Image Processing and Imaging

Image Segmentation

Dominik Söllinger
Fachbereich AIHI
Universität Salzburg

Wintersemester 2022/23

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Motivation

What is segmentation?

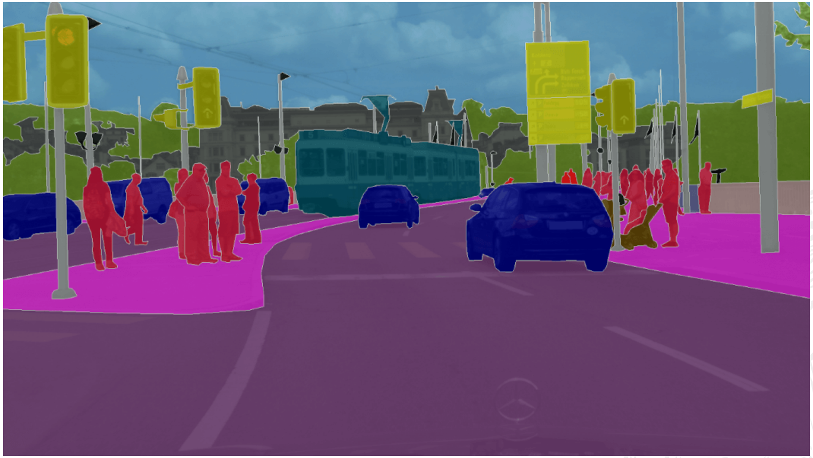
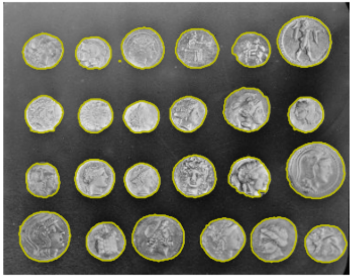
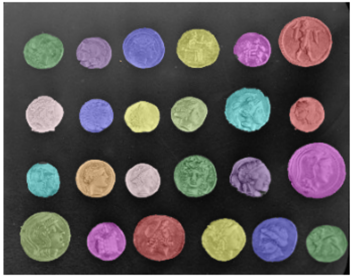


Image Segmentation

- Image segmentation is the division of an image into **regions** or **objects**
- Spatially close regions very often can be combined to an **object**
- Objects are typically what we are interested in (depends on the application). The rest of the image is background.





Complete and partial image segmentation (1)

Complete Segmentation

- Divides an image into non-overlapping regions that match to the real world objects
- Nowadays this is very often tackled with deep learning (DL) based techniques (see **semantic segmentation** or **instance segmentation**)
- Simple tasks can be approached with simple (non-DL based) techniques
  - Text with letters and numbers
  - Objects in front of uniform background or high contrast

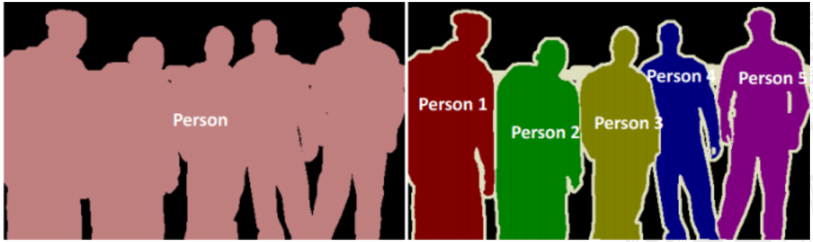


Figure: Semantic segmentation (left) vs. Instance segmentation (right)

Complete and partial image segmentation (2)

Partial Segmentation:

- Regions which are homogeneous with respect to some criterion are identified
- These regions do NOT necessarily correspond to real-world objects
- In many cases, over-segmentation occurs, i.e. to many regions are found

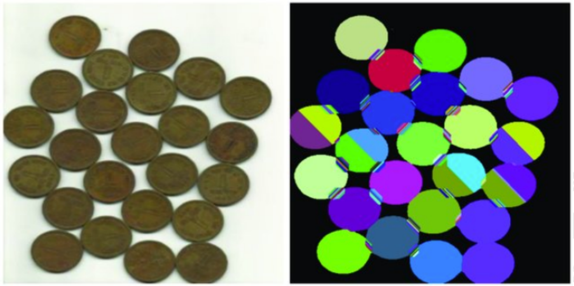


Figure: Example of oversegmentation

Categories of segmentation-based techniques

Segmentation techniques can be grouped into three categories:

1. **Thresholding-based**: Use global information of the image or parts of it (e.g. the histogram).
2. **Edge-based**: An edge filter is applied to the image. Closed edge-chains are generated as object borders
3. **Region-based**: Group together pixels which are neighbours and have similar values and splitting groups of pixels which are dissimilar in value

- The latter two techniques solve a dual problem:
  Each region can be described by its closed border curve and each curve describes a region which is enclosed by the curve

Outline

1 Introduction

2 Thresholding-based segmentation

- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation

- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation

- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Thresholding

- Great due to its **simplicity** and **computational speed**
- A threshold T is determined to separate objects and the background
- Input image $f(x, y)$ is transformed into a binary segmented output image $g(i, j)$ as follows:

$$g(i,j) = \begin{cases} 1 & f(i,j) \geq T \\ 0 & f(i,j) < T \end{cases}$$

- If T is constant over the entire image, we call refer to it as **global thresholding**
- Thresholding is a suited approach if objects are not joined and the object's gray-scale values are different from background's gray-scale values

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Threshold Variations (1)

- Using a one fixed threshold is successful in rare cases only
- **Variable thresholding** is often more successful. The value of T is changed dependent on a local image region's characteristics.

Thresholding Variations (2)

Using a single threshold T typically works well if the distribution of the is bimodal.
If the histogram has, for example, three dominant modes (i.e., cause by two types of light objects on a dark background) we require **multiple thresholds**.

