

Image Segmentation

Siyue Yu

siyue.yu02@xjtlu.edu.cn

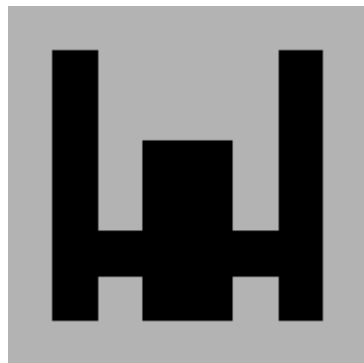
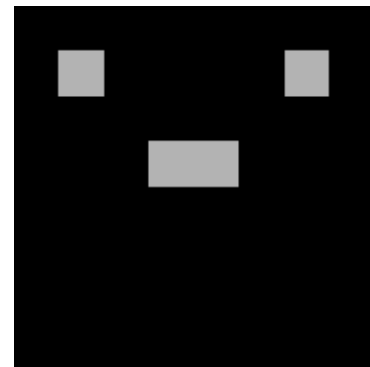
Announcements

- Finish MQ
 - Please kindly be reminded that you are encouraged to provide your feedback to the module INT302 through the Module Feedback Questionnaire.
 - Please take your time to help improving the quality of this module by sharing your comments.

Shape detection

hit-and-miss

- Local maximums have **upper neighbors** in the BG → **we need to detect them**



Why we need hit-and-miss

- Problems with shape detection using **Opening** !
 - The **problem** here is that we are **only** looking at the **foreground**, but we **didn't** consider the **background**



Why we need hit-and-miss

opening

Input

Eroded

Dilated

SE

	1	
	1	
	1	

	1	1	1	1	1		1	
			1				1	
			1				1	
			1				1	
			1				1	

			1				1	
			1				1	
			1				1	
			1				1	

			1				1	
			1				1	
			1				1	
			1				1	
			1				1	

Why we need hit-and-miss

	1	
	1	
	1	

SE1

	1	1	1	1	1		1	
			1				1	
			1				1	
			1				1	
			1				1	

			1				1	
			1				1	
			1				1	

							1	
							1	
							1	
							1	
							1	
							1	

↑ Dilation
SE1

1		1
1		1
1		1
1		1
1		1
1		1
1		1

SE2

1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1						1		1
1	1	1		1	1	1		1
1	1	1		1	1	1		1
1	1	1		1	1	1		1
1	1	1		1	1	1		1
1	1	1	1	1	1	1	1	1

							1	
							1	
							1	
			1					

							1	
							1	
							1	

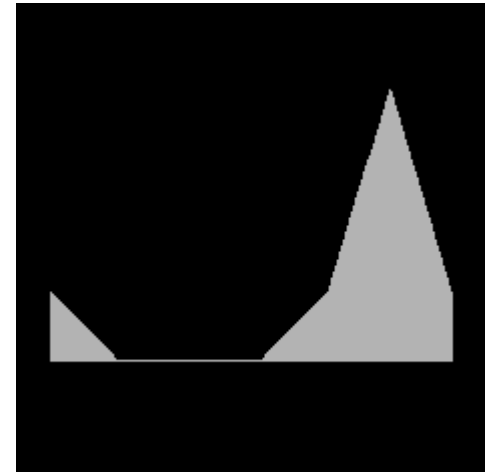
Difference between Erosion and Thinning

SE FG

0	0	0
1	1	1
1	1	1

SE BG

1	1	1
0	0	0
0	0	0



One pixel width

Outline

- Introduction
 - Methods
 - Traditional Methods
 - CNN based
 - Transformer based
 - Language model based
 - Review
- } Next lecture

Introduction

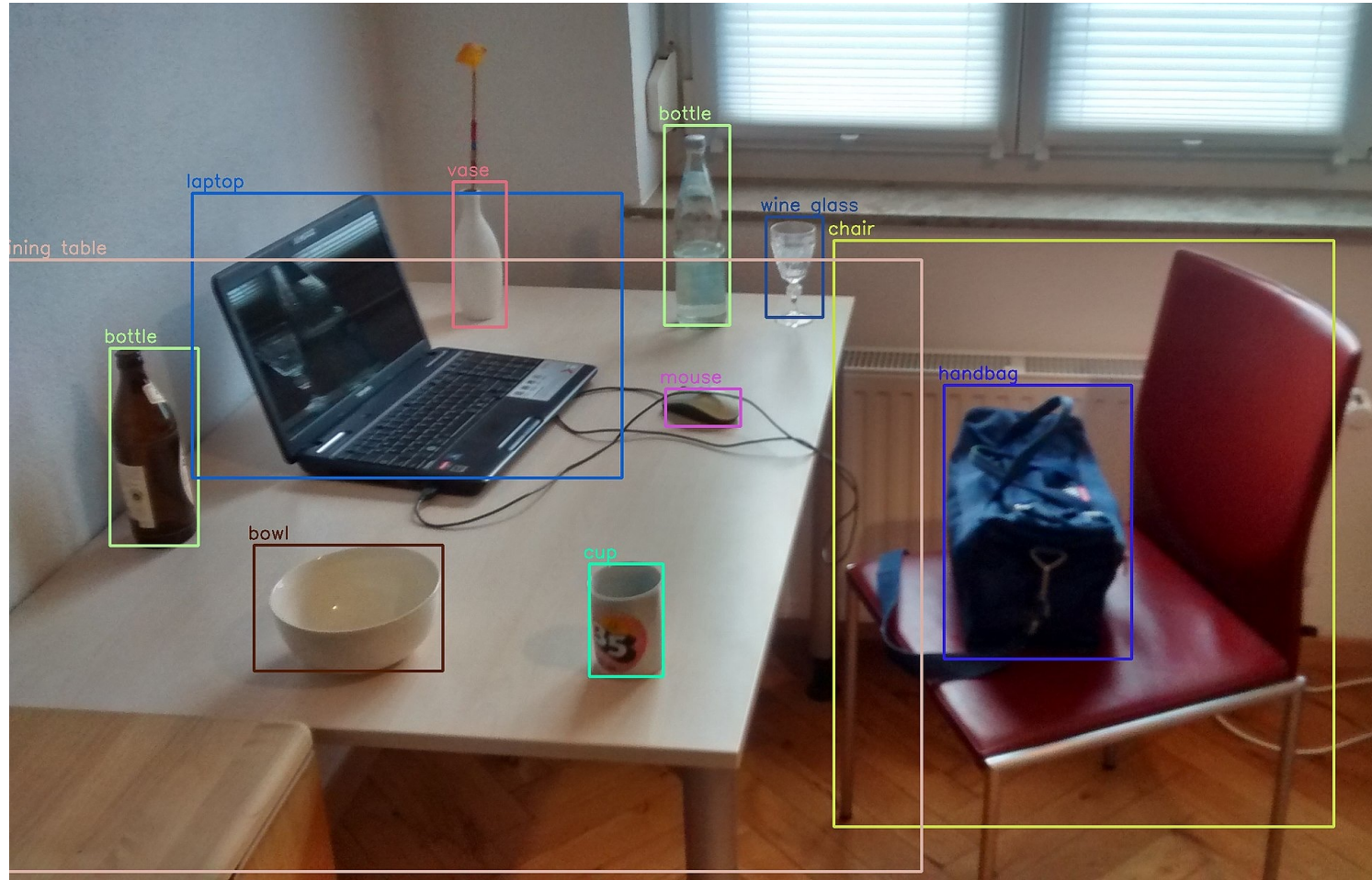
- Image segmentation the process of dividing an image into **multiple segments or regions**, each of which corresponds to a different object or part of the image.
- The goal of image segmentation is to **separate** meaningful regions **from the background** of the image.
 - Background
 - Foreground

Introduction

- It is an initial and vital step in **pattern recognition**-a series of processes aimed at overall **image understanding**.
- There are various techniques for image segmentation, including thresholding, clustering, and region growing, Deep learning-based approaches.

