Name: ..... Neptun: .....

PPCU FIT ● fall semester 2015

BASIC IMAGE PROCESSING ALGORITHMS

ORAL EXAM

18 December 2015

PÁZMÁNY PÉTER CATOLICH UNIVERSITY

FACULTY OF INFORMATION TEHCNOLOGY AND BIONICS

Important information

Dear Candidate!

This booklet is made for the Oral Exam in Basic Image Processing Algorithms. The booklet contains the titles of the exam topics as well as the detailed form of these. In some cases you may find errors or typos in the text. If this happens please, feel free to report them on my website.

We wish you a successful preparation!

Edited by:
Márton Bese NASZLADY– 2015

Table of Contents

Exam Topics

Topic 1 Introduction to Human Vision, Digital representation of an image, Color Spaces

Introduction to Human Vision

The human vision gives us the ability to process visual stimulus, to be able to detect and interpret information from visible light (build a representation of the surrounding environment). The ultimate goal of computer vision is to build a system that is capable of seeing as a human can (or even better).

The Physiology of the Human Eye

Photoreceptor cells of the retina

Rods:

- sensitive to intensity, but not color
- form blurred images
- rods are more sensitive to light than cones
- at low levels of illumination the rods provide a visual response called scotopic vision

Cones:

- color sensitive: 3 types, each maximally sensitive to one of three different wavelengths
- form sharp images,
- cones respond to higher levels of illumination; their response is called photopic vision

Information transfer from the retina to the brain

The eye contains about 6 million cone and 100-120 million rod cells distributed over the retina. The density of the cones is greatest at the fovea. The optic nerve bundle contains on the order of 800,000 nerve fibers. Therefore, the rods and cones must **be interconnected to nerve fibers on a many-to-one basis.**

The Visual Pathway

Parts of the Visual Patway

Optic Nerve – The information from the retina is transmitted to the brain.

Optic Chiasm – The information coming from both eyes is combined and split according to the visual fields.

Optic Track – Transfers the information from each visual fields to the LGN.

LGN – primary relay center for visual information received from the retina of the eye

Optic Radiation – The optic radiation is a collection of axons from relay neurons in the LGN of the thalamus carrying visual information through two divisions (called Upper and Lower division) to the visual cortex

Visual Cortex – It is the largest system in the human brain, it is responsible for processing the visual image.

Visual Cortex

The **dorsal stream,** sometimes called the **"Where Pathway" or "How Pathway",** is associated **with motion, representation of object locations,** and **control of the eyes and arms,** especially when visual information is used to guide saccades or reaching.

The **ventral stream,** sometimes called the **"What Pathway"**, is associated with **form recognition and object representation.** It is also associated with storage of long-term memory.

A Few Properties of the Human Vision

Contrast Sensitivity
The response of the eye to changes in the intensity of illumination is nonlinear. Consider a patch of light of intensity $I + \Delta I$ surrounded by a background intensity I. Over a wide range of intensities, it is found that the ratio $\Delta I/I$, called the **Weber fraction,** is nearly constant at a value of about 0.02. This does not hold at very low or very high intensities. Furthermore, contrast sensitivity is dependent on the intensity of the surround.

Many image processing systems assume that the eye's response is logarithmic instead of linear with respect to intensity:

$$\log(I + \Delta I) - \log(I) = \log\left(\frac{I + \Delta I}{I}\right) = \log(1 + c) = \text{const}$$

Lateral Inhibition
The response of receptor *A* to illumination is decreased, if the nearby receptors *B* are also illuminated.

Chromatic Adaption
An object may be viewed under various conditions; it may be illuminated by sunlight, the light of a fire, or electric light. In all of these situations, human vision perceives that the object has the same color.

Illusions

The visual system is optimized to process natural images (through evolution). It is faced with an ill-posed problem:

- Ambiguity due to projection from 3D to 2D image
- Uncertainty due to incomplete knowledge of the environment
- Uncertainty due to noise in photoreceptors and neurons

The visual system relies on a set of assumptions to solve this ill-posed problem

- Assumptions presumably learned via evolution
- Assumptions tailored for the natural visual world
- Assumptions cause illusions/failures under impoverished conditions

Illusions can provide insights into the brain's assumptions.

Digital Representation of an Image

A **digital image** is discreet representation of a continuous measurement, usually a **2 or 3 dimensional array.** An element of this array is a **pixel** (picture element). A pixel has a **position** and an **intensity value.** A digital image is **discretized** both in space and intensity:

- Spatial discretization is referred to as **sampling**
- Intensity discretization is referred to as **quantization**

Sampling
Sampling is the reduction of a continuous signal to discrete signal. A finite set of values (called **samples**) are selected to represent the original continuous signal.
In case of 2D signals (images) a grid is used for sampling. The grid points will be represented as pixels. The frequency of the sampling defines:

- How many grid points we have
- What is the resolution of the image
- How detailed the discretized image is

Sampling usually leads to information loss. The sampling frequency determines how much information we lose. We have to decide what is the smallest detail that we still want to keep.

Quantization

Intensity discretization is referred to as quantization. The digital image quality is highly depending on how many **bits** we use for coding the discreet intensity values:

- black&white coded on 1 bit
- grayscale coded on 2/4/8/16/24/32 bits

Color Spaces

Humans can distinguish thousands of color shades and intensities, but only a few dozens of gray. Color can be useful descriptor for image segmentation, tracking, detection etc.

Color Characteristics

Brightness used to describe color sensation (it is similar to intensity of achromatic light)

Hue it indicates the dominant wavelength in the mixture of light waves

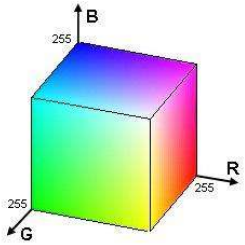
Saturation relative purity or amount of white light in the mixture

Color Models

They specify a coordinate system and a subspace within that system, where each color is represented by a single point.

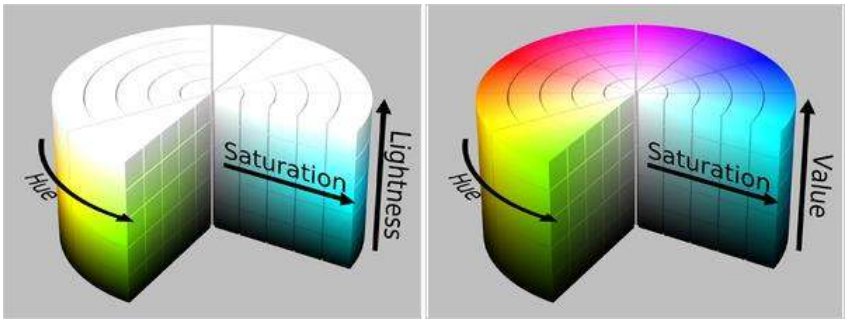
RGB
Channels: Red, Green, Blue

Most common color model. All components are depending on luminosity. All channels needs to be coded with the same bandwidth. Changing the intensity level is not efficient, all 3 channels has to be modified.

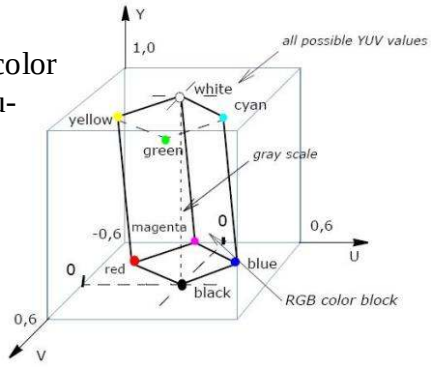


HSL, HSV
Channels: Hue, Saturation, Lightness or Value

The components are more intuitive. Hue is the angle around the central vertical axis (defined in degrees). Saturation is the distance from the central axis. Lightness or Value is the height.



Y'UV
Channels: Luma (Y'), Chrominance components (U, V)



It is used in compression, in the PAL and SECAM composite color video standards. Luma is coded in a separate channel. It takes human perception into account allowing reduced bandwidth for chrominance components.

Conversion from RGB to Y'UV:
$Y' = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$
$U = 0.492 \cdot (B - Y')$
$V = 0.877 \cdot (R - Y')$

CMY, CMYK
Channels: Cyan, Magenta, Yellow, (Black = Key)

This color space is used in printing. It is based on the subtractive color model: describes what kind of inks need to be applied so the reflected light produces the given color.

The CMYK model contains black as fourth channel because the black produced by the mixture of CMY is not really black in practice.

CIE
Channels: X (mix of cone response curves), Y (luminance), Z (blue stimulation)

The CIE color model is based on how humans perceive color. It was developed to be completely independent of any device.

Topic 2 2D Convolution and its Applications, Canny Edge Detector, Hough Transformation

2D Convolution and its Applications

Unit impulse function
The 2D unit impulse function (Delta function) on $\mathbb{Z}$ as follows:

$$\delta(n_1, n_2) = \begin{cases} 1, & n_1 = n_2 = 0 \\ 0, & \text{otherwise} \end{cases}$$

For any 2D function $x(n_1, n_2)$:

$$x(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \delta(n_1 - k_1, n_2 - k_2) \cdot x(k_1, k_2)$$

Convolution
Impulse response is the output of an LSI transformation if the input was the Delta function. If T is an LSI system, then we can define convolution as follows:

$$y(n_1, n_2) = x(n_1, n_2) * h(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(n_1 - k_1, n_2 - k_2) \cdot x(k_1, k_2)$$

The Properties of Convolution
- Commutative: $f * g = g * f$
- Associative: $f * (g * h) = (f * g) * h$
- Distributive: $f * (g + h) = f * g + f * h$
- Associative with scalar multiplication: $\alpha(f * g) = (\alpha f) * g$

2D Convolution in Practice
In practice both the kernel and the image have finite size.
Let h and $\hat{h}$ be $(2r_1 + 1) \times (2r_2 + 1)$ sized kernels, where $\hat{h}$ is the 180° rotated version of h.

$$h = \begin{bmatrix} a_{-r_1,-r_2} & \cdots & a_{-r_1,r_2} \\ \vdots & \ddots & \vdots \\ a_{r_1,-r_2} & \cdots & a_{r_1,r_2} \end{bmatrix}, \qquad \hat{h} = \begin{bmatrix} a_{r_1,r_2} & \cdots & a_{r_1,-r_2} \\ \vdots & \ddots & \vdots \\ a_{-r_1,r_2} & \cdots & a_{-r_1,-r_2} \end{bmatrix}$$

$$y(n_1, n_2) = \sum_{k_1=-r_1}^{r_1} \sum_{k_2=-r_2}^{r_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) = \sum_{k_1=-r_1}^{r_1} \sum_{k_2=-r_2}^{r_2} \hat{h}(k_1, k_2) x(n_1 + k_1, n_2 + k_2)$$

Size of the Convolved Image
In general, if the size of the input image is $(A \times B)$, the size of the kernel is $(C \times D)$ then the size of the output image will be $(A + C - 1) \times (B + D - 1)$.

