .. 50. It can be either uniformly or normally distributed. In case of Normal (Gaussian) distribution: higher noise values are happened to be rarely.

Signal Dependent Noise (Noise Correlated with the Image)

Such noise can be modeled as the non-linear function of the image content.

Example: replace pixel with its neighbor at random.

Poisson Noise

One common distribution for the values of a each pixel is determined by the nature of light itself. Light is not a continuous quantity, but occurs in discrete photons. These photons don't arrive in a steady stream, but sometimes vary over time. Think of it like a flow of cars on a road – sometimes they bunch together, sometimes they spread out, but in general, there is an overall average flow.

Discrete arrivals over a period of time are modeled statistically by a *Poisson distribution*.

So, one way to decrease the effects of Poisson noise is to increase the photon count. This is not, however, as easy as it sounds. Increasing photon count may mean

Taking longer to acquire the image.

This often is not feasible when imaging moving objects. Sports photographers, astronomers, and cardiologists all have to deal with this tradeoff.

Increasing the strength of the light being imaged.

High-speed photographers may want to use brighter light to illuminate their scene. This is usually harmless, but may be something over which he has no control.

Doctors can get better medical images by increasing the strength of the radiation. Make the X-ray beam stronger, give the patient a little more radiopharmaceutical to drink, etc. The patients may object to this, however, so this sometimes is not feasible.

Use a larger area over which to acquire the photons.

If the problem is not catching enough photons, use a larger photon-catcher. This, however, means a loss of image resolution.

This tradeoff between noise and resolution is an inevitable one in vision. If you want a perfect-resolution image, use an infinitely small imaging element, but do not hold your breath waiting for a photon to hit this infinitely small hole. If you want a noise-free image, use an infinitely large imaging element, but do not hope to discern any spatial structure.

White Noise and the Frequency Domain

Noise that is normal (Gaussian)-distributed, zero-mean (does not change the average intensity level), uncorrelated, and additive is called *white noise*.

Just as white light includes all parts of the visual spectrum, white noise includes all parts of the frequency spectrum. White noise has a uniform frequency spectrum (it consist of random signals of all wavelengths)

Some forms of noise are "colored". "Blue noise", for example, has light low-frequency content and large high-frequency content.

4.2. Image Filtering

Local window operator

Filtering is an image processing operation where the value of a pixel depends on the values of its neighboring pixels. Each of the pixels is processed separately with a predefined *window* (or *template*, or *mask*). Weighted sum of the pixels inside the window is calculated using the weights given by a mask, see figure below. The result of the sum replaces the original value in the processed image:

$$f(x) = \frac{1}{k} \sum_{i=1}^9 w_i \cdot x_i$$

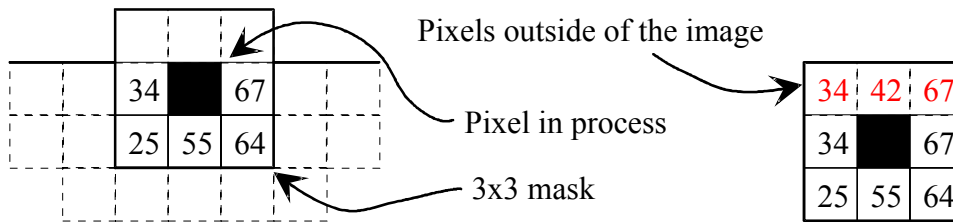
where k is chosen so that

$$k = \sum_{i=1}^9 w_i$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

General mask for filtering with a 3×3 window.

In case of border pixels, the part of the mask falling outside of the image is assumed to have the same pixel values that border pixels have, see picture below. Note that the neighboring values that are used in calculations are always taken from the original image, not from the processed image. Therefore the filtering using local window operator can be performed in parallel (all pixels at once).



Example of local window operation in case of border pixel.

Smoothing

Low-pass filtering (or *averaging filtering*, or *smoothing*) reduces the *high frequency components* (or *noise*) in the image by averaging the pixel values over a small region (block). This reduces noise and makes the image generally smoother, especially near the edges. The level of smoothing can be changed by increasing the size of the window.

Sharpening

High-pass filtering eliminates the *low frequency components* in the image. The operation can be applied in image enhancement by adding the result of the filtering to the original image and is known as *sharpening*. It enhances the pixels near edges and makes it easier to observe details in the image. The level of sharpening can be controlled by varying the magnitude of the weight for the processed pixel itself (here, the central pixel in the mask). Higher the magnitude, less the sharpening effect.

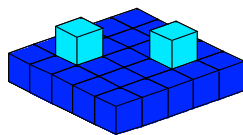
$$\text{Smooth}(A) := \text{Filter} \left[A, \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, 9 \right]$$

$$\text{Sharp}(A) := \text{Filter} \left[A, \begin{pmatrix} -1 & -1 & -1 \\ -1 & 12 & -1 \\ -1 & -1 & -1 \end{pmatrix}, 4 \right]$$

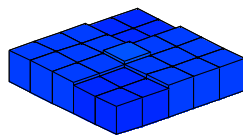
$$A := \begin{bmatrix} 10 & 10 & 10 & 10 & 10 \\ 10 & 20 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 20 & 10 \\ 10 & 10 & 10 & 10 & 10 \end{bmatrix}$$

$$\text{Smooth}(A) = \begin{bmatrix} 11 & 11 & 11 & 10 & 10 \\ 11 & 11 & 11 & 10 & 10 \\ 11 & 11 & 12 & 11 & 11 \\ 10 & 10 & 11 & 11 & 11 \\ 10 & 10 & 11 & 11 & 11 \end{bmatrix}$$

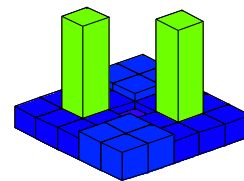
$$\text{Sharp}(A) = \begin{bmatrix} 8 & 8 & 8 & 10 & 10 \\ 8 & 40 & 8 & 10 & 10 \\ 8 & 8 & 5 & 8 & 8 \\ 10 & 10 & 8 & 40 & 8 \\ 10 & 10 & 8 & 8 & 8 \end{bmatrix}$$



