Computer Vision

Anthony Cioppa, Marc Van Droogenbroeck

Academic year: 2024-2025

Practical informations I

- Instructors: Anthony Cioppa and Marc Van Droogenbroeck
- Assistant: Renaud Vandeghen
- Slides: `http://orbi.uliege.be`
- Evaluation
  1. personal project (split in several sub-tasks): 3 students, evaluated in December.
     The project is compulsory!
     If a sub-task is not done, then the student gets a note of 0 for the sub-task AND following tasks.
     No possibility to resubmit in August!
  2. written exam (also compulsory!), closed book, during the exam session.
  3. [*August only*] written examination.
  4. Notation:
     1. January: 2/3 for the project, 1/3 for the exam
     2. August/September: 1/2 for the project, 1/2 for the exam

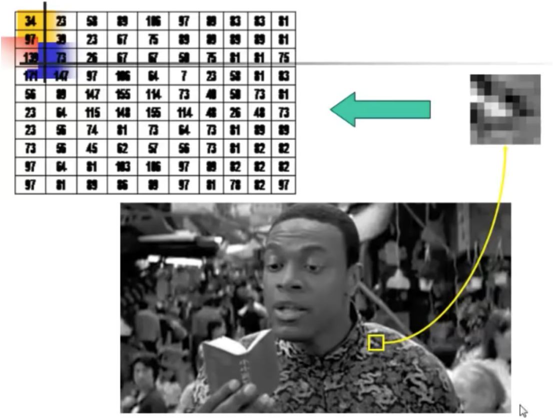
Practical informations II

- Hands on computer vision ⇒ “practice” computer vision

Reference book

- Szeliski R., *Computer Vision: Algorithms and Applications*, second edition, Springer, 2022.

The 4 “pillars” of computer vision I



34	23	58	89	106	97	89	83	83	81
97	38	23	67	75	89	89	89	89	81
139	73	26	67	67	58	75	81	81	75
171	147	97	106	64	7	23	58	81	83
56	89	147	155	114	73	48	58	73	81
23	64	115	148	155	114	48	26	48	73
23	56	74	81	73	64	73	81	89	89
73	56	45	62	57	56	73	81	82	82
97	64	81	103	106	97	89	82	82	82
97	81	89	86	89	97	81	78	82	97

The 4 “pillars” of computer vision II

A Computer Vision (CV) workflow

1. Acquisition (sensors, camera, representation)
2. → Processing (filtering, signal “shaping”)
3. → Computer vision (CV) algorithms (generic algorithms, not application-tuned)
4. → Application

The 4 “pillars” of computer vision III

1. Acquisition
    1. Human perception (color, reflection)
    2. Sensor, camera
    3. 3D vision, video
2. Processing
    1. Linear filters
    2. Morphological tools (openings, geodesic distance)
3. “Generic” CV algorithms
    1. Classical tools for images (edge detection, watershed, granulometry)
    2. Motion detection
    3. Data-driven tools (machine learning / deep learning)
4. Application
    1. Tasks: counting, segmentation, detection, tracking
    2. Evaluation

The 4 “pillars” of computer vision IV

- Don’ forget the *acquisition* step
  - understanding your data is essential (also for machine learning applications!)
- *Linear* framework → *non-linear* frameworks
- A world of trade-offs (computational load ↔ framerate, etc.)
- There is *no unique*, universal, *solution* to a computer vision problem
- Lectures in the spirit of a “*catalog*”

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Outline

1. Fundamentals of 2D imaging
    - Color perception and representation
    - Data representations
    - Applications
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Topics

- Elements of visual perception
  - Colors: representation and colorspaces
  - Transparency
- Data structure for images
- Examples of industrial applications:
  - Segmentation
  - Optical character recognition

Outline

1. Fundamentals of 2D imaging
   - Color perception and representation
   - Data representations
   - Applications
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Human visual system, light and colors I

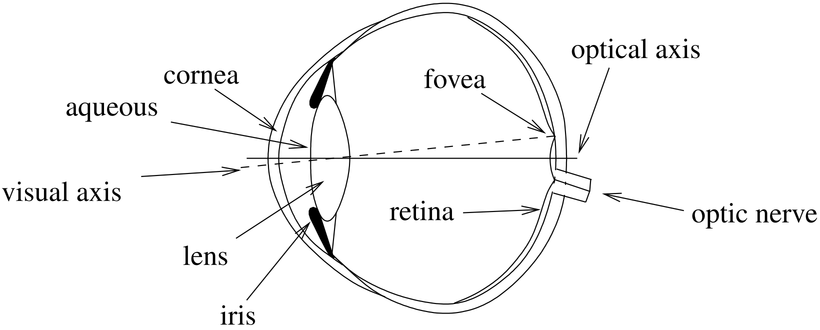


Figure: Lateral view of the eye globe (*rods* and *cones* are receptors located on the retina).

Human visual system, light and colors II

color	wavelength interval $\lambda\,[m]$	frequency interval $f\,[Hz]$
purple	$\sim$ 450–400 $[nm]$	$\sim$ 670–750 $[THz]$
blue	$\sim$ 490–450 $[nm]$	$\sim$ 610–670 $[THz]$
green	$\sim$ 560–490 $[nm]$	$\sim$ 540–610 $[THz]$
yellow	$\sim$ 590–560 $[nm]$	$\sim$ 510–540 $[THz]$
orange	$\sim$ 635–590 $[nm]$	$\sim$ 480–510 $[THz]$
red	$\sim$ 700–635 $[nm]$	$\sim$ 430–480 $[THz]$

Figure: Visible colors (remember that $\lambda = \frac{3\times10^8}{f}$).

Human visual system, light and colors III

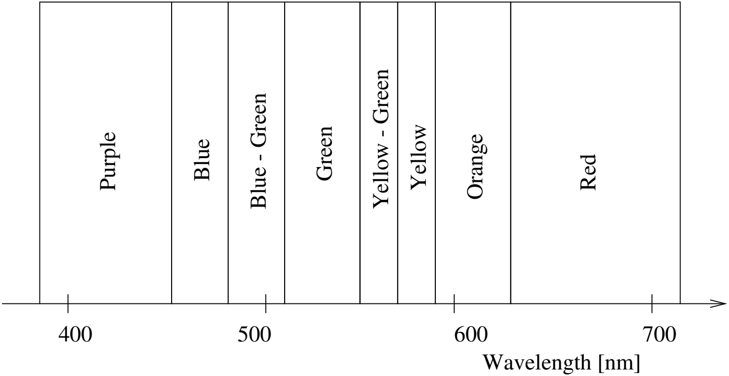


Figure: Colors on the visible spectrum.

Frequency representation of colors

$$\int_{\lambda} L(\lambda)\, d\lambda \qquad (1)$$

Impossible from a practical perspective because this would require *one sensor for each wavelength.*

Solution: use colorspaces

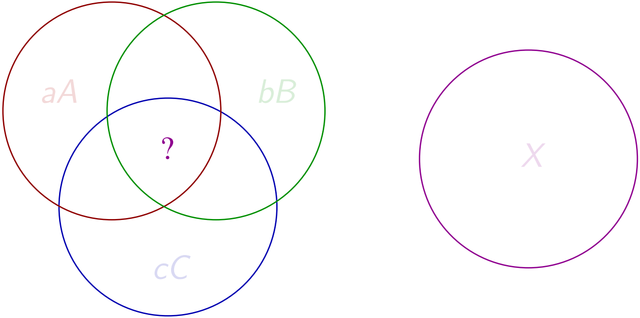


Figure: Equalization experiment for colors. The aim is to mix A, B, and C to get as close as possible to X.

Frequency representation of colors

$$\int_{\lambda} L(\lambda)\, d\lambda \tag{1}$$

Impossible from a practical perspective because this would require *one sensor for each wavelength.*

Solution: use colorspaces

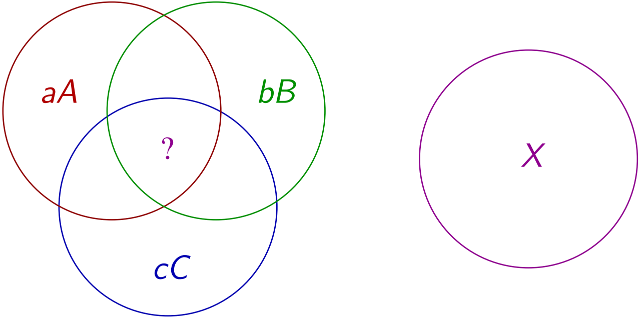


Figure: Equalization experiment for colors. The aim is to mix A, B, and C to get as close as possible to X.

The RGB additive colorspace

Three fundamental colors: red R $(700\,[nm])$, green G $(546,1\,[nm])$ and blue B $(435,8\,[nm])$,

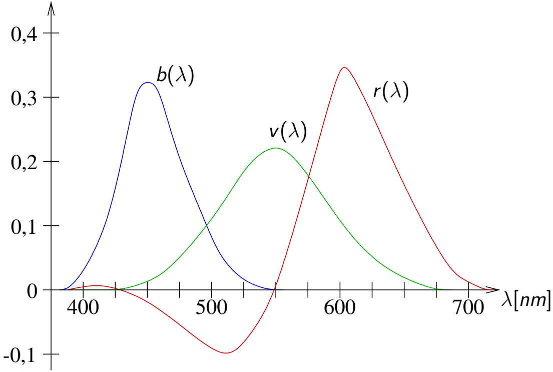
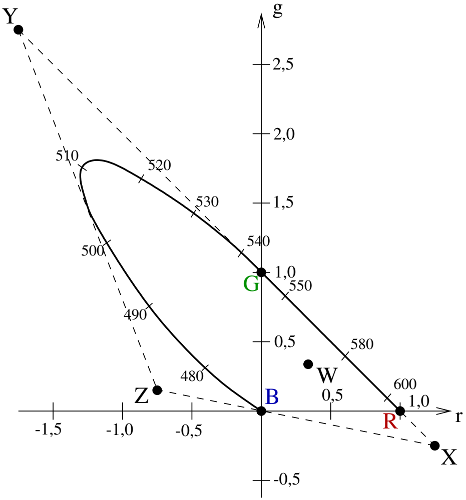


Figure: Equalization curves obtained by mixing the three fundamental colors to simulate a given color.

CIE chromatic diagram for RGB

We consider R+G+B=1, so positive RGB values are in a triangle



Notion of intensity

$$R+G+B = \text{intensity} \qquad (\text{intensity} \in [0,1])$$

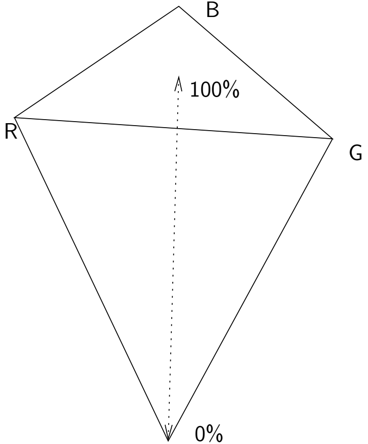


Figure: Pyramid derived from an RGB color representation.

Towards other colorspaces: the XYZ colorspace I

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2,769 & 1,7518 & 1,13 \\ 1 & 4,5907 & 0,0601 \\ 0 & 0,0565 & 5,5943 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2}$$

$$x = \frac{X}{X+Y+Z} \tag{3}$$

$$y = \frac{Y}{X+Y+Z} \tag{4}$$

$$z = \frac{Z}{X+Y+Z} \tag{5}$$

$$x+y+z = 1 \tag{6}$$

Towards other colorspaces: the XYZ colorspace II

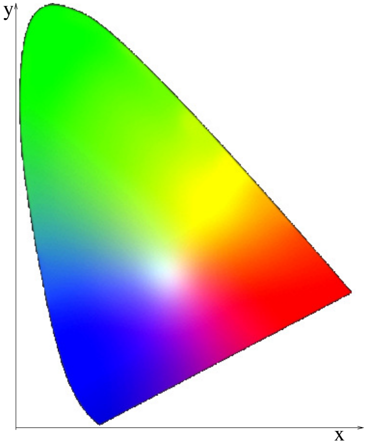


Figure: Approximative chromatic colorspace defined by two chrominance variables x and y.

Luminance

Luminance: $Y = 0.2126\times R + 0.7152\times G + 0.0722\times B$

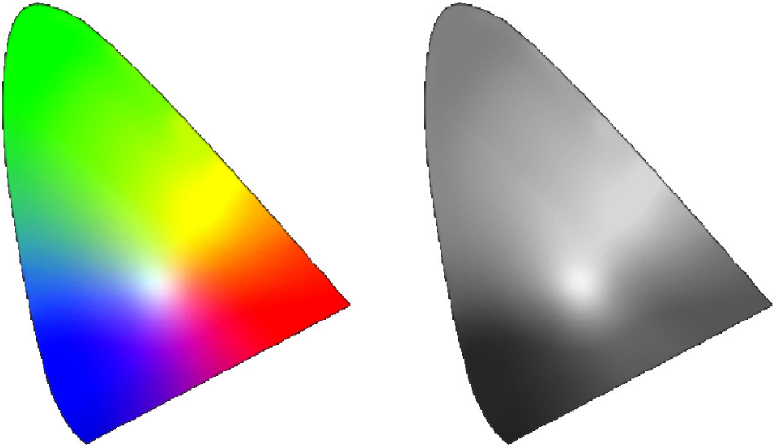
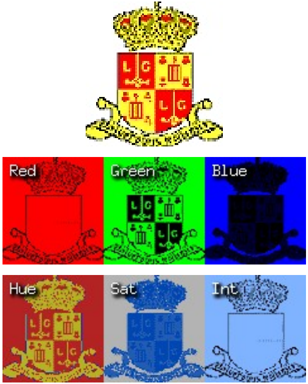


Figure: *xy* chromatic diagram and maximal luminance for each color.

The HSI colorspace

Colorspace that has a better physical meaning:

- hue
- saturation
- intensity



Other colorspaces

- a subtractive colorspace: Cyan, Magenta, and Yellow (CMY)
- Luminance + chrominances (YIQ, YUV or YC_bC_r)

In practice, we use 8 bits ($\equiv$ 1 byte) to describe one color channel (instead of a value between 0 and 1). So, colors $\equiv$ 24 bits.

Hexadecimal				Decimal		
00	00	00	⬛	0	0	0
00	00	FF	🟦	0	0	255
00	FF	00	🟩	0	255	0
00	FF	FF	🟦	0	255	255
FF	00	00	🟥	255	0	0
FF	00	FF	🟪	255	0	255
FF	FF	00	🟨	255	255	0
FF	FF	FF	⬜	255	255	255

Table: Two representations of RGB color values (8 bit per color channel).

Acquisition: three sensors

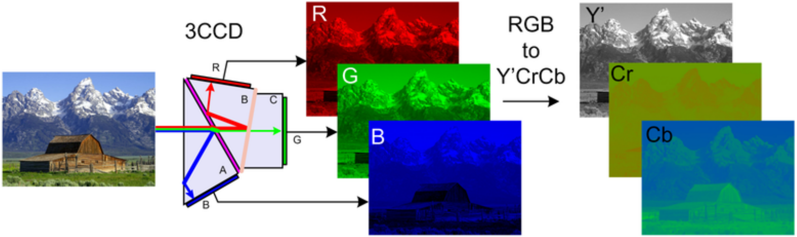


Figure: Acquisition of a $Y\,C_b\,C_R$ signal [Wikipedia]

There are variations, such as the $Y\,U\,V$ colorspace, mainly developed for compression:

1. information concentrated in the Y channel ⇒ better compression.
2. better decorrelation between channels.

Acquisition: one sensor + Bayer filter I

A Bayer filter mosaic is a color filter array for arranging RGB color filters on a square grid of photo sensors.

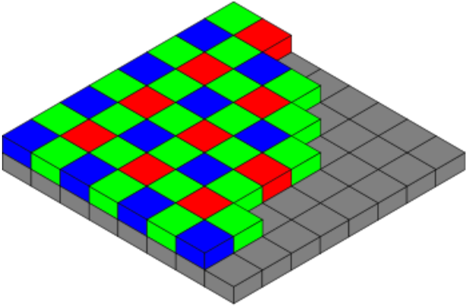


Figure: The Bayer arrangement of color filters on the pixel array of an image sensor.

Acquisition: one sensor + Bayer filter II

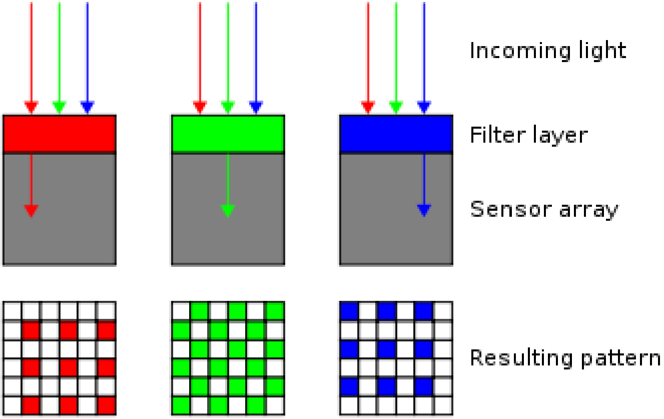


Figure: Profile/cross-section of sensor.

Bayer filter: practical considerations

- Most mono-sensor cameras use the Bayer pattern, except for professional 3CCD cameras (three sensor planes + prism to split the incoming light)
- The filter pattern is 50% green, 25% red and 25% blue. Why?
- We only have one value per pixel. Other values are re-built by interpolation, but they might not even exist... !
- For compression or processing,
  - 1 sensor plane ⇒ normally only one byte to process. Possible if the processing occurs in the sensor, not anymore if the interpolated image is forwarded.
  - 3 sensor planes ⇒ 3 planes to process or to compress.
    Expected compression rate: more or less the same as for a grayscale image.
- It might be wiser, for processing, to have a black-and-white (or a monochromatic, such as red) camera, instead of a color camera.

Bayer filter: practical considerations

- Most mono-sensor cameras use the Bayer pattern, except for professional 3CCD cameras (three sensor planes + prism to split the incoming light)
- The filter pattern is 50% green, 25% red and 25% blue. Why?
- We only have one value per pixel. Other values are re-built by interpolation, but they might not even exist... !
- For compression or processing,
  - 1 sensor plane ⇒ normally only one byte to process. Possible if the processing occurs in the sensor, not anymore if the interpolated image is forwarded.
  - 3 sensor planes ⇒ 3 planes to process or to compress. Expected compression rate: more or less the same as for a grayscale image.
- It might be wiser, for processing, to have a black-and-white (or a monochromatic, such as red) camera, instead of a color camera.

Visual effects

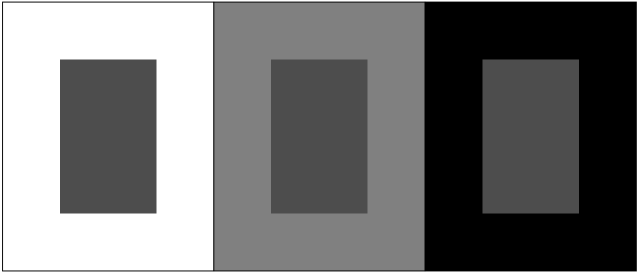


Figure: Illustration of a masking visual effect.

A machine does not take psycho-visual effects into account. But, for evaluating the *subjective quality* of a compression algorithm, one has to!

Outline

1. Fundamentals of 2D imaging
   - Color perception and representation
   - Data representations
   - Applications
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Sampling grid and frame organization

- Each sample located on a grid is named a *pixel* (which stands for *picture element*).
- There are two common sampling grids and they induce certain types of connectivity (defined the *number of neighbors*).

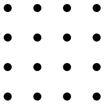
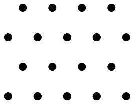
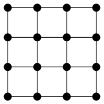
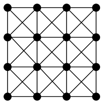
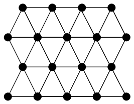
Square *grid*		Hexagonal *grid*
		
4-*connectivity*	8-*connectivity*	6-*connectivity*
		

Table: Types of grid and associated connectivities.

Data structure for dealing with images

- Typical data structure for representing images: **matrices** (or 2D tables), vectors, trees, lists, piles, . . .
- For **matrices**, there are several ways to organize the memory:
  - 1 channel (for example, luminance):
    - upper left corner coordinate correspond to the $(0, 0)$ location in the matrix.
    - one byte per pixel
  - 3 channels, such RGB images. We have two options:
    1. store 3 matrices separately (according to the 1 channel storage scheme)
    2. store the RGB values consecutively (intermixed) for each pixel. For example, if $R_1G_1B_1$ and $R_2G_2B_2$ are the color values of two consecutive pixels, then we could store the values as $R_1G_1B_1R_2G_2B_2$ in a single matrix.

The bitplanes of an image

A grayscale image is an array of bytes like 1 0 1 1 0 0 0 0.

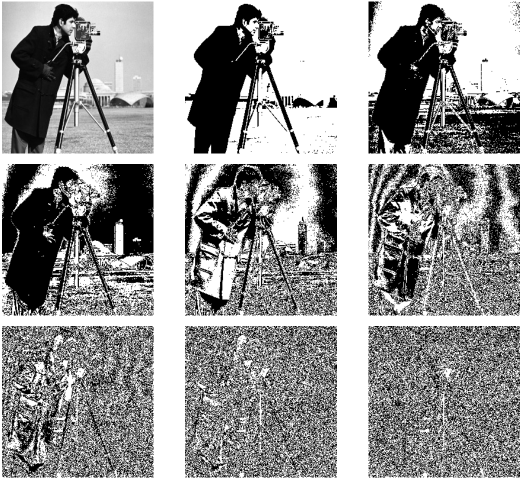


Table: Image and its 8 bitplanes starting with the Most Significant Bitplane (MSB) to the Least Significant Bit (LSB).

Typology of images and videos

2D. This types refers to a “classic” image and is usually expressed as a 2D array of values. It might represent the luminance, a color, depth, etc.

3D. 3D images are obtained with devices that produce 3D images (that is with x, y, z coordinates). Medical imaging devices produce this type of images.

2D+t. t refers to time. Therefore, $2D + t$ denotes a video composed over successive 2D images, indexed by t.

3D+t. $3D + \mathrm{t}$ images are in fact animated $3D$ images. A typical example is that of animated 3D graphical objects, like that produced by simulations.

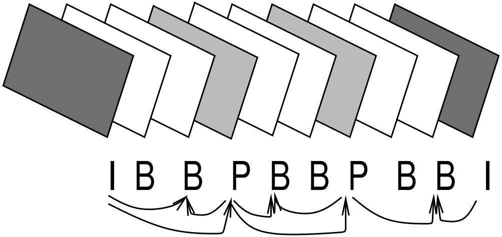
Compression I

Compression ratios: ITU-R 601 (ex **CCIR 601**)

- YC_bC_r
- 4:4:4 (720/720/720) $= 270\,[Mb/s]$
- 4:2:2 (720/360/360) $= 180\,[Mb/s]$

Two main principles:

1. remove some redundancies inside the frame (intraframe coding)
2. remove some redundancies between frames (interframe coding)



MPEG-2 levels and profiles

Profile	Simple	Main	SNR	Spatial	High
Low level (235 x 288 x 30Hz)		X	X		
Main level (720 x 576 x 30Hz)	X	X	X		X
High-1440 level (1440 x 1152 x 60Hz)		X		X	X
High level (1920 x 1152 x 60Hz)		X			X

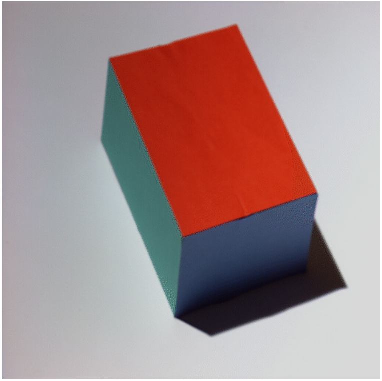
A 50 Hz video stream of size 720 × 576 (PAL) corresponds to 270 $[Mb/s]$ uncompressed

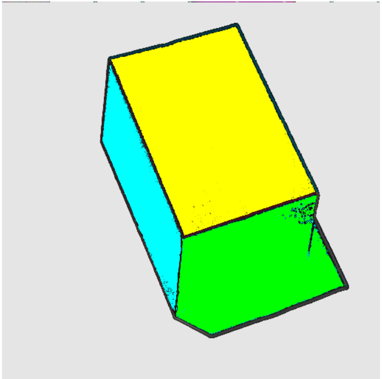
- *Lossless* compression: $\approx 100\,[Mb/s]$
- *Lossy* compression (but *intraframe* only): $\approx 27\,[Mb/s]$
- *Lossy* compression (including *interframe* coding): MPEG-2 format $\approx 5\,[Mb/s]$ or less

Outline

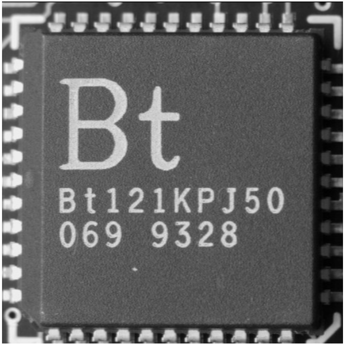
1. Fundamentals of 2D imaging
   - Color perception and representation
   - Data representations
   - Applications
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

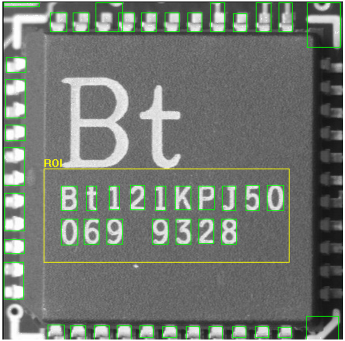
Image segmentation





Character recognition

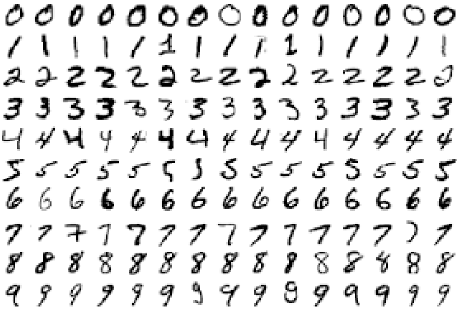




Several successive stages in a classical approach:

- Selection of a Region of Interest (ROI). Processing is limited to that area.
- Detection of edges (contours).
- Identification and classification of characters.

Example of classification task: classify handwritten digits



- Goal: **learn** to recognize handwritten digits (by **machine learning**)
- Dataset: the MNIST (Mixed National Institute of Standards and Technology) database of handwritten digits has
  - a *training set* of $60,000$ examples.
  - a *test/validation set* of $10,000$ examples.
  - *normalization*: the digits have been size-normalized and centered in a fixed-size image.

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
   - Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

What is filtering?

$f(x)$	25	27	30	24	17	15	22	23	25
x	0	1	2	3	4	5	6	7	8

$\min\{f(x+1), f(x), f(x-1)\}$		25	24	17	15	15	15	22	
$\max\{f(x+1), f(x), f(x-1)\}$		30	30	30	24	22	23	25	
"better" version of $f(x)$?									

But, *why* do we *filter* an image?

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
   - Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Linear filtering

- Fourier transform and filtering operation
- The notion of “ideal” filter
- Categories of ideal filters
- Typical filters:
  - Low-pass filters
  - High-pass filters
  - Gabor filters

Fourier transform: definition and interpretation I

Let us take $f(x, y)$ as the values of a single channel image (for example, the grayscale component):

- (x, y) are the coordinates
- $f(x, y)$ is the single-valued image of the pixel located at (x, y)

Definition ((Continuous) Fourier transform)

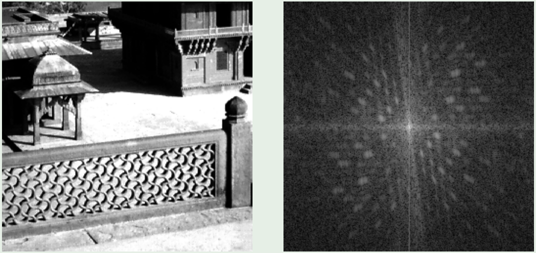
The (continuous) Fourier transform $\mathcal{F}(u, v)$ of $f(x, y)$ is defined by

$$\mathcal{F}(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-2\pi j(ux+vy)} dxdy \tag{7}$$

Fourier transform: definition and interpretation II

Example

The Fourier transform is a complex image (it has a real component and an imaginary component). Often, we are interested in the amplitude.



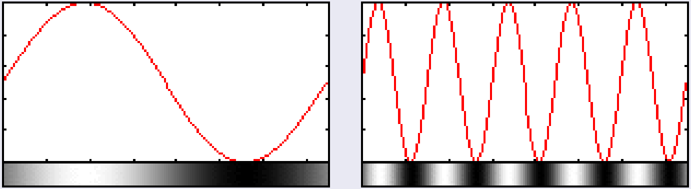
(a) (b)

Figure: (a) original image and (b) *centered* Fourier transform of its amplitude.

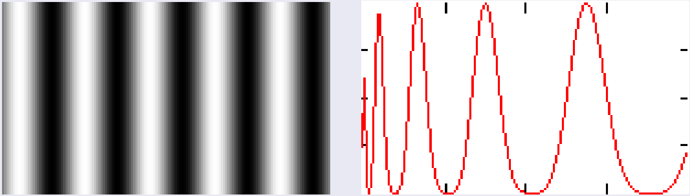
Fourier transform: definition and interpretation III

What is a spatial frequency?

Simple waveforms and profiles

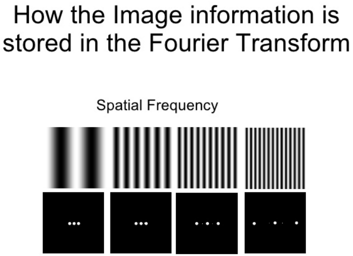


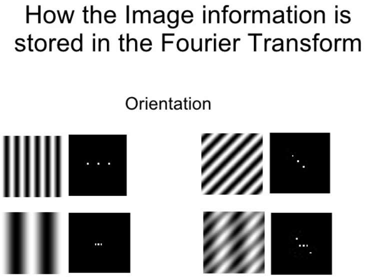
Complex waveform



Fourier transform: definition and interpretation IV

The Fourier transform is a representation of $f(x, y)$ in terms of spatial frequencies:


How the Image information is stored in the Fourier Transform
Spatial Frequency


How the Image information is stored in the Fourier Transform
Orientation

Fourier transform: definition and interpretation V

Definition ((Continuous) *inverse* Fourier transform)

The inverse (continuous) Fourier transform is defined by

$$f(x,y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\mathcal{F}(u,v)e^{2\pi j(ux+vy)}du\,dv \tag{8}$$

There is a one-to-one relationship between a $f(x,y)$ and $\mathcal{F}(u,v)$:

$$f(x,y)\rightleftharpoons\mathcal{F}(u,v) \tag{9}$$

Definition (Filtering)

Applying a filter $\mathcal{H}(u,v)$ corresponds to modify the Fourier coefficient of some frequencies

$$\mathcal{F}(u,v)\ \longrightarrow\ \mathcal{F}(u,v)\mathcal{H}(u,v) \tag{10}$$

Notion of “ideal” filter

A filter is said to be “ideal” if every transform coefficient is multiplied by 0 or 1.

Definition (*Ideal filter*)

An *ideal* filter is such that its transfer function is given by

$$\forall (u,v),\ \mathcal{H}(u,v) = 0 \text{ or } 1. \tag{11}$$

The notion of ideal filter is closely related to that of *idempotence*. The idempotence for a filter is to be understood such that, for an image $f(x,y)$,

$$\mathcal{F}(u,v)\mathcal{H}(u,v) = \mathcal{F}(u,v)\mathcal{H}(u,v)\mathcal{H}(u,v) \tag{12}$$

Typology of ideal filters I

For one-dimensional signals (such as the image function along an image line):

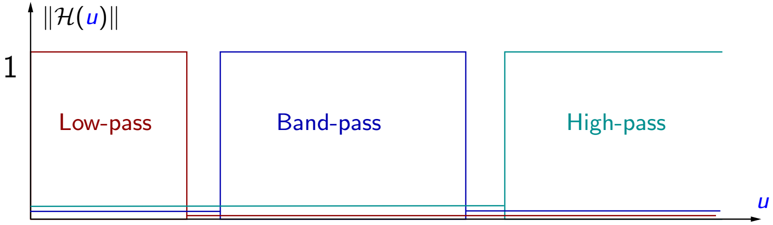


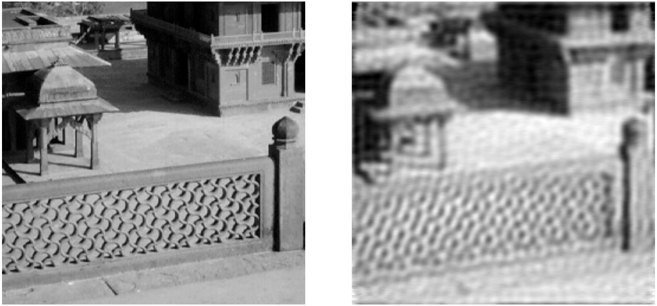
Figure: One-dimensional filters.

Typology of ideal filters II

There are three types of circular ideal filters:

- *Low-pass filters:*

$$\mathcal{H}(u,v) = \begin{cases} 1 & \sqrt{u^2+v^2} \leq R_0 \\ 0 & \sqrt{u^2+v^2} > R_0 \end{cases} \tag{13}$$



(a) Original image (b) Low-pass filtered image

Typology of ideal filters III

- *High-pass filters:*

$$\mathcal{H}(u,v) = \begin{cases} 1 & \sqrt{u^2+v^2} \geq R_0 \\ 0 & \sqrt{u^2+v^2} < R_0 \end{cases} \tag{14}$$

- *Band-pass filters.* There are equivalent to the complementary of a low-pass filter and a high-pass filter:

$$\mathcal{H}(u,v) = \begin{cases} 1 & R_0 \leq \sqrt{u^2+v^2} \leq R_1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Typology of ideal filters IV

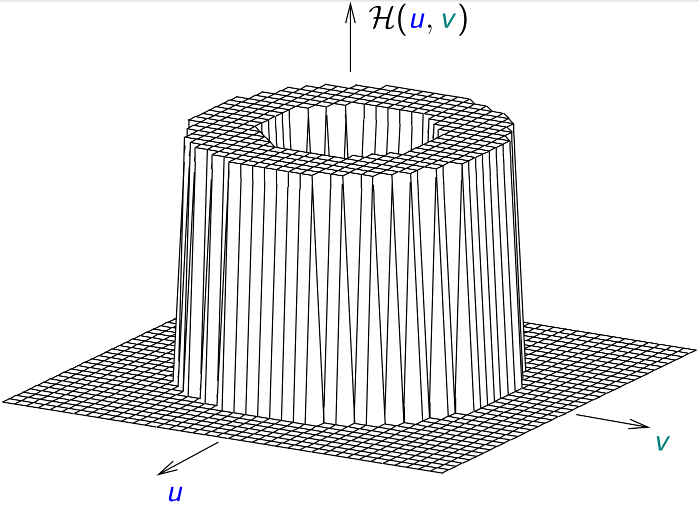


Figure: Transfer function of band-pass filters.

Effects of filtering

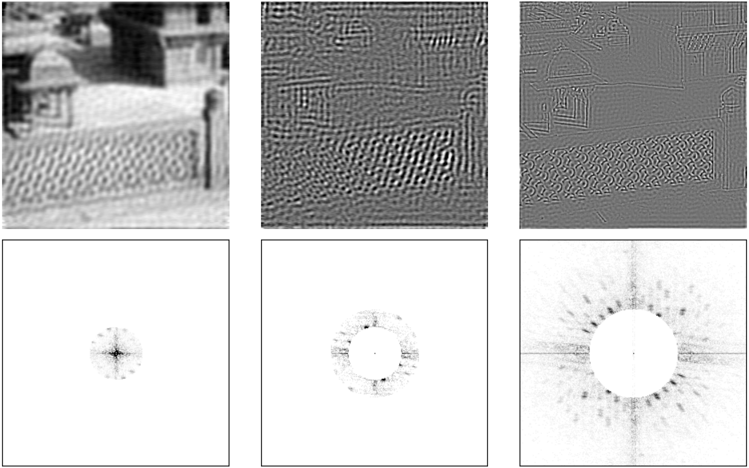


Figure: Fourier spectra of images filtered by three types of circular filters.

Low-pass filters I

A typical low-pass filter is the Butterworth filter (of order n) defined as

$$\mathcal{H}\left(u,v\right)=\frac{1}{1+\left(\frac{\sqrt{u^2+v^2}}{R_0}\right)^{2n}}. \tag{16}$$

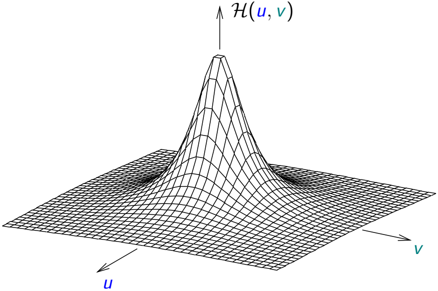
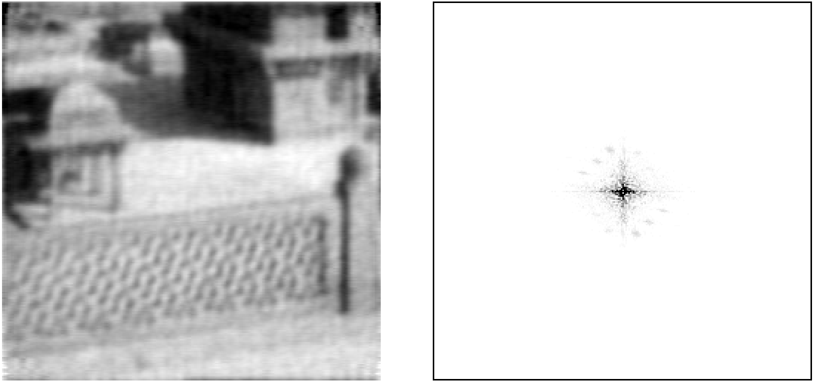


Figure: Transfer function of a low-pass Butterworth filter (with $n=1$).

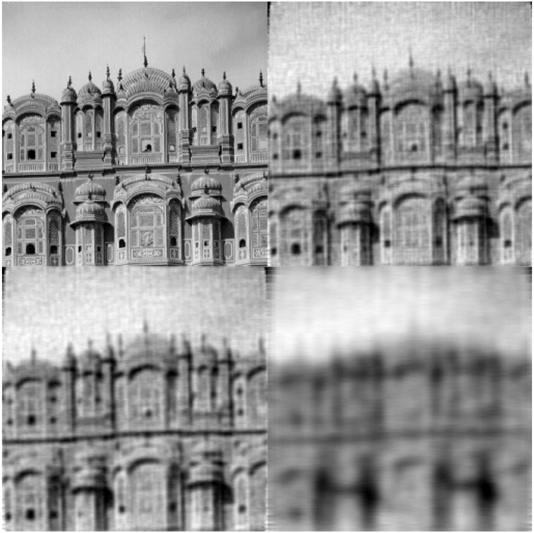
Low-pass filters II



(a) Input image (b) Spectrum of the filtered image

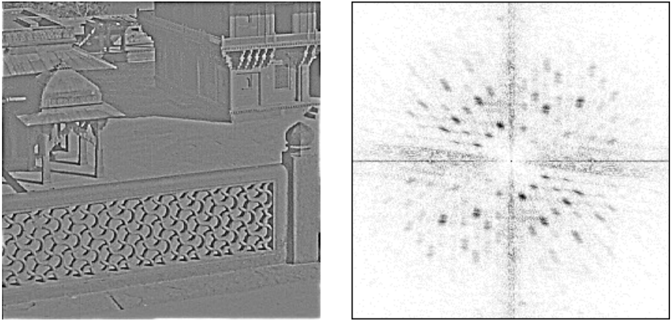
Figure: Effects of an order 1 Butterworth filter (cut-off frequency: $f_c = 30$).

Effect of a low-pass filter (with decreasing cut-off frequencies)



High-pass filters I

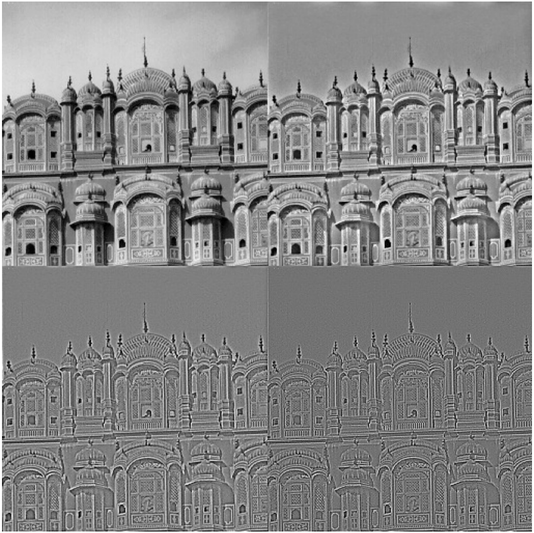
$$\mathcal{H}(u,v) = \frac{1}{1+\left(\frac{R_0}{\sqrt{u^2+v^2}}\right)^{2n}} \tag{17}$$



(a) Filtered image (b) Spectrum of (a)

Figure: Effects of an order 1 Butterworth filter 1 (cut-off frequency: $f_c = 50$).

Effect of a high-pass filter (with increasing cut-off frequencies)



Gabor filters I

Definition

Gabor filters belong to a particular class of linear filters. There are directed filters with a Gaussian-shaped impulse function:

$$h(x,y) = g(x',y')e^{2\pi j(Ux+Vy)} \tag{18}$$

- $(x',y') = (x\cos\phi + y\sin\phi\,, -x\sin\phi + y\cos\phi)$, these are the (x,y) coordinates rotated by an angle ϕ, and
- $g(x',y') = \frac{1}{2\pi\sigma^2}e^{(-(x'/\lambda)^2+y'^2)/2\sigma^2}$.

The corresponding Fourier transform is given by

$$\mathcal{H}(u,v) = e^{-2\pi^2\sigma^2[(u'-U')^2\lambda^2+(v'-V')^2]} \tag{19}$$

- $(u',v') = (u\cos\phi + v\sin\phi\,, -u\sin\phi + v\cos\phi)$, and
- (U',V') is obtained by rotating (U,V) with the same angle ϕ.

Gabor filters II

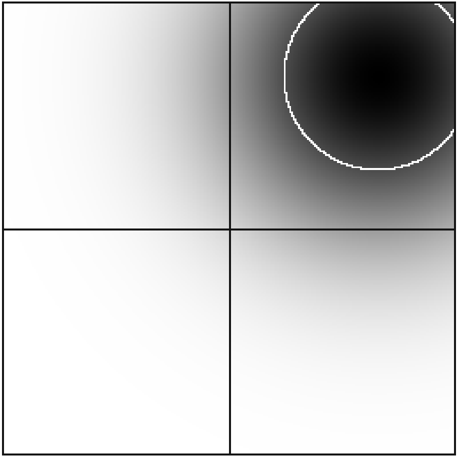


Figure: Transfer function of Gabor filter. The white circle represents the -3 [dB] circle ($=$ half the maximal amplitude).

Gabor filters III

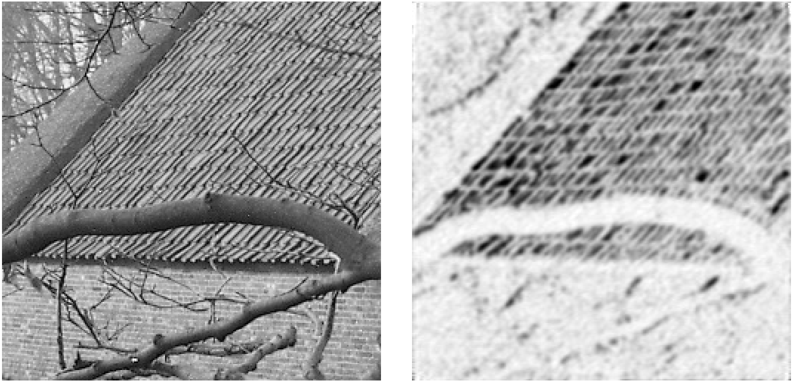


Figure: Input image and filtered image (with an filter oriented at 135^o).

Implementation

There are mainly 4 techniques to implement a Gaussian filter:

1. *Convolution* with a restricted Gaussian kernel. One often chooses $N_0 = 3\sigma \text{ or } 5\sigma$

$$g_{1D}[n] = \begin{cases} \frac{1}{\sqrt{2\sigma}} e^{-(n^2/2\sigma^2)} & |n| \leq N_0 \\ 0 & |n| > N_0 \end{cases} \tag{20}$$

2. *Iterative convolution* with a uniform kernel:

$$g_{1D}[n] \simeq u[n] \otimes u[n] \otimes u[n] \tag{21}$$

where

$$u[n] = \begin{cases} \frac{1}{(2N_0+1)} & |n| \leq N_0 \\ 0 & |n| > N_0 \end{cases} \tag{22}$$

3. Multiplication in the Fourier domain.
4. Implementation as a *recursive filter*.

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
   - Camera calibration: the pinhole model
   - Calibration algorithm
   - 3D reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Motivation

The physical world is in 3D (but flattened/2D by color cameras)

Why is it important?

We may want to:

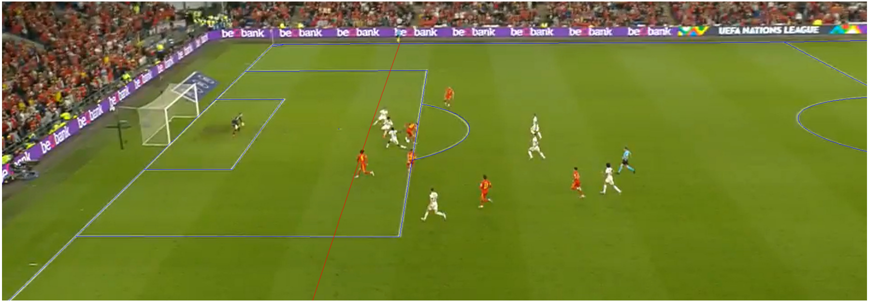
- Acquire 3D information (such as inferred/measured distances)
- Understand 3D geometry (in the eyes of a camera)
- Exploit 3D information for computer vision tasks
- Change the viewpoint of an observer

Two common tasks in 3D vision I

Camera calibration

Definition. Camera calibration is defined as the technique of estimating the characteristics of a camera.

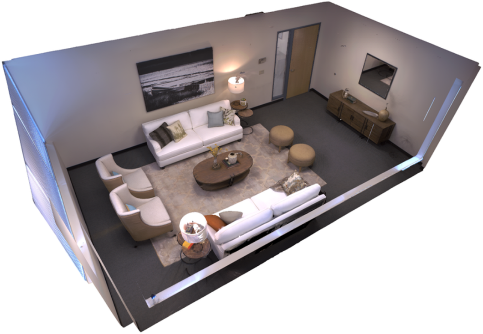
Goal. Determine an accurate relationship between (X, Y, Z), a 3D point in the real world, and (x, y), its corresponding 2D projection in the image acquired by a calibrated camera.



Two common tasks in 3D vision II

3D reconstruction

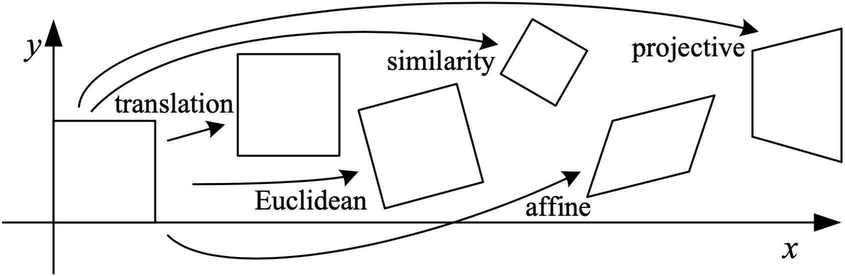
Definition. 3D reconstruction is the process of capturing the shape and appearance of real objects.



Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
   - Camera calibration: the pinhole model
   - Calibration algorithm
   - 3D reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Basic set of 2D planar transformations



Building a mathematical model for cameras I

A camera projects a 3D world onto a 2D image.
⇒ this is the concept of *projection*.

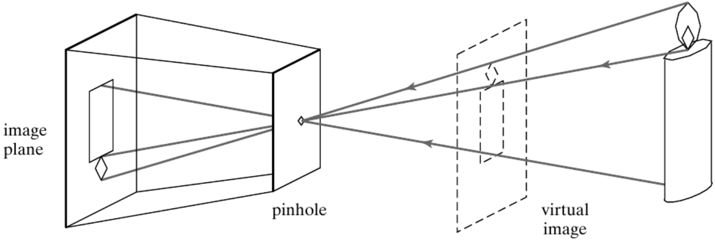
Remarks:

- when we apply a *transformation*, some characteristics are preserved. For example, when you translate an object, its dimensions and angles are preserved.
- translations and rotations (which can be expressed as the product of a matrix on the $(X, Y, Z)^T$ real world coordinates) preserves distances, angles, etc. These leads to so-called Euclidean transformations.
- But what is preserved in general?
  - distances? angles? parallelism? *alignment of pixel* ($\equiv$ *lines*)?

Building a mathematical model for cameras II

The real problem if that we have some ambiguities when we project an 3D object to a 2D plane.