Boundary Effects
The convolution along the edges is not possible due to missing pixels. The original image can be padded so the convolution is possible. There are four main types of image padding:
- zero padding
- mirroring
- circular padding
- repeating border

Applications

The possible application of convolution:

- Smoothing/Noise reduction
- Edge detection
- Edge enhancement

Depending on the task the sum of the elements of the kernel matrix can be different:

1 → smoothing, edge enhancement
0 → edge detection

Smoothing/Blurring

Simple average

$$\frac{1}{N^2}\begin{bmatrix} 1_{1,1} & \cdots & 1_{1,N} \\ \vdots & \ddots & \vdots \\ 1_{N,1} & \cdots & 1_{N,N} \end{bmatrix}$$

Parameters:

- size of the window

Gaussian blur

$$\begin{bmatrix} 0.0947 & 0.1183 & 0.0947 \\ 0.1183 & 0.1478 & 0.1183 \\ 0.0947 & 0.1183 & 0.0947 \end{bmatrix}$$

Parameters:

- size of the window
- standard deviation (σ)

Edge Detection

Edge Location on the image where intensity changes sharply (usually at the contour of objects)

So we are searching for places where the gradient of the 2D function is high. The main types of edge detection are:

- <u>first order derivative</u>

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

since the smallest $\mathrm{d}x$ and $\mathrm{d}y$ are both 1. So the kernels for the gradient calculation with convolution (Prewitt kernels):

$$[-1 \quad 1] \xrightarrow{\text{better localization}} [-1 \quad 0 \quad 1] \xrightarrow{\text{noise reduction}} [-1 \quad 0 \quad 1]\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \xrightarrow{\text{better localization}} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{\text{noise reduction}} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}[1 \quad 1 \quad 1] = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- <u>second order derivative</u>

$$\frac{\partial^2 f}{\partial x^2} = \lim_{d\to 0} \frac{\frac{\partial f}{\partial x}(x+d) - \frac{\partial f}{\partial x}x}{d} \approx \frac{\partial f}{\partial x}(x+1) - \frac{\partial f}{\partial x}x = f(x+2) - 2f(x+1) + f(x)$$

and for the y direction the result is very similar. The kernel for the second order gradient calculation with convolution:

$$\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} + [1 \quad -2 \quad 1] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- <u>other complex methods</u>

Edge Enhancement

Kernel for edge enhancement with Laplace operator: $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Canny Edge Detector

Properties of a „good" edge detector

- Good detection:
  - detects as many real edges as possible
  - does not create false edges (because of e.g. image noise)
- Good localization:
  - the detected edges should be as close to the real edges as possible
- Isotropic:
  - all edges are detected regardless of their direction

Main steps of the algorithm

1. Noise reduction
The original image is convolved with a Gaussian kernel to reduce image noise.

2. Gradient intensity and direction calculation
The horizontal and vertical derivative image is calculated (e.g. with Prewitt kernel). Based on them the gradient intensity and direction can be calculated:

$$d = \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}, \qquad \Theta = \arctan\left(\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial y}\right)$$

3. Non-Maximum Suppression
The goal is thinning the edges. Each edge is categorized into one of 4 main edge directions (0°, 45°, 90°, 135°), based on the gradient direction image (Θ). At every pixel, it suppresses the edge, by setting its value to 0, if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction:

4. Hysteresis thresholding
Problem with simple thresholding:

- if the threshold is low, many false edges will appear
- if the threshold is high, true edges will disappear

Solution: using two threshold instead of only one: t_1, t_2 where $t_1 > t_2$

- if the edge magnitude at (i,j) point is higher than t_1 then it is an edge
- if the edge magnitude at (i,j) point is lower than t_2 then it is not an edge
- if the edge magnitude at (i,j) point is lower than t_1 but higher than t_2, then it is an edge, only if one of its neighbors in the direction of $\Theta(i,j)$ is an edge

Hough Transformation

The objective of the Hough transformation is to find the lines on a binary image, from fragments/points of the line.

Basic idea

A line can be written in the following form: $y = mx + b$, where m is the slope and b is the y-intercept. The above equation can be re-written in terms of m and b:

$$m = -\frac{1}{x}b + \frac{y}{x}$$

For a fixed $y = y', x = x'$point in the image's (x,y) space, we get a line in the (m,b) space. For the points that lie on the same line in the Euclidean space, their corresponding line in the parameter space will cross each other in one point. This point will be $m = m'$ and $b = b'$.

Hough space

There is a problem with the Euclidean equation of the line: vertical lines cannot be described (their slope would be infinite). To be able to describe all possible lines we will use the polar equation of the line.

The (r, θ) parameter space is called the Hough space. A point in the Euclidean space is a sinusoid in the Hough space, described by the following equation:

$$r(\theta) = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

All the sinusoid curves of the points in one line in the Euclidian space cross each other in one point in the Hough space.

Topic 3 Fourier Transformation, Sampling, Nyquist Theorem

Fourier Transformation

The Fourier Transform changes between the representation in the time domain and in the frequency domain. The information is the same in both domains, only the representation is different. The Fourier transformation is a reversible transform. It builds on the fact that any function can be represented as a weighted sum of sinusoid functions. If we can describe sinusoids we can describe every function.

Complex Exponential Function

A sinusoid is defined by its frequency, amplitude and phase. In the frequency domain we need to "store" its frequency and phase. We use complex exponential functions to describe both.

The complex exponential function is the following:

$$e^{j\omega n} = \cos(\omega n) + j \cdot \sin(\omega n)$$

As n changes, the $e^{j\omega n}$ point rotates around the complex unit circle. The "speed" of this rotation is determined by ω.

Periodicity

In a continuous case (both ω and n are continuous):

- with the respect to the frequency (ω) it is not periodic
- with the respect to the time/spatial variable (n) it is periodic

In case of discrete time/spatial variable (ω is continuous, n discrete):

- with the respect to the frequency (ω) it is periodic with period 2π
$$e^{j(\omega+2\pi)n} = e^{j\omega n} \cdot e^{j2\pi n} = e^{j\omega n}$$
- with the respect to the time/spatial variable (n) it is may or may not be periodic depending on the frequency ω
If periodic with period N:
$$e^{j\omega(n+N)} = e^{j\omega n} \Longrightarrow e^{j\omega N} = 1 \Longrightarrow \omega N = k \cdot 2\pi \Longrightarrow N = k \cdot \frac{2\pi}{\omega}$$
so $2\pi/\omega$ has to be a rational number.

Eigenfunction of LSI systems

Let T be an LSI system with impulse response function $h(n_1, n_2)$, and $x(n_1, n_2)$ be a 2D complex exponential function:

$$x(n_1, n_2) = e^{j(\omega_1 n_1 + \omega_2 n_2)}$$

Then

$$y(n_1, n_2) = T\{x(n_1, n_2)\} = x(n_1, n_2) * h(n_1, n_2) = e^{j(\omega_1 n_1 + \omega_2 n_2)} * h(n_1, n_2) =$$

$$= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} e^{j(\omega_1(n_1-k_1)+\omega_2(n_2-k_2))} \cdot h(k_1, k_2) =$$

$$= \underbrace{e^{j(\omega_1 n_1 + \omega_2 n_2)}}_{\substack{\text{input function} \\ \text{went through} \\ \text{unchanged}}} \cdot \underbrace{\sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} e^{-j(\omega_1 k_1 + \omega_2 k_2)} \cdot h(k_1, k_2)}_{\substack{\text{frequency response of the LSI system:} \\ H(\omega_1, \omega_2)}}$$

Fourier and inverse Fourier transform

$$X(\omega_1, \omega_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) \cdot e^{-j\omega_1 n_1} \cdot e^{-j\omega_2 n_2}$$

$$x(n_1, n_2) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(\omega_1, \omega_2) \cdot e^{j\omega_1 n_1} \cdot e^{j\omega_2 n_2} \, \mathrm{d}\omega_1 \, \mathrm{d}\omega_2$$

Properties

- *Periodicity*: $X(\omega_1, \omega_2) = X(\omega_1 + 2\pi, \omega_2 + 2\pi)$
- *Translation*: $x(n_1 - m_1, n_2 - m_2) \leftrightarrow X(\omega_1, \omega_2) \cdot e^{-j\omega_1 m_1} \cdot e^{-j\omega_2 m_2}$
- *Modulation*: $x(n_1, n_2) \cdot e^{j\theta_1 n_1} \cdot e^{j\theta_2 n_2} \leftrightarrow X(\omega_1 + \theta_1, \omega_2 + \theta_2)$
- *Hermitian property*: for real $x(n_1, n_2)$

  magnitude: $|X(\omega_1, \omega_2)| = |X(-\omega_1, -\omega_2)|$

  phase: $\arg\big(X(\omega_1, \omega_2)\big) = -\arg\big(X(-\omega_1, -\omega_2)\big)$
- *Parseval's theorem*:

$$\sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} |x(n_1, n_2)|^2 = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} |X(\omega_1, \omega_2)|^2 \, \mathrm{d}\omega_1 \, \mathrm{d}\omega_2$$

- *Convolution theorem*:

$$y(n_1, n_2) = x(n_1, n_2) * h(n_1, n_2)$$
$$Y(\omega_1, \omega_2) = X(\omega_1, \omega_2) \cdot H(\omega_1, \omega_2)$$

Discrete Fourier transform

We sample one period of the Fourier transform in evenly spaced frequencies:

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) \cdot e^{-j\frac{2\pi}{N_1} k_1 n_1} \cdot e^{-j\frac{2\pi}{N_2} k_2 n_2}$$

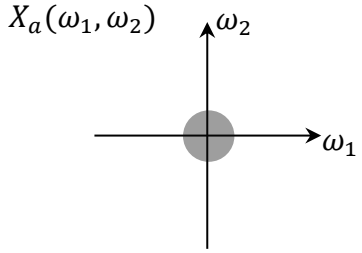
$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) \cdot e^{j\frac{2\pi}{N_1} k_1 n_1} \cdot e^{j\frac{2\pi}{N_2} k_2 n_2}$$

DFT is an exact transform, there is no transformation error. Most of the properties of continuous FT hold for DFT, except linear shift of FT become circular shift for DFT. DFT and inverse DFT are computable transformations. There are fast ways to compute the DFT: Fast Fourier Transform. The FFT with row/column decomposition requires only $N^2 \log_2(N)$ multiplications. So the FFT makes the Fourier transformation applicable in many practical cases.

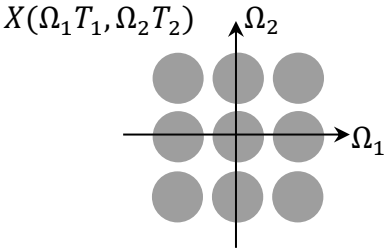
Sampling

Sampling is the conversion from analog to discrete signal.

Support of the spectrum of the analog image

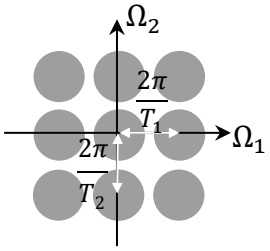


Support of the spectrum of the digital image



The spectrum of the digital image is periodic. The sampling periods T_1 and T_2 controls:

- in the frequency domain: how far away the replicas will be located
- in the spatial domain: how often we take samples from the analog image



$$X(\Omega_1 T_1, \Omega_2 T_2) = \frac{1}{T_1 T_2} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} X_a\left(\Omega_1 - k_1 \cdot \frac{2\pi}{T_1}, \Omega_2 - k_2 \cdot \frac{2\pi}{T_2}\right)$$

Critically Sampled

T_1 and T_2 are chosen so that the supports of the spectrums are closest to each other, but they are not overlapping. The analog signal can be reconstructed without error, using only the center part of the spectrum.

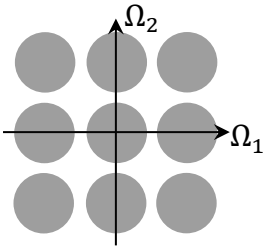
Oversampled

T_1 and T_2 are smaller than absolutely necessary, the supports of the replicas are farther apart, the spectrums are not overlapping. More samples are used than necessary.

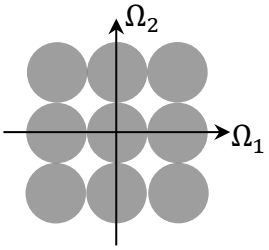
Under-sampled

T_1 and T_2 are too high; the supports of the spectrums are overlapping. The low and high frequencies are mixed; we cannot reproduce the original signal. Aliasing effect: the high frequencies are aliasing themselves as low frequencies.

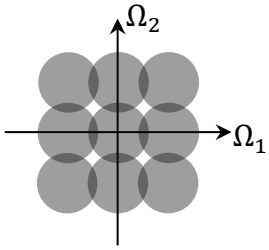
Oversampled



Critically sampled



Under-sampled



Nyquist Theorem

Let the highest frequency in the horizontal and vertical directions be Ω_{N_1} and Ω_{N_2}. As long as the following inequality holds, the spectrums won't overlap:

$$\frac{2\pi}{T_1} \geq 2 \cdot \Omega_{N_1}, \qquad \frac{2\pi}{T_2} \geq 2 \cdot \Omega_{N_2}$$

If we use sampling frequency at least two times as high as the highest frequency of the original analog signal, the analog signal can be reconstructed from the digital signal without error.

The minimum frequency that is required for the sampling to be able to reconstruct the analog signal from the sampled signal is the Nyquist frequency.

Topic 4 Image Enhancement

Image enhancement is the manipulation or transformation of the image to improve the visual appearance or to help further automatic processing steps. There is no general theory behind it, the result is highly application dependent and subjective. Image enhancement is closely related to image recovery.

The Histogram of the Image

Histogram is the function $h(k)$ which gives the number of pixels on the image with value k. The histogram normalized with the total number of pixels gives us the probability density function of the intensity values.