A

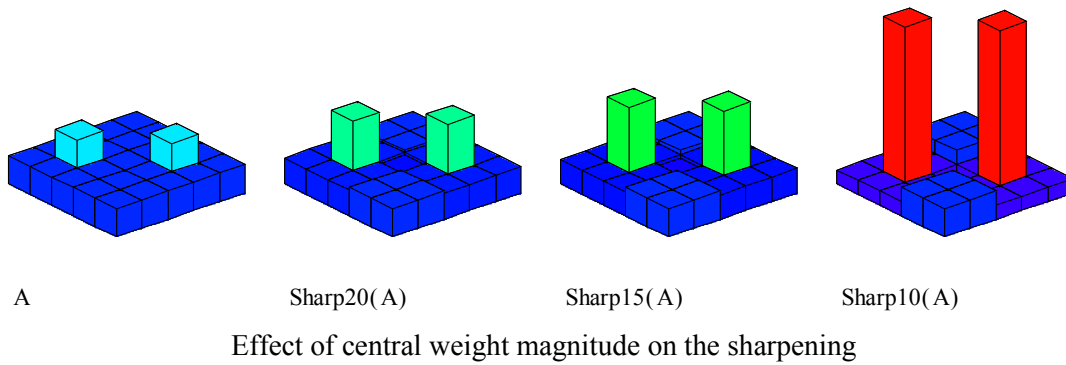


Smooth(A)



Sharp(A)

Smoothing and sharpening examples.



Median filter

Median filtering belongs to a class of *rank filters*. The pixels within a window are *ranked* (or *sorted*) and the result of the filtering is chosen according to the ordering of the pixel values. In median filtering, the new value for a pixel is the median of the pixel values in the window. The parameter of the filtering is the size and the shape of the filtering window (mask).

The median filter is used for removing additive noise. It can remove isolated impulses and at the same time, preserve the edges and other fine structures in the image. In contrary to average filtering, it does not smooth the edges.



Image with noise



Median filtered image

Chapter 5. Image Segmentation

The purpose of *Image Segmentation* is to divide the image into homogenous regions corresponding to the certain objects in the image. What are these objects (segments), depends on the application. It might be cliffs and valleys on satellite planetary image, necessary for successful space-apparatus landing. It might be text on the paper. It might be cancer cells on the radiograph.

Segmentation algorithms for monochrome images (gray-scale images) generally are based on one or two basic properties of gray-level values: *discontinuity* and *similarity*. Notice the underlying principles here: we can find places that are *different* (edges) or the *same* (regions).

The *discontinuity*-based segmentation methods produce image segmentations by finding places where the images differ, i.e. edges. Although we can find pixels whose neighborhoods indicate local differences, the challenge of edge-based methods is often to link these together somehow to produce coherent contours. The basic image processing operations for detecting discontinuities include point, line and edge detection.

Similarity or region-based methods produce segmentations by finding coherent regions directly, thus avoiding much of the difficulty of point-by-point linking of edge pixels to produce contours. However, the decision-making process for adding new pixels to the region is itself a difficult problem. The basic similarity-based segmentation algorithms are *thresholding* and *region growing*.

Formally image segmentation can be defined as a process that divides the image R into *regions* $R_1, R_2, \dots R_n$ such that:

- the segmentation is complete: $\bigcup_{i=1}^n R_i = R$
- each region is uniform
- the regions do not overlap each other: $R_i \cap R_j = \emptyset, \forall i \neq j$
- the pixels of a region have a common property: $P(R_i)$
- neighboring regions have different properties: $P(R_i) \neq P(R_j), \forall R_i, R_j$ are neighbors

5.1. Detection of discontinuities

The filtering operations described above are based on local window operator and belong to the low level image processing. Global features, such as shape and texture can be detected on the image using global image analysis based on the information gathered from the entire image. At the same time, the features such as points, lines or contours still can be detected by using only local window operator if considered as discontinuities.

The discontinuities are detected by using similar masks than in the sharpening with higher accent on the pixel positions that are included in the shape being detected. The absolute value of the weighted sum given by equation

$$f(x) = \frac{1}{k} \sum_{i=1}^9 w_i \cdot x_i, \text{ where } k = \sum_{i=1}^9 w_i$$

indicates how strongly that particular pixel corresponds to the property described by the mask: greater absolute value delivers stronger response. Note that dividing coefficient k is not used if equals to zero! The pixels whose response function exceeds a predefined threshold value will be marked.

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

Horizontal		
-1	-1	-1
2	2	2
-1	-1	-1

+45°		
-1	-1	2
-1	2	-1
2	-1	-1

Vertical		
-1	2	-1
-1	2	-1
-1	2	-1

-45°		
2	-1	-1
-1	2	-1
-1	-1	2

point detection mask

line detection masks

Edge detection

The edge detectors are based on measuring the intensity of the gradient at a point in the image. The gradient operator for two-dimensional function $I(\cdot, \cdot)$ is:

$$\nabla I = \begin{bmatrix} \frac{\partial}{\partial x} I \\ \frac{\partial}{\partial y} I \end{bmatrix}$$

As with any vector, we can compute its magnitude $\|\nabla I\|$ and orientation $\phi(\nabla I)$. The gradient magnitude gives the *amount* of the difference between pixels in the neighborhood (the *strength* of the edge). The gradient orientation gives the *direction* of the greatest change, which presumably is the direction across the edge (the edge normal).

Most edge-detecting operators can be thought of as gradient-calculators. Because the gradient is a continuous-function concept and we have discrete functions (images), we have to approximate it. Since derivatives are linear and shift-invariant, gradient calculation is most often done using local window operators. Numerous masks have been proposed for finding edges.

Prewitt operators

The Prewitt kernels (named after Judy Prewitt) are based on the idea of the *central difference*:

$$\frac{dg}{dx} \approx (g(x+1) - g(x-1))/2$$

or for two-dimensional images:

$$\frac{\partial I}{\partial x} \approx (I(x+1, y) - I(x-1, y))/2$$

$$\frac{\partial I}{\partial y} \approx (I(x, y+1) - I(x, y-1))/2$$

These kernels are, however, sensitive to noise. We can reduce some of the effects of noise by *averaging*. This is done in the Prewitt kernels by averaging in y when calculating $\frac{\partial I}{\partial x}$ and by

averaging in x when calculating $\frac{\partial I}{\partial y}$.