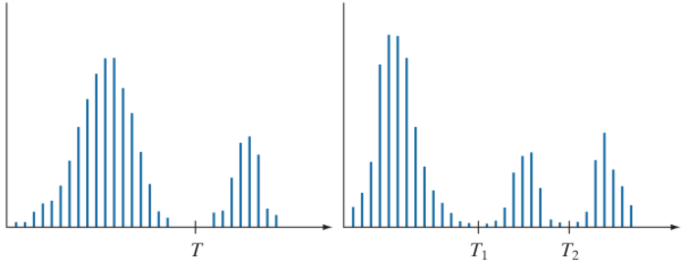


Figure: Intensity histograms that can be partioned by a single threshold (left) and by dual thresholds (right)

Thresholding Variations (3)

Band Thresholding:

- Grayscales values in a specific grayscale range are determined to be object pixels
- Example application: CT imaging — Different tissue types cause different grayscale values

$$g(i,j) = \begin{cases} 1 & f(i,j) \in D \\ 0 & \text{otherwise} \end{cases}$$

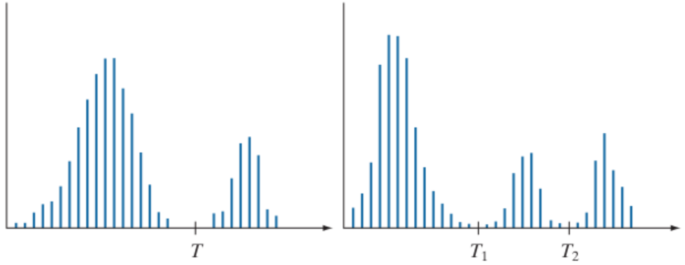


Figure: Intensity histograms that can be partioned by a single threshold (left) and by dual thresholds (right)

Thresholding Variations (4)

Multi Thresholding: Similar to band thresholding but the output image is not binary

$$g(i,j) = \begin{cases} 1 & \text{for } f(i,j) \in D_1 \\ 2 & \text{for } f(i,j) \in D_2 \\ 3 & \text{for } f(i,j) \in D_3 \\ \ldots \end{cases}$$

Semi Thresholding: Removes the background but keeps the grayscale information in the objects

$$g(i,j) = \begin{cases} f(i,j) & \text{for } f(i,j) \geq T \\ 0 & \text{otherwise} \end{cases}$$

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Threshold Selection

The crucial question — How to choose a threshold?

In general we can say, the more **priori information** the better.

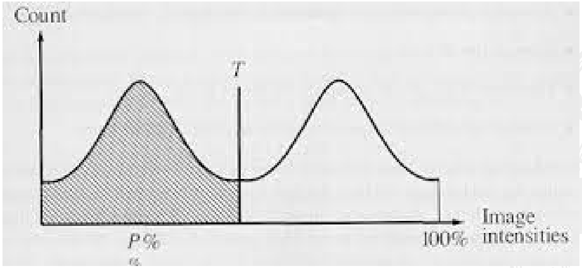
Illustrative examples:

- Shape of the object region
- Position or orientation of the object
- Known initial and final point of the boundary
- Relation of the region considered to other regions with required properties (e.g., above or inside)

P-Tile thresholding

- **Assumption:**
  We have dark objects against a light background and can estimate the area/size of the objects present in the image (e.g., written letters on a sheet)

- If p % of the image is occupied by the objects of interest, choose the threshold T so that the p % darkest pixels are classified as object pixels.
- **How?** Compute the commulative histogram distribution. Select T such that $1/p$ of the image is $\leq T$.



Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Optimal thresholding

- Optimal thresholding selects a threshold value that is statistically optimal
- If the histogram is **bimodal**, the threshold is selected to be the minimal value between the two extrema
- If the histogram is **multimodal**, the thresholds can be selected between two corresponding maxima.

Attention: A biomodal histogram does not guarantee a correct segmentation.
Example: 50% B/W pixel mixed or concentrated on one image side result in an identical bimodal histogram.

Concept of Optimal Thresholding

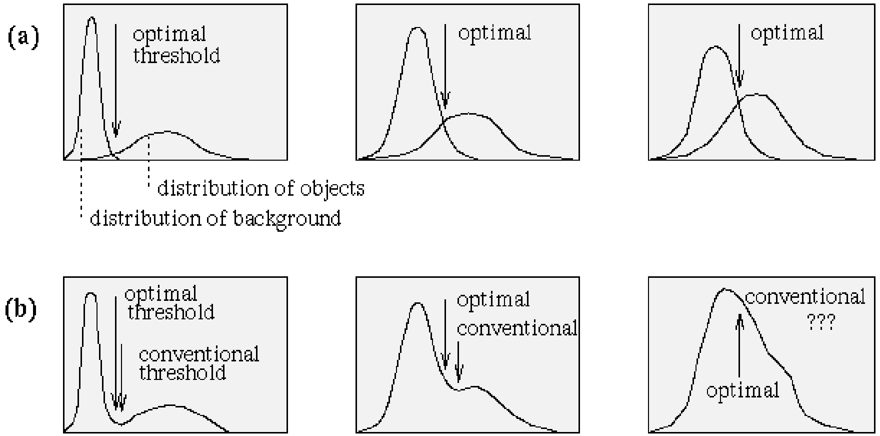


Figure: Concept of optimal thresholding, first row represents the individual distribution, second row is the combined distribution.

Optimal global thresholding using Iterative Threshold Selection (1)

Assumption: Existence of regions with two dominant grayscale values.

Algorithm:

1. Select an initial estimate for the global threshold T^0.
2. Produce two regions by global thresholding with T^0: Foreground and object region
3. Compute the mean intensity value μ_B^t of the background and object region μ_O^t.

$$\mu_B^t = \frac{\sum_B f(i,j)}{\text{number of background pixel}}$$
$$\mu_O^t = \frac{\sum_O f(i,j)}{\text{number of object pixel}}$$

4. Compute a new threshold value midway between μ_B and μ_O.

$$T^{t+1} = \frac{\mu_B^t + \mu_O^t}{2}$$

5. Repeat Steps 2 through 4 until the difference between values of T^{t+1} in successive iterations is smaller than a predefined value ΔT.

Optimal global thresholding using Iterative Threshold Selection (2)

How to choose the initial threshold value?

- Assuming that the corner pixels contain background values → Choose the average corner pixel value
- Choose the global average pixel value
- **Note:** The initial intensity value must be greater than the minimum and less than the maximum intensity value.

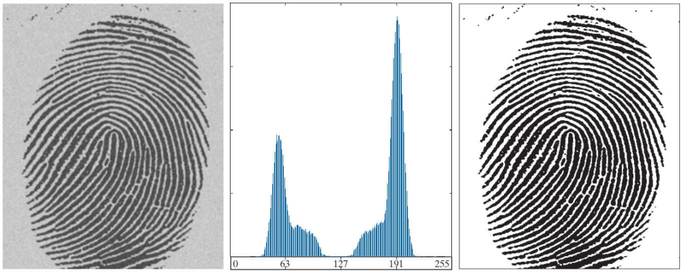


Figure: Example of segmentation using the preceding iterative algorithm. Application of the algorithm resulted in the threshold T=125 4 after three iterations, starting with T equal to the average intensity of the image, and using $\Delta T = 0$.