Point-wise Intensity Transformation

Point wise transformations are operating directly on pixel values, independently of the values of its neighboring pixels. We can describe the transformation as follows:
Let x and y be two grayscale images and let T be a pointwise image enhancement transformation that transforms x to y:

$$y(n_1, n_2) = T\{x(n_1, n_2)\}$$

Inverse Transformation

$$y(n_1, n_2) = 255 - x(n_1, n_2)$$

Log Transformation

$$y(n_1, n_2) = c \cdot \log(x(n_1, n_2) + 1)$$

Expands low and compress high pixel value range. It is commonly used to visualize the Fourier transform of an image.

Power-law transformation

$$y(n_1, n_2) = c \cdot \left(x(n_1, n_2)\right)^{\gamma}$$

Commonly referred as gamma transformation. Originally it was developed to compensate the input-output characteristics of CRT displays. The expanded/compressed region depends on γ.

Histogram Transformations

Histogram Stretching
Based on the histogram we can see that the image does not use the whole range of possible intensities. The following transformation stretches the intensity values so they use the whole available range:

$$y(n_1, n_2) = \frac{255}{x_{\max} - x_{\min}} \cdot (x(n_1, n_2) - x_{\min})$$

$$x_{\max} = \max_{n_1, n_2}\left(x(n_1, n_2)\right), \qquad x_{\min} = \min_{n_1, n_2}\left(x(n_1, n_2)\right)$$

Histogram Equalization
The goal is to increase the contrast, by distributing the occurrences of the intensity values evenly through the entire dynamic range.

Adaptive Histogram Equalization
This applies histogram equalization on parts of the image (called tiles) independently. We use post processing to reduce artifacts at the borders of the tiles.

Spatial Filtering

The goal of smoothing is to reduce the noise that may corrupt the image. The two types of noise (that we will work with):

- impulse noise (salt&pepper noise
- additive Gaussian noise

Gaussian Smoothing
Gaussian blur with a Gaussian kernel. It works not so well.

Spatially Adaptive Noise Smoothing
The smoothing takes into account the local characteristics of the image.

$$y(n_1, n_2) = \left(1 - \frac{\sigma_n^2}{\sigma_l^2}\right) \cdot x(n_1, n_2) + \frac{\sigma_n^2}{\sigma_l^2} \cdot \overline{x}(n_1, n_2)$$

where

$$\sigma_l^2 = \sum_{(n_1,n_2)\in N} \sum \left(x(n_1, n_2) - \overline{x}(n_1, n_2)\right)^2 \qquad \overline{x}(n_1, n_2) = \frac{1}{|N|} \sum_{(n_1,n_2)\in N} \sum x(n_1, n_2)$$

local variance of the image — local average of the image

Variance of the noise (σ_n^2) is either known a priori, or has to be measured.

Median Filter
This filter replaces each pixel with the median value of its analyzed neighborhood. (Median value: the center element of sorted values). This transformation is non-linear and very effective against impulse noise, however, not so effective against Gaussian noise.

Order Statistic Filtering
Based on the sorted pixel intensity levels in the analyzed neighborhood.

Mid-point filtering

$$y(n_1, n_2) = \frac{1}{2}\left(\max_{(m_1,m_2)\in N}\{x(m_1, m_2)\} + \min_{(m_1,m_2)\in N}\{x(m_1, m_2)\}\right)$$

This filter works well on Gaussian or uniform noise.

Alpha-trimmed mean filter

$$y(n_1, n_2) = \frac{1}{|N| - \alpha} \sum_{(m_1,m_2)\in N_r} x(m_1, m_2)$$

where N_r is a reduced neighborhood, not containing the lowest and highest α element of N.

- If $\alpha = 0$, we get back the arithmetic mean.
- If $\alpha = |N| - 1$, we get back the median filter.

Homomorphic Filtering

It simultaneously normalizes the brightness across an image and increases contrast. This method assumes the following image model: the image is formed by recording the light reflected from the objects illuminated by a light source.

$$y(n_1, n_2) = i(n_1, n_2) \cdot r(n_1, n_2)$$

where
$i(n_1, n_2)$: illumination; slowly varying, main contributor to dynamic range
$r(n_1, n_2)$: reflectance; rapidly varying, main contributor to local contrast

We want to reduce the illumination component, and increase the reflectance component.

The main steps of homomorphic filtering

1. To separate the two component we first use log transformation:
$$\log\big(x(n_1,n_2)\big) = \log\big(i(n_1,n_2)\big) + \log\big(r(n_1,n_2)\big)$$
2. Since we assume that the illumination component varies slowly and the reflectance varies rapidly, we can get the two component by using low and high pass filters:
$$\log\big(i(n_1,n_2)\big) = LPF\{\log\big(x(n_1,n_2)\big)\}$$
$$\log\big(r(n_1,n_2)\big) = HPF\{\log\big(x(n_1,n_2)\big)\}$$
3. Weight the two component:
$$\log\big(y(n_1,n_2)\big) = \gamma_1 \log\big(i(n_1,n_2)\big) + \gamma_2 \log\big(r(n_1,n_2)\big)$$
4. Transform back to the original range, using the exponential transform.

Wallis Operator

The Wallis operator can help to adjust local contrast. We can describe the image the following way:

$$x(n_1,n_2) = \underbrace{[x(n_1,n_2) - \overline{x}(n_1,n_2)]}_{(1)} + \underbrace{\overline{x}(n_1,n_2)}_{(2)}$$

where (2) is the local mean and (1) is the deviation from the local mean. With the transformation we want to push the local mean and standard deviation to a predefined desired value:

$$y(n_1,n_2) = [x(n_1,n_2) - \overline{x}(n_1,n_2)]\frac{\sigma_d}{\sigma_l(n_1,n_2)} + [p\overline{x}_d + (1-p)\overline{x}(n_1,n_2)]$$

We are almost there, but if the local contrast is too low, the weighting in (1) may get too high, this is why we maximize it with $A_{\max}$.

$$y(n_1,n_2) = [x(n_1,n_2) - \overline{x}(n_1,n_2)]\frac{A_{\max}\sigma_d}{A_{\max}\sigma_l(n_1,n_2) + \sigma_d} + [p\overline{x}_d + (1-p)\overline{x}(n_1,n_2)]$$

where

σ_l : local contrast:

$$\sigma_l(n_1,n_2) = \frac{1}{|N|}\sqrt{\sum_{(n_1,n_2)\in N}\sum \big(x(n_1,n_2) - \overline{x}(n_1,n_2)\big)^2}$$

$\overline{x}$: local average:

$$\overline{x}(n_1,n_2) = \frac{1}{|N|}\sum_{(n_1,n_2)\in N}\sum x(n_1,n_2)$$

σ_d: the desired local contrast
$\overline{x}_d$: the desired mean value of all pixels
p: weighting factor of the mean compensation
$A_{\max}$: minimizing the local contrast modification

Anisotropic Diffusion

The anisotropic diffusion is a technique aiming at reducing image noise without blurring significant parts of the image content. This is a non-linear and space-variant transformation. The main idea is that the effect of blurring in each direction is inversely proportional to the gradient value in that direction: the transformation allows diffusion along the edges or in edge-free territories, but penalizes diffusion orthogonal to the edge direction. AD is an iterative process.

Total Variation Regularization

Assumption

- The image is smooth inside the objects, with jumps across the boundaries.
- The noise component has high variation.

The goal of Total Variation based noise removal is to minimize the total variation of the image while keep the result as close to the original input image as possible. This transformation is defined as the integral of the absolute gradient of the signal:

$$V(x) = \sum_{n_1} \sum_{n_2} \sqrt{|x(n_1+1, n_2) - x(n_1, n_2)|^2 + |x(n_1, n_2+1) - x(n_1, n_2)|^2}$$

The goal function for total variation based regularization:

$$\hat{y} = \arg\min_{y} \big(E(x, y) + \lambda V(y) \big)$$

where E is the L_2 norm and λ is the regularization parameter.

Non-Local Means Denoising

The local smoothing methods aim at a noise reduction and at a reconstruction of the main geometrical configurations but not at the preservation of the fine structure, details and texture.

High frequency image components are removed along with the noise, because they behave in all functional aspects as noise.

The non-local means algorithm tries to take advantage of the high degree of redundancy of any natural image:

- Every small window in a natural image has many similar windows in the same image.
- The non-local means algorithm estimates the value of a pixel x as an average of the values of all the pixels whose Gaussian neighborhood looks like the neighborhood of x.

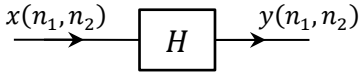
Topic 5 Image Recovery

Sources of Degradation and Forms of Recovery

Sources of Degradation	Forms of Restoration
1. Motion	
2. Atmospheric turbulence	1. Restoration/Deconvolution
3. Out-of-focus lens	2. Removal of Compression Artifacts
4. Finite resolution of the sensors	
5. Limitations of the acquisition system	3. Super-Resolution
6. Transmission error	4. Inpainting/Concealment
7. Quantization error	
8. Noise	5. Noise smoothing

Inverse problem formulation of Recovery

The original image x goes through a system H, that introduces some type of degradation resulting the observed image y.



The Goal of Recovery

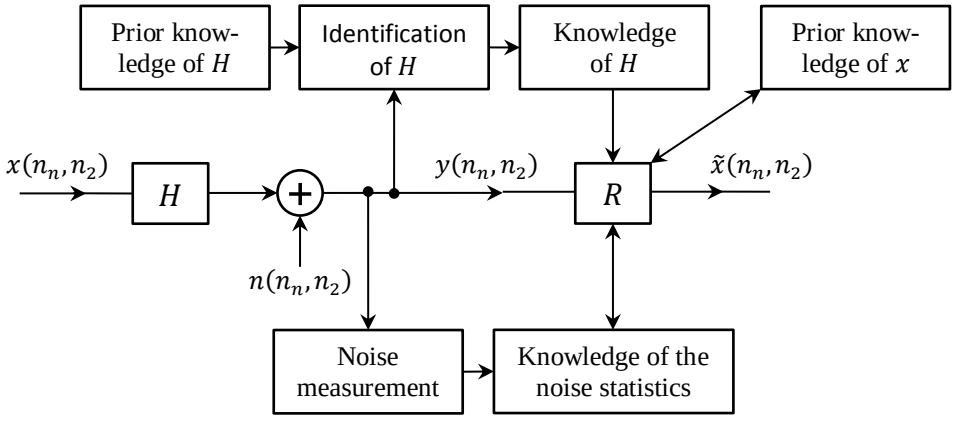
The objective is to reconstruct x based on…

- y and H → recovery
- y → blind recovery
- y and partially H → semi-blind recovery

These are the inverse problems. If we know x and…

- y → system identification
- H → system implementation

Degradation and Restoration



Degradation Model

The model of degradation for restoration problems:

$$y(n_1, n_2) = H\{x(n_1, n_2)\} + n(n_1, n_2)$$

If an LSI degradation system is assumed with signal independent additive noise:

$$y(n_1, n_2) = x(n_1, n_2) * h(n_1, n_2) + n(n_1, n_2)$$

The restoration problem in this case is called deconvolution.

Degradation/Restoration Metrics

Signal to Noise Ratio

$$SNR = 10 \lg\left(\frac{\sigma_x^2}{\sigma_n^2}\right)$$

Blurred Signal to Noise Ratio

$$BSNR = 10 \lg\left(\frac{\frac{1}{MN}\sum_i \sum_j [g(i,j) - \overline{g}(i,j)]^2}{\sigma_n^2}\right)$$

where
$g(i,j) = x(i,j) * h(i,j)$
$\overline{g}(i,j) = \mathbb{E}\{g(i,j)\}$
σ_n^2: variance of the noise

Improvement in Signal to Noise Ratio

$$ISNR = 10 \lg\left(\frac{\sum_i \sum_j [x(i,j) - y(i,j)]^2}{\sum_i \sum_j [x(i,j) - \tilde{x}(i,j)]^2}\right)$$

ISNR is computable only in a simulation environment, where the original image is available.