It will result in the following masks:

Horizontal edge	Vertical edge																		
<table border="1" style="border-collapse: collapse; width: 60px; height: 60px;"> <tr><td style="padding: 5px;">-1</td><td style="padding: 5px;">-1</td><td style="padding: 5px;">-1</td></tr> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td></tr> <tr><td style="padding: 5px;">1</td><td style="padding: 5px;">1</td><td style="padding: 5px;">1</td></tr> </table>	-1	-1	-1	0	0	0	1	1	1	<table border="1" style="border-collapse: collapse; width: 60px; height: 60px;"> <tr><td style="padding: 5px;">-1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">1</td></tr> <tr><td style="padding: 5px;">-1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">1</td></tr> <tr><td style="padding: 5px;">-1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">1</td></tr> </table>	-1	0	1	-1	0	1	-1	0	1
-1	-1	-1																	
0	0	0																	
1	1	1																	
-1	0	1																	
-1	0	1																	
-1	0	1																	
Prewitt masks																			

The gradients are calculated separately for horizontal and vertical directions. It is a common practice to approximate the overall gradient by taking the sum of the absolute values of these two:

$$\|\nabla I\| \approx \left| \frac{\partial I}{\partial x} \right| + \left| \frac{\partial I}{\partial y} \right|$$

Sobel operators

The Sobel kernels (named after Irwin Sobel) also rely on central differences, but give greater weight to the central pixels when averaging:

Horizontal edge	Vertical edge																		
<table border="1" style="border-collapse: collapse; width: 60px; height: 60px;"> <tr><td style="padding: 5px;">-1</td><td style="padding: 5px;">-2</td><td style="padding: 5px;">-1</td></tr> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td></tr> <tr><td style="padding: 5px;">1</td><td style="padding: 5px;">2</td><td style="padding: 5px;">1</td></tr> </table>	-1	-2	-1	0	0	0	1	2	1	<table border="1" style="border-collapse: collapse; width: 60px; height: 60px;"> <tr><td style="padding: 5px;">-1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">1</td></tr> <tr><td style="padding: 5px;">-2</td><td style="padding: 5px;">0</td><td style="padding: 5px;">2</td></tr> <tr><td style="padding: 5px;">-1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">1</td></tr> </table>	-1	0	1	-2	0	2	-1	0	1
-1	-2	-1																	
0	0	0																	
1	2	1																	
-1	0	1																	
-2	0	2																	
-1	0	1																	
Sobel masks																			

5.2. Region growing

Region growing (or *pixel aggregation*) is the method to locate the homogenous area in an image. Region growing starts from an *initial region*, which can be a group of pixels (or a single pixel) called *seed points*. The region is then extended by including new pixels into it. The included pixels must be neighboring pixels to the region and they must satisfy a *uniform criterion*. If the pixel satisfies this criterion it is added to the region otherwise skipped. The growing continues until a *stopping rule* takes effect or until no more pixels satisfy *uniform criterion*.

All of the three parameters of the method depend on the application, and their choice will determine the algorithm in particular. In the infrared images in military applications, the interesting parts are the areas that are hotter compared to their surroundings. The natural choice for the seed points in this case, will be the brightest pixel(s) in the image. In interactive applications, the seed point(s) can be defined by the user.

The uniform criterion might be the local or global one. The following criteria are commonly used in practical applications. *Global ones*: the *difference* between processed pixel and the average value of the region or seed point, the *variance* of the region if the pixel is added to it. *Local ones*: *maximal difference* between processed pixel and the nearest neighboring pixel inside the region, the *variance* of the local neighborhood, *gradient* of the pixel (should be low enough to avoid edges).

The growing will stop automatically when there are no more pixels that meet the uniform criterion. The stopping can also take effect whenever the size of the region gets too large.

INITIALIZATION:

choose *seed point* x_0 ;

initial region $R_0 = \{x_0\}$;

$k = 0$;

ITERATE:

$R_{k+1} = R_k \cup \{x\}, \quad \forall x \in \text{Neighborhood}(R_k) \mid P(x),$

where $P(x)$ is the uniform criterion;

$k = k + 1$;

UNTIL *stopping rule* takes effect (eg. $|R_k| > m$)

Region-growing algorithm outline

5.3. Thresholding

Image segmentation involves separating an image into homogenous regions corresponding to objects. The simplest common property that pixels in a region can share is intensity. So, a natural way to segment the image is to apply *thresholding*, the separation of light and dark regions. Bright region may represent the object and dark region – the background, or vice versa.

Thresholding converts grayscale image to binary turning all pixels below some threshold value to zero and all pixels above that value to one.

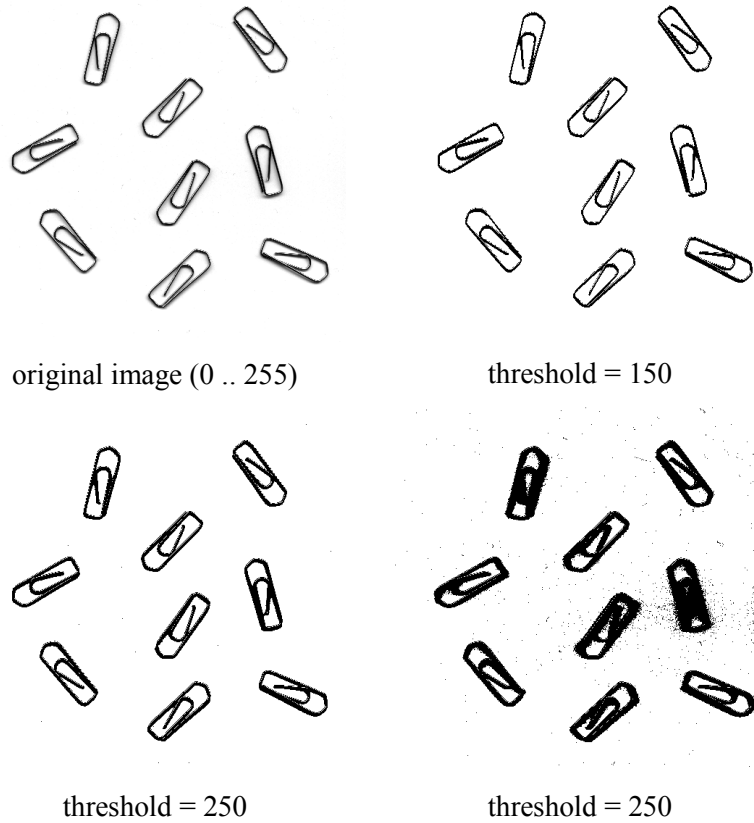
$$f(x_{i,j}) = \begin{cases} 1 & \text{if } x_{i,j} > T \\ 0 & \text{if } x_{i,j} \leq T \end{cases}$$

Problems with Thresholding

The major problem with thresholding is that we consider only the intensity, not any relationships between the pixels. There is no guarantee that the pixels identified by the thresholding process are contiguous.