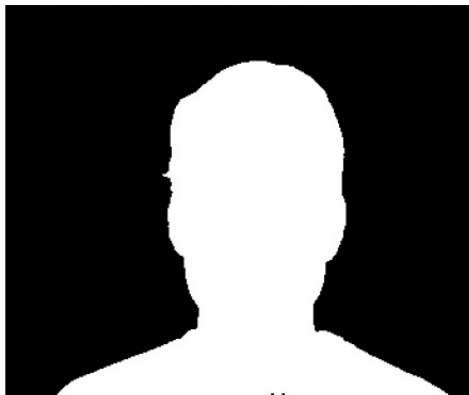
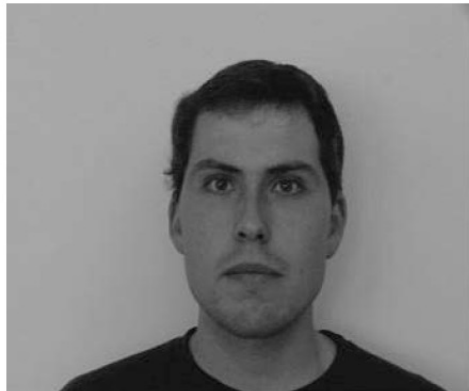
Introduction

- Object detection – differ from image segmentation



Introduction

- Image segmentation



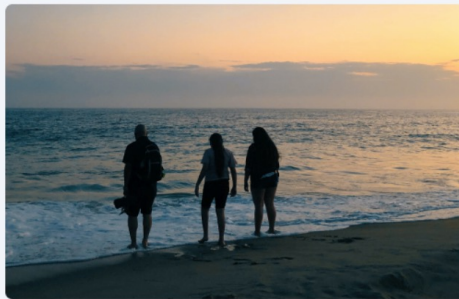
Output: mask

Object Detection vs. Image segmentation

- **Object detection** involves identifying the location and class of objects in an image.
 - **Output:** bounding boxes with a class label
- **Image segmentation** involves dividing an image into multiple regions or segments, each of which corresponds to a different object or part of the image.
 - **Output:** a mask or pixel-level labeling

Introduction

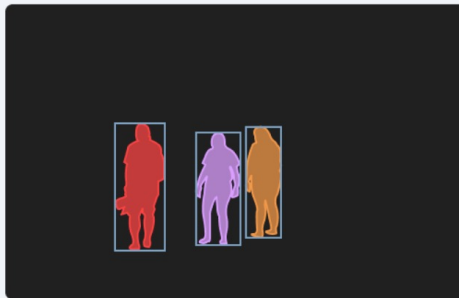
- Image segmentation



(a) Image



(b) Semantic Segmentation



(c) Instance Segmentation



(d) Panoptic Segmentation

Output: mask

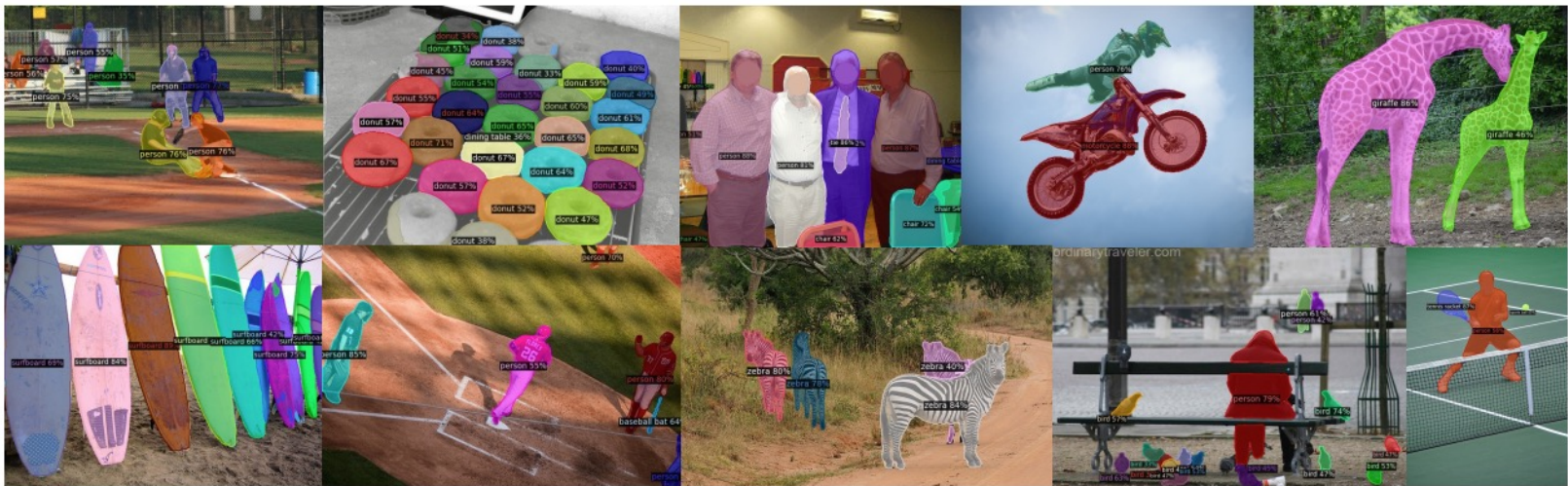
Introduction

- Semantic segmentation



Introduction

- Instance segmentation



Ref: Cheng T, Wang X, Chen S, et al. Sparse instance activation for real-time instance segmentation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 4433-4442.