Optimal thresholding using Otsu's method (1)

Idea: View thresholding as statistical-decision theory problem $\rightarrow$ Minimize the average error that occurs when assigning pixels to two or more regions (also called classes).

How?

By choosing a threshold T that **minimizes the intra-class variance** σ^2_{intra}, defined as a **weighted sum of variances** of the two classes.

$$\sigma^2_{intra}(T) = q_O(T)\sigma^2_O(T) + q_B(T)\sigma^2_B(T)$$

where

$\sigma^2_O(T)$ ... Threshold-dependent variance the object region
$\sigma^2_B(T)$ ... Threshold-dependent variance of the background region
$q_O(T)$, $q_B(T)$ ... Weight terms

Optimal thresholding using Otsu's method (2)

The **weight terms** are chosen based on the **probability of the class** occuring.

Assuming that $P(i)$ denotes the normalized histogram, $q_O(T)$ and $q_B(T)$ can simply be computed as follows:

$$q_O(T) = \sum_{i=0}^{t-1} P(i) \qquad \text{and} \qquad q_B(T) = \sum_{i=T}^{L-1} P(i)$$

Calculating the class variance $\sigma_O^2(T)$ where $P_{O,T}(i)$ is the probability distribution of class O:

$$\begin{aligned}\sigma_O^2(T) &= \sum_{i=1}^{T} (i - \mu_O(T))^2 P_{O,T}(i) \\ &= \sum_{i=1}^{T} (i - \mu_O(T))^2 P(i|O) \qquad \text{[Apply Bayes rule]} \\ &= \sum_{i=1}^{T} (i - \mu_O(T))^2 P(O|i)P(i)/P(O)\end{aligned}$$

Optimal thresholding using Otsu's method (3)

Finally we obtain for $\sigma_O^2(T)$:

$$\sigma_O^2(T) = \sum_{i=1}^{T} (i - \mu_O(T))^2 \frac{P(i)}{q_O(t)}$$

Similary, we can derive the **formula for the object mean** $\mu_O(T)$:

$$\mu_O(t) = \sum_{i=1}^{t} i \frac{P(i)}{q_O(t)}$$

(background mean and variance by analogy)

Last but not least, all we need to do is just **run through the full range of** T **values** and **pick the value that minimizes** $sigma_{intra}^2$.

Optimal thresholding using Otsu's method (4)

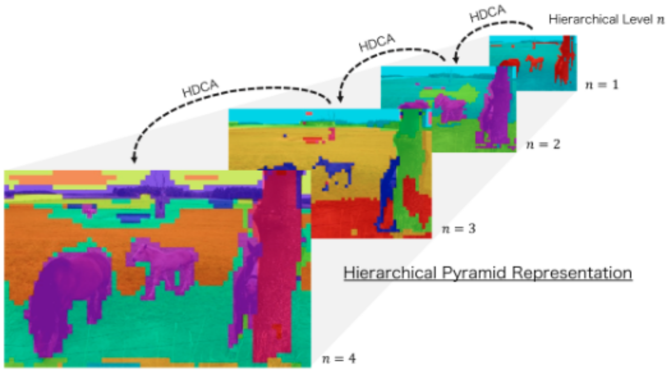
Final note: Having to do this computation for the entire range of pixel values is not very efficient computationally. However, there also exists a **fast recursive approach**.

The recursive approachs, exploits the relationship between the intra-class σ^2_{intra} and inter-class σ^2_{inter} variances $\rightarrow$ Minimizing the intra-class variance is the same as **maximizing inter-class variance**.

There exists a fast recursive algorithm to maximizing inter-class variance (see script for more details).

Further variants - Hierarchical Data Structures (1)

Basic idea: View the image as a pyramidal data structure. Identify regions in the low resolution image and refine them in the higher resolutions



Advantage: Robustness against noise, since the initial segmentations start with a significantly smoothed image.

Further variants - Hierarchical Data Structures (2)

Variants:

- Segment the low-resolution image. In the next higher resolution pixels close to the region border are re-assigned to object(s) or background (depending on a similarity criterion) $\rightarrow$ Repeat until the highest resolution
- Apply a *significant pixel detector* (*) in the lowest resolution which identifies pixels different to their neighbourhood.

  The corresponding image part in the full-resolution image is thresholded with T being between the gray-scale of the significant pixel and the average of the other 8-neighbours (this is done locally with different thresholds for different regions).

  (*) e.g, a 3x3 masks which indicates if the central pixel is different from its neighbors

Further variants - Variable Local Thresholding (1)

Basic idea: Adapt the threshold for every point in the image based on properties of the local neighbourhood (e.g., based on its mean and standard deviation).

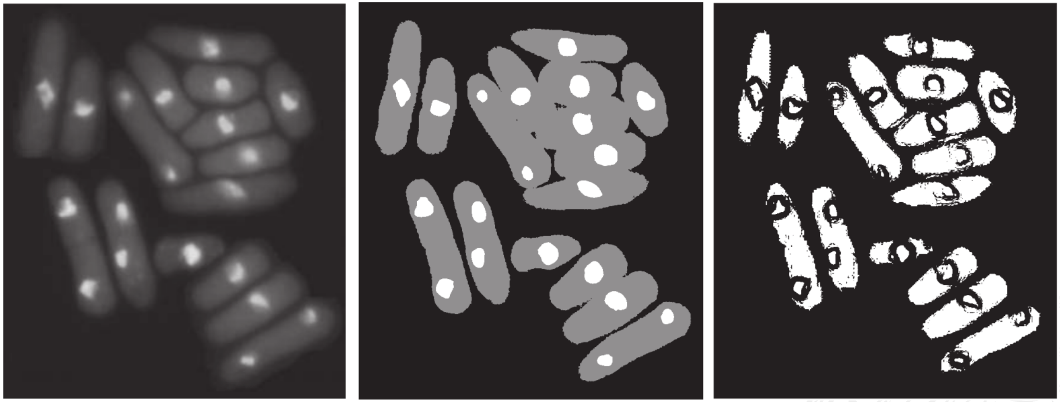
Illustrative example:
Let m_{xy} and σ_{xy} denote the mean and standard deviation of a pixel (x, y)'s local neighbourhood.