One of the simplest model (and most common): *pinhole camera* (central projection)

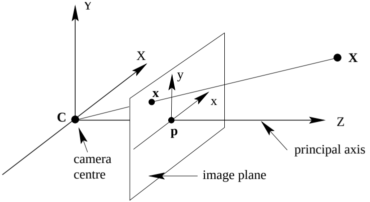


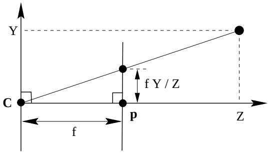
What is preserved?

- distances? angles? "parallelism"? No.
- *alignment of pixel* ($\equiv$ lines)? Yes

Pinhole camera model

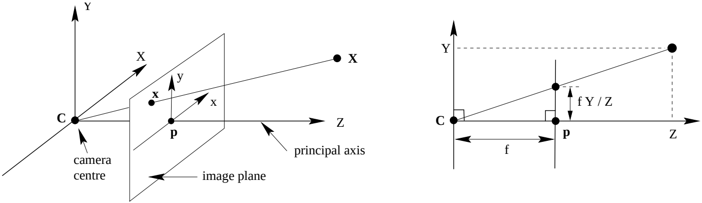
- All the light rays converge to a unique point (*camera center*) before arriving on the sensor plane.





- Vocabulary terms:
  - *camera center*: center of the central projection.
  - *image plane*: plane of the sensor.
  - *principal axis*: line passing through the camera center and orthogonal to the image plane.
  - *principal* point: intersection between the principal axis and the image plane.
  - f is called the *focal length*.

Mathematical model of the pinhole camera I



If $\mathbf{X} = (X, Y, Z)^T$ is a point in space[1] and $\mathbf{x} = (x, y)^T$ is its projection on the image plane, then *similar triangles* gives

$$\frac{x}{f} = \frac{X}{Z} \text{ and } \frac{y}{f} = \frac{Y}{Z} \tag{23}$$

Ambiguity: although f is fixed for a camera, *we cannot derive the absolute values of X and Y from the measured pair of values* (x, y).

⇒ a convenient representation is that of *homogeneous coordinates*

[1] Capital/uppercase letters denote coordinates in the real world.

Homogeneous coordinates I

Homogeneous coordinates

Idea: new representation, obtained by adding a trailing 1: $(x, y, 1)$

$$(x, y) \equiv (x, y, 1) \tag{24}$$

By definition, we assume that

$$(x, y) \equiv (\lambda x, \lambda y, \lambda) \tag{25}$$

so that all pixels with varying λ are equivalent. In other words, in homogeneous coordinates, a point may be expressed by an infinite series of (scaled) coordinates.

Note: this is similar to how we define rational numbers, that have many different equivalent representations:

$$\frac{1}{3} = \frac{2}{6} = \frac{13}{39} \tag{26}$$

Homogeneous coordinates II

Properties:

- $(x, y, 0)$ is not equivalent to (x, y). It is a special point.
- A point of $\mathbb{R}^n$ is represented by a vector of size $n+1$. Example:

$$\mathbf{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3 \mapsto \left(\frac{x_1}{x_3}, \frac{x_2}{x_3}\right)^T \in \mathbb{R}^2$$

- A line in a plane is represented by the following equation $ax + by + c = 0$. With homogeneous coordinates, this becomes:

$$\mathbf{l} = (a, b, c)^T$$

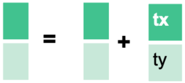
- A point belongs to a line if and only if $\mathbf{x}^T\mathbf{l} = 0$.
- Intersection between two lines: $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$ where $\times$ is the product between vectors.

Simple example to show the interest of homogeneous coordinates

Let us consider translations.

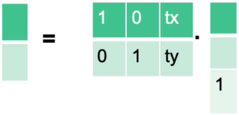
non-homog

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$$



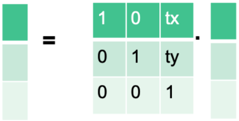
homog in, non-h out,

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}}$$



homog in, homog out

$$\bar{\boldsymbol{x}}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \\ 0^T & 1 \end{bmatrix} \bar{\boldsymbol{x}}$$



Camera matrix I

With the help of homogeneous coordinates, the relationships of a pinhole camera model can be expressed in a *convenient* matrix form:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{27}$$

where λ is equal to the depth Z.
Remember that we don't know the depth Z from the observation in the place image (depth ambiguity).

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathtt{P}\mathbf{X}$$

Camera matrix II

- $\mathtt{P}$ is a *camera (projection) matrix*
- For convenience, $\mathtt{P}$ is decomposed as follows:

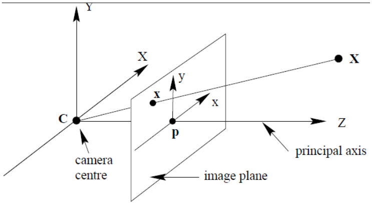
$$\begin{aligned} \mathtt{P} &= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] & (28) \\ &= \mathtt{K}\left[\mathtt{I}_{3\times 3} | \mathbf{0}_{3\times 1}\right] & (29) \end{aligned}$$

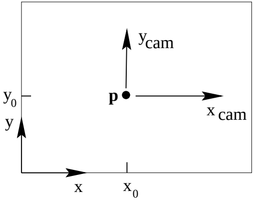
- $\mathtt{K} = \mathrm{diag}(f, f, 1)$ is the calibration matrix.

Generalization of the pinhole camera model I

(1) The central point might not be at (0, 0) in the image plane

⇒ two additional parameters in the calibration matrix





- If $(x_0, y_0)^T$ are the coordinates of the principal point, the projection becomes:

$$(X, \ Y, \ Z)^T \mapsto \left(f\frac{X}{Z} + x_0, \ f\frac{Y}{Z} + y_0 \right)^T \tag{30}$$

Generalization of the pinhole camera model II

- The matrix form is

$$\lambda \begin{bmatrix} x+x_0 \\ y+y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & x_0 & 0 \\ 0 & f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{31}$$

- The matrix $\mathtt{K}$ becomes

$$\mathtt{K} = \begin{bmatrix} f & & x_0 \\ & f & y_0 \\ & & 1 \end{bmatrix} \tag{32}$$

Generalization of the pinhole camera model III

(2) When light-elements on the sensor are not rectangular (they are trapezoidal)

$\rightarrow$ *skew* parameter s

(3) When pixels are not perfect squares

This modifies the aspect ratio $\rightarrow$ α_x and α_y.

Finally,

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{33}$$

Intrinsic parameters

Intrinsic parameters

In a refined camera model,

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \quad (34)$$

This matrix has 5 parameters. These parameters are called *intrinsic parameters*.

They characterize a camera and should be estimated for each camera separately. But once they are known, there is no need to estimated them again!

Calibration

Definition

A camera is said to be *calibrated* if

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

is known.

In general:

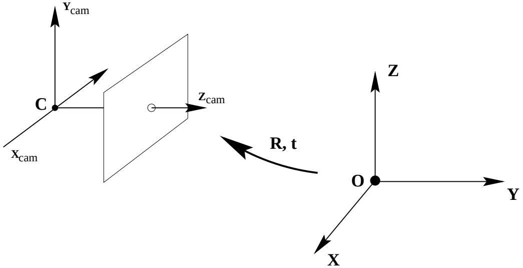
- $\alpha_x \simeq \alpha_y$
- $s \simeq 0$
- x_0 and y_0 close to 0 pixel (typically a few pixels maximum).

But we have a problem: **we don't know where the center of camera is located in the real world**. So there is no way to measure $(X, Y, Z, 1)^T$.

Towards *extrinsic* parameters I

Points in the 3D world need to be expressed in a system coordinate different from that of the camera (which is not known).

- Both coordinate systems are related by a rotation and a translation:



Towards *extrinsic* parameters II

For a translation:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} X_0 - t_1 \\ Y_0 - t_2 \\ Z_0 - t_3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -t_1 \\ 0 & 1 & 0 & -t_2 \\ 0 & 0 & 1 & -t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{35}$$

$$= \begin{bmatrix} \underline{\mathtt{I}} & -t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{36}$$

More generally (translation + rotation):

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \underline{\mathtt{R}}^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{\mathtt{I}} & -t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{37}$$

Towards *extrinsic* parameters III

In conclusion:

$$\mathbf{X_c} = \begin{bmatrix} \underline{\mathrm{R}}^T & -\underline{\mathrm{R}}^T t \\ 0 & 1 \end{bmatrix} \mathbf{X_0} \tag{38}$$

where **R** is a rotation matrix and t a translation vector. This is not a projection, but an Euclidean transform between coordinate systems.

Extrinsic parameters

There are 6 degrees of freedom/parameters (called *extrinsic* parameters):

- a translation vector (3 values)
- 3 rotation angles

Conclusions on $\mathtt{P}$ I

Starting from the pinhole camera model:

$$\lambda \mathbf{x} = \left[\begin{array}{ccc} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \mathbf{X_c} = \mathtt{K}\left[\mathtt{I}_{3\times 3} | \mathbf{0}_{3\times 1}\right] \mathbf{X_c} \qquad (39)$$

Link between the coordinate system of the camera and an arbitrary coordinate system:

$$\mathbf{X_c} = \left[\begin{array}{cc} \underline{\mathtt{R}}^T & -\underline{\mathtt{R}}^T t \\ 0 & 1 \end{array}\right] \mathbf{X} \qquad (40)$$

By combination:

$$\lambda \mathbf{x} = \mathtt{K}\mathtt{R}^T \left[\mathtt{I} | -t\right] \mathbf{X} = \mathtt{P}\mathbf{X} \qquad (41)$$

where $\mathtt{P} = \mathtt{K}\mathtt{R}^T \left[\mathtt{I} | -t\right]$.

Conclusions on $\mathtt{P}$ II

Definition

A camera represented with a camera matrix of the form $\mathtt{P} = \mathtt{K}\mathtt{R}^T\left[\mathtt{I}| - t\right]$ is called **normalized camera**.

$\mathtt{P}$ is a 3×4 matrix, which is subject to scale ambiguity. Therefore, $\mathtt{P}$ has 11 degrees of freedom (unknown parameters):

- 5 intrinsic parameters (*related to camera itself*; the manufacturer can computed them and give them to the user who buys the camera).
- 6 extrinsic parameters (*related to* the choice of an external *coordinate system*).

What if we:

- move the camera?
- zoom in?

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
   - Camera calibration: the pinhole model
   - Calibration algorithm
   - 3D reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Calibration algorithm I

How do we find the parameters of $\mathtt{P}$?

- We need correspondences between some 3D positions on their location on the projected 2D image $\mathbf{X}_i \leftrightarrow \mathbf{x}_i = (x_i, y_i, z_i)$.
- As a result

$$\mathbf{x}_i = \mathtt{P}\mathbf{X}_i \Rightarrow \mathbf{x}_i \times \mathtt{P}\mathbf{X}_i = 0$$

Let $\mathbf{P}^j$ be the vector with the jth line of $\mathtt{P}$,

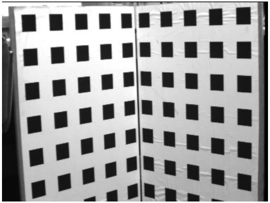
$$\begin{bmatrix} \mathbf{0}^T & -z_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ z_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

- Two of these equations are linearly independent. But how many correspondences do we need to solve this linear systems?

At least 6

Calibration algorithm II

- In practice, we have an over-determined system of linear equations $\mathtt{A}\mathbf{p} = \mathbf{0}$ if there is no noise. But, due to noise, we use an optimization procedure, such as the minimum square optimization
- Use of a precise and well-known 3D calibration pattern



Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
   - Camera calibration: the pinhole model
   - Calibration algorithm
   - 3D reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

3D reconstruction

Several difficulties of 3D reconstruction:

- detection of features. Usually, these are special points of objects to be detected (corners, centers, etc).
- 3D reconstruction of the scene might be
  - dense reconstruction (we have to find the depth for every pixels of the scene)
  - non-dense reconstruction
- additional difficulty: objects in the scene might be *moving*

Feature detection and motion are often related to each other!

Technologies for dealing with 3D information

Acquisition

- Single monochromatic/color camera
- Multiple cameras (stereoscopy, network of cameras)
- Depth (range) cameras

Rendering

- Glasses
  - Color anaglyph systems

  - Polarization systems
- Display
  - Auto-stereoscopic display technologies

3D reconstruction

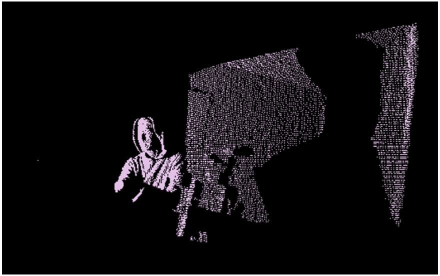
There are several ways to reconstruct the 3D geometry of a scene (typology of methods)

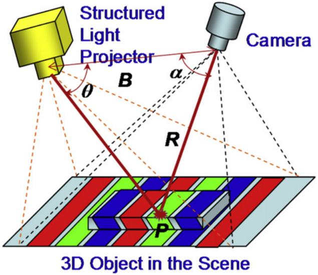
- Direct methods:
  - depth cameras (also named range or 3D cameras)
  - stereoscopic vision or multi-view camera systems
- Indirect methods (more or less out of fashion due the existence of 3D cameras):
  - “depth from motion”
  - “depth from focus”
  - “depth from shadow”
  - ...

Depth cameras I

There are two *acquisition* technologies for depth-cameras, also called range- or 3D-cameras:

[1] estimation of the deformations of a pattern sent on the scene (*structured light*).





Depth cameras II

First generation of the Kinects

Depth cameras III

[2] *measurements* by *time-of-flight* (ToF). Time to travel forth and back between the source led (camera) and the sensor (camera).

If d is the distance to a point in the scene and t is the time for the signal to travel from and back to the sensors, then

$$d = \frac{c}{2t}$$

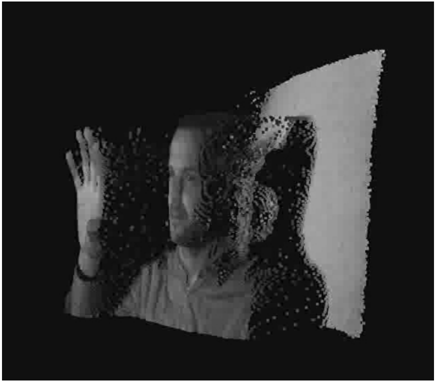
where $c = 3 \times 10^8\, m/s$ is the speed of light.

Mesa Imaging, Kinect (second generation), PMD cameras

Depth cameras IV

Illustration of a depth map acquired with a range camera

Two informations are provided by a range camera:
depth and *intensity*



Systems composed of multiple cameras

Three elements interact:

- calibration
- possible motion between the cameras (in the following, we only consider cameras with no relative motion)
- 3D structure of the scene

There are some theoretical results such as:

Theorem

Assume a series of images taken with an uncalibrated moving camera for which we have establish correspondences between pairs of points, then it is only possible to reconstruct the 3D scene up to a projective transform.

Steps to reconstruct a scene from multiple cameras (*stereoscopic* system)

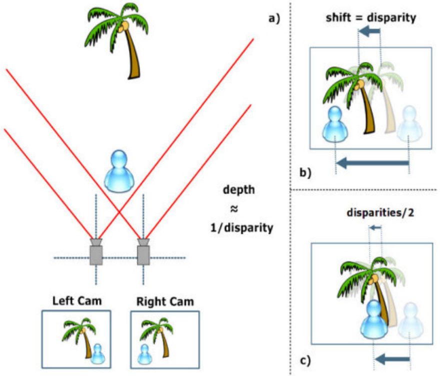
Generally, we have to find correspondences between points of the **views of two color cameras**.

There are several possibilities (note that we do not measure but infer information in these cases):

1. *cameras are aligned* (mechanically of numerically, after a transformation) → computation of the disparity map.
2. *cameras are not aligned* → corresponding points in the two camera planes are related to each other via the fundamental matrix. The fundamental matrix imposes a constraint; it is not sufficient to reconstruct a scene (calibration is needed, etc). This is studied under the name of epipolar geometry.
3. *all the points are in the same plane* → this constraint (knowledge) facilitates the correspondence finding between two views. The transformation is named an homography.

Stereoscopic cameras (aligned cameras)

If cameras are aligned, the disparity is the inverse of depth.



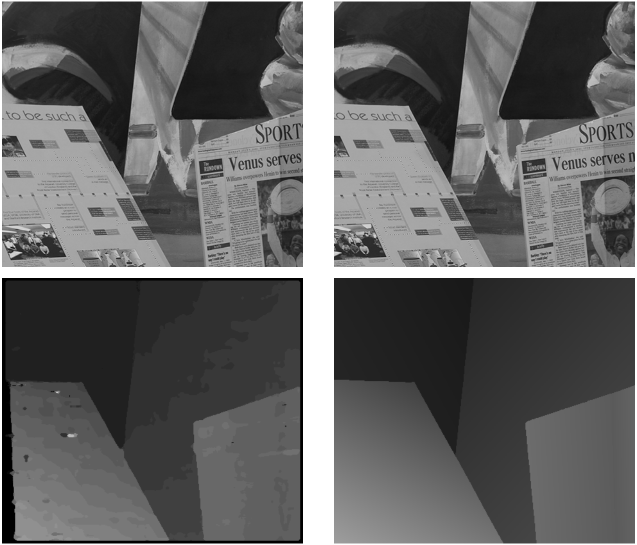
Computation of a disparity (depth) map

If two cameras only differ in the horizontal direction, the horizontal difference between two locations provides an information about the depth.

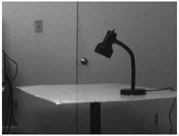
Attention:

- cameras and alignments are not perfect!
- disparity only estimates Z!

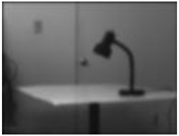
Disparity maps: illustration and difficulties I



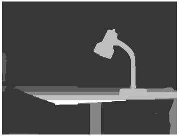
Disparity maps: illustration and difficulties II



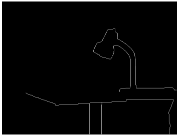
left view



right view



disparity



contours

The computation is disparity is difficult:

- close to borders (diffraction effects)
- in texture-less zones

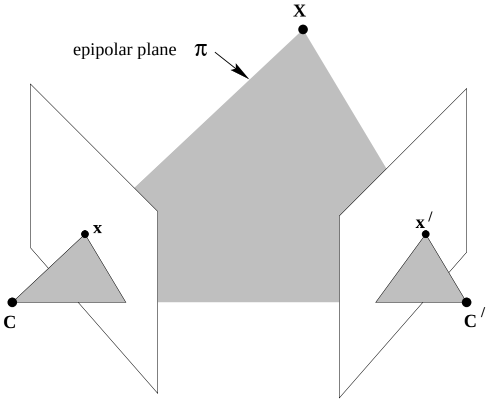
Discussion

Advantages (+) or drawbacks (-) of a reconstruction (≡ depth computation) by the technique of disparity maps:

- (+) the process is simple. One only has to estimate the distance between two corresponding points along a line.
- (-) cameras must be aligned,
    - either mechanically,
    - or by software. This usually involves re-sampling the image, leading to precision losses.

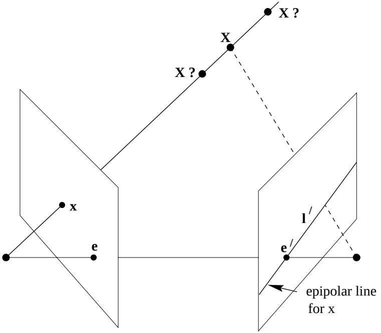
We need an alternative when the cameras are not aligned.

Unaligned cameras: epipolar geometry and fundamental matrix I



C and C′ are the center of the two cameras.

Unaligned cameras: epipolar geometry and fundamental matrix II



The use of a second camera is supposed to help determining the location of a point **X** along the projection line of the other camera.

Major challenges I

Objective: find a way to use the correspondences between points in a pair of views to be able to find (X, Y, Z) for every point in the scene.

- Calibration:
  - each camera has its own calibration matrix:
    - solution: find the intrinsic and extrinsic parameters
  - when there are two cameras
    - solution: there exists a way to build the relationship between the projections in the camera planes that is expressed by the *fundamental* matrix or the *essential* matrix.
- Camera placement, two options:
  - *fixed*: the manufacturer can calibrate the camera.
  - *changing or unknown*: one can isolate the intrinsic parameters, but this requires a calibration procedure.
- Object structures: they might lead to ambiguities (holes, shadows, etc).
  - "realistic" solution: do some prior assumptions about the objects. It is impossible to solve all the ambiguities, even with an infinite number of cameras.

Towards the *fundamental* matrix

Consider a point $\mathbf{X}$ in space. It is viewed by the two cameras as:

$$\begin{aligned} \lambda_1 \mathbf{x}_1 &= \mathtt{P}_1 \mathbf{X} = [\mathtt{A}_1 |\, b_1] \mathbf{X} \quad (42) \\ \lambda_2 \mathbf{x}_2 &= \mathtt{P}_2 \mathbf{X} = [\mathtt{A}_2 |\, b_2] \mathbf{X} \quad (43) \end{aligned}$$

We find X, Y, Z with the help of the equations of the first camera:

$$\lambda_1 \mathbf{x}_1 = \mathtt{P}_1 \mathbf{X} = [\mathtt{A}_1 |\, b_1] \mathbf{X} = \mathtt{A}_1 \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + b_1 \quad (44)$$

therefore,

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathtt{A}_1^{-1} (\lambda_1 \mathbf{x}_1 - b_1) \quad (45)$$

By substitution, in the second equation, we get

$$\lambda_2 \mathbf{x}_2 = \mathtt{A}_2 \mathtt{A}_1^{-1} (\lambda_1 \mathbf{x}_1 - b_1) + b_2 = \lambda_1 \mathtt{A}_{12} \mathbf{x}_1 + (-\mathtt{A}_{12} b_1 + b_2) \quad (46)$$

Consequently, $\mathbf{x}_1$ and $\mathbf{x}_2$ are *linearly independent*.

Fundamental matrix

Epipolar constraint

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T \mathtt{F} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \tag{47}$$

- The **fundamental matrix** $\mathtt{F}$ is a 3×3 matrix that constraints the values of the coordinates in the image planes of two cameras.
- $\mathbf{x}'\mathtt{F}\mathbf{x} = 0$ for all the corresponding pairs $\mathbf{x} \leftrightarrow \mathbf{x}'$.

Characteristics:

- 7 degrees of freedom (rank 2 matrix)
- the determinant is null
- it is defined up to a scaling factor

Link between the fundamental matrix and the calibration matrices of the cameras I

We can derive the fundamental matrix, starting from the two camera models. Assume that:

1. we are able to isolate the intrinsic and extrinsic parameters of both cameras.
2. the absolute 3D coordinate systems is placed on the internal coordinate of one camera. Then, we have

$$\mathtt{P} = \mathtt{K}\left[\mathtt{I}|0\right] \quad \text{and} \quad \mathtt{P}' = \mathtt{K}'\left[\mathtt{R}|t\right] \tag{48}$$

Link between the fundamental matrix and the calibration matrices of the cameras II

Then, it can be shown that the fundamental matrix is given by

$$\mathtt{F} = \mathtt{K}'^{-T}[t]_{\times}\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-T}\mathtt{R}[\mathtt{R}^{T}t]_{\times}\mathtt{K}^{-1} \tag{49}$$

where

$$[a]_{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{50}$$

If the intrinsic parameters are known, the coordinates can be normalized to remove the effect of the intrinsic parameters (according to transformation such as $\hat{u} = \mathtt{K}^{-1}u$ and $\hat{u'} = \mathtt{K}'^{-1}u'$), leading to

$$\mathtt{P} = [\mathtt{I}|0] \quad \text{and} \quad \mathtt{P}' = [\mathtt{R}|t] \tag{51}$$

Link between the fundamental matrix and the calibration matrices of the cameras III

The fundamental matrix simplifies, which results in the *essential* matrix.

$$\mathtt{E} = [t]_\times \mathtt{R} = \mathtt{R}[\mathtt{R}^T t]_\times \tag{52}$$

The major advantage of this matrix is that only the extrinsic parameters play a role (the essential matrix has 5 degrees of freedom).

Computation of the fundamental matrix: similar ideas to that of the calibration matrix

- We have to find a number of pairwise correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ (minimum of 4)
- We build a system of linear equations by means of $\mathbf{x}'_i \mathtt{F} \mathbf{x}_i = 0$
- The resolution of the system leads to $\mathtt{F}$

There exist several algorithms to proceed to the determination of $\mathtt{F}$.

Remember that, if the fundamental matrix links the points between two planes, it does not suffice to reconstruct a scene.

Reconstruction of a 3D: steps I

1. Find correspondences between pairs of points in the two views
2. Calculate the fundamental/essential matrix
3. Find the camera parameters, with the help of the essential matrix:

$$\mathtt{P} = [\mathtt{I} | \mathbf{O}] \qquad \mathtt{P}' = \left[[\mathbf{e}']_\times \mathtt{F} | \mathbf{e}' \right]$$

where

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

and

$$\mathtt{F}^T \mathbf{e}' = \mathbf{0}$$

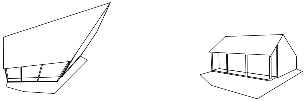
Reconstruction of a 3D: steps II

4. For each pixel $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, triangulate $\mathbf{X}_i$:

$$\mathtt{P}\mathbf{X}_i \times \mathbf{x}_i = 0 \quad \text{et } \mathtt{P}'\mathbf{X}_i \times \mathbf{x}'_i = 0$$

We have 4 equations for 3 unknowns.

Attention! Remember that this reconstruction is valid up to a projective transform.



Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
   - Morphology on sets/objects
   - Geodesic operators
   - Grayscale morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Mathematical morphology

- Reminders of the set theory
- Basic morphological transforms
- Neighboring transformations
- Geodesy and reconstruction
- Grayscale morphology

Reminders of the set theory I

Sets will be denoted with capital letters, such as A, B, ..., and elements of these sets by lowercase letters a, b, ...

- **Set equality**
  Two sets are *equal* if they contain the same elements:
  $X = Y \Leftrightarrow (x \in X \Rightarrow x \in Y \text{ and } x \in Y \Rightarrow x \in X)$. The empty set is denoted as $\emptyset$.
- **Inclusion**
  X is a *subset* of Y (that is, X is included in Y) if all the elements of X also belong to Y: $X \subseteq Y \Leftrightarrow (x \in X \Rightarrow x \in Y)$.
- **Intersection**
  The *intersection* between X and Y is the set composed of the elements that belong to both sets:
  $X \cap Y = \{x \text{ such that } x \in X \text{ and } x \in Y\}$.

Reminders of the set theory II

- **Union**

  The *union* between two sets is the set that gathers all the elements that belong to at least one set:
  $X \cup Y = \{x \text{ such that } x \in X \text{ or } x \in Y\}$.

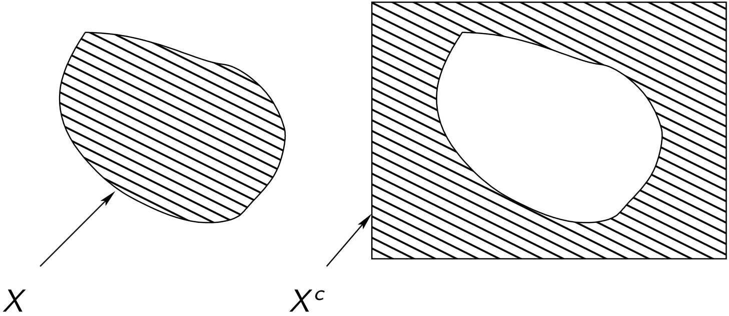
- **Difference**

  The *set difference* between X and Y, denoted by $X - Y$ or $X \backslash Y$ is the set that contains the elements of X that are not in Y :
  $X - Y = \{x | x \in X \, and \;\, x \notin Y\}$.

Reminders of the set theory III

- **Complementary**

  Assume that X is a subset of a $\mathcal{E}$ space, the *complementary set* of X with respect to $\mathcal{E}$ is the set, denoted X^c, given by
  $X^c = \{x \text{ such that } x \in \mathcal{E} \text{ and } x \notin X\}$.

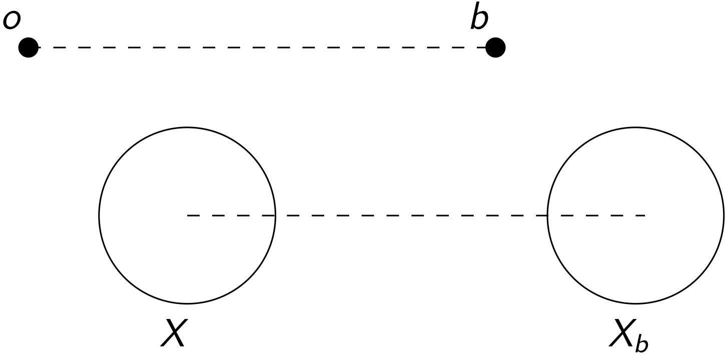


- **Symmetric**

  The *symmetric set*, $\check{X}$, of X is defined as $\check{X} = \{-x | x \in X\}$.

Reminders of the set theory IV

- **Translated set**
  The *translate* of X by b is given by $\{z \in \mathcal{E} | z = x + b, \, x \in X\}$.



Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
   - Morphology on sets/objects
   - Geodesic operators
   - Grayscale morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Basic morphological operators I

Erosion

Definition (*Morphological erosion*)

$$X \ominus B = \{z \in \mathcal{E} | B_z \subseteq X\}. \tag{53}$$

The following algebraic expression is equivalent to the previous definition:

Definition (*Alternative definition for the morphological erosion*)

$$X \ominus B = \bigcap_{b \in B} X_{-b}. \tag{54}$$

B is named "**structuring element**".

Basic morphological operators II

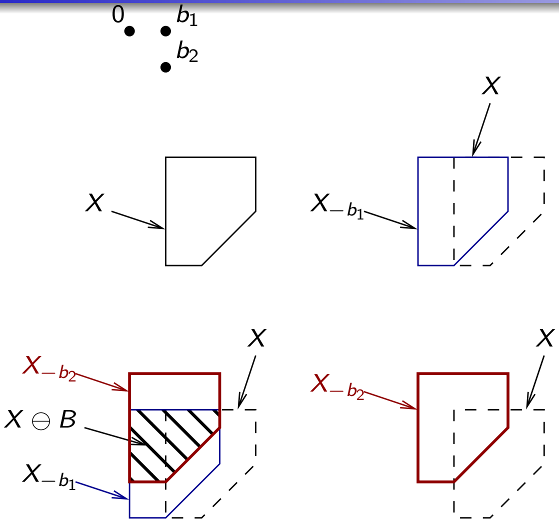


Figure: Algebraic interpretation of the erosion.

Erosion with a disk

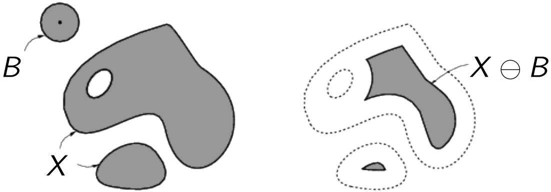


Figure: Erosion of X with a disk B. The origin of the structuring element is drawn at the center of the disk (with a black dot).

Dilation I

Definition (Dilation)

From an algebraic perspective, the *dilation* (*dila***ta***tion* in French!), is the union of translated version of X:

$$X \oplus B = \bigcup_{b \in B} X_b = \bigcup_{x \in X} B_x = \{x + b | x \in X,\ b \in B\}. \tag{55}$$

Dilation II

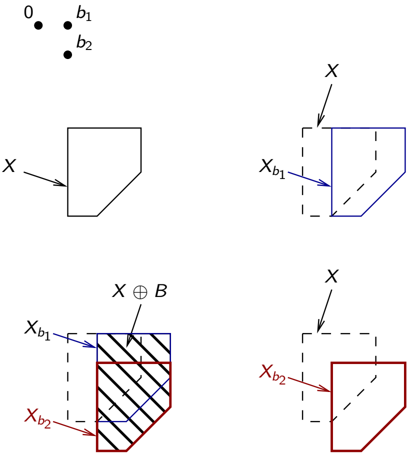


Figure: Illustration of the algebraic interpretation of the dilation operator.

Dilation III

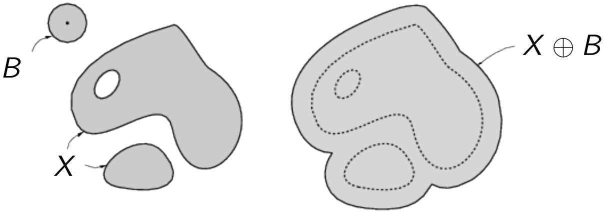


Figure: Dilation of X with a disk B.

Properties of the erosion and the dilation

Duality

Erosion and dilation are dual operators with respect to complementation*:*

$$X \ominus \check{B} = (X^c \oplus B)^c \tag{56}$$

$$X \ominus B = (X^c \oplus \check{B})^c \tag{57}$$

Some properties

Erosion and dilation obey the principles of “ideal” morphological operators:

1. erosion and dilation are invariant to translations: $X_z \ominus B = (X \ominus B)_z$. Likewise, $X_z \oplus B = (X \oplus B)_z$;
2. erosion and dilation are compatible with scaling: $\lambda X \ominus \lambda B = \lambda(X \ominus B)$ and $\lambda X \oplus \lambda B = \lambda(X \oplus B)$;
3. erosion and dilation are local operators (if B is bounded);
4. it can be shown that erosion and dilation are continuous transforms.

Algebraic properties

1. erosion and dilation are *increasing* operators: if $X \subseteq Y$, then $(X \ominus B) \subseteq (Y \ominus B)$ and $(X \oplus B) \subseteq (Y \oplus B)$;
2. if the structuring element contains the origin, then the erosion is *anti-extensive* and the dilation is *extensive*, that is $X \ominus B \subseteq X$ and $X \subseteq X \oplus B$.

Morphological opening I

Definition (Opening)

The *opening* results from cascading an erosion and a dilation with the same structuring element:

$$X \circ B = (X \ominus B) \oplus B \tag{58}$$

Interpretation of openings (alternative definition)

The interpretation of the opening operator (which can be seen as an alternative definition) is based on

$$X \circ B = \bigcup \{B_z | z \in \mathcal{E} \,\text{and}\, B_z \subseteq X\} \tag{59}$$

In other words, the opening of a set by structuring element B is the set of all the elements of X that are covered by a translated copy of B when it moves inside of X.

Morphological closing

Definition (Closing)

A *closing* is obtained by cascading a dilation and an erosion with a unique structuring element:

$$X \bullet B = (X \oplus B) \ominus B \tag{60}$$

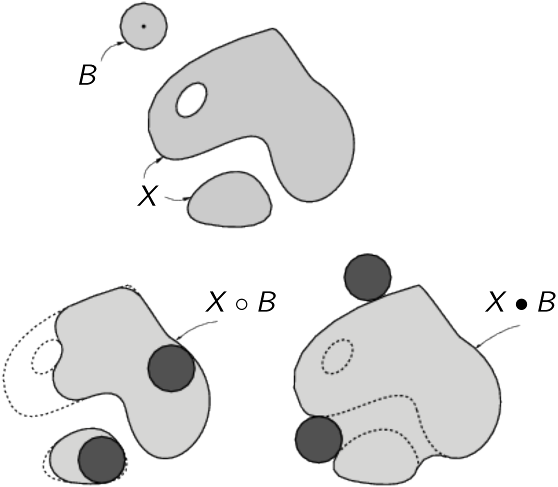
Opening and closing are dual operators with respect to set complementation: indeed,

$$(X \circ B)^c = X^c \bullet \check{B} \tag{61}$$

and

$$(X \bullet B)^c = X^c \circ \check{B} \tag{62}$$

Opening and closing of X with a disk B



Opening and closing properties

By construction, the opening and closing follow the “ideal” principles of morphological operators.

The most important algebraic properties of $X \circ B$ and $X \bullet B$ are

1. opening and closing are *increasing*. If $X \subseteq Y$, then
$$(X \circ B) \subseteq (Y \circ B) \text{ and } (X \bullet B) \subseteq (Y \bullet B) \tag{63}$$
2. opening is *anti-extensive*, and closing is *extensive* (no condition related on the origin here!)
$$X \circ B \subseteq X, \quad X \subseteq X \bullet B \tag{64}$$
3. opening and closing are *idempotent* operators (projective operators). This means that
$$(A \circ B) \circ B = A \circ B \text{ and } (A \bullet B) \bullet B = A \bullet B \tag{65}$$

General properties I

- Dilation is *commutative* and *associative*

$$X \oplus B = B \oplus X \tag{66}$$

$$(X \oplus Y) \oplus C = X \oplus (Y \oplus C) \tag{67}$$

- Dilation distributes the union

$$(\bigcup_j X_j) \oplus B = \bigcup_j (X_j \oplus B) \tag{68}$$

- The erosion distributes the intersection

$$(\bigcap_j X_j) \ominus B = \bigcap_j (X_j \ominus B) \tag{69}$$

- *Chain rule* ($\equiv$ *cascading rule*):

$$X \ominus (B \oplus C) = (X \ominus B) \ominus C \tag{70}$$

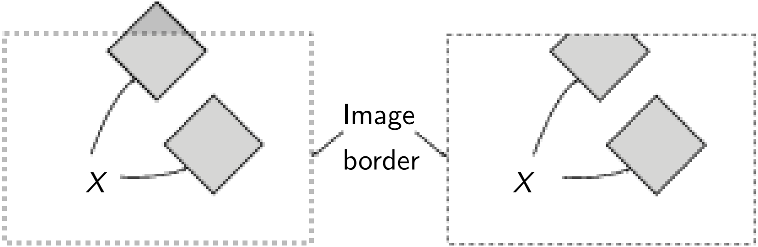
General properties II

- The opening and closing are not related to the exact location of the origin (so they do not depend on the location of the origin when defining B). Let $z \in \mathcal{E}$

$$X \circ B_z = X \circ B \tag{71}$$

$$X \bullet B_z = X \bullet B \tag{72}$$

A practical problem: dealing with borders I

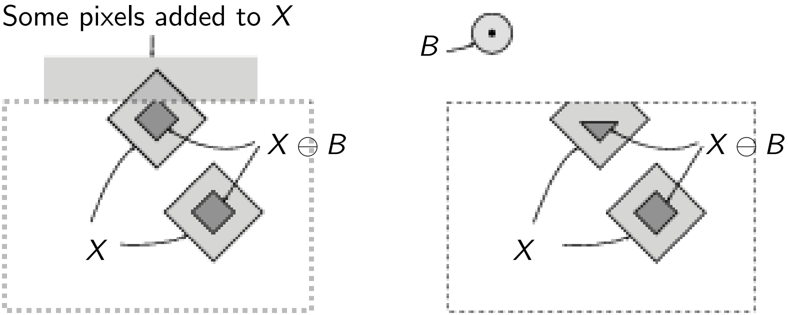


Physical assumption 1

Physical assumption 2

Figure: Two possible physical assumptions for borders.

A practical problem: dealing with borders II



Physical assumption 1

Physical assumption 2

Figure: Comparison of the effects of two physical assumptions on the computation of the erosion of X.

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology
- Morphology on sets/objects
- Geodesic operators
- Grayscale morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

Neighboring transforms

Definition

The *Hit or Miss transform* $X \Uparrow (B, C)$ *is defined as*

$$X \Uparrow (B, C) = \{x | B_x \subseteq X,\ C_x \subseteq X^c\} \tag{73}$$

If $C = \emptyset$ the transform reduces to an erosion of X by B.

Geodesy and reconstruction I

Geodesic dilation

A geodesic dilation is always based on two sets (images).

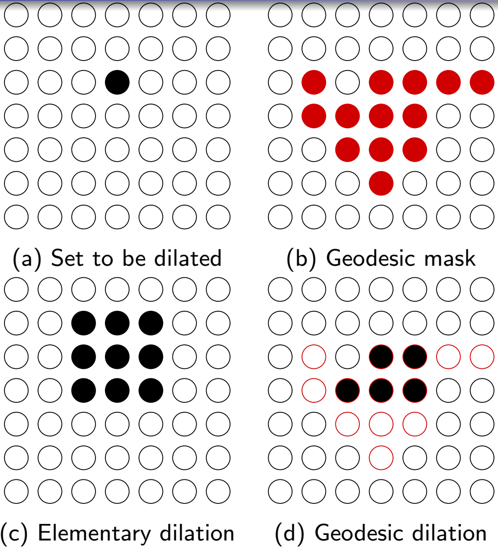
Definition

The *geodesic dilation of size* 1 of X conditionally to Y, denoted $D_Y^{(1)}(X)$, is defined as the intersection of the dilation of X and Y:

$$\forall X \subseteq Y,\ \ D_Y^{(1)}(X) = (X \oplus B) \cap Y \tag{74}$$

where B is usually chosen according to the frame connectivity (a 3×3 square for a 8-connected grid).

Geodesy and reconstruction II


(a) Set to be dilated

(b) Geodesic mask

(c) Elementary dilation

(d) Geodesic dilation

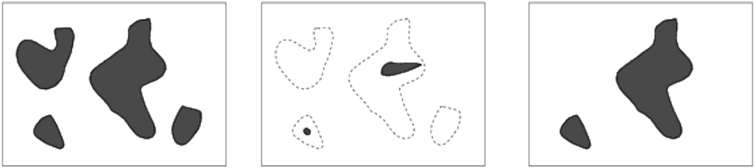
Figure: Geodesic dilation of size 1.

Morphological reconstruction

Definition

The *reconstruction* of X conditionally to Y is the geodesic dilation of X until idempotence. Let i be the iteration during which idempotence is reached, then the reconstruction of X is given by

$$R_Y(X) = D_Y^{(i)}(X) \text{ with } D_Y^{(i+1)}(X) = D_Y^{(i)}(X). \tag{76}$$



(a) Blobs (b) Marking blobs (c) Reconstructed blobs

Figure: Blob extraction by marking and reconstruction.

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
   - Morphology on sets/objects
   - Geodesic operators
   - Grayscale morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Grayscale morphology I

Notion of a function

Let $\mathcal{G}$ be the range of possible grayscale values. An image is represented by a function $f : \mathcal{E} \rightarrow \mathcal{G}$, which projects a location of a value of $\mathcal{G}$. In practice, an image is not defined over the entire space $\mathcal{E}$, but on a limited portion of it, a compact D.

We need to define an order between functions.

Definition (Partial ordering between functions)

Let f and g be functions. f is inferior to g,

$$f \leq g \quad \text{if} \quad f(x) \leq g(x), \forall x \in \mathcal{E} \tag{77}$$

Grayscale morphology II

Definition (Infimum and supremum)

Let f_i be a family of functions, $i \in I$. The *infimum* (respectively the *supremum*) of this family, denoted $\wedge_{i \in I} f_i$ (resp. $\vee_{i \in I} f_i$) is the largest lower bound (resp. the lowest upper bound).

In the practical case of a finite family I, the supremum and the infimum correspond to the maximum and the minimum respectively. In that case,

$$\forall x \in \mathcal{E}, \begin{cases} (f \vee g)(x) = \max(f(x), g(x)) \\ (f \wedge g)(x) = \min(f(x), g(x)) \end{cases} \tag{78}$$

Definition (Translate of a function)

The translate of *a function f* by b, denoted by f_b, is defined as

$$\forall x \in \mathcal{E}, \quad f_b(x) = f(x - b). \tag{79}$$

Additional definitions related to operators I

Definition (Idempotence)

An operator ψ is *idempotent* if, for each function, a further application of it does not change the final result. That is, if

$$\forall f,\ \psi(\psi(f)) = \psi(f) \tag{80}$$

Definition (Extensivity)

An operator is *extensive* if the result of applying the operator is larger that the original function

$$\forall f,\ f \leq \psi(f) \tag{81}$$

Additional definitions related to operators II

Definition (Anti-extensivity)

An operator is *anti-extensive* if the result of applying the operator is lower that the original function

$$\forall f,\ f \geq \psi(f) \tag{82}$$

Definition (Increasingness)

An increasing operator is such that it does not modify the ordering between functions:

$$\forall f, g,\ f \leq g \Rightarrow \psi(f) \leq \psi(g) \tag{83}$$

By extension, an operator ψ_1 is lower that an operator ψ_2 if, for every function f, $\psi_1(f)$ is lower to $\psi_2(f)$:

$$\psi_1 \leq \psi_2 \Leftrightarrow \forall f,\ \psi_1(f) \leq \psi_2(f) \tag{84}$$

Erosion and dilation

Definition (Grayscale dilation and erosion)

Let B be the domain of definition of a structuring element. The *grayscale dilation* and *erosion* (with a flat structuring element) are defined, respectively as,

$$f \oplus B = \bigvee_{b \in B} f_b(x) \tag{85}$$

$$f \ominus B = \bigwedge_{b \in B} f_{-b}(x) \tag{86}$$

Numerical example ($B = \{-1, 0, 1\}$)

x	0	1	2	3	4	5	6	7	8	9	10
$f(x)$	25	27	30	24	17	15	22	23	25	18	20
$f(x-1)$		25	27	30	24	17	15	22	23	25	18
$f(x)$	25	27	30	24	17	15	22	23	25	18	20
$f(x+1)$	27	30	24	17	15	22	23	25	18	20	
$f \ominus B(x) = \min$		25	24	17	15	15	15	22	18	18	
$f \oplus B(x) = \max$		30	30	30	24	22	23	25	25	25	

Typical questions:

- best algorithms? (note that there is some redundancy between neighboring pixels)
- how do we handle borders?

Algorithms

- Based on the decomposition of the structuring element:
  - $f \ominus (H \oplus V) = (f \ominus H) \ominus V$
  - $f \ominus (B \oplus B) = (f \ominus B) \ominus \partial(B)$
- Appropriate structure for storing and propagating the local min and max
  - queues
  - histogram

Illustration I

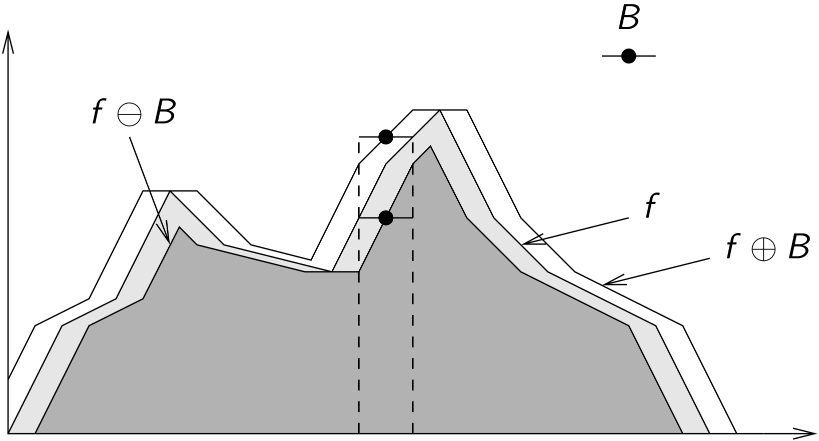


Figure: Erosion and dilation of a function.

Illustration II

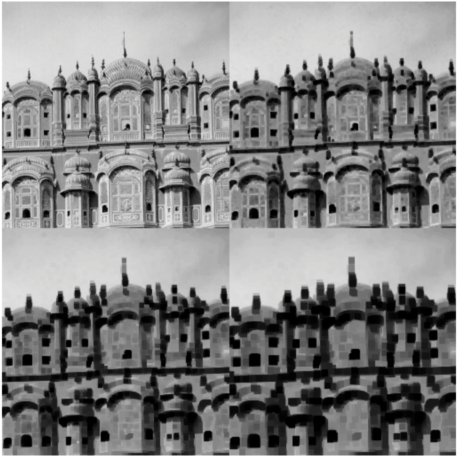


Figure: Erosions with squares of increasing sizes.

Illustration III

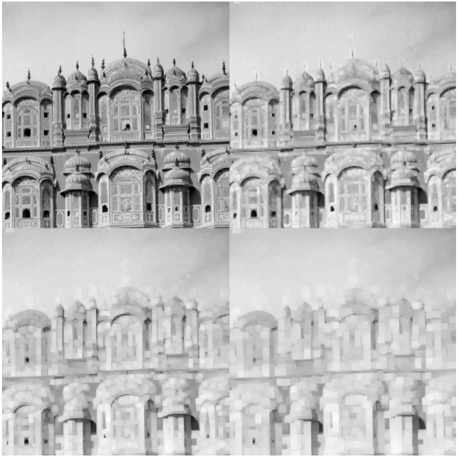


Figure: Dilations with squares of increasing sizes.

Morphological opening and closing I

The opening $f \circ B$ is obtained by cascading an erosion followed by a dilation. The closing $f \bullet B$ is the result of a dilation followed by an erosion.

Definition (Morphological opening and closing)

$$\begin{aligned} f \circ B &= (f \ominus B) \oplus B && (87) \\ f \bullet B &= (f \oplus B) \ominus B && (88) \end{aligned}$$

Morphological opening and closing II

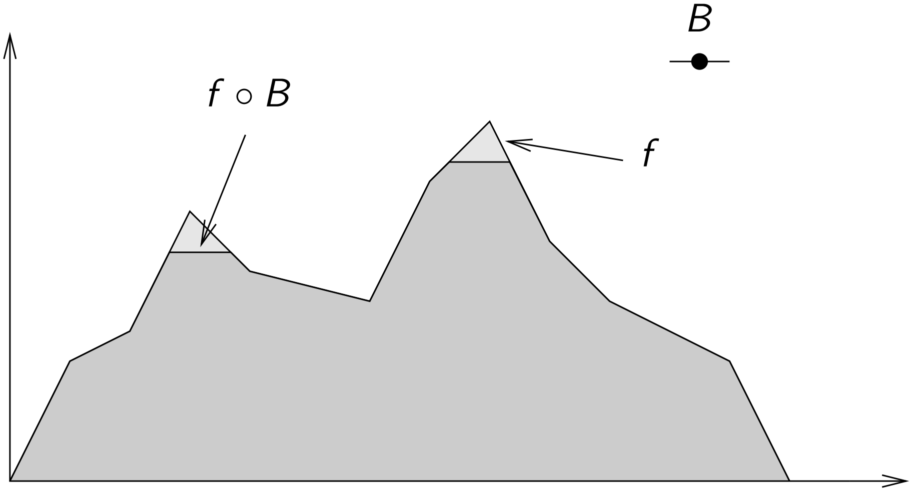


Figure: Opening of a function.