Introduction

- Panoptic segmentation



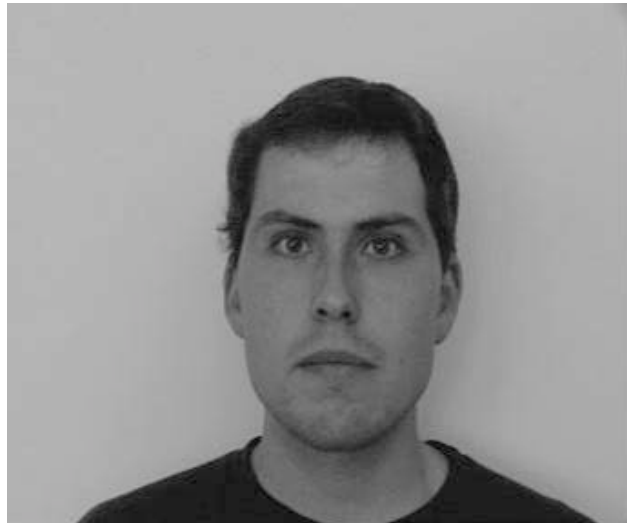
Thing-objects with a distinct identity

Stuff-do not have a distinct or discernible boundary

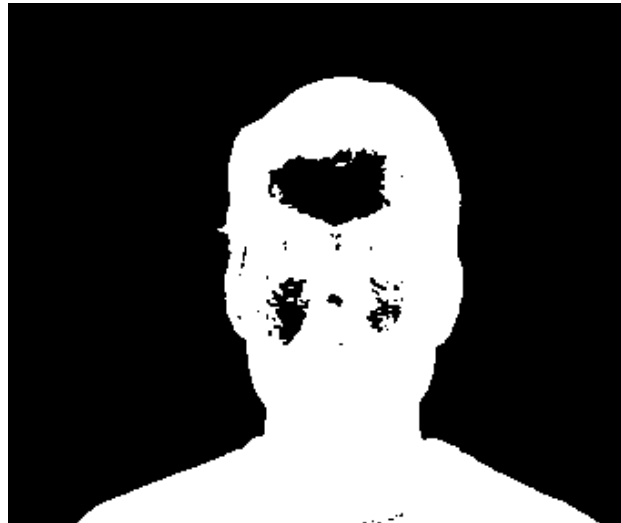
Ref: Wang H, Zhu Y, Adam H, et al. Max-deeplab: End-to-end panoptic segmentation with mask transformers[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 5463-5474.

How to do?

Thresholding



Original image
Peter $f[x,y]$

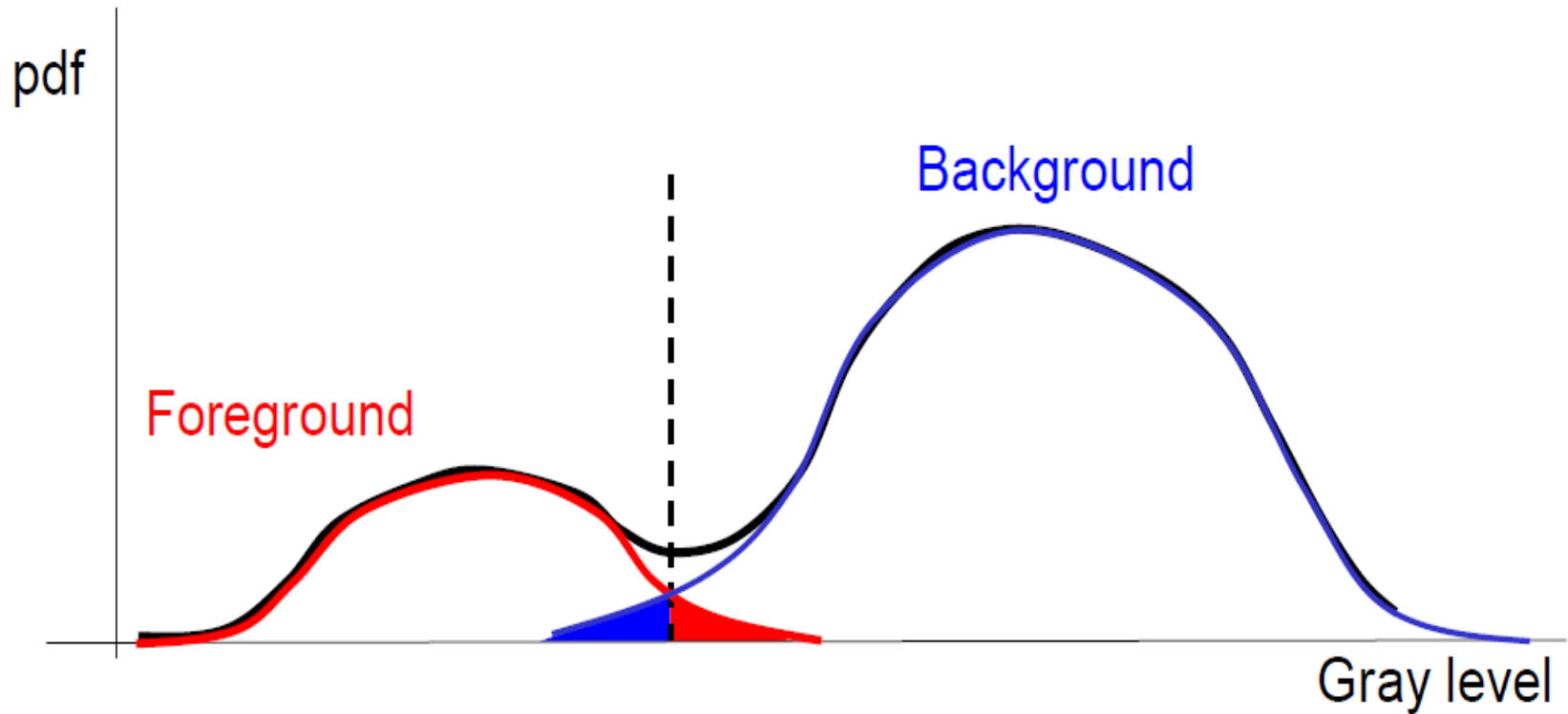


Thresholded
Peter $m[x,y]$



$f[x,y] \cdot m[x,y]$

How to choose the threshold?



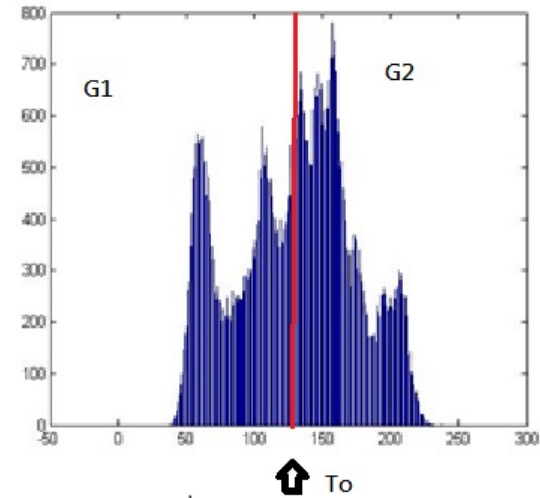
pdf: Probability Distribution Function

Bayes decision

Basic Global Thresholding

Different T_0 will get different results?

- 1) Select an initial T_0
- 2) Segment image use:
$$g(x,y)=\begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{if } f(x,y) < T \end{cases}$$
- 3) Compute the average intensity m_1 and m_2 for the pixels in G_1 and G_2 .
- 4) Compute a new threshold: $T = \frac{1}{2}(m_1 + m_2)$
- 5) Until the difference between values of T is smaller than a predefined parameter.



