7-9 Monday – 309-GD2

Xử lý ảnh
INT3404 1

Giảng viên: TS. Nguyễn Thị Ngọc Diệp

Email: ngocdiep@vnu.edu.vn

Slide & code: https://github.com/chupibk/INT3404_1

Week 7 recall: Fourier transform

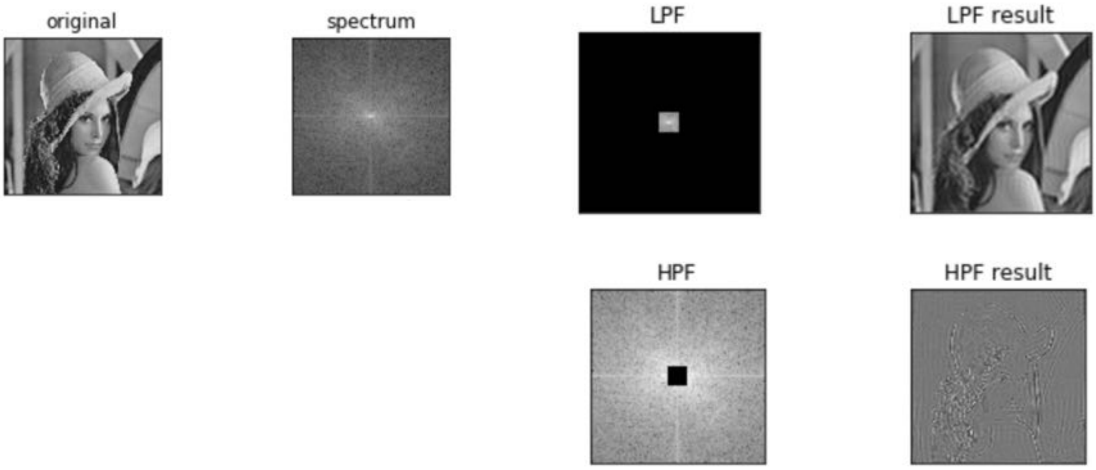
f(x, y) → Fourier Transform → F(u, v)

$$F(u,v) = \underbrace{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}}$$

Basic function corresponding to each point F(u, v) in the Fourier space
= sine and cosine waves with increasing frequencies

the value of each point *F(u, v)* is obtained by multiplying the spatial image with the corresponding base function and summing the result.

Week 7 recall: Filtering in frequency domain



Final project registration

Registration form

- https://forms.gle/Qpp1hbX9QPG875gT8

Homework 2 review

Delayed to next week ☺

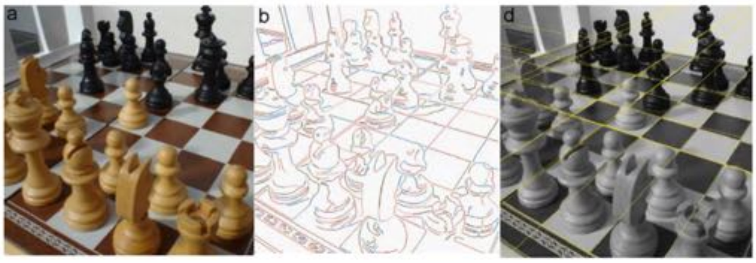
Lịch trình

Tuần	Nội dung	Yêu cầu đối với sinh viên
1	Giới thiệu môn học Làm quen với OpenCV + Python	Cài đặt môi trường: Python 3, OpenCV 3, Numpy, Jupyter Notebook
2	Phép toán điểm (Point operations) – Điều chỉnh độ tương phản – Ghép ảnh	Làm bài tập 1: điều chỉnh gamma tìm contrast hợp lý
3	Histogram - Histogram equalization - Phân loại ảnh dùng so sánh histogram	Thực hành ở nhà
4	Phép lọc trong không gian điểm ảnh (linear processing filtering) - làm mịn, làm sắc ảnh	Thực hành ở nhà Tìm hiểu thêm các phép lọc
5	Tìm cạnh (edge detection)	Thực hành ở nhà
6	Các phép toán hình thái (Erosion, Dilation, Opening, Closing) - tìm biển số	Làm bài tập 2: tìm barcode
7	Chuyển đổi không gian - miền tần số (Fourier) - Hough transform	Thực hành ở nhà
8	Phân vùng (segmentation) - depth estimation - threshold-based - watershed/grabcut	Đăng ký thực hiện bài tập lớn
9	Mô hình màu Chuyển đổi giữa các mô hình màu	Làm bài tập 3: Chuyển đổi mô hình màu và thực hiện phân vùng
10	Mô hình nhiễu -Giảm nhiễu -Khôi phục ảnh -Giảm nhiễu chu kỳ - Ước lượng hàm Degration -Hàm lọc ngược, hàm lọc Wiener	Thực hành ở nhà
11	Template matching – Image Matching	Làm bài tập 4: puzzle
12	Nén ảnh	Thực hành ở nhà
13	Hướng dẫn thực hiện đồ án môn học	Trình bày đồ án môn học
14	Hướng dẫn thực hiện đồ án môn học	Trình bày đồ án môn học
15	Tổng kết cuối kỳ	Ôn tập