Convolution in matrix-vector form

1D convolution can be represented in a matrix-vector form:

$$y(n) = x(n) * h(n) = \sum_k x(k)h(n-k) \qquad \text{where } \begin{matrix} x: 1 \times N \\ h: 1 \times L \\ y: 1 \times (N+L-1) \end{matrix}$$

$$\begin{bmatrix} y(0) \\ y(1) \\ \\ \\ \\ \vdots \\ \\ \\ \\ y(N+L-2) \end{bmatrix} = \begin{bmatrix} h(0) & 0 & 0 & \cdots & 0 \\ h(1) & h(0) & 0 & \cdots & 0 \\ h(2) & h(1) & h(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ h(L-1) & h(L-2) & h(L-3) & \cdots & h(0) \\ 0 & h(L-1) & h(L-2) & \cdots & h(1) \\ 0 & 0 & h(L-1) & \cdots & h(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & h(L-1) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \\ \\ \\ \vdots \\ \\ \\ \\ x(N-1) \end{bmatrix}$$

$$\mathbf{y}^{[1\times(N+L-1)]} = \mathbf{H}^{[N\times(N+L-1)]}\mathbf{x}^{[1\times N]}$$

Circular convolution represented in a matrix-vector form:

$$\begin{bmatrix} y(0) \\ y(1) \\ \\ \vdots \\ \\ y(N+L-2) \end{bmatrix} = \begin{bmatrix} h(0) & 0 & \cdots & h(L-1) & \cdots & h(1) \\ h(1) & h(0) & 0 & \cdots & \cdots & h(2) \\ h(2) & h(1) & h(0) & 0 & \cdots & h(3) \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \cdots & h(L-1) & \cdots & h(1) & h(0) & 0 \\ 0 & \cdots & h(L-1) & \cdots & h(1) & h(0) \end{bmatrix} \begin{bmatrix} x(0) \\ \vdots \\ x(N-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\mathbf{y}^{[1\times(N+L-1)]} = \mathbf{H}^{[(N+L-1)\times(N+L-1)]}\mathbf{x}^{[1\times(N+L-1)]}$$

Eigenvalues and eigenvectors of circulant matrices

Let $\mathbf{H}$ be an $(M \times M)$ size circulant matrix, with eigenvalues λ_n and eigenvectors $\mathbf{w}_n$, where $n = 1, \dots, M$

$$\mathbf{H} = \begin{bmatrix} h(0) & \cdots & h(M-1) \\ & \ddots & \\ h(M-1) & \cdots & h(0) \end{bmatrix}, \qquad \mathbf{H}\mathbf{w}_n = \lambda_n \mathbf{w}_n$$

Then

$$\mathbf{w}_n = \begin{bmatrix} 1 & e^{j\frac{2\pi}{M}n} & e^{j\frac{2\pi}{M}2n} & \dots & e^{j\frac{2\pi}{M}(M-1)n} \end{bmatrix}^T$$

so the eigenvalues of this circulant matrix equals M times the DFT of the $\mathbf{h}$ vector:

$$\{\lambda_0, \dots, \lambda_{M-1}\} = M \cdot DFT\{h(0), \dots, h(M-1)\}$$

The Singular Value Decomposition of this $\mathbf{H}$ is the following:

$$\mathbf{H} = [\mathbf{w}_0 | \cdots | \mathbf{w}_{M-1}] \begin{bmatrix} \lambda_0 & & \\ & \ddots & \\ & & \lambda_{M-1} \end{bmatrix} [\mathbf{w}_0 | \cdots | \mathbf{w}_{M-1}]^{-1} = \mathbf{W}\mathbf{D}\mathbf{W}^{-1}$$

Back to images

If we stack the observed image lexicographically into a vector, the degradation can be described the following way:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}$$

If the system is LSI, then $\mathbf{H}$ is a block circulant matrix, matrix, which can be decomposed as a circulant matrix:

$$\mathbf{H} = \mathbf{W}\mathbf{D}\mathbf{W}^{-1}$$

So the degradation of the image:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} = \mathbf{W}\mathbf{D}\mathbf{W}^{-1}\mathbf{x} + \mathbf{n}$$

$$\mathbf{W}^{-1}\mathbf{y} = \mathbf{D}\mathbf{W}^{-1}\mathbf{x} + \mathbf{W}^{-1}\mathbf{n}$$

$$\mathbf{Y} = \mathbf{D}\mathbf{X} + \mathbf{N}$$

Since $\mathbf{D}$ is diagonal, we have the following element-wise equation:

$$Y(\omega_1, \omega_2) = H(\omega_1, \omega_2)X(\omega_1, \omega_2) + N(\omega_1, \omega_2), \qquad \text{where } \begin{matrix} \omega_1 = 0, \dots, M-1 \\ \omega_2 = 0, \dots, M-1 \end{matrix}$$

Restoration Algorithms

Inverse Filter

It is the simplest deconvolution filter, developed for LSI systems. This filter can be easily implemented in the frequency domain as the inverse of the degradation filter. Main limitations and drawbacks:

- Strong noise amplification
- The degradation system has to be known a priori

The objective is to find $\mathbf{x}$ that minimizes the following goal function:

$$\arg\min_{\mathbf{x}}\big(J(\mathbf{x})\big) = \arg\min_{\mathbf{x}}(\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2)$$

This goal function leads to the following equation:

$$-2\mathbf{H}^T\mathbf{y} + 2\mathbf{H}^T\mathbf{H}\mathbf{x} = \mathbf{0} \rightarrow \boldsymbol{x} = (\mathbf{H}^T\mathbf{H})^{\dagger}\mathbf{H}^T\mathbf{y}$$

If the degradation system is LSI, then $\mathbf{H}$ is a block circulant matrix. We can take the calculation to the frequency domain:

$$\boldsymbol{x} = (\mathbf{H}^T\mathbf{H})^{\dagger}\mathbf{H}^T\mathbf{y} \quad \overset{DFT}{\longrightarrow} \quad X(\omega_1,\omega_2) = \frac{H^*(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2}\, Y(\omega_1,\omega_2)$$

from which we can get $X(\omega_1,\omega_2)$ as

$$X(\omega_1,\omega_2) = \begin{cases} \dfrac{H^*(\omega_1,\omega_2)Y(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2}, & |H(\omega_1,\omega_2)| \neq 0 \\ 0 \quad , & |H(\omega_1,\omega_2)| = 0 \end{cases}$$

The drawback of this method is the strong amplification of noise:

$$\frac{H^*(\omega_1,\omega_2)Y(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2} = \frac{Y(\omega_1,\omega_2)}{H(\omega_1,\omega_2)} = \frac{H(\omega_1,\omega_2)X(\omega_1,\omega_2) + N(\omega_1,\omega_2)}{H(\omega_1,\omega_2)} = X(\omega_1,\omega_2) + \underbrace{\frac{N(\omega_1,\omega_2)}{H(\omega_1,\omega_2)}}_{\text{amplified noise}}$$

To reduce this effect a threshold can be used on H:

$$X(\omega_1,\omega_2) = \begin{cases} \dfrac{H^*(\omega_1,\omega_2)Y(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2}, & |H(\omega_1,\omega_2)| \geq T \\ 0 \quad , & |H(\omega_1,\omega_2)| < T \end{cases}$$

Constrained Least Square Methods

The objective is to reduce the noise amplification effect of the inverse filter by adding extra constraints about the restored image. In this case we have two terms, one describing the solutions fidelity, and the other gives some prior knowledge about the smoothness of the original image:

$$\arg\min_{\mathbf{x}}\big(J(\mathbf{x})\big) = \arg\min_{\mathbf{x}}(\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2), \qquad \|\mathbf{C}\mathbf{x}\|_2^2 < \varepsilon$$

Putting together the two terms with the introduction of α:

$$\arg\min_{\mathbf{x}}(\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2 + \alpha\|\mathbf{C}\mathbf{x}\|_2^2)$$

This goal function leads to the following equation:

$$\mathbf{x} = (\mathbf{H}^T\mathbf{H} + \alpha\mathbf{C}^T\mathbf{C})^{\dagger}\,\mathbf{H}^T\mathbf{y}$$

The $\mathbf{C}$ is a high pass filter and α is the regularization parameter. In the frequency domain (for $\mathbf{H}$ and $\mathbf{C}$ block circulant) we have the following formula:

$$X(\omega_1,\omega_2) = \frac{H^*(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2 + \alpha|C(\omega_1,\omega_2)|^2}\, Y(\omega_1,\omega_2)$$

Different types of regularization

If $\alpha = 0$, we get back the simple Least Square method (the Inverse Filter).

CLS

$$\tilde{x}(\alpha)_{CLS} = \arg\min_{\mathbf{x}}(\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2 + \alpha\|\mathbf{C}\mathbf{x}\|_2^2)$$

Maximum Entropy Regularization

$$\tilde{x}(\alpha)_{ME} = \arg\min_{\mathbf{x}}\left(\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2 + \alpha\sum_{i=1}^{N} x_i\log(x_i)\right)$$

Total Variation Regularization

$$\tilde{x}(\alpha)_{ME} = \underset{x}{arg\,min}\left(\|y-Hx\|^2 + \alpha\sum_{i=1}^{N}|[\Delta x_i]|\right)$$

l_p norms

$$J(z) = \|z\|_p^p = \sum_{i=1}^{N}|z_i|^p, \qquad 1 \leq p \leq 2$$

Iterative Restoration Algorithms

Pros	Cons
– Do not need to have the inverse of the degradation system explicitly	– Convergence of the algorithm is not always guaranteed
– The process can be monitored as it progresses, the number of iteration can be used as some kind of regularization (noise amplification can be controlled)	– Possibly takes more time
– Can be applied to spatially varying degradations and blind degradations	

Successive Approximation Algorithm:
Find a root for $\phi(x)$ by taking a reasonable initial point and iteratively go toward the root:

$$x_0 = 0, \qquad x_{k+1} = x_k + \beta\phi(x_k)$$

Restoration

$$\phi(\mathbf{x}) = \mathbf{y} - \mathbf{H}\mathbf{x}, \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \beta(\mathbf{y} - \mathbf{H}\mathbf{x}_k) = \beta\mathbf{y} + (\mathbf{I} - \beta\mathbf{H})\mathbf{x}_k$$

If the degradation system is assumed to be LSI, the in the frequency domain we will have the following:

$$X_{k+1}(\omega_1, \omega_2) = \beta Y(\omega_1, \omega_2) + \big(1 - \beta H(\omega_1, \omega_2)\big)X_k(\omega_1, \omega_2)$$

Convergence

$$X_k(\omega_1, \omega_2) = R_k(\omega_1, \omega_2)Y(\omega_1, \omega_2), \qquad R_k(\omega_1, \omega_2) = \beta\sum_{l=0}^{k-1}\big(1 - \beta H(\omega_1, \omega_2)\big)^l$$

So X_k converges if $|1 - \beta H(\omega_1, \omega_2)| < 1$.

Methods

Iterative Least-Squares

$$\phi(x) = \frac{1}{2}\nabla_x\|y - Hx\|^2, \qquad \mathbf{x}_{k+1} = \beta\mathbf{H}^T\mathbf{y} + (\mathbf{I} - \beta\mathbf{H}^T\mathbf{H})\mathbf{x}_k$$

In the frequency domain (assuming **H** is block circulant):

$$X_{k+1}(\omega_1, \omega_2) = \beta H^*(\omega_1, \omega_2)Y(\omega_1, \omega_2) + (1 - \beta|H(\omega_1, \omega_2)|^2)X_k(\omega_1, \omega_2)$$

Iterative Constrained Least-Squares

$$\phi(x) = \frac{1}{2}\nabla_x(\|y - Hx\|^2 + \alpha\|Cx\|_2^2), \qquad \mathbf{x}_{k+1} = \beta\mathbf{H}^T\mathbf{y} + \big(\mathbf{I} - \beta(\mathbf{H}^T\mathbf{H} + \alpha\mathbf{C}^T\mathbf{C})\big)\mathbf{x}_k$$

In the frequency domain (assuming **H** is block circulant):

$$X_{k+1}(\omega_1, \omega_2) = \beta H^*(\omega_1, \omega_2)Y(\omega_1, \omega_2) + \big(1 - \beta(|H(\omega_1, \omega_2)|^2 + \alpha|C(\omega_1, \omega_2)|^2)\big)X_k(\omega_1, \omega_2)$$

Spatially Adaptive CLS Iteration

$$\phi(x) = \frac{1}{2}\nabla_x\big(\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_{w_1}^2 + \alpha\|\mathbf{C}\mathbf{x}\|_{w_2}^2\big), \quad \text{where } \|\mathbf{x}\|_w^2 = \mathbf{x}^T\mathbf{W}^T\mathbf{W}\mathbf{x} = \sum_{i=1}^{N} w_i^2 x_i^2$$

$$\mathbf{x}_{k+1} = \beta\mathbf{H}^T\mathbf{W}_1^T\mathbf{W}_1\mathbf{y} + \big(\mathbf{I} - \beta(\mathbf{H}^T\mathbf{W}_1^T\mathbf{W}_1\mathbf{H} + \alpha\mathbf{C}^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{C})\big)\mathbf{x}_k$$

With the weight we can take into account the local variation of the image. For a human observer the noise is most disturbing in flat regions, while it is more acceptable around the edges. We can achieve the goal with the following weights:

$$W_2 = \frac{1}{\sigma_x^2}, \qquad W_1 = 1 - W_2$$

where σ_x^2 is the local variance of the image. Since W is not block circulant, we cannot take this to the frequency domain.