Otsu's Method

- find threshold T that *minimizes within-class variance* of both foreground and background.

Or:

- find threshold T that *maximizes between-class variance* of both foreground and background.

Otsu's Method

- find threshold T that **minimizes *within-class variance*** of both foreground and background.

$$\sigma_{within}^2(T) = \frac{N_{Fgrnd}(T)}{N} \sigma_{Fgrnd}^2(T) + \frac{N_{Bgrnd}(T)}{N} \sigma_{Bgrnd}^2(T)$$

Or:

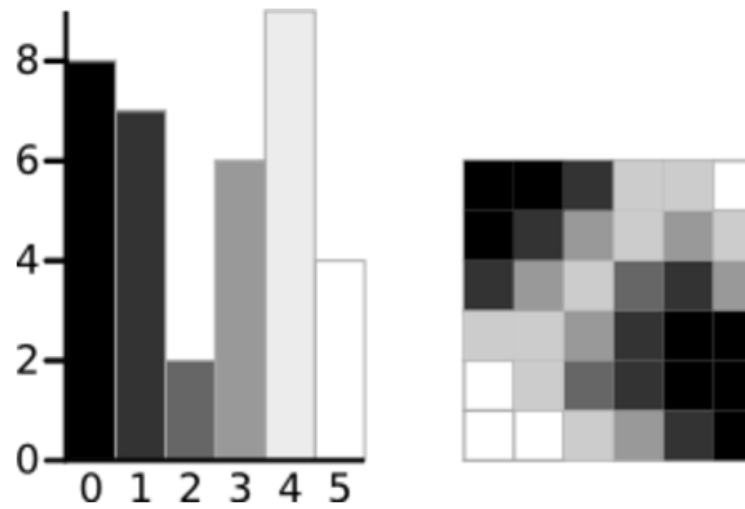
- find threshold T that **maximizes *between-class variance*** of both foreground and background.

$$\begin{aligned} \sigma_{between}^2(T) &= \sigma^2 - \sigma_{within}^2(T) \\ &= \left(\frac{1}{N} \sum_{x,y} f^2[x,y] - \mu^2 \right) - \frac{N_{Fgrnd}}{N} \left(\frac{1}{N_{Fgrnd}} \sum_{x,y \in Fgrnd} f^2[x,y] - \mu_{Fgrnd}^2 \right) - \frac{N_{Bgrnd}}{N} \left(\frac{1}{N_{Bgrnd}} \sum_{x,y \in Bgrnd} f^2[x,y] - \mu_{Bgrnd}^2 \right) \\ &= -\mu^2 + \frac{N_{Fgrnd}}{N} \mu_{Fgrnd}^2 + \frac{N_{Bgrnd}}{N} \mu_{Bgrnd}^2 = \frac{N_{Fgrnd}}{N} (\mu_{Fgrnd} - \mu)^2 + \frac{N_{Bgrnd}}{N} (\mu_{Bgrnd} - \mu)^2 \\ &= \frac{N_{Fgrnd}(T) \cdot N_{Bgrnd}(T)}{N^2} (\mu_{Fgrnd}(T) - \mu_{Bgrnd}(T))^2 \end{aligned}$$

Otsu's Method

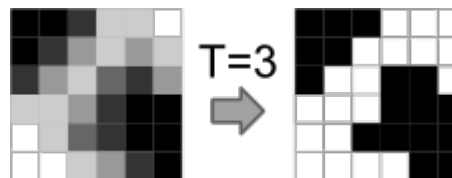
- Original image

Within-class variance



A 6-level greyscale image and its histogram

for the threshold value : 3

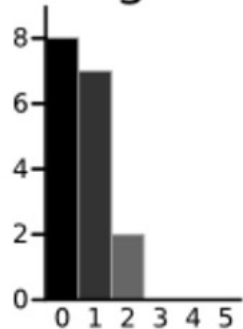


$$\begin{aligned} \text{Within Class Variance } \sigma_W^2 &= W_b \sigma_b^2 + W_f \sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152 \\ &= 0.4909 \end{aligned}$$

Otsu's Method

Within-class variance

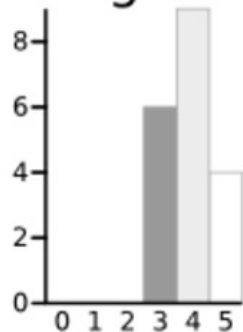
Background



8 pixels 7 pixels 2 pixels

$$\begin{aligned} \text{Weight } W_b &= \frac{8 + 7 + 2}{36} = 0.4722 \\ \text{Mean } \mu_b &= \frac{(0 \times 8) + (1 \times 7) + (2 \times 2)}{17} = 0.6471 \\ \text{Variance } \sigma_b^2 &= \frac{((0 - 0.6471)^2 \times 8) + ((1 - 0.6471)^2 \times 7) + ((2 - 0.6471)^2 \times 2)}{17} \\ &= \frac{(0.4187 \times 8) + (0.1246 \times 7) + (1.8304 \times 2)}{17} \\ &= 0.4637 \end{aligned}$$

Foreground



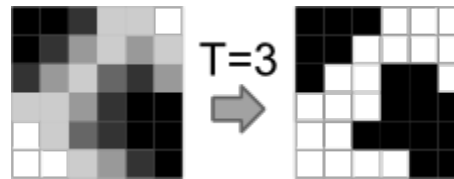
6 pixels 9 pixels 4 pixels

$$\begin{aligned} \text{Weight } W_f &= \frac{6 + 9 + 4}{36} = 0.5278 \\ \text{Mean } \mu_f &= \frac{(3 \times 6) + (4 \times 9) + (5 \times 4)}{19} = 3.8947 \\ \text{Variance } \sigma_f^2 &= \frac{((3 - 3.8947)^2 \times 6) + ((4 - 3.8947)^2 \times 9) + ((5 - 3.8947)^2 \times 4)}{19} \\ &= \frac{(4.8033 \times 6) + (0.0997 \times 9) + (4.8864 \times 4)}{19} \\ &= 0.5152 \end{aligned}$$