Hough transform

Hough transform

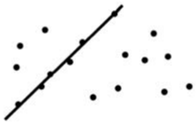
- Robust method to find a shape in an image
- Shape can be described in parametric form
- A voting scheme is used to determine the correct parameters

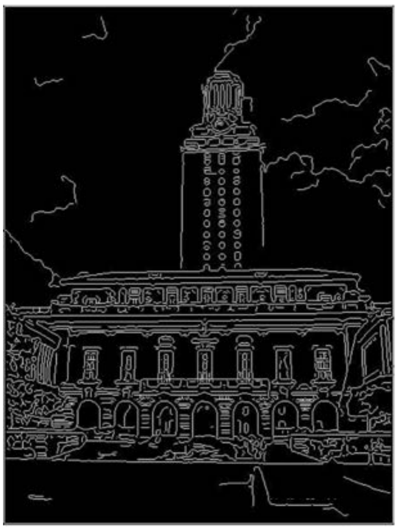


Example: Line fitting

- Many objects characterized by presence of straight lines

Difficulty of line fitting





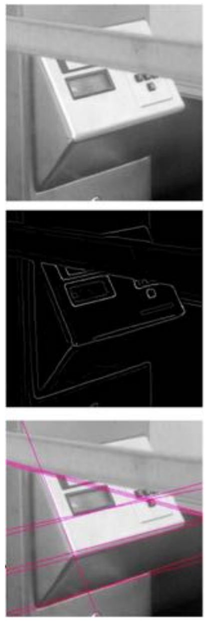
- Extra edge points (clutter), multiple models
  - Which points go with which line, if any?
- Only some parts of each line detected, and some parts are missing:
  - How to find a line that bridges missing evidence?
- Noise in measured edge points, orientations:
  - How to detect true underlying parameters

Voting

- It is not feasible to check all combinations of features by fitting a model to each possible subset
- Voting is a general technique where we let the features vote for all models that are compatible with it
  - Cycle through features, cast votes for model parameters
  - Look for model parameters that receive a lot of votes
- Noise & clutter features with cast votes too, but typically their votes should be inconsistent with the majority of "good" features

Fitting lines with Hough transform

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?
- Hough transform is a voting technique that can be used to answer all of these questions
- Main idea:
  1. Record vote for each possible line on which each edge point lies
  2. Look for lines that get many votes

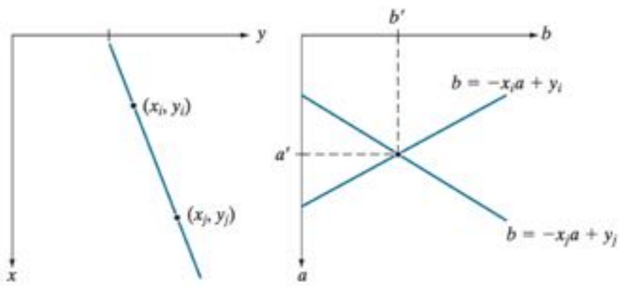


Line planes

a b

FIGURE 10.28
(a) xy-plane.
(b) Parameter space.

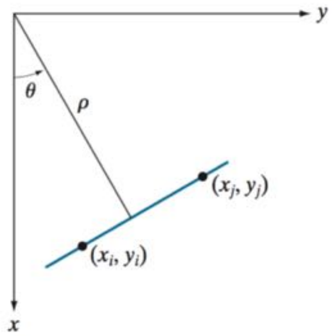
ab-plane or parameter space



y = a'x + b'

What if the line approaches the vertical or horizontal direction?
(i.e., infinity slope)

Polar representation for lines



rho: perpendicular distance from line to origin

theta: angle the perpendicular makes with the x-axis

Normal presentation of a line:

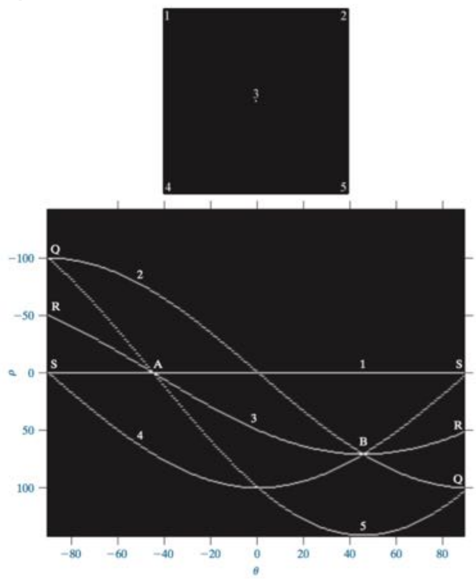
$$\boxed{x\cos\theta + y\sin\theta = \rho}$$

→ Point in image space is now sinusoid segment in Hough space

Example of Hough transform

a
b

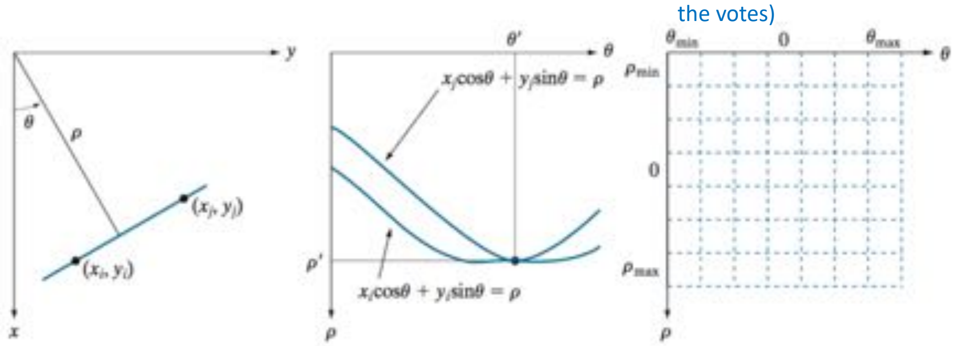
FIGURE 10.30 (a) Image of size 101×101 pixels, containing five white points (four in the corners and one in the center). (b) Corresponding parameter space.



Finding lines in an image: Hough algorithm

1. Using the polar parameterization

2. Create a Hough Accumulator Array (keeps the votes)



- Domain of the parametric space:

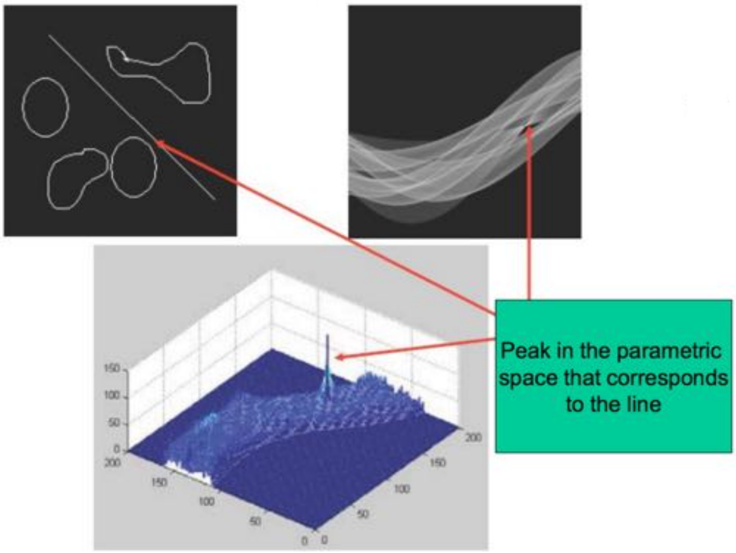
$$r \in \left[-\sqrt{M^2+N^2}, \sqrt{M^2+N^2}\right], \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

M and N image resolution

Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. For each ***edge*** point in $E(x, y)$ in the image
   for θ = 0 to 180 // some quantization; why not 2pi?
   $d = x\cos\theta + y\sin\theta$ // maybe negative
   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta + y\sin\theta$