Properties of grayscale morphological operators

Erosion and dilation are increasing operators

$$f \leq g \Rightarrow \begin{cases} f \ominus B \leq g \ominus B \\ f \oplus B \leq g \oplus B \end{cases} \tag{89}$$

Erosion distributes the infimum and dilation distributes the supremum

$$(f \wedge g) \ominus B = (f \ominus B) \wedge (g \ominus B) \tag{90}$$
$$(f \vee g) \oplus B = (f \oplus B) \vee (g \oplus B) \tag{91}$$

Opening and closing are idempotent operators

$$(f \circ B) \circ B = f \circ B \tag{92}$$
$$(f \bullet B) \bullet B = f \bullet B \tag{93}$$

Opening and closing are anti-extensive and extensive operators respectively

$$f \circ B \leq f \tag{94}$$
$$f \leq f \bullet B \tag{95}$$

Reconstruction of grayscale images I

Definition

The *reconstruction* of f, conditionally to g, is the geodesic dilation of f until idempotence is reached. Let i, be the index at which idempotence is reached, the reconstruction of f is then defined as

$$R_g(f) = D_g^{(i)}(f) \text{ with } D_g^{(i+1)}(f) = D_g^{(i)}(f). \tag{96}$$

Reconstruction of grayscale images II

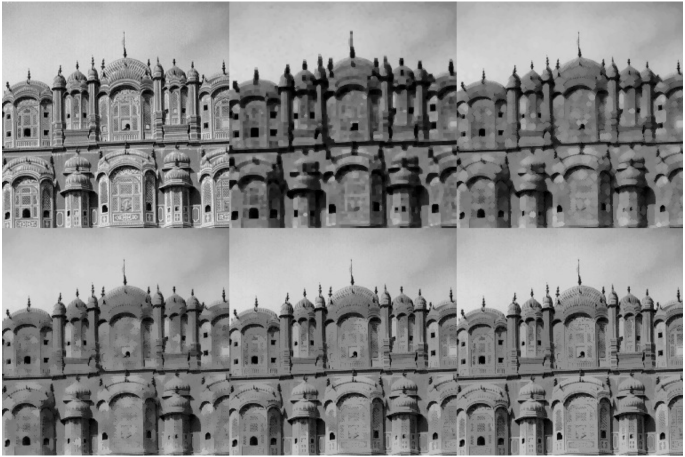


Figure: Original image, eroded image, and several successive geodesic dilations.

Reconstruction of grayscale images III

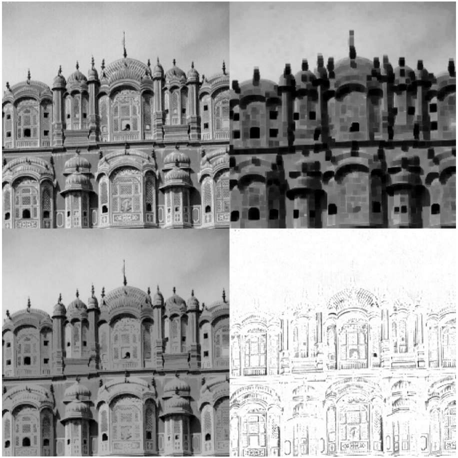


Figure: Original image, eroded image, reconstructed image starting from the eroded image, and difference image (reverse video).

Reconstruction of grayscale images IV

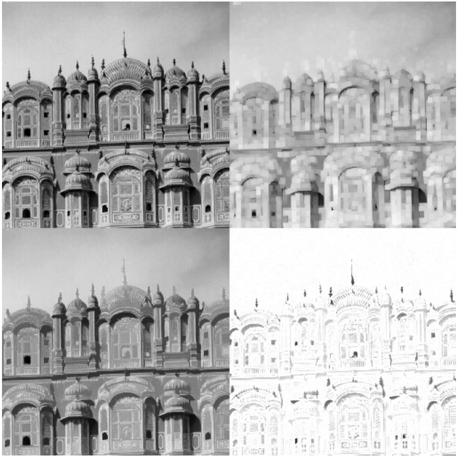


Figure: Original image, dilated image, reconstructed image starting from the dilated image (dual reconstruction), and difference image (reverse video).

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. **Non-linear filtering**
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Non-linear filtering

- *Rank* filters
  - Median
- *Morphological* filters
  - Algebraic definition
  - How to build a filter?
  - Examples of filters
    - Alternate sequential filters
    - Morphological filter

Introduction to rank filters

$f(x)$	25	27	30	24	17	15	22	23	25	18	20
$f(x-1)$	?	25	27	30	24	17	15	22	23	25	18
$f(x)$	25	27	30	24	17	15	22	23	25	18	20
$f(x+1)$	27	30	24	17	15	22	23	25	18	20	?
$f \ominus B(x) = \min$		25	24	17	15	15	15	22	18	18	
$f \oplus B(x) = \max$		30	30	30	24	22	23	25	25	25	

We could also order the values:

$f(x)$	25	27	30	24	17	15	22	23	25	18	20
1 (smallest)		25	24	17	15	15	15	22	18	18	18
2	25	27	27	24	17	17	22	23	23	20	20
3 (largest)	27	30	30	30	24	22	23	25	25	25	
$f \ominus B(x) = \min$		25	24	17	15	15	15	22	18	18	
$f \oplus B(x) = \max$		30	30	30	24	22	23	25	25	25	

Definition of rank filters

Let $k \in \mathbb{N}$ be a threshold.

Definition (*Rank filter*)

The operator or k-order rank filter, denoted as $\rho_{B,k}(f)(x)$, defined with respect to the B structuring element (which serves as a neighborhood), is

$$\rho_{B,k}(f)(x) = \bigvee \{t \in \mathcal{G} | \sum_{b \in B} [f(x+b) \geq t] \geq k\} \tag{97}$$

The simplest interpretation is that $\rho_{B,k}(f)(x)$ is the k-est value when all the $f(x+b)$ values are ranked in decreasing (increasing) order.
Rank filters are ordered. Let $\sharp(B)$, be the surface of B, then

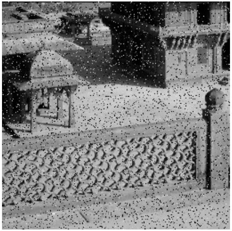
$$\rho_{B,\sharp(B)}(f)(x) \leq \rho_{B,\sharp(B)-1}(f)(x) \leq \ldots \leq \rho_{B,1}(f)(x) \tag{98}$$

Median filter I

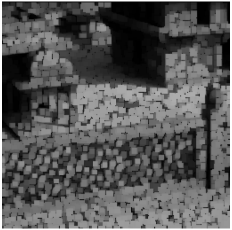
If n is odd, the $k = \frac{1}{2}(\sharp(B) + 1)$ choice leads to the definition of a *self-dual* operator, that is a filter that produces the same result as if applied on the dual function. This operator, denoted med_B, is the *median filter*.

$f(x)$	25	27	30	24	17	15	22	23	25	18	20
1		25	24	17	15	15	15	22	18	18	18
med_B	25	27	27	24	17	17	22	23	23	20	20
3	27	30	30	30	24	22	23	25	25	25	
$f \ominus B(x) = \min$		25	24	17	15	15	15	22	18	18	
$f \oplus B(x) = \max$		30	30	30	24	22	23	25	25	25	

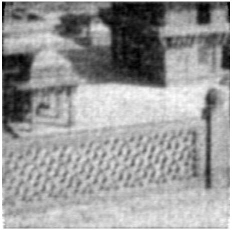
Median filter II



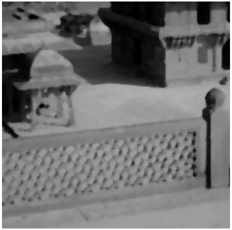
(a) Original image f + noise



(b) Opening with a 5×5 square

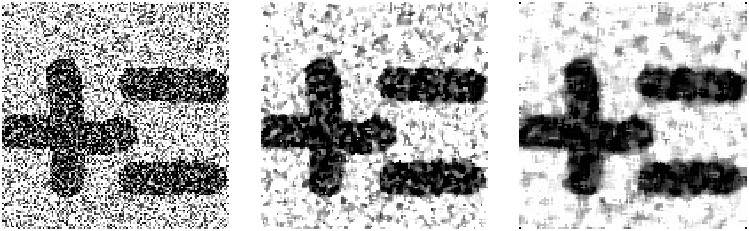


(c) Low-pass Butterworth ($f_c = 50$)



(d) Median with a 5×5 square

Effect of the size of the median filter



(a) Image f (b) 3×3 median (c) 5×5 median

Notes about the implementation

The median filter is *not idempotent*. Successive applications can result in *oscillations* (theoretically if the domain of the function is infinite)

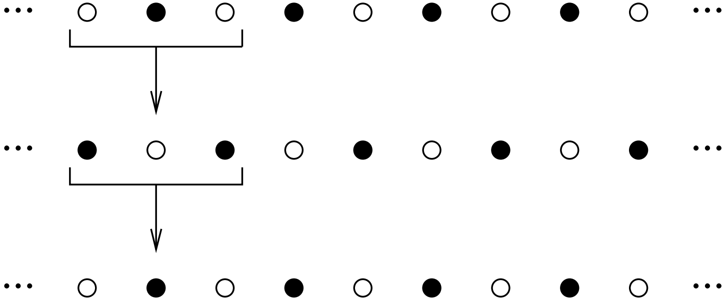


Figure: Repeated application of a median filter.

Also,

$$\mathrm{med}_{5\times5}(f) \neq \mathrm{med}_{1\times5}(\mathrm{med}_{5\times1}(f)) \tag{99}$$

but it is an acceptable *approximation*!

Morphological filters

Definition (Algebraic filter)

By definition, a filter is an *algebraic filter* if and only if the operator is (1) *increasing* and (2) *idempotent*:

$$\psi \text{ is an algebraic filter } \Leftrightarrow \forall f, g \begin{cases} f \leq g \Rightarrow \psi(f) \leq \psi(g) \\ \psi(\psi(f)) = \psi(f) \end{cases} \tag{100}$$

Definition (Algebraic opening)

An *algebraic opening* is an operator that is *increasing*, *idempotent*, **and** (3) *anti-extensive*. Formally,

$$\forall f, g,\ f \leq g \Rightarrow \psi(f) \leq \psi(g) \tag{101}$$

$$\forall f,\ \psi(\psi(f)) = \psi(f) \tag{102}$$

$$\forall f,\ \psi(f) \leq f \tag{103}$$

An *algebraic closing* is defined similarly, except that the operator is *extensive*.

How to build a filter? I

By combining know filters!?

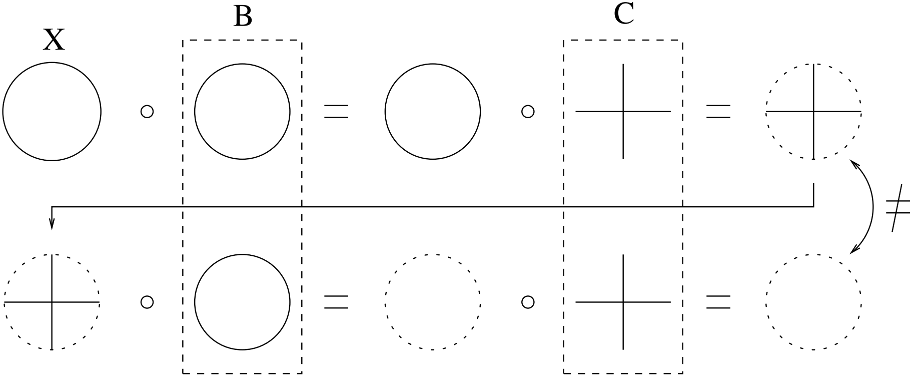


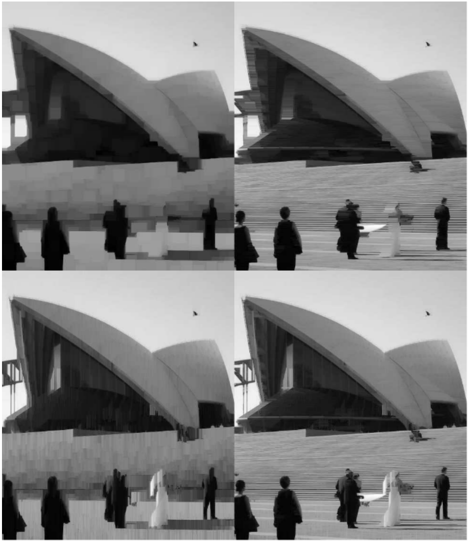
Figure: Unfortunately, the composition (cascading) of two openings is not an opening.

How to build a filter? II

New filters can be built starting from openings, denoted γ_i, and closings, denoted ϕ_i. The rules to follow are:

1. the supremum of openings is an opening: $(\bigvee_i \gamma_i)$ is an opening;
2. the infimum of closings is a closing: $(\bigwedge_i \phi_i)$ is a closing.

Supremum of two openings



Openings: $\gamma_{mH\oplus nV}(f)$, $\gamma_{mH}(f)$, $\gamma_{nV}(f)$, and $\gamma_{mH}(f) \vee \gamma_{nV}(f)$

Composition rules: structural theorem

Let ψ_1 and ψ_2 be two filters such that $\psi_1 \geq I \geq \psi_2$ (for example, ψ_1 is a closing and ψ_2 an opening).

Theorem (*Structural theorem*)

Let ψ_1 and ψ_2 be two filters such that $\psi_1 \geq I \geq \psi_2$, then

$$\psi_1 \geq \psi_1\psi_2\psi_1 \geq (\psi_2\psi_1 \vee \psi_1\psi_2) \geq (\psi_2\psi_1 \wedge \psi_1\psi_2) \geq \psi_2\psi_1\psi_2 \geq \psi_2 \quad (104)$$

$$\psi_1\psi_2,\ \psi_2\psi_1,\ \psi_1\psi_2\psi_1,\ \psi_2\psi_1\psi_2 \ \textit{are all filers} \quad (105)$$

Note that there is no ordering between $\psi_1\psi_2$ and $\psi_2\psi_1$.

Examples of filters I

Alternate Sequential Filters (ASF)

Let γ_i (ϕ_i) be an opening (resp. a closing) of size i and I be the identity operator (i.e. $I(f) = f$).

We assume that there is the following order:

$$\forall i, j \in \mathbb{N}, \;\; i \leq j, \quad \gamma_j \leq \gamma_i \leq I \leq \phi_i \leq \phi_j, \tag{106}$$

For each index i, we define these operators:

$$\begin{aligned} m_i &= \gamma_i \phi_i, & r_i &= \phi_i \gamma_i \phi_i, \\ n_i &= \phi_i \gamma_i, & s_i &= \gamma_i \phi_i \gamma_i. \end{aligned}$$

Examples of filters II

Definition (Alternate Sequential Filters (ASF))

For each index $i \in \mathbb{N}$, the following operators are the alternate sequential filters of index i

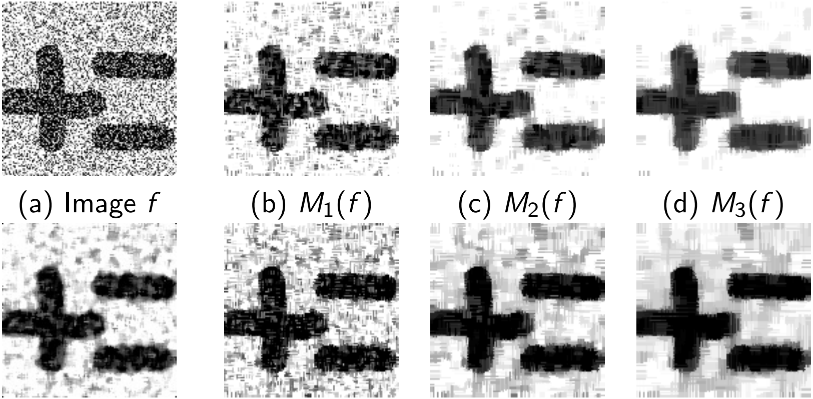
$$M_i = m_i m_{i-1} \dots m_2 m_1 \qquad R_i = r_i r_{i-1} \dots r_2 r_1 \tag{107}$$

$$N_i = n_i n_{i-1} \dots n_2 n_1 \qquad S_i = s_i s_{i-1} \dots s_2 s_1 \tag{108}$$

Theorem (*Absorption law*)

$$i \leq j \Rightarrow M_j M_i = M_j \;\; but \;\; M_i M_j \leq M_j \tag{109}$$

Examples of filters III



(a) Image f (b) $M_1(f)$ (c) $M_2(f)$ (d) $M_3(f)$

(e) 5×5 median (f) $N_1(f)$ (g) $N_2(f)$ (h) $N_3(f)$

Figure: Use of alternate sequential filters to remove some noise.

Toggle mappings I

The *morphological center* is a typical example of *toggle mapping*.

Definition (Morphological center)

Let ψ_i be a family of operators. The *morphological center* β of a function f with respect to the ψ_i family is defined, for each location x of the domain of f as follows:

$$\beta(f)(x) = (f(x) \vee (\bigwedge_i \psi_i(x))) \wedge (\bigvee_i \psi_i(x)) \tag{110}$$

Toggle mappings II

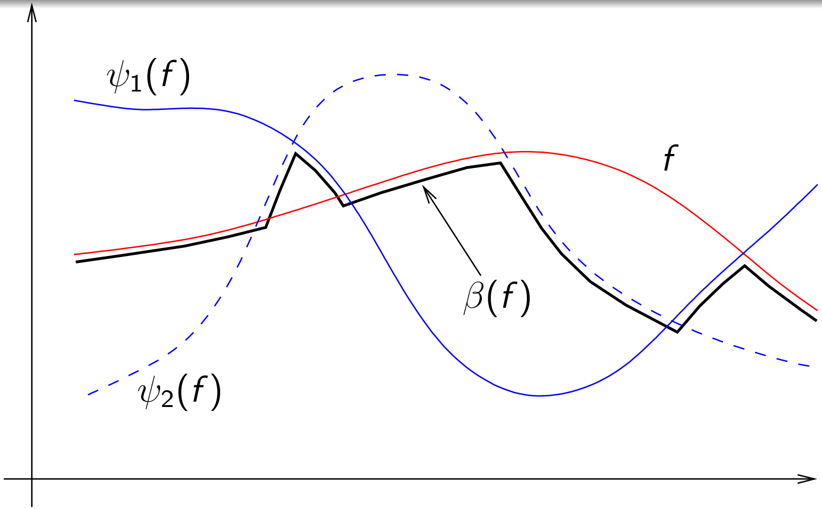


Figure: Morphological center of a one-dimensional signal.

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
   - General considerations
   - Texture analysis
   - Shape description
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

- General considerations
- Texture analysis
- Shape description

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

General considerations

Objects have:

- a texture (inside)
- a shape (border)

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
   - General considerations
   - Texture analysis
   - Shape description
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Texture analysis: outline

- Definition?
- *Statistical characterization* of textures
  - Local mean
  - Local standard deviation
  - Local histogram
  - Co-occurrence matrix of a grayscale image
- *Geometrical characterization* of textures
  - Spectral approach
  - Texture and energy

Goals of texture-related applications

The major question related to texture are:

- *texture analysis*. The purpose is to characterize a texture by a set of parameters called "texture descriptors".
- *texture recognition*.
- *image segmentation*.

Definition

Definition (Tentative definition)

A texture is a signal than can be *extended naturally* outside of its domain.

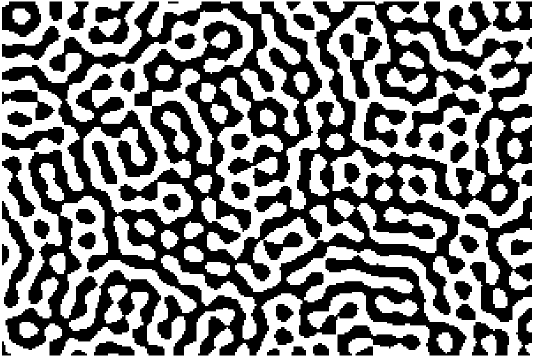
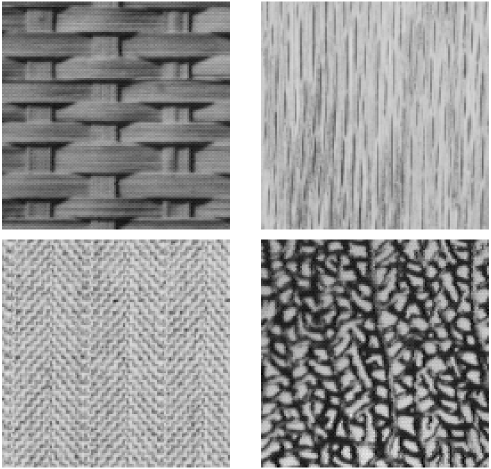


Figure: One possible texture (according to Lantuéjoul).

Examples of “real” (grayscale) textures



Simple analysis of a grayscale image

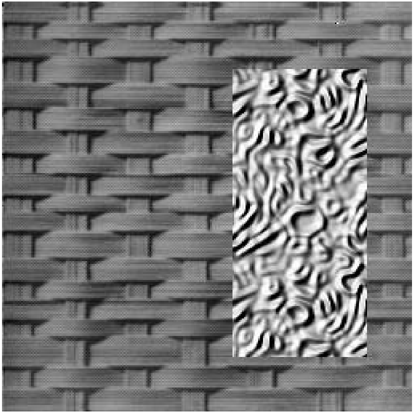


Figure: Example of an image with two textures.

Statistical descriptors of textures

Simple descriptors:

- mean
- variance

But, there is a problem

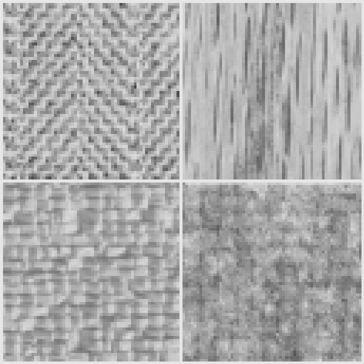


Figure: Textures with identical means and variances.

Statistics defined inside of a local window I

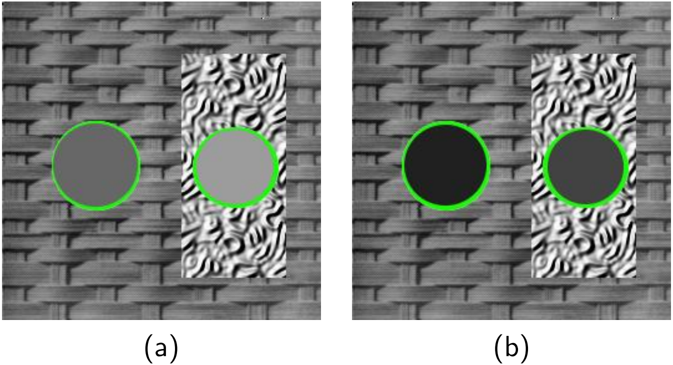


Figure: Illustration of texture statistics computed over a circle. (a) grayscale mean (103 and 156 respectively) (b) standard deviation (32 and 66 respectively).

Local statistics

Definition (Local mean)

The *local mean* over a spatial window W is defined as

$$\mu_f = \frac{1}{\sharp(W)} \sum_{(x,y)\in W} f(x,y) \tag{111}$$

where $\sharp(W)$ is the cardinality of W.

Definition (Local standard deviation)

The *standard deviation* over a spatial window W is defined as

$$\sigma_f = \sqrt{\frac{\sum_{(x,y)\in W} \left[f(x,y) - \mu_f\right]^2}{\sharp(W)}} \tag{112}$$

Global and local histograms I

Definition (Histogram)

The *histogram* of an image is the curve that displays the frequency of each grayscale level.

0	0	0	0	0	0
0	2	1	2	2	2
2	1	1	1	2	2
2	1	1	1	2	2
3	2	1	0	0	0
3	3	3	3	2	0

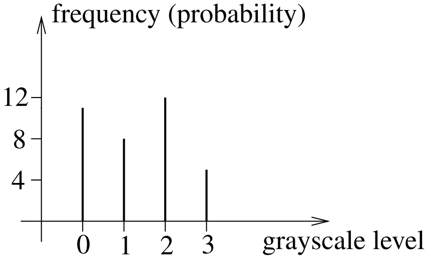
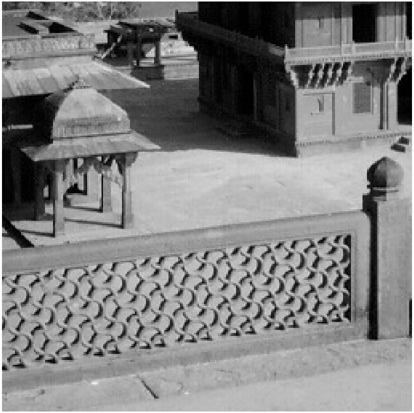
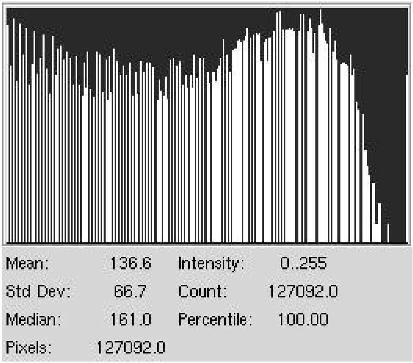


Figure: Non-normalized histogram of an image.

Global and local histograms II





```
Mean:        136.6   Intensity:   0..255
Std Dev:      66.7   Count:     127092.0
Median:      161.0   Percentile:  100.00
Pixels:   127092.0
```

Figure: An image and its global histogram (here W accounts for the whole image domain).

Global and local histograms III

Definition (Local histogram)

With a smaller window W, it is possible to define a local (normalized) histogram $p(l)$ as

$$p(l) = \frac{\sharp\left\{(x, y) \in \mathrm{W} \mid f(x, y) = l\right\}}{\sharp(\mathrm{W})} \tag{113}$$

Histogram statistics

- **Mean**

$$\mu_L = \sum_{l=0}^{L-1} l\, p(l) \tag{114}$$

  where L denotes the number of possible grayscale levels inside the W window.

- **Standard deviation**

$$\sigma_L = \sqrt{\sum_{l=0}^{L-1} (l-\mu_L)^2 p(l)} \tag{115}$$

- **Obliquity**

$$S_s = \frac{1}{\sigma_L^3} \sum_{l=0}^{L-1} (l-\mu_L)^3\, p(l) \tag{116}$$

- “**Kurtosis**”

$$S_k = \frac{1}{\sigma^4} \sum_{l=0}^{L-1} (l-\mu_L)^4\, p(l) - 3 \tag{117}$$

Co-occurrence matrix of a grayscale image I

Let us define a geometrical relationship. For example, we take

$$x_2 = x_1 + 1 \tag{118}$$

$$y_2 = y_1 \tag{119}$$

for which (x_2, y_2) is at the right of (x_1, y_1).

Definition (Co-occurrence matrix)

A co-occurrence matrix is defined by means of a geometrical relationship R between two pixel locations (x_1, y_1) and (x_2, y_2).
The co-occurrence matrix $C_R(i, j)$ is squared, with the $L \times L$ dimensions, where L is the range of all possible grayscale values inside of B. Indices of the co-occurrence matrix then indicate the amount of grayscale level value pairs as defined by R.

Co-occurrence matrix of a grayscale image II

Construction of the $C_R(i,j)$ matrix:

1. Matrix initialization: $\forall i,j \in [0,L[\ :\ C_R(i,j)=0$.
2. Filling the matrix. If the relationship/*condition* R between two pixels (x_1,y_1) and (x_2,y_2) is followed/*met*, then

$$C_R\left(f(x_1,y_1),f(x_2,y_2)\right) \leftarrow C_R\left(f(x_1,y_1),f(x_2,y_2)\right)+1$$

Example I

Let us consider an image with four grayscale levels ($L = 4$, and $l = 0, 1, 2, 3$):

$$f(x, y) = \begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \\ 2 & 2 & 3 & 3 \end{matrix} \tag{120}$$

$$\begin{aligned} P_{0^0,d}(i,j) \quad = \quad & \sharp\{(x_1, y_1), (x_2, y_2) \in B \mid y_1 = y_2, |x_2 - x_1| = d, \\ & f(x_1, y_1) = i \text{ and } f(x_2, y_2) = j\} \end{aligned} \tag{121}$$

The $P_{0^0,1}$ and $P_{90^0,1}$ matrices are 4×4 matrices respectively given by

$$P_{0^0,1} = \begin{bmatrix} 6 & 2 & 1 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 0 & 4 & 2 \\ 0 & 0 & 2 & 2 \end{bmatrix} \qquad P_{90^0,1} = \begin{bmatrix} 6 & 1 & 2 & 0 \\ 1 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \tag{122}$$

Example II

About the use of co-occurrence matrices:

[+] Rich information about the texture

[−] Explosion of the number of features in the case of small textures

There is a tradeoff between the number of parameters used to describe a texture and the discriminant/interpretation power of a method based on them.

Geometrical characterization of textures

Use of the Fourier transform

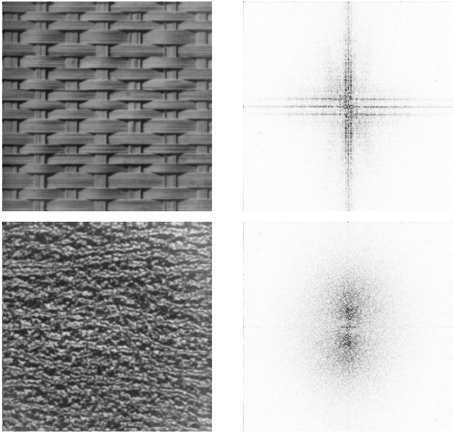


Figure: Spectral characterization of a texture. Right-hand images are the modules of the Fourier transforms (inverse video).

Textures and energy I

Measures are derived from three simple vectors: (1) $L_3 = (1, 2, 1)$ that computes the *mean*, (2) $E_3 = (-1, 0, 1)$ that detects edges, and (3) $S_3 = (-1, 2, -1)$ which corresponds to the second derivate. By convolving these symmetric vectors, Laws has derived 9 basic convolution masks:

$$\frac{1}{36}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Laws 1

$$\frac{1}{12}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Laws 2

$$\frac{1}{12}\begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix}$$

Laws 3

$$\frac{1}{12}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Laws 4

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Laws 5

$$\frac{1}{4}\begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

Laws 6

$$\frac{1}{12}\begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix}$$

Laws 7

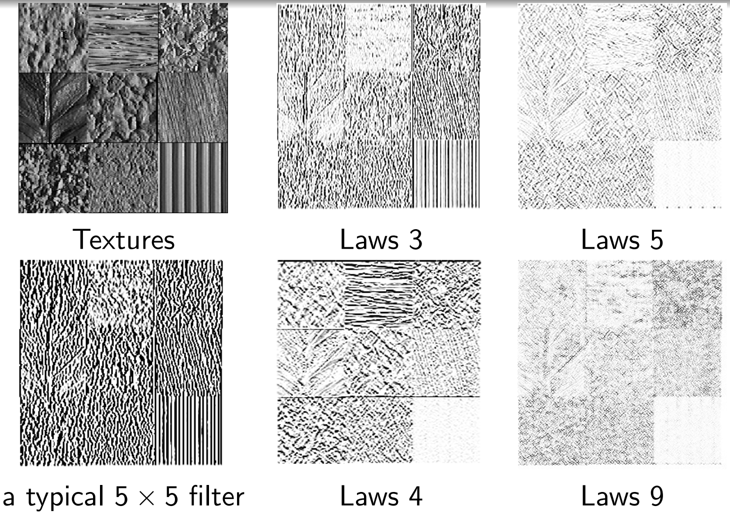
$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{bmatrix}$$

Laws 8

$$\frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Laws 9

Textures and energy II



Textures

Laws 3

Laws 5

a typical 5 × 5 filter

Laws 4

Laws 9

Figure: Laws “residues” (reverse video).

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
   - General considerations
   - Texture analysis
   - Shape description
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Shape analysis: outline

- Shape *description*. There are many of them:
  - Lineic shape description
  - Quadtree as a shape descriptor
  - Morphological skeleton
  - Other surfacic shape descriptors
- *Measures*
  - Basic geometrical measures
  - Shape factors
  - Moments
  - Morphological measures

Shape descriptors

There are **two main families** for describing a shape:

- lineic shape descriptors. They follow the border and encode its characteristics.
- surfacic shape descriptors. They represent the surface surrounded by the border.

Properties

- translation, rotation, and scale (affine) invariance
- robustness to noise
- robustness to partial occlusions

Shape descriptors

There are **two main families** for describing a shape:

- lineic shape descriptors. They follow the border and encode its characteristics.
- surfacic shape descriptors. They represent the surface surrounded by the border.

Properties

- translation, rotation, and scale (affine) invariance
- robustness to noise
- robustness to partial occlusions

How to encode a series of pixels? I

Chain code

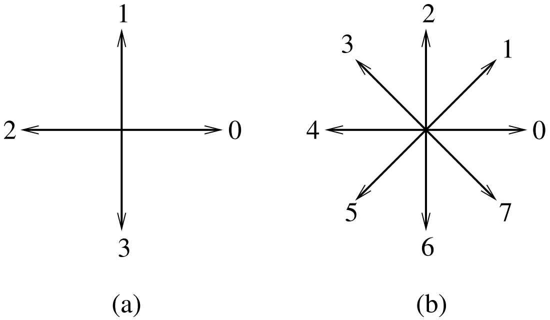


Figure: Definitions of directions in (a) 4-connectivity and (b) 8-connectivity.

How to encode a series of pixels? II

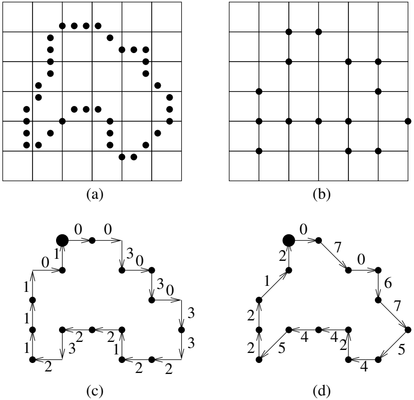
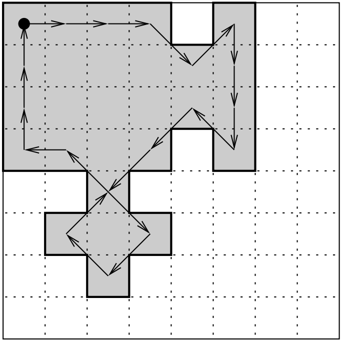
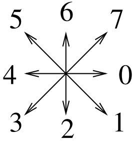

(a) (b) (c) (d)

Figure: (a) A contour, (b) contour sampled on a digital grid, (c) 4-connected chain code, and (d) 8-connected chain code.

How to encode a series of pixels? III





Code : 00017222533135754666

Figure: Contour with a crossing border pixel.

How to encode a series of pixels? IV

Direction of the descriptor:

- clockwise: external border
- counter-clockwise: internal border

Polygonal approximation

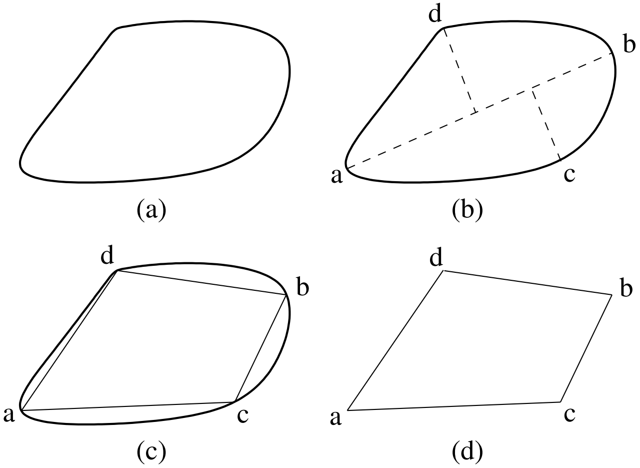


Figure: Polygonal approximation of a shape.

Fourier descriptors I

Some shape descriptors see the object as:

- a binary function

$$f(x,y)=\begin{cases}1 & \text{if } (x,y) \text{ belongs to the object}\\ 0 & \text{otherwise}\end{cases} \tag{123}$$

- a series of points in the complex 2D space

$$s[n]=x[n]+jy[n] \tag{124}$$

$s[n]$ is a one-dimensional discrete collection of complex numbers.

Fourier descriptors II

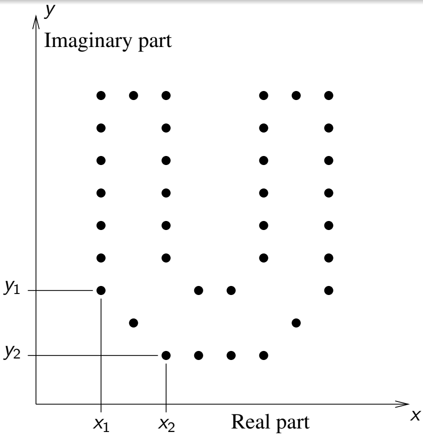


Figure: Representation of a shape as a series of points in the complex 2D space.

Fourier descriptors III

Definition (Fourier descriptor)

The discrete Fourier transform of $s[n]$ is defined as

$$\mathcal{S}[u] = \frac{1}{N} \sum_{n=0}^{N-1} s[n] e^{-2\pi jun/N} \tag{125}$$

The $\mathcal{S}[u]$ coefficients are the Fourier descriptors of the shape.

Approximations.

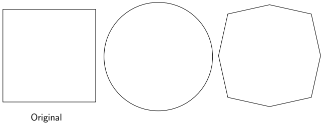


Figure: Original shape and approximations with a selection 1 (mid image) or 2 (right image) Fourier descriptors.

Approximation of a human silhouette

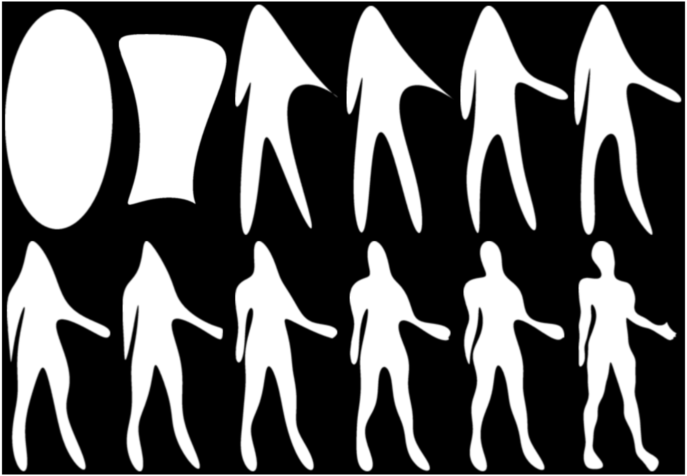


Figure: Approximations of a human silhouette by an increasing number of Fourier descriptors.

The quadtree as a shape descriptor

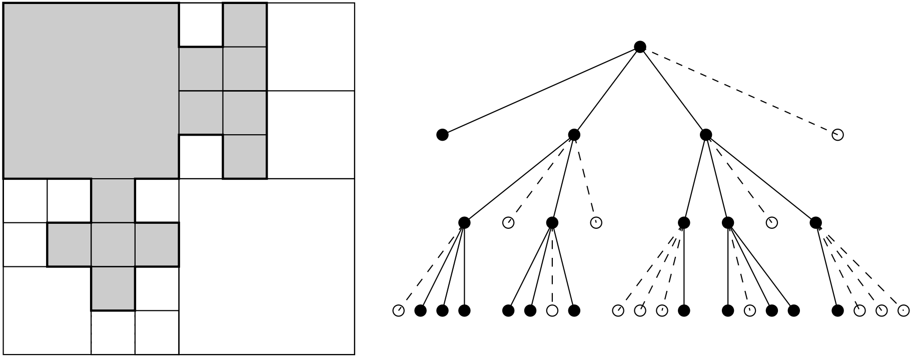
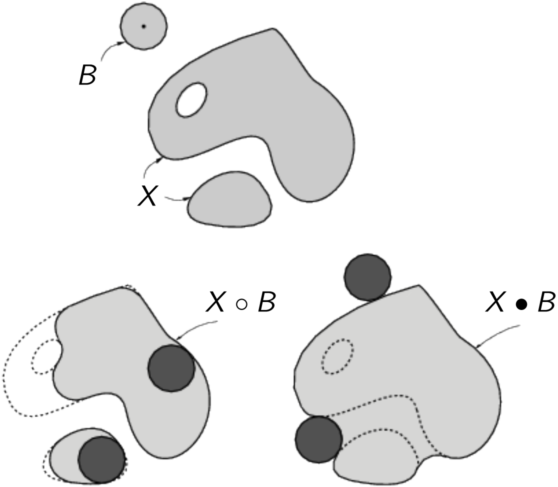


Figure: Quadtree decomposition and quadtree representation.

Opening and closing of X with a disk B



The morphological skeleton to describe a shape I

First attempts to define the skeleton

- Consider a continuous set X ans its frontier ∂X; an element x of the object X belongs to the skeleton of X, denoted by $S(X)$, if there exists a disk centered on x, included in X, that touches ∂X at least twice (maximal balls with two contact points).
- Locus of the center of all the maximal balls B contained in X.

These definitions are not fully equivalent, but they have the same topological closure in $\mathbb{R}^2$.

The morphological skeleton to describe a shape II

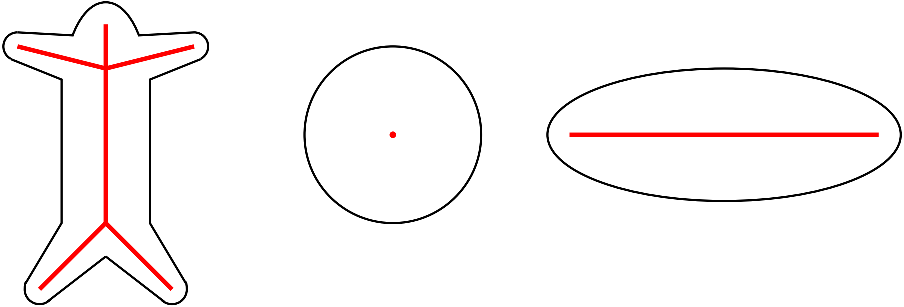


Figure: Shapes and their skeleton $S(X)$.

Skeletons are sensitive to noise on the object.

Difficulty

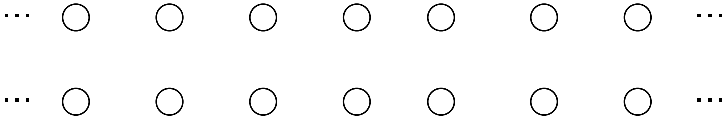


Figure: What is the skeleton of this object X? Ideally, it should be located between the two rows. In practice, it is on the upper or lower row.

Properties of a skeleton transform

1. The skeleton transform is not increasing, nor is it non-increasing. Indeed, $X \subseteq Y$ does not imply that $S(X) \subseteq S(Y)$, nor that $S(Y) \subseteq S(X)$.
2. The skeleton transform is anti-extensive: $S(X) \subseteq X$.
3. The skeleton transform is idempotent: $S(S(X)) = S(X)$.

Difficulty

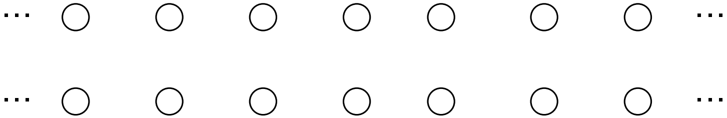


Figure: What is the skeleton of this object X? Ideally, it should be located between the two rows. In practice, it is on the upper or lower row.

Properties of a skeleton transform

1. The skeleton transform is not increasing, nor is it non-increasing. Indeed, $X \subseteq Y$ does not imply that $S(X) \subseteq S(Y)$, nor that $S(Y) \subseteq S(X)$.
2. The skeleton transform is anti-extensive: $S(X) \subseteq X$.
3. The skeleton transform is idempotent: $S(S(X)) = S(X)$.

Morphological size I

Definition (Continuous object size)

Let B be a continuous set whose size is arbitrary set to 1. Then, we can define a scale space of continuous sets by

$$rB = \{rb | b \in B\} \;\; r \geq 0 \tag{126}$$

where r is a continuous parameter.

Morphological size II

Definition

The *dilation* (*dila***ta***tion* in French!) of X by B is defined as

$$X \oplus B = \{x + b | x \in X,\ b \in B\} \tag{127}$$

Definition (Discrete object size)

Let B be a discrete, convex, finite, set of $\mathbb{Z}^2$, whose size is arbitrary set to 1. Then, we build a family of homethetic versions of B, for $n = 0,\ 1,\ \ldots,$ by

$$nB = \underbrace{B \oplus \ldots \oplus B}_{n-1 \text{ dilations}} \tag{128}$$

In this expression, nB is obtained as a cascade of $(n-1)$ successive dilations. By convention, for $n = 0$, $0B = \{(0,0)\}$.
Note that $nB \oplus mB = (n+m) \oplus B$ for every $n,\ m$.

Morphological size III

Theorem

Let $\partial(B)$ be the frontier of B in $\mathbb{R}^2$. Then

$$\partial(B) \oplus B = B \oplus B \tag{129}$$

Therefore,

$$nB = \underbrace{B \oplus \ldots \oplus B}_{n-1 \ \textit{dilations}} = B \oplus \underbrace{\partial(B) \oplus \ldots \oplus \partial(B)}_{n-2 \ \textit{dilations}} \tag{130}$$

Multiresolution filters

Definition

We define the multiresolution opening and closing respectively as

$$X \circ nB = (X \ominus nB) \oplus nB \tag{131}$$

$$X \bullet nB = (X \oplus nB) \ominus nB \tag{132}$$

Since $(X \ominus B) \ominus C = X \ominus (B \oplus C)$ and $(X \oplus B) \oplus C = X \oplus (B \oplus C)$,

$$X \circ nB = [\underbrace{(X \ominus B) \ominus B \ldots \ominus B]}_{\text{n erosions}} \underbrace{\oplus B \oplus B \ldots \oplus B}_{\text{n dilations}} \tag{133}$$

By definition,

$$X \circ nB = \bigcup_{(nB)_z \subseteq X} (nB)_z \tag{134}$$

Formal definition of the skeleton I

Definition (Skeleton [Lantuéjoul])

The skeleton of a set X is the union of a family of subsets S_n, each of them being the set of references to translates of nB included in X, except all the references of translates of $(n+1)B$ contained in X. Formally, we have

$$S_n(X) = (X \ominus nB) \backslash ([X \ominus nB] \circ B), \;\; n = 0, 1, \ldots, N \tag{135}$$

$$S(X) = \bigcup_{n=0}^{N} S_n(X) \tag{136}$$

where B is a 2×2 wide pixels set for a square digital grid or an hexagon for an hexagonal grid.

Formal definition of the skeleton II

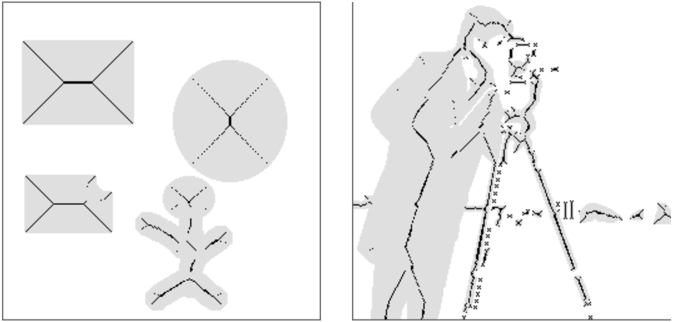


Figure: Some skeletons obtained with Lantuéjoul's formula.

Inverse skeleton

We define

$$\pi(X) = [[[S_N(X) \oplus B] \cup S_{N-1}(X)] \oplus B \cup S_{N-2}(X)...] \oplus B \cup S_0(X) \quad (137)$$

It can be shown that

$$\pi(X) = \bigcup_{n=0}^{N} [S_n(X) \oplus nB] = X \quad (138)$$

⇒ it is possible to reconstruct the shape perfectly based on the skeleton

Alternative skeleton formulas I

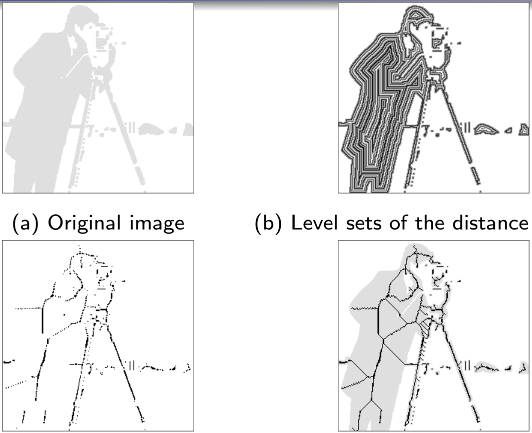
Definition (Distance function)

The distance function of a set $X \subseteq \mathcal{E}$, denoted by $\phi(X)$, is defined as

$$[\phi(X)](h) = d(X^c,\, h) \tag{139}$$

with the convention that $d(\emptyset, h) = +\infty$ for $h \in \mathcal{E}$.