Otsu's Method

Within-class variance

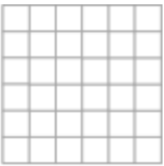
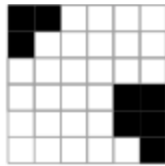
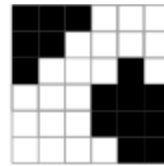
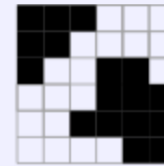
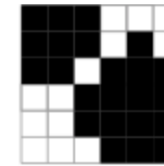
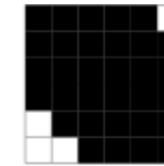
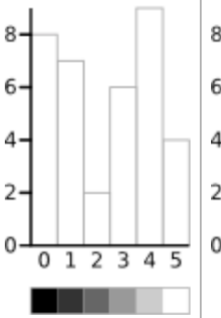
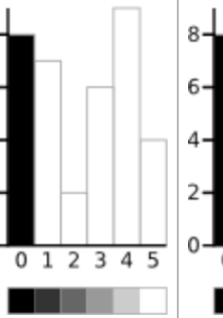
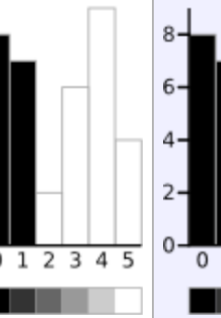
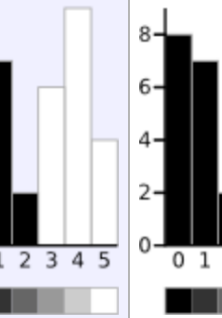
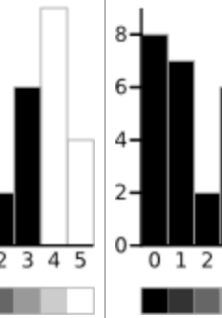
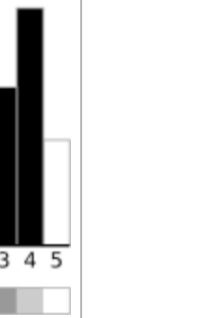
for the threshold value : 3



Within Class Variance $\sigma_W^2 = W_b \sigma_b^2 + W_f \sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152$
 $= 0.4909$

Otsu's Method

Within-class variance

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
						
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$	$W_b = 0.4722$	$W_b = 0.6389$	$W_b = 0.8889$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$	$M_b = 0.6471$	$M_b = 1.2609$	$M_b = 2.0313$
Variance, Background	$\sigma_b^2 = 0$	$\sigma_b^2 = 0$	$\sigma_b^2 = 0.2489$	$\sigma_b^2 = 0.4637$	$\sigma_b^2 = 1.4102$	$\sigma_b^2 = 2.5303$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$	$W_f = 0.5278$	$W_f = 0.3611$	$W_f = 0.1111$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$	$M_f = 3.8947$	$M_f = 4.3077$	$M_f = 5.000$
Variance, Foreground	$\sigma_f^2 = 3.1196$	$\sigma_f^2 = 1.9639$	$\sigma_f^2 = 0.7755$	$\sigma_f^2 = 0.5152$	$\sigma_f^2 = 0.2130$	$\sigma_f^2 = 0$
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$

Otsu's Method

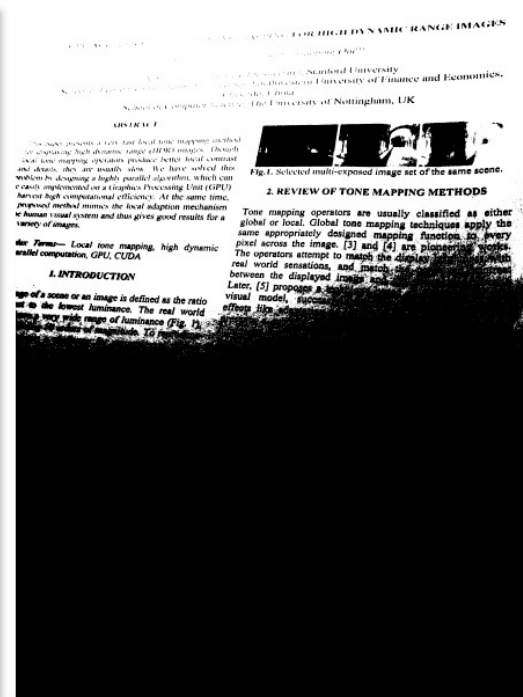
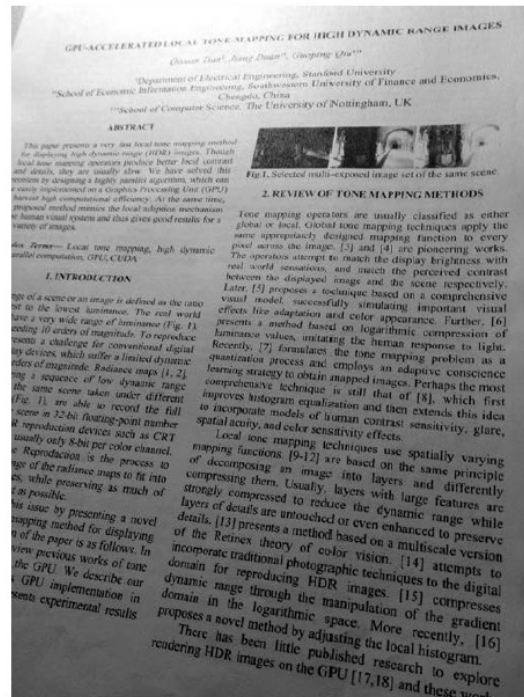
Between-class variance

Between Class Variance $\sigma_B^2 = \sigma^2 - \sigma_W^2$
 $= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2$ (where $\mu = W_b \mu_b + W_f \mu_f$)
 $= W_b W_f (\mu_b - \mu_f)^2$

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$
Between Class Variance	$\sigma_B^2 = 0$	$\sigma_B^2 = 1.5928$	$\sigma_B^2 = 2.5635$	$\sigma_B^2 = 2.6287$	$\sigma_B^2 = 2.1417$	$\sigma_B^2 = 0.8705$

Sometimes, a global threshold does not work

Original image

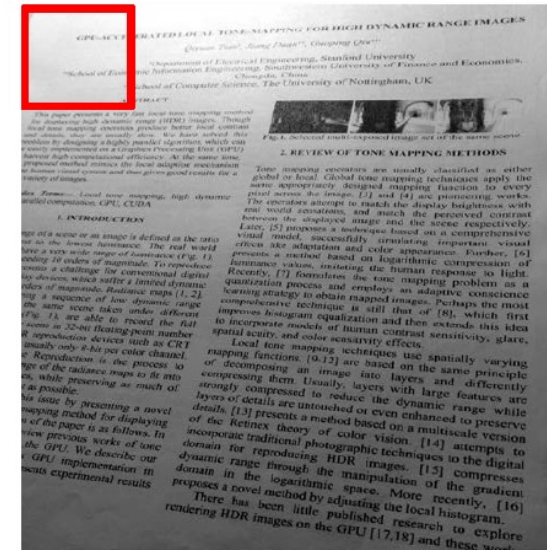


Thresholded with Otsu's Method

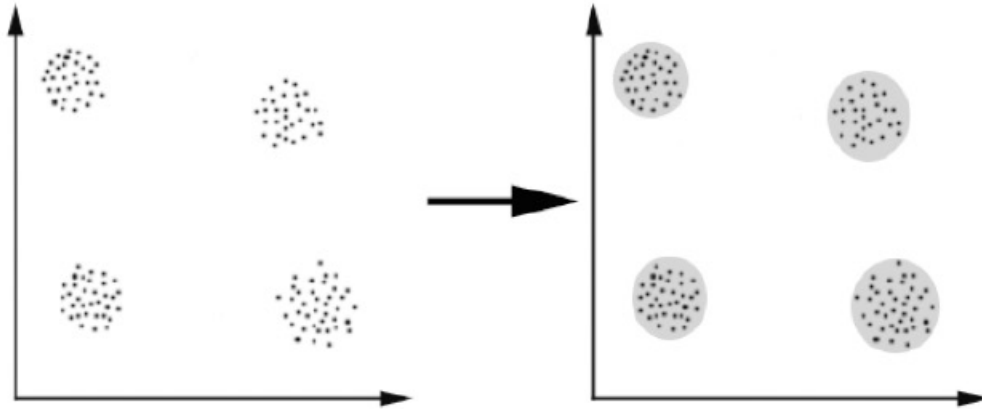
Locally adaptive thresholding

- Slide a window over the image
- For each window position, decide whether to perform thresholding
 - Thresholding should not be performed in uniform areas
 - Use variance or other suitable criterion
- Non-uniform areas: apply Otsu's method (based on local histogram)
- Uniform areas: classify the entire area as foreground or background based on mean value