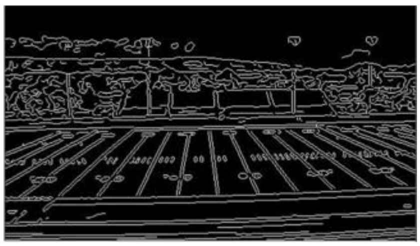
Line detection example



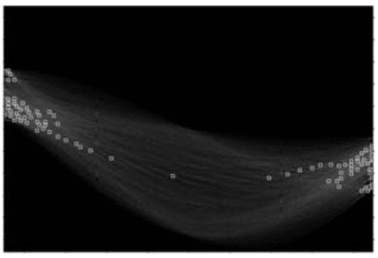
Line detection example

original

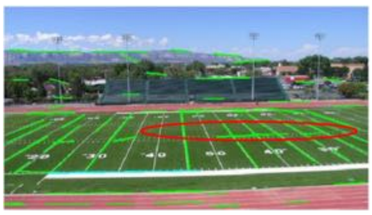
Canny edges



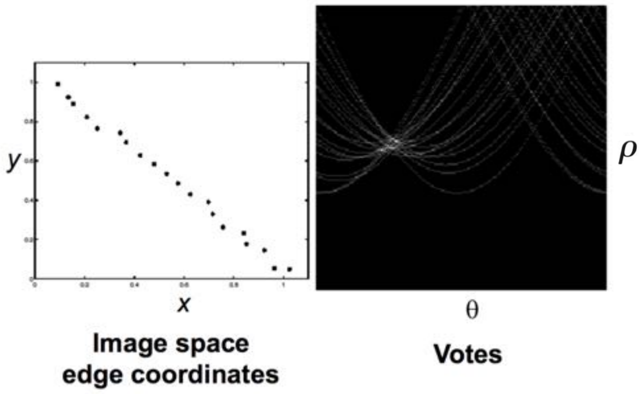
Vote space and
top peaks



Longest segments
found

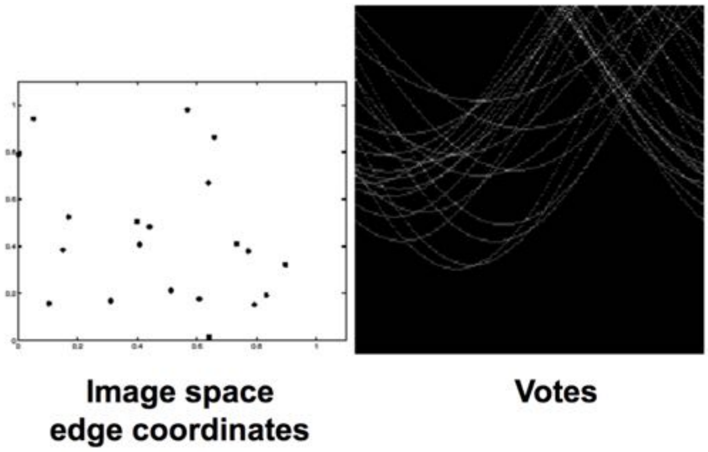


Impact of noise on Hough



What difficulty does this present for an implementation?

Impact of noise on Hough



Everything appears to be “noise”, or random edge points, but we still see some peaks in the vote space

Extensions of Hough algorithm

Extension 1: Use the image gradient

1. same
2. for each edge point I[x,y] in the image

   θ = gradient at (x,y)

   $d = x\cos\theta - y\sin\theta$

   H[d, θ] += 1
3. same
4. same

$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point I[x,y] in the image

   compute unique (d, θ) based on image gradient at (x,y)

   H[d, θ] += 1
3. same
4. same

(Reduces degrees of freedom)

Extension 2

- give more votes for stronger edges (use magnitude of gradient)

Extension 3

- change the sampling of (d, θ) to give more/less resolution

Extension 4

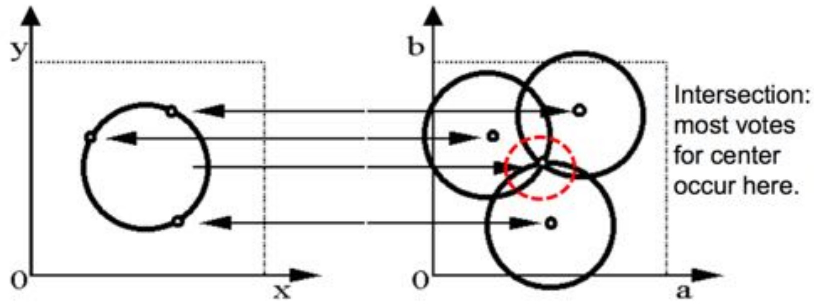
- The same procedure can be used with circles, squares, or any other shape…

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r, unknown gradient direction



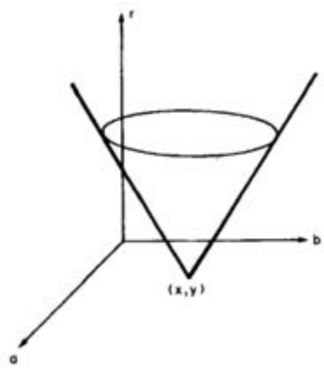
Hough transform for circles

Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

If radius is not known: 3D Hough Space!