Wiener Filter

Stochastic restoration approach: Treat the image as a sample from a 2D random field. The image is part of a class of samples (an ensemble), realizations of the same random field.

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) P_{xx}(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 P_{xx}(\omega_1, \omega_2) + P_{NN}(\omega_1, \omega_2)}$$

Autocorrelation:

$$R_{ff}(n_1, n_2, n_3, n_4) = \mathbb{E}\{f(n_1, n_2) f^*(n_3, n_4)\}$$

Power-spectrum:

$$P_{ff}(\omega_1, \omega_2) = \mathcal{F}\{R_{ff}(d_1, d_2)\}$$

The Wiener-filter is the following formula:

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) P_{xx}(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 P_{xx}(\omega_1, \omega_2) + P_{NN}(\omega_1, \omega_2)} = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 + \frac{P_{NN}(\omega_1, \omega_2)}{P_{xx}(\omega_1, \omega_2)}}$$

if we assume white noise, $P_{NN}(\omega_1, \omega_2) = \sigma_N^2$ so the filter's formula is the following:

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 + \frac{\sigma_N^2}{P_{xx}(\omega_1, \omega_2)}}$$

We can see, that with the right choice of $\mathbf{C}$ and α, CLS filter is the same as the Wiener filter.

Ringing Artifact

Ringing artifact comes from the fact that if we convolve the degradation and the restoration filters we get the following:

$$s_{all}(n_1, n_2) = h(n_1, n_2) * r(n_1, n_2), \qquad \tilde{x}(n_1, n_2) = s_{all} * x(n_1, n_2)$$

In an ideal case

$$s_{all}(n_1, n_2) = \delta(n_1, n_2), \qquad S_{all}(\omega_1, \omega_2) = 1 \quad \forall \omega_1, \omega_2$$

But in practice this is not true. We can use a positivity constraint to get rid of (some of) the ringing artifact effect. The positivity constraint means that s_{all} must be positive (or zero):

$$s_{all}(n_1, n_2) = \begin{cases} s_{all}, & s_{all} \geq 0 \\ 0, & s_{all} < 0 \end{cases}$$

Topic 6 Introduction to Machine Learning

What is Machine Learning?

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

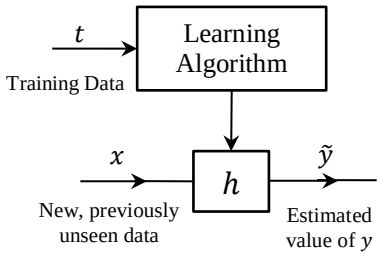
Supervised Learning

The supervised algorithms are trained on labeled data, where the desired output is known. The goal is to train a classifier that can work on previously unknown data. It has two branches:

- regression: prediction of continuous valued output
- classification: prediction of discrete valued output

Learning Algorithms in General

Summarization of a Learning Algorithm



Learning

In the case of supervised learning we have a training set and a hypothesis function (h). We want to find the best parameters for the h, where the error is minimal. For this reason we create a function called cost function and we search for

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Linear Regression

In this method we try to fit a line on the point of the training set. One point of the training set can be described with two parameters: $\left(x^{(i)}, y^{(i)}\right)$. The hypothesis function is therefore a line:

$$h_{\boldsymbol{\theta}} = \theta_0 + \theta_1 x$$

To find the best values for the parameter $\boldsymbol{\theta}$ we find parameters (θ_0, θ_1) so that $h_{\boldsymbol{\theta}}(x)$ is closely to y for the training examples:

$$\min_{\boldsymbol{\theta}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

where m is the number of training examples. With the conventional notation of cost function the above is

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Gradient Descent Method
Gradient Descent method will be used to find the minimum of the cost function. The steps are:

1. Start with arbitrary initial values (e.g. $\theta_0 = 0, \theta_1 = 0$)
2. In each iteration change θ_j so that J is reduced, until it reaches its minimum value. To achieve this the following update rule is used:
$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad j = 0,1$$
3. The update is done simultaneously for all the θ_j.

Linear Regression with Multiple Variables
Linear regression can be more powerful with multiple variables. The new hypothesis function:
$$h_{\boldsymbol{\theta}}(x_1, x_2, \dots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$
More convenient to write it in a matrix-vector form:
$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right)$$

Logistic Regression
Logistic Regression produces answers between $[0,1]$: $0 \leq h_{\boldsymbol{\theta}}(\mathbf{x}) \leq 1$. To achieve this we take the logistic function of $\boldsymbol{\theta}^T \mathbf{x}$:
$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

Interpretation of the hypothesis
If for some $\mathbf{x}$ the $h_{\boldsymbol{\theta}}(\mathbf{x}) = 0.8$, it means, that $\mathbf{x}$ has 80% probability to belong to the positive class. To predict binary class labels we use a threshold 0.5:
$$\mathbb{P}(y = 1 | \mathbf{x}; \boldsymbol{\theta}) \geq 0.5 \rightarrow y = 1$$
$$\mathbb{P}(y = 1 | \mathbf{x}; \boldsymbol{\theta}) < 0.5 \rightarrow y = 0$$

Cost function
In linear regression the cost function was the following:
$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \text{cost}\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right), y^{(i)} \right)$$
The problem is that in the case of logistic regression the hypothesis function is non-linear and if we put it into the $J(\boldsymbol{\theta})$ the result will be a non-convex cost function. We need to replace the $\text{cost}\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right) - \mathbf{y}^{(i)}\right)$ function. We will use the following:
$$\text{cost}\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right), y^{(i)} \right) = \begin{cases} -\log\left(h_{\boldsymbol{\theta}}(\mathbf{x})\right), & y = 1 \\ -\log\left(1 - h_{\boldsymbol{\theta}}(\mathbf{x})\right), & y = 0 \end{cases}$$

if $y = 1$
- the cost is equal to zero if $h_{\boldsymbol{\theta}}(\mathbf{x}) = 1$
- as $h_{\boldsymbol{\theta}}(\mathbf{x})$ goes to 0, the cost goes to ∞

if $y = 0$
- the cost is equal to zero if $\mathrm{h}_{\theta}(\mathrm{x}) = 0$
- as $\mathrm{h}_{\theta}(\mathrm{x})$ goes to 1, the cost goes to ∞

The unified cost function of logistic regression is as follows:
$$J(\boldsymbol{\theta}) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log\left(h_{\boldsymbol{\theta}}\left(x^{(i)}\right)\right] + \left[\left(1 - y^{(i)}\right) \log\left(h_{\boldsymbol{\theta}}\left(x^{(i)}\right) \right) \right]$$

Regularization

If we have too many features we can learn a hypothesis that fits the training data very well, but fails on new samples (does not generalize well).

To handle underfitting we can introduce new features.

To handle overfitting:

- We can reduce the number of features (but this might mean we lose information):
  - We can select manually which features to keep.
  - Use a model selection algorithm.
- We can apply regularization:
  - We can keep all the features but we reduce their magnitude (the value of the θ parameters).
  - Works well if we have a lot of features and each contributes a little bit to predict y.
  - The idea is to keep the parameters low, to get a simpler hypothesis function, which is less prone to overfitting.

Regularization term

The cost function for linear/logistic regression with regularization:

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) + \lambda \sum_{j=1}^{n} \theta_j^2$$

The regularization parameter λ controls the trade-off between two goals:

- Fitting the data well
- Keeping the parameters low, to avoid overfitting

If λ is too large all the parameters (exceptθ_0) will be close to 0, the model won't fit the data, we will see underfitting.

Support Vector Machines

There could be many decision boundaries that separate two classes. Which one is the best? The SVM aims to keep as large margin between the decision boundary and the closest sample as possible.

Case of the logistic regression

In case of logistic regression this would mean:

- if $y = 1$ we want $h_{\boldsymbol{\theta}}(\mathbf{x}) \approx 1,\ (\boldsymbol{\theta}^T\mathbf{x} \gg 0)$
- if $y = 0$ we want $h_{\boldsymbol{\theta}}(\mathbf{x}) \approx 0,\ (\boldsymbol{\theta}^T\mathbf{x} \ll 0)$

To achieve this goal we need a different cost function:

$$J(\boldsymbol{\theta}) = C \cdot \sum_{i=1}^{m}\left(y^{(i)}\,\text{cost}_1\left(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right)\text{cost}_0\left(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}\right)\right) + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

Hypothesis function of SVM

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{cases} 1, & \boldsymbol{\theta}^T\mathbf{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Using Kernels

One way to define a complex non-linear decision boundary is by the use of high order terms:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_3^2 + \cdots \rightarrow \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots$$

What can we use as f? For example we can use the distance from landmark points $l^{(1)}, l^{(2)}, \ldots$ In this case the f functions will be

$$f_1 = \exp\left(-\frac{\left\|x - l^{(1)}\right\|^2}{2\sigma^2}\right), \dots, f_n = \exp\left(-\frac{\left\|x - l^{(n)}\right\|^2}{2\sigma^2}\right)$$

where these f_i kernel functions measure the similarity between point x and the landmark $l^{(i)}$. If x is far from $l^{(i)}$ then $f_i \approx 0$, but if x is close to $l^{(i)}$, then $f_i \approx 1$. These kernels are called Gaussian kernels. In this case the hypothesis function is the following:

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{cases} 1, & \boldsymbol{\theta}^T\mathbf{f} = \theta_0 + \theta_1 f_1 + \cdots + \theta_n f_n \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

How do we get the landmarks?
We place a landmark at the position of each training example. The decision is made based on how close/similar the test samples are to the positive and negative training samples. The feature vector **x** which represented a sample is now replaced a new feature vector **f**, which contains the similarities to the training samples

Parameters of the SVM
The C parameter controls the trade-off between two goals:

- Fitting the data well (high value for C)
- Keeping the parameters low, to avoid overfitting (low value for C)

The used similarity kernels may have further parameters. For the Gaussian kernel the bandwidth parameter σ contros:

- High σ, results a slowly changing Gaussian, which can cause high bias.
- Low σ, results a more rapidly changing Gaussian, which can cause high variance.

Unsupervised Learning

In case of unsupervised learning the training data is not labeled. The goal is to find meaningful structure in the data.

K-Means Clustering

It aims to partition the data samples into k clusters. Each sample will belong to the cluster with the nearest mean. The objective is to minimize the within-cluster sum of squares:

$$\arg\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

where $x_1, x_2, \dots, x_n$ are the data samples, μ_i is the mean of the points in the cluster S_i, and $i = 1, \dots, k$ where k is the total number of clusters.

Iterative heuristic method for k-means clustering

1. Initialize the k cluster means
2. Assignment step: assign each sample to the nearest mean
3. Update step: calculate the new mean for each cluster:

$$\mu_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

Limitation of k-means

- Number of clusters has to be known a priori
- Spherical cluster shapes
- Could stuck in a local minimum

Mean-Shift Clustering

Non-parametric iterative clustering technique introduced in 1975 by Fukunaga and Hostetler. We do not need to know the number of clusters a priori. This method does not constrain the shape of the cluster. Mean shift considers the points in the feature space as samples from an underlying probability density function. The objective of the algorithm is to find the modes of this probability density function, and associate each point with the node it is "attracted to".

Main steps

1. A density estimation window (e.g. a Gaussian window) is placed on each sample point.
2. Within each window the mean shift vector is calculated, which points toward the maximum density:

$$m_h(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x$$

   where x is a d-dimensional feature point, $g(x) = -K'(x)$, where K is a kernel function (e.g. Gaussian kernel) and h is the bandwidth parameter of the kernel.
3. The window is shifted with the mean shift vector.
4. Step 2 and 3 are repeated until convergence to a local density maximum.
5. The sample points that converged to the same local maximum will belong to the same cluster.

Other Clustering Methods

DBSCAN (Density-based Spatial Clustering for Applications with Noise)

- Don't need to know the number of clusters
- Can find arbitrary shaped clusters
- Robust to outliers: has a built in noise handling technique
- Quality depends on the distance measure (usually Euclidean distance, which doesn't respond well to high dimensionality)

Topic 7 Local Feature Descriptors

Types of Descriptors

The detection and description of local features has an important role in many applications. There are different types of use of the descriptors. When we are talking about local feature descriptors we usually talking about one or both of the following two tasks:

- Keypoint or feature detection
- Feature extraction: generation of a descriptor for the feature point's local neighborhood.