Uniform areas are those regions where the pixel intensities or other characteristics are relatively consistent

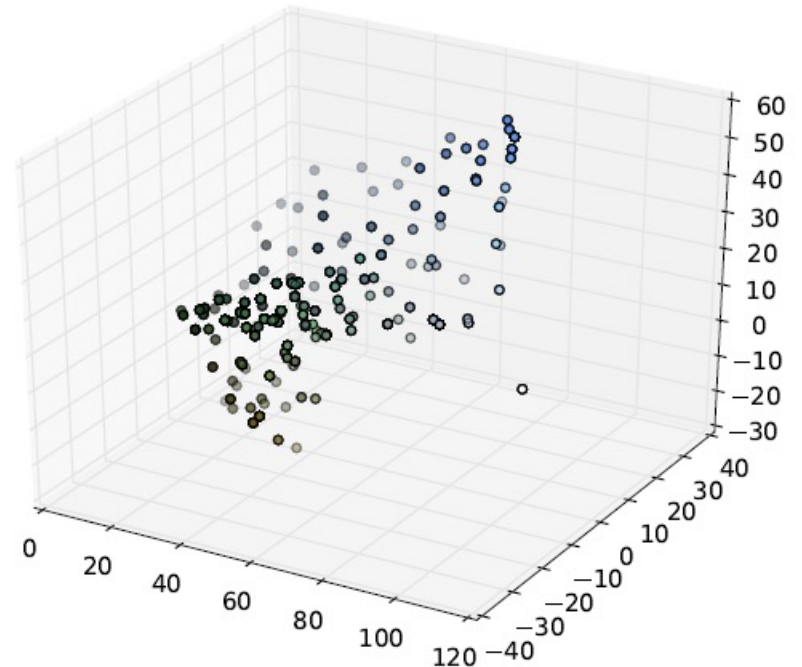


Clustering



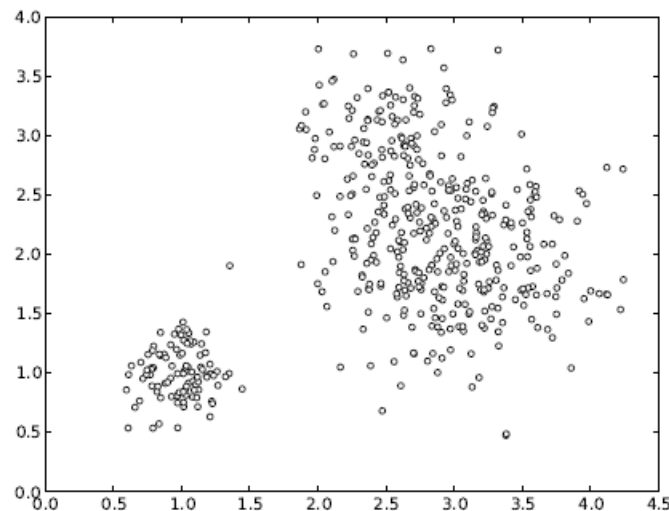
As a very basic and fast overview, what clustering does is: given a set of data points and a **feature** associated with each, it **groups** the data points in a way that **each one belongs to one cluster/group**. In the example shown here, the feature is 2 dimensional and there are 4 clusters among the data points.

Segmentation as clustering



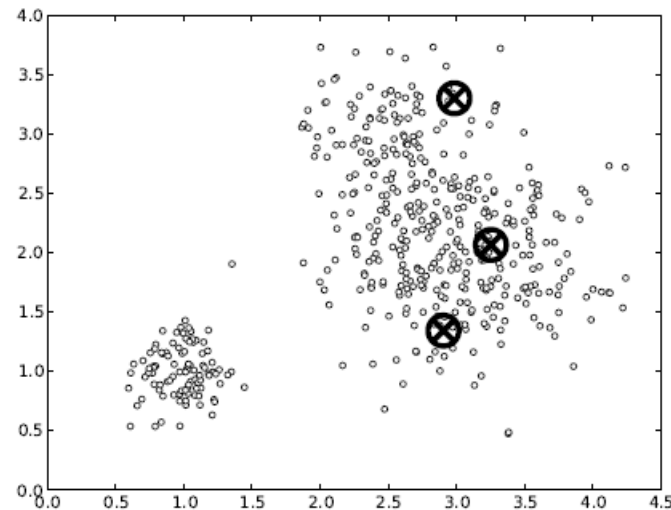
- Pixels are points in a high dimensional space
 - ▶ color: 3d
 - ▶ color+location:5d
- Cluster pixels into segment

K-Means clustering



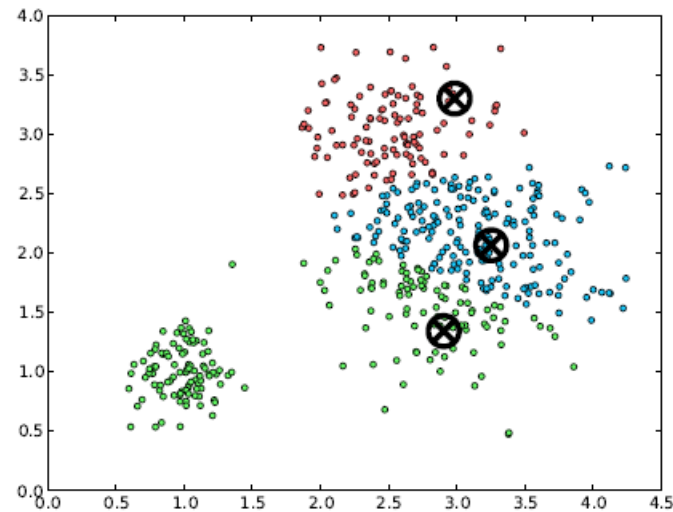
- ① Randomly initialize K cluster centers, c_1, \dots, c_k
- ② Given cluster centers, determine points in each cluster
 - ▶ For each point p , find the closest c_i . Put p into cluster i .
- ③ Given points in each cluster, solve for c_i
 - ▶ Set c_i to be the mean of points in cluster i
- ④ If c_i have changed, repeat Step 2