Alternative skeleton formulas II



(a) Original image

(b) Level sets of the distance function

(c) Local maximum of (b)

(d) Skeleton by maximal balls

Figure: Skeleton by maximal balls.

Alternative skeleton formulas III

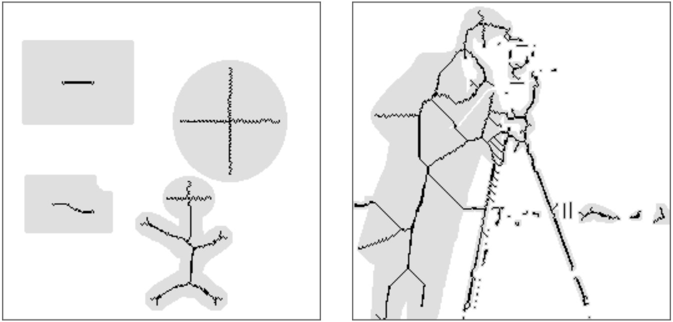


Figure: Some skeletons obtained with Vincent's algorithm.

Surfacic shape descriptors based on a catalog of primitives

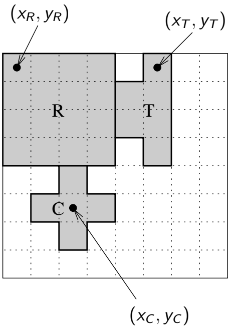


Figure: A shape is described as the union of a rectangle, a triangle and a diamond.

Summary of shape descriptors

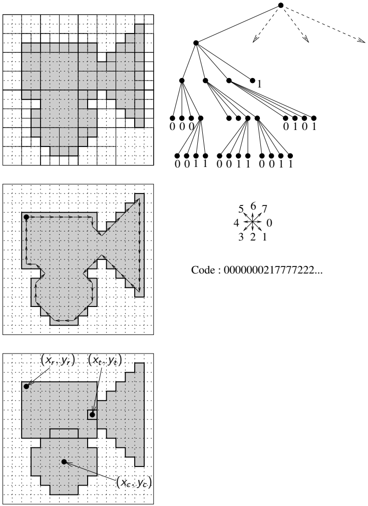


Figure: Comparison of some shape descriptors.

Measures I

There are three fundamental measures

1. Perimeter
2. Area
3. Euler-Poincarré number

There is a relationship between the number of connected components C and the number of holes H.

Definition (Euler number)

The Euler number E is defined as

$$E = C - H \tag{140}$$

Measures II

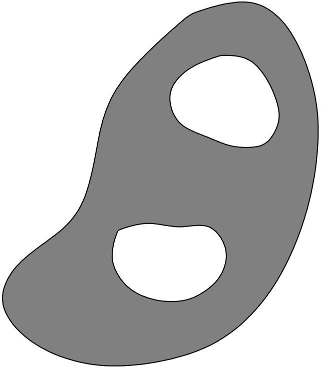


Figure: Euler number: $E = 1 - 2 = -1$.

Measures III

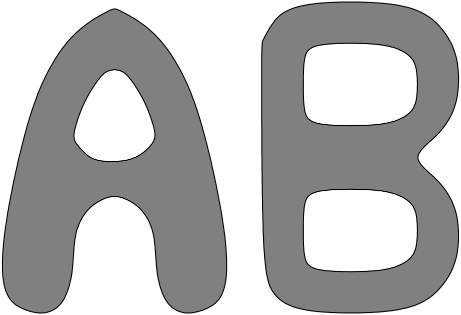
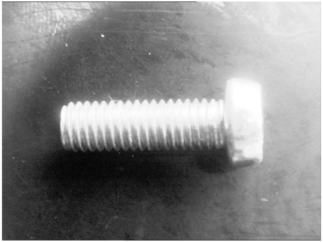


Figure: Two letters with a different Euler number.

Shape factors



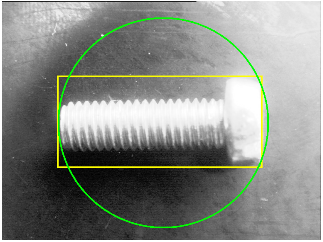


Figure: An object X and two reference sets (circumscribing disk/ellipse and smallest rectangular bounding box R).

Shape factor name	Expression
compactdness	$\frac{P(X)^2}{4\pi A(X)}$
rectangularity	$\frac{A(X)}{A(R)}$
circularity	$\frac{4A(X)}{\pi D_{max}^2}$
anisometry	$\frac{D_{max}}{D_{min}}$

Moments

Moments are surfacic measures, used for example in character recognition. Remember that we can see an object as a binary function

$$f(x,y)=\begin{cases}1 & \text{if } (x,y) \text{ belongs to the object}\\ 0 & \text{otherwise}\end{cases} \tag{141}$$

Definition (Moment of order $p+q$)

The $p+q$ moment of a function $f(x,y)$ is defined as

$$m_{pq}=\sum_x\sum_y x^p y^q f(x,y) \tag{142}$$

All these moments can be centered:

$$\mu_{pq}=\sum_x\sum_y (x-m_{10})^p(y-m_{01})^q f(x,y) \tag{143}$$

and they can be normalized:

$$\eta_{pq}=\frac{\mu_{pq}}{\mu_{00}^{\frac{p+3}{2}}} \tag{144}$$

Morphological measures and granulometry I

Granulometry is an approach to compute a size distribution of grains in binary images, using a series of opening operations.

Consider a ball nB whose radius is given by n, then we define a granulometric curve by

$$\psi(n) = A(X \circ nB) \tag{145}$$

where $A()$ denotes the area.
Abrupt changes is this curve are interesting for detecting typical sizes.

Definition (Pattern spectrum)

The Pattern Spectrum (PS) is defined as

$$\text{PS}(n) = -\frac{A(X \circ nB)}{dn} \tag{146}$$

Morphological measures and granulometry II

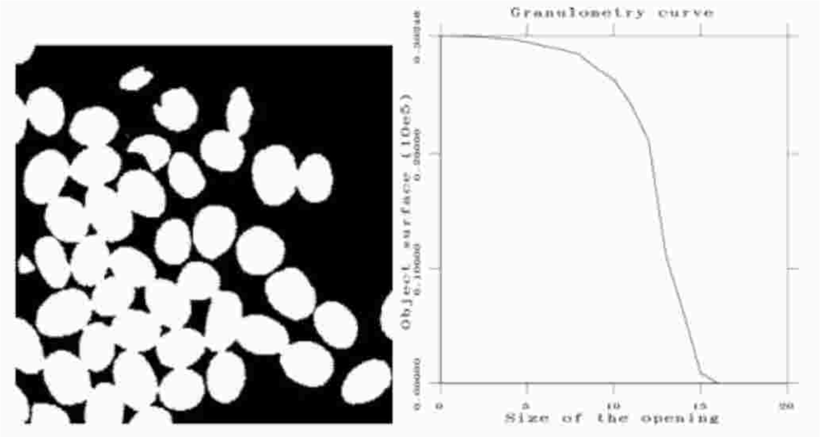


Figure: A binary object and a corresponding granulometric curve.

Convexity

Definition (Convexity)

A set $X \subseteq \mathcal{E}$ is convex if $rx + (1-r)y \in X$ for every $x, y \in X$ where $r \in [0,1]$.

In other words, a line joining two arbitrary points of X has to be entirely included in X.

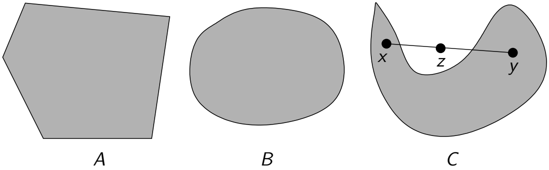


Figure: A, B are convex; C is not convex.

Convex envelope I

The notion of convexity leads to that of convex envelope

Definition (Convex envelope)

The convex envelope of a set $X \subseteq \mathcal{E}$, denoted by $co(X)$, is the intersection of all the convex sets comprising X.

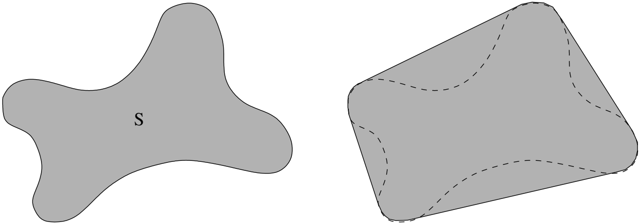


Figure: A set X and its convex envelope.

Convex envelope II

Theorem

For $X,\ Y \subseteq \mathcal{E}$,

$$co(X \oplus Y) = co(X) \oplus co(Y) \tag{147}$$

Definition (Convexity shape factor)

The convexity shape factor is defined as

$$C = \frac{A(X)}{A(co(X))} \tag{148}$$

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
   - Motion analysis by tracking
   - Background subtraction
   - Technical details of background subtraction
   - Evaluation
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Detection of motion

There are basically two “pure” approaches for motion analysis/detection in a video sequence:

1. Motion analysis by *tracking* (= *motion estimation* based techniques):
   - detects some particular points in a video frame.
   - find the corresponding points/objects in the next frame.
   - based on a model, interpret the trajectories of the points/objects (usually at the object level).
2. Motion detection by *background subtraction*:
   - build a reference frame or model with no foreground in it.
   - compare a next frame to the reference.
   - update the reference.

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
   - Motion analysis by tracking
   - Background subtraction
   - Technical details of background subtraction
   - Evaluation
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Motion analysis by tracking: principles

There are several techniques but, usually, they involve the following steps:

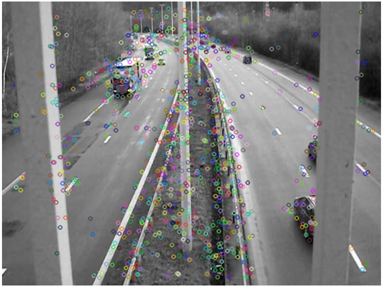
1. detect features in successive frames.
2. make some correspondences between the features detected in consecutive frames
3. based on a model, regroup some features to facilitate the tracking of objects.

Feature detection

Some known feature detectors:

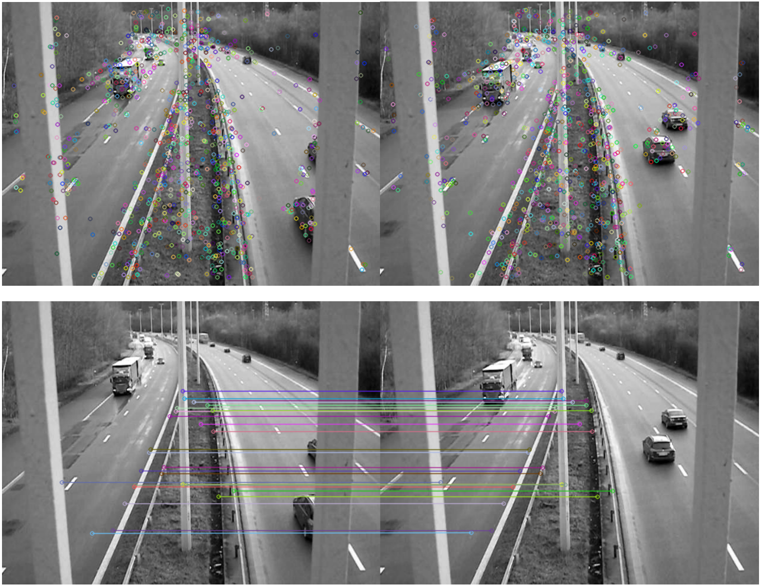
- Harris's corner detector
- Scale Invariant Feature Transform (SIFT)
- Speeded Up Robust Features from an image (SURF)
- Features from Accelerated Segment Test (FAST)
- ...

Original image



Features detected by SURF

Feature correspondence



Difficulties for feature correspondence

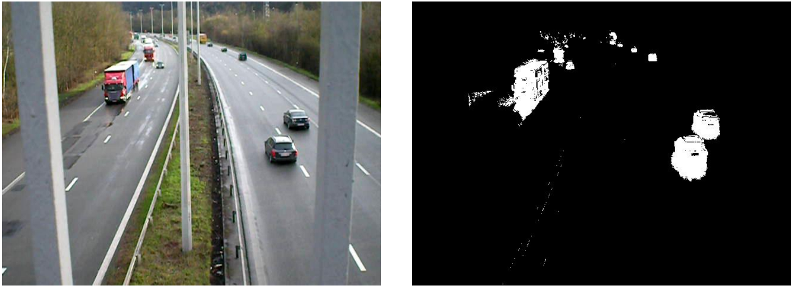
Typical questions/difficulties for tracking approaches (targeting *motion estimation*):

- How to filter the features? (remove some useless features)
- How do we regroup features?
  - we need a *model*. But this introduces a **bias towards the model**.
- How do we ensure continuity over time?
- What happens when occlusions occur?
- How to solve ambiguities (one feature in one image is identical to several in the next frame)?
- ...

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
   - Motion analysis by tracking
   - Background subtraction
   - Technical details of background subtraction
   - Evaluation
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Motion detection by background subtraction I



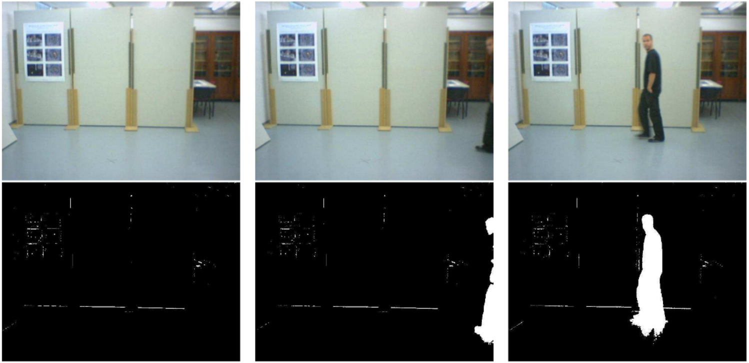
Original image

Features detected by ViBe

Figure: Segmentation (pixels that are “in motion”) by background subtraction.

Background subtraction is also referred to as *change detection*.

Example of a pixelwise background subtraction technique



- Any application with moving objects
- Video-surveillance

Challenges

- Input related issues
    - lighting/illumination changes
        - slow (day/night cycles)
        - fast (light switch, clouds in the sky, ...)
    - unwanted motions
        - camera shaking (wind)
        - in the background (tree leaves, waving grass, water)
    - appearance changes of foreground objects (reflections), shadows, camouflage, ...
- Implementation related issues
    - robustness
    - real time

Applications

- Motion detection (you don't need a precise segmentation map):
  - raise an alarm
  - start/stop recording
- Foreground/background segmentation:
  - counting
  - traffic analysis
  - human motion interpretation
  - sport analysis
  - coding
  - aerial imaging
  - detection of exoplanets
  - ...

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
   - Motion analysis by tracking
   - Background subtraction
   - Technical details of background subtraction
   - Evaluation
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Background subtraction

Common tasks:

- background subtraction ⇒ segmentation
- change detection ⇒ generate alarms (and reduce rate of false alarms!)

Plethora of methods (2023):

- 'motion detection' on IEEE Xplore: 34,620 papers
- 'background subtraction video' on IEEE Xplore: 3,750 papers

Different spatial contexts:

- pixel-based (pixelwise)
- region-based
- tracking (spatio-temporal neighborhood)

Techniques for background subtraction

Objective of pixel-based background subtraction techniques

Separate the *foreground* (pixels “in motion”) from the *background* (“static” pixels).

Implementation perspective

A video sequence is like a data cube whose dimension is only fixed in 2 dimensions.

- The data cube extends with time.

Challenges for building a background subtraction algorithm:

1. need to find a way to accumulate knowledge of increasing size inside of a constant sized memory block.
2. this knowledge should be updated regularly to deal with changes.

Techniques for background subtraction

Objective of pixel-based background subtraction techniques

Separate the *foreground* (pixels "in motion") from the *background* ("static" pixels).

Implementation perspective

A video sequence is like a data cube whose dimension is only fixed in 2 dimensions.

- The data cube extends with time.

Challenges for building a background subtraction algorithm:

1. need to find a way to accumulate knowledge of increasing size inside of a constant sized memory block.
2. this knowledge should be updated regularly to deal with changes.

Implementation of background subtraction in 3 steps

Steps

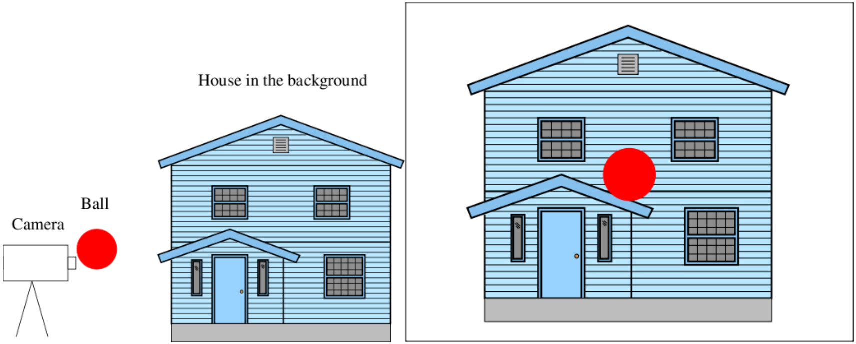
[1. Initialization] build a *reference frame* or a *statistical model* for the background.

[2. Subtraction or segmentation] *compare* the current frame to the reference frame or model, and "*subtract*" the frame to get a binary image indicating pixels who have changed.

[3. Updating] *update* the reference frame or model.

When we develop a technique, we have to detail these three steps!

Principles

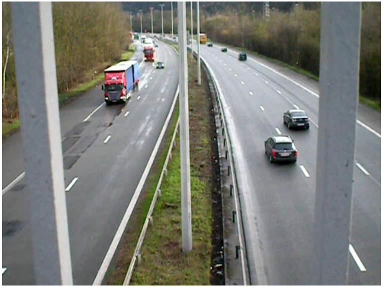


Real scene

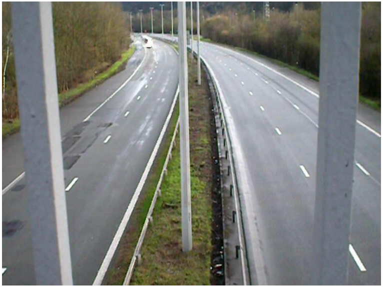
What a camera sees

- Major assumption: fixed camera
- Otherwise,
  1. we have to compensate for the camera motion
  2. what for new areas (for which we have no past information)?

Reference frame or a model?



One frame in the sequence



Built reference frame

Figure: Building a reference frame (or an unimodal model).

Building a reference frame from a video sequence is the *task* of background estimation/background generation.
For real-time processing, we need a reference frame as soon as possible.

Elementary method (reference frame)

Naive approach for the step of segmentation (static background)

Foreground is detected, pixel by pixel, as the difference between the current frame and a static reference image (background).

Let p be a pixel location. p belongs to the background if

$$|I_t(p) - B(p)| \leq \text{threshold} \tag{149}$$

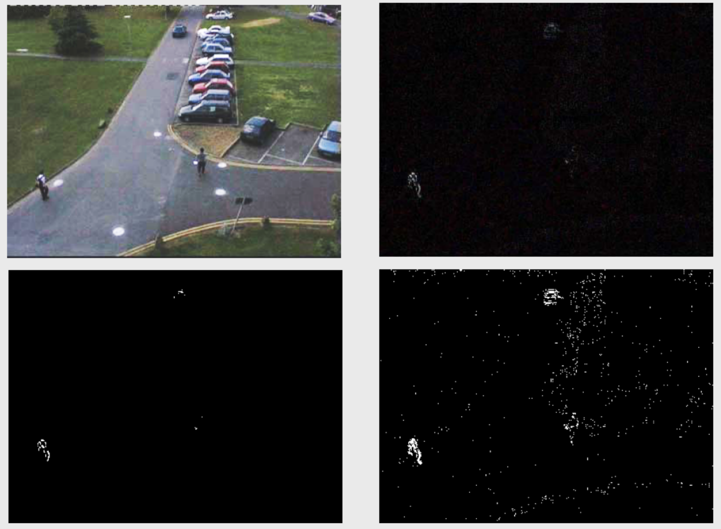
where

- $I_t(p)$ is the current pixel value (at time t),
- $B(p)$ is the reference background value for that pixel.

Problems:

- How do we *choose* the reference image?
- The reference image should *change over time*.
- What is the best *threshold*?

Problem with a unique threshold for the whole image



Choices for a better reference image

Simple techniques (for the step of updating)

- One “good” image is chosen (usually a frame empty of foreground objects).
- *Exponentially* updated reference image (*blending*)

$$B_t(p) = \alpha I_t(p) + (1 - \alpha) B_{t-1}(p) \tag{150}$$

Typical value for α: 0.05 (in that case, $B_t = 0.05\, I_t + 0.95\, B_{t-1}$)

- Median of the last N frames (for each pixel separately).

Intermediate summary: remember the components

- Initialization
  - ⇒ fill in the first reference image or initialize the model
  - difficulty: problem with the presence of *ghosts*.
- Segmentation
  - ⇒ rules + parameters ...
  - difficulties:
    - homogeneous rule for the whole image?
    - how to set the values of the parameters?
- Updating
  - ⇒ we need strategies
  - difficulty: how to adapt the scene dynamics?

There are two main strategies for the updating

Blind *update* (the most common)

update the reference image or the model for all pixels, regardless of the segmentation result (that is if the pixel is a foreground or a background pixel)

- does it make sense?

Conservative update

only update when the pixel belongs to the background

- what if the segmentation was wrong (deadlock)?

There are two main strategies for the updating

Blind update (the most common)

update the reference image or the model for all pixels, regardless of the segmentation result (that is if the pixel is a foreground or a background pixel)

- does it make sense?

Conservative update

only update when the pixel belongs to the background

- what if the segmentation was wrong (deadlock)?

Advanced techniques

The background is modeled as a probability density function to be estimated

- One Gaussian distribution per pixel
- *Mixture of Gaussians for each pixel*
- *Kernel-based estimation of the probability* density function

One Gaussian per pixel

[Model]
For each pixel, the probability density function of observed values is modeled by a **single** Gaussian.
Let v be an observed value for a pixel p:

$$\mathrm{pdf}(v) \sim G(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(v-\mu)^2}{2\sigma^2}} \tag{151}$$

Once the model is built (here it means that we need to estimate the mean and variance), we evaluate the distance to the mean.

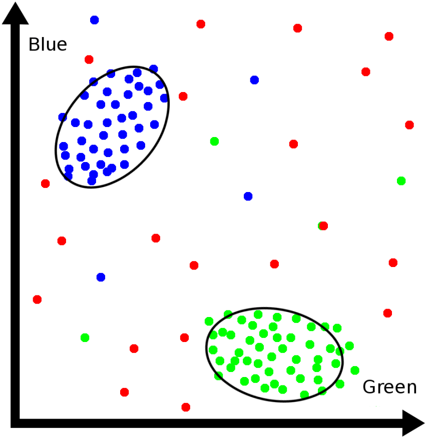
[Segmentation step]
If

$$|v_t - \mu| \leq \text{parameter} \times \sigma \tag{152}$$

then the pixel p belongs to the background.

Mixture of Gaussians Model

Motivation: the probability density function of the background is multi-modal



[Stauffer, 1999, 2000] [Power, 2002] [Zivkovic, 2006]

Mixture of Gaussians (MoG) Model

For each pixel:

$$\mathrm{pdf}(v) \sim \sum_{i=1}^{N} \alpha_i G(\mu_i, \sigma_i) \tag{153}$$

Typical values for n: 3 or 5

Fundamental assumptions

- The background has a low variance.
- The background is more frequently visible than the foreground.

Kernel Density Estimation (KDE) methods

For each pixel:

$$\mathrm{pdf}(v) \sim \sum_{i=1}^{N} \alpha_i K_\sigma(v - v_i) \tag{154}$$

where $\{v_i\}_{i=1,\ldots,N}$ are the N last values observed (samples) for that pixel, and $K_\sigma()$ is a kernel probability function centered at v_i.

Decision rule

The pixel belongs to the background if $\mathrm{pdf}\,(v) \geq$ threshold.

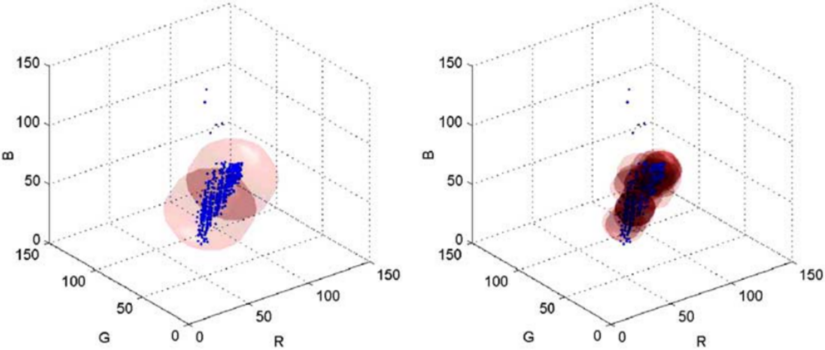
[Elgammal, 2000] [Zivkovic, 2006]

Parameters of KDE techniques

Typical values $\text{pdf}(v) \sim \sum_{i=1}^{N} \alpha_i K_\sigma(v - v_i)$ with

- Number of samples: $N = 100$
- Weight: $\alpha_i = \alpha = \frac{1}{N}$
- Spreading factor: $\sigma = \text{Variance}(v_i)$
- Probability density function chosen to be Gaussian: $K_\sigma(v - v_i) = G(v_i, \sigma^2)$

GMM techniques vs KDE techniques



Other advanced techniques

- Codebook
- Principal Components Analysis (PCA)
- Relaxation techniques (mainly post-processing of binary segmentation maps)
- Local Binary Patterns
- ViBe: no model, all based on samples
- Deep learning based methods

More details about ViBe: design rules

The design of ViBe was motivated by:

1. *information at the pixel level* should be preferred to aggregated features (for speed and efficiency).
2. *collect samples*, rather than build a statistical model:
    1. sample values have been observed in the past.
    2. no bias towards a model.
3. *minimize the number* of collected samples.
4. *no "planned obsolescence"* notion for samples.
    1. don't use the commonly (and exclusively) adopted substitution rule that consists to replace oldest samples first or to reduce their associated weight.
    2. all samples are equally valid.
5. foresee a mechanism to ensure *spatial consistency*.
6. keep the *decision* process *simple*.

Requirements while designing ViBe

1. No pre- or post-processing!
2. Be real-time.
3. A unique set of parameters.
4. Working as soon as you get the second frame.

Keys of a background subtraction technique

1. **Model**. What is a good model for the background?
2. How to **classify** pixels in the background/foreground? Need for a classification criterion.
3. How to **update** the model?
4. **Initialization**?

Model

- Model each background pixel with *a set of samples* (non-parametric method), instead of with an explicit pixel model. Each background pixel x is modeled by a collection of N background sample values: $M(x) = \{v_1, v_2, \ldots, v_N\}$
  - Avoid the difficult task of estimating the probability density function.
  - No statistical notion. Assume a binary image, what is the meaning of the mean? It is even a value that might never be observed ...
  - How could a model consider values of its immediate neighborhood?

Pixel classification

- A pixel belongs to the background if there are at least two matches: $\#\{S_R(v(x)) \cap \{v_1, v_2, ..., v_N\}\} \geq \#_{\text{min}}$
- $N = 20$ and $\#_{\text{min}} = 2$

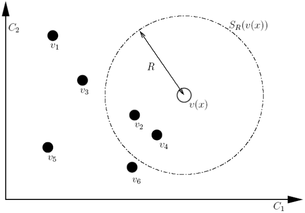


Fig. 1. Comparison of a pixel value with a set of samples in a two dimensional Euclidean color space (C_1, C_2). To classify $v(x)$, we count the number of samples of $\mathcal{M}(x)$ intersecting the sphere of radius R centered on $v(x)$.

Updating the model over time

Principles:

- **Conservative update**: update the model only if a pixel value is considered as a background. If it is foreground, do not update at all.

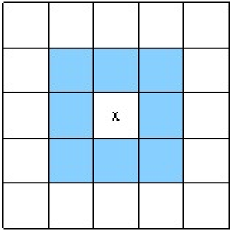
Mechanisms:

1. The background model of a pixel is **updated randomly**! One sample in the model is chosen randomly and replaced. The lifetime of the replaced value is ignored.
2. Random **time subsampling**. Not every model is updated. Which one is decided randomly (typically 1 out of 16, on average).
3. Randomly select a neighbor and modify the model of it with the value of the local pixel and according to the 1 and 2 mechanisms (**spatial diffusion**).

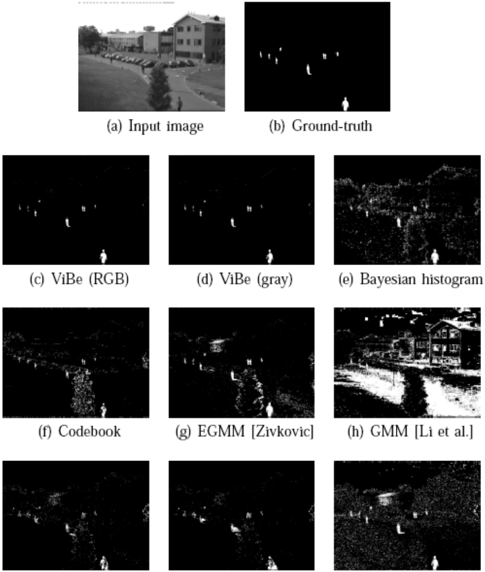
Model Initialization

From a single frame, populate the pixel models with values found in the spatial neighborhood of each pixel.

- Values randomly taken in their neighborhood
- Values chosen in the close 8-connected neighborhood



Results



(a) Input image

(b) Ground-truth

(c) ViBe (RGB)

(d) ViBe (gray)

(e) Bayesian histogram

(f) Codebook

(g) EGMM [Zivkovic]

(h) GMM [Li et al.]

(i) Gaussian model

(j) 1^{st} order filter

(k) Sigma-Delta Z.

Deep learning based background subtraction techniques

Traditionally

We choose the best feature set and optimize the background subtraction algorithm. But there are many (sub-optimal) possibilities.

- T. Bouwmans, C. Silva, C. Marghes, M. Zitouni, H. Bhaskar, and C. Frelicot. On the role and the importance of features for background modeling and foreground detection. CoRR, abs/1611.09099:**1–131**, 2016.

What if we automate the search (quest) for the best features?

What's next? I

More and more machine learning for background subtraction.

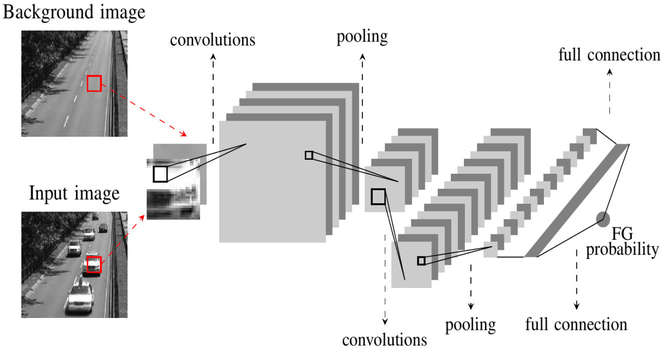


Figure: Deep learning for extracting the background in video scenes (M. Braham and M. Van Droogenbroeck. **Deep Background Subtraction with Scene-Specific Convolutional Neural Networks**. In *IEEE IWSSIP*, May 2016).

What's next? II

Method	$F_{overall}$	$F_{Baseline}$	F_{Jitter}	$F_{Shadows}$	$F_{LowFramerate}$
ConvNet-GT	**0.9046**	**0.9813**	**0.9020**	**0.9454**	**0.9612**
IUTIS-5	0.8093	0.9683	0.8022	0.8807	0.8515
SuBSENSE	0.8018	0.9603	0.7675	0.8732	0.8441
PAWCS	0.7984	0.9500	0.8473	0.8750	0.8988
PSP-MRF	0.7927	0.9566	0.7690	0.8735	0.8109
ConvNet-IUTIS	0.7897	0.9647	0.8013	0.8590	0.8273
EFIC	0.7883	0.9231	0.8050	0.8270	0.9336
Spectral-360	0.7867	0.9477	0.7511	0.7156	0.8797
SC_SOBS	0.7450	0.9491	0.7073	0.8602	0.7985
GMM	0.7444	0.9478	0.6103	0.8396	0.8182
GraphCut	0.7394	0.9304	0.5183	0.7543	0.8208

Table: Overall and per-category F scores for different methods.

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

- Motion analysis by tracking
- Background subtraction
- Technical details of background subtraction
- Evaluation

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

Evaluation

Elements

1. Metric
2. Ground truth data
3. Interpretation

Evaluation tasks

- Determine the intrinsic performance of a method
- Compare methods
- Rank methods

Outline

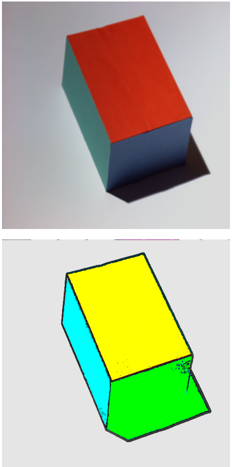
1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
   - General discussion
   - Evaluation criteria and metrics
   - Illustrations
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Outline

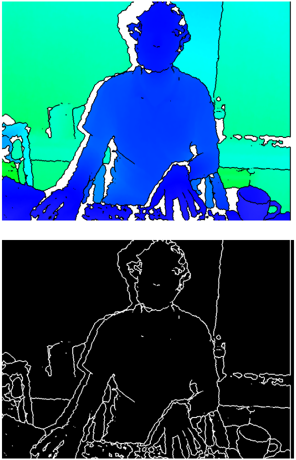
1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
   - General discussion
   - Evaluation criteria and metrics
   - Illustrations
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

What do we want to evaluate?

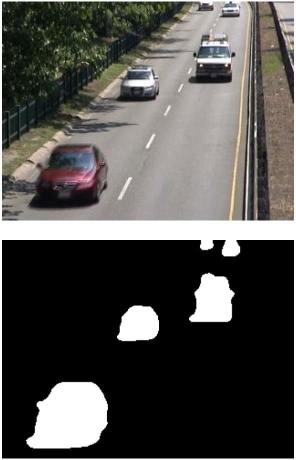
Segmentation



Edge detection



Background subtraction



It is **challenging**...

General principles of evaluation I

The process of evaluation might involve:

- The definition of a *methodology* that comprises *experiments* (note that it is important to be explicit on experimental conditions).
- *Criteria* or *scores.*
- *Reference data*:
    - *unlabeled* data, which is collected "in the wild".
    - *labeled* or *annotated* data, sometimes named as *ground truth* data.

General principles of evaluation II

Typology of evaluation methods:

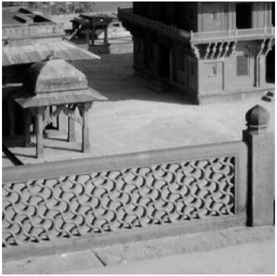
1. Subjective. The evaluation process involves a representative group of human viewers.

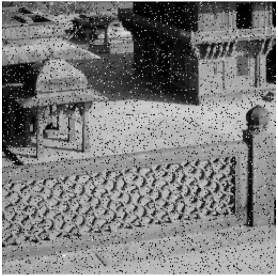
   [+] takes the *user's experience* into account.
   [−] is *time-consuming* (→ expensive),
   provides *subjective evaluation score*s,
   depends on the *experimental setup*.

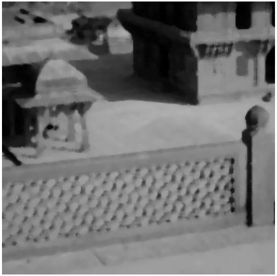
2. Objective. Based on objective measurements.

Objective quality measures and distortion measures I

Let f be the input image (whose size is $N \times N$) and $\hat{f}$ be the image after some processing.







Original data	Noisy image	Cleaned image
Ground truth	*Input*	*Output*
Not always available	Available	Calculated
	f	$\hat{f}$

Objective quality measures and distortion measures II

Definition (**Mean Square Error**)

$$\mathrm{MSE} = \frac{1}{NM} \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} \left(f(j,k) - \hat{f}(j,k) \right)^2 \tag{155}$$

Definitions (**Signal to noise ratios**)

[Signal to Noise Ratio]

$$\mathrm{SNR} = \frac{\sum_{j=0}^{N-1} \sum_{k=0}^{M-1} (f(j,k))^2}{\sum_{j=0}^{N-1} \sum_{k=0}^{M-1} \left(f(j,k) - \hat{f}(j,k) \right)^2} \tag{156}$$

[Peak Signal to Noise Ratio]

$$\mathrm{PSNR} = \frac{NM \times 255^2}{\sum_{j=0}^{N-1} \sum_{k=0}^{M-1} \left(f(j,k) - \hat{f}(j,k) \right)^2} \tag{157}$$

On the importance of ground truth data

Depending of the availability of ground truth, we have:

- *Standalone evaluation*.
  When a reference is not available.
- *Relative evaluation*.
  When ground truth data is available for comparison.

Properties of “good” ground truth data

1. representative for the application.
2. different conditions for the acquisition (large variety of samples, always directly related to the application).
3. correctly annotated (→ there is some agreement on the quality of the reference).

On the importance of ground truth data

Depending of the availability of ground truth, we have:

- *Standalone evaluation*.
  When a reference is not available.
- *Relative evaluation*.
  When ground truth data is available for comparison.

Properties of “good” ground truth data

1. representative for the application.
2. different conditions for the acquisition (large variety of samples, always directly related to the application).
3. correctly annotated (→ there is some agreement on the quality of the reference).

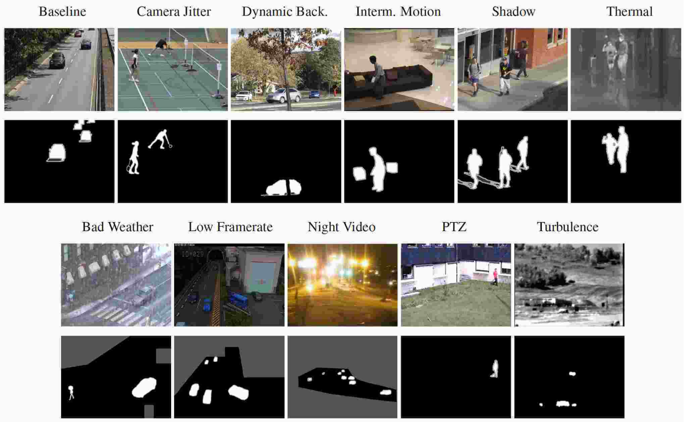
Difficulties specific to ground truth data

Annotation process

- the "annotation" process (which consists to build the ground truth data) is complex and sometimes debatable.
  Two "flavors":
  - synthetic data (which comes with synthetic annotations)
  - real data with manually annotated data.
- the annotation process is partly subjective.

An example of ground truth data I

ChangeDetection.NET (CDNET) dataset: a dataset for testing background subtraction algorithms



An example of ground truth data II

Main characteristics:

- 11 categories: about 5 videos per category
- long video sequences
- thousands of annotated images (ground truth data)
  - 5 labels: *static* ($\equiv$ background), *shadow*, *non-ROI*, *unknown*, and *moving* ($\equiv$ foreground)
- 7 performance metrics are computed
- ranking per video, category, and globally

Evaluation tasks

We can distinguish 3 main reasons for evaluation:

1. Optimization/analysis of a solution. For example, we have different parameter sets, then
   - ▷ *what are the best parameters?*
2. Comparison between algorithms
   - ▷ *Is algorithm A better than algorithm B?*
3. Ranking.
   - ▷ *Which one is the best?*

Ideally, evaluation tools should be different for these tasks. In practice, there is a confusion.

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

- General discussion
- Evaluation criteria and metrics
- Illustrations

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

A framework for developing evaluation criteria

Terminology from the *classification theory*.

Definition (Classification)

In machine learning and statistics, classification is the process of identifying the *category/class of a new observation*, on the basis of a training set of data containing observations (or instances) whose category membership is known.

The categories are named "*classes*".

Example (In background subtraction, there are two classes:)

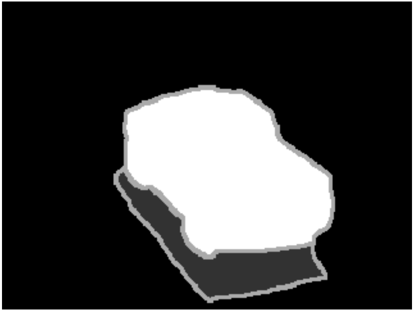
1. Foreground
2. Background

But there could be more: shadows, sky, road, etc.

The notion of positive and negative

Let us consider a two-class problem and denote the "negative" and "positive" classes by c^- and c^+, respectively. In background subtraction, background $\equiv$ negative and foreground $\equiv$ positive .

Original image



Binary segmentation map

Figure: Background subtraction identifies pixels belonging to the foreground (positive class c^+, white pixels) and pixels belonging to the background (negative class c^-, black pixels).

Some definitions and notations

A classifier estimates the class of each new sample (pixel in the case of background subtraction).

There are two notions of classes:

[true class or ground truth] it is denoted by $y \in \{c^-, c^+\}$.

[estimated class] the class estimated by the classifier is denoted by $\widehat{y} \in \{c^-, c^+\}$

Other notions:

- $\pi^- = p\left(y = c^-\right)$ and $\pi^+ = p\left(y = c^+\right)$ are the priors.
- the *rates of negative* and *positive predictions*, denoted by $\widehat{\pi^-} = p\left(\widehat{y} = c^-\right)$ and $\widehat{\pi^+} = p\left(\widehat{y} = c^+\right)$ respectively (not to be confused with the priors!).

Confusion matrix

Definition (Confusion matrix)

There are four possibilities, described by the confusion matrix:

		prediction $\widehat{y}$	
		c^+	c^-
real class y	c^+	True Positive (TP)	False Negative (FN)
	c^-	False Positive (FP)	True Negative (TN)

Remarks:

- There are four quantities: TP, FN, FP, TN.
- It is easy to extend a confusion matrix to more than two classes.
- How do we build metrics/performance indicators?

Evaluation metrics

Definitions (Conditional probabilities)

They involve the true class:

- **True Positive Rate**, also named **sensitivity** or **recall**, is defined as

$$\mathrm{TPR} = p\left(\widehat{y} = c^{+} | y = c^{+}\right) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{158}$$

- **True Negative Rate** (also named **specificity**) is

$$\mathrm{TNR} = p\left(\widehat{y} = c^{-} | y = c^{-}\right) = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{159}$$

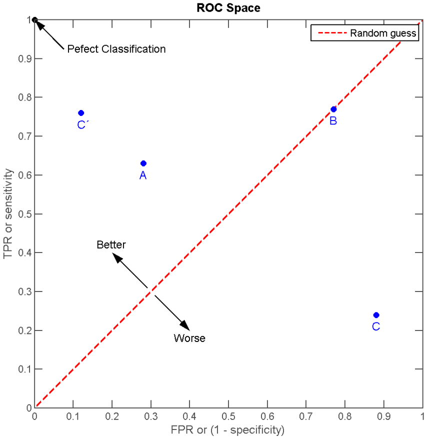
These quantities are linked to the

- *False Positive Rate* $\mathrm{FPR} = (1 - \mathrm{TNR}) = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- *False Negative Rate* $\mathrm{FNR} = (1 - \mathrm{TPR})$

Receiver Operating Characteristic (ROC) evaluation space

Definition (Receiver Operating Characteristic (ROC))

The (FPR, TPR) pair defines the ROC evaluation space.



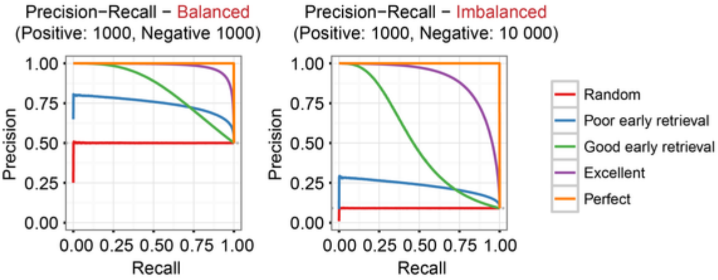
Precision/recall evaluation space

Definition (Precision)

$$\mathrm{P} = p\left(y = c^{+} | \widehat{y} = c^{+}\right) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{160}$$

Definitions (Recall [≡ TPR])

$$\mathrm{R} = p\left(\widehat{y} = c^{+} | y = c^{+}\right) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{161}$$



Particularities of the ROC and Precision/Recall space

- Unachievable zone in the PR space, but only the upper part of the ROC curve makes sense.
- In the ROC space, the diagonal represents the random classifiers.

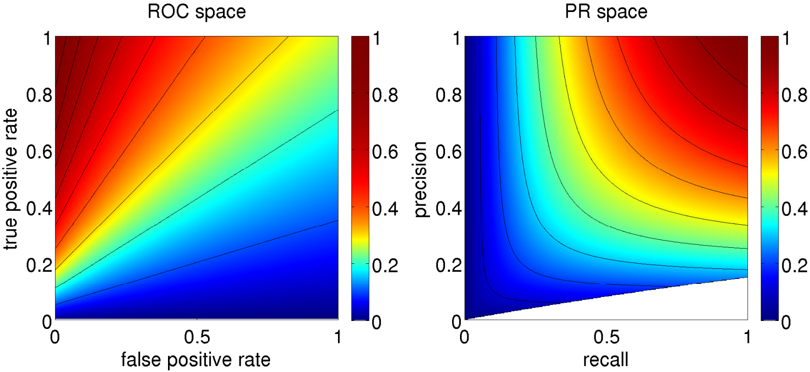


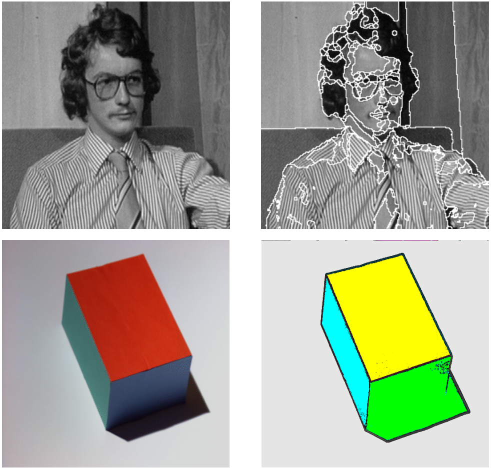
Figure: Isoperformance lines for the F metric ($F = 2\frac{\text{P}\times\text{R}}{\text{P}+\text{R}} = \frac{2\text{TP}}{2\text{TP}+\text{FP}+\text{FN}}$).

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
   - General discussion
   - Evaluation criteria and metrics
   - Illustrations
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Evaluation of segmentation quality I

Problem statement



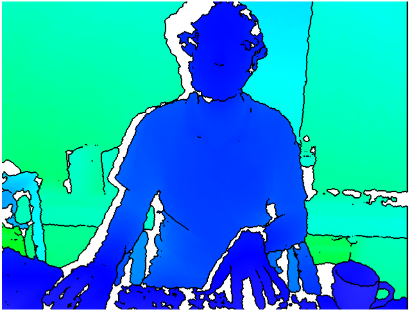
Evaluation of segmentation quality II

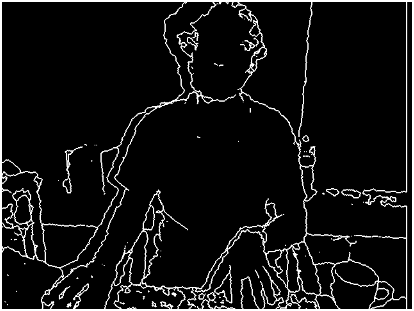
Practice:

- Does not nicely fit into the framework of classification
- Some difficulties:
  - *individual* object segmentation quality vs *overall* segmentation quality evaluation
  - subjectivity for defining the ground truth
- Two major classes of metrics for **standalone** segmentation quality evaluation
  - intra-object homogeneity
  - inter-object disparity
- Metrics for **relative** segmentation quality evaluation
  - spatial accuracy
  - temporal accuracy

Evaluation of edge detection I

Problem statement





Evaluation of edge detection II

Practice:

- Some evaluation methods rely on the classification theory.
- Use of the Precision/Recall evaluation space.
- The F score is a trade-off (harmonic mean between P and R): $F = \frac{2}{\frac{1}{\mathrm{P}}+\frac{1}{\mathrm{R}}} = 2\frac{\mathrm{P}\times\mathrm{R}}{\mathrm{P}+\mathrm{R}}$

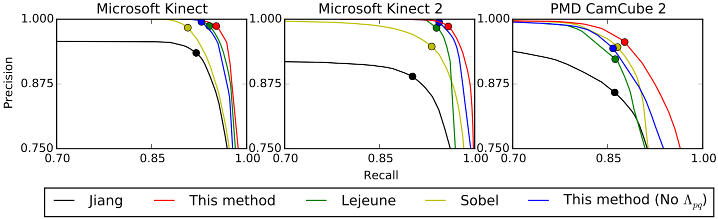
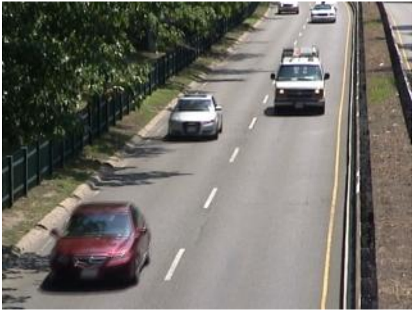
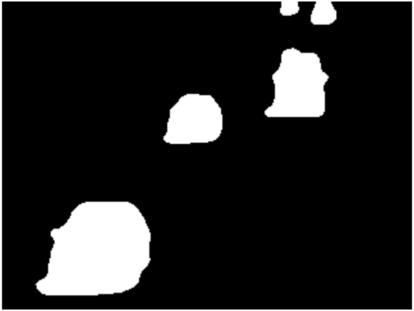


Figure: Maximum Precision and Recall curve obtained by varying the parameter set. Circles indicate where the F measure is maximal.