There are methods that do both or only one of the tasks:

Feature point detectors	**Feature descriptors**
– Hessian/Harris corner detector	– SIFT
– Laplacian of Gaussian	– SURF
– Difference of Gaussian (in SIFT)	– HOG
– SURF (uses Hessian Blob detector with integral image)	– BRIEF
	– LBP

SIFT: Scale Invariant Image Transform

Advantages

- Invariant to translation, scaling, and rotation
- Robust to illumination changes, noise, minor changes in viewpoint
- Robust to local geometric distortion
- Highly distinctive
- SIFT based object detectors are robust to partial occlusion

Steps of the Algorithm

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint description

1. Scale-space extrema detection

Keypoint detection with Difference of Gaussians:

Let $I(x, y)$ is the original image and $G(x, y, k\sigma)$ a Gaussian blur at scale σ. The original image convolved with Gaussian kernel at different scales:

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$$

The convolved images are grouped by octave (in an octave σ is doubled). The difference of consecutive convolved images is taken in an octave:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L\big(x, y, k_j\sigma\big)$$

Then we choose all extrema within a $3 \times 3 \times 3$ scale-space neighborhood. These extremas are the keypoints.

2. Keypoint localization

Localization is done with sub pixel accuracy, based on the interpolation of nearby data. Rejection of weak candidates: low contrasted points and poorly localized points along edges.

3. Orientation assignment
Goal is to ensure rotation invariance: find the main orientation(s) and assign it to the key point and give the description of the keypoint relative to this orientation. Steps:

1. Gaussian smoothed image is taken at the scale of the keypoint.
2. The edge magnitude and orientation is calculated for each point in the neighborhood.
3. A 36 bin orientation histogram is composed, where each bin represents a 10 degree interval, and each neighboring point's bin is determined based on its edge orientation and its weight based on the edge magnitude.
4. Also the points are weighted with a Gaussian window, so the points farther away have less effect than the points closer to the keypoint.
5. The orientation of the keypoint will correspond to the peak of the histogram.

4. Keypoint description
For every keypoint (x, y, σ, θ):

1. Take a 16×16 point neighborhood around the keypoint and divide it into 4×4 gradient window.
2. Build the orientation histogram of the 4×4 samples in each window with 8 direction bins.
3. Gaussian weighting around center (size is based on σ)
4. $4 \times 4 \times 8 = 128$ dimensional feature vector

HOG: Histogram of Oriented Gradients

Originally developed for pedestrian detection by N. Dalal, B. Triggs in 2005

Steps of the Algorithm

1. Gradient Computation
2. Orientation Binning
3. Block Description
4. Block Normalization
5. Classification

1. Gradient Computation
Gradient calculation with the simple $[-1\ 0\ 1]$ and $[-1\ 0\ 1]^T$ gradient detectors.

2. Orientation Binning for a cell
A cell is a rectangular (or circular) shaped 8×8 window. The histogram of gradient orientations is calculated over the cell, each pixel votes based on its magnitude on the gradient image. A 9 bin histogram is made (form 0°to 180°).

3. Block Description
A block contains 2×2 cells. Pixels in the block are weighted by a Gaussian window.

4. Block Normalization
The blocks are overlapping; every cell is used 4 times in 4 different blocks. Also there are different versions of the normalization:

L_1 norm
$$v \to \frac{v}{\|v\|_1 + \varepsilon}$$

L_1 sqrt
$$v \to \sqrt{\frac{v}{\|v\|_1 + \varepsilon}}$$

L_2 norm
$$v \to \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon^2}}$$

L_2 Hys
max value of v is limited to 0.2

Haar-Like Features

They are named after the Haar wavelets.

Cascade classifier

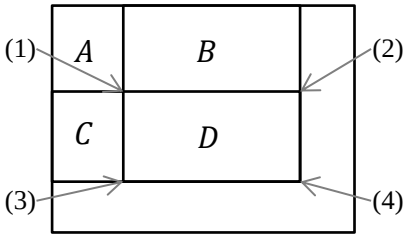
The goal is to be able to reject many obvious non-face samples quickly and concentrate the computational power on the more difficult samples. By the concatenation of a lot of weak classifiers a highly effective classifier is built. Each weak classifier can reject a sample, so the following weak classifiers don't have to evaluate it. Each weak classifier is tuned to compensate the previous classifiers' errors

Integral Image trick

A way to calculate Haar-like features very quickly, in constant time, regardless of the size of the feature.

The integral image is defined as follows:

$$I_{int}(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} I(x',y')$$



Using the integral image the sum of any rectangular shaped area can be calculated with 4 operations:

$$D = \underbrace{(A+B+C+D)}_{(4)} + \underbrace{(A)}_{(1)} - \underbrace{(A+B)}_{(2)} - \underbrace{(A+C)}_{(3)} = (4)+(1)-(2)-(3)$$

LBP: Local Binary Patterns

LBP is a computationally effective texture descriptor. It is comparable to the state of the art, while computable in $O(n)$ time. LBP is robust to monotonic changes in the illumination, no image preprocessing or parameter tuning is required. Produces a compact, 59 bin descriptor (SIFT has 128 bins), so it is faster to match. Its efficiency was proved in many applications such as face detection, face recognition, image retrieval, texture analysis etc.

Steps of the Algorithm

1. LBP Calculation
2. Histogram Building
3. Histogram Matching

1. LBP Calculation

Calculate the difference of a pixel and its 8 neighbors in a fixed radius circular pattern then binarize the result. Represent the result as a decimal number, this is the LBP value.

Uniform LBP

So far we have 256 dimension descriptor. In general 90% of the LBPs has one or two continuous regions in it:

- 2 patterns with one region (full 1, or full 0)
- 7 patterns with 2 regions

For each of the 7 pattern with 2 regions there can be 8 different orientations. Plus we keep one joker bin for everything else. Therefore we get a $2 + 7 \times 8 + 1 = 59$ bin descriptor. This reduced dimension descriptor is also more robust to noise.

2. Histogram Building

Build the histogram from LBP values.

3. Histogram Matching

To match histograms the following measures is commonly used: Histogram Intersection, Chi-Squared, Log-Likelihood.

Binary Descriptors

The SIFT, SURF and HOG methods are based on histograms of gradients, which is costly to compute and the size of the descriptors can be problematic if we have many of them. Also SIFT is patent protected.

Binary descriptors use simple intensity value comparisons to create binary strings to encode the information of the patch. It is fast to compute, easy to store and fast to match (the Hamming distance is equivalent with XOR)

In general, Binary descriptors are composed of three parts

- sampling pattern
- sampling pairs
- orientation compensation

Sampling pattern

The use of binarized intensity value differences: take a sample at point A and compare its value to a sample in an other point, B. If A's intensity is higher add a 1 to the descriptor string, otherwise add 0. The sampling pattern defines the way we take samples: BRISK, FREAK, BRIEF, ORB…

Sampling pairs

BRIEF uses random sampling pattern and selects random pairs from them.
BRISK uses only short distance pairs from the predefined pattern.
FREAK and ORB learns the sampling pairs so that

- their information content is maximal, the redundancy is minimal between the pairs,
- the variance of the pairs is high to make the feature more discriminative.

In case of FREAK the resulted pairs follow a coarse-to-fine structure:

- the first pairs selected are comparing points in the outer ring
- the last selected points make comparisons in the dense region
- this resembles to the way the human vision operates.

Orientation compensation

The binary pairs are sensitive to rotation. In the orientation compensation phase the orientation angle of the patch is measured and the pairs are rotated by that angle to ensure that the description is rotation invariant. Different descriptors have different methods for orientation compensation:

- BRIEF: does not have orientation compensation
- ORB: based on the moments of the patch
- BRISK: comparing gradient of long pairs
- FREAK: comparing gradient of preselected pairs

Feature Matching

1. Given two keypoints, first procedure the binary descriptor for both of them, using the same sampling pattern and the same sequence of pairs.
2. Once we have two binary strings, just count the number of bits where the strings are different.

Topic 8 Video Processing, Motion Estimation, Object Tracking

Video Processing

The main difference between a still image and a dynamic video is the motion. Estimating the motion on the video is a fundamental step for many algorithms: object tracking, video surveillance, video compression etc.

3D vs. 2D Motion

In general, we are interested in the motion in the 3D scene, but we can only work with its 2D projection on the image plane. The interpretation of the 2D image can be ambiguous: "does the size of the object really changed or is it just further away from the camera"?

True and Apparent Motion

We want to know how an object, an image region or a single pixel moved on the image plane from one frame to the next. But the change of the pixel does not necessarily means that motion occurred in the real 3D world and a real world change might not result a change in the pixel value.

Motion Estimation

Direct Methods: they compute the optical flow between two consecutive frames.

- Optical Flow
- Phase Correlation
- Block Matching
- Spatio-Temporal Gradient

Indirect methods:

- Feature Matching, which locates feature points on both images, finds the pair of each feature point on the other image and calculates the motion based on the displacement of the feature points. For this different features can be used: SIFT, SURF, Harris…

Phase Correlation

This method can be used to estimate global motion, for image registration. It is based on the translation property of the Fourier transform:

$$x(n_1 - m_1, n_2 - m_2) \leftrightarrow X(\omega_1, \omega_2)e^{-j\omega_1 m_1}e^{-j\omega_2 m_2}$$

The assumption is that between two consecutive frames the image was shifted by (m_1, m_2):

$$x_t(n_1, n_2) \qquad x_{t+1}(n_1, n_2) = x_t(n_1 - m_1, n_2 - m_2)$$

$$X_t(n_1, n_2) \qquad X_{t+1}(k_1, k_2) = X_t(k_1, k_2)e^{-j\frac{2\pi}{N_1}m_1 k_1}e^{-j\frac{2\pi}{N_2}m_2 k_2}$$

Calculation of the crow power spectrum

$$C(k_1, k_2) = \frac{X_t(k_1, k_2)X_{t+1}^*(k_1, k_2)}{|X_t(k_1, k_2)X_{t+1}^*(k_1, k_2)|} = \frac{|X_t(k_1, k_2)|^2 \cdot e^{-j\frac{2\pi}{N_1}m_1 k_1}e^{-j\frac{2\pi}{N_2}m_2 k_2}}{|X_t(k_1, k_2)|^2}$$

Transforming the C back to the spatial domain we get the normalized cross correlation:

$$c(n_1, n_2) = \delta(n_1 - m_1, n_2 - m_2)$$

Dealing with the border regions

Since we use DFT the linear shifts becomes circular shifts. In most cases the images are related by linear shift not circular, so the border regions may corrupt our calculation. Solution: use a 2D Hamming window (or something similar) to down weight the values close to the border.

Block Matching

Block matching is an intuitive motion estimation algorithm, widely used in video compression. It is based on the following assumptions:

- The objects are rigid
- The illumination of the scene is constant
- Objects are not entering or leaving the scene
- The motion is parallel to the image plane

To estimate the local motion between two frames (A and B):

1. Frame A is divided into (usually squared shaped) blocks.
2. For each block of A, a search is made on frame B, to find its best matching pair.

Different measures can be used as matching criteria, to find the best fit for a block:

- Mean Square Error
- Mean Absolute Error
- Correlation Function

To get a dens field of motion vectors overlapping blocks can be used. To find the best match for a block with exhaustive search is computationally highly demanding. There are different ways to reduce the computational burden:

- Reduced search window or block size
- Block sub-sampling
- Logarithmic search
- Pixel projection

Hierarchical Block Matching

With the hierarchical structure we reduce the size of the images. Do the motion estimation first at the lowest resolution, with relatively large search window. Use the (properly up-scaled) estimated vectors for initialization of the search at a higher level. The final motion vectors are calculated at the original dimension.

Optic Flow

Constant brightness constraint: It is assumed that the brightness of an object remains the same from one frame to an other, hence all the changes in brightness are solely due to the motion in the scene.

$$I(x, y, 0) = I(x + u, y + v, \tau)$$

where u and v are the displacement of the pixel in each dimension and τ is the time ($\tau = 0$ is the reference frame).