K-Means clustering



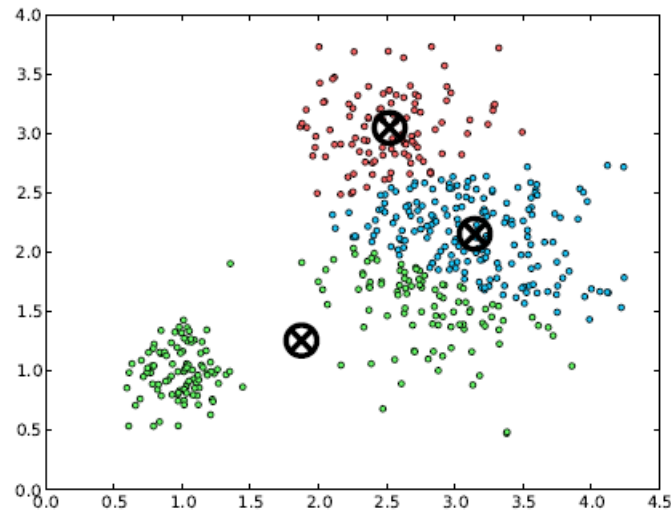
- 1 Randomly initialize K cluster centers, c_1, \dots, c_k
- 2 Given cluster centers, determine points in each cluster
 - For each point p , find the closest c_i . Put p into cluster i .
- 3 Given points in each cluster, solve for c_i
 - Set c_i to be the mean of points in cluster i
- 4 If c_i have changed, repeat Step 2

K-Means clustering



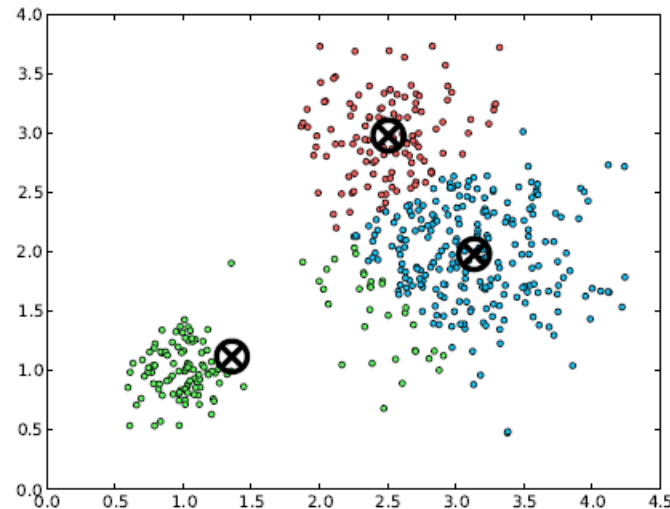
- 1 Randomly initialize K cluster centers, c_1, \dots, c_k
- 2 Given cluster centers, determine points in each cluster
 - For each point p , find the closest c_i . Put p into cluster i .
- 3 Given points in each cluster, solve for c_i
 - Set c_i to be the mean of points in cluster i
- 4 If c_i have changed, repeat Step 2

K-Means clustering



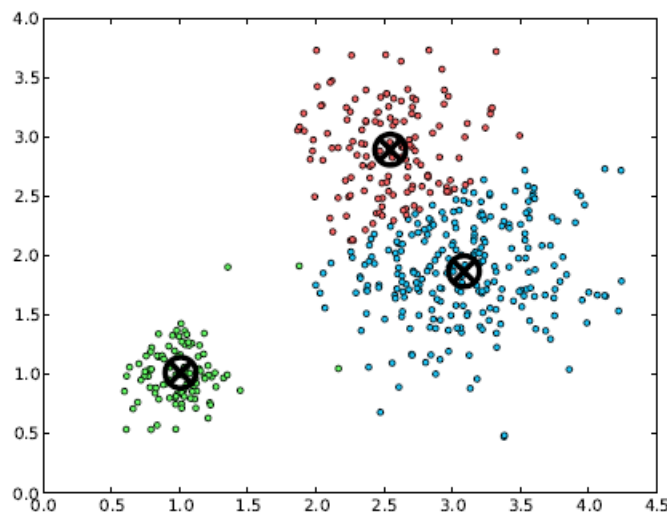
- ① Randomly initialize K cluster centers, c_1, \dots, c_k
- ② Given cluster centers, determine points in each cluster
 - ▶ For each point p , find the closest c_i . Put p into cluster i .
- ③ Given points in each cluster, solve for c_i
 - ▶ Set c_i to be the mean of points in cluster i
- ④ If c_i have changed, repeat Step 2

K-Means clustering



- ① Randomly initialize K cluster centers, c_1, \dots, c_k
- ② Given cluster centers, determine points in each cluster
 - ▶ For each point p , find the closest c_i . Put p into cluster i .
- ③ Given points in each cluster, solve for c_i
 - ▶ Set c_i to be the mean of points in cluster i
- ④ If c_i have changed, repeat Step 2

K-Means clustering



- ① Randomly initialize K cluster centers, c_1, \dots, c_k
- ② Given cluster centers, determine points in each cluster
 - ▶ For each point p , find the closest c_i . Put p into cluster i .
- ③ Given points in each cluster, solve for c_i
 - ▶ Set c_i to be the mean of points in cluster i
- ④ If c_i have changed, repeat Step 2

Overview

Topic 1: basic introduction

- Why image processing is important?
 - everywhere
- How the human vision system works ?
 - eye physiology
- What is light & color?
 - electromagnetic spectrum
 - Color model
- What is image ?
 - Matrix representation, pixel
 - Image formats
 - Image resolution

Topic 1: basic introduction

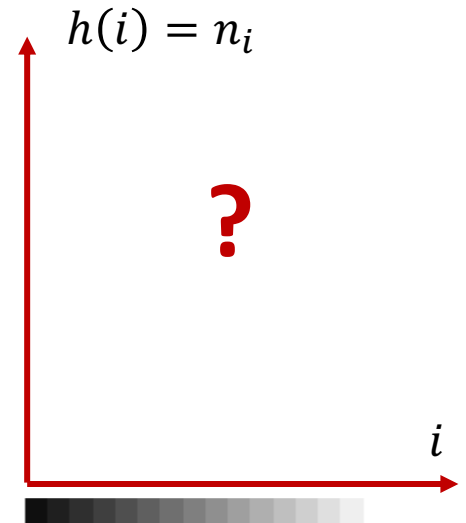
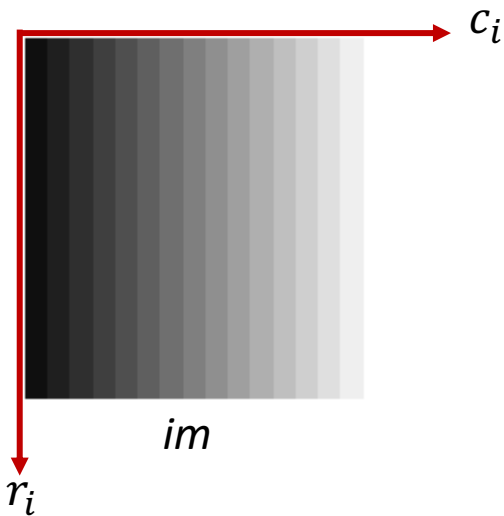
- Image resolution and quantization
- Image Interpolation
 - Nearest Neighbor Interpolation
 - Bilinear interpolation
 - Bicubic interpolation
- Basic relationships among pixels
 - Neighbors
 - Connectivity
 - Distance measures