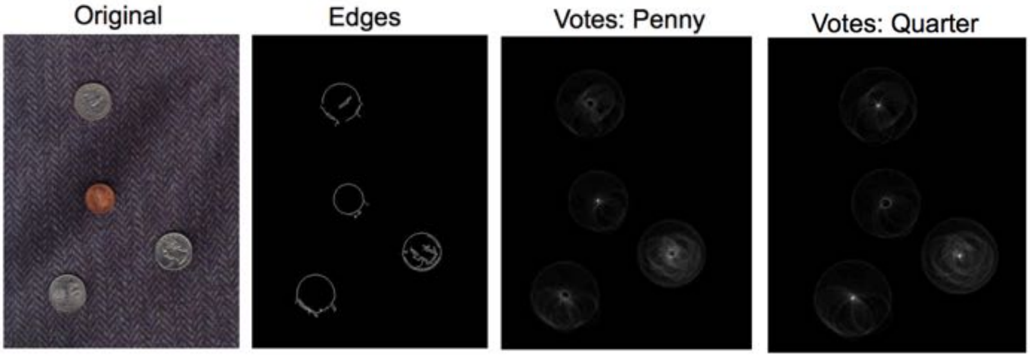
Use Accumulator array $A(a,b,r)$



Hough transform for circles

For every edge pixel (x,y) :
  For each possible radius value r:
    For each possible gradient direction θ:
      // or use estimated gradient at (x,y)
        $a = x - r\cos(\theta)$ // column
        $b = y + r\sin(\theta)$ // row
        $H[a,b,r] += 1$
  end
end

Example: Detecting circles with Hough



Note: a different Hough transform (with a separate accumulator) was used for each circle radius

Example: iris detection

Gradient+threshold | Hough space (fixed radius) | Max detections

Voting: practical tips

- Minimize irrelevant tokens first
- Choose a good grid/discretization

Too fine ← ? → Too coarse

- Vote for neighbors, also (smoothing in accumulator array)
- Use direction of edge to reduce parameters by 1
- To read back which points voted for "winning" peaks, keep tags on the votes

Hough transform: pros and cons

- Pros:
  - All points are processed independently, so can cope with occlusion, gaps
  - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
  - can detect multiple instances of a model in a single pass
- Cons:
  - Complexity of search time increases exponentially with the number of model parameters
  - Non-target shapes can produce spurious peaks in parameter space
  - Quantization: can be tricky to pick a good grid size

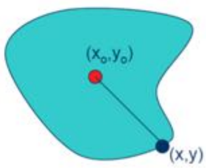
Generalized Hough Transform

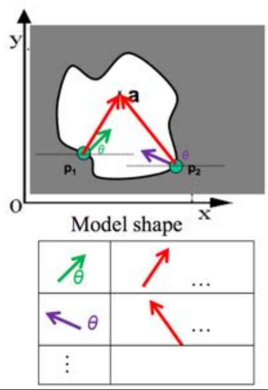
- Detect any arbitrary shape
  - Requires specification of the exact shape of the object
  - Define a model shape by its boundary points and a reference point

- Compute centroid
- For each edge compute its distance to centroid

$$r = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

- Find edge orientation (gradient angle)
- Construct a table of angles and r values





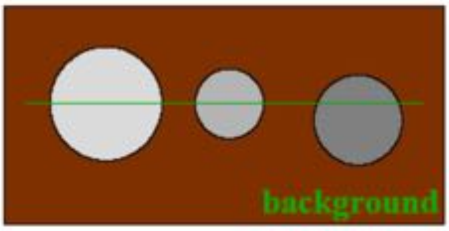
Generalized Hough algorithm

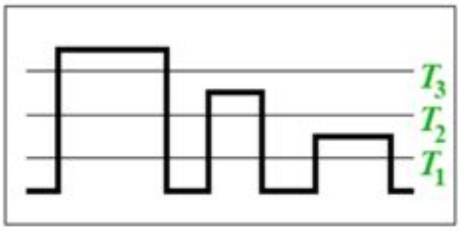
- As homework ☺

Threshold-based Segmentation

Thresholding based segmentation

- Goal: to identify an object based on uniform intensity
- Use the histogram to compute the best threshold that can separate the object intensity





Thresholding principles

- Basic image segmentation technique
- Assumes following conditions:
  - Scene contains uniformly illuminated, flat surfaces
  - Image is set of approximately uniform regions
- Goal:
  - Set one or more thresholds which split intensity range into intervals
  → define intensity classes
- Result:
  - Objects labelled by classifying pixel intensities into classes
  → Objects separated from background

