Image Segmentation

Instructor
LE Thanh Sach, Ph.D.

Lt sach@cse.hcmut.edu.vn

http://www.cse.hcmut.edu.vn/~ltsach/
Outline

- Introduction
- Histogram-based segmentation
  - Thresholding
  - Clustering
- Region growing segmentation
- Relaxation segmentation
- Expectation maximization segmentation
Introduction to image segmentation

- The purpose of image segmentation is to partition an image into meaningful regions with respect to a particular application.

- The segmentation is based on measurements taken from the image and might be greylevel, colour, texture, depth or motion.
Introduction to image segmentation

- Usually image segmentation is an initial and vital step in a series of processes aimed at overall image understanding.

- Applications of image segmentation include:
  - Identifying objects in a scene for object-based measurements such as size and shape.
  - Identifying objects in a moving scene for object-based video compression (MPEG4).
  - Identifying objects which are at different distances from a sensor using depth measurements from a laser range finder enabling path planning for a mobile robots.
Introduction to image segmentation

Example 1

- Segmentation based on greyscale
- Very simple ‘model’ of greyscale leads to inaccuracies in object labelling
Introduction to image segmentation

Example 2

- Segmentation based on texture
- Enables object surfaces with varying patterns of grey to be segmented
Introduction to image segmentation
Introduction to image segmentation

Example 3

- Segmentation based on motion
- The main difficulty of motion segmentation is that an intermediate step is required to (either implicitly or explicitly) estimate an optical flow field
- The segmentation must be based on this estimate and not, in general, the true flow
Introduction to image segmentation
Introduction to image segmentation

Example 4

- Segmentation based on depth
- This example shows a range image, obtained with a laser range finder
- A segmentation based on the range (the object distance from the sensor) is useful in guiding mobile robots
Introduction to image segmentation

Original image

Range image

Segmented image
Greylevel histogram-based segmentation

- We will look at two very simple image segmentation techniques that are based on the greylevel histogram of an image
  - Thresholding
  - Clustering

- We will use a very simple object-background test image
  - We will consider a zero, low and high noise image
Greylevel histogram-based segmentation

- Noise free
- Low noise
- High noise
Greylevel histogram-based segmentation

- How do we characterise low noise and high noise?
- We can consider the histograms of our images:
  - For the noise free image, its simply two spikes at $i=100, i=150$
  - For the low noise image, there are two clear peaks centred on $i=100, i=150$
  - For the high noise image, there is a single peak – two greylevel populations corresponding to object and background have merged
Greylevel histogram-based segmentation

![Greylevel histogram-based segmentation graph](image)
Greylevel histogram-based segmentation

- We can define the input image signal-to-noise ratio in terms of the mean greylevel value of the object pixels and background pixels and the additive noise standard deviation

$$S / N = \frac{|\mu_b - \mu_o|}{\sigma}$$
Greylevel histogram-based segmentation

- For our test images:
  - $S/N$ (noise free) = $\infty$
  - $S/N$ (low noise) = 5
  - $S/N$ (low noise) = 2
Greylevel thresholding

- We can easily understand segmentation based on thresholding by looking at the histogram of the low noise object/background image.

- There is a clear ‘valley’ between two peaks.
Greylevel thresholding
Greylevel thresholding

❖ We can define the greylevel thresholding algorithm as follows:

☞ **IF the greylevel of pixel** $p \leq T$
   
   ✓ then pixel $p$ is an object pixel

☞ **ELSE**
   
   ✓ Pixel $p$ is a background pixel
Greylevel thresholding

- This simple threshold test begs the obvious question **how do we determine the threshold?**

- Many approaches possible
  - Interactive threshold
  - Adaptive threshold
  - Minimisation method
Greylevel thresholding

- We will consider in detail a minimisation method for determining the threshold
  - Minimisation of the within group variance
  - Robot Vision, Haralick & Shapiro, volume 1, page 20
Greylevel thresholding

- Idealized object/background image histogram
Greylevel thresholding

- Any threshold separates the histogram into 2 groups with each group having its own statistics (mean, variance).
- The homogeneity of each group is measured by the within group variance.
- The optimum threshold is that threshold which minimizes the within group variance thus maximizing the homogeneity of each group.
Greylevel thresholding

- Let **group o** (object) be those pixels with greylevel \( \leq T \)
- Let **group b** (background) be those pixels with greylevel \( > T \)
- The prior probability of group o is \( p_o(T) \)
- The prior probability of group b is \( p_b(T) \)
Greylevel thresholding

