
Lê Đức Khải

Computer Vision Summary

Written on October 18, 2019

Last updated on December 04, 2019

CONTENTS:

A. Image Processing

1. Basic Image Processing

- 1.1. Spatial transformation
- 1.2. Fourier transform
- 1.3. Convolution

2. Image Enhancement

- 2.1. Denoising
- 2.2. Deblurring
- 2.3. Anisotropic diffusion

3. Feature Extraction

- 3.1. Edge detection
- 3.2. Canny edge detector
- 3.3. Hough transform

4. Image Segmentation

- 4.1. Morphological method
- 4.2. Thresholding method
- 4.3. Watershed
- 4.4. Region growing
- 4.5. Snakes
- 4.6. Level sets
- 4.7. K-Means clustering

5. Image Registration

- 5.1. asdf
- 5.2. asdf

6. Image Reconstruction

- 6.1. asdf
- 6.2. asdf

B. Deep Learning Models

1. Object Detection

- 1.1. YOLO

2. Face verification and recognition

- 2.1. Overview

A. Image Processing

1. Basic Image Processing

1.1. Spatial transformation

- Translation: $\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \vec{v} = \begin{pmatrix} x + t_x \\ y + t_y \end{pmatrix}$
- Rotation: $\begin{pmatrix} u \\ v \end{pmatrix} = x \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix} + y \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
- Scaling: $\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \vec{s} = \begin{pmatrix} x \cdot s_x \\ y \cdot s_y \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
- Shearing: $\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 1 & v_x \\ v_y & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
- Converting transformation between two coordinates:

$$M_B = P_{AB} M_A P_{BA} = P_{BA}^{-1} M_A P_{BA}$$

P_{AB} : change-of-basis matrix that converts from A to B coordinates

P_{BA} : change-of-basis matrix that converts from B to A coordinates

M_A : transformation matrix applied in A coordinates.

1.2. Fourier transform

- Fourier series in theory: $f(x) = a_0 + \sum_{k=1}^{\infty} \left[a_k \cos\left(\frac{2\pi kx}{N}\right) + b_k \sin\left(\frac{2\pi kx}{N}\right) \right]$
- Fourier series in practice: $f(x) = a_0 + \sum_{k=1}^m \left[a_k \cos\left(\frac{2\pi kx}{N}\right) + b_k \sin\left(\frac{2\pi kx}{N}\right) \right]$
- Continuous Fourier transform: $F(w) = \mathcal{F}\{f(x)\}(w) = \int_{-\infty}^{\infty} f(x) \cdot e^{-2\pi i w x} dx$
- Discrete Fourier transform (DFT): $F_k = \sum_{n=0}^{N-1} f_n \cdot e^{\frac{-2\pi i n k}{N}}$ with $k = 0, \dots, N-1$
- Fast Fourier transform (FFT): $\delta(w) = \int_{-\infty}^{\infty} e^{2\pi i w x} dx$
- 2D Fourier transform:

$$F(w, \lambda) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \cdot e^{-2\pi i (w x + \lambda y)} dx dy = \int \int f(x, y) \cdot e^{-2\pi i w x} dx \cdot e^{-2\pi i \lambda y} dy$$

1.3. Convolution

- Convolution definition: $(f * g)(x) = \int_{-\infty}^{\infty} f(t) \cdot g(x - t) dt$
- Convolution for discrete signals f_n and g_n : $(f * g)_n = \sum_{k=0}^{N-1} f_k \cdot g_{n-k}$ with $n = 0, \dots, N-1$
- Convolution of Fourier transforms:

$$\mathcal{F}\{(f * g)(x)\}(w) = \mathcal{F}\{f(x)\}(w) \cdot \mathcal{F}\{g(x)\}(w) = F(w) \cdot G(w)$$

2. Image Enhancement

2.1. Denoising

- Mean filter:
- Median filter:
- Gaussian filter:
- Gabor filter:

2.2. Deblurring

3. Feature Extraction

3.1. Sadf

3.2. Adsf

3.3. Hough transform

- Line Hough transform:
 - ❖ Angle-distance parameter space:

$$d = x \cos(\theta) + y \sin(\theta)$$

where:

θ : angle from origin to the line (0° to 180° by default)

d : distance from origin to the line, maximal distance is the diagonal length of the image

❖ Algorithm:

- Corner or edge detection (using Canny, Sobel, adaptive thresholding...)
- Initialize accumulator $H[d, \theta] = 0$
- For each edge point in $E[x, y]$:
 - For $\theta = 0^\circ$ to 180° :
 - $d = x \cos(\theta) + y \sin(\theta)$
 - $H[d, \theta] += 1$
- Find the value(s) of (d, θ) where $H[d, \theta]$ is maximum
- The detected line in the image is given by: $d = x \cos(\theta) + y \sin(\theta)$

❖ Library:

- Scikit-image: [skimage.transform.probabilistic_hough_line](#)
- MATLAB: [houghlines](#)

- Circle Hough transform:

❖ Polar coordinates of circle center:

$$\begin{cases} a = x - r \cos(\theta) \\ b = y - r \sin(\theta) \end{cases}$$

where:

a, b : coordinates of circle center $I(a, b)$

θ : angle between radius and x-axis

❖ Algorithm:

- Corner or edge detection (using Canny, Sobel, adaptive thresholding...)
- For each edge point in $E[x, y]$:
 - For each possible value of radius r :
 - For each angle θ :
 - $a = x - r \cos(\theta)$
 - $b = y - r \sin(\theta)$
 - Voting matrix: $H[a, b, r] += 1$
- Find the value(s) of (a, b, r) where $H[a, b, r]$ is maximum

- ❖ Library:
 - Scikit-image: [skimage.transform.hough_circle](#), [skimage.transform.hough_circle_peaks](#)
 - MATLAB: [imfindcircles](#)

4. Image Segmentation

4.1. Asdf

4.2. Asdf

4.3. Watershed

- Watershed transformation:
 - ❖ Principle: Any greytone image can be considered as a topographic surface. If we flood this surface from its minima and, if we prevent the merging of the waters coming from different sources, we partition the image into two different sets: the catchment basins and the watershed lines. Watershed is often used to segment or detect overlapping objects.
 - ❖ Algorithm:
 - Convert the input image to binary image with thresholding methods
 - Apply distance transformation to the image
 - Apply watershed function. The watershed lines of the output matrix are labeled by zero-valued elements. We can detect or segment objects by taking advantage of these elements.
 - ❖ Library:
 - Scikit-image:
 - MATLAB:
 - Marker-controlled watershed:
 - ❖ Principle: Applying morphological methods to the image can avoid over-segmenting.
 - ❖ Algorithm:
 - Apply morphological methods to the input image
 - Binarize the image by thresholding
 - Apply distance transformation to the image
 - Apply watershed function. The watershed lines of the output matrix are labeled by zero-valued elements. We can detect or segment objects by taking advantage of these elements.
- 4.4. Asdf
- 4.5. asdf
- 4.6. Level sets
- Chan-Vese active contours without edges:
 - ❖ Energy functional:

$$F(c_1, c_2, C) = \mu \cdot \text{Length}(C) + v \cdot \text{Area}(\text{inside}(C)) + \lambda_1 \int_{\text{inside}(C)} |u_0(x, y) - c_1|^2 dx dy \\ + \lambda_2 \int_{\text{outside}(C)} |u_0(x, y) - c_2|^2 dx dy$$

where:

$\mu, v \geq 0$: fixed parameters ($\mu = 0.25, v = 0$ by default)

$\lambda_1, \lambda_2 > 0$: fixed parameters ($\lambda_1 = \lambda_2 = 1$ by default)

C : contour

$u_0(x, y)$: position x, y of the image u_0

c_1 : average pixels' intensity inside C

c_2 : average pixels' intensity outside C

❖ Library:

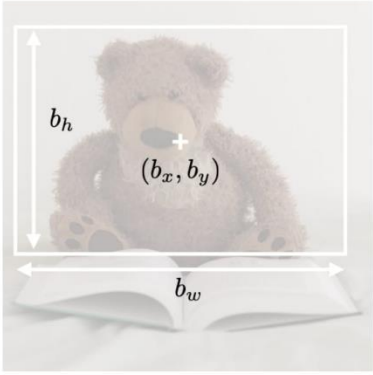
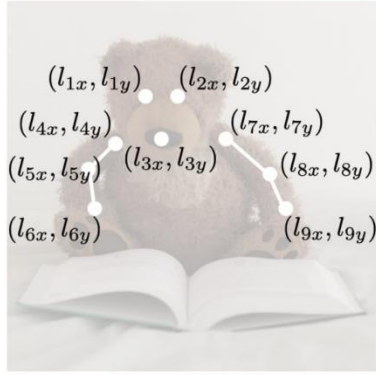
- Scikit-image:
- MATLAB:

B. Deep Learning Models

1. Object Detection

1.1. YOLO

- Data labeling:

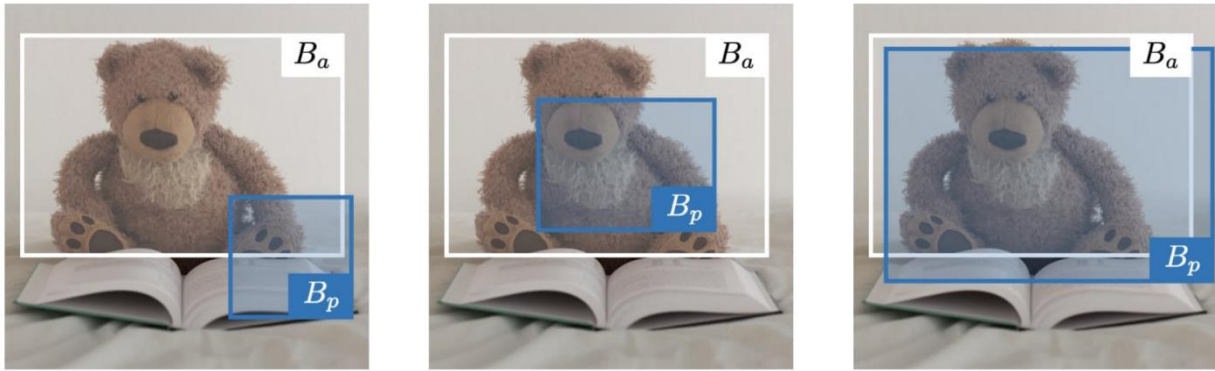
Bounding box detection	Landmark detection
<ul style="list-style-type: none"> • Detects the part of the image where the object is located 	<ul style="list-style-type: none"> • Detects a shape or characteristics of an object (e.g. eyes) • More granular
	
Box of center (b_x, b_y) , height b_h and width b_w	Reference points $(l_{1x}, l_{1y}), \dots, (l_{nx}, l_{ny})$

- Intersection over Union (IoU): quantifies how correctly positioned a predicted bounding box B_p is over the actual bounding box B_a

$$IoU(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

Remarks: $IoU \in [0, 1]$

$IoU \geq 0.5$ is considered as good



$$\text{IoU}(B_p, B_a) = 0.1$$

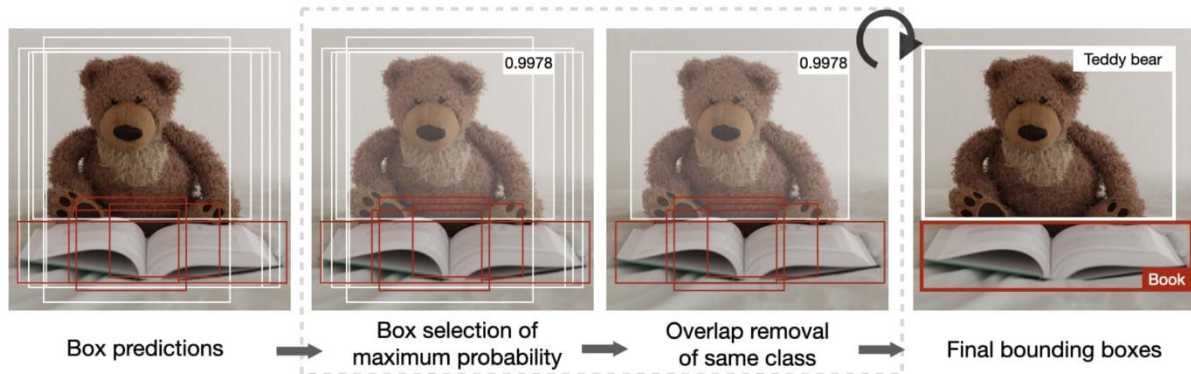
$$\text{IoU}(B_p, B_a) = 0.5$$

$$\text{IoU}(B_p, B_a) = 0.9$$

- Anchor boxes: Anchor boxing is a technique used to predict overlapping bounding boxes. In practice, the network is allowed to predict more than one box simultaneously, where each box prediction is constrained to have a given set of geometrical properties. For instance, the first prediction can potentially be a rectangular box of a given form, while the second will be another rectangular box of a different geometrical form.
- Non-max suppression: is a technique to remove duplicate overlapping bounding boxes of a same object by selecting the most representative ones.