Evaluation of background subtraction I

Problem statement





Evaluation of background subtraction II

Practice:

- Many dataset with ground truth available.
- Background subtraction is often considered as a classification task.
- Many algorithms available ⇒ most people try to *rank* algorithms.
- Many metrics. No agreement over the best one (if it exists). Therefore, an algorithm is evaluated with respect to multiple metrics.
- It is still hard to evaluate the dynamic ($\equiv$ over time) behavior. Usually, there is one score for the whole video sequence.
- The role of priors is underestimated or just ignored.

Problems with the ChangeDetection dataset I

Reminder: main characteristics

- 11 categories: about 5 videos per category
- long video sequences
- thousands of annotated images (ground truth data)
  - 5 labels: *static* ($\equiv$ background), *shadow*, *non-ROI*, *unknown*, and *moving* ($\equiv$ foreground)
- 7 performance metrics are computed
- ranking per video, category, and globally

Are there problems?

Problems with the ChangeDetection dataset II

- About the content:
  - Some categories are irrelevant for most targeted applications.
  - Some hand-made annotations are debatable. For example, how do you handle ghosts or static objects? How do you define the shadow?
- About the methodology:
  - All the data is available ⇒ *it is possible to* **fine tune all the parameters**.
  - Parameters of methods are supposed to be the same for all videos. How do we treat methods that adaptively tune parameters?
  - Problem with the ranking methodology.
  - The averaging process for the metrics (per category and globally) is incorrect.
  - No source code available for some methods ⇒ some results are impossible to check.
  - Some metrics are redundant

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

- Linear operators
- Non-linear operators
- Hough's transform

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

What's a border/contour/edge? Towards a definition

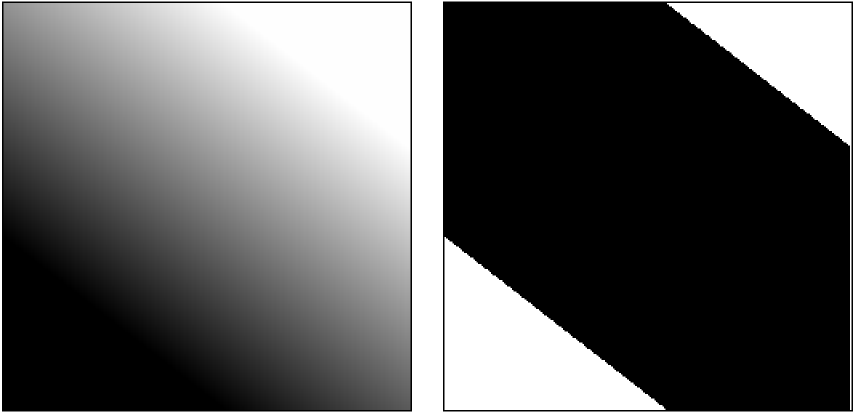
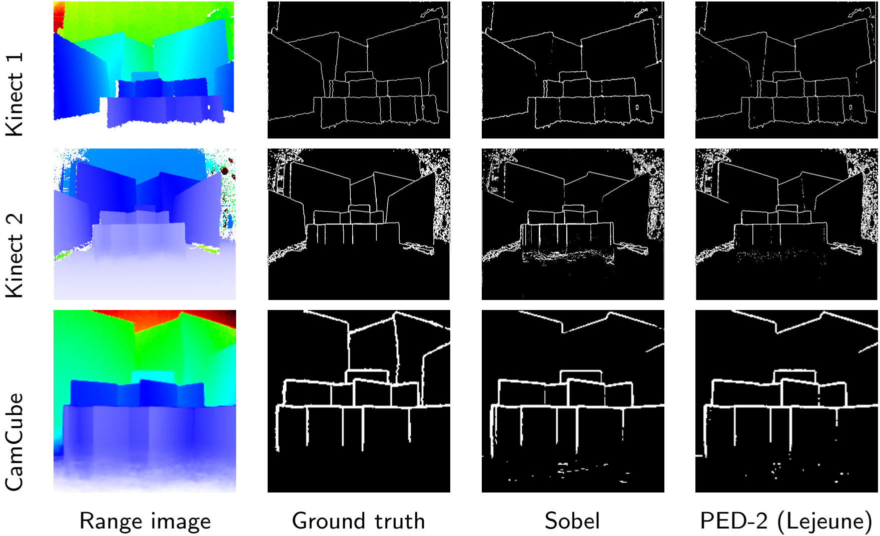


Figure: An image (diagonal gradient) and its contours (in black).

(physical) Edges in depth images



Kinect 1

Kinect 2

CamCube

Range image

Ground truth

Sobel

PED-2 (Lejeune)

Can we locate edge points?

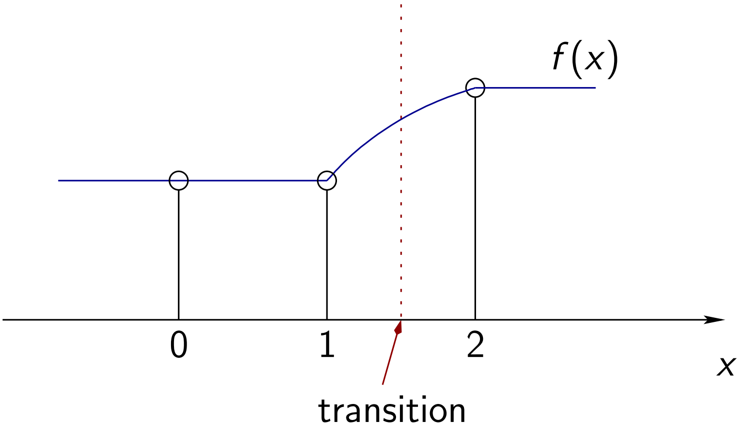


Figure: Problem encountered in locating an edge point.

Border/contour/edge detection

Outline

1. Linear operators
   - First derivate operators
   - Second derivate operators
   - Sampling the derivate
     - Residual error
     - Synthesis of operators for a fixed error
   - Practical expressions of gradient operators and convolution masks
2. Non-linear operators
   - Morphological gradients
3. Hough's transform

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection
- Linear operators
- Non-linear operators
- Hough's transform

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

Linear operators I

For derivate operators, we have to address two problems simultaneously:

1. find the best **approximate** for the derivate.
2. **avoid** an excessive **amplification** of the **noise**.

These are two apparent contradictory requirements ⇒ *trade-offs*

Linear operators II

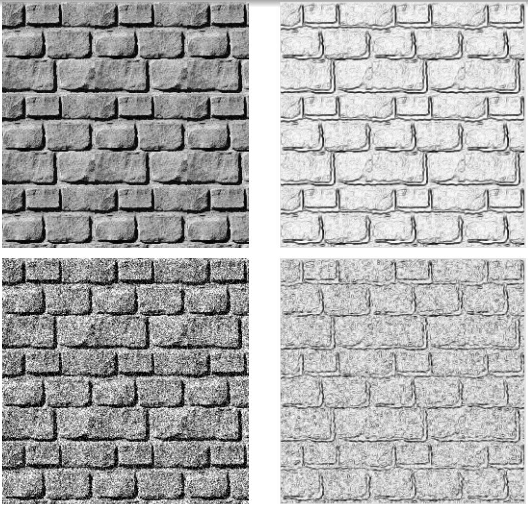
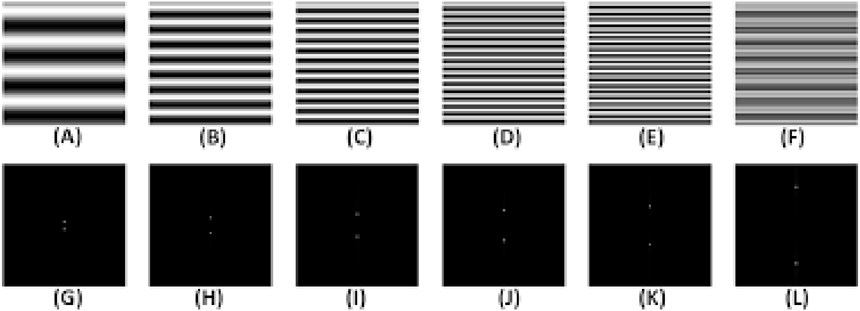
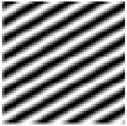


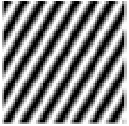
Figure: Images (left-hand side) and gradient images (right-hand side)

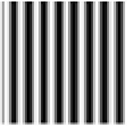
Fourier transform (quick reminder) I

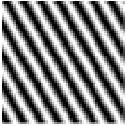


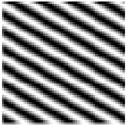
Fourier transform (quick reminder) II

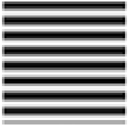

(A)

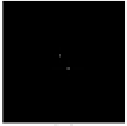

(B)

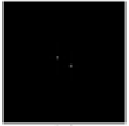

(C)

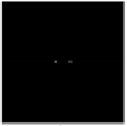

(D)

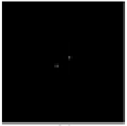

(E)

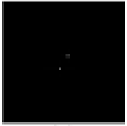

(F)

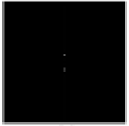

(G)


(H)

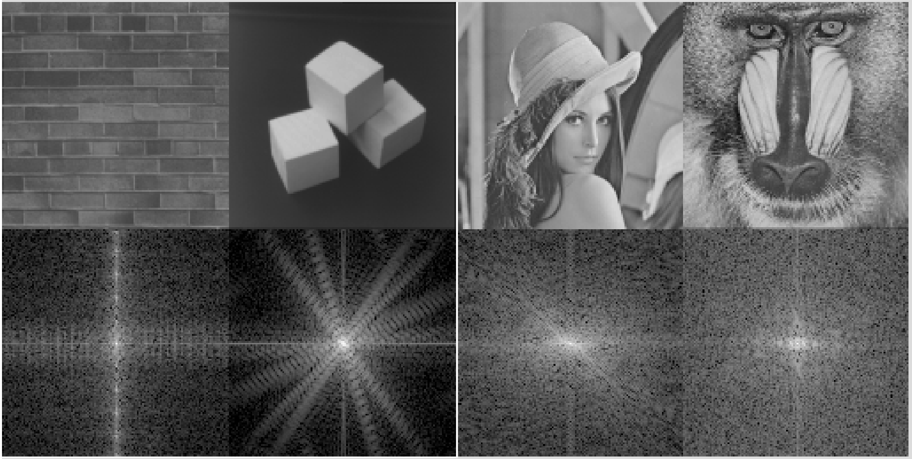

(I)


(J)


(K)


(L)

Fourier transform (quick reminder) III



First derivate operator I

Let us consider the *partial* derivate of a function $f(x,y)$ with respect to x. Its Fourier transform $\mathcal{F}(u,v)$ becomes

$$\frac{\partial}{\partial x} f(x,y) \rightleftharpoons 2\pi j u\, \mathcal{F}(u,v) \tag{162}$$

In other words, deriving with respect to x consists of multiplying the Fourier transform of $f(x,y)$ by the following transfer function $\mathcal{H}_x(u,v) = 2\pi j u$, or of filtering $f(x,y)$ with the following impulse function:

$$h_x(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (2\pi j u)\, e^{2\pi j(xu+yv)} du dv \tag{163}$$

If we adopt a *vectorial notation* of the derivate, we define the *gradient* ∇f of image f by

$$\nabla f = \frac{\partial f}{\partial x}\overrightarrow{e_x} + \frac{\partial f}{\partial y}\overrightarrow{e_y} = (h_x \otimes f)\overrightarrow{e_x} + (h_y \otimes f)\overrightarrow{e_y} \tag{164}$$

First derivate operator II

Definition (Gradient amplitude)

$$|\nabla f| = \sqrt{(h_x \otimes f)^2 + (h_y \otimes f)^2} \tag{165}$$

The amplitude of the gradient is sometimes approximated by

$$|\nabla f| \simeq |h_x \otimes f| + |h_y \otimes f| \tag{166}$$

which introduces a still acceptable error (in most cases) of 41%!
But we still have to find a way to calculate $h_x \otimes f$ and $h_y \otimes f$ ...

Definition (Gradient orientation)

$$\varphi_{\nabla f} = \tan^{-1}\left(\frac{h_y \otimes f}{h_x \otimes f}\right) \tag{167}$$

Second derivate operator

Definition (Laplacian $\nabla^2 f$)

The Laplacian is the scalar defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = (h_{xx} \otimes f) + (h_{yy} \otimes f) \tag{168}$$

As the first derivate, it can be shown that in the Fourier domain, the Laplacian consists to apply the following filter

$$\begin{aligned} \nabla^2 f &\rightleftharpoons \left((2\pi j u)^2 + (2\pi j v)^2\right) \mathcal{F}(u,v) && (169) \\ &\rightleftharpoons -4\pi^2(u^2+v^2)\mathcal{F}(u,v) && (170) \end{aligned}$$

As can be seen, high frequencies tend to be amplified. Is this suitable?

Sampling the gradient and residual error I

In order to derive practical expressions for the computation of a derivate, we adopt the following approach:

1. develop some approximations and derive the error due to the approximation,
2. study the spectral behavior of these approximations, and
3. discuss some practical approximations expressed in the terms of convolution masks.

Sampling the gradient and residual error II

Centered approximations (along a single axis!, for example a line of the image)?

First derivate: an approximation is

$$f_a'(x) = \frac{f(x+h) - f(x-h)}{2h} = \frac{(+1)f(x+h) + (-1)f(x-h)}{2h} \quad (171)$$

where h is the distance between two samples and index a denotes that it is an approximation. Note that this approximation consists to filter $f(x)$ by the following "*multiplicative*" *mask*

$$\frac{1}{2h}\begin{bmatrix} -1 & 0 & +1 \end{bmatrix} \quad (172)$$

or the *convolution mask*

$$\frac{1}{2h}\begin{bmatrix} +1 & 0 & -1 \end{bmatrix} \quad (173)$$

Sampling the gradient and residual error III

Second derivate: one possible approximation is

$$f''_a(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{174}$$

Sampling the gradient and residual error IV

We make use of the following representation of a function.

Theorem (Taylor series)

The Taylor *series of a real valued function $f(z)$ that is infinitely differentiable at a real number a is the following power series*

$$f(z) = f(a) + \frac{f'(a)}{1!}(z-a) + \frac{f''(a)}{2!}(z-a)^2 + \dots \tag{175}$$

Then, we substitute:

- location a by location x
- location z by either $x+h$ or $x-h$

Sampling the gradient and residual error V

Computation of the residual error. Let's consider the following Taylor series

$$\begin{aligned} f(x+h) &= f(x) + h\,f'(x) + \frac{h^2}{2}\,f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + \ldots \quad (176) \\ f(x-h) &= f(x) - h\,f'(x) + \frac{h^2}{2}\,f''(x) + \ldots + (-1)^n \frac{h^n}{n!} f^{(n)}(x) + \ldots \end{aligned}$$

First derivate. By subtraction, member by member, these two equalities, one obtains

$$f(x+h) - f(x-h) = 2h\,f'(x) + \frac{2}{3!} h^3 f^{(3)}(x) + \ldots = 2h\,f'(x) + O(h^3) \quad (177)$$

After re-ordering,

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (178)$$

Sampling the gradient and residual error VI

Second derivate.

$$\begin{aligned} f(x+h) &= f(x) + h\,f'(x) + \frac{h^2}{2} f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + \ldots & (179) \\ f(x-h) &= f(x) - h\,f'(x) + \frac{h^2}{2} f''(x) + \ldots + (-1)^n \frac{h^n}{n!} f^{(n)}(x) + \ldots & (180) \end{aligned}$$

Like for the first derivate, we use the Taylor extension by add them this time,

$$f(x+h)+f(x-h) = 2f(x) + h^2 f''(x) + \frac{2}{4!} h^4 f^{(4)}(x) + \ldots \qquad (181)$$

As a result:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \qquad (182)$$

The $f''_a(x)$ approximation is also of the second order in h.

Sampling the gradient and residual error VII

Synthesis of expressions with a pre-defined error.

Another approximation, of order $O(h^4)$, can be built.

It corresponds to

$$f_a'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} \tag{183}$$

Spectral behavior of discrete gradient operators I

Consider the one-dimensional continuous function $f(x)$ and the following first derivate:

$$f'_a(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{184}$$

Its Fourier is given by

$$\frac{f(x+h) - f(x-h)}{2h} \rightleftharpoons \frac{e^{2\pi j u h} - e^{-2\pi j u h}}{2h} \mathcal{F}(u) \tag{185}$$

which is, given that $\sin(a) = \frac{e^{ja} - e^{-ja}}{2j} \Leftrightarrow e^{2\pi j u h} - e^{-2\pi j u h} = 2j \sin(2\pi h u)$,

$$\frac{f(x+h) - f(x-h)}{2h} \rightleftharpoons (2\pi j u)\ \frac{\sin(2\pi h u)}{2\pi h u}\ \mathcal{F}(u) \tag{186}$$

where the $(2\pi j u)$ factor corresponds to the ideal (continuous) expression of the first derivate.

Spectral behavior of discrete gradient operators II

Let us now consider the approximation of the second derivate

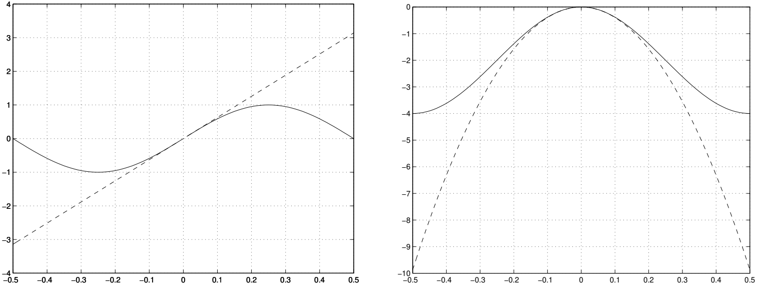
$$f_a''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{187}$$

Its Fourier transform is given by

$$\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \rightleftharpoons (-4\pi^2 u^2) \left(\frac{\sin(\pi h u)}{(\pi h u)}\right)^2 \mathcal{F}(u) \tag{188}$$

Spectral behavior of discrete gradient operators III

Figure: Spectral behavior of the derivate approximations (for $h = 1$).



	first derivate	second derivate
ideal	$2\pi j u$	$-4\pi^2 u^2$
approx.	$(2\pi j u)\ \frac{\sin(2\pi h u)}{2\pi h u}$	$(-4\pi^2 u^2)\ \left(\frac{\sin(\pi h u)}{(\pi h u)}\right)^2$

Practical expressions of gradient operators and convolution/multiplication masks I

Practical expression are based on the notion of convolution masks

$$\left[\begin{array}{cc} +1 & -1 \end{array}\right] \tag{189}$$

corresponds to the following non-centered approximation of the first derivate:

$$\frac{(-1)\times f(x,y)+(+1)\times f(x+h,y)}{h} \tag{190}$$

This "convolution mask" has an important drawback. Because it is not centered, the result is shifted by half a pixel. One usually prefers to use a centered (larger) convolution mask such as

$$\left[\begin{array}{ccc} +1 & 0 & -1 \end{array}\right] \tag{191}$$

Practical expressions of gradient operators and convolution/multiplication masks II

In the y (vertical) direction, this becomes

$$\begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} \tag{192}$$

But then, it is also possible to use a diagonal derivate:

$$\begin{bmatrix} +1 & . & . \\ . & 0 & . \\ . & . & -1 \end{bmatrix} \tag{193}$$

Practical expressions of gradient operators and convolution/multiplication masks III

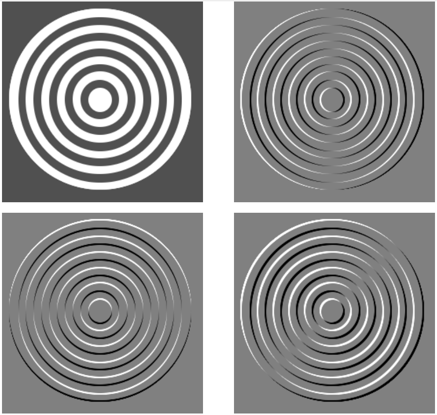


Figure: (a) original image, (b) after the application of a horizontal mask, (c) after the application of a vertical mask, and (d) mask oriented at 135^0.

Practical problems

The use of (centered) convolution masks still has some drawbacks:

- **Border effects**.
  Solutions:
  - (i) put a *default* value *outside* the image;
  - (ii) *mirroring extension*: copy inside values starting from the border;
  - (iii) *periodization* of the image –pixels locate on the left are copied on the right of the image,
  - (iv) *copy* border values to fill an artificial added border.
- **The range (dynamic) of the possible values is modified.**
- **It might be needed to apply a normalization factor**.

Prewitt gradient filters

$$[h_x] = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \tag{194}$$

$$[h_y] = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{195}$$

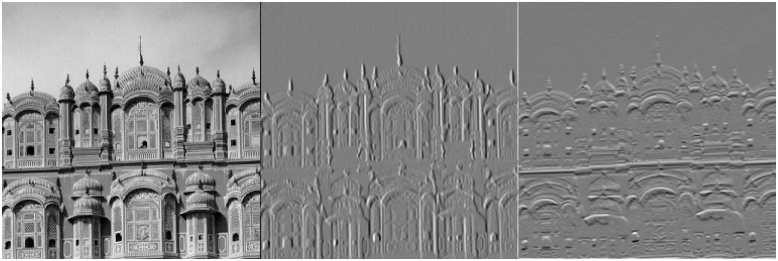


Figure: Original image, and images filtered with a horizontal and vertical Prewitt filter respectively.

Sobel gradient filters

$$[h_x] = \frac{1}{4}\left[\begin{array}{ccc} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{array}\right] = \frac{1}{4}\left[\begin{array}{c} 1 \\ 2 \\ 1 \end{array}\right]\left[\begin{array}{ccc} 1 & 0 & -1 \end{array}\right] \tag{196}$$

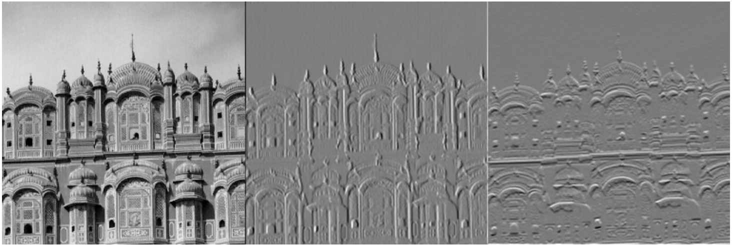
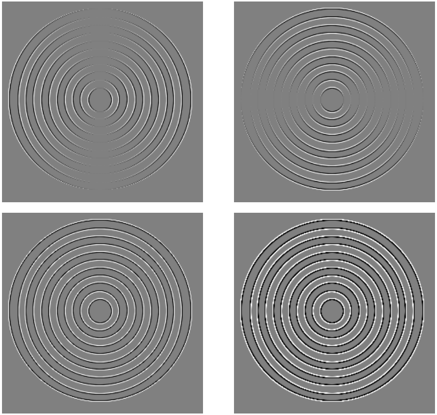


Figure: Original image, and images filtered with a horizontal and vertical Sobel filter respectively.

Second derivate: basic filter expressions

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
   - Linear operators
   - Non-linear operators
   - Hough's transform
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Non-linear operators I

Morphological gradients

- *Erosion gradient* operator:

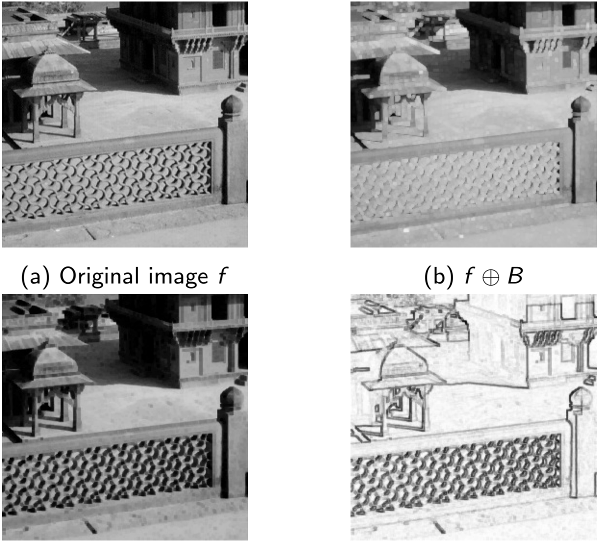
$$GE(f) = f - (f \ominus B) \tag{197}$$

- *Dilation gradient* operator:

$$GD(f) = (f \oplus B) - f \tag{198}$$

- *Morphological gradient* of Beucher: $GE(f) + GD(f)$.
- *Top-hat* operator: $f - f \circ B$;
- min/max gradient operators: $min(GE(f), GD(f)), max(GE(f), GD(f))$
- Non-linear Laplacian: $GD(f) - GE(f)$.

Gradient of Beucher



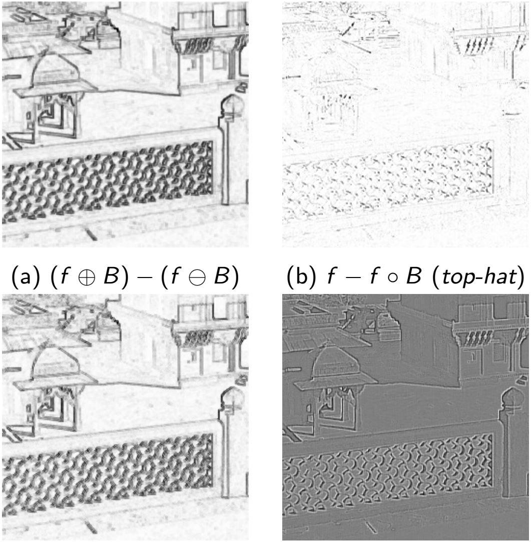
(a) Original image f

(b) $f \oplus B$

(c) $f \ominus B$

(d) $(f \oplus B) - (f \ominus B)$ (inverse video)

Different non-linear border detectors



(a) $(f \oplus B) - (f \ominus B)$

(b) $f - f \circ B$ (*top-hat*)

(c) $max(GE(f), GD(f))$

(d) $GD(f) - GE(f)$

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
   - Linear operators
   - Non-linear operators
   - Hough's transform
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

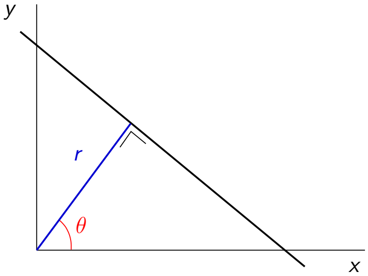
Detecting lines

Challenge: detect lines in an image

Towards the Hough transform

- Difficulty: matching a set of points arranged as a line
- Idea: instead of considering the family of points (x, y) that belong to a line $y = ax + b$, consider the two parameters
  1. the slope parameter a (but a is unbounded for vertical lines)
  2. the intercept parameter b (that is for $x = 0$)

Definition of the Hough transform I



With the Hough transform, we consider the (r, θ) pair where

- the parameter r represents the distance between the line and the origin,
- while θ is the angle of the vector from the origin to the line closest point

Definition of the Hough transform II

We have several ways to characterize a line:

1. Slope a and b, such that $y = ax + b$.
2. The two parameters (r, θ), with $\theta \in [0, 2\pi[$ and $r \geq 0$.

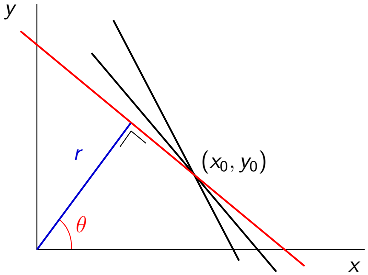
Link between these characterizations:
The equation of the line becomes

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right) \tag{199}$$

Check:

- For $x = 0$, $r = y\sin\theta \rightarrow$ ok.
- For $x = r\cos\theta$, $y = r\sin\theta \rightarrow$ ok.

Families of lines passing through a given point (x_0, y_0)

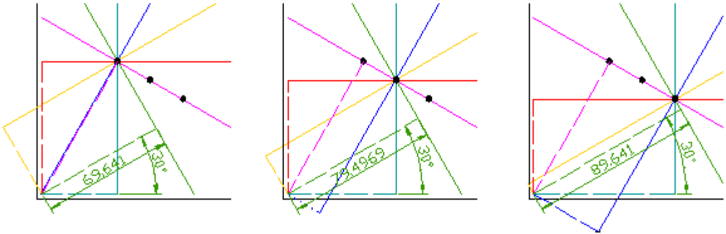


By re-arranging terms of $y = \left(-\frac{\cos\theta}{\sin\theta}\right) x + \left(\frac{r}{\sin\theta}\right)$, we get that, for an arbitrary point on the image plane with coordinates, e.g., (x_0, y_0), the family of lines passing through it are given by

$$r = x_0 \cos\theta + y_0 \sin\theta \tag{200}$$

Example

For three points (x_0, y_0), we explore the Hough space.
That is, we compute r for a given set of orientations θ:

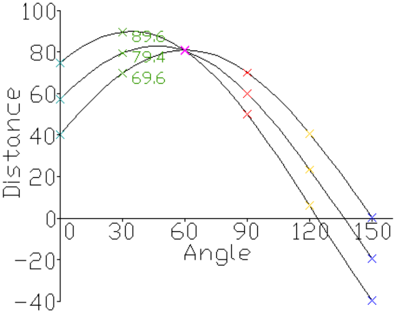


θ	r
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

θ	r
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	19.5

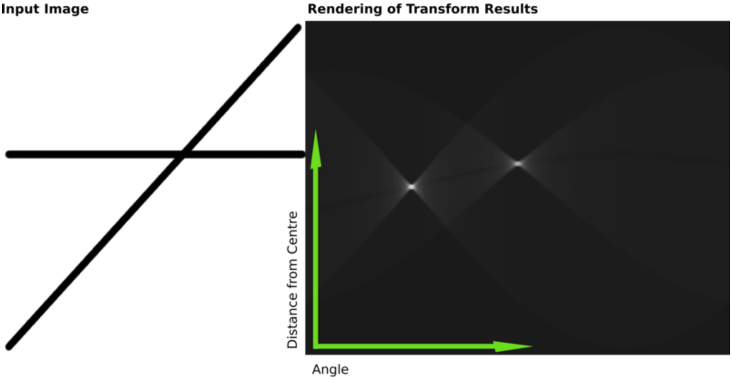
θ	r
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	39.6

Hough space



Thus, the problem of detecting colinear points can be converted to the problem of finding concurrent curves.

Hough space


Input Image
Rendering of Transform Results
Distance from Centre
Angle

Algorithm for detecting lines

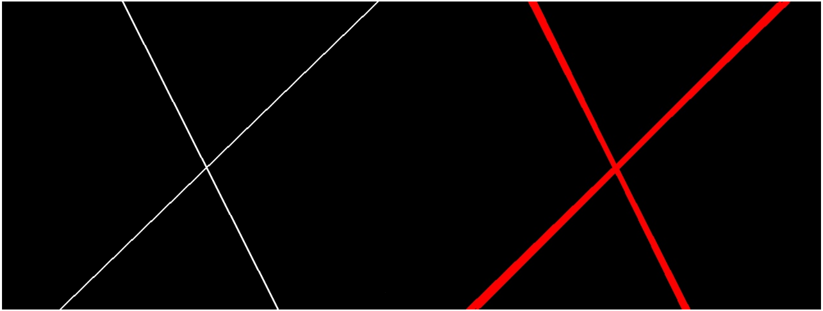
Algorithm

1. Detect edges in the original image.
2. Select only some pixels corresponding to "strong edges" for which there is enough evidence that they belong to lines.
3. For each selected pixel, accumulate values in the corresponding bins of the Hough space:
    1. For example, we take each θ in $[0^0, 360^0]$ with a step of 1^0.
    2. We calculate the corresponding r by $r = x_0 \cos\theta + y_0 \sin\theta$.
    3. If $\text{Hough}(r, \theta)$ is one bin of the Hough space, then we do

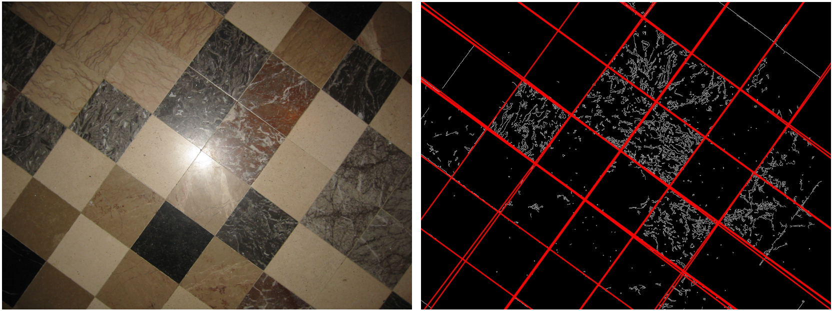
$$\text{Hough}(r, \theta) \longleftarrow \text{Hough}(r, \theta) + 1 \tag{201}$$

4. Threshold the accumulator function $\text{Hough}(r, \theta)$ to select bins that correspond to lines in the original image.
5. Draw the corresponding lines in the original image.

"Toy" example



Real example



Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
    - Introduction
    - Feature detection
    - Tracking: Kalman filtering
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Outline

1 Fundamentals of 2D imaging
2 Linear filtering
3 3D vision: calibration and reconstruction
4 Mathematical morphology
5 Non-linear filtering
6 Object description and analysis
7 Motion analysis and background subtraction
8 Performance analysis
9 Edge detection
10 Feature detection and tracking
- Introduction
- Feature detection
- Tracking: Kalman filtering

11 Introduction to machine learning
12 Fundamentals of deep learning for visual understanding
13 Deep learning architectures for image analysis
14 Segmentation
15 Template Matching & Image Registration

Introduction

- Features are edges, corners, etc.
  Most of them are based on derivatives.
- The choice of features is application-dependent!
- For tracking:
  - We need matching measures and metrics to find the best matching locations between frames.
  - Features need to be “strong”: resilient to illumination changes, to local deformations, etc.

Historical example: the Laplacian of Gaussians I

Principle:

- first apply a Gaussian filter (with the purpose of reducing noise).
- compute the Laplacian of the filtered image.
- detect the locations of zero-crossings in the image.

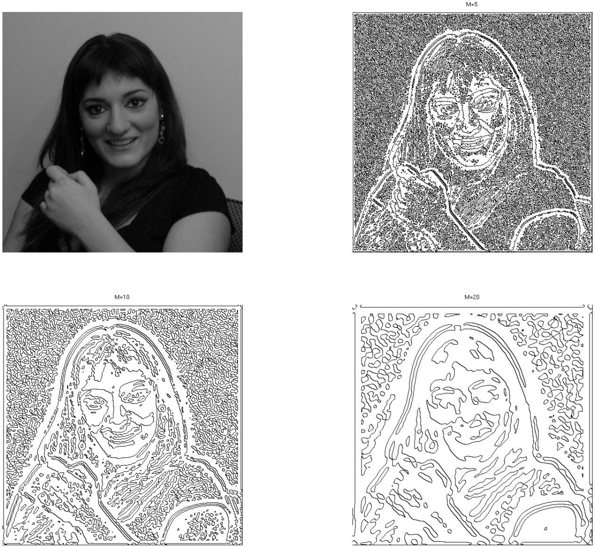
Parameters:

- variance of the Gaussian filter

Result:

- collection of images with features detected at different scales.
- not suitable for tracking.

Historical example: the Laplacian of Gaussians II



Feature-based matching

Steps:

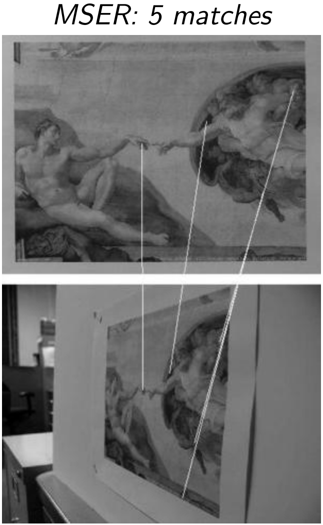
1. select a set of robust features in the reference frame (or sub-frame/Region of Interest [ROI])
2. compute the same set of features in the target image
3. establish the relationship between the two sets of features. Many strategies are possible:
   - global optimization
   - local optimization
   - RANSAC algorithm

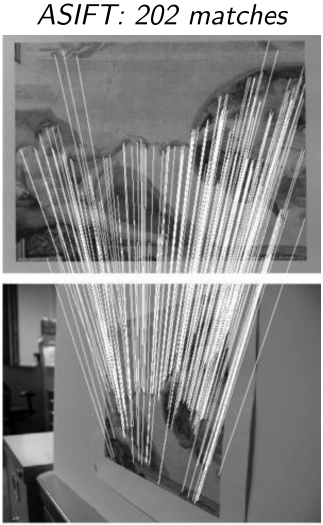
Feature detectors I

Classification [source: wikipedia]

Feature	Edge	Corner	Blob
Sobel	X		
Canny	X		
Harris	X	X	
Laplacian of Gaussians		X	X
Difference of Gaussians		X	X
FAST		X	X
MSER			X

Feature detectors II





Outline

1 Fundamentals of 2D imaging
2 Linear filtering
3 3D vision: calibration and reconstruction
4 Mathematical morphology
5 Non-linear filtering
6 Object description and analysis
7 Motion analysis and background subtraction
8 Performance analysis
9 Edge detection
10 Feature detection and tracking
- Introduction
- Feature detection
- Tracking: Kalman filtering

11 Introduction to machine learning
12 Fundamentals of deep learning for visual understanding
13 Deep learning architectures for image analysis
14 Segmentation
15 Template Matching & Image Registration

Harris detector (for detecting corners) I

Let p be a pixel with the (x, y) coordinates. We compute

$$M = \sigma_D^2 g(\sigma_I) \otimes \left[\begin{array}{cc} I_x^2(p, \sigma_D) & I_x(p, \sigma_D) I_y(p, \sigma_D) \\ I_x(p, \sigma_D) I_y(p, \sigma_D) & I_y^2(p, \sigma_D) \end{array} \right] \tag{202}$$

where

$$I_x(p, \sigma_D) = \frac{\partial}{\partial x} g(\sigma_D) \otimes I(p) \tag{203}$$

$$g(\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{204}$$

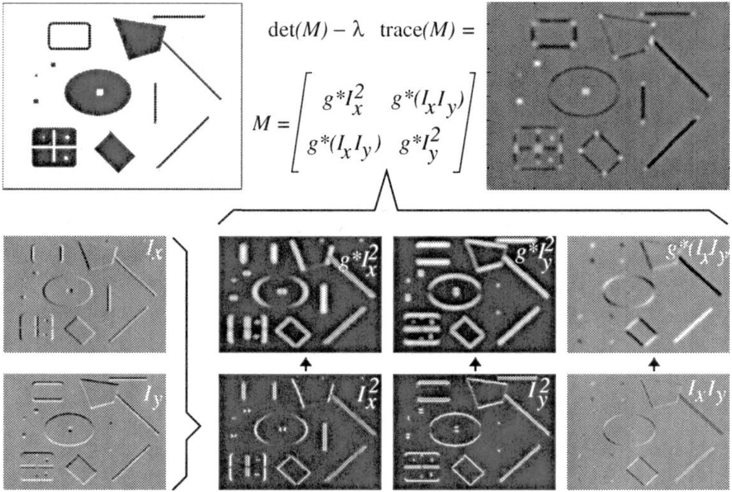
Definition (Cornerness)

Cornerness is defined as

$$\mathbf{det}(M) - \lambda\, \mathbf{trace}^2(M) \tag{205}$$

where λ is a tunable sensitivity parameter.

Harris detector (for detecting corners) II



$$\det(M) - \lambda \ \operatorname{trace}(M) =$$

$$M = \begin{bmatrix} g * I_x^2 & g*(I_x I_y) \\ g*(I_x I_y) & g*I_y^2 \end{bmatrix}$$

Other feature detectors I

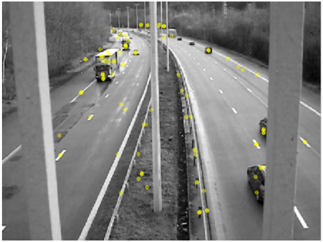
The calculation of features combines several steps and usually includes a scale space and derivates.
Some variants:

- MSER: Maximally Stable Extremal Region extractor (2002)
- SIFT: Scale-Invariant Feature Transform (2004)
- SURF: Speeded Up Robust Features (2006)
- FAST (2006)
- BRISK: Binary Robust Invariant Scalable Keypoints (2011)
- ORB: Oriented BRIEF (2011)
- FREAK: Fast Retina Keypoint (2012)
- Learned features (by deep learning): Superpoint (2018), ...

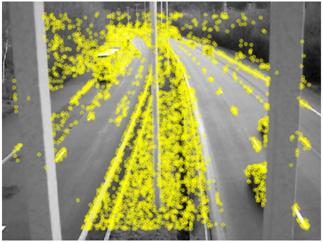
Don't forget that these methods have some **parameters**.

Other feature detectors II

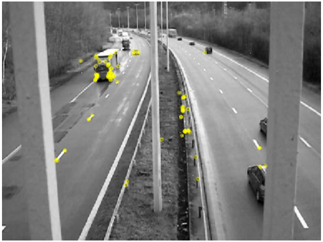
Original



MSER



FAST



ORB

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

- Introduction
- Feature detection
- Tracking: Kalman filtering

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

Approach

There is an object, characterized by some variables put in a state vector X, amongst which some variables might no be observable directly.

Steps:

1. We assume that we can trust a dynamic model of the state that allows us to predict how it evolves over time.
2. Periodically, we observe the scene and produce a measure Y, that is a function of the state of the system.
3. With this observation, we correct our estimate of the system state.

The tracking loop in practice (for one object)

1. Wait for a new image.
2. **Predict** the current state of the tracked object.
3. Use that prediction to delineate a region of interest.
4. Extract the tracked object in the new image.
5. Derive the object state from the previous step and use it to **correct** the estimate of the object state.

Independence assumptions

- The system state and the observation are considered as *random* vectors.
- We limit the model to a Markov model.
  If X_i defines the object state in frame i, then

$$p(X_i|X_{i-1}, X_{i-2}, \ldots) = p(X_i|X_{i-1}) \tag{206}$$

  In other words, only the immediate past matters.
- The measurement only depends on the current state (no memory):

$$p(Y_i|X_i, \ldots) = p(Y_i|X_i) \tag{207}$$

Assumptions

1. The dynamic model is linear.
2. Errors in the modeling of state variables are considered as additive Gaussian noise.
3. The measurement process is linear.
4. Noise on the measurement is characterized by an additive Gaussian noise.

Prediction and correction (probabilistic models)

Prediction of the state

Multiply the state by a matrix D, and add Gaussian noise:

$$x_t \sim p(X_t|X_{t-\Delta t}) = \mathcal{N}(D_t x_{t-\Delta t}; \Sigma_{Dt}) \tag{208}$$

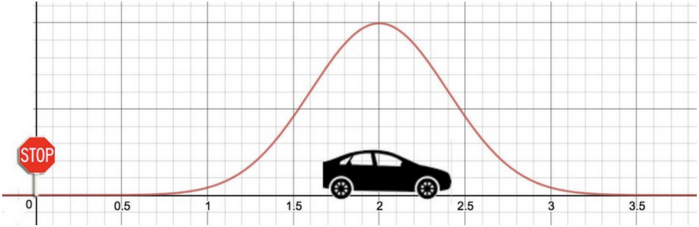
where Σ_{Dt} is the covariance matrix.

Measurement/Observation

Multiply the state by a matrix M, and add Gaussian noise:

$$y_t \sim p(Y_t|X_t) = \mathcal{N}(M_t x_t; \Sigma_{Mt}) \tag{209}$$

Example: first order model for the displacement of an object I



State vector

$$x = [x,\, y,\, v_x,\, v_y]^T \tag{210}$$

where $(v_x,\, v_y)$ are the components of the speed vector.

Example: first order model for the displacement of an object II

Model

We have that

$$x(t+\Delta t) = x(t) + v_x(t)\Delta t \tag{211}$$

$$\begin{bmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ v_x(t+\Delta t) \\ v_x(t+\Delta t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ v_x(t) \\ v_x(t) \end{bmatrix} \tag{212}$$

Matrix of the dynamic system (state-transition model):

$$D = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{213}$$

Example: first order model for the displacement of an object III

Measurement

$$y = [x,\ y]^T \tag{214}$$

Observation matrix: $M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

A linear, recursive estimator

Compute: x_t given $x_{t-\Delta t}$ and y_t

Object state and measurement models:

$$\begin{aligned} x_t &\sim \mathcal{N}(D_t x_{t-\Delta t}; \Sigma_{Dt}) \qquad (215) \\ y_t &\sim \mathcal{N}(M_t x_t; \Sigma_{Mt}) \qquad (216) \end{aligned}$$

Starting assumptions: $\overline{x}_0^-$ and Σ_0^- are known.

Under these assumptions (linear dynamics, linear measurement function, Gaussian noise), the Kalman filter is the optimal recursive estimator.

Kalman filtering equations

Prediction	
Predicted state estimate	$\overline{x}_t^- = D_t \overline{x}_{t-\Delta t}^+$
Predicted estimate covariance	$\Sigma_t^- = \Sigma_{Dt} + D_t \Sigma_{t-\Delta t}^+ D_t^T$
Correction/update	
Measurement residual	$r_t = y_t - M_t \overline{x}_t^-$
Residual covariance	$R_t = M_t \Sigma_t^- M_t^T + \Sigma_{Mt}$
Optimal Kalman gain	$K_t = \Sigma_t^- M_t^T \left(R_t\right)^{-1}$
Updated state estimate	$\overline{x}_t^+ = \overline{x}_t^- + K_t r_t$
Updated estimate covariance	$\Sigma_t^+ = (I - K_t M_t) \Sigma_t^-$

Limitations

- We are not using future measurements to refine past state estimate.
  Solution: forwards-backwards Kalman filtering.
- Non-linear dynamic and measurement models.
  Solutions:
  - Extended Kalman Filter (EKF): local linearization representation.
  - Unscented Kalman filter: non-linear propagation of the mean and covariance using the unscented transform.

How to deal with missing or multiple measurements?

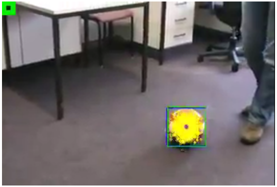
Missing measurements. Skip the correction step.

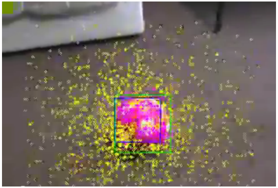
Multiple measurements. Choose the closest measurement.

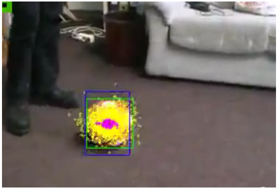
Multiple measurements and multiple targets. Find the best global target assignment by computing a distance matrix.

The choice of the (generally probabilistic) distance metric is crucial in this data association problem.

Alternative to Kalman filter: particle filter



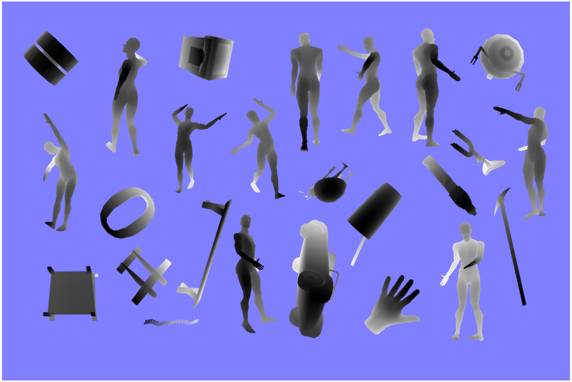




Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
    - Introduction to machine learning (ML)
    - Understanding classification
    - Conclusion
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

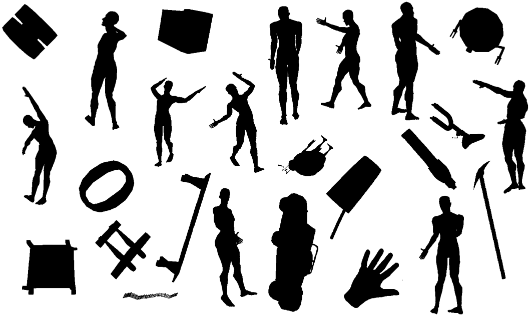
A starter: silhouette recognition I



- Can you decide which silhouettes are those of humans?
- Try to write an algorithm to solve this problem!

A starter: silhouette recognition II

After binarization of the silhouettes



- Can you decide which silhouettes are those of humans?
- Try to write an algorithm to solve this problem!

Very first understanding

Observation

- Most of the tasks related to video scene interpretation are complex.
- A human expert can easily take the right decision, but usually without being able to explain how he does it.