- The following expressions can easily be derived for prior probabilities of object and background:

\[
p_{o}(T) = \sum_{i=0}^{T} P(i)
\]

\[
p_{b}(T) = \sum_{i=T+1}^{255} P(i)
\]

\[
P(i) = h(i) / N
\]

- where \( h(i) \) is the histogram of an \( N \) pixel image.
Greylevel thresholding

- The mean and variance of each group are as follows:

\[
\mu_o(T) = \sum_{i=0}^{T} iP(i) / p_o(T)
\]
\[
\mu_b(T) = \sum_{i=T+1}^{255} iP(i) / p_b(T)
\]
\[
\sigma^2_o(T) = \sum_{i=0}^{T} \left[ i - \mu_o(T) \right]^2 P(i) / p_o(T)
\]
\[
\sigma^2_b(T) = \sum_{i=T+1}^{255} \left[ i - \mu_b(T) \right]^2 P(i) / p_b(T)
\]
Greylevel thresholding

- The within group variance is defined as:
  \[
  \sigma_W^2(T) = \sigma_o^2(T)p_o(T) + \sigma_b^2(T)p_b(T)
  \]

- We determine the optimum \( T \) by minimizing this expression with respect to \( T \)
  
  \( \text{Only requires 256 comparisons for an 8-bit greylevel image} \)
Greylevel thresholding
Greylevel thresholding

- We can examine the performance of this algorithm on our low and high noise image
  - For the low noise case, it gives an optimum threshold of \( T = 124 \)
  - Almost exactly halfway between the object and background peaks
  - We can apply this optimum threshold to both the low and high noise images
Greylevel thresholding

Low noise image

Thresholded at $T=124$
Greylevel thresholding

Low noise image
Thresholded at $T=124$
Greylevel thresholding

Exercises:
• Implement the algorithm by MATLAB or Java/C/C++
Greylevel thresholding

- High level of pixel miss-classification noticeable
- This is typical performance for thresholding
  - The extent of pixel miss-classification is determined by the overlap between object and background histograms.
Greylevel thresholding

\[ p(x) \]

Object and Background

\[ \mu_o \quad \mu_b \]

\[ T \]
Greylevel thresholding

\[ p(x) \]

\[ \mu_o \quad \mu_b \]

Object

Background

\( T \)
Greylevel thresholding

- Easy to see that, in both cases, for any value of the threshold, object pixels will be miss-classified as background and vice versa.
- For greater histogram overlap, the pixel mis-classification is obviously greater.

We could even quantify the probability of error in terms of the mean and standard deviations of the object and background histograms.
Greylevel clustering

- Consider an idealized object/background histogram

![Diagram of greylevel clustering with object and background peaks at $c_1$ and $c_2$]
Greylevel clustering

- Clustering tries to separate the histogram into 2 groups
- Defined by two cluster centres $c_1$ and $c_2$
  - Greylevels classified according to the nearest cluster centre
Greylevel clustering

- A nearest neighbour clustering algorithm allows us perform a greylevel segmentation using clustering
  - A simple case of a more general and widely used *K-means* clustering
  - A simple iterative algorithm which has known convergence properties
Greylevel clustering

- Given a set of greylevels
  \[ \{ g(1), g(2), \ldots, g(N) \} \]

- We can partition this set into two groups
  \[ \{ g_1(1), g_1(2), \ldots, g_1(N_1) \} \]
  \[ \{ g_2(1), g_2(2), \ldots, g_2(N_2) \} \]

\[ N_1 + N_2 = N \]
Greylevel clustering

- Compute the local means of each group

\[
c_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} g_1(i)
\]

\[
c_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} g_2(i)
\]
Greylevel clustering

- Re-define the new groupings

\[ |g_1(k) - c_1| < |g_1(k) - c_2| \quad k = 1..N_1 \]

\[ |g_2(k) - c_2| < |g_2(k) - c_1| \quad k = 1..N_2 \]

- In other words all grey levels in set 1 are nearer to cluster centre \( c_1 \) and all grey levels in set 2 are nearer to cluster centre \( c_2 \)
Greylevel clustering

- But, we have a *chicken and egg* situation
  - The problem with the above definition is that each group mean is defined in terms of the partitions and vice versa
  - The solution is to define an iterative algorithm and worry about the convergence of the algorithm later
Greylevel clustering

The iterative algorithm is as follows

1. Initialize the label of each pixel randomly
2. Repeat
   - \( c_1 \) = mean of pixels assigned to object label
   - \( c_2 \) = mean of pixels assigned to background label
3. Compute partition \( \{ g_1(1), g_1(2), \ldots, g_1(N_1) \} \)
4. Compute partition \( \{ g_2(1), g_2(2), \ldots, g_2(N_2) \} \)

Until none pixel labelling changes
Greylevel clustering

- Two questions to answer
  - Does this algorithm converge?
  - If so, to what does it converge?