For instance, we can compute the segmentation mask $g(x, y)$:

$$g(x,y) = \begin{cases} 1 & f(x,y) \geq a\sigma_{x,y} \text{ AND } f(x,y) \geq b\mu_{x,y} \\ 0 & \text{otherwise} \end{cases}$$

where a and b are nonnegative constants.

Further variants - Variable Local Thresholding (2)



(a) (b) (c)

Figure: Segmentation of a yeast image. (a) Original yeast image. (b) Yeast image segmented using a global dual thresholding approach. The mid-gray regions on the right side of the image were not separated properly. (c) Yeast image segmented using variable thresholding ($a = 30$ and $b = 1.5$). All regions are nicely separated.

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Region-based techniques

- **Idea:** Partition the image into regions of maximal homogeneity
- Primary applied to noisy images or in circumstances where the difference between objects is not "just" the luminance (e.g. they exhibit a different structures — texture)

Homogeneity:

- We need a metric for measuring the similarity between pixels/regions
- For example: average gray-value, shape of a local histogram, texture properties
- Target regions should exhibit the following property (apart from their zero intersection):

$$\begin{aligned} H(R_i) &= \text{True} && i = 1, \dots, S \\ H(R_i \cup R_j) &= \text{False} && i \neq j \text{ AND } R_i \text{ adjacent to } R_j \end{aligned}$$

  S ... number of regions, $H(R_i)$ is a binary evaluation of homogeneity of R_i
- This means that regions are homogeneous and maximal (i.e. if they would be larger, homogeneity is lost)

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Region Growing (1)

Idea: Group pixels or subregions into larger regions while the homogeneity criterion is satisfied.

Compare one pixel with its neighbors. If the homogeneity criterion is satisfied, the pixel belong to the same region.

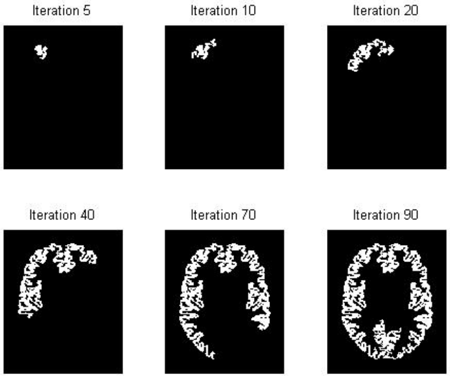


Figure: Illustrative example of region growing

Region Growing (2)

Region Growing Algorithm:

1. Choose seed pixel(s)
2. Check neighbouring pixels following a chosen strategy and add them to the region if they are "similar" to the seed pixel
3. Repeat (2) and (3) for the newly added pixels; stop if no more pixels can be added

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Region Merging and Splitting (1)

Region Growing requires seed points.

Instead, **Region Splitting and Merging** subdivides the entire image into a set of disjoint regions and then merges/splits those regions.

Region Merging and Splitting (2)

Region Splitting:
The goal is to recursively split the image into regions until all regions satisfiy the homogeneity requirement.

Algorithm:

1. Start with the whole image. If the homogeneity criterion is not satisfied, split the image into four disjoint quadrants.
2. Keep subdividing the quadrants until the homogeneity is satisfied for all quadrants of them.

The splitting technique has a convenient representation in the form of structure called **quadtree**.

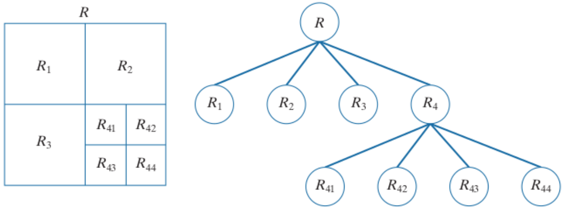


Figure: Left: Partitioned image. Right: Corresponding quadtree

Region Merging and Splitting (3)

Note that if only splitting is used, the final partition often contains adjacent regions with identical properties. We can combine those regions with **region merging**.

Region Merging Algorithm:

1. Start with a set of regions which satisfy the homogeneity requirement.
2. Keep merging regions until the homogeneity requirement is no longer satisifed.

Region Merging and Splitting (4)

Splitting/merging techniques differ in terms of their initial segmentations and the different criteria for homogeneity.

Improvements: Additional employment of edge information

- Neighbouring regions are merged if a significant share of their common border consists of weak edges
- Result of edge relaxation can be used to determine if an edge is weak or not

Region Splitting and Merging Comparison (1)

Remark: Region splitting does not result in the same segmentation even if the same homogeneity criteria are used.

Example: Segmentation of the chessboard pattern using either region splitting or merging. We consider the homogeneity criterion $H(R_i)$.

$$H(R_i) = \begin{cases} \mathit{TRUE} & \text{if the region contains the same number of B/W cells} \\ \mathit{FALSE} & \text{else} \end{cases}$$

Region Splitting and Merging Comparison (2)

$$H(R_i) = \begin{cases} TRUE & \text{if the region contains the same number of B/W cells} \\ FALSE & \text{else} \end{cases}$$

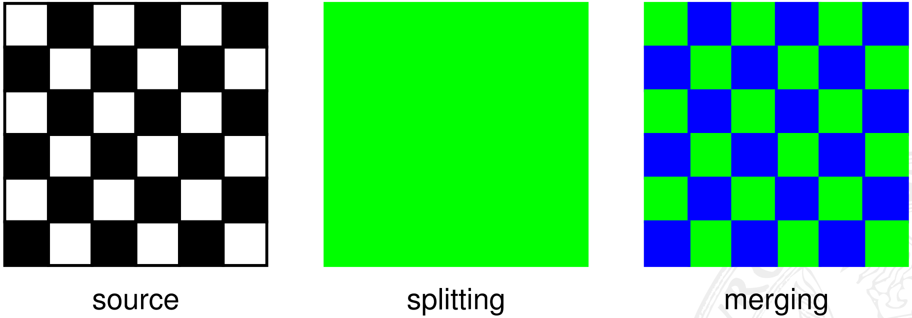


Figure: Left: Non-segmented chessboard image. Center: Region splitting (the upper pyramid level is homogeneous, no splitting possible). Right: Region merging (lowest pyramid level initially consists of inhomogeneous regions, no merging possible)

Split-and-Merge Algorithm

In practice, typically a combination of splitting and subsequent merging is applied. The employed data structure is often a quad-tree.