One possible solution

Use machine learning techniques that have proven to be a powerful tool in computer science and vision

Outline

1 Fundamentals of 2D imaging
2 Linear filtering
3 3D vision: calibration and reconstruction
4 Mathematical morphology
5 Non-linear filtering
6 Object description and analysis
7 Motion analysis and background subtraction
8 Performance analysis
9 Edge detection
10 Feature detection and tracking
11 Introduction to machine learning
- Introduction to machine learning (ML)
- Understanding classification
- Conclusion

12 Fundamentals of deep learning for visual understanding
13 Deep learning architectures for image analysis
14 Segmentation
15 Template Matching & Image Registration

Machine learning techniques

Machine learning (ML) techniques[2] aim is to

- build a decision rule automatically.
- speed up the decisions.
- be able to generalize to unseen objects. Really?!

Computational cost:

- The model is learned only once.
- The model is used many times.
- **[Q]** Which operation should be the fastest?

[2] We consider only "supervised" machine learning techniques.

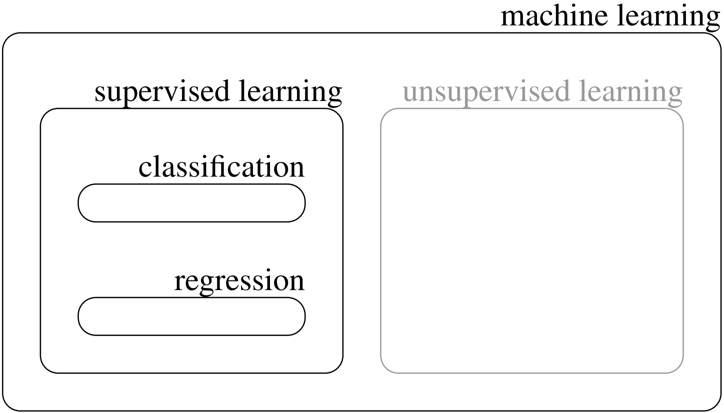
Machine learning techniques

Examples of techniques are:

- Naive Bayes classifier
- Nearest neighbor
- Artificial neuronal networks (→ deep learning)
- *Support Vector Machines* (SVM) [CV95]
- Random forests (*ExtRaTrees* [GEW06])

A good reference book on this topic is [HTF09].

Families of machine learning methods



For supervised learning, we have labeled (annotated) training data.

Classification *vs* regression

Example of classification:

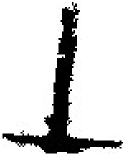
yes

yes

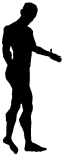
no

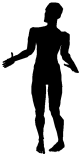
yes

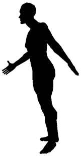
no

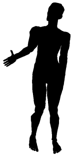

no

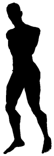
Example of regression:

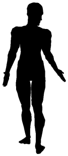

65.2^0


-2.0^0


-71.5^0


15.4^0


-47.4^0


-5.5^0

Applications I

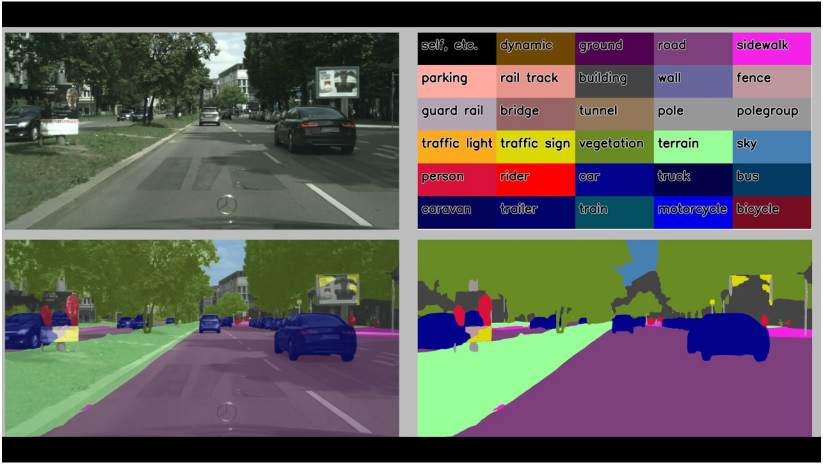
ML techniques have proven to be successful for many purposes:

- detecting people in images [DT05];
- recognizing people [BHP05];
- analyzing people's behavior [SFC^{+}11];
- detecting faces with software embedded in cameras [VJ04];

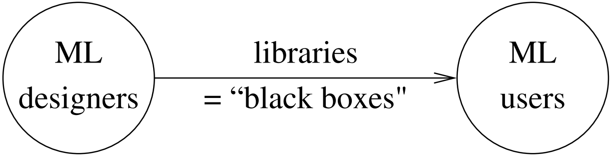
[image source: Shotton2011RealTime]

Applications II

- semantic segmentation
- etc



In practice . . .



There exists many machine learning libraries. For example,

- scikit-learn (Python)
- libSVM (Matlab, Java, Python, etc)
- Regression trees (C/Matlab)
- Java-ML (Java)
- Shark (C++)
- . . .

More specifically for deep learning,

- Caffe2, TensorFlow, Theano, Torch, Keras, CNTK (Python)

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
    - Introduction to machine learning (ML)
    - Understanding classification
    - Conclusion
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

How does it work? What is learned?

Example of learning database:

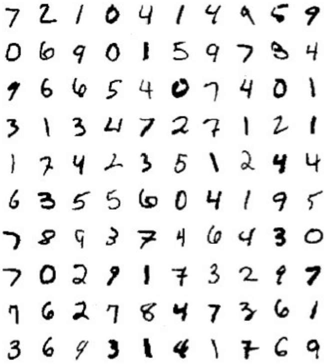
	x_1	x_2	x_3	x_4	x_5	y
sample 1	7.99	6.77	9.75	1.58	1.00	0
sample 2	2.24	9.51	1.14	8.00	7.66	0
sample 3	2.18	2.83	2.96	5.14	9.73	0
sample 4	8.44	7.39	4.57	4.94	2.70	1
sample 5	9.55	5.92	2.52	0.46	1.53	1
sample 6	3.32	9.13	0.50	5.07	8.22	2

$$MODEL \equiv y\left(x_1, x_2, x_3, x_4, x_5\right) = ?$$

- y is the output variable (the class)
- Samples are described by *attributes* (or *features*) x_1, x_2, ...
- The same number of attributes should be used for all samples.
- The meaning of an attribute should not depend on the sample.

Example of classification task

handwritten character recognition



- size = 100 samples
- choice : attributes = raw pixels
- the size of the images is 32 × 32
    - dimension = 1024 attributes

[image source: P. Geurts, "An introduction to Machine Learning"]

The intrinsic difficulty of machine learning

The theoretical rule to minimize the error rate is

$$y\left(\overrightarrow{x}\right) = \underset{y_i \in \{0,1,\ldots\}}{\arg\max} \left(p(y = y_i|\overrightarrow{x})\right) \tag{217}$$

Let ρ be the probability density function (pdf) of all objects in the attributes space, and ρ_i be the pdf of the objects belonging to class y_i. Using Bayes' rule (that is $p(A|B)p(B) = p(B|A)p(A)$):

$$p(y = y_i|\overrightarrow{x}) = \frac{\rho_i\left[\overrightarrow{x}\right]\, p(y = y_i)}{\rho\left[\overrightarrow{x}\right]} \tag{218}$$

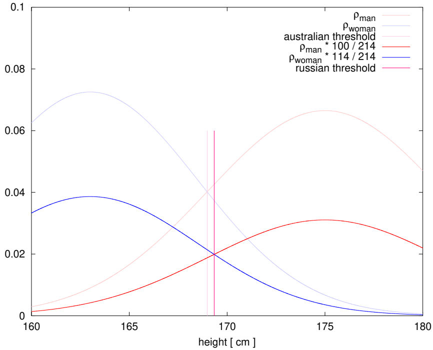
Therefore,

$$y\left(\overrightarrow{x}\right) = \underset{y_i \in \{0,1,\ldots\}}{\arg\max} \left(\rho_i\left[\overrightarrow{x}\right]\, p(y = y_i)\right) \tag{219}$$

The intrinsic difficulty is that it is very difficult to estimate ρ_i from the learning database because the space is not densely sampled.

An example of decision rule in 1D

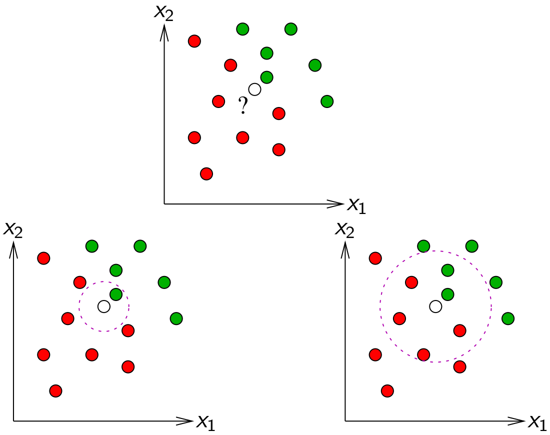
Let us assume that we have to recognize men and women based on a single attribute: the height. In Australia, there are 100 women for 100 men. But in Russia, there are 114 women for 100 men.



$$y = (\text{ height } < 169.34\ )?\ \text{"woman"} : \text{"man"}\ ;$$

Example of classifier: the nearest neighbors (NN) I

Let us consider a problem in 2 dimensions (2 attributes x_1, x_2):



1-NN

k-NN

Example of classifier: the nearest neighbors (NN) II

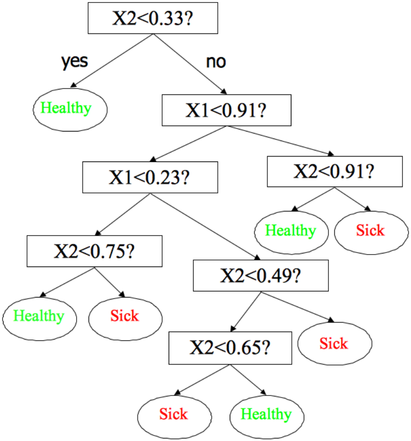
Advantages:

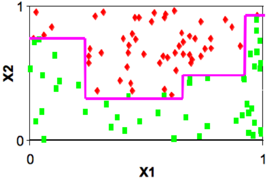
- The size of the neighborhood is automatically chosen depending on k.
- The model is the learning set (or a pruned version of it).

Drawbacks

1. The time needed to take a decision is $\mathcal{O}(n)$, where n is the learning set size.
2. Which distance measure should we select? There exists an infinity of possible choices! [DD09]

Example of classifier: the decision trees

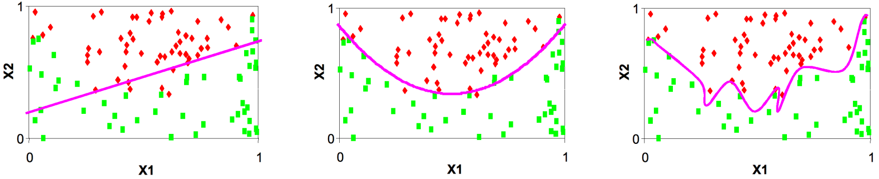




[image source: P. Geurts, "An introduction to Machine Learning"]

Choosing the complexity of the model

Which model is the best?



[image source: P. Geurts, "An introduction to Machine Learning"]

$$\mathit{error}(\mathrm{LS}) = 3.4\,\%$$
$$\mathit{error}(\mathrm{TS}) = 3.5\,\%$$

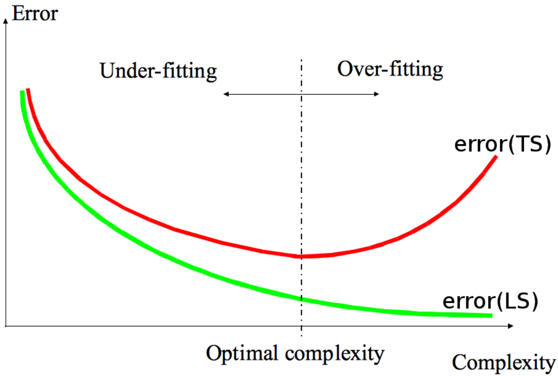
$$\mathit{error}(\mathrm{LS}) = 1.0\,\%$$
$$\mathit{error}(\mathrm{TS}) = 1.5\,\%$$

$$\mathit{error}(\mathrm{LS}) = 0.0\,\%$$
$$\mathit{error}(\mathrm{TS}) = 3.5\,\%$$

Two questions:

- Does the model explain the learning set (LS)?
  → *resubstitution error* = error estimated on the learning set
- Is the model able to predict the classes for unknown samples?
  → *generalization error* = error estimated on the test set (TS)

Choosing the complexity of the model



[image source: P. Geurts, "An introduction to Machine Learning"]

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
    - Introduction to machine learning (ML)
    - Understanding classification
    - Conclusion
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Conclusion

ML = automatic + generalization + preprocessing

Machine learning techniques are

- powerful methods.
- a complement to traditional methods.
- essential in computer science.
- adequate for real time computation.
- “easy” to use (not to engineer, nor to optimize [3]).

[3] This is why researchers are still working on machine learning methods.

Bibliography I

N. Boulgouris, D. Hatzinakos, and K. Plataniotis.
Gait recognition: a challenging signal processing technology for biometric identification.
IEEE Signal Process. Mag., 22(6):78–90, Nov. 2005.

C. Cortes and V. Vapnik.
Support-vector networks.
Mach. Learn., 20(3):273–297, Sept. 1995.

M. Deza and E. Deza.
Encyclopedia of Distances.
Springer, 2009.

N. Dalal and B. Triggs.
Histograms of oriented gradients for human detection.
In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, volume 1, pages 886–893, San Diego, CA, USA, Jun. 2005.

Bibliography II

P. Geurts, D. Ernst, and L. Wehenkel.
Extremely randomized trees.
Mach. Learn., 63(1):3–42, Apr. 2006.

T. Hastie, R. Tibshirani, and J. Friedman.
The elements of statistical learning: data mining, inference, and prediction.
Springer Series in Statistics. Springer, second edition, Sept. 2009.

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake.
Real-time human pose recognition in parts from single depth images.
In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1297–1304, Providence, RI, USA, Jun. 2011.

Bibliography III

Paul Viola and Michael J. Jones.
Robust real-time face detection.
Int. J. Comput. Vis., 57(2):137–154, May 2004.

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

- Motivation
- General view
- Deep learning building blocks
- Applications of deep learning with images

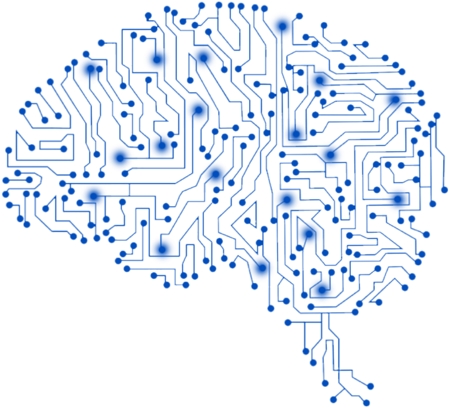
13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

Outline

1. Motivation
2. General aspects of deep learning
3. Deep learning building blocks for visual understanding
4. Applications of deep learning in computer vision



Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

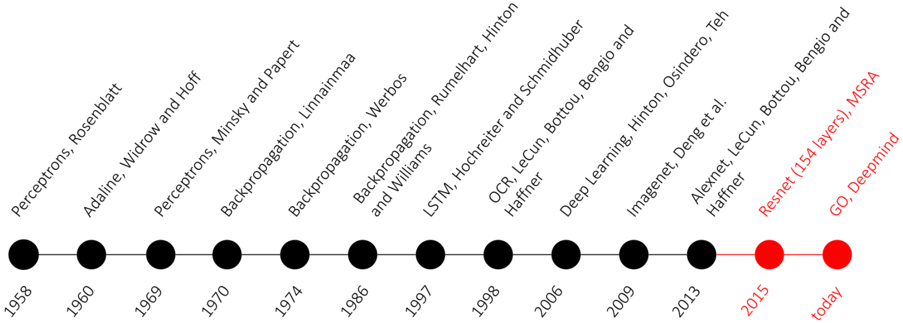
- Motivation
- General view
- Deep learning building blocks
- Applications of deep learning with images

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

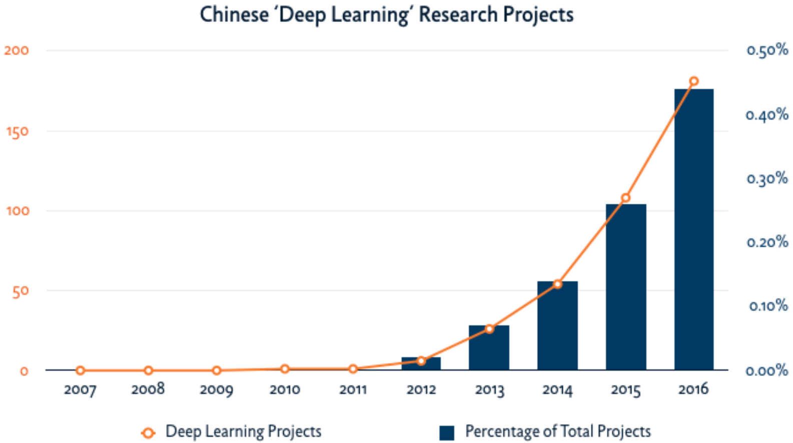
Deep learning research timeline



1958	Perceptrons, Rosenblatt
1960	Adaline, Widrow and Hoff
1969	Perceptrons, Minsky and Papert
1970	Backpropagation, Linnainmaa
1974	Backpropagation, Werbos
1986	Backpropagation, Rumelhart, Hinton and Williams
1997	LSTM, Hochreiter and Schmidhuber
1998	OCR, LeCun, Bottou, Bengio and Haffner
2006	Deep Learning, Hinton, Osindero, Teh
2009	Imagenet, Deng et al.
2013	Alexnet, LeCun, Bottou, Bengio and Haffner
2015	Resnet (154 layers), MSRA
today	GO, Deepmind

Figure: Timeline of the research in deep learning. The first principles, like the perceptron, were invented 60 years ago.

Evolution of the number of deep learning projects



Government Funded Research Projects. Source: Datenna

Why now and not before?

- Lack of processing power
  - No GPUs at the time.
  - Alexnet was published recently (in 2013).
- Lack of data
  - No big, annotated datasets at the time.
  - The Imagenet dataset was only published in 2009.
- Overfitting
  - Because of the above, models could not generalize all that well.

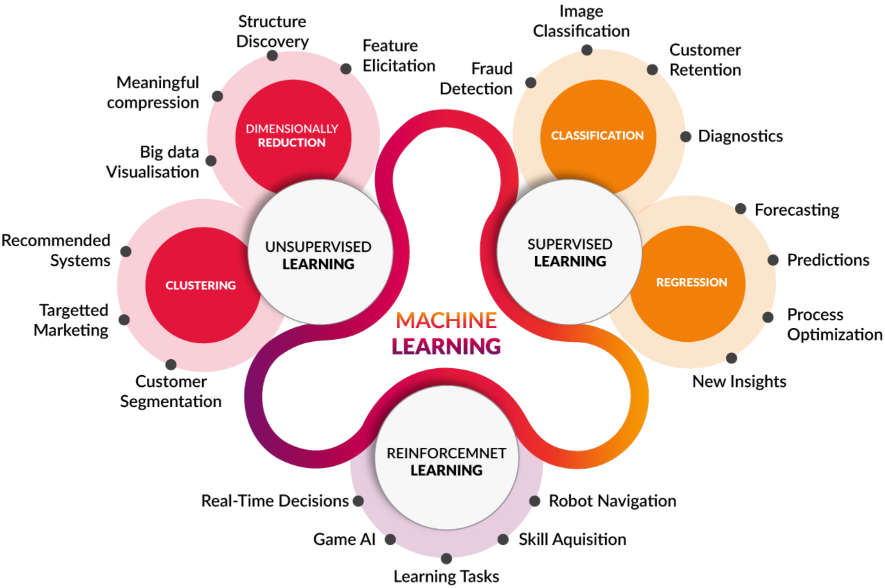
Deep learning today

- Deep Learning is used for many tasks
  - Object classification
  - *Object detection*, segmentation
  - Pose estimation
  - Image captioning
  - Question answering
  - Speech recognition
  - Robotics
- But some tasks remain difficult
  - Action classification
  - Action detection

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
    - Motivation
    - General view
    - Deep learning building blocks
    - Applications of deep learning with images
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

General aspects of machine learning



Basic principles of supervised learning

- The goal is to output the correct labels by optimizing on the θ parameters of the model.
- This is done by minimizing a loss function.
- This loss function takes different forms based on the typology of the problem.
- Update on the θ parameters is done using backpropagation of the error.

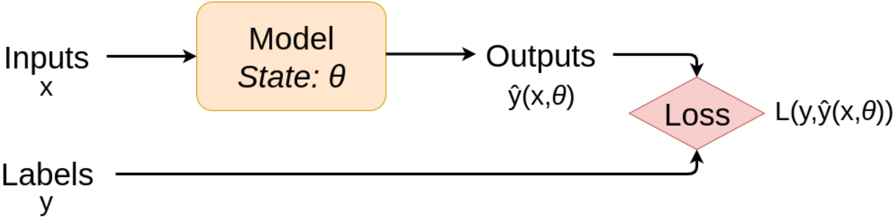


Figure: General supervised learning setup.

Mathematical formulation

Over a training set (X, Y), where X contains the inputs (x) and Y contains the labels (y), the goal is to find the optimal parameters θ^* of the model such that:

$$\theta^* \leftarrow \text{argmin}_\theta \sum_{(x,y)\in(X,Y)} L\left(y, \hat{y}(x, \theta)\right) \tag{220}$$

where $L(y, \hat{y}(x, \theta))$ is the loss function between the true label y and the predicted output $\hat{y}$.

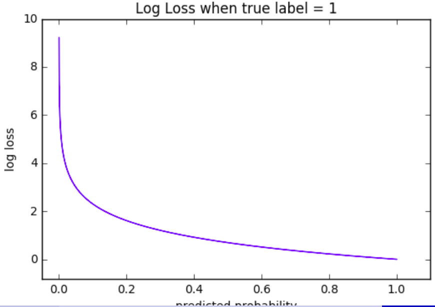
Some loss functions

- Mean squared error (MSE) for regression

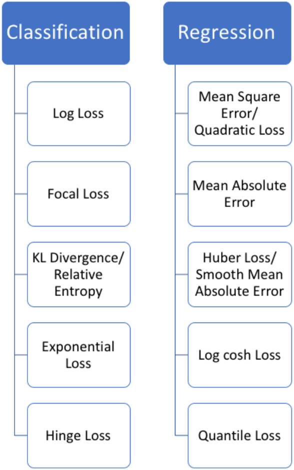
$$L(y,\hat{y}) = (y-\hat{y})^2 \tag{221}$$

- Cross entropy for binary classification

$$L(y,\hat{y}) = -y\log(\hat{y}) + (1-y)\log(1-\hat{y}) \tag{222}$$

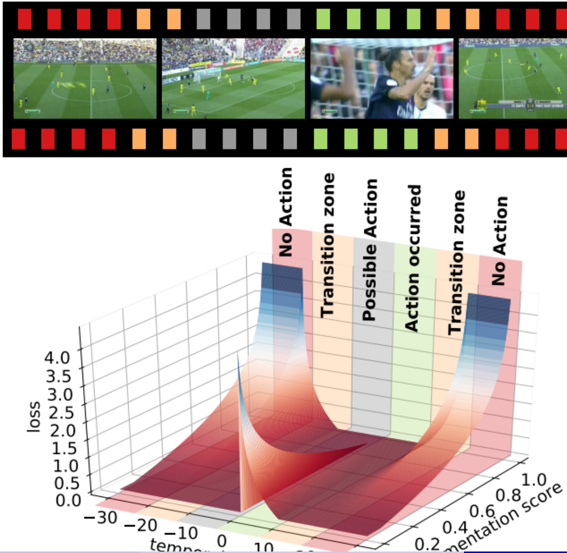


Some typical losses

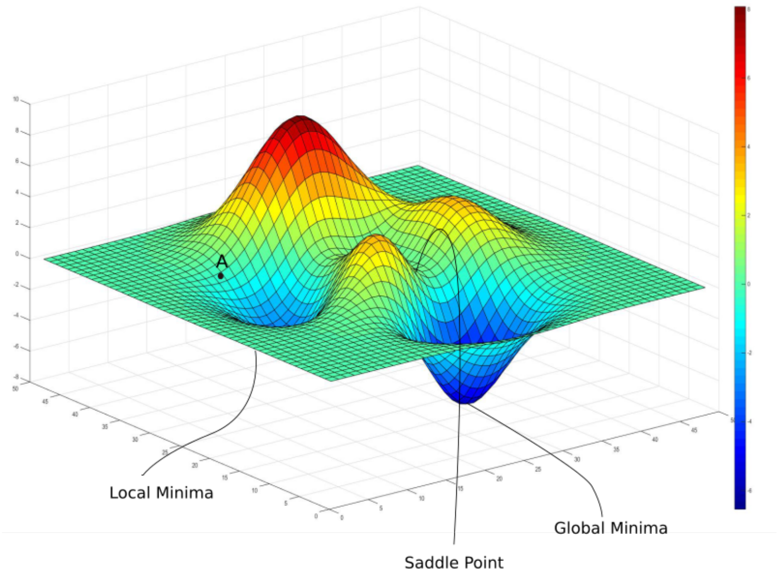


Example of a complex loss function

- An example of a complex loss for spotting actions in a soccer video.



Learning process: loss surface



Learning process: gradient descend

Gradient descend

The weights are updated, at each epoch k, according to the following equation:

$$\theta_{k+1} \leftarrow \theta_k - \eta_k \frac{\delta L}{\delta \theta_k} \tag{223}$$

where an epoch is when the entire dataset passes both forward and backward through the neural network once and η_k is the learning rate at epoch k.

Learning process: mini-batch stochastic gradient descend

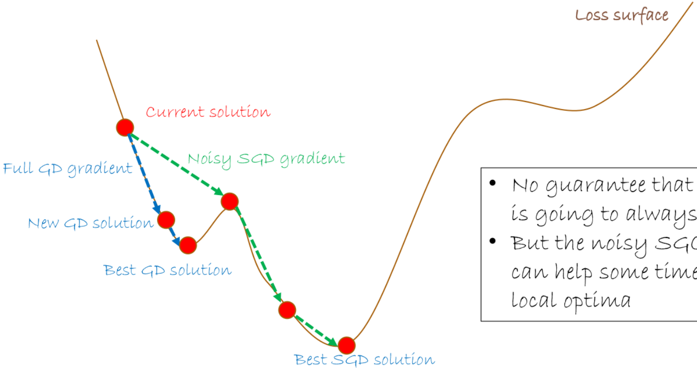
Mini-batch stochastic gradient descend

The weights are updated, at each mini-batch j, according to the following equation:

$$\theta_{j+1} \leftarrow \theta_j - \frac{\eta_j}{|B_j|} \sum_{i \in B_j} \frac{\delta L_i}{\delta \theta_j} \tag{224}$$

where B_j is a “mini-batch” from the train set X.

Gradient descend summary



- No guarantee that this is what is going to always happen.
- But the noisy SGC gradients can help some times escaping local optima

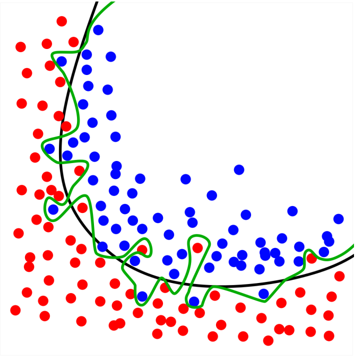
Figure: Comparison between gradient descend and (mini-batch) stochastic gradient descend.

Evaluation methodology

The data should be split in 3 disjoint sets:

1. Training set
   - To optimize the parameters θ of the network.
2. Validation set
   - To optimize the network architecture.
   - To stop the training and avoid overfitting.
   - To optimize other hyper-parameters.
3. Test set
   - To evaluate the performances on *unseen* data.

Overfitting



Where are the difficulties ?

It might seem now that deep learning can virtually solve any problem using big enough networks and computational resources.

But:

- The process of *training is complex*:
  - issues with convergence
  - avoid overfitting
  - usually no knowledge about the data statistics
- The *architecture* and the type of *building blocks* used inside the network play a great role.
  - It is better to have a small architecture with operations adapted to your problem. But no rules of thumb, directly applicable, available.

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

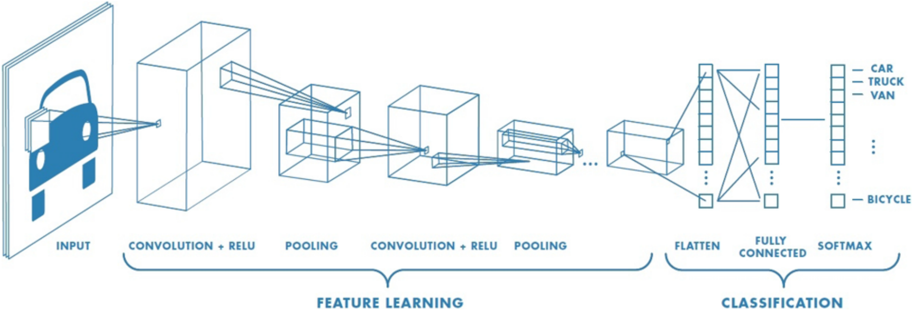
- Motivation
- General view
- Deep learning building blocks
- Applications of deep learning with images

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

Deep learning building blocks



The single perceptron (neuron)

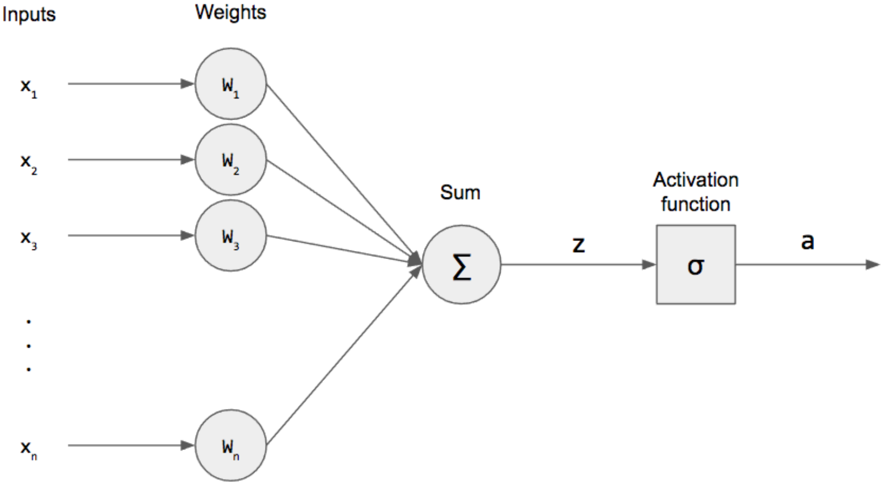


Figure: Illustration of a single perceptron. The output is a weighted sum of the inputs. The parameters w_i are the ones to be learned.

The single perceptron (neuron)

- Main idea:
  - One weight w_i per input x_i
  - Compute a weighted sum of the inputs
  - Add a bias term with weight w_0 and input $x_0 = 1$
  - Apply a non-linear activation function σ on the weighted sum

$$\hat{y}\left(x, w_{0,\ldots,n}\right) = \sigma\left(\sum_{i=1}^{n} w_i x_i + w_0\right) \tag{225}$$

Difficulties

- One perceptron = one decision
- Some functions cannot be represented by such a simple model

The multi-layer perceptron (MLP)

Main idea:

- Organize multiple perceptrons in layers and connect each neuron of a layer to all other neurons of the next layer.
- The hidden layers will capture different features of the input data.

$$\hat{y}(x, \theta_{1,\dots,L}) = h_L\left(h_{L-1}\left(\dots h_1\left(x, \theta_1\right), \theta_{L-1}\right), \theta_L\right) \tag{226}$$

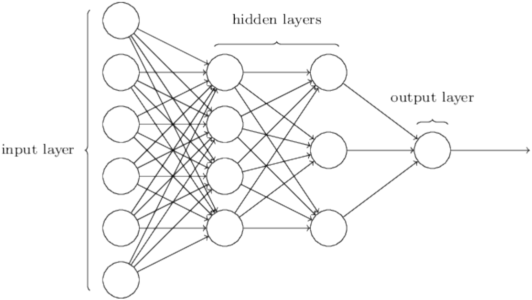


Figure: Illustration of a multi layer perceptron (MLP).

Some activation functions

Some activation functions $\sigma(x)$ can be used at the output of each neurons in order to add non-linearities inside of the model.

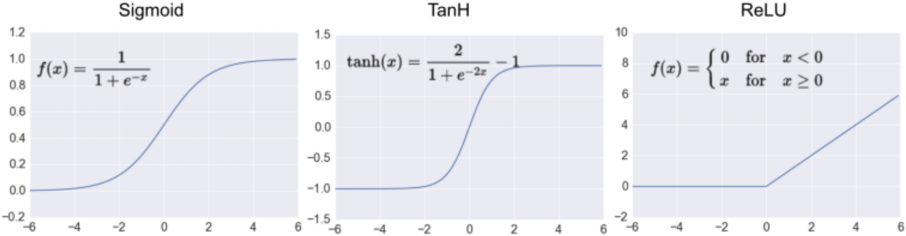


Figure: Different types of commonly used activation functions.

High dimensional data

- Raw data live in huge dimensionalities.
- Semantically meaningful raw data prefer lower dimensional manifolds.
- The goal is to discover this manifold to embed our data.

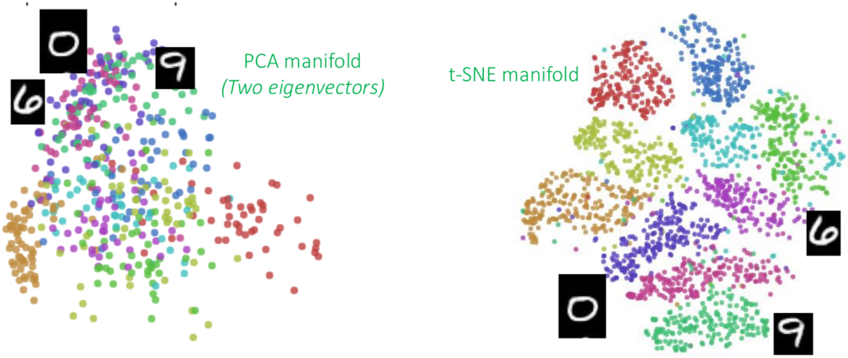
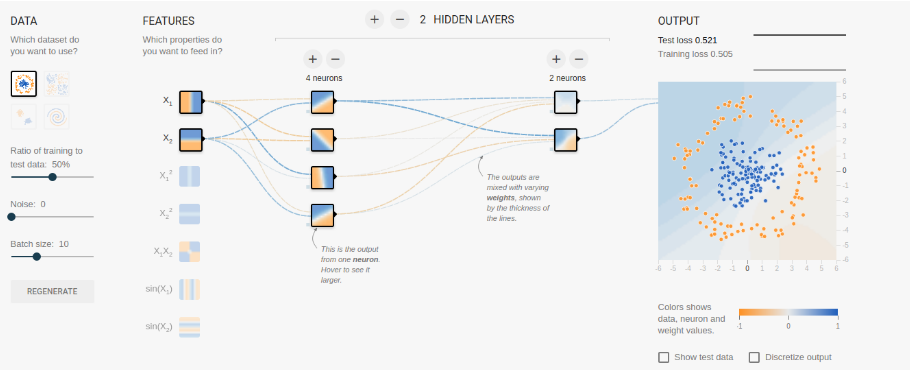


Figure: Pixel vs feature representation of the digits of the MNIST dataset.

Tensorflow playground

It is possible to visualize the training of a neural network thanks to tensorflow playground:

https://playground.tensorflow.org



DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

2 neurons

This is the output from one **neuron**. Hover to see it larger.

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

OUTPUT

Test loss 0.521

Training loss 0.505

Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

Figure: Tensorflow playground running in a web browser.

Deep learning with images

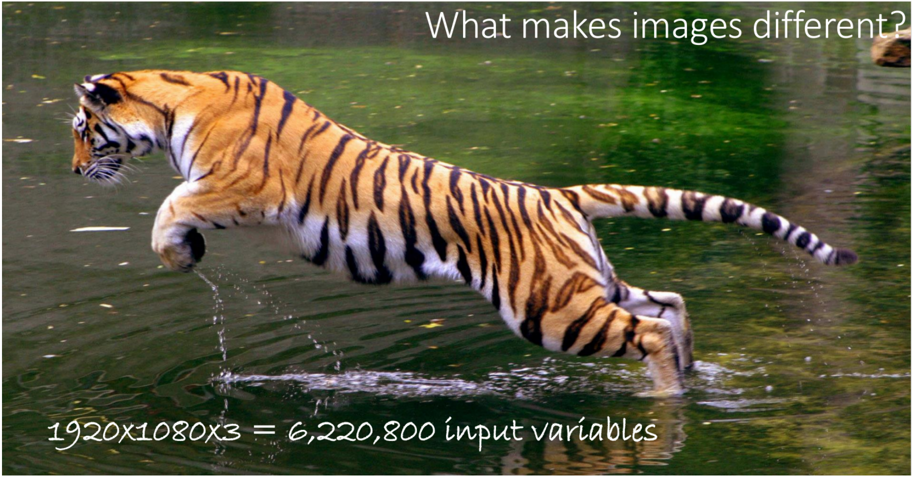

What makes images different?
1920x1080x3 = 6,220,800 input variables

Figure: The number of input variables in images explodes.

The image data structure

- Images are 2-dimensional with either:
  - 1 channel (e.g. grey images)
  - 3 channels (e.g. RGB)
  - k channels (e.g. multi-spectral)
- Huge dimensionality
  - A 256×256 RGB image amounts to $200K$ input variables.
  - A 1-layered NN with 1000 neurons would have 200 millions parameters.

Spatial correlation

- Images are stationary signals, they share features spatially.
  - After modifications (color, rotation, flipping, ...), images are still meaningful.
  - Small visual changes are often invisible to naked eye.
  - Basic natural image statistics are the same.
- Neighboring variables (pixels) are correlated !
  - Information from neighboring pixels is meaningful when grouped.

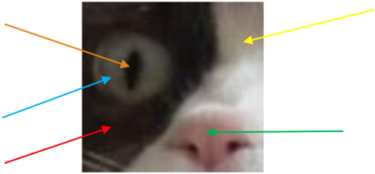


Figure: The black pixel of the eye of the cat only makes sense in its own context.

Filters to analyze images

- Several, handcrafted filters in computer vision:
  - Canny, Sobel, Gaussian blur, smoothing, morphological filters, Gabor filters
- Are they optimal for feature extraction and recognition?

e.g. Sobel 2-D filter

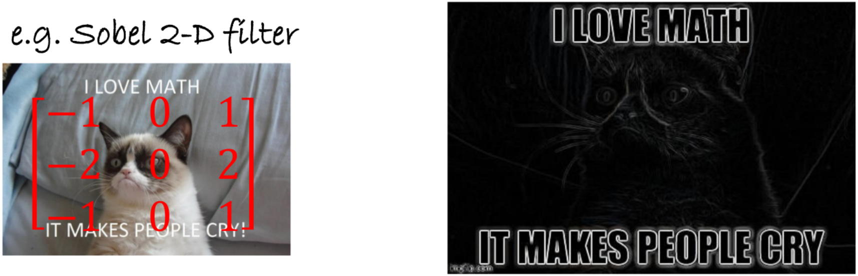

$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
I LOVE MATH
IT MAKES PEOPLE CRY!
I LOVE MATH
IT MAKES PEOPLE CRY

Figure: Example of the well-known handcrafted Sobel 2-D filter.

Learning the filters to use - Convolutional layer

Convolutional layer

The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map. The goal is to optimize the θ_{ij} parameters of the filters.

$$\text{filter} \equiv \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{bmatrix} \tag{227}$$

Quick reminder on convolutions I

Original image

Quick reminder on convolutions II

Original image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Quick reminder on convolutions III

Original image

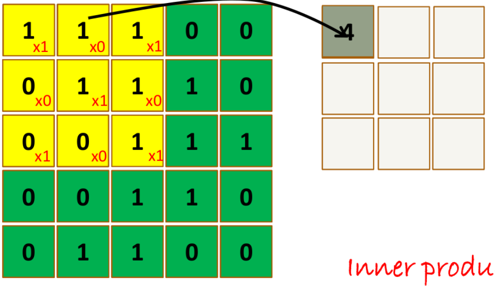
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

Result



1 x1	1 x0	1 x1	0	0
0 x0	1 x1	1 x0	1	0
0 x1	0 x0	1 x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

$$I(x,y)*h=\sum_{i=-a}^{a}\sum_{j=-b}^{b} I(x\overbrace{-\,i,y-j)\cdot h(i,j)}^{\text{Inner product}}$$

Quick reminder on convolutions IV

Original image

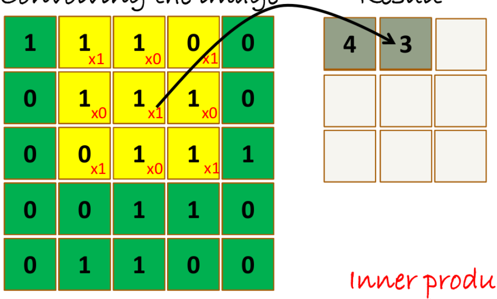
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

Result



1	1 x1	1 x0	0 x1	0
0	1 x0	1 x1	1 x0	0
0	0 x1	1 x0	1 x1	1
0	0	1	1	0
0	1	1	0	0

4	3	

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x \overbrace{- i, y - j) \cdot h(i,j)}^{\text{Inner product}}$$

Quick reminder on convolutions V

Original image

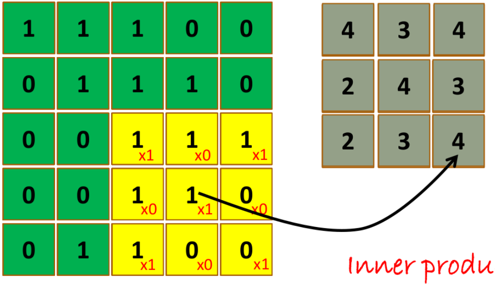
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

Result

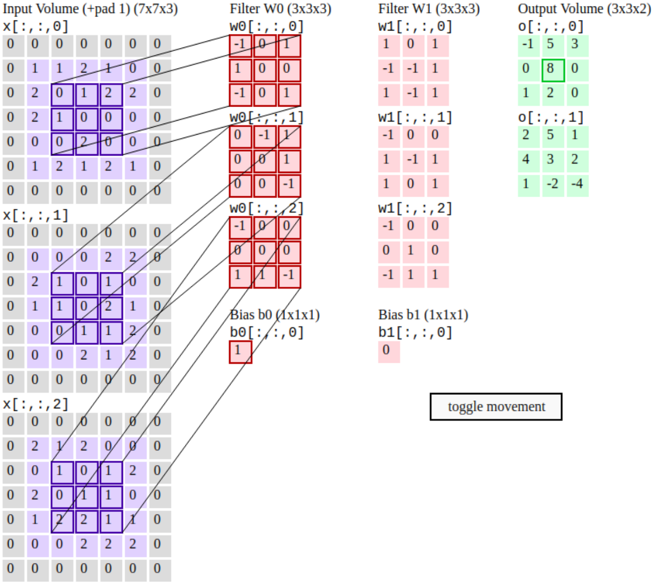


1	1	1	0	0
0	1	1	1	0
0	0	1 x1	1 x0	1 x1
0	0	1 x0	1 x1	0 x0
0	1	1 x1	0 x0	0 x1

4	3	4
2	4	3
2	3	4

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x \overbrace{- i, y - j) \cdot h(i,j)}^{\text{Inner product}}$$

Filter convolution for *k* channels images



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

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

x[:,:,1]

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

x[:,:,2]

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

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
1	0	0
-1	0	1

w0[:,:,1]

0	-1	1
0	0	1
0	0	-1

w0[:,:,2]

-1	0	0
0	0	0
1	1	-1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

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

w1[:,:,1]

-1	0	0
1	-1	1
1	0	1

w1[:,:,2]

-1	0	0
0	1	0
-1	1	1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

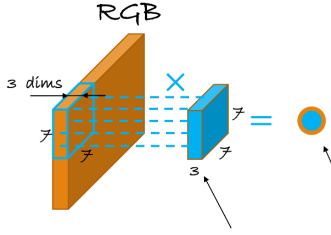
o[:,:,0]

-1	5	3
0	8	0
1	2	0

o[:,:,1]

2	5	1
4	3	2
1	-2	-4

Towards the convolutional layer I

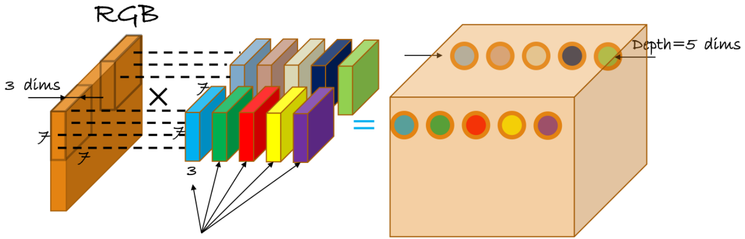


How many weights for this neuron?

$$7 \cdot 7 \cdot 3 = 147$$

Towards the convolutional layer II

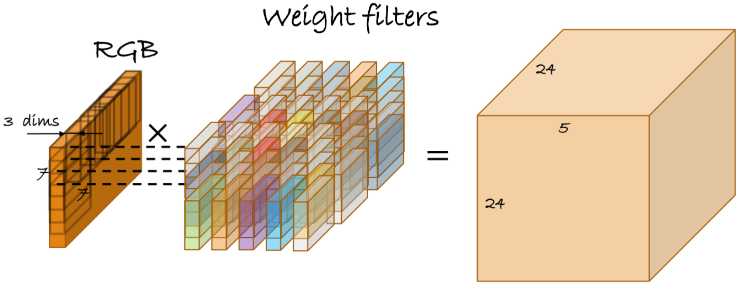
The activations of a hidden layer form a volume of neurons, not a 1-d "chain"
This volume has a depth 5, as we have 5 filters



How many weights for these 5 neurons?

$5 \cdot 7 \cdot 7 \cdot 3 = 735$

Towards the convolutional layer III



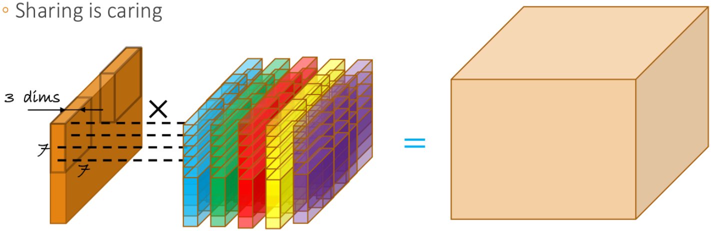
Assume the image is 30x30x3.
1 filter every pixel (stride =1)
How many parameters in total?

24 filters along the x axis
24 filters along the y axis
Depth of 5
× $7 * 7 * 3$ parameters per filter

$423K$ parameters in total

Towards the convolutional layer IV

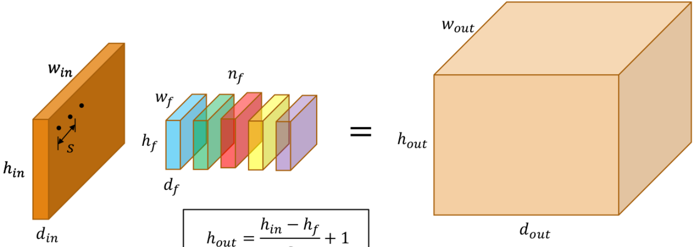
- So, if we are anyways going to compute the same filters, why not share?
  - Sharing is caring



Assume the image is 30x30x3.
1 column of filters common across the image.
How many parameters in total?

$$\begin{array}{rl} & \text{Depth of } 5 \\ \times & 7 * 7 * 3 \text{ parameters per filter} \\ \hline & 735 \text{ parameters in total} \end{array}$$

Towards the convolutional layer V



$$h_{out} = \frac{h_{in} - h_f}{s} + 1$$

$$w_{out} = \frac{w_{in} - w_f}{s} + 1$$

$$d_{out} = n_f$$

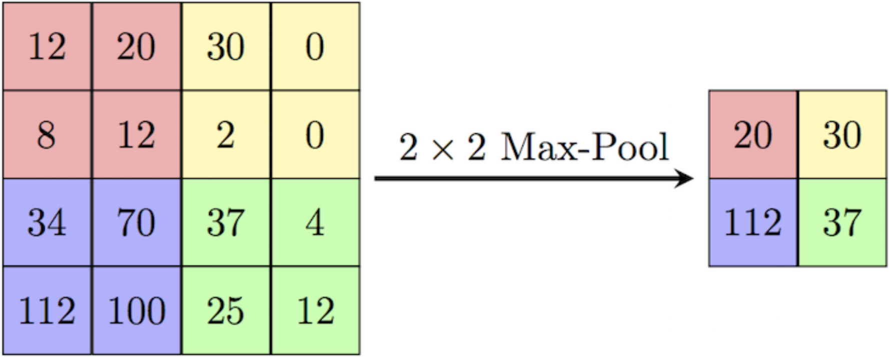
$$d_f = d_{in}$$

Pooling the features

- Aggregate multiple values into a single value. Advantages:
  - is invariant to small transformations.
  - reduces the size of the layer output/input to next layer.
  - keeps most important information for the next layer.
- Examples:
  - Max pooling
  - Average pooling

Max pooling

- Run a sliding window.
- At each location keep the maximum value.
- The most usual choice of pooling.
- The goal is to keep the most intense activations.