Topic 2: Image enhancement

- Mapping function
 - Log transforms
 - Power-law
 - Piecewise-Linear
- Image histogram
- Histogram Equalization
- Spatial filtering
 - Overlap, multiply, add
- Spatial frequency
 - LPF, HPF
- Practical filters
 - Averaging filter
 - Edge Detection Filters
 - sharpening filters
 - Order Statistic Filters
- Convolution Neural Networks
 - Convolution Layer
 - Activation functions
 - Pooling layer

Image histogram

- Image histogram is a **graphical** representation of the **distribution** of **gray levels** in an image
- Is a **discrete function** $h(i) = n_i$ where i is a **gray level** and n_i is the **number of pixels** having that gray level



Normalized image histogram

- In the **Normalized Histogram**, h_n , each value of the histogram is **divided by the total number** of pixels in the image $N = \sum_{\forall i} n_i$
$$h_n(i) = \frac{n_i}{N}$$
- When $N \rightarrow \infty$ then $h_n(i)$ represents the **estimate** of the **probability** of the **gray level i** .
 - In this case $h_n(i)$ becomes **equivalent** to the **Probability Density Function** (pdf) of a random *variable representing the gray level*.

Topic 3: Image transform

- Spatial domain
- Transform domain
 - Basis
 - Coefficients
- Discrete Fourier Transform (DFT)
 - transforms an image from the spatial domain to the frequency domain.
- Discrete Cosine Transform (DCT)

Topic 4: image compression

- Why compression
 - Memory storage, transmission time
- How compression
 - Lossless and lossy compression
 - Type of redundancy
 - Coding redundancy (data representation)
 - Fixed-length code
 - Variable-length code
 - » Huffman coding
 - » Shannon-Fano code
 - » Entropy
 - Interpixel redundancy (Correlation between adjacent samples)
 - Predictive coding
 - Psychovisual redundancy
 - Transform coding
 - The standard JPEG encoding
 - Color space transformation, 8*8 blocks DCT, Quantization, Zig-zag reading, Entropy coding

Fixed-Length Codes

- Properties
 - Use the **same number** of bits to represent all **possible** symbols (pixels) produced by the **source**
 - **Simplify** the **decoding** process

Variable-Length Codes

- Main **properties** of variable-length codes (**VLC**)
 - Use a **different number** of bits to **represent** each **symbol**
 - Allocate **shorter-length codewords** to symbols that occur **more frequently**
 - Allocate **longer-length codewords** to **rarely occurred** symbols
 - More **efficient representation**; good for compression

Huffman Coding

The Huffman algorithm involves the following steps:

1. Calculate the **frequency** of each symbol in the data stream.
2. Sort the symbols by frequency, **from lowest to highest**.
3. Take **the two least frequent** symbols and combine them into a new internal node with a frequency equal to the sum of the frequencies of the two symbols.
4. **Repeat** step 3 until all the symbols are merged into a single tree.
5. Assign codes to the symbols by traversing the tree from the root to each leaf. **Assign a '0' bit for each left branch and a '1' bit for each right branch.**

Bounds on lossless coding

- The average codeword length of a lossless code cannot be less than entropy
- The average amount of information carried by a source is the *entropy*
$$-\sum_{i=1}^M p_i \log_2 p_i = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} [\text{bit} / \text{symbol}]$$
- Entropy represents the target average number of bits/symbol of a lossless encoder
- *Coding Efficiency:*

$$\eta = H(X) / n$$

where n is the average codeword length

Shannon-Fano Code

- Algorithm

- Line up symbols by decreasing probability of occurrence
- Divide symbols into 2 groups so that both have similar combined probability
- Assign **0** to 1st group and **1** to the 2nd
- Repeat step 2

- Example

Symbols	Prob.	Code-word
A	0.35	00
B	0.17	01
C	0.17	10
D	0.16	110
E	0.15	111

$$\begin{aligned}\text{Average code-word length} &= \\ &0.35 \times 2 + 0.17 \times 2 + 0.17 \times 2 \\ &\quad + 0.16 \times 3 + 0.15 \times 3 \\ &= 2.31 \text{ bits per symbol}\end{aligned}$$

Review

0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	7	1	1	1	1	0	0
0	7	5	5	5	5	2	2
0	7	0	0	0	0	2	2
0	7	2	2	2	2	2	2
0	0	2	2	2	2	2	2
0	0	0	0	0	0	0	0

0 0 0 1
2 2 5 5
2 7 7 0
0 0 0 0

1. How many bits are needed if encoded by fixed-length coding?
2. How to do Huffman coding for compression?
3. Based on 2, how many bits on average would be required to represent each pixel?
4. What is the entropy? Calculate the histogram of this image and plot its normalized cumulative distribution function.

Comparison between 3 & 4, observations:

The entropy will equal to the average bits required to represent each pixel and Huffman is a code which is optimal when all symbols' probabilities are integral power of $\frac{1}{2}$

If the probabilities are not integral power of $\frac{1}{2}$, the average codeword length of a lossless code cannot be less than entropy

Topic 5: Morphological Operations

- Morphological Operations
 - Erosion
 - If the **structuring element fits** in the **foreground object**, write “1” at the **origin** of the structuring element in the **output image**!
 - Dilation
 - If the **structuring element touches** the **foreground object**, write “1” at the **origin** of the structuring element in the **output image**!
 - Opening
 - **erosion followed** by a **dilation**
 - Closing
 - **Dilatation** followed by an **Erosion**
 - Hit-and-miss Operation, Thinning, Thickening
 - **foreground** pixels **exactly** matches the **foreground structuring element (SE1)**
 - **background** pixels **exactly** matches the **background structuring element (SE2)**.

Erosion and Dilation

im =

0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0

se =

1	1	1
1	1	1
1	1	1

Do it yourself

Padding the boundary with the same value as the neighbor if it is needed.

Padding

se =

```
1  1  1
1  1  1
1  1  1
```

im =

```
0  0  0  0  0  0  0  0
0  1  0  0  0  0  1  0
0  1  0  0  0  0  1  0
0  1  1  1  1  1  1  0
0  1  1  1  1  1  1  0
0  1  1  1  1  1  1  0
0  1  1  1  1  1  1  0
0  0  1  1  1  1  0  0
```



```
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  1  0  0  0  0  1  0  0
0  0  1  0  0  0  0  1  0  0
0  0  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  0  0
0  0  0  1  1  1  1  0  0  0
0  0  0  1  1  1  1  0  0  0
```

Erosion Result

im =

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0

imo =

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0

Dilation Result

im =

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0

imo =

1	1	1	0	0	1	1	1
1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1