1. Define an initial segmentation into regions, a criterion for homogeneity and a pyramidal data structure
2. In case a region in the data structure is not homogeneous, it is split into its four children regions; In case four regions corresponding to the same parent can be merged, they are merged. If no further region can be processed, GOTO 3)
3. In case two neighbouring regions (either in different levels of the pyramid or with different parent nodes) can be merged according the criterion for homogeneity they are merged
4. Regions being too small are merged with the most similar neighbouring region

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

K-Means Clustering Segmentation (1)

Idea: Image segmentation can be viewed as a clustering task. "Similar" pixels should be clustered into one region

Procedure:

1. Represent each pixel in the image with a vector (e.g. intensity, colour and location, texture descriptors, etc.)
2. Choose distance weights (which vector component is more important than others, e.g. color vs. location)
3. Apply k-means clustering
4. Pixels belong to the segment corresponding to cluster centers

K-Means Clustering Segmentation (2)

K-means Clustering Algorithm:

Let $\{x_1, \ldots, x_n\}$ be a set of pixels (observations), where each observation x is a vector and has the form $x = [x_1, ..., x_D]^T$. For instance, in case of a grayscale image, x is a scalar.

The aim is to partition the n pixels into K sets $S = \{S_1, S_2, \ldots, S_K\}$ in a way that the **within-cluster sum of squares (WCSS) is minimized**.

$$\arg\min_S \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

where μ_i the mean or centroid of pixels in cluster S_i

K-Means Clustering Segmentation (3)

Lloyd Algorithm

1. Specify an initial set of means μ_i.
2. Assign samples to clusters: Each sample should be assigned to the cluster set whose mean is the closest.

$$S_i^{(t)} = \{x_j : ||x_j - m_i^{(t)}||^2 \leq ||x_j - m_{\mathring{i}}^{(t)}||^2 \textit{ for all } \mathring{i} = 1, ..., k\}$$

3. Update the cluster centers (means)

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

4. Repeat (2) and (3) until the WCSS is less than certain threshold T.

K-Means Clustering Segmentation (4)

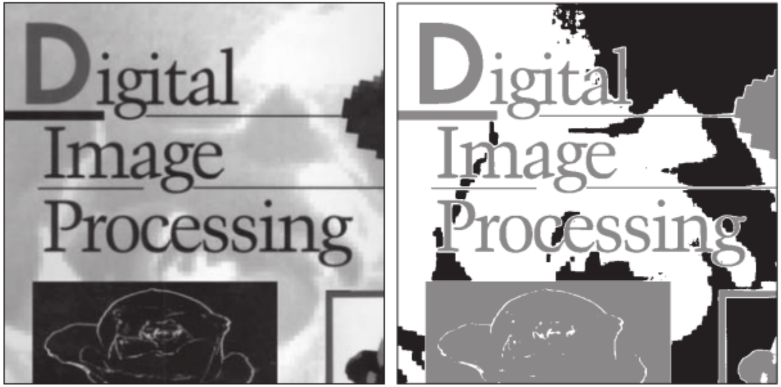


Figure: Left: Original image. Right: K-Means clustered image with $k = 3$. Clustering was done solely based on the pixel intensities

K-Means Clustering Segmentation (5)

Drawbacks:

- Sensitivity to initialisation
  - How many cluster centers K?
  - How should we initialize μ_i (e.g., random uniform)?
- Sensitivity to outliers
- Spherical clusters are assumed by the model. However, depending on the distribution of points in space, a sphere might not be tightly enclose the class' features.

A solution is provided by the **Mean Shift Algorithm**.

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Mean Shift Segmentation (1)

Advantages compared to K-Means:

- Non-parametric clustering technique
- No prior knowledge of the number of clusters is required
- No restriction on the shape of the clusters

Idea:

- Clusters are places where data points (i.e. pixels) are closely together in the feature space $\rightarrow$ regions with a high density
- Instead of initialising the cluster centers and selecting the number of cluster, the algorithm **seeks modes or local maxima of density in the feature space**.
- Each modes defines a regions. Each pixel belongs to a mode and therefore a region.

Mean Shift Segmentation (2)

Mean Shift Procedure:

1. Selects a pixel and apply a window in feature space around the pixels' feature
2. Compute the mean of the feature vectors in the neighbourhood
3. Shift the center of the windows to the position of the mean (centroid)
4. Repeat step 1 to 3 until the mean does not change its position
5. A local maximum – a mode – is found

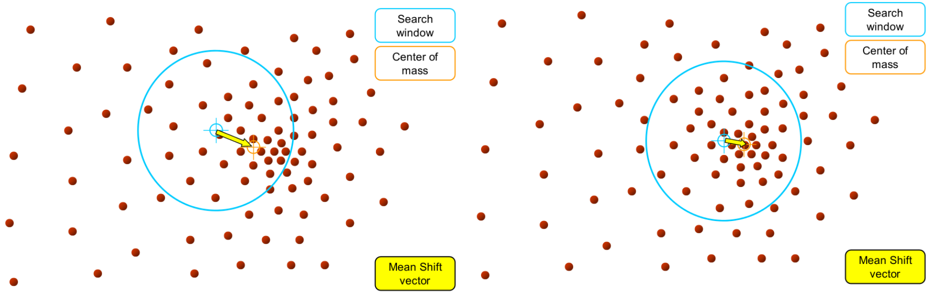


Figure: Mean-shift procedure in feature space

Mean Shift Segmentation (3)

- A set of all feature vectors that converge to the same mode defines the basin of attraction of that mode
- Points – pixels – which are in the same basin of attraction are associated with the same cluster and form image regions
- Kernel functions to determine if a data point is within a cluster or not: usually either flat kernel or Gaussian kernels are used

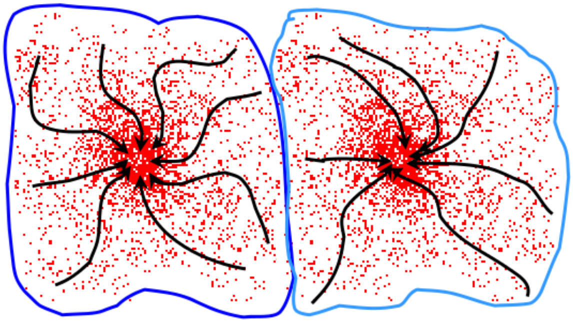


Figure: Constructing basins of attraction

Mean Shift Segmentation (4)