12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

$\xrightarrow{2 \times 2 \text{ Max-Pool}}$

20	30
112	37

General structure of a Convolutional Neural Network (CNN)

- Convolution layer followed by activations and max pooling.
- Repeat this configuration several times.
- The end of the network is a MLP.

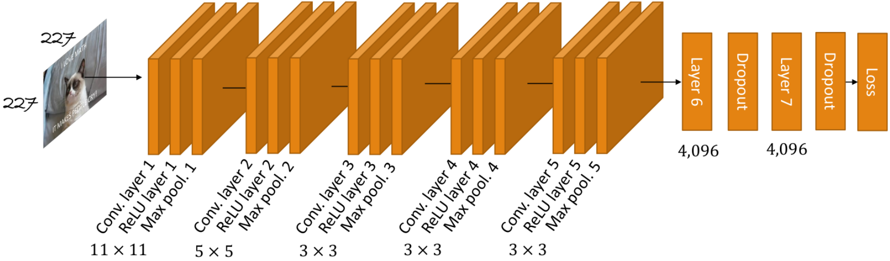
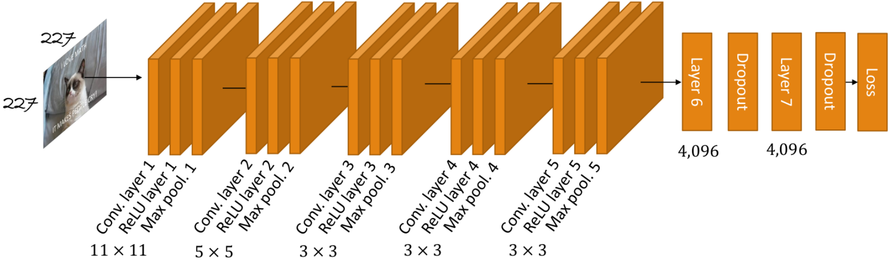


Figure: General structure of a CNN for image classification.

End-to-end learning of feature hierarchies

- A pipeline of successive modules.
- Each module's output is the input for the next module.
- Modules produce features of higher and higher abstractions.
  - Initial modules capture low-level features (e.g. edges or corners).
  - Middle modules capture mid-level features (e.g. circles, squares, textures).
  - Last modules capture high level, class specific features (e.g. face detector).



Residual blocks

- Residual blocks allow to only learn the residuals that needs to be added to the input rather than the whole transformation.
  - Useful for learning transformations (e.g. semantic segmentation).
  - Allows to train deeper networks.

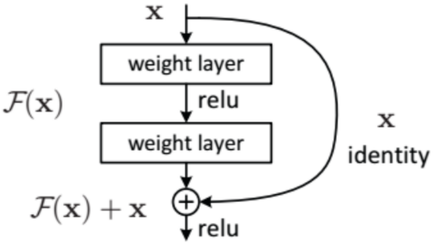


Figure: Residual block: the weights learn the *residues* to be added to the original features.

ResNet

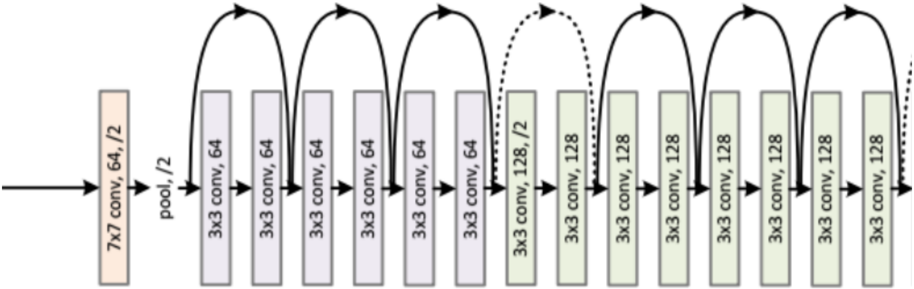


Figure: Example of a network, *ResNet*, using skip connections.

Dropout I

- During training, some activations are randomly set to 0.
  - Neurons sampled at random from a Bernoulli distribution with $p = 0.5$.
- At test time all neurons are used.
- Benefits:
  - Reduce complex co-adaptations or co-dependencies between neurons.
  - Every neuron becomes more robust.
  - Significantly decrease the overfitting.
  - Improve the training speed significantly.

Dropout II

- Effectively, a different architecture is used at every training epoch.

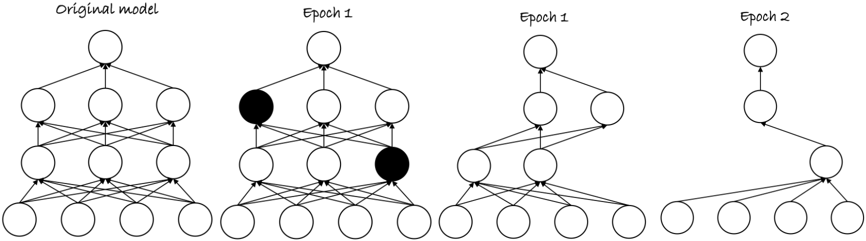


Figure: The network and thus the path to the output is different at every epoch but the weights are still shared.

Data normalization

- Center data to be roughly 0.
  - Activation functions are usually "centered" around 0.
  - The gradient of most activation functions is maximum around 0.
  - Convergence is usually faster.

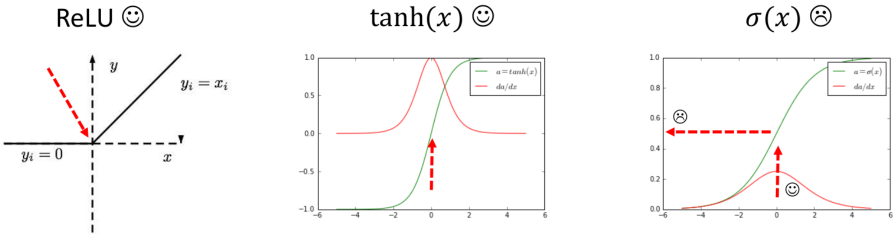

ReLU ☺ tanh(x) ☺ $\sigma(x)$ ☹

Figure: Most activation functions are centered around zero.

Data normalization

- Hypothesis:
  - Input variables follow a Gaussian distribution (roughly)
- Goal:
  - Go from $\mathcal{N}\left(\mu, \sigma^2\right)$ to $\mathcal{N}(0,1)$
- In practice:
  - First, compute the mean and standard deviation on the training set.
  - Then subtract the mean from training samples.
  - Finally, divide the result by the standard deviation.

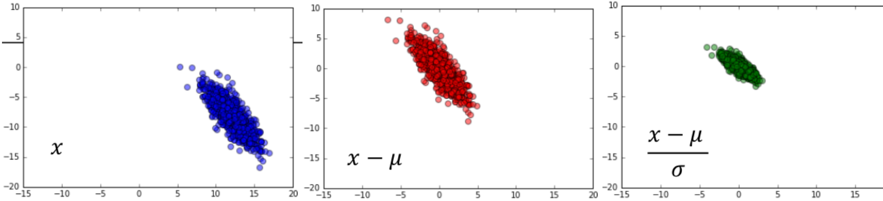


Figure: Normalization of the distribution in the training set.

Data normalization in images

- If dimensions have similar values (e.g. pixels in natural images):
    - Compute one (μ, σ^2) instead of as many as the input variables (not per pixel).
    - Or the per color channel pixel mean and variance.

$$\left(\mu_{\text{red}}, \sigma^2_{\text{red}}\right), \left(\mu_{\text{green}}, \sigma^2_{\text{green}}\right), \left(\mu_{\text{blue}}, \sigma^2_{\text{blue}}\right) \tag{228}$$

- When input dimensions have similar ranges centering might be enough.
    - All pixels have the same range so dividing by 255 and centering might be enough normalization in some cases.

Batch normalization

- It is also possible to normalize the inputs of hidden layer over a mini-batch.
  - The weights change the distribution of the features inside of the network and it might be good to re-center them.
  - It has two learnable parameters γ and β to shift and scale the normalization.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{229}$$

$$\hat{y}_i = \gamma \hat{x}_i + \beta \tag{230}$$

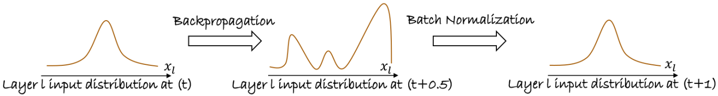

Backpropagation
Batch Normalization
x_l
Layer l input distribution at (t)
Layer l input distribution at (t+0.5)
Layer l input distribution at (t+1)

Figure: Batch normalization for the features inside of the networks.

Class imbalance

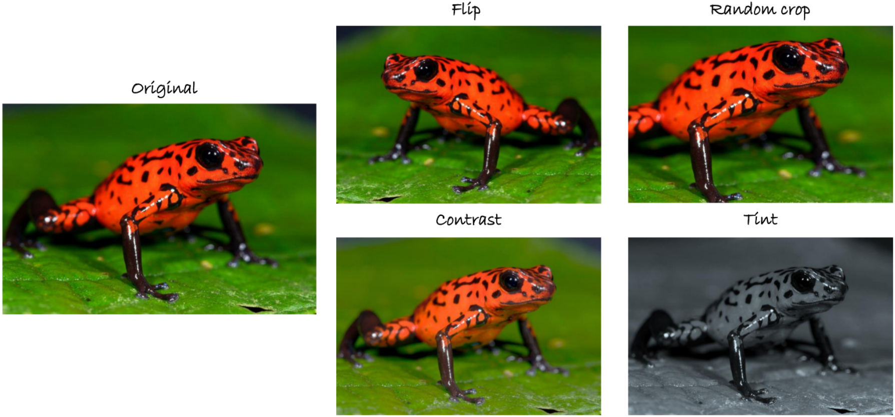
- Classes might not be represented by the same number of samples.
  - Most datasets do not have exactly equal number of instances in each class.
  - The training can converge to simpler sub-optimal solutions, such as always predicting the most frequent class.
  - The evaluation can be biased. For example, in a dataset with 20:1 ratio, the accuracy of a classifier always predicting the first class is of $\simeq 95\%$.

Possible solutions

- Weighting the samples in the loss function.
- Resampling the dataset.
- Use data augmentation for the less frequent classes.

Data augmentation

- Data augmentation can be performed in order to increase the number of training samples.
  - Translations, rotations, warping,...
  - Only use transformations that are relevant for your problem.



Original

Flip

Random crop

Contrast

Tint

Figure: Examples of data augmentation in the data space

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
    - Motivation
    - General view
    - Deep learning building blocks
    - Applications of deep learning with images
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration

Applications of deep learning with images

Classification

- Goal:
  - Classify each image in one of the N classes.
- Datasets:
  - MNIST
  - ImageNet
  - ...
- Networks:
  - LeNet-5
  - ResNet
  - ...

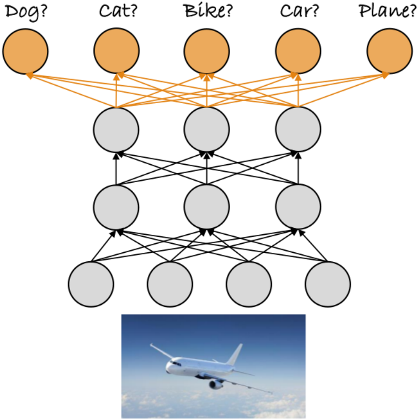


Figure: Examples of a 5-class classification problem for images.

Regression

- Goal:
  - Regress one or many continuous values.
  - Object detection
- Datasets:
  - Pascal VOC
  - COCO
  - ...
- Networks:
  - MaskRCNN
  - Yolo
  - ...

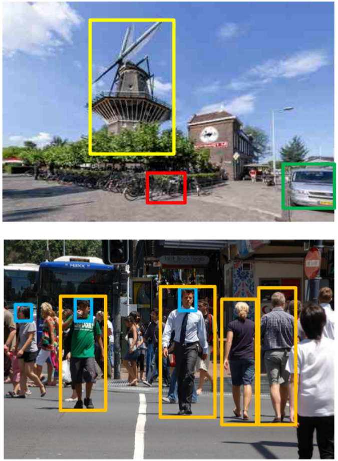


Figure: Examples of a bounding box regression problem.

Semantic segmentation

- Goal:
  - Assign a class to each pixel.
- Datasets:
  - Kitti, CityScapes,...
- Networks:
  - PSPNet, MaskRCNN,...

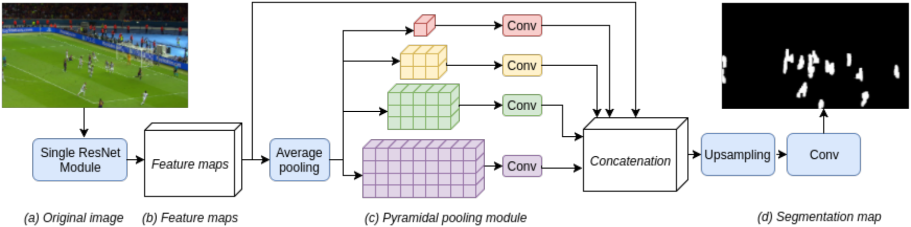


Figure: Semantic segmentation for player segmentation in soccer.

Interpretability

- It is important to understand what happens inside the network.
  - Medical applications
  - Self-driving cars
- Some tools can help to understand how the network understands its environment.
  - Capsule networks, prototypes, etc

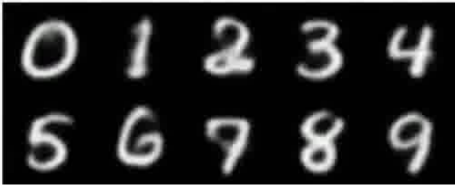
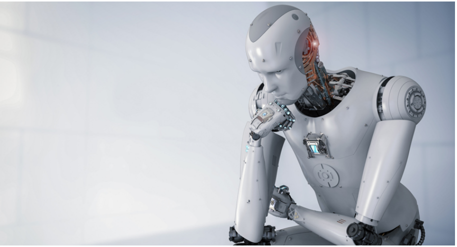


Figure: Prototypes of the MNIST digit classes conceptualized by a network (HitNet).

Conclusion

Take away messages

- Understand your data.
- Use an appropriate architecture and operations relevant and for your data.
- Use a valid evaluation methodology.
- Try to interpret the results.



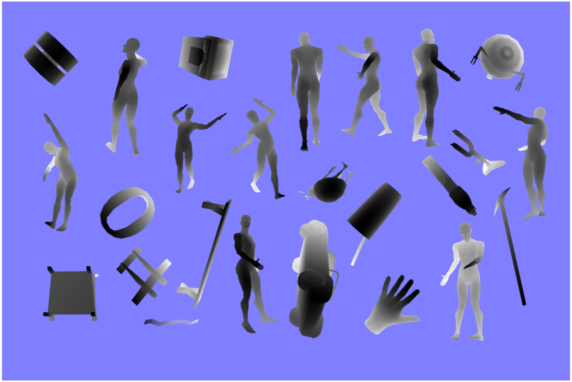
Sources

The content of this presentation is partly inspired by the slides of Efstratios Gavves from the University of Amsterdam (UVA). Some slides and figures are directly copied from that source. http://uvadlc.github.io/

Outline

1 Fundamentals of 2D imaging
2 Linear filtering
3 3D vision: calibration and reconstruction
4 Mathematical morphology
5 Non-linear filtering
6 Object description and analysis
7 Motion analysis and background subtraction
8 Performance analysis
9 Edge detection
10 Feature detection and tracking
11 Introduction to machine learning
12 Fundamentals of deep learning for visual understanding
13 **Deep learning architectures for image analysis**
14 Segmentation
15 Template Matching & Image Registration

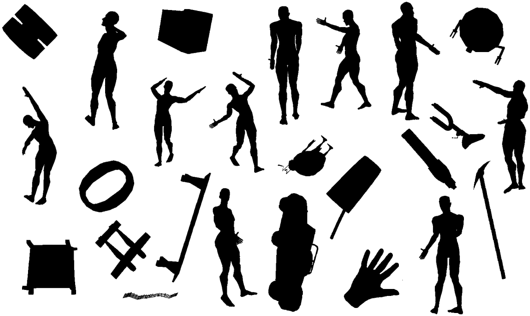
Start here... I



- Can you decide which silhouettes are those of humans?
- Try to write an algorithm to solve this problem!

Start here... II

After binarization of the silhouettes

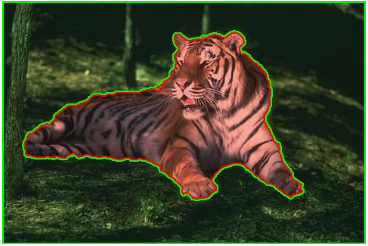


- Can you decide which silhouettes are those of humans?
- Try to write an algorithm to solve this problem!

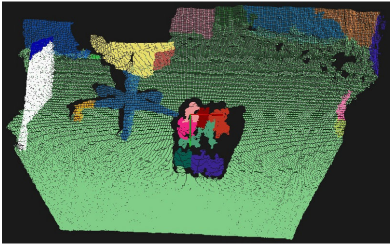
Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
    - Problem statement
    - Segmentation by thresholding
    - Segmentation by region growing: illustration with the watershed
15. Template Matching & Image Registration

Image segmentation I



Segmentation of a color image



Segmentation of a depth image

- Problem statement
- Segmentation by thresholding
- Segmentation by region detection (region growing)
  - Watershed
- Segmentation by classification (semantic classification)

Image segmentation II

General considerations:

- a very specific problem statement is not always easy.
- chicken-and-egg problem; segmentation is an ill-conditioned problem.

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
    - Problem statement
    - Segmentation by thresholding
    - Segmentation by region growing: illustration with the watershed
15. Template Matching & Image Registration

Problem statement I

Definition (Segmentation)

Generally, the problem of segmentation consists in finding a set of non-overlapping regions $R_1, \dots, R_n$ such that

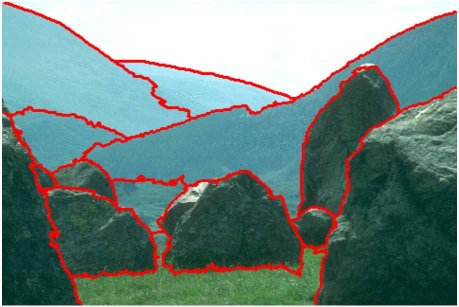
$$\mathcal{E} = \bigcup_{i=1}^{n} R_n \quad \text{and} \quad \forall i \neq j,\, R_i \cap R_j = \emptyset \tag{231}$$

Definition (Alternative definition)

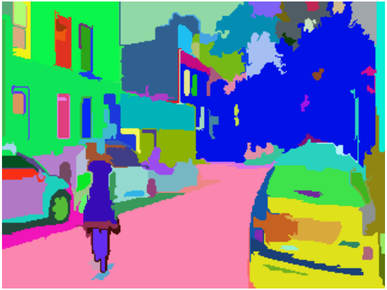
More formally, the segmentation process is an operator ϕ on an image I that outputs, for example, a binary image $\phi(I)$ that differentiates regions by selecting their borders.

An alternative consists to attribute a different label to each pixel of different regions (this is called *scene labeling*).

Problem statement II



Segmented image



Labelled image

Problem statement III

Segmentation is a spatial process

As any similar operator, segmentation can be *local* or *global*:

- for *local* segmentation techniques, the results for one given pixel does not impact the segmentation result outside a close neighborhood.
- for *global* techniques, changing one pixel value can impact the whole result.

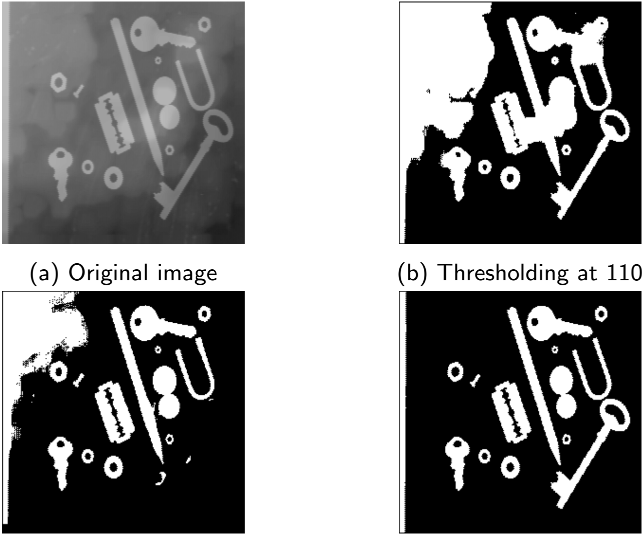
A first typology of segmentation techniques and comparisons

Family of segmentation techniques	input	local/global	markers
Thresholding	image	local (pixel)	no
Watershed	image, gradient, etc	global	yes

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
    - Problem statement
    - Segmentation by thresholding
    - Segmentation by region growing: illustration with the watershed
15. Template Matching & Image Registration

Segmentation by thresholding I



(a) Original image

(b) Thresholding at 110

(c) Thresholding at 128

(d) Thresholding after background equalization

Segmentation by thresholding II

Rationale

There are two classes of pixels:

1. *background* pixels
2. *foreground* pixels

[Note that background and foreground do not refer to motion in this case!].
Sometimes, there is a "*don't know*" class

Assumptions to solve the segmentation problem:

1. the probability density functions of the two content types are different.
2. one threshold or two thresholds (Otsu's method) are sufficient.

Segmentation by thresholding III

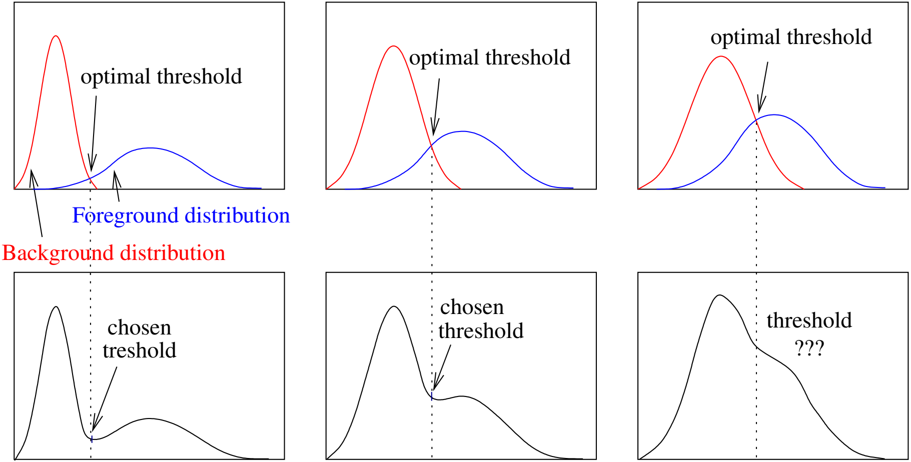
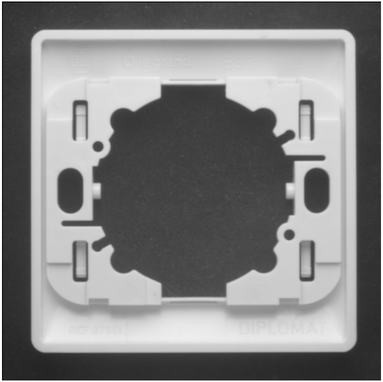


Figure: Optimal threshold.

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
    - Problem statement
    - Segmentation by thresholding
    - Segmentation by region growing: illustration with the watershed
15. Template Matching & Image Registration

Images can be seen as topographic surfaces



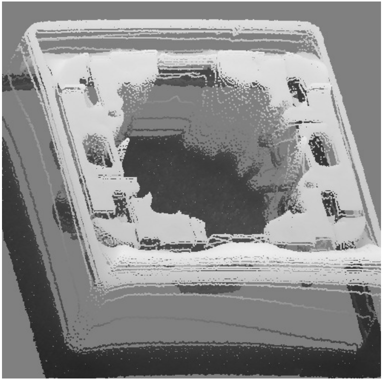


Figure: An image (left-hand side) and a view of its corresponding topographic surface (right-hand side).

Segmentation by watershed

In the terms of a topographic surface, a catchment basin $\mathcal{C}(M)$ is associated to every minimum M.

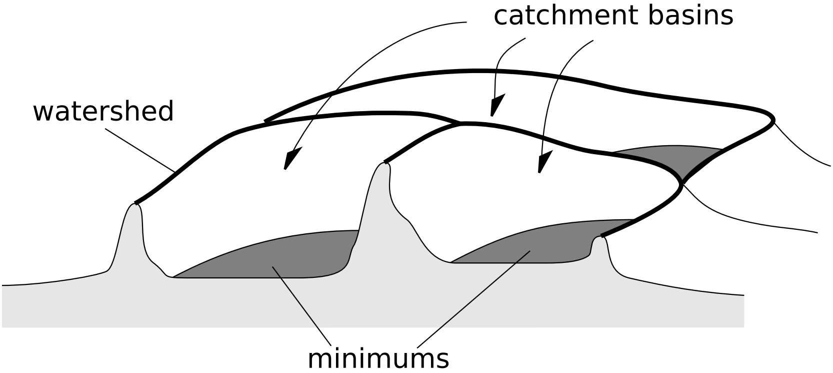
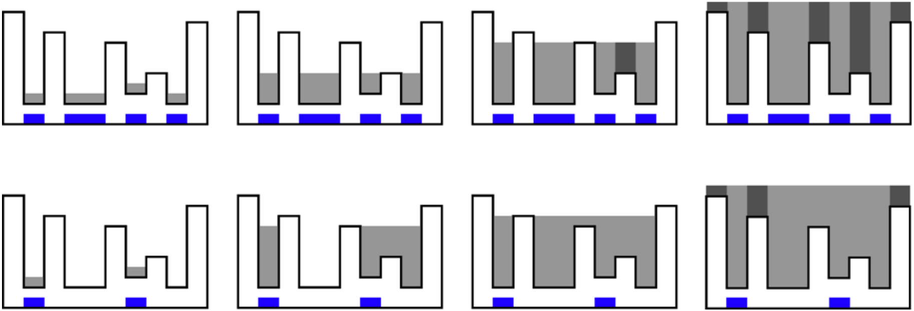


Figure: Minimums, catchment basins, and watershed.

The general principles of the watershed



Steps:

1. identify valleys (minimums)
2. proceed to flooding
3. construct dams between neighboring catchment basins

A formal description of a segmentation algorithm based on the watershed

Approach: proceed level by level

- An horizontal “slice” is binary image. Therefore, we first study the case of binary images.
- definition of geodesic path and distance.
- description of an algorithm that handles a stack of thresholded images.

Geodesic path

Let X be a binary image.

Definition (Geodesic path)

A geodesic path, of length l, between two points s and t is a series of $l+1$ pixels $x_0 = s, x_1, \ldots, x_l = t$ such that

$$\forall i \in [0, l],\, x_i \in X \;\text{ and }\; \forall i \in [0, l],\, x_{i-1}, x_i \text{ are neighbors} \tag{232}$$

Note that this definition applies to images defined on digital grids.

Geodesic distance

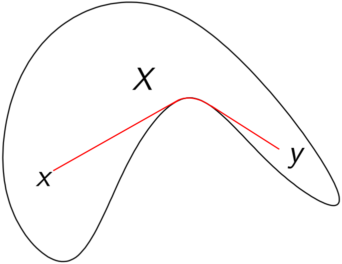


Figure: The shortest path between x and y.

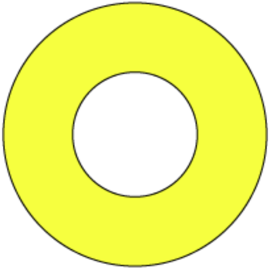
Definition (Geodesic distance)

The geodesic distance between two points s and t is the length of the shortest geodesic path linking s to t; the distance is infinite if such a path does not exist.

About the geodesic distance

Questions:

1. is the geodesic *distance* between x and y *unique*?
2. is the geodesic *path* between x and y *unique*?



Algorithm for the construction of the geodesic skeleton by growing the zone of influence

Notations:

The zone of influence of a set Z_i, is denoted by $ZI(domain = \text{X}, center = Z_i)$ and its frontier by $FR(domain = \text{X}, center = Z_i)$.
The skeleton by zone of influence (SZI) is obtained via the following algorithm:

- first, one delineates the Z_i zones of each region;
- for remaining pixels, an iterative process is performed until stability is reached: if a pixel has a neighbor with an index i, then this pixel gets the same index; pixels with none or two different indices in their neighborhood are left unchanged;
- after all the iterations, all the pixels (except pixels at the interface) are allocated to one region of the starting regions Z_i.

Example

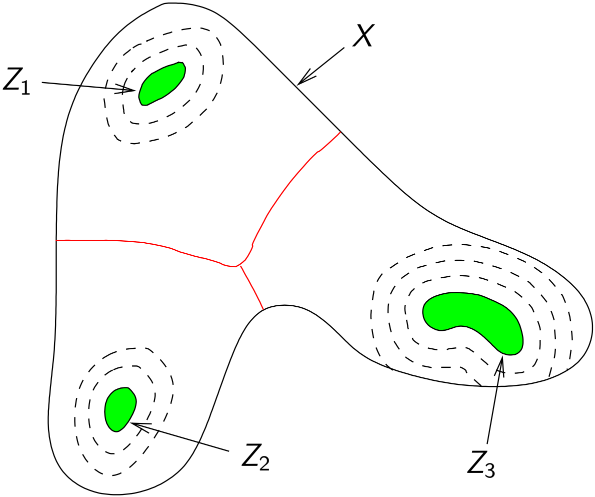


Figure: Geodesic skeleton.

The case of grayscale images (a gradient image for example) I

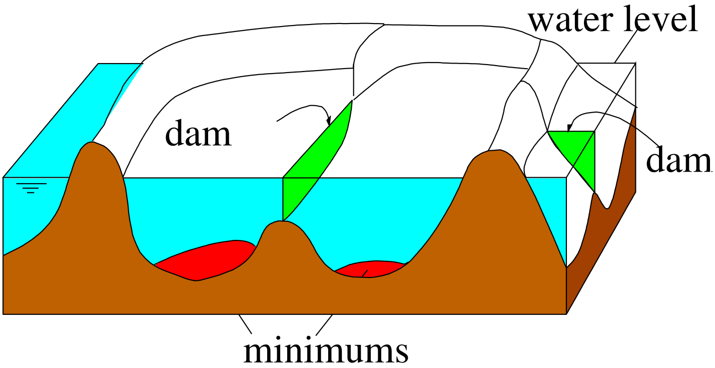


Figure: A dam is elevated between two neighboring catchment basins.

The case of grayscale images (a gradient image for example) II

Notations:

- f is the image.
- h_{min} and h_{max} are the limits of the range values of f on the function support (typically, $h_{min} = 0$ and $h_{max} = 255$).
- $T_h(f) = \{x \in dom\, f : f(x) \leq h\}$ is a set obtained by thresholding f with h. For h growing, we have a stack of decreasing sets.
- M_i are the minimums and $\mathcal{C}(M_i)$ are the catchment basins.

Step by step construction

Let $\mathcal{C}_h(M_i)$ be the subset of the M_i basin filled at “time” (or “height”) h. Then

$$\mathcal{C}_h(M_i) = \mathcal{C}(M_i) \cap T_h(f) \tag{233}$$

In this expression, $\mathcal{C}(M_i)$ is unknown.

Initialization:

- $\mathcal{C}_{h\ min}(M) = T_{h\ min}(f)$; the initialization considers that all the local minimums are valid catchment basin originators.

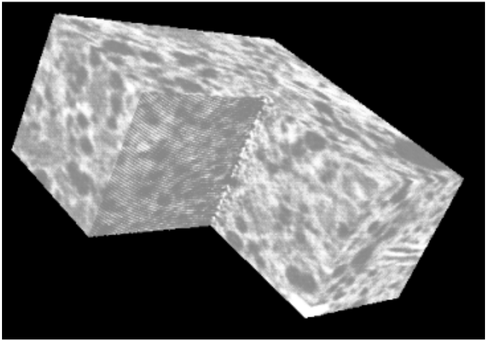
Construction

$$\forall h \in [h_{min}+1,\ h_{max}] : \mathcal{C}_h(M) = ZI_h \cup Min_h \tag{234}$$

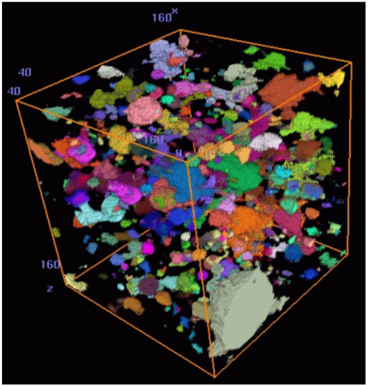
with

- $ZI_h =$ influence zone (with domain $T_h(f)$);
- Min_h is the set of all the points of $T_h(f)$ that have no label after the growing process of influence zones. They correspond to minimums that are introduced at level h.

Illustration: detection of pores in gypsum



Microtomograhy image



Labeled image

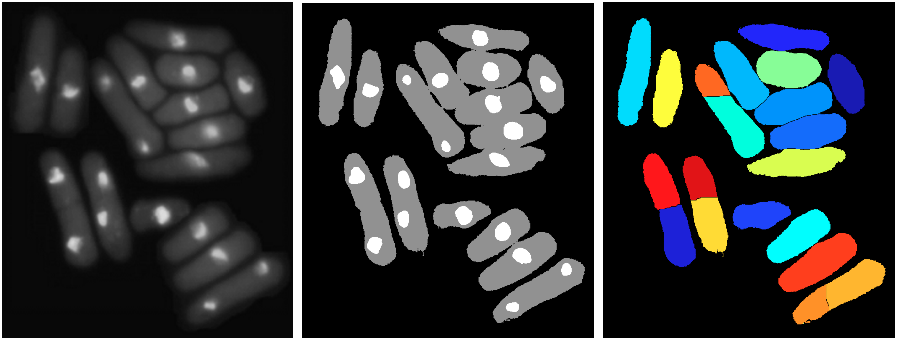
Markers

Marking is a process that allows to select only some of the local minimums.

Watershed has the following advantages with respect to other techniques (such as thresholding):

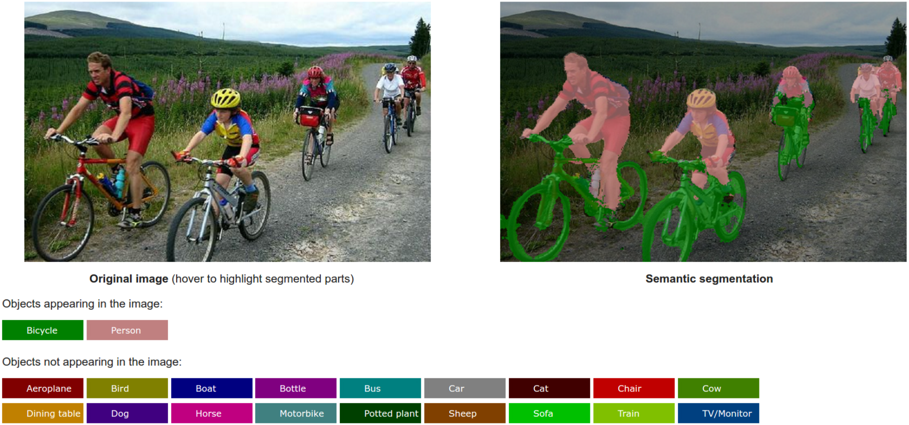
- the possibility to be applicable to *any sort of input image* (original image, gradient, etc).
- the flexibility to put some **markers** to select only a few local minimums. With markers, the amount of regions is exactly equal to the number of markers put in the image.

Illustration: segmentation of cells



Semantic segmentation (based on deep learning)

- Based on classification techniques and machine learning
- Pixel-based
- A series of semantic notions/objects (persons, cars, bicycles, etc)



Original image (hover to highlight segmented parts)

Semantic segmentation

Objects appearing in the image:

Bicycle | Person

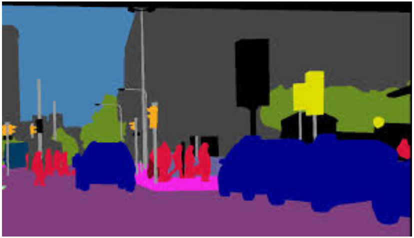
Objects not appearing in the image:

Aeroplane | Bird | Boat | Bottle | Bus | Car | Cat | Chair | Cow

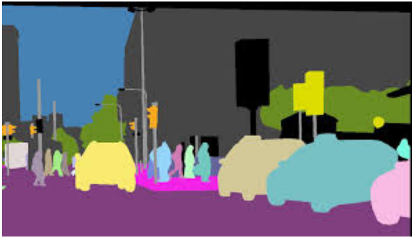
Dining table | Dog | Horse | Motorbike | Potted plant | Sheep | Sofa | Train | TV/Monitor

Panoptic segmentation (based on deep learning)

- Panoptic segmentation $\equiv$ *semantic* segmentation + *instance* segmentation



Semantic segmentation



Panoptic segmentation

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration
    - Introduction & Applications
    - Some solutions and their corresponding approach
    - Template matching & image registration components
    - Implementation speed-up

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration
    - Introduction & Applications
    - Some solutions and their corresponding approach
    - Template matching & image registration components
    - Implementation speed-up

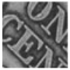
Table of content

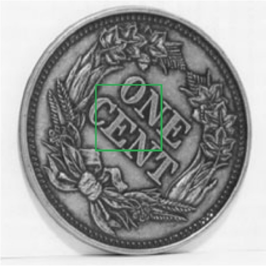
- Introduction & Applications
- Some solutions and their corresponding approach
- Template matching & image registration components
- Implementation speed-up

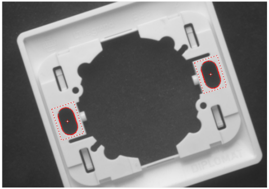
Table of content

- Introduction & Applications
    - **Definition & Examples**
    - Applications
- Some solutions and their corresponding approach
- Template matching & image registration components
- Implementation speed-up

Template matching - Example





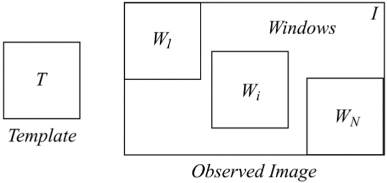


Template matching - Definition

Definition

Template matching is the process of either finding any instance of a template image T within another image I or finding which ones of the templates $T_1, T_2, \ldots, T_N$ correspond in some way to another image I.

We search how to transform or warp a template (resp. image) to make it similar to a reference image (resp. template).
The template is also called the pattern or the model.



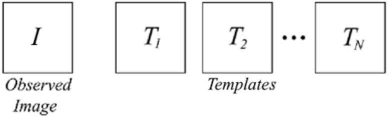
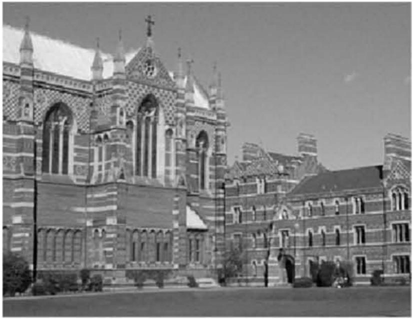


Figure: Find where the template T is located in the observed image I.

Figure: Find the template T_k which correspond to the observed image I.

Image registration or alignment - Example



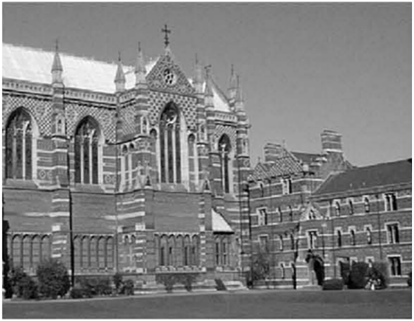


Image registration or alignment - Definition

Definition

Image registration is the process of spatially aligning two images of a scene/object so that corresponding points assume the same coordinates.

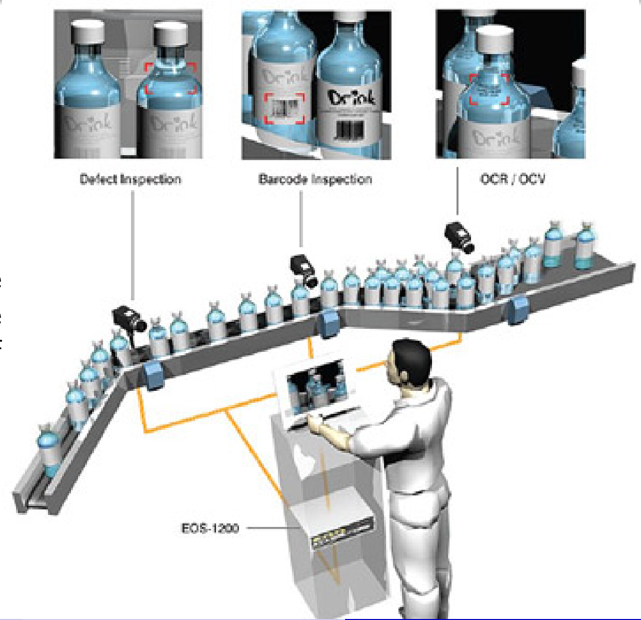
- Given two images taken, for example,
  - at different times,
  - from different devices
  - or different point of view;
- the goal is to determine a reasonable transformation of the images
- such that a transformed version of the first image is similar to the second one.

Table of content

- Introduction & Applications
  - Definition & Examples
  - **Applications**
- Some solutions and their corresponding approach
- Template matching & image registration components
- Implementation speed-up

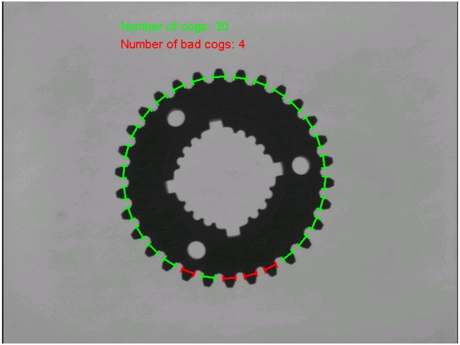
Machine vision I

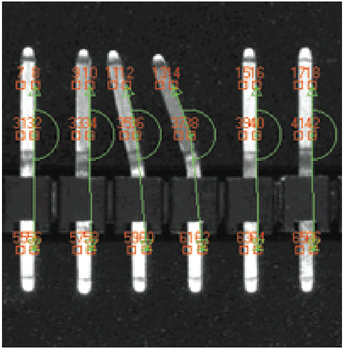
Machine vision (and especially template matching) is pervasive in almost all steps of an industrial production chain.



Machine vision II

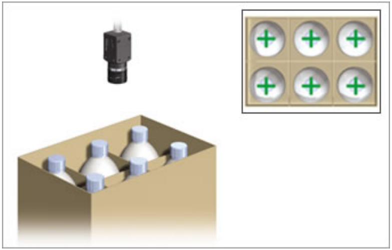
- **Measuring and assessing** the matching quality, the component presence/absence/type and/or position and orientation → Non destructive testing (NDT), defect detection, model conformance assessment

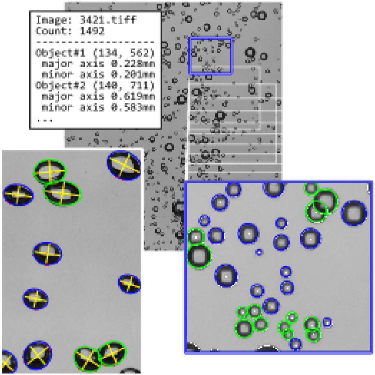




Machine vision III

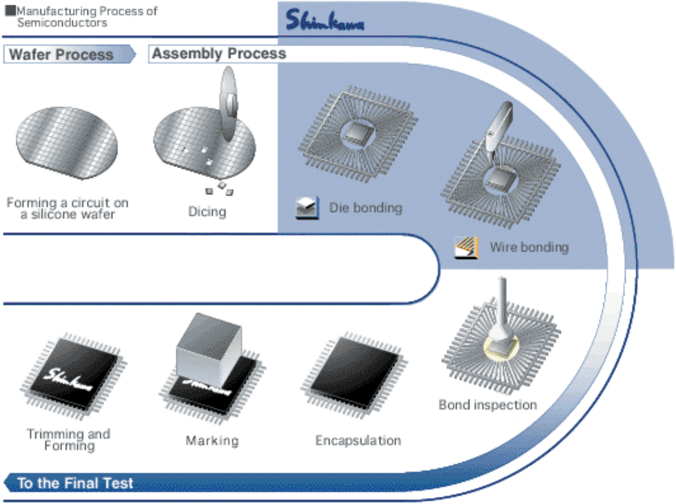
- **Counting the number of instances** that matched the template
  → Detection of given objects, computing their location, type and/or properties





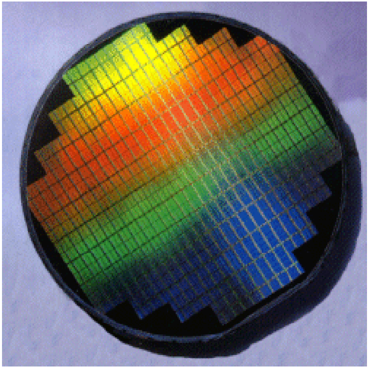
```
Image: 3421.tiff
Count: 1492
--------------------
Object#1 (134, 562)
 major axis 0.228mm
 minor axis 0.201mm
Object#2 (140, 711)
 major axis 0.619mm
 minor axis 0.583mm
...
```

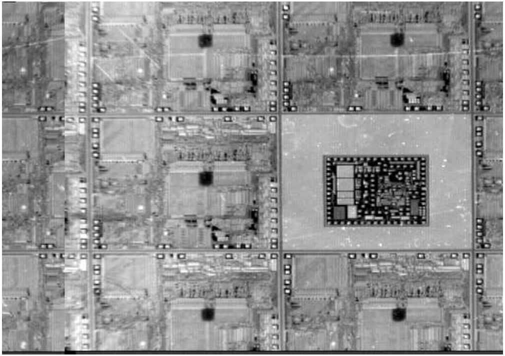
Electronic components manufacturing I



Electronic components manufacturing II

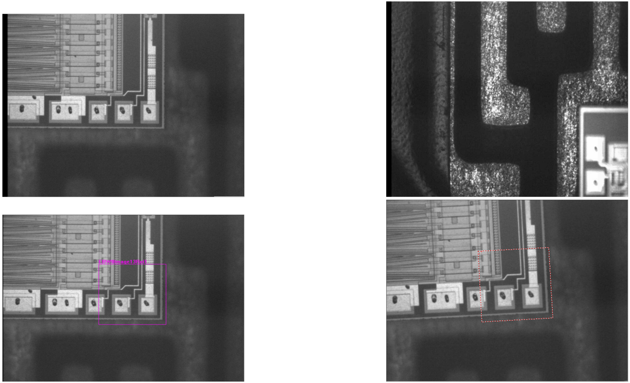
- Wafer dicing is the process by which die are separated from a wafer of semiconductor.





Electronic components manufacturing III

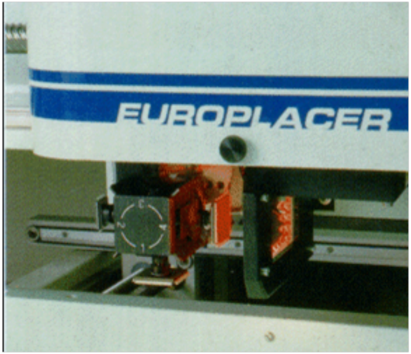
- Die bonding is the process of attaching the semiconductor die either to its package or to some substrate.
- Wire bonding is the process of making interconnections between an integrated circuit or other semiconductor device and its packaging.