- We can show that the algorithm is guaranteed to converge and also that it converges to a sensible result
Greylevel clustering

- Outline proof of algorithm convergence
  - Define a ‘cost function’ at iteration \( r \)

\[
E^{(r)} = \frac{1}{N_1} \sum_{i=1}^{N_1} \left( g_1^{(r)}(i) - c_1^{(r-1)} \right)^2 + \frac{1}{N_2} \sum_{i=1}^{N_2} \left( g_2^{(r)}(i) - c_2^{(r-1)} \right)^2
\]

- \( E^{(r)} > 0 \)
Greylevel clustering

- Now update the cluster centres

\[ c_1^{(r)} = \frac{1}{N_1} \sum_{i=1}^{N_1} g_1^{(r)}(i) \]

\[ c_2^{(r)} = \frac{1}{N_2} \sum_{i=1}^{N_1} g_2^{(r)}(i) \]

- Finally update the cost function

\[ E_1^{(r)} = \frac{1}{N_1} \sum_{i=1}^{N_1} (g_1^{(r)}(i) - c_1^{(r)})^2 + \frac{1}{N_2} \sum_{i=1}^{N_2} (g_2^{(r)}(i) - c_2^{(r)})^2 \]
Greylevel clustering

- Easy to show that
  \[ E^{(r+1)} < E_1^{(r)} < E^{(r)} \]

- Since \( E^{(r)} > 0 \), we conclude that the algorithm must converge
  \[ \text{but} \]

- What does the algorithm converge to?
Greylevel clustering

$E_1$ is simply the sum of the variances within each cluster which is minimised at convergence.

- Gives sensible results for well separated clusters
- Similar performance to thresholding
Greylevel clustering

\( g_1 \)

\( g_2 \)

\( c_1 \)

\( c_2 \)
Exercises:
Implement Clustering Segmentation by Matlab, Java/C/C++
Region Growing

Seed pixels → Expand Area using predefined criteria → Larger region

Grow $S_1$

$S_1 = 1$
$E_1 = 2$
$S_2 = 7$
$E_2 = 3$

$S_1 \Rightarrow \mu = \frac{5}{8}$
$S_2 \Rightarrow \mu = \frac{83}{13}$

Grow $S_2$

$S_1 \Rightarrow \mu = \frac{6}{10} = \frac{3}{5}$
Region Growing (Flow chart)

Select seed pixels or seed region

Set Criteria (E)
i=1; j=1

Compare Features

Meet criteria?

Y

Combine seed region and input pixel or input region

Update µ

All pixels or regions around seed region?

N

j=j+1

Y

Change new seed
i=i+1

i>MAX?

Y

Combining occurrence?

N

END
Relaxation labelling

- All of the segmentation algorithms we have considered thus far have been based on the histogram of the image. This ignores the greylevels of each pixels’ neighbours which will strongly influence the classification of each pixel.
- Objects are usually represented by a spatially contiguous set of pixels.
Relaxation labelling

The following is a trivial example of a likely pixel miss-classification

- Object
- Background

https://fb.com/tailieudientucntt
Relaxation labelling

- *Relaxation labelling* is a fairly general technique in computer vision which is able to incorporate constraints (such as spatial continuity) into image labelling problems.

- We will look at a simple application to greylevel image segmentation.

- It could be extended to colour/textures/motion segmentation.
Relaxation labelling

- Assume a simple object/background image
  - $p(i)$ is the probability that pixel $i$ is a background pixel
  - $(1- p(i))$ is the probability that pixel $i$ is an object pixel
Relaxation labelling

- Define the 8-neighbourhood of pixel $i$ as 
  $$\{i_1, i_2, \ldots, i_8\}$$
Relaxation labelling

- Define *consistencies* $c_s$ and $c_d$
  - Positive $c_s$ and negative $c_d$ encourages neighbouring pixels to have the same label
  - Setting these consistencies to appropriate values will encourage spatially contiguous object and background regions
Relaxation labelling

- We assume again a bi-modal object/background histogram with maximum grey level $g_{\text{max}}$. 