Thresholding example

- Set $N-1$ thresholds T_k, $k = 1, \ldots, N-1$, $N \geq 2$, so that pixel $f(x, y)$ is classified into class n if

$$T_{n-1} \leq f(x, y) < T_n, \quad n = 1, \ldots, N$$

- By definition, $T_0 = 0$ and $T_N = G_{max} + 1 = 256$

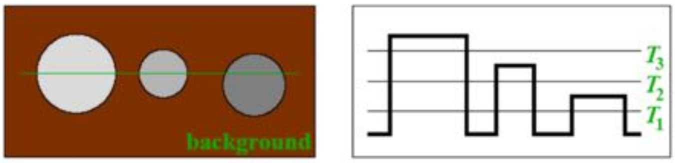


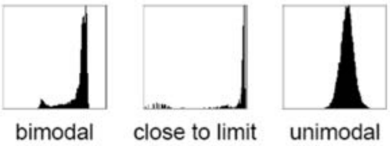
Illustration of 4-level thresholding. $T_0 = 0$ and $T_4 = 256$. First level is background.

Thresholding example

original image | bilevel thresholding | trilevel thresholding

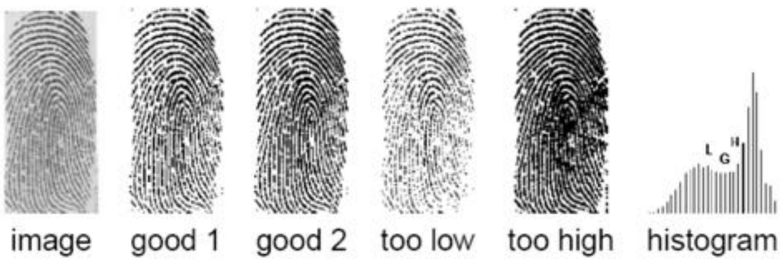
- Single threshold: $N = 2$
  - **bilevel** (binary) thresholding, or **binarisation**
  - $\Rightarrow$ considered in this course
- **Multilevel** thresholding: $N > 2$
  - case $N = 3$ often called **trilevel**

Histogram profiles



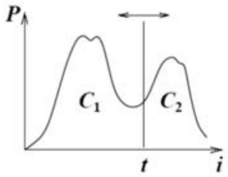
- Desirable histogram shape
  - bimodal with distinct modes and valley between modes
  - ⇒ minimum of valley separates classes
- Undesirable histogram shapes
  - **mode at limit** of intensity range
  - ⇒ modelling the histogram difficult
  - **mode not distinct**
  - ⇒ setting good threshold not easy
  - **unimodal**
  - ⇒ thresholding difficult but still possible

Good and bad histograms



- Several thresholds are acceptable
  - near valley (G) in histogram
- Bad thresholds have different effects
  - too low threshold (L) tends to split lines
  - too high threshold (H) tends to merge lines

Maximum separation



- Proposed by N.Otsu (Japan), 1978
- Consider a **candidate threshold** t
  - t defines two classes of grayvalues
- Define measure of **separation of classes**
  - distance between classes as function of t
- Find optimal threshold t_{opt} that **maximises separation**

Adaptive thresholding

Mean and variance of **total** normalised histogram $P(i)$:

$$\mu = \sum_{i=0}^{G_{max}} iP(i) \qquad \sigma^2 = \sum_{i=0}^{G_{max}} (i-\mu)^2 P(i)$$

Threshold t splits $P(i)$ into **two classes** C_1, C_2 with

$$\mu_1(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} iP(i) \qquad \sigma_1^2(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} [i - \mu_1(t)]^2 P(i)$$

$$\mu_2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{G_{max}} iP(i) \qquad \sigma_2^2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{G_{max}} [i - \mu_2(t)]^2 P(i)$$

$$q_1(t) = \sum_{i=0}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{G_{max}} P(i) \qquad q_1(t) + q_2(t) = 1$$

Two types of variance

- Total variance σ^2 has two components
  - **within-class variance** for given t
  - $\Rightarrow$ weighted sum of two class variances
  - **between-class variance** for given t
  - $\Rightarrow$ distance between classes
- Within-class variance is

$$\sigma_W^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$\Rightarrow$ note that $\mu = q_1(t)\mu_1(t) + q_2(t)\mu_2(t)$

- Between-class variance is the rest of σ^2

$$\begin{aligned}\sigma_B^2(t) &= \sigma^2 - \sigma_W^2(t) \\ &= q_1(t)q_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2 \\ &= q_1(t)\left[1 - q_1(t)\right]\left[\mu_1(t) - \mu_2(t)\right]^2\end{aligned}$$

Threshold selection via optimization

- Optimal threshold t_{opt} best separates the two classes
- $\sigma_W^2(t) + \sigma_B^2(t)$ is constant $\longrightarrow$ two equivalent options
  - *minimise* $\sigma_W^2(t)$ as *overlap of classes*
  - *maximise* $\sigma_B^2(t)$ as *distance between classes*