Advantages:

- Does not assume any predefined shape on data clusters
- Depends on a single parameter only (window size)
- Finds variable number of modes (number of clusters does not need to be known in advance)
- Robust to outliers

Disadvantages:

- Output depends on window size
  - The selection of a window size is not trivial
  - Inappropriate window size can cause modes to be merged, or generate additional “shallow” modes
  - Often requires using adaptive window size
- Computationally expensive: $\mathcal{O}(T^2 n)$ with n is the number of data points and T is the number of iterations
- Does not scale well with the dimension of the feature space

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

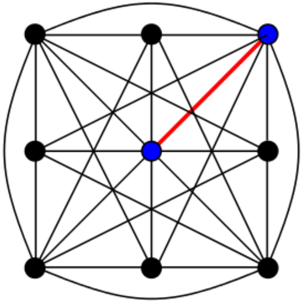
Graph Cut Segmentation (1)

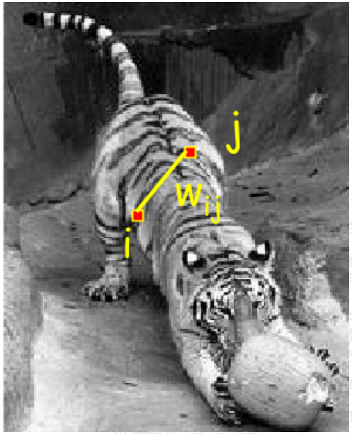
Graph Cut Segmentation treats image segmentation as a graph partitioning problem. Therefore the images needs to converted into a graph.

How can we turn an image into a graph?

- Assign a vertex to each pixel value
- Add add eges between pairs of pixels to describe the relationship between pixels
- Add edge weights to describe the similarity of pixels (a.k.a. *affinity* of vertices). For example: Distance between representation vectors

Graph Cut Segmentation (2)





Graph Cut Segmentation (3)

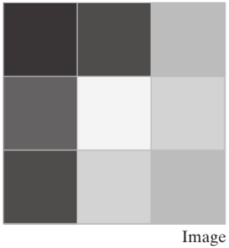
We explain how graph-based segmentation works based on a simple example.

Let's define an image as an **undirected** graph $G = (V, E)$ where

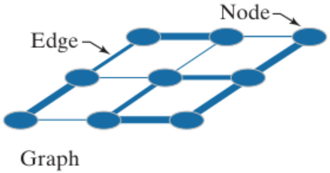
- Each pixel in the image is a vertex V in the graph
- Each pixel (vertex) is connected to its neighbouring pixels (vertices) by an edge E (for simplicity, we assume 4-connectivity)
- Weights are formed according to spatial relationships.

  For illustrative purposes, we assume $w(i, j) = 1/(|I(n_i) - I(n_j)|) + c$ where $I(n_i)$ and $I(n_j)$, are the intensities of the two nodes (pixels) and c is a constant included to prevent division by 0.

Graph Cut Segmentation (4)



(a) A 3x3 image

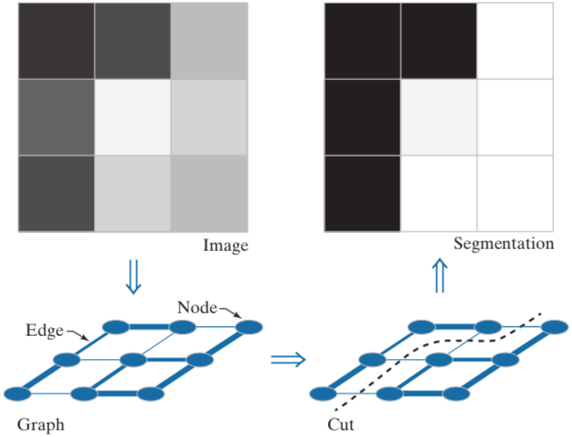


(b) The corresponding graph

Figure: Illustrative example that shows the corresponding graph for a 3x3 image. The thickness of each edge is shown proportional to the degree of similarity between the pixels (see previous slide). Hence, edges between the dark pixels are stronger than the edges between dark and light pixels, and vice versa.

Graph Cut Segmentation (5)

Once an image has been expressed as a graph, the next step is to cut the graph into two or more subgraphs. More precisely, we are looking for the **minimum graph cut**. A partition of the graph into two disjoint subsets that is minimal in some metric.



Graph Cut Segmentation (6)

Let's denote A as the set of vertices that belong to the foreground region and B the background region.

This implies that $A \cap B = \emptyset$ and $A \cup B = V$.

We can think of the optimal "cut" as the one with the smallest total weight of edges between foreground and background nodes where the cut value is calculated as follows:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

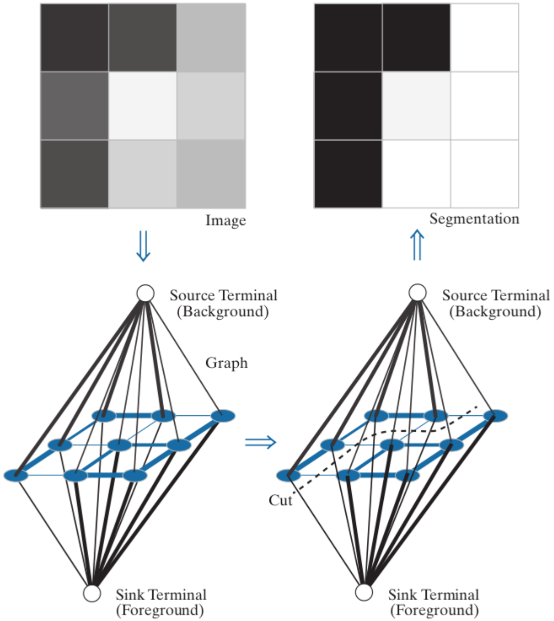
Graph Cut Segmentation (7)

Luckily, graph theory already provides with an algorithm for solving the optimization problem. This algorithm is based on the so-called **Max-Flow, Min-Cut Theorem**.

— *In a flow network, the maximum amount of flow passing from the source to the sink is equal to the minimum cut*

$\rightarrow$ We have to add two more nodes (source and sink) to turn our network into a flow network. The resulting network becomes a **directed graph**.

Graph Cut Segmentation (8)



Graph Cut Segmentation (9)

Note that ...