We can easily include extraneous pixels that are not part of the desired region, and we can just as easily miss some pixels originally belonging to the region, especially the isolated pixels near the region boundaries. These effects get worse with the noise because it is more likely that pixel intensity does not represent the normal intensity in the region.

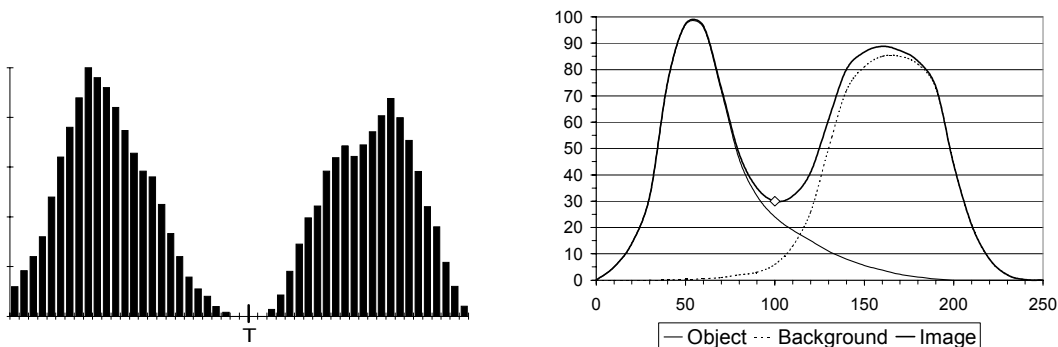
Therefore, when using thresholding, we have typically have to *play* with it, sometimes losing too much of the region and sometimes getting too many extraneous background pixels, see picture below. The shadows of objects in the image can also cause problems and be mistakenly included as part of a dark object on a light background.



We will consider here three methods for calculating the threshold parameter.

Peaks and Valleys (direct thresholding)

One extremely simple way to estimate a threshold value is to find in the image histogram the points of local maxima (peaks) and then find the minimum (valley) between them. This method may work only in ideal case, when there are two distinct regions in an image, whose intensity values do not overlap. In practice, there might be pixels with equal intensity values in both object and background regions, and estimated by this method threshold value might be not necessary the optimal one (see picture below)

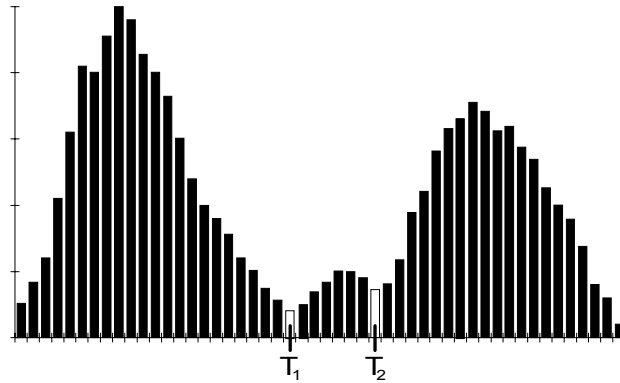


Histogram partitioned with the threshold.

Threshold inaccuracy

Hysteresis thresholding.

This method has been designed to reduce the misclassification of pixels near the region boundaries. First, the two thresholds T_1 and T_2 are calculated for the image. Then all pixels below the first threshold value T_1 are turned to zero and all pixels above the second threshold T_2 are turned to one. Finally, the pixels, whose values fall in between of two thresholds, are turned to one if majority of their adjacent pixels are turned to one on the previous step, otherwise they are turned to zero.



Histogram partitioned with two thresholds.

Iterative thresholding

This method aims at minimizing the pixel variances inside the two segments. It calculates the threshold value iteratively as follows. The algorithm starts by calculating the average value of the pixels as a tentative threshold T . The image is then segmented into two regions using this threshold and the average values (\bar{x}_1 and \bar{x}_2) are calculated for each region. New threshold value T is taken as the average of \bar{x}_1 and \bar{x}_2 , and the segmentation is redone. The process is then repeated until the threshold value T (and therefore segmentation) does not change. The iterative method does not necessarily give the optimal threshold value but is practical, and in most cases sufficiently well.

INITIALIZATION:

initial threshold $T = \bar{x}$

ITERATE:

segment the image using threshold T ;

calculate segment averages: $\bar{x}_1 = \frac{1}{n_1} \sum_{x_i \leq T} x_i$ and $\bar{x}_2 = \frac{1}{n_2} \sum_{x_i > T} x_i$,

where $n_1 = |\{x | x \leq T\}|$ and $n_2 = |\{x | x > T\}|$;

recalculate threshold: $T = \frac{\bar{x}_1 + \bar{x}_2}{2}$;

UNTIL T does not change.

Iterative thresholding algorithm outline.

Chapter 6. Binary image processing

6.1. Resolution reduction

Resolution reduction of binary images is similar to that of grayscale images.

The image of size $N \times M$ pixels is usually converted to the image of $N/2 \times M/2$ pixels twice reducing its size. The pixel in low-resolution, resulting image is chosen regarding the color of pixels in the corresponding block in high-resolution, original image. Following methods are applied:

a) *logical-sum* – output pixel is black if any of input pixels in the block is black;

$$y_{i,j} = x_{2i,2j} \vee x_{2i+1,2j} \vee x_{2i,2j+1} \vee x_{2i+1,2j+1}$$

b) *majority* – output pixel is black if there is more or equal number of black pixels in the block than white pixels;

$$y_{i,j} = \begin{cases} 1, & \text{if } (x_{2i,2j} + x_{2i+1,2j} + x_{2i,2j+1} + x_{2i+1,2j+1}) \geq 2 \\ 0, & \text{otherwise} \end{cases}$$

c) *JBIG resolution reduction* algorithm, in which the value of the target pixel is calculated as a linear function of the preceding neighboring pixels from high and low-resolution layers. The already-committed pixels at the low-resolution layer participate in the sum with negative weights that exactly offset the corresponding positive weights, Specifically:

$$L = 4x_{22} + 2(x_{23} + x_{32}) + x_{33} + (x_{11} - y_{00}) + 2(x_{21} - y_{10}) + (x_{31} - y_{10}) + 2(x_{12} - y_{01}) + (x_{13} - y_{01})$$