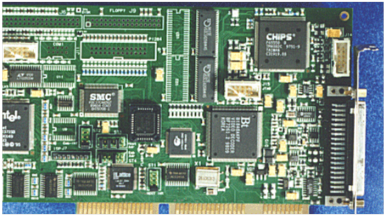


Printed board assembly (pick & place) I

- Position of picked components
- Position of placement area
- Control of welding after the process

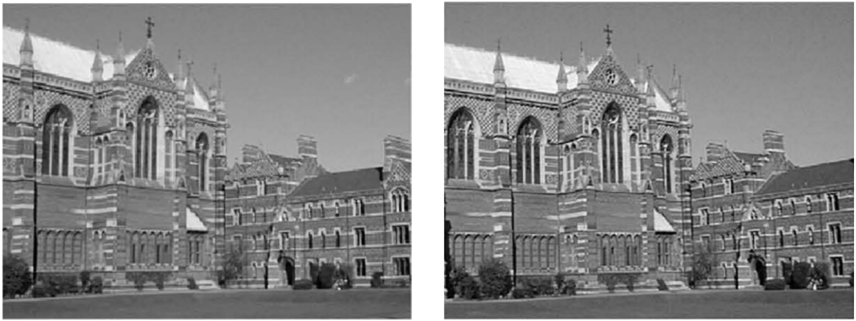
Printed board assembly (pick & place) II





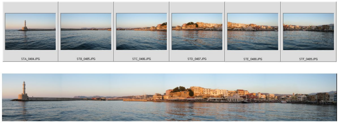
Multi-view correspondences I

- **3D reconstruction**
  → find the correspondences between the left and right view of the same scene



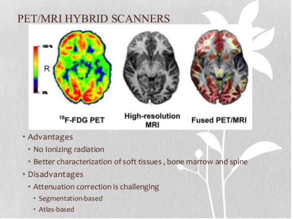
Multi-view correspondences II

- **Panoramic images**: Image alignment for stitching
  → find correspondences between several views of the same scene



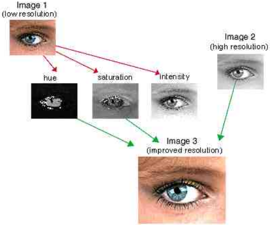
Multi-modality correspondences and fusion

- Image alignment and fusion

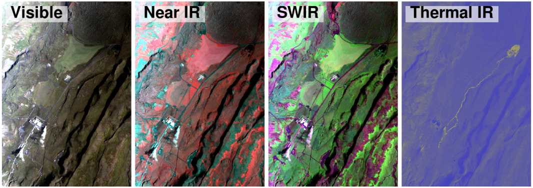


PET/MRI HYBRID SCANNERS

- Advantages
  - No ionizing radiation
  - Better characterization of soft tissues , bone marrow and spine
- Disadvantages
  - Attenuation correction is challenging
    - Segmentation-based
    - Atlas-based

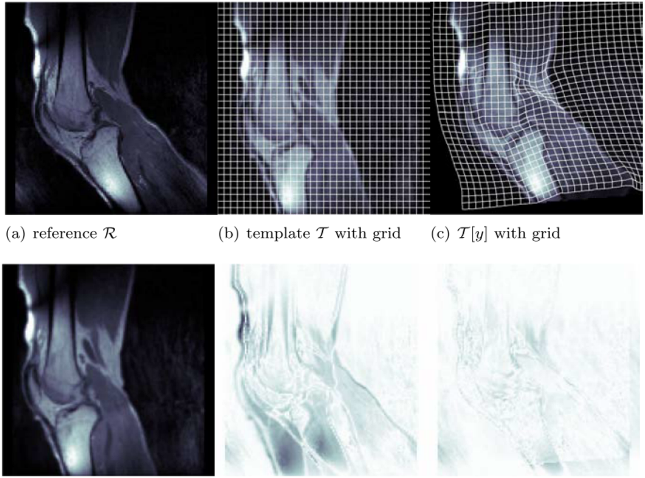


- Remote sensing: satellite image fusion



Biomedical (elastic) image registration

Non-rigid (elastic) image registration



(a) reference $\mathcal{R}$

(b) template $\mathcal{T}$ with grid

(c) $\mathcal{T}[y]$ with grid

(d) template $\mathcal{T}$

(e) difference $|\mathcal{T} - \mathcal{R}|$

(f) difference $|\mathcal{T}[y] - \mathcal{R}|$

From [Modersitzki Jan, "FAIR Flexible Algorithms for Image Registration", 2009, Figure 1.1]

Template matching vs. Image registration

- Template matching and image registration processes have essentially the same goal:
  - They compare two (or more) images, and
  - look for a transformation/warping of one (or both) image(s),
  - in order to match/align the images (to make them fit).
- They differ by the way users consider the images;
  - In template matching, one of the image is special (the template) and is often (not always) smaller in size.
    The other image represents/spans the *work space* where we would like to *locate* the template.
  - In image registration both images play a similar role.
    They are both embedded in a global *work space* where we would like to find their relative *position*.

Outline

1 Fundamentals of 2D imaging

2 Linear filtering

3 3D vision: calibration and reconstruction

4 Mathematical morphology

5 Non-linear filtering

6 Object description and analysis

7 Motion analysis and background subtraction

8 Performance analysis

9 Edge detection

10 Feature detection and tracking

11 Introduction to machine learning

12 Fundamentals of deep learning for visual understanding

13 Deep learning architectures for image analysis

14 Segmentation

15 Template Matching & Image Registration

- Introduction & Applications
- Some solutions and their corresponding approach
- Template matching & image registration components
- Implementation speed-up

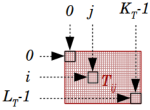
Table of content

- Introduction & Applications.
- Some solutions and their corresponding approach.
  - **Naive solution & Block diagram.**
  - Pixel based approach.
  - Feature based approach.
  - Comparisons.
- Template matching & image registration components.
- Implementation speed-up.

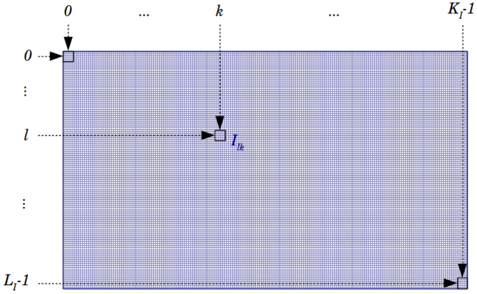
Naive solution I

We consider:

- the image **I** with L_I lines and K_I columns, represented by the matrix $\underline{I} = (I_{lk})$ where $l \in [0, L_I[$, $k \in [0, K_I[$,
- the template **T** with L_T lines and K_T columns, represented by the matrix $\underline{T} = (T_{ij})$ where $i \in [0, L_T[$, $j \in [0, K_T[$.



Template matrix $\underline{\mathbf{T}}$



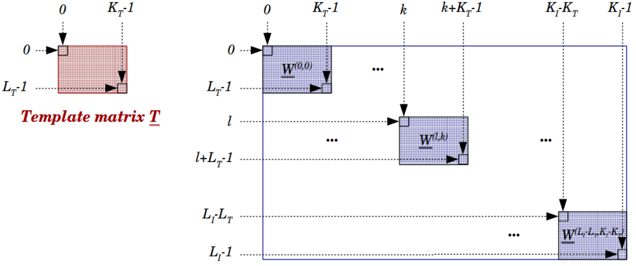
Observed image matrix $\underline{\mathbf{I}}$

Naive solution II

We define all the admissible (sub-)windows $\mathbf{W}^{(\mathbf{l},\mathbf{k})}$ completely included within the image $\mathbf{I}$ and of the same size as the template $\mathbf{T}$ by the following sub-matrices $\underline{W}^{(l,k)}$:

$$W_{ij}^{(l,k)} = \begin{cases} I_{l+i,k+j} & \text{for } i \in [0, L_T - 1]\,,\ j \in [0, K_T - 1] \\ 0 & \text{otherwise} \end{cases} \tag{235}$$

where $l \in [0, L_I - L_T]$ and $k \in [0, K_I - K_T]$ are the indices, in the image $\mathbf{I}$, of the upper left pixel of $\mathbf{W}^{(\mathbf{l},\mathbf{k})}$.



Template matrix $\underline{T}$

Observed image matrix $\underline{I}$

Naive solution III

Compute the Euclidean distance

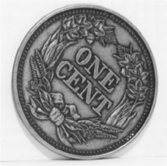
$$\text{dist}\left(\underline{T}, \underline{W}^{(l,k)}\right) = \sum_{i=0}^{L_T-1}\sum_{j=0}^{K_T-1}\left[T_{ij} - W_{ij}^{(l,k)}\right]^2 \tag{236}$$

then create the distance map **D**, represented by the matrix $\underline{D}$:

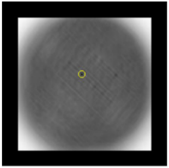
$$D_{\left\lfloor \frac{K_T}{2} \right\rfloor + k, \left\lfloor \frac{L_T}{2} \right\rfloor + l} = \begin{cases} \text{dist}\left(\underline{T}, \underline{W}^{(l,k)}\right) & \text{for } l \in [0, L_I - L_T]\,,\; k \in [0, K_I - K_T] \\ 0 & \text{otherwise} \end{cases} \tag{237}$$

and find the position of the minimum in these map.

Template **T**



Observed image **I**

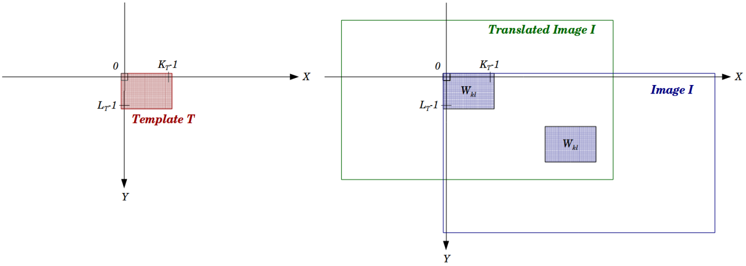


Distance map **D**

Naive solution in pixel coordinates - Image translation

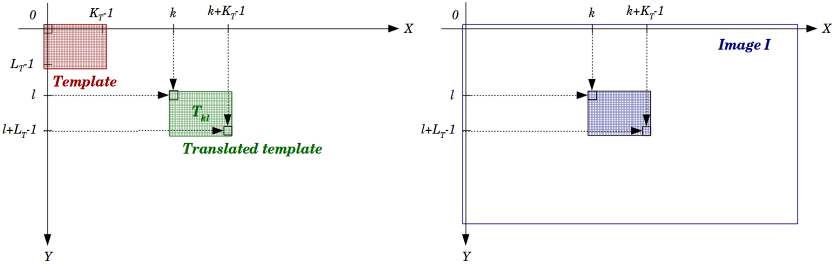
Previously, in the naive solution, we matched the template T to a part of the translated version of the image (the sub-image function $W_{k,l}(x, y) = I(x + k, y + l)$).

In pixel coordinates, this may be illustrated by:



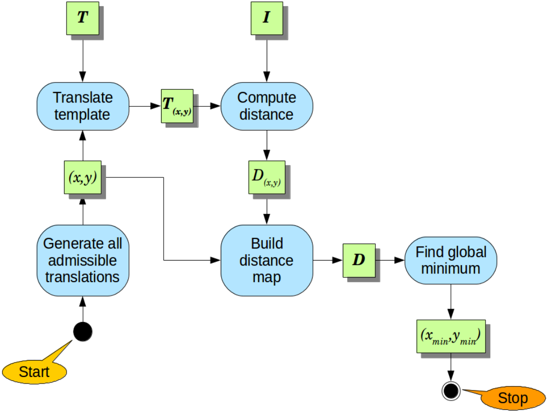
Naive solution in pixel coordinates - Template translation

It is equivalent to match a translated template to the original image:



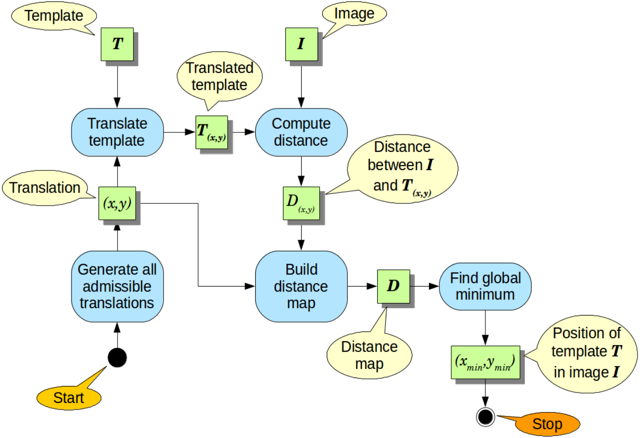
Naive solution - Block diagram I

The block diagram of this pixel based (naive) solution is:



Naive solution - Block diagram II

The block diagram of this pixel based (naive) solution is:

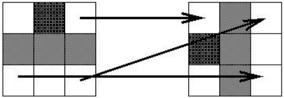


Pixel-based ... ?

Question?

Which “elements” are we going to match in the reference and template images ?

- All pixels of an image/template:



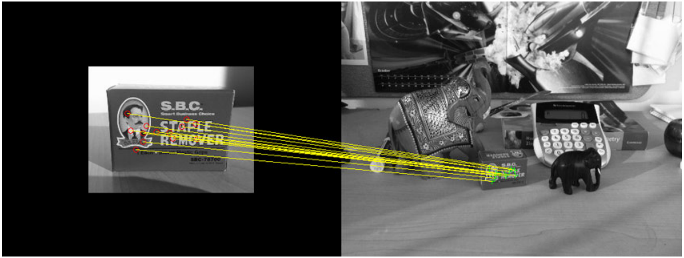
- For all warped templates, compare all pairs of corresponding pixels (≡ located at the same place in the images and the warped template).
- Then compute a global score based on the individuals comparisons.
- And choose the warping for which the score is maximum/minimum

… or Feature-based ?

Question?

Which “elements” are we going to match in the reference and template images ?

- “Interesting” points of an image/template:



- First search for the feature points in the image/template,
- Then find the best matching between feature points in the image and the template
- And finally compute the warping based on the best feature points matching

Table of content

- Introduction & Applications.
- Some solutions and their corresponding approach.
  - Naive solution & Block diagram.
  - **Pixel based approach.**
  - Feature based approach.
  - Comparisons.
- Template matching & image registration components.
- Implementation speed-up.

Pixel-based approach I

Pixel-based approach

Shift or warp the images relatively to each other, then look at how much the pixels agree and find the warping parameters for which the agreement is maximum.

- So, we first need to decide which kind of warping is eligible between the template and the image.
- In the previous naive solution, it is only translation.
- The eligible warping defines the parameter space or the search space:
  - Translation (2D) + rotation (1D) + isotropic scaling (1D) → 4D search space
  - Affine / Projective transform → 6D / 8D search space
- Applying the warping
  - to the pattern
  - or the image
  - or both?
- Image re-sampling and sub-pixel accuracy?

Pixel-based approach II

Pixel-based approach

Shift or warp the images relatively to each other, then look at how much the pixels agree and find the warping parameters for which the agreement is maximum.

- Then a suitable similarity or dissimilarity measure must be chosen to compare the images
- In the previous naive solution, it is the Euclidean distance.
- The similarity or dissimilarity measure depend on the image characteristics to which it is necessary to be invariant. The insensitivity properties guide the choice of a score/distance measure.
    - Lighting conditions (linear gain and offset)
    - Noise
    - “Small” rotation or scaling
    - Thinning
    - → Define the similarity/dissimilarity measure

Pixel-based approach III

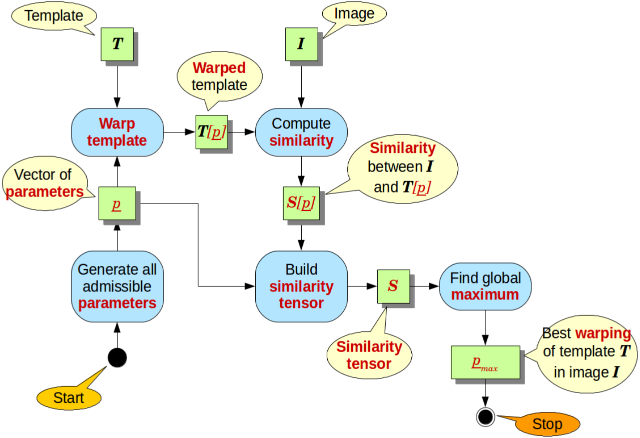
Pixel-based approach

Shift or warp the images relatively to each other, then look at how much the pixels agree and find the warping parameters for which the agreement is maximum.

- In the previous naive solution, we try all possible alignment (an exhaustive search).
- But this solution is often impractical and hierarchical coarse-to-fine techniques based on image pyramids are often used.
- The search technique must be devised:
  - Exhaustive search
  - Coarse to fine hierarchical refinement.
  - Steepest descent
  - Conjugate gradient
  - Quasi-Newton method
  - Levenberg-Marquardt
  - Simulated annealing

General pixel-based solution - Block diagram I

The block diagram of a more general solution could be



General pixel-based solution - Block diagram II

The block diagram of a more general solution could be

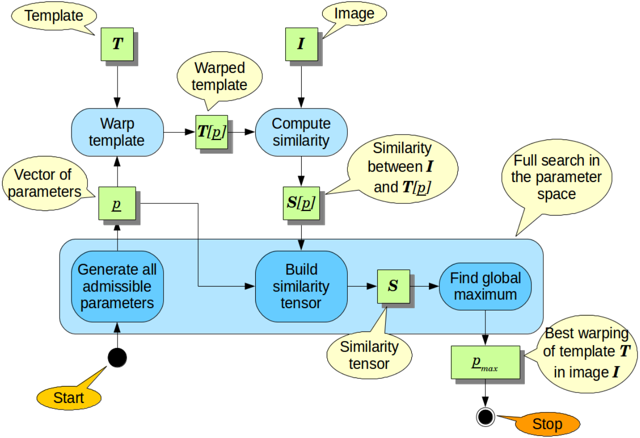

Template
$\boldsymbol{T}$
Image
$\boldsymbol{I}$
Warped template
Warp template
$\boldsymbol{T}[\underline{p}]$
Compute similarity
Similarity between $\boldsymbol{I}$ and $\boldsymbol{T}[\underline{p}]$
Vector of parameters
$\underline{p}$
$\boldsymbol{S}[\underline{p}]$
Full search in the parameter space
Generate all admissible parameters
Build similarity tensor
$\boldsymbol{S}$
Find global maximum
Similarity tensor
$\underline{p}_{max}$
Best warping of template $\boldsymbol{T}$ in image $\boldsymbol{I}$
Start
Stop

Table of content

- Introduction & Applications.
- Some solutions and their corresponding approach.
  - Naive solution & Block diagram.
  - Pixel based approach.
  - **Feature based approach.**
  - Comparisons.
- Template matching & image registration components.
- Implementation speed-up.

Feature-based approach

Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

Let's consider two images of the same (or similar/related) scene/object taken

- at different moment, or
- from a different point of view, or
- with different sensor (parameters).

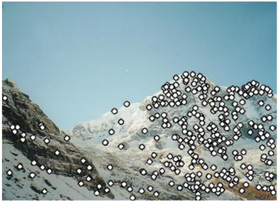
Feature points definition I

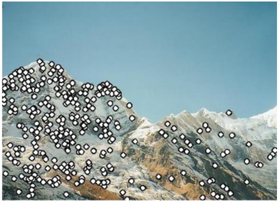
Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

Feature points in each image carry critical information about (local) scene structure.

- Also called critical points, interest points, key points, extremal points, anchor points, landmarks, control points, tie points.
- For instance corners, vertices, junctions, edges, dark/light blob center, unique patches, moments, . . .





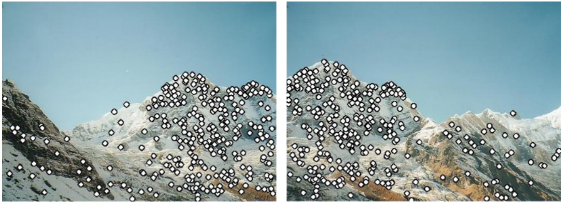
Feature points definition II

Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

Feature points are/should be:

- Independent of noise, blurring, contrast, lightning conditions.
- Dependent or Independent of geometric changes (rotation, scaling, affine transform).
- Widely used in image analysis (not only for image registration).

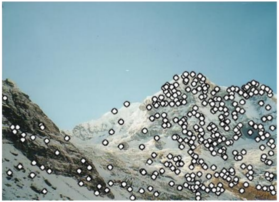


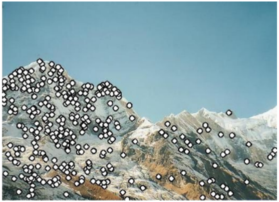
Feature points definition III

Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

- Defining and computing a “featureness” function
  - Invariant to some image variation (translation, rotation, scaling, brightness, contrast, ...)
  - For instance; central moments, pixel intensity variances, gradient module, LoG/DoG, entropy, Harris cornerness, ...
- And keeping only the most-significant maxima/minima/zeroes of the “featureness” function.





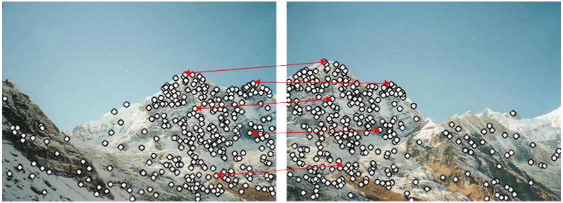
Feature points representation & matching I

Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

Represent each feature point by a descriptor vector

- Vector of characteristic values describing the feature point in an unique and discriminatory way.
- Position, gradient, image moment, scale, orientation, Histogram Of Gradients (HOG), Local Binary Pattern (LPB), ...



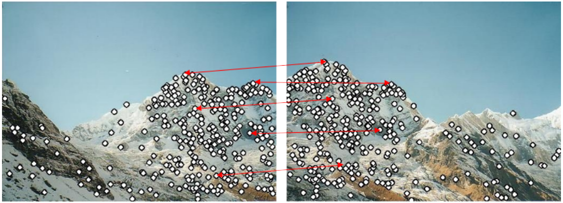
Feature points representation & matching II

Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

Find/match the corresponding feature points in both images:

- Compare the descriptor vectors to find the best correspondence between feature points in each images.



Feature points transformation

Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

Compute the transformation/warping such that the feature points in the left image fit their corresponding point in the right image.

- Define which kind of warping is admissible; rigid global warping (homography) or elastic/local warping.
- Use robust fitting methods:
  - RANSAC,
  - Hough Transform,
  - ICP (Iterative Closest Point)

Table of content

- Introduction & Applications.
- Some solutions and their corresponding approach.
  - Naive solution & Block diagram.
  - Pixel based approach.
  - Feature based approach.
  - **Comparisons.**
- Template matching & image registration components.
- Implementation speed-up.

Pixel-based vs. feature based approach I

Pixel-based approach

Shift or warp the images relatively to each other, then look at how much the pixels agree and find the warping parameters for which the agreement is maximum.

Feature-based approach

In both images, extract feature points and compute their descriptor vector. Then, match the corresponding feature points and compute the image warping that transforms at best (maximum agreement) each feature points into its corresponding one.

Template matching and/or image registration ARE optimization problems

Pixel-based vs. feature based approach II

	Pixel based	Feature based
Pattern information usage	Use all pixels in the pattern in an uniform way. No need to analyze or understand the pattern.	Find and use pattern features (most informative part of the pattern). → Sensitive operation.
Occlusion or pose variation	Sensitive	Could be design to be insensitive
Sub-pixel accuracy	Interpolation of the similarity/dissimilarity measure	Naturally accurate at the sub-pixel level.
Admissible warping	The choice has to be done at the beginning of the process (orientation and scaling)	Mostly insensitive to differences in orientation and scaling
Noise and lighting conditions	Sensitive	Naturally much more insensitive

Pixel-based vs. feature based approach III

	Pixel based	Feature based
Rigid pattern warping	Mostly limited to rigid pattern warping	Enable non-rigid warping.
Dimensionality of the search space	Mostly limited to low dimensionality (the search time is exponential in the search space dimensionality)	Higher dimensionality search space are more easily reachable
Implementation	Easy to implement, natural implementation on GPUs	Much more difficult to implement and/or to optimize
Complexity	Complexity proportional to the image size. Need specific search strategies to reach real-time.	Complexity roughly proportional to the number of feature points (depend more on the content of the scene than on the image size)

Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration
    - Introduction & Applications
    - Some solutions and their corresponding approach
    - Template matching & image registration components
    - Implementation speed-up

Table of content

- Introduction & Applications
- Some solutions and their corresponding approach
- Template matching & image registration components
  - **Pixel-based approach: Similarity/Dissimilarity measures**
  - Feature-based approach: Feature detection
- Implementation speed-up

Pixel-based approach: Similarity/dissimilarity measures

Similarity / Correlation / Score	Dissimilarity / Distance
Pearson correlation coefficient	L1 Norm
Tanimoto Measure	Median of Absolute Differences
Stochastic Sign Change	Square L2 Norm
Deterministic Sign Change	Median of Square Differences
Minimum Ratio	Normalized Square L2 Norm
Spearman’s Rho	Incremental Sign Distance
Kendall’s Tau	Intensity-Ratio Variance
Greatest Deviation	Intensity-Mapping-Ratio Variance
Ordinal Measure	Rank Distance
Correlation Ratio	Joint Entropy
Energy of Joint Probability	Exclusive F–Information
Distribution Material	
Similarity Shannon Mutual Information	
Rényi Mutual Information	
Tsallis Mutual Information	
F-Information Measures	

Pixel-based approach: Similarity/dissimilarity measures

Similarity / Correlation / Score	Dissimilarity / Distance
Pearson correlation coefficient	*L1 Norm*
Tanimoto Measure	*Median of Absolute Differences*
Stochastic Sign Change	Square L2 Norm
Deterministic Sign Change	*Median of Square Differences*
Minimum Ratio	*Normalized Square L2 Norm*
Spearman's Rho	*Incremental Sign Distance*
Kendall's Tau	*Intensity-Ratio Variance*
Greatest Deviation	*Intensity-Mapping-Ratio Variance*
Ordinal Measure	*Rank Distance*
Correlation Ratio	*Joint Entropy*
Energy of Joint Probability	*Exclusive F–Information*
Distribution Material	
Similarity Shannon Mutual Information	
Rényi Mutual Information	
Tsallis Mutual Information	
F-Information Measures	

Pixel-based approach: Similarity/dissimilarity measures

- Given two corresponding sequences of measurement $\{a_i \mid i = 1, \cdots, n\}$ and $\{b_i \mid i = 1, \cdots, n\}$, we will represent them by the following (column) vectors:
  - $\underline{A} = (a_i)_{i=1,\cdots,n} \in \mathbb{R}^n$
  - $\underline{B} = (b_i)_{i=1,\cdots,n} \in \mathbb{R}^n$
  - $\underline{A}$ and $\underline{B}$ might represent measurements from two objects or phenomena. Here, in our case, we assume they represent images and a_i and b_i are the intensities of the corresponding pixels in the images.
- The similarity (dissimilarity) between them is a measure that quantifies the dependency (independency) between the sequences.
- In the naive solution, we used the Euclidean distance (the square L2 norm).

Pearson correlation coefficient

Definition

The Pearson correlation coefficient of two vectors $\underline{A}$ and $\underline{B}$ is:

$$r\left(\underline{A},\underline{B}\right)=\frac{covar\left(\underline{A},\underline{B}\right)}{\sqrt{var\left(\underline{A}\right)var\left(\underline{B}\right)}} \tag{238}$$

With the usual notations:

- $\mu_A=\frac{1}{n}\sum_{i=1}^{n}a_i$ $\qquad var\left(\underline{A}\right)=\sigma_A^2=\frac{1}{n}\sum_{i=1}^{n}\left(a_i-\mu_A\right)^2=\frac{1}{n}\sum_{i=1}^{n}a_i^2-\mu_A^2$
- $\mu_B=\frac{1}{n}\sum_{i=1}^{n}b_i$ $\qquad var\left(\underline{B}\right)=\sigma_B^2=\frac{1}{n}\sum_{i=1}^{n}\left(b_i-\mu_B\right)^2=\frac{1}{n}\sum_{i=1}^{n}b_i^2-\mu_B^2$
- $covar\left(\underline{A},\underline{B}\right)=\frac{1}{n}\sum_{i=1}^{n}\left(a_i-\mu_A\right)\left(b_i-\mu_B\right)=\frac{1}{n}\sum_{i=1}^{n}a_ib_i-\mu_A\mu_B$

We may write:

$$r\left(\underline{A},\underline{B}\right)=\frac{\frac{1}{n}\sum_{i=1}^{n}\left(a_i-\mu_A\right)\left(b_i-\mu_B\right)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(a_i-\mu_A\right)^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(b_i-\mu_B\right)^2}}$$

$$r\left(\underline{A},\underline{B}\right)=\frac{\frac{1}{n}\sum_{i=1}^{n}a_ib_i-\mu_A\mu_B}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}a_i^2-\mu_A^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}b_i^2-\mu_B^2}}$$

The Pearson coefficient $r\left(\underline{A},\underline{B}\right)$ is then easily computed with one pass on the images $\underline{A}$ and $\underline{B}$.

Pearson correlation coefficient properties I

If we introduce the reduced or normalized vectors:

$$\underline{\tilde{A}} = \frac{1}{\sigma_A}(\underline{A} - \mu_A \underline{1}) \text{ and } \underline{\tilde{B}} = \frac{1}{\sigma_B}(\underline{B} - \mu_B \underline{1})$$

We have the following relation:

Theorem

$$r\left(\underline{A}, \underline{B}\right) = \frac{1}{n} \underline{\tilde{A}}^T \underline{\tilde{B}} = r\left(\underline{\tilde{A}}, \underline{\tilde{B}}\right) \tag{239}$$

Which is obvious from

$$r\left(\underline{A}, \underline{B}\right) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{a_i - \mu_A}{\sigma_A}\right) \left(\frac{b_i - \mu_B}{\sigma_B}\right) = \frac{1}{n} \underline{\tilde{A}}^T \underline{\tilde{B}} = r\left(\underline{\tilde{A}}, \underline{\tilde{B}}\right)$$

Pearson correlation coefficient properties II

The Pearson coefficient doesn't depend on the image's gain and offset.

Theorem (Invariance to affine transformation of pixel values)

For any values $\alpha, \beta, \gamma, \delta$ such that $\alpha \neq 0$ and $\gamma \neq 0$, we have:

$$r\left(\alpha\underline{A} + \beta\underline{1}, \gamma\underline{B} + \delta\underline{1}\right) = sign\left(\alpha\gamma\right) r\left(\underline{A}, \underline{B}\right)$$

This result comes immediately from

$$(\alpha\widetilde{\underline{A} + \beta}\underline{1}) = \frac{1}{|\alpha|\,\sigma_A}\left[\left(\alpha\underline{A} + \beta\underline{1}\right) - \left(\alpha\mu_A + \beta\right)\underline{1}\right] = \frac{sign\left(\alpha\right)}{\sigma_A}\left(\underline{A} - \mu_A\underline{1}\right) = sign\left(\alpha\right)\underline{\tilde{A}}$$

Then

$$r\left(\alpha\underline{A} + \beta\underline{1}, \gamma\underline{B} + \delta\underline{1}\right) = r\left((\alpha\widetilde{\underline{A} + \beta}\underline{1}), (\gamma\widetilde{\underline{B} + \delta}\underline{1})\right)$$

$$r\left(\alpha\underline{A} + \beta\underline{1}, \gamma\underline{B} + \delta\underline{1}\right) = r\left(sign\left(\alpha\right)\underline{\tilde{A}}, sign\left(\gamma\right)\underline{\tilde{B}}\right) = sign\left(\alpha\gamma\right) r\left(\underline{A}, \underline{B}\right)$$

Pearson correlation coefficient properties III

Due to the usual properties of variance and covariance;

Theorem

The range of values of $r\left(\underline{A}, \underline{B}\right)$ *is* $[-1, +1]$*:*

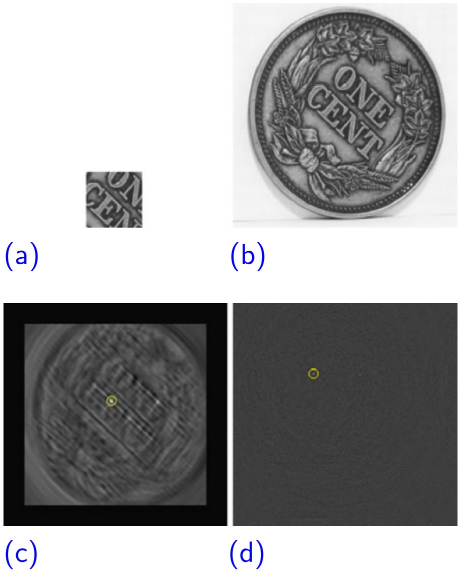
$r = +1$ *if and only if* $\underline{B} = \alpha\underline{A} + \beta\underline{1}$ *with* $\alpha > 0$*. It is a perfect direct matching between* $\underline{A}$ *and* $\underline{B}$ *.*

$r = -1$ *if and only if* $\underline{B} = \alpha\underline{A} + \beta\underline{1}$ *with* $\alpha < 0$*. It is a perfect inverse matching between* $\underline{A}$ *and* $\underline{B}$ *.*

$r = 0$ *if and only if there is no linear correlation between* $\underline{A}$ *and* $\underline{B}$*.*

Pearson correlation coefficient map

(a) A template T.
(b) An image I containing the template T.
(c) The correlation image $C\left[T, I\right]$ with intensity at a pixel showing the correlation coefficient between the template and the window centered at the pixel in the image.
(d) The real part of image $C_p\left[T, I\right]$, showing the phase correlation result with the location of the spike encircled.


(a) (b)

(c) (d)

Spearman rank correlation or Spearman's rho I

- The Spearman rank correlation or Spearman's Rho (ρ) between vectors $\underline{A} = (a_i)_{i=1,\cdots,n}$ and $\underline{B} = (b_i)_{i=1,\cdots,n}$ is given by

$$\rho = 1 - \frac{6\sum_{i=1}^{n}\left[R\left(a_i\right) - R\left(b_i\right)\right]^2}{n\left(n^2 - 1\right)} \tag{240}$$

  where $R\left(a_i\right)$ and $R\left(b_i\right)$ represent ranks of a_i and b_i in images $\underline{A}$ and $\underline{B}$.

- Remark: to eliminate possible ties among discrete intensities in images, the images are smoothed with a Gaussian of a small standard deviation, such as 1 pixel, to produce unique floating-point intensities.

Spearman rank correlation or Spearman's rho II

- Comparison with the Pearson correlation coefficient:
  - ρ is less sensitive to outliers and, thus, less sensitive to impulse noise and occlusion.
  - ρ is less sensitive to nonlinear intensity difference between images than Pearson correlation coefficient.
  - Spearman's ρ consistently produced a higher discrimination power than Pearson correlation coefficient.
  - Computationally, ρ is much slower than r primarily due to the need for ordering intensities in I and J .

Kendall's tau I

- If a_i and b_i, for $i = 0, ..., n$, show intensities of corresponding pixels in $\underline{A}$ and $\underline{B}$, then for $i \neq j$, two possibilities exist:
  - Either concordance : $sign(a_j - a_i) = sign(b_j - b_i)$
  - Or discordance : $sign(a_j - a_i) = -sign(b_j - b_i)$
- Assuming that out of the C_n^2 possible combinations, N_c pairs are concordant and N_d pairs are discordant, Kendall's τ is defined by:

$$\tau = \frac{N_c - N_d}{\frac{n(n-1)}{2}} \tag{241}$$

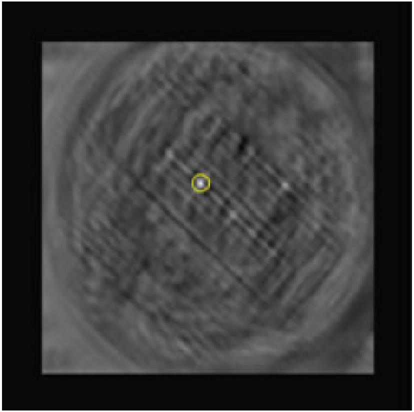
- If bivariate $(\underline{A}, \underline{B})$ is normally distributed, Kendall's τ is related to Pearson correlation coefficient r by:

$$r = \sin\left(\frac{\pi\tau}{2}\right) \tag{242}$$

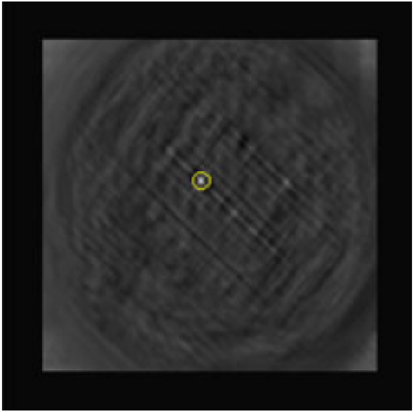
Kendall's tau II

- Comparison with other similarity measures:
  - Pearson correlation coefficient can more finely distinguish images that represent different scenes than Kendall's τ .
  - Conversely, Kendall's τ can more finely distinguish similar images from each other when compared to Pearson correlation coefficient.
  - Spearman's ρ and Kendall's τ have the same discrimination power when comparing images of different scenes.
  - Kendall's τ is one of the costliest similarity measures.

Spearman's rho and Kendall's tau maps



Spearman's Rho



Kendall's Tau

Table of content

- Introduction & Applications
- Some solutions and their corresponding approach
- Template matching & image registration components
  - Pixel-based approach: Similarity/Dissimilarity measures
  - **Feature-based approach: Feature points detection**
- Implementation speed-up

Feature point category

A large number of point detectors have been developed throughout the years:

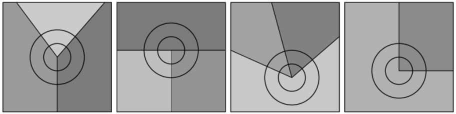
- Corner-based detectors
- Edge-based detectors
- Model-based detectors
- Uniqueness-based detectors
- Curvature-based detectors
- Laplacian-based detectors
- Gradient-based detectors
- Hough Transform-based detectors
- Symmetry-based detectors
- Filtering-based detectors
- Transform Domain detectors
- Pattern Recognition-based detectors
- Moment-based detectors
- Entropy-based detectors

Feature point category

A large number of point detectors have been developed throughout the years:

- Corner-based detectors
- Edge-based detectors
- Model-based detectors
- Uniqueness-based detectors
- Curvature-based detectors
- Laplacian-based detectors
- Gradient-based detectors
- Hough Transform-based detectors
- Symmetry-based detectors
- Filtering-based detectors
- Transform Domain detectors
- Pattern Recognition-based detectors
- Moment-based detectors
- Entropy-based detectors

Corner-based detectors I

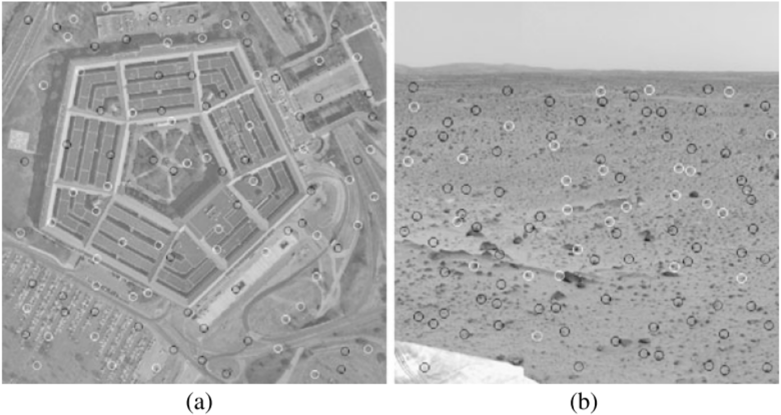


- The angle between the line connecting pixel (x, y) to the ith pixel on the smallest circle and the x-axis is θ_i, and the intensity at the ith pixel is $I_1(\theta_i)$
- If $\tilde{I}_j(\theta_i)$ represents the normalized intensity at θ_i in the jth circle, then

$$C(x, y) = \sum_{i=1}^{n} \prod_{j=1}^{m} \tilde{I}_j(\theta_i) \tag{243}$$

  is used to measure the strength of a vertex or a junction at (x, y) .
- In the following formula, if $m = 2$ (2 circles), $C(x, y)$ is the Pearson coefficient of the two vectors $\tilde{I}_1(\theta_i)$ and $\tilde{I}_2(\theta_i)$.
- Pixel (x, y) is then considered a **corner** if $C(x, y)$ is locally maximum.

Corner-based detectors II



Laplacian-based detectors I

- A number of detectors use either the Laplacian of Gaussian (LoG) or the difference of Gaussians (DoG) to detect points in an image.
- For the following development, we consider a "continuous" image

$$I : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R} : (x,y) \in \Omega \rightarrow I(x,y)\,, \tag{244}$$

and we define a scaled version of this image

$$L(x,y;\sigma) = g(x,y;\sigma) \otimes I(x,y)\,, \tag{245}$$

where $g(x,y;\sigma) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$ is the Gaussian kernel of variance σ^2.

- The Laplacian of Gaussian $LoG = \triangle L(x,y;\sigma)$ is defined as

$$\triangle L(x,y;\sigma) = \triangle\left[G(x,y;\sigma) \otimes I(x,y)\right] = \left[\triangle g(x,y;\sigma)\right] \otimes I(x,y)\,. \tag{246}$$

Laplacian-based detectors II

- So, we compute the Laplacian of the Gaussian

$$\triangle g\left(x,y;\sigma\right)=\frac{x^2+y^2-2\sigma^2}{2\pi\sigma^6}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{247}$$

and its partial derivative relatively to σ

$$\frac{\partial}{\partial\sigma}g\left(x,y;\sigma\right)=\frac{x^2+y^2-2\sigma^2}{2\pi\sigma^5}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right). \tag{248}$$

Then, we deduce

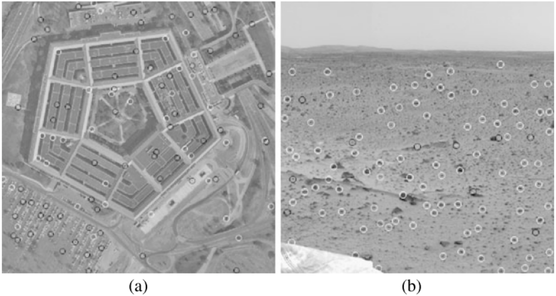
$$\sigma\triangle L\left(x,y;\sigma\right)=\frac{\partial}{\partial\sigma}L\left(x,y;\sigma\right)\simeq\frac{\left[L\left(x,y;k\sigma\right)-L\left(x,y;\sigma\right)\right]}{k\sigma-\sigma} \tag{249}$$

and the DoG operator is an approximation to the LoG operator

$$L\left(x,y;k\sigma\right)-L\left(x,y;\sigma\right)=\left(k-1\right)\sigma^2\triangle L\left(x,y;\sigma\right). \tag{250}$$

Laplacian-based detectors III

- Local extrema of LoG or its approximation DoG detect **centers of bright or dark blobs** in an image.
  - They are less influenced by noise than points representing corners and junctions
  - They are stable and discriminative for image matching.
- SIFT (Scale Invariant Feature Transform) used the difference of Gaussians (DoG) to find points in an image.



Outline

1. Fundamentals of 2D imaging
2. Linear filtering
3. 3D vision: calibration and reconstruction
4. Mathematical morphology
5. Non-linear filtering
6. Object description and analysis
7. Motion analysis and background subtraction
8. Performance analysis
9. Edge detection
10. Feature detection and tracking
11. Introduction to machine learning
12. Fundamentals of deep learning for visual understanding
13. Deep learning architectures for image analysis
14. Segmentation
15. Template Matching & Image Registration
    - Introduction & Applications
    - Some solutions and their corresponding approach
    - Template matching & image registration components
    - Implementation speed-up

Table of content

- Introduction & Applications
- Some solutions and their corresponding approach
- Template matching & image registration components
- Implementation speed-up
  - FFT
  - Multiresolution: Coarse to fine
  - Hybrid approach: feature extraction in one image only

Correlation and Fourier Transform I

The 2D Discrete Fourier Transform (2D-DFT) enable the fast computation of the cross-correlation of images.

Definition

A discrete image of L lines and K columns is first represented by:
the matrix of its pixel values $\underline{I} = (I_{lk})$
where $l = 0, \cdots, L-1$ and $k = 0, \cdots, K-1$,
and then, we extend the image by defining $a_{lk} = 0$
when $l \notin [0, L-1]$ or $k \notin [0, K-1]$

Correlation and Fourier Transform II

Definition (2D-DFT on a grid of L lines by K columns)

The 2D-DFT of $\underline{I}$ is the matrix

$$\underline{\mathcal{I}} = \mathcal{F}\left[\underline{I}\right] = (\alpha_{uv})$$

where $u = 0, \cdots, L-1$ and $v = 0, \cdots, K-1$, and

$$\alpha_{uv} = \sum_{l=0}^{L-1}\sum_{k=0}^{K-1} I_{lk} e^{-j2\pi\left(\frac{lu}{L}+\frac{kv}{K}\right)} = \sum_{l,k\in\mathbb{Z}} I_{lk} e^{-j2\pi\left(\frac{lu}{L}+\frac{kv}{K}\right)}$$

And the inverse formulas are

$$I_{lk} = \frac{1}{LK}\sum_{u=0}^{L-1}\sum_{v=0}^{K-1} \alpha_{uv} e^{j2\pi\left(\frac{lu}{L}+\frac{kv}{K}\right)}$$

Correlation and Fourier Transform III

In the previous definition, we extend the range of values of u and v and obtain a periodic “infinite” array, thanks to the following relation:

$$\alpha_{u+L,v+K} = \sum_{l,k\in\mathbb{Z}} I_{lk} e^{-j2\pi\left(\frac{l(u+L)}{L}+\frac{k(v+K)}{K}\right)}$$

$$\alpha_{u+L,v+K} = \sum_{l,k\in\mathbb{Z}} I_{lk} e^{-j2\pi\left(\frac{lu}{L}+\frac{kv}{K}\right)} \overbrace{e^{-j2\pi(l+k)}}^{=1} = \alpha_{uv}$$

So, we define α_{uv} for all integral values of u and v by $\alpha_{u+nL,v+mK} = \alpha_{uv}$.

Correlation and Fourier Transform IV

Definition

The cross-correlation of two “extended” images $\underline{I}$ (initially L_I lines by K_I columns) and $\underline{T}$ (initially L_T lines by K_T columns) is

$$\Gamma\left(\underline{I}, \underline{T}\right) = \underline{C} = \left(c_{lk}\right)$$

where

$$c_{lk} = \sum_{n=0}^{L_I-1}\sum_{m=0}^{K_I-1} I_{nm} T_{n+l,m+k} = \sum_{n,m\in\mathbb{Z}} I_{nm} T_{n+l,m+k}$$

with $c_{lk} \neq 0$ if $l \in \{-L_I + 1, \cdots, L_T - 1\}$ and
$k \in -\{K_I + 1, \cdots, K_T - 1\}$,
and $L_C = L_I + L_T - 1$ and $K_C = K_I + K_T - 1$.

Correlation and Fourier Transform V

The 2D-DFT of the cross-correlation of these two images is

$$\gamma_{uv} = \sum_{l,k\in\mathbb{Z}} c_{lk}\, e^{-j2\pi\left(\frac{lu}{L_C}+\frac{kv}{K_C}\right)}$$

$$\gamma_{uv} = \sum_{l,k\in\mathbb{Z}} \left[\sum_{n,m\in\mathbb{Z}} I_{nm}\, T_{n+l,m+k}\right] e^{-j2\pi\left(\frac{lu}{L_C}+\frac{kv}{K_C}\right)}$$

$$\gamma_{uv} = \sum_{n,m\in\mathbb{Z}} I_{nm} \sum_{l,k\in\mathbb{Z}} T_{n+l,m+k}\, e^{-j2\pi\left(\frac{lu}{L_C}+\frac{kv}{K_C}\right)}$$

$$\gamma_{uv} = \sum_{n,m\in\mathbb{Z}} I_{nm} \sum_{l',k'\in\mathbb{Z}} T_{l'k'}\, e^{-j2\pi\left(\frac{\left(l'-n\right)u}{L_C}+\frac{\left(k'-m\right)v}{K_C}\right)}$$

$$\gamma_{uv} = \left(\sum_{n,m\in\mathbb{Z}} I_{nm} e^{+j2\pi\left(\frac{nu}{L_C}+\frac{mv}{K_C}\right)}\right)\left(\sum_{l',k'\in\mathbb{Z}} T_{l'k'}\, e^{-j2\pi\left(\frac{l'u}{L_C}+\frac{k'v}{K_C}\right)}\right)$$

$$\gamma_{uv} = \alpha^*_{uv}\beta_{uv}$$

Correlation and Fourier Transform VI

- The best-matching template window in the image is located at the peak of the cross-correlation

$$\underline{C}\left[\underline{T},\underline{I}\right]=\frac{1}{n}\Gamma\left(\underline{T},\underline{I}\right)=\frac{1}{n}\mathcal{F}^{-1}\left\{\mathcal{F}\left\{\underline{T}\right\}^{*}\mathcal{F}\left\{\underline{I}\right\}\right\} \tag{251}$$

- This is different from the Pearson correlation coefficient, where each sub-window of the image $\underline{I}$ is first normalized before being correlated.
- Phase correlation: the information about the displacement of one image with respect to another is included in the phase component of the cross-power spectrum of the images:

$$\underline{C_p}\left[\underline{T},\underline{I}\right]=\mathcal{F}^{-1}\left\{\frac{\mathcal{F}\left\{\underline{T}\right\}^{*}\mathcal{F}\left\{\underline{I}\right\}}{\left\|\mathcal{F}\left\{\underline{T}\right\}^{*}\mathcal{F}\left\{\underline{I}\right\}\right\|}\right\} \tag{252}$$

Multiresolution - Coarse-to-fine approach

- Compute image and pattern down-scaled pyramids.
- Proceed to a full search of the most reduced (coarser) pattern within the most reduced image.
- Find a number of possible candidates at the coarsest scale by an exhaustive search.
- For each candidates at a given scale:
  - Upscale the image and the candidate and look for the best matching pattern location in a neighborhood of the candidate.
  - Reduce the number of candidates
  - If the finer scale has not yet been reached, proceed to the next scale level

Hybrid approach: feature extraction in one image only

- Search for some feature points in the template.
- Consider very simple feature points as
    - the local maxima of the gradient,
    - regularly spaced patches.
- Scan the warping parameter space following a given strategy:
    - Transform the feature points of the template following the current eligible warping parameters.
    - Superimpose the warped feature points (of the template) on the observed image.
    - At each warped feature points location in the observed image, check if a compatible feature point exists in the observed image and measure its similarity/dissimilarity score.
    - Compute a global measure of similarity/dissimilarity by adding all the individual scores of the feature points.
    - Find the optimum of this measure on the search space.
- “Detect and track” instead of “detect and match”.

Bibliography

A. Goshtasby.
Image registration – Principles, Tools an Methods. Springer-Verlag London, 2012, DOI: 10.1007/978-1-4471-2458-0

R. Brunelli.
Template matching techniques in computer vision – Theory and practice. John Wiley & Sons, 2009.

R. Szeliski.
Image alignment and stitching: a tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, 2006