❖ Algorithm:

- Each output prediction is $[p_c \ b_x \ b_y \ b_h \ b_w]^T$. Discard all boxes with $p_c \leq 0.6$.
- While there are any remaining boxes:
 - Pick the box with the largest p_c output that as a prediction
 - Discard any remaining box with $\text{IoU} \geq 0.5$ with the box output in the previous step



- YOLO algorithm:
 - Step 1: Divide the input image into a $G \times G$ grid
 - Step 2: For each grid cell, run a CNN that predicts y of the following form:

$$y = \underbrace{[p_c, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_p, \dots]}_{\text{repeated k times}}^T \in R^{G \times G \times k \times (5+p)}$$

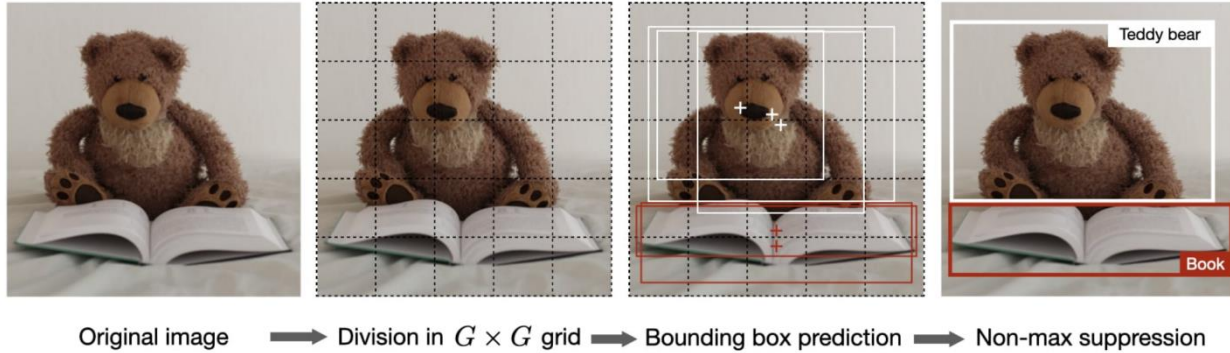
where: p_c is the probability of detecting an object

b_x, b_y, b_h, b_w are the properties of the detected bounding box

c_1, c_2, \dots, c_p is a one-hot representation of which of the p classes were detected

k is the number of anchor boxes

- Step 3: Run the non-max suppression algorithm to remove any potential duplicate overlapping bounding boxes.



2. Face verification and recognition

2.1. Overview

- Types of models:

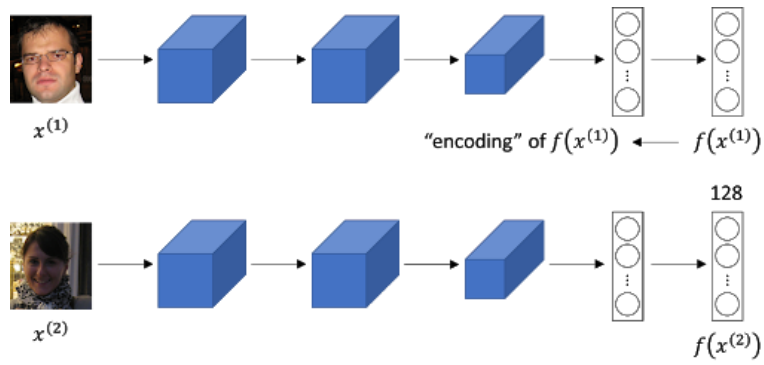
Face verification	Face recognition
<ul style="list-style-type: none"> Is this the correct person? One-to-one lookup 	<ul style="list-style-type: none"> Is this one of the K persons in the database? One-to-many lookup
<p>Query</p> <p>Reference</p>	<p>Query</p> <p>Database</p>

- One-shot learning: is a face verification algorithm that uses a limited training set to learn a similarity function that quantifies how different two given images are:

Degree of differences between images = $d(\text{image 1, image 2})$

$$d(\text{image 1, image 2}) = \begin{cases} \text{"same person"}, & d \leq \tau \\ \text{"different person"}, & d > \tau \end{cases}$$

- Siamese Network: aims at learning how to encode images to then quantify how different two images are. For a given input image $x^{(i)}$, the encoded output is often noted as $f(x^{(i)})$.



$$d(x^{(1)}, x^{(2)}) = \|f(x^{(1)}) - f(x^{(2)})\|_2^2$$

- Triplet loss: The triplet loss l is a loss function computed on the embedding representation of a triplet of images A (anchor), P (positive) and N (negative). By calling $\alpha \in \mathbb{R}^+$ the margin parameter, this loss is:

$$l(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$