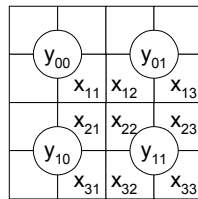
Or equally,

$$L = 4x_{22} + 2(x_{12} + x_{21} + x_{23} + x_{32}) + (x_{11} + x_{13} + x_{31} + x_{32}) - 3(y_{01} + y_{10}) - y_{00}$$

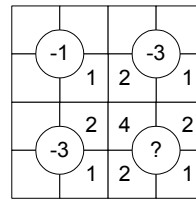
It is made to preserve the overall grayness of the image. If black and white pixels are equally likely and the pixels are statistically independent, the expected value of the target pixel is 4.5. It is chosen to be black if the value is 5 or more, and white if it is 4 or less:

$$y_{11} = \text{black}, \quad \text{if } L > 4.5$$

Pixel positions:

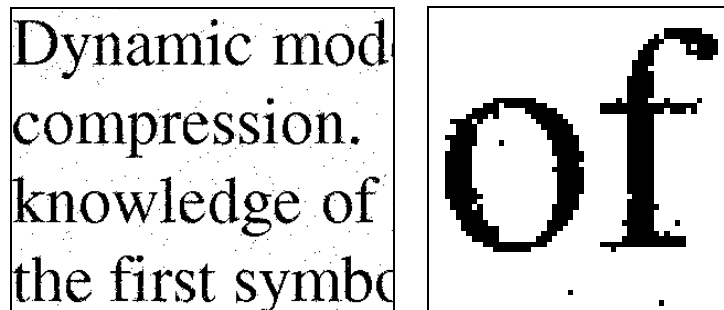


Resulting pixel weights:



6.2. Image filtering

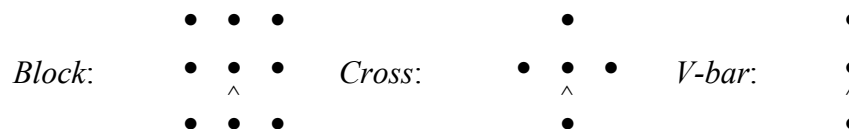
The binary document images are also subject to noise degradation as color or grayscale images. The noise appears on images as black and white pixels scattered at random (additive component) and as pixels randomly swapped with their neighbors (content-dependent component, result of image quantization in the digitizing device).



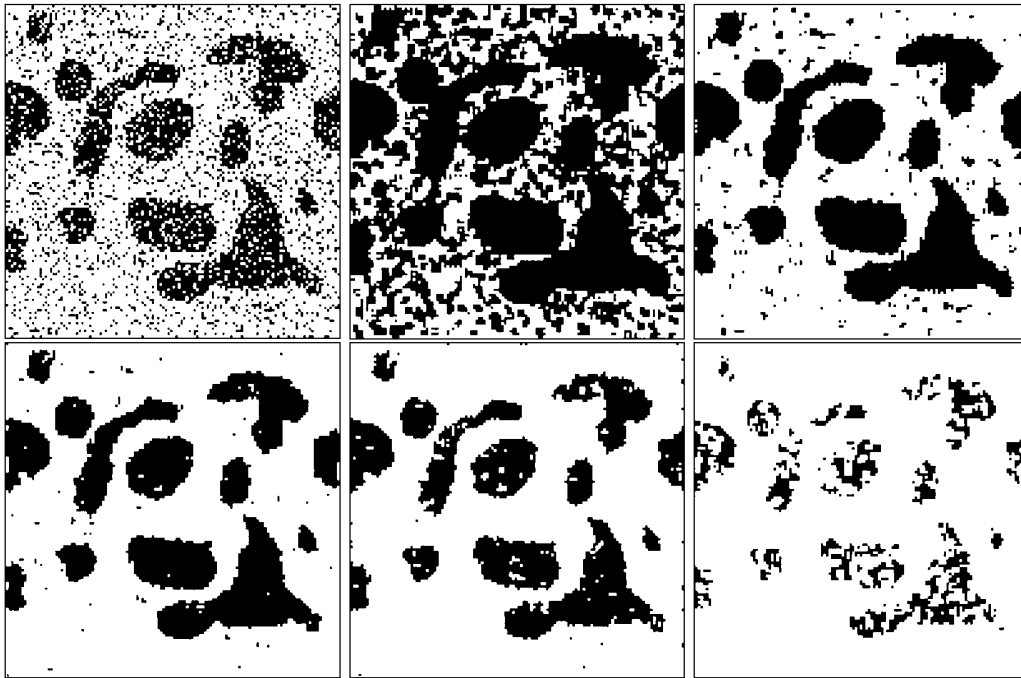
Sample of the document image containing noise.

We will consider the following typical binary image filters:

- **Annular filter** – turns the pixel over, if its adjacent neighbors are all of the opposite color. Adjacent pixels defined by *incidence matrix* or *mask* :



- **Soft annular filter** with rank k – preserves the pixel if it has at least k same-colored neighbors, otherwise turns it over.
- **Median filter** – defines pixel color as dominant color among pixels from local window. Local window is again defined by mask. For binary images is the same as *average filter*.
- **k -rank operator** – turns the pixel to foreground color (i.e. black) if at least k pixels from local window of this color. If the mask size n is odd, and $s = \frac{1}{2}(n + 1)$, the s -rank operator equals to median filter.



Original image (upper-left) and k -rank operator for mask block and $k = 2, 4, 5$ (median), 6, and 8.

Noise removal problems

Existing filtering procedures observe only local neighborhood of processed pixel and make the decisions depending on the distribution of black and white pixels in it or matching the neighborhood with a priori defined masks. These filtering schemes can efficiently remove only additive noise and restore binary patterns: isolated pixels removed by annular filters, when smoothing and morphological operations restore binary patterns. Content-dependent noise (jagged line contours) is problematic to remove. Smoothing can remove content-dependent noise but does not preserve fine structures. The methods based on global features, such as statistical or feature-based techniques tuned for specialized image types (textual images, line-drawings, etc) must be applied to restore the image from content-dependent degradation.