$\Rightarrow$ Use second option

Otsu threshold selection algorithm

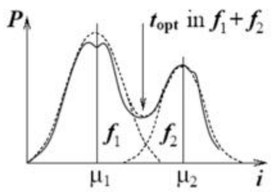
$$q_1(t+1) = q_1(t) + P(t+1) \quad \text{with } q_1(0) = P(0)$$
$$\mu_1(t+1) = \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)} \quad \text{with } \mu_1(0) = 0 \quad (2)$$
$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)}$$

1. Compute image histogram $P(i)$, calculate μ and σ
2. For each $0 < t < G_{max}$
   - recursively compute $q_1(t)$, $\mu_1(t)$ and $\mu_2(t)$ by eq.(2)
   - calculate $\sigma_B^2(t)$ by eq.(1)
3. Select threshold as $t_{opt} = \arg\max_t \sigma_B^2(t)$

Properties

- **Advantages**
  - general: no specific histogram shape assumed
  - works well, stable
  - extension to *multilevel thresholding* possible
  - $\Rightarrow$ for N thresholds and $M = G_{max} + 1$ grey levels, maximum search in array of M^N size
- **Drawbacks**
  - assumes that $\sigma_B^2(t)$ is unimodal: not always true
  - $\sigma_B^2(t)$ is often flat, false maxima may occur
  - tends to artificially enlarge small classes
  - $\Rightarrow$ small classes may be merged and missed

Gaussian mixture modeling of Histograms



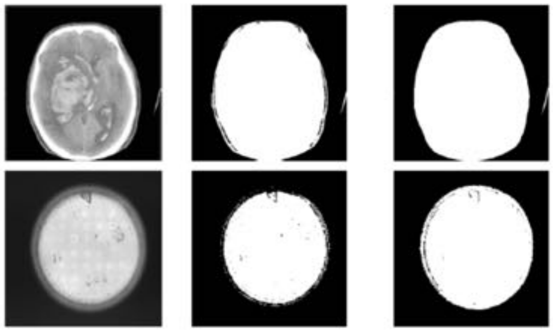
- Assume histogram $P(i)$ is mixture of **two Gaussian distributions**
- Fit this model to $P(i)$, estimate parameters of model
- Find optimal threshold **analytically** as valley in model function

Algorithm: as homework!

Properties of Gaussian Mixture approach

- **Advantages**
  - reasonably general histogram model
  - when model is valid, minimises classification error probability
  - may work for small-size classes
- **Drawbacks**
  - many histograms are not Gaussian mixtures
  - ⇒ greyvalues are **finite** and **non-negative**
  - ⇒ peak close to intenisity limit do not fit Gaussian
  - extension to multithresholding practically impossible
  - ⇒ needs irrealistic simplification of model
  - difficult to detect near and flat modes of histogram

Examples



images Otsu results Gaussian results

- Gaussian algorithm sets lower thresholds in both cases
  - ⇒ fits object contours better than Otsu

Otsu vs Gaussian approach

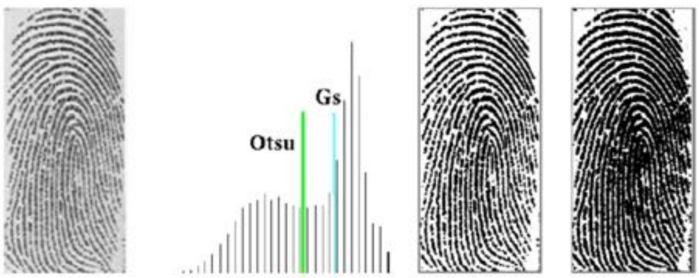


image histogram Otsu Gauss

- **Otsu** algorithm sets threshold $T = 158$ in valley
  - ⇒ lines are well-separated
- **Gaussian** algorithm sets slightly high threshold $T = 199$
  - ⇒ some lines touch

Gaussian gives poor results

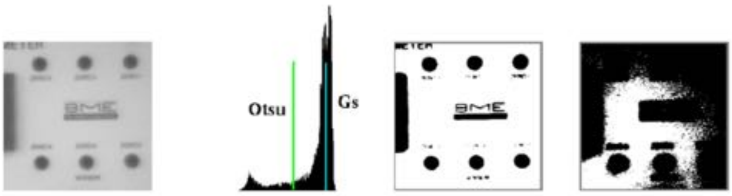


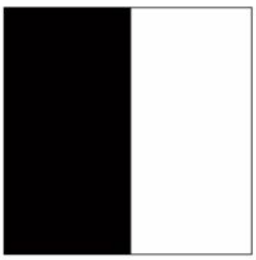
image | histogram | Otsu $T = 159$ | Gauss $T = 201$