- Source and sink nodes are called *terminal nodes*.
- Image pixel nodes are called *non-terminal nodes*.
- Terminal nodes are not part of the image. Their role, is to associate with each pixel a probability that it is a background or foreground pixel.

How to choose the terminal weights?

- Typically the user first marks some pixels which are known to belong to the foreground/background
- Terminal weights connected to these pixels receive a probability/weight of 1. or 0., respectively (depending on the region assignment)
- The remaining terminal weights can be computed, for instance, by fitting a Gaussian distribution to the marked pixels and computing the parameters μ, σ. (e.g. using minimum least squares estimation from the pixel's intensities). Then the probabilites for the remaining pixels can be computed from the fitted pdf.

Graph Cut Segmentation (10)

- Various algorithms exists for finding the maximum/minimum flow cut. (e.g., Edmonds-Karp Algorithm)
- Note that, if someone is not satisfied with the resulting segmentation, results can simply be improved by manually marking additional foreground/background pixels (which sets the corresponding terminal probabilities/weights to 1./0.)

Graph Cut Segmentation (11)

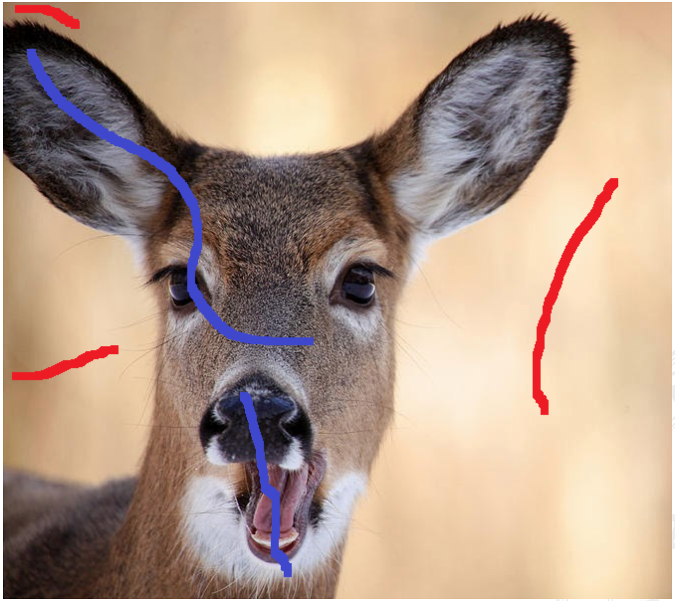


Figure: Example of the marked Input Image

Graph Cut Segmentation (12)

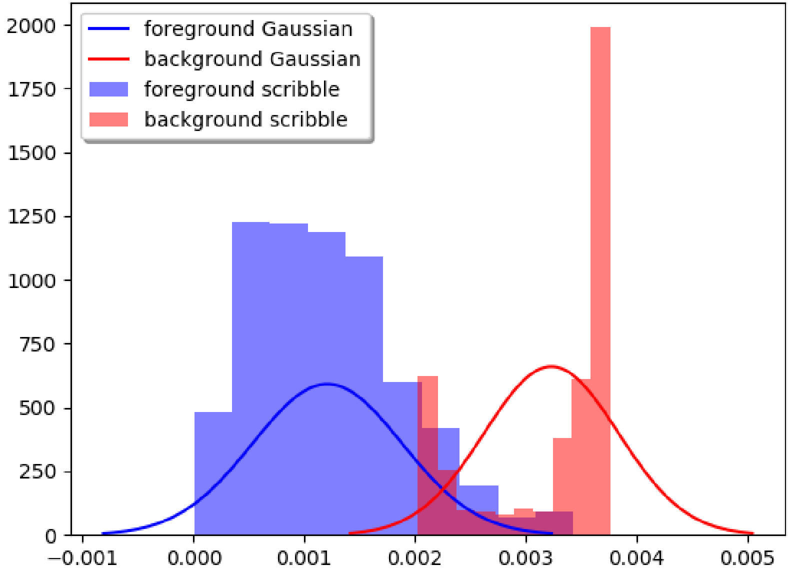


Figure: Example of the estimated probability densities

Graph Cut Segmentation (13)

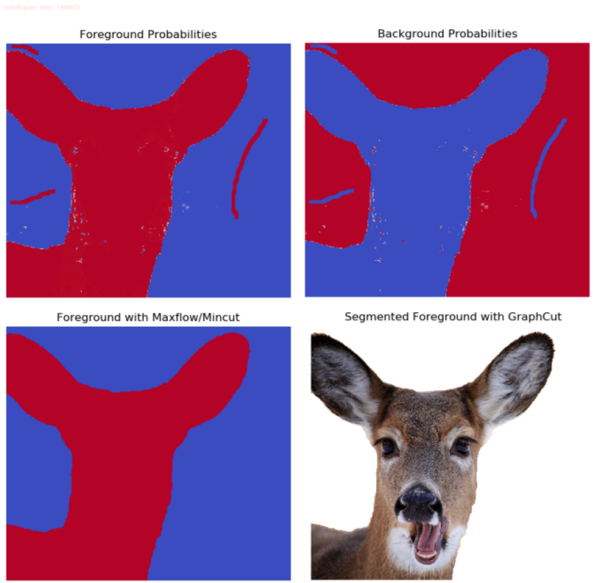


Figure: Example Background/Foreground Probabilities + Result

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Template Matching / Watershed Segmentation

Template Matching

- Known objects are identified by computing the difference to given *templates*
- A template needs to be generated representing the object of interest as well and as general as possible

Watershed Segmentation:

- Uses techniques from morphological image processing
- Will be discussed in the next lecture!

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Edge-based techniques

- Edge-based segmentation techniques are the oldest ones

- Relies on edges found in an image by edge detecting operators — these edges mark image locations of discontinuities in gray level, color, texture, etc.
- For this purpose, edge pixels need to be connected to form *edge chains*, which correspond to region borders

Problems to addressed:

- Wrong edges which are caused by noise where there is no border
- Missing edges where a real border exists

Thresholding of edge images

- Almost no zero-value pixels are present in an edge image, but small edge values correspond to non-significant gray level changes resulting from quantization noise, small lighting irregularities, etc.
- These values can be excluded by a simple thresholding procedure
- Furthermore, non-maximal suppression and hysteresis thresholding can be applied (Canny edge detector)

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Edge Relaxation (1)

- Edge thresholding is often impacted by noise which results in missing edge-chain parts
- Edge relaxation assesses edge property in the context of neighbouring pixels (cf. thresholding with hysteresis)

Basic idea:

- A "weak" edge positioned between two strong edges most likely should be part of the resulting boundary
- A "strong" edge positioned between two "weak" edges most likely should not be part of the resulting boundary.