![Graph showing bi-modal distribution with peaks at different grey levels]

0 $g_{\text{max}}$
Relaxation labelling

- We can initialize the probabilities

\[ p^{(0)}(i) = \frac{g(i)}{g_{\text{max}}} \]

- Our relaxation algorithm must ‘drive’ the background pixel probabilities \( p(i) \) to 1 and the object pixel probabilities to 0
Relaxation labelling

- We want to take into account:
  - Neighbouring probabilities $p(i_1), p(i_2), \ldots p(i_8)$
  - The consistency values $c_s$ and $c_d$

- We would like our algorithm to ‘saturate’ such that $p(i) \sim 1$

- We can then convert the probabilities to labels by multiplying by 255
Relaxation labelling

We can derive the equation for relaxation labelling by first considering a neighbour $i_1$ of pixel $i$.

- We would like to evaluate the contribution to the increment in $p(i)$ from $i_1$.
- Let this increment be $q(i_1)$.

We can evaluate $q(i_1)$ by taking into account the consistencies.
Relaxation labelling

- We can apply a simple decision rule to determine the contribution to the increment $q(i_k)$ from pixel $i_k$
  - If $p(i_k) > 0.5$ the contribution from pixel $i_k$ increments $p(i)$
  - If $p(i_k) < 0.5$ the contribution from pixel $i_k$ decrements $p(i)$

These rules are valid with respect to the bimodel in Page 61
Relaxation labelling

- Since $c_s > 0$ and $c_d < 0$ it's easy to see that the following expression for $q(i_1)$ has the right properties:

$$q(i_1) = c_s p(i_1) + c_d (1 - p(i_1))$$

- We can now average all the contributions from the 8-neighbours of $i$ to get the total increment to $p(i)$:

$$\Delta p(i) = \frac{1}{8} \sum_{h=1}^{8} (c_s p(i_h) + c_d (1 - p(i_h)))$$
Relaxation labelling

- Easy to check that $-1 < \Delta p(i) < 1$ for $-1 < c_s, c_d < 1$
- Can update $p(i)$ as follows

$$p^{(r)}(i) \sim p^{(r-1)}(i)(1 + \Delta p(i))$$

- Ensures that $p(i)$ remains positive
- Basic form of the relaxation equation
Relaxation labelling

- We need to *normalize* the probabilities $p(i)$ as they must stay in the range \( \{0..1\} \)
  - After every iteration $p^{(r)}(i)$ is rescaled to bring it back into the correct range

- Remember our requirement that likely background pixel probabilities are ‘driven’ to 1
Relaxation labelling

One possible approach is to use a constant normalisation factor

\[ p^{(r)}(i) \mapsto \frac{p^{(r)}(i)}{\max_i p^{(r)}(i)} \]

In the following example, the central background pixel probability may get stuck at 0.9 if \( \max(p(i)) = 1 \)

\[
\begin{array}{ccc}
0.9 & 0.9 & 0.9 \\
0.9 & 0.9 & 0.9 \\
0.9 & 0.9 & 0.9 \\
\end{array}
\]
Relaxation labelling

- Algorithm performance on the high noise image
  - Comparison with thresholding

High noise circle image  · Optimum’ threshold  · Relaxation labeling - 20 iterations
Relaxation labelling

The following is an example of a case where the algorithm has problems due to the thin structure in the *clamp* image.

*clamp* image original  *clamp* image noise added  segmented *clamp* image 10 iterations
Relaxation labelling

- Applying the algorithm to normal greyscale images we can see a clear separation into *light* and *dark* areas.

![Original](image1.jpg) ![2 iterations](image2.jpg) ![5 iterations](image3.jpg) ![10 iterations](image4.jpg)
Relaxation labelling

- The histogram of each image shows the clear saturation to 0 and 255.
In relaxation labelling we have seen that we are representing the probability that a pixel has a certain label.