Using Taylor series expansion:

The Taylor series expansion of a function f, that is infinitely differentiable at a, is given with the following formula:

$$\sum_{n=0}^{\infty} \frac{\mathrm{d}^n}{\mathrm{d}x^n} f(a) \frac{(x-a)^n}{n!}$$

Then we take the Taylor series expansion of $I(x + u, y + v, \tau)$ at 0:

$$I(x + u, y + v, \tau) = I(x, y, 0) + \underbrace{\frac{\partial I(x, y, 0)}{\partial x}}_{I_x} u + \underbrace{\frac{\partial I(x, y, 0)}{\partial y}}_{I_y} v + \underbrace{\frac{\partial I(x, y, 0)}{\partial t}}_{I_t} \tau + HOT$$

So

$$0 = I_x u + I_y v + I_t \tau$$

If we divide everything with τ, we get velocities. Hence the optic flow equation is as follows:

$$0 = I_x V_x + I_y V_y + I_t$$

This equation is under determined: 2 unknowns, 1 equation. Assuming that neighboring pixels undergo the same motion we can increase the number of equations:

$$\begin{cases} I_x(n_1)V_x + I_y(n_1)V_y = I_t(n_1) \\ I_x(n_2)V_x + I_y(n_2)V_y = I_t(n_2) \\ \vdots \\ I_x(n_N)V_x + I_y(n_N)V_y = I_t(n_N) \end{cases} \rightarrow \begin{bmatrix} I_x(n_1) & I_y(n_1) \\ I_x(n_2) & I_y(n_2) \\ \vdots & \vdots \\ I_x(n_N) & I_y(n_N) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} I_t(n_1) \\ I_t(n_2) \\ \vdots \\ I_t(n_N) \end{bmatrix} \rightarrow \mathbf{Ax} = \mathbf{b}$$

This is an inverse problem: we know **A** and **b** and look for **x**. Least square or constrained least square solutions can be used:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^\dagger\mathbf{A}^T\mathbf{b}, \qquad \mathbf{x} = (\mathbf{A}^T\mathbf{A} + \boldsymbol{\lambda}\mathbf{C}^T\mathbf{C})^\dagger\mathbf{A}^T\mathbf{b}$$

Feature Matching

The algorithm is based on the matching of key points that has a well-defined position on the image (e.g. corner). A descriptor is given to each point that is preferably…

- Shift and rotation invariant
- Robust against the changes of illumination
- Scale invariant
- Low dimensional
- Robust to noise, etc.

From the matched point pairs the global motion of the camera can be estimated. As feature point detectors SIFT or SURF can be used, and as feature descriptors SIFT, SURF, HOG, LBP, etc.

Object Tracking

The objective of object tracking is to associate target objects in consecutive video frames. It can be applied in human-computer interaction, traffic monitoring, vehicle navigation, motion-based recognition, video indexing, automated surveillance, etc.

The tracking task can be divided into two subtasks:

- Build a model of the object you want to track
- Use what you know about where the object was in the previous frame(s) to make predictions about the current frame to restrict the search.

Repeat the two subtasks and possibly update the model.

Tracking objects can be complex due to:

- Loss of information caused by projection from 3D to 2D
- Noise
- Complex object shapes/motion
- Non-rigid or articulated nature of objects
- Partial and full occlusions of the object
- Changes of the illumination
- Real-time processing requirements

Simplify tracking by making assumptions and the use of prior information:

- The motion of the object is smooth with no abrupt changes
- The object motion is assumed to be of constant velocity
- Prior knowledge about the number and the size of objects, or the object appearance.

Modeling of the object:

Shape:

- Point model
- Simple geometrical forms
- Contour/Silhouette
- Articulated shape models
- Skeletal models

Appearance:

- Template (if the appearance is not changing)
- Probabilistic representation of the object appearance (e.g. Histogram)

Different features can be used to describe the appearance:

- Color, Edge, Motion, Texture
- HOG, SIFT, LBP, …

Topic 9 Image Segmentation

What is on the image? – This is maybe the most important question we want to answer about an image. For a human observer it is a trivial task, for a machine it is still an unsolved problem. An important step toward our goal is to segment the image into meaningful parts. The objective is to group pixels together based on some common characteristics:

- they belong to the same physical object
- they have the same intensity level/color/texture
- they belong to the background/foreground
- …

The segmentation can be **knowledge-driven** (top-down) or **data-driven** (bottom-up).
Knowledge driven segmentation methods builds prior knowledge into the segmentation algorithm:

- hard to implement
- cannot stand alone: need cues from bottom-up segmentation

Data-driven methods builds on the raw pixel data:

- they are easier to implement
- they often fail on real life images

There is the so-called semantic gap between the two approaches. The complex, high level definitions of top-down methods are hard to embed efficiently into low level algorithms.

Intensity Level Based Segmentation

Thresholding
Assumption: the image parts (e.g. object and background) can be separated based on their intensity level.

$$s(n_1, n_2) = \begin{cases} \text{object}, & x(n_1, n_2) < T \\ \text{background}, & x(n_1, n_2) \geq T \end{cases}$$

where $s(n_1, n_2)$ is the cluster of the (n_1, n_2) pixel of the x image and T is a threshold.

Otsu's Method
Automatically determines the optimal global threshold by minimizing the intra-class variance. The intra-class variance is defined as follows:

$$\sigma_w^2(k) = \omega_1(k)\sigma_1^2(k) + \omega_2(k)\sigma_2^2(k)$$

where ω_i and σ_i are the probability and the variance of the two classes separated by the threshold k. Otsu showed that minimizing the intra-class variance is the same as maximizing inter-class variance:

$$\sigma_b^2(k) = \sigma^2 - \sigma_w^2(k) = \omega_1(k)\omega_2(k)\big(\mu_1(k) - \mu_2(k)\big)^2$$

where μ_i are the means of the two classes separated by threshold k. To calculate ω_i and μ_i the normalized histogram of the image is used:

$$\omega_1(k) = \sum_{i=0}^{k} p_i \qquad \omega_2(k) = \sum_{i=k+1}^{L-1} p_i$$

$$\mu_1(k) = \sum_{i=0}^{k} \frac{ip_i}{\omega_1} \qquad \mu_2(k) = \sum_{i=k+1}^{L-1} \frac{ip_i}{\omega_2}$$

where p_i is the i-th entry in the normalized histogram of the image.

Region Based Segmentation Methods

Let R be the entire image region, and $R_1, \dots, R_n$ are subregions. We want to find a segmentation that is

- Complete

$$\bigcup_{i=1}^{n} R_i = R$$

- Points in the region R_i are connected $i = 1, \dots, n$
- The regions are disjoint

$$R_i \cap R_j = \emptyset \Leftrightarrow \forall i \neq j$$

- All the pixels in a region have common properties that they do not share with pixels from other regions.

Region Growing

The method is initialized with a set of seed points as regions. We start growing the regions by adding neighboring pixels to the region if they have similar predefined properties as the seed points. The seeds can be selected based on prior information, or evenly, or random…

The similarity criteria is usually depending on the segmentation result we want. (Commonly used properties are the intensity level, color, texture, motion, …)

Pros: simple, works well on images with clear edges, prior knowledge can be easily utilized, robust to noise…

Cons: time consuming

Region Splitting and Merging

Let R represent the entire image region and P be a predicate. The splitting and merging steps are alternating:

- We split the region R_i into 4 sub regions if $P(R_i) = \text{false}$
- We merge 2 neighboring regions R_i and R_j if $P(R_i \cup R_j) = \text{true}$

The minimum region size has to be selected.

Clustering in the Feature Space

A clustering algorithm is used to find structure in the data. The pixels are represented in the feature space. Usual features: colors, pixel coordinates, texture descriptors, …

Video Segmentation

In bottom-up segmentation we want to group pixels with similar properties together. In case of video segmentation motion is an important feature to extract object of interest from the irrelevant background.

Frame Difference

We detect motion based on the difference of two consecutive frames:

$$d_{i,j}(x,y) = \begin{cases} 1, & |f(x,y,i) - f(x,y,j)| > T \\ 0, & \text{otherwise} \end{cases}$$

where T is a predefined threshold.

Problems with frame difference:

- We detect only the contour of the moving object
- It is sensitive to the speed of the motion

Background Subtraction

Main steps

1. Model the background and create an image with only the background on it
2. Subtract the background from the current frame.
3. Threshold the difference frame to get the foreground mask

Single frame as background

Assumption: we have a frame which does not contain foreground objects. This method cannot handle changes of the background (illumination changes, object becomes part of the background, …) Because of the above constraints it cannot be used in a real life scenario.

Running average

The following equation is used to calculate the background (B) for each time instance:

$$B(x,y,i+1) = \alpha \cdot B(x,y,i) + (1-\alpha) \cdot f(x,y,i)$$

where α is a predefined constant, which defines the adaptivity of the background.

Running average with foreground masking

The following equation is used to calculate the background (B) for each time instance:

$$B(x,y,i+1) = \begin{cases} B(x,y,i), & \text{if } (x,y) \text{ is a foreground pixel} \\ \alpha \cdot B(x,y,i) + (1-\alpha) \cdot f(x,y,i), & \text{otherwise} \end{cases}$$

where α is a predefined constant, which defines the adaptivity of the background.

Temporal Histogram

The temporal histogram of each pixel is used to generate the background image. The value with the highest peak is the value of the background pixel at that location.

Mixture of Gaussians

Using the temporal histogram of each pixel means we have to store the histograms and have to find its maximum for each pixel for each frame. This means high memory and high computational power requirements! Better idea is to use Mixture of Gaussians to estimate the background. The main parameters of the model:

- K is the number of Gaussians,
- the mean (μ_i), the variance (σ_i) and the weight (w_i) of each Gaussian $(i = 1, \dots, K)$

A pixel is considered background if it belongs to the Gaussian with the highest weight:

$$M(x,G) = \begin{cases} 1, & \left|\frac{x-\mu}{\sigma}\right| < T \\ 0, & \text{otherwise} \end{cases}$$

where T is a predefined constant.

Parameter updating:

- Weight updating:

$$w_i(t+1) = (1-\alpha) \cdot w_i(t) + \alpha M(x,G_i), \qquad w_i = \frac{w_i}{\sum_{j=1}^{K} w_j}$$

- Gaussian parameter updating:
  If there is a matching Gaussian $(M(x,G_i) = 1)$ we update its parameter as follows:

$$\mu_i(t+1) = (1-\rho) \cdot \mu_i(t) + \rho \cdot x$$
$$\sigma_i^2(t+1) = (1-\rho) \cdot \sigma_i^2(t) + \rho(x-\mu_i)^2$$

  If there is no matching Gaussian we replace the Gaussian with the lowest weight with a new Gaussian with the following parameters:

$$\mu_i(t+1) = x, \qquad \sigma_i^2(t+1) = \sigma_{\text{init}}^2, \qquad w_i(t+1) = w_{\text{init}}$$

Morphological Operations

Morphological operations are affecting the form, structure or shape of an object. They are used in pre- or postprocessing (filtering, thinning, and pruning) or for getting a representation or description of the shape of objects/regions (boundaries, skeletons convex hulls). Two basic operations:

- Dilation: expands the object, fills in small holes and connects disjoint objects.
- Erosion: shrinks objects by removing (eroding) their boundaries.

The basic idea in binary morphology is to probe an image with a structuring element (a simple, pre-defined shape), drawing conclusions on how this shape fits or misses the shapes in the image.

Dilation

A shift-invariant operator that expands the object, fills in small holes and connects disjoint objects. Steps:

- The structuring element is placed on each pixel on the image.
- If the pixel belongs to the foreground pixel, we do nothing.
- If the pixel belongs to the background, we change it to a foreground pixel if any pixel covered by the structuring element is a foreground pixel.

Erosion

A shift-invariant operator that erodes away the boundaries of regions of foreground pixels. Thus areas of foreground pixels shrink in size, and holes within those areas become larger. Steps:

- The structuring element is placed on each pixel on the image
- If the pixel is a background pixel, we do nothing
- If the pixel is a foreground pixel, we change this pixel to a background if any pixel covered by the structuring element is a background pixel.

Using dilation and erosion

Erosion on the image has the same effect as dilatation on the inverse image.
Opening: Erosion + Dilation
Closing: Dilation + Erosion

Topic 10 Image and Video Compression

Compression is the reduction of the number of bits used for the representation of an image or video, while

- being able to exactly reconstruct the original data **(lossless compression)**
- maintaining an acceptable quality of the reconstructed data **(lossy compression)**

Why are the signals compressible?

- The signals contain **redundancy** (spatial or temporal), they have a structure which can be described in a more compact way.
- There are parts of the image which are perceptually irrelevant, which can be discarded.