Chapter 7. Global methods – high level image processing

Under global methods, we understand the methods based on information gathered from the entire image, which could not be utilized by local window operators. The extracted features depend on the image type and applications. These may be useful in difficult image processing operations such as removal of the content-dependent noise, and also in applications containing the image analyzing and understanding procedures, such as image segmentation, skew-detection, indexing, and recognition. We will consider binary images in following.

7.1. Statistical context-based filtering

The idea of statistical context-based filtering is to achieve better selection of the noise pixels through applying statistical context modeling and measuring information content of the entire document image.

Context-based modeling

A binary document image can be considered as a message, generated by an information source. The idea of statistical modeling is to describe the message symbols according to the probability distribution of the source alphabet (binary alphabet, in our case). The information content of a single pixel in the image is measured by the *entropy* (introduced by C. Shannon in 1948):

$$H(x) = -\log_2 p(x),$$

where x is the pixel and $p(x)$ its probability. The higher is the symbol probability, the lower is its entropy. Deterministic source carries no information and may be totally replaced by the model. Entropy of the entire image is calculated as the average entropy of all pixels:

$$H = -\frac{1}{n} \sum_{i=1}^n \log_2 p(x_i),$$

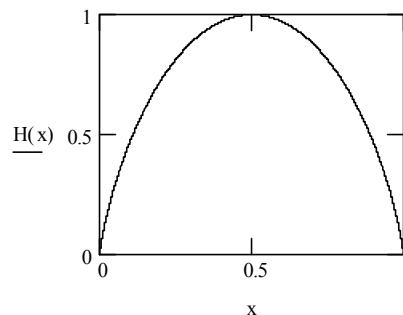
where n is the total number of pixels in the image.

If the probability distribution of black and white pixels in the image is a priori known, the entropy equation simplifies to:

$$H = -p_W \cdot \log_2 p_W - p_B \cdot \log_2 p_B,$$

where p_B and p_W are the probabilities of the black and white pixels, respectively.

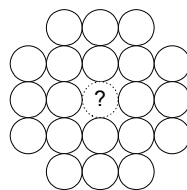
Entropy gives the optimal expected number of code bits in respect to the model required to code a single pixel on average.



The pixels in an image form geometrical structures with appropriate spatial dependencies. The dependencies can be localized to a limited neighborhood, and described by *context-based statistical model*. In this model, the pixel probability is measured relative to the *context C* by counting the number of black (n_B^C) and white (n_W^C) pixels appeared with that context in the entire image.

A context is defined by the combination of neighboring pixels within the local template:

20-pixel filtering
context template



The entropy $H(C)$ of a context C is defined as the average entropy of all pixels within the context. Entropy is calculated as:

$$H(C) = -p_W^C \cdot \log_2 p_W^C - p_B^C \cdot \log_2 p_B^C.$$

Here, p_B^C and p_W^C are the corresponding probabilities of the black and white pixels relative to the context, calculated as:

$$p_W^C = \frac{n_W^C}{n_W^C + n_B^C}, \quad p_B^C = \frac{n_B^C}{n_W^C + n_B^C},$$

where n_B^C and n_W^C are the number of black and white pixels appeared within the context C in the entire image

A context with skew probability distribution has smaller entropy and therefore smaller information content. The entropy of N -level context model is the weighted sum of the entropies of individual contexts:

$$H_N = -\sum_{j=1}^N p(C_j) \cdot (p_W^{C_j} \cdot \log_2 p_W^{C_j} + p_B^{C_j} \cdot \log_2 p_B^{C_j}).$$

Filtering

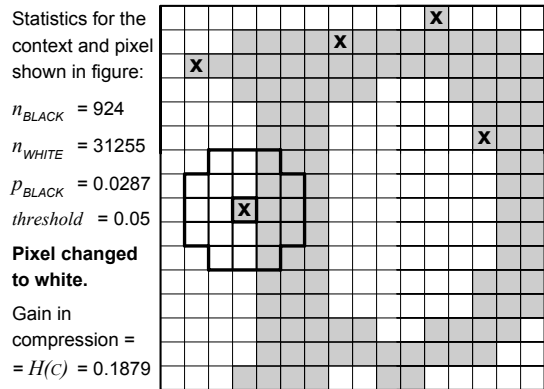
Statistical context filtering is based on determining the statistical content of the image using the context-based modeling, and inverting the pixels with low probability values using the assumption that they are noise. The filtering process consist of two phases, each requires one pass over the image.

In the *analyzing phase*, the context modeling with a 20-pixel filtering template is applied for input image, and the number of black (n_B^C) and white (n_W^C) pixels for each context C and respective probabilities are calculated:

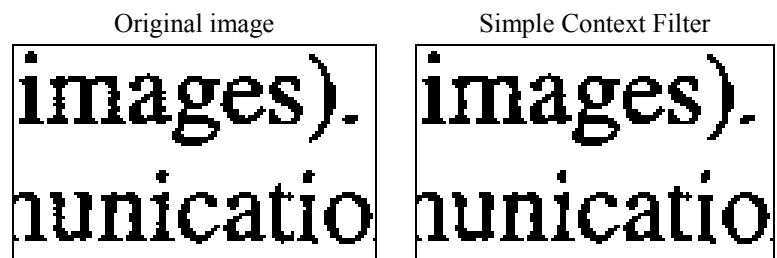
$$p_W^C = \frac{n_W^C}{n_W^C + n_B^C}, \quad p_B^C = \frac{n_B^C}{n_W^C + n_B^C}.$$

After the analyzing, the contexts are categorized as *low information contexts*, if the probability of either black or white pixel (p_B^C or p_W^C) does not exceed a predefined threshold value (e.g. 0.05). The probabilities are calculated on the base of observed pixel frequencies. The less probable pixels in low information contexts are classified as *rare*, and most probable as *common* pixels.

The output image is generated in the *filtering phase* when all rare pixels in low entropy contexts are changed to opposite color. The threshold value is a trade-off between amount of removed noise and image degradation caused by filtering. (Remember that there exist no ideal filtering operation, and all filters degrade the image content).



Example of the context filter. Original image content is shown in gray, and inverted pixels are marked with 'x'.