In general we may imagine that an image comprises \( L \) segments (labels):

- Within segment \( l \) the pixels (feature vectors) have a probability distribution represented by \( p_l(x \mid \theta_l) \)
- \( \theta_l \) represents the parameters of the data in segment \( l \):
  - Mean and variance of the greylevels
  - Mean vector and covariance matrix of the colours
  - Texture parameters
The Expectation/Maximization (EM) algorithm
Once again a *chicken and egg* problem arises:

- If we knew $\theta_i : l = 1..L$ then we could obtain a labelling for each $x$ by simply choosing that label which maximizes $p_l(x | \theta_i)$.
- If we knew the label for each $x$ we could obtain $\theta_i : l = 1..L$ by using a simple maximum likelihood estimator.

The EM algorithm is designed to deal with this type of problem but it frames it slightly differently:

- It regards segmentation as a missing (or incomplete) data estimation problem.
The Expectation/Maximization (EM) algorithm

- The **incomplete** data are just the measured pixel greylevels or feature vectors.  
  *We can define a probability distribution of the incomplete data as* \( p_i(x; \theta_1, \theta_2, \ldots, \theta_L) \)

- The **complete** data are the measured greylevels or feature vectors *plus* a mapping function \( f(\cdot) \) which indicates the labelling of each pixel.  
  *Given the complete data (pixels plus labels) we can easily work out estimates of the parameters* \( \theta_l : l = 1, \ldots, L \)  
  *But from the incomplete data no closed form solution exists.*
The Expectation/Maximization (EM) algorithm

- Once again we resort to an iterative strategy and hope that we get convergence
- The algorithm is as follows:
  - Initialize an estimate of $\theta_i : l = 1..L$
  - Repeat
    - **Step 1**: (E step)
      - Obtain an estimate of the labels based on the current parameter estimates
    - **Step 2**: (M step)
      - Update the parameter estimates based on the current labelling
  - Until Convergence
The Expectation/Maximization (EM) algorithm

A recent approach to applying EM to image segmentation is to assume the image pixels or feature vectors follow a mixture model.

- Generally we assume that each component of the mixture model is a Gaussian.

A Gaussian mixture model (GMM)

\[
p(x | \Theta) = \sum_{l=1}^{L} \alpha_l p_l(x | \theta_l)
\]

\[
p_l(x | \theta_l) = \frac{1}{(2\pi)^{d/2}\det(\Sigma_l)^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_l)^T \Sigma_l^{-1}(x - \mu_l)\right)
\]

\[
\sum_{l=1}^{L} \alpha_l = 1
\]
The Expectation/Maximization (EM) algorithm

- Our parameter space for our distribution now includes the mean vectors and covariance matrices for each component in the mixture plus the mixing weights

\[ \Theta = (\mu_1, \Sigma_1, \alpha_1, \ldots, \mu_L, \Sigma_L, \alpha_L) \]

- We choose a Gaussian for each component because the ML estimate of each parameter in the E-step becomes linear
The Expectation/Maximization (EM) algorithm

- Define a posterior probability $P(l \mid x_j, \theta_l)$ as the probability that pixel $j$ belongs to region $l$ given the value of the feature vector $x_j$

- Using Bayes rule we can write the following equation

$$P(l \mid x_j, \theta_l) = \frac{\alpha_l p_l(x_j \mid \theta_l)}{\sum_{k=1}^L \alpha_k p_k(x \mid \theta_k)}$$

- This actually is the E-step of our EM algorithm as allows us to assign probabilities to each label at each pixel
The Expectation/Maximization (EM) algorithm

- The M step simply updates the parameter estimates using ML estimation

\[
\alpha_{l}^{(m+1)} = \frac{1}{n} \sum_{j=1}^{n} P(l \mid x_j, \theta_l^{(m)})
\]

\[
\mu_{l}^{(m+1)} = \frac{\sum_{j=1}^{n} x_j P(l \mid x_j, \theta_l^{(m)})}{\sum_{j=1}^{n} P(l \mid x_j, \theta_l^{(m)})}
\]

\[
\Sigma_{l}^{(m+1)} = \frac{\sum_{j=1}^{n} P(l \mid x_j, \theta_l^{(m)}) \{(x_j - \mu_l^{(m)})(x_j - \mu_l^{(m)})^T\}}{\sum_{j=1}^{n} P(l \mid x_j, \theta_l^{(m)})}
\]
Conclusion

- We have seen several examples of greylevel and colour segmentation methods:
  - Thresholding
  - Clustering
  - Relaxation labelling
  - EM algorithm

- Clustering, relaxation labelling and the EM algorithm are general techniques in computer vision with many applications.