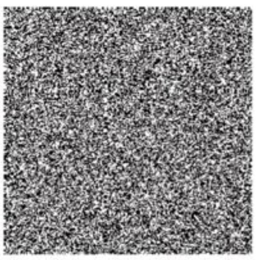
- **Otsu algorithm** finds small class of pixels (dark discs)
- **Gaussian algorithm** tries to separate two high peaks formed by background

⇒ Selects noisy valley because true class is
  - too small
  - too far away

Issues with Thresholding

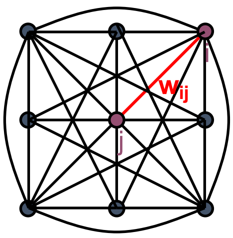
- Histogram based thresholding is very effective
- Even with low noise, if one class is much smaller than the other, we might still be in trouble
- Remember also that both these images have the same histogram:

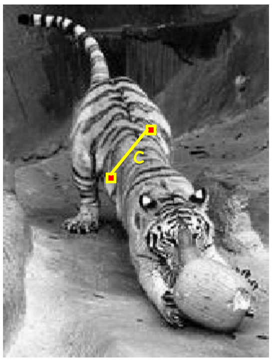




Graph-based segmentation

Images as graphs



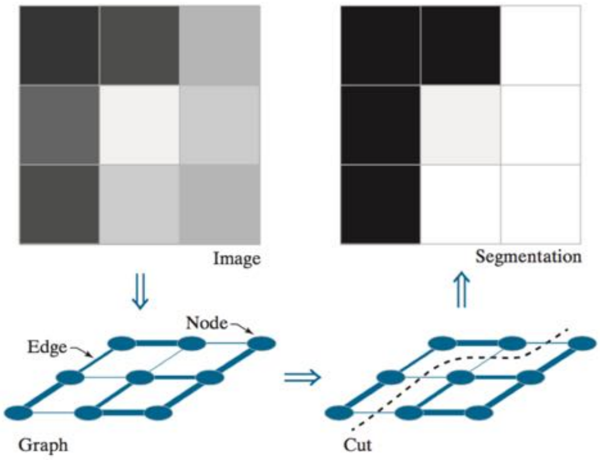


- *Fully-connected* graph
  - node for every pixel
  - link between *every* pair of pixels, **p,q**
  - similarity $\mathbf{w_{ij}}$ for each link

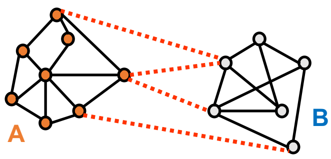
Source: Seitz

Segmentation by graph cuts

- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low cost (low similarity)
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments



Cuts in a graph



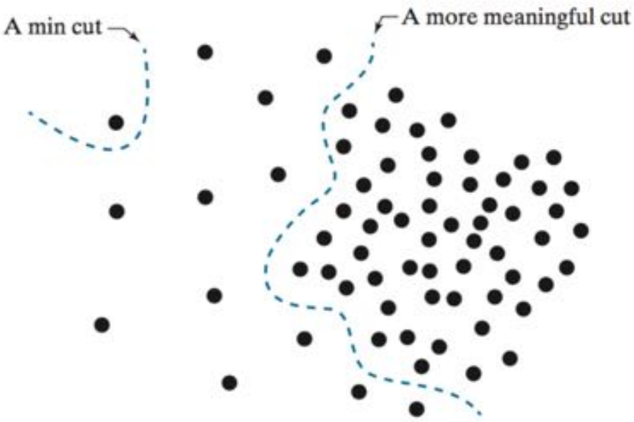
- Link Cut
  - set of links whose removal makes a graph disconnected
  - cost of a cut:

$$cut(A,B) = \sum_{p \in A, q \in B} c_{p,q}$$

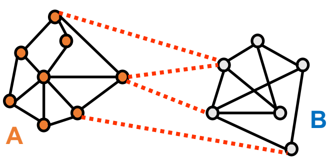
One idea: Find minimum cut
- gives you a segmentation
- fast algorithms exist for doing this

Source: Seitz

But min cut is not always the best cut...



Normalized cut



Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

$$Ncut(A,B) = \frac{cut(A,B)}{volume(A)} + \frac{cut(A,B)}{volume(B)}$$

- volume(A) = sum of costs of all edges that touch A

Source: Seitz

Normalized cut examples

Details: http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Many other segmentation algorithm

- Mean-shift
- K-mean
- Watershed
- MRFs with graph cut
- Grabcuts
- Soft segmentation
- ...