Example of the filtering with threshold 0.25

7.2. Feature-based filtering

Content-dependent noise can be removed from digitized line drawings using feature-based filtering. This filtering technique is based on the semantic image modeling, utilizing the global spatial dependencies in the image. The line drawings consist mainly of straight-line elements, and global information is gathered by extracting *line features*, which are utilized in the filtering. The filtering is applied usually as a part of an image compression system so that feature extraction and filtering are considered as preprocessing steps before the compression. The noise removal improves an image quality and restores the loss in the compression ratio caused by noise.

Hough transform

The line features are extracted from the image using Hough transform (after P. Hough, who has patented this technique in 1962). Let us suppose an image consisting of only several samples of a straight line. Hough proposed a method to find the line among these samples. Considering a point (x_i, y_i) , there is an infinite number of lines passing through this point, however, they all can be described as

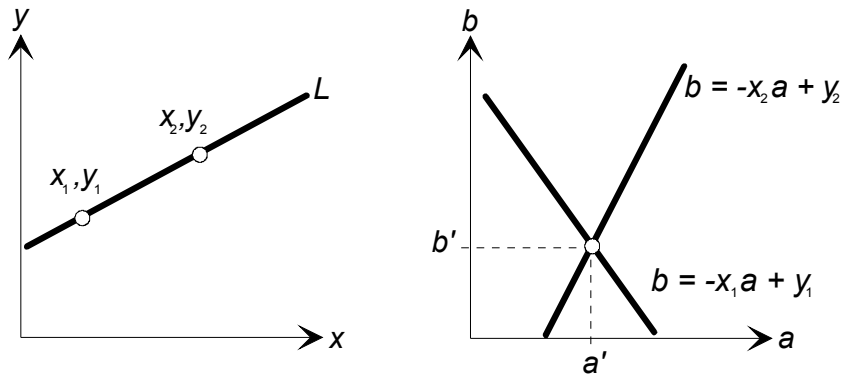
$$y_i = a \cdot x_i + b. \quad (0.1)$$

It means that all the lines passing (x_i, y_i) defined by two parameters (a, b) can be expressed as

$$b = -x_i \cdot a + y_i. \quad (0.2)$$

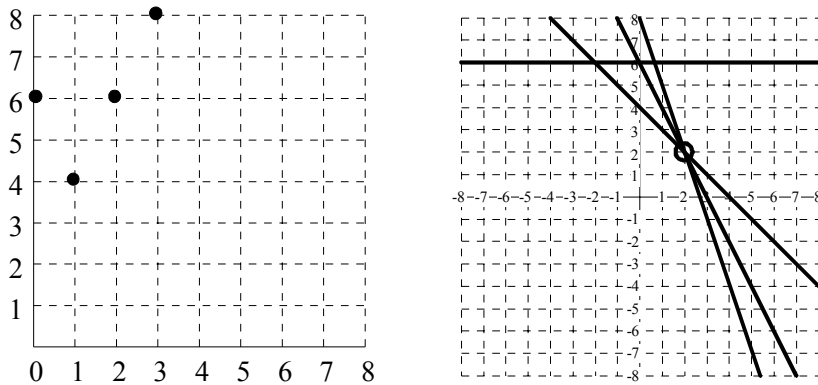
The Hough transform is a process where each pixel sample (x, y) in the original pixel space is transformed to a curve in the parameter space, representing all-possible lines passing this pixel. If there is evidence of line presence in the image – that is, when pixel samples are

located along the line, the Hough curves will intersect in the same point (a', b') , providing the parameters value for the line in question.



Hough transform: pixel xy -space (left), and parameter ab -space (right).

To implement the Hough transform, the parameter space is represented as a $k \times k$ accumulator array where k can be tuned according to the image size. In each cell of the matrix, there is a counter of how many parametric curves are crossing that point. Each curve increases the counter of the cells lying along its way. The lines are extracted from the positions in the array, exceeding predefined threshold parameter.



Small example of Hough transform.

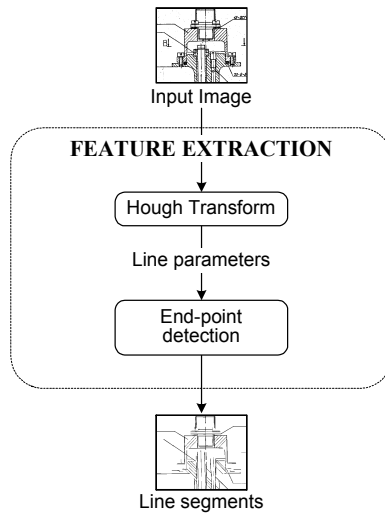
A problem in this implementation is that both the slope (a) and intercept (b) approach infinity as the line approaches the vertical. One way around this difficulty is to use the normal representation of a line:

$$x \cdot \cos \theta + y \cdot \sin \theta = \rho,$$

where ρ represents the shortest distance between origin and the line, and θ represents the angle of the shortest path in respect to the x -axis. Their corresponding ranges are $\rho \in [0, \sqrt{2}D]$, and $\theta \in [-90^\circ, 90^\circ]$, where D is the distance between corners of the image.

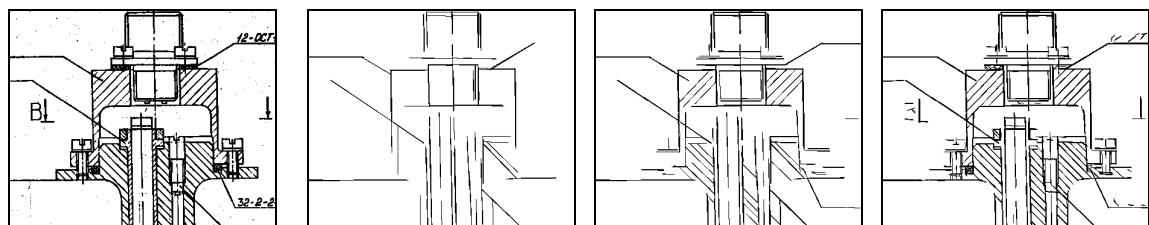
Line features extraction

The entire feature extraction procedure is summarized in the picture below. The motivation is to find rigid fixed length straight lines in the image. The extracted line segments are represented as their end-points and encoded into the feature file.



Block diagram of the feature extraction.