Lossless Compression

Reversible process: the original data can be **exactly** reproduced from the compressed data. Only limited **compression ratio** can be achieved with lossless compression, determined by the **entropy** of the source data. There is a tradeoff between:

- Efficiency (compression ratio)
- Complexity (required memory, computational power, etc.)
- Coding Delay (how long does it take to code the signal)

The two main groups of lossless coding techniques:

1. Statistical methods: the statistics of the source is known. (e.g. Huffman coding, Extended Huffman coding)
2. Universal methods: the statistics of the source is unknown (e.g. Arithmetic methods, Dictionary methods, Adaptive Huffman coding)

Background

Source: any information generating process can be viewed as a source that emits random sequence of symbols, from a finite alphabet.

Discrete Memoryless Source (DMS): the generated successive symbols are independent identically distributed random variables. This is the simplest model which can be described by its symbols and the associated probabilities. However, it is not perfect, in the above examples there is a dependency among the symbols.

Self information: How much information is provided by the emission of a certain symbol? The occurrence of a less probable event provides more information:

$$I(s_i) = \log\left(\frac{1}{p_i}\right) = -\log(p_i), \quad \text{where} \quad \begin{matrix} S = \{s_1, \dots, s_n\} \\ \{p_1, \dots, p_n\} \end{matrix}$$

In case of independent symbols:

$$I(s_1 s_2) = \log\left(\frac{1}{p(s_1 s_2)}\right) = \log\left(\frac{1}{p_1 p_2}\right) = -\log(p_1) - \log(p_2) = I(s_1) + I(s_2)$$

Entropy: It is the property of the source not only one symbol. For a DMS it is the average information per symbol:

$$H(S) = \sum_{i=1}^{n} p_i I(s_i) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Simple example for the entropy of a two-symbol alphabet:

$$H = -p\log_2(p) - (1-p)\log_2(1-p)$$

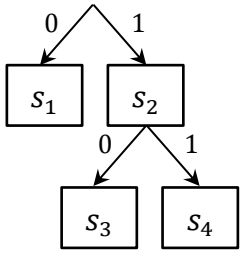
The entropy is the maximum if the probability of the symbols is equal. The entropy is minimum if one symbol has probability 1, the others 0.

Uniquely Decodable (UD): any finite sequence of code words corresponds to only one message sequence.

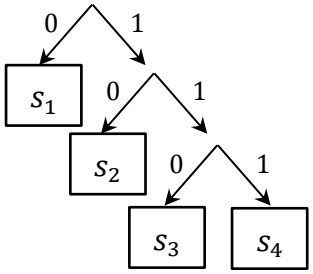
Prefix Code: no codeword is a prefix to an other codeword. A prefix code is always UD, but a UD code is not necessarily prefix. Prefix codes are easy to design and easy to decode. Every UD code can be converted to a prefix code with same rate.

Binary Tree representation

Not prefix code



Prefix code



First order codes

Encodes each symbol independently of the others, every symbol has a code word.

Block codes

Group the symbols of the source S into N length blocks and generate a codeword for each block. It can be regarded as a new source (S_N) that generates symbols from an alphabet with n^N number of symbols (where n is the number of symbols in S).

$$H(S_N) = N \cdot H(S)$$

Non-block codes

The non-block codes are arithmetic codes and Lempel-Ziv codes.

Lossless coding pipeline

$$\text{source} \rightarrow \text{symbols: } s_1, s_2, \dots, s_n \rightarrow \text{coding} \rightarrow \text{code words with length: } l_1, \dots, l_n$$

The average code word length is the measure of code efficiency:

$$l_{avg} = \sum_{i=1}^{n} l_i p_i$$

Shannon's Source Coding Theorem

Let S be a source with alphabet size n and entropy $H(S)$ and let consider coding N source symbols into one binary codeword (block coding). Then for every $\delta > 0$ it is possible by choosing the N large enough, to construct a code with average number of bits per symbol l_{avg} that satisfies the following inequality:

$$H(S) \leq l_{avg} < H(S) + \delta$$

This means that entropy is the lower bound of the code efficiency, we cannot beat it but we can come arbitrarily close to it by increasing N. Increasing N results larger dictionary and a delay in decoding. In general it is not straightforward to calculate entropy (the formula is only for DMS).

Huffman Coding

Variable length, prefix code. Based on the Morse principle: the more common symbols have shorter code word.

Algorithm

1. Sort the symbols according to their probability.
2. Combine the two least probable symbol to a composite symbol with probability equal the sum of the probabilities of its components.
3. Repeat the first two steps until only one composite symbol remains. (if the alphabet has n symbols, the algorithm will have $n-1$ steps)

The output of the algorithm is a binary tree that describes the code. The result is not unique.

Properties

- The average codeword length is bounded:
$$H(S) \leq l_{avg} < H(S) + 1$$
- If the maximum probability is less than 0.5:
$$H(S) \leq l_{avg} < H(S) + p_{\max}$$
- If the maximum probability is higher than 0.5:
$$H(S) \leq l_{avg} < H(S) + p_{\max} + 0.086$$
- Best possible outcome is achieved when the probabilities are $p_i = 2^{-i}$:
$$l_{avg} = H(S)$$
- **In general** Huffman codes work better with **large alphabet,** since then the max probability is likely to be lower than 0.5, which **means lower upper bound.**

Extended Huffman Coding

The extended Huffman coding uses block code. Therefore it produces a more efficient code:

$$H(S_N) = N \cdot H(S) \quad \text{and} \quad l_{avg} = N \cdot l_{avg}$$
$$\downarrow$$
$$H(S_N) \leq l_{avg_N} < H(S_N) + 1$$
$$N \cdot H(S) \leq N \cdot l_{avg} < N \cdot H(S) + 1$$

The drawback is that the number of codewords increases exponentially which leads to storage, computational and delay problems.

Arithmetic Coding

Coding sequences of symbols or blocks together is more efficient than generating codewords for each symbol separately. The problem with Huffman coding is that the alphabet can get huge easily: if we want to assign codeword to an N-length block of symbols, we have to assign codewords to all possible N-length blocks.

In arithmetic coding a unique **tag** is generated for a sequence of symbols, and then the tag is coded into a binary code. One possible source of tags is the numbers between 0 and 1. Since there are infinitely many real numbers in this interval, we can generate a tag for arbitrary long sequence of symbols.

Tag generation

We want to map a sequence of symbols onto an interval. The cumulative distribution function will be used. Let S be an alphabet with n symbols and known probabilities:

$$S = \{s_1, s_2, \dots, s_n\}, \qquad P(s_i), \qquad \sum_{i=1}^{n} P(s_i) = 1$$

Let X be a random variable that maps the event of the appearance of a symbol onto the real line:

$$X(s_i) = i, \qquad P(X = i) = P(s_i)$$

The cumulative distribution function is the following:

$$F_X(k) = \sum_{i=1}^{k} P(s_i)$$

We map a sequence of symbols onto an interval using the cumulative distribution function. A sequence of symbols will be described by a tag (number between 0 and 1) and the number of symbols coded by the tag. The tag itself is coded.

1. We can find the first symbol by placing the number on the 0-1 interval and check which symbol's interval it is placed on.
2. Proportionally map the division of the original intervals to the selected interval and check again which interval the tag is on.
3. Repeat the first 2 steps as many times as many symbols are coded by the tag.

Comparison with Huffman coding:
AC has higher upper bound:

$$H(S) \le R_{Huffman} < H(S) + \frac{1}{N}$$

$$H(S) \le R_{AC} < H(S) + \frac{2}{N}$$

As the number of the symbols (coded by one tag) increases, the precision of the tag has to be increased too. AC does not suffer from the exponentially increasing alphabet size. In practice better rates can be achieved with AC. Arithmetic coding is used taking the local (past) neighborhood statistics into account.
Progressive transmission:

- First a low resolution version of the image is transmitted.
- Then if the higher resolution image is required, it can be coded assuming that the low res image is available at the decoding side.

Dictionary Coding
In many applications there are frequently repeated patterns emitted by the source. It can be efficient to create a list (a dictionary) of the most frequent patterns, so they can be encoded by their address in the dictionary. The source is split into two parts:

- Frequently appearing patterns (coded by the dictionary address)
- Infrequently appearing patterns (coded with a less efficient technique)

The dictionary can be static or adaptive.

Predictive Coding
In predictive coding we use the "past" of the signal to predict the "present". If we use the same prediction model at encoding and decoding, only the prediction error (the unpredictable part) needs to be coded for lossless compression:

$$\text{Reconstructed signal} = \text{Prediction} + \text{Prediction Error}$$

The prediction error can be coded with some of the lossless compression methods. It is better to encode the prediction error instead of the original signal, because the entropy of the original signal is higher than the entropy of the prediction error.

Lossy Compression

Main steps of a lossy data compression system:

1. Redundancy Removal:
   - Predictor or Transformation
2. Entropy Reduction:
   - Scalar quantization
   - Vector quantization
3. Lossless Coding:
   - Huffman coding
   - Arithmetic coding
   - LZ coding techniques

Scalar Quantization

Uniform quantization
Irreversible process: all the values in the same interval will receive the same quantized value.

Non-uniform quantization
The overall error can be reduced by using non-uniform quantizer. The region with more values can be quantized with higher frequencies.
Instead of using a non-uniform quantizer first we use histogram transformation to stretch or compress the dynamic range of the input image. After that we use a uniform qunatizer in the compression process. At the end we invert the transformation used for histogram stretching or compression.

Vector Quantization

The input image is divided into small blocks, which are coded using a look-up table. The parameters of the codebook defines the compression rate. If we use n by n block size and each pixel is represented by b bites, and the length of the codebook is $L = 2^l$, which means l bits are needed to code an entry of the codebook, then

$$\text{rate} = \frac{n \cdot n \cdot b}{l}$$

Vector quantization is more effective than scalar quantization, as it can take into account the correlation in the data.

Generalized Lloyd Algorithm
With a set of training images it finds a locally optimal codebook:

1. Start with an n sized random initial codebook.
2. Partition the training vector set using the current codebook by assigning each training vector to the nearest vector in the codebook.
3. Calculate the centroid of each cluster. These centroids will form the new codebook.
4. Repeat step 2 and 3 until the distance between the old and new codebook vectors are below a predefined threshold.

This algorithm will only find local optimum. The final result will be sensitive to the initialization of the codebook.

Transform Coding

Very popular approach, it is part of most of the current image and video coding standards. Basic idea is to decorrelate the data with a suitable transformation, so that the transformation coefficients will describe the image perfectly. This transformation

- has to decorrelate the data
- has to compact the energy of the image
- has to have image independent basis
- has to have a fast implementation

We do a coarse or fine quantization of the transformation coefficients based on their significance (their variance, or their contribution to the total energy of the image).

Encoding

$$\text{image block} \rightarrow \text{transformation} \rightarrow \text{quantization} \rightarrow \text{entropy coding} \rightarrow 0110101$$

Decoding

$$0110101 \rightarrow \text{entropy decoding} \rightarrow \begin{matrix}\text{"inverse"}\\ \text{quantization}\end{matrix} \rightarrow \begin{matrix}\text{"inverse"}\\ \text{transformation}\end{matrix} \rightarrow \begin{matrix}\text{reconstructed}\\ \text{image block}\end{matrix}$$

Linear transformations

Karhunen-Loeve Transformation:	Discrete Cosine Transform:
– Statistically optimal – Basis functions are image dependent	– Close to KLT for typical images – Basis functions are image independent – Efficient implementation exist – Wildly used in image/video compression standards

Given an N by N orthonormal matrix set: $\varphi^{(u,v)}$
Then any N by N image can be represented as follows:

$$f = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \cdot \varphi^{(u,v)}$$

where

$$F(u,v) = f \cdot \varphi^{(u,v)}, \qquad \begin{matrix} u = 0, \dots, N-1 \\ v = 0, \dots, N-1 \end{matrix}$$

The 8 by 8 basis matrices for DCT:

- In the first row we have a cosine function with increasing horizontal frequency.
- In the first column we have a cosine function with increasing vertical frequency.

JPEG (Joint Photographic Experts Group)

International standard since 1991. Capable of compressing continuous-tone still images (gray-scale and color images) with ratio 10-50

Algorithm

1. •Uses DCT on 8×8 blocks:
   - The blocks' grey level is shifted by -128 to the range $[-128,\ 127]$.
   - The first coefficient is called DC, the rest of the coefficients AC coefficient.
2. The DC coefficients of the blocks are quantized, then coded differentially.
3. The AC coefficients are first quantized, vectorized by zig-zag scan and then entropy coded.
4. •The quantizer is uniform, using quantization tables with different step sizes for the different frequencies (in general higher step sizes for the higher frequency coefficients).