The Hough transform is capable to determine the location of a line as a linear function but it cannot resolve the end-points of the line. In fact, HT does not even guarantee that there exists any finite length line in the image but it only indicates that the pixels (x, y) along $y = a \cdot x + b$ may represent a line. The existence of a line segment must therefore be verified. The verification is performed by scanning the pixels along the line and checking whether they meet certain criteria. For example, one can use the scanning width, the minimum number of pixels, and the maximum gap between pixels in a line as the selection criteria. If predefined threshold values are met, a line segment is detected and its end-points are stored for later use. The features extracted with different parameter setup are shown in the following picture.



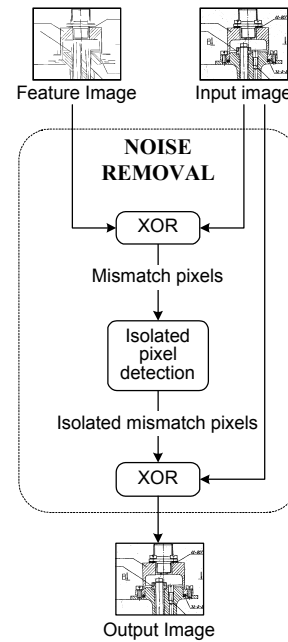
number of segments	117 line segments	289 line segments	752 line segments
min segment length	150	70	30
max segment width	1	1	1
max length of single gap	2	2	3
accumulator threshold	20	20	17

Example of the feature images made using different parameter setup.

Filtering

The filtering is based on the following noise removal procedure. First, an equal size *feature image* is created from the extracted line segments to approximate the original image. Feature image consists of non-local information of the image that cannot be utilized using local spatial filters.

Then, a *mismatch image* is constructed from the differences between the original and the feature image. Small isolated marks, such as single isolated pixels and pixel groups up to two, are detected on the mismatch image and the corresponding pixels in the original image are changed. This removes additive noise and smooths edges along the detected line segments. The quality of the filtering is controlled by allowing only isolated groups of noise pixels to be changed. Objects that are not recognized by the feature extraction process are left untouched.



Block diagram of the noise removal procedure

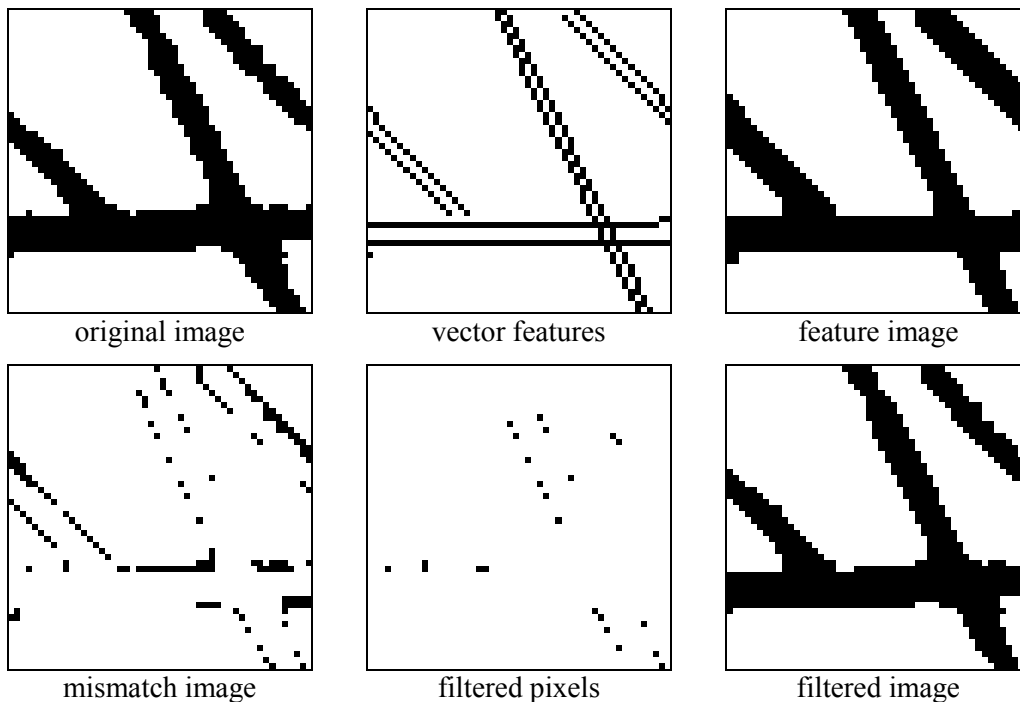


Illustration of the feature-based filtering procedure

Appendix: Terminology overview.

Computer Vision is the set of processes created in order to give the computer system ability to interpret visual information. Usually, Computer Vision simulates the Human Visual System wherever is possible (HVS is not yet totally researched).

Computer Vision Hierarchy:

1. *Image Processing*: image \rightarrow image
2. *Image Recognition*: image \rightarrow data
3. *Scene Recognition*: data \rightarrow scene description

Image Processing – in general, the conversion of one image to another with modification of the content or parameters (such as resolution, size, brightness, color, etc).

Hierarchy of Image Processing operations:

1. *Local operations* (low level): local window operators, filtering, etc.
2. *Global operations* (high level): statistical methods, thresholding, thinning (skeleton).

Image Analysis – the extraction of useful information from the image for further processing (i.e. histogram)

Hierarchy of the features extracted from the image:

1. *Local features*: features specific for single pixel (brightness, color) or local area (sharpness, as area contrast)
2. *Global features*: features specific for image as whole or to objects in the image (shape, texture, statistics – histogram)

Image Restoration – the image processing operation applied to the image whose quality has been degraded (e.g. noise) and resulting in an image, which is more similar to the original non-degraded (noise-free) image. It is defined as:

$$R(x): A \rightarrow B, \text{ so that } \rho(B, \Theta) < \rho(A, \Theta),$$

where $\rho(\cdot, \cdot)$ is the image similarity measure, and Θ is the original non-degraded image.

Image Enhancement – is the image processing operation resulting in an image more suited for some image operation, such as analysis, compression, recognition or another processing operation. The operation is defined as:

$$E_p(x): A \rightarrow B, \text{ so that } F_p(B) > F_p(A),$$

where, $P(\cdot)$ is the posterior image operation and $F_p(\cdot)$ is the efficiency (eg. compression rate or number of recognition errors). The image may be also enhanced for viewing purposes.