Important: The algorithm considers so-called **crack edges**. A crack edge simply is defined by the gradient "between" two neighbouring pixels. In a 4-neighbourhood, each pixel has four crack edges.

Edge Relaxation (2)

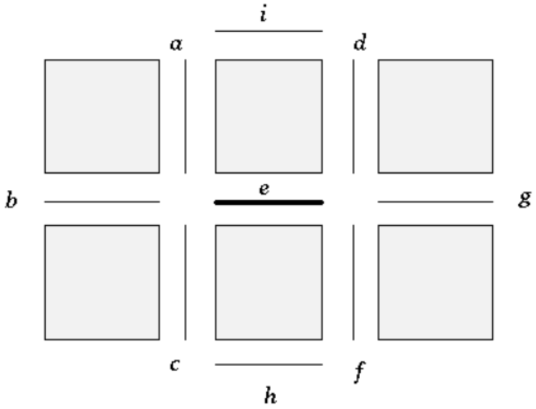
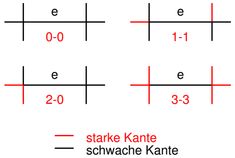


Figure: Crack edges surrounding the centeral edge e

Edge Relaxation (3)

- The central edge *e* has a crossing at each of its ends and three possible border continuations from each of these crossings
- The number of "leaving" edges defines the type of the crossing.
- The edge type *e* can be represented using a number pair $i - j$ describing edge patterns at each crossing, where *i* and *j* are the type of the crossings.



0-0	isolated edge	-	0-2	dead end	-
0-3	dead end	-			
0-1	neutral	0	2-2	bridge	0
2-3	bridge	0	3-3	bridge	0
1-2	continuation	+	1-3	continuation	+
1-1	continuation	++			

Figure: Edge type for different neighbourhoods

Edge Relaxation (4)

The **Edge relaxation algorithm** is an iterative method, with each edge confidence $c(e)$ converging either to zero (edge termination) or one (the edge forms a border).

Edge Relaxation Algorithm:

1. Set the initial edge confidence $c^1(e)$ to be the normalised magnitude of each crack edge (e.g., divide each gradient magnitude by the largest magnitude)
2. Determine the edge type in the neighbourhood and the crossing types (see next slide)
3. Update $c^{k+1}(e)$ for each edge corresponding to its type and previous confidence $c^k(e)$
4. Stopping criterion (e.g. in case of convergence to 0 or 1)

Edge Relaxation (5)

Assessment of crossing types:

Crossing is of type i, if $type(i) = \max_k((type(k)), k = 0, 1, 2, 3$

$$type(0) = (m - a)(m - b)(m - c) \qquad type(1) = a(m - b)(m - c)$$
$$type(2) = ab(m - c) \qquad type(3) = abc$$

a, b, c	...	normalised values of neighbouring edges
m	...	$m = \max(a, b, c, q)$
q	...	constant; $q \sim 0.1$ (ensures that $type(0)$ is non-zero for small values of a)

Example: $(a, b, c) = (0.5, 0.05, 0.05)$ is a type 1 crossing, $(0.3, 0.2, 0.2)$ is a type 3 one.

Similar results are obtained by counting the number of edges at a crossing above a threshold

Edge Relaxation (6)

Once we computed the vertex types, it's easy to determine the edge type for any edge e based on the type of two neighbouring vertices.

Finally, we update the edge confidence with the following rule:

Update Step:

Increasing edge property : $c^{k+1}(e) = min(1, c^k(e) + \delta)$

Decreasing edge property : $c^{k+1}(e) = max(0, c^k(e) - \delta)$

Increment if e has the edge type 1-1, 1-2, 1-3 and decrement if 0-0, 0-2, 0-3.

- δ is typically selected from 0.1 - 0.3 (for strong and weak modification)

Edge Relaxation (7)

- Edge relaxation rapidly improves the initial edge labeling in the first few iterations.
- Unfortunately, it often provides worse results than expected after larger numbers of iterations.
- The problem can be mitigated by setting the edge confidences to zero under a certain threshold, and to one over another threshold.

Improved Update Step:

$$c^{k+1}(e) = \begin{cases} 1 & c^k(e) > T_1 \\ 0 & c^k(e) \leq T_2 \end{cases}$$

Edge Relaxation (8)

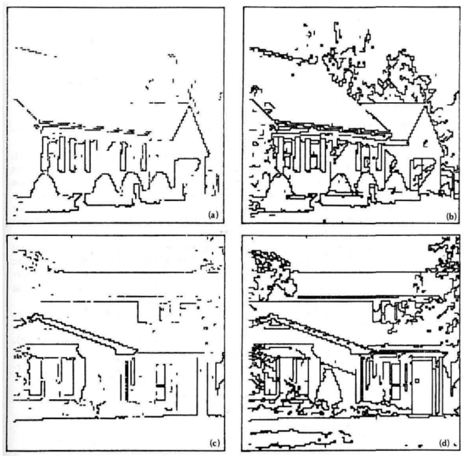


Figure: Example of edge relaxation. (a) Raw edge data. Edge strength have been thresholded at 0.25 for display purposes. (b) Results after 5 iterations of edge relaxation. (c) Different version of (a). Edge strength have been thresholded at 0.25 for display purposes. (d) Results after 5 iterations of edge relaxation.

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Completing Edge Chains using Graph Search (1)

Initial situation: We are given a gradient magnitude image and the corresponding gradient direction.

Goal: Find edge chains using graph search.

Basic idea: Edge search is transformed into search for an optimal path in a weighted graph (graph problem). We then look for the best path (minimum cost path) connecting begin and end of an edge chain.

Completing Edge Chains using Graph Search (2)

But how to construct a graph from the gradients / orientations?

Definition:

- Edge magnitude (gradient magnitude) ... $s(x)$
- Edge orientation (gradient direction) ... $\phi(x)$

where x is an edge

To construct the graph, each (edge-)pixel becomes to a node in the graph, weighted with $s(x)$.

$\rightarrow$ But how should we connect the nodes?

Completing Edge Chains using Graph Search (3)

Two neighbouring edges x_i and x_j can be connected by a path in the graph, if $\phi(x_i)$ and $\phi(x_j)$ "fit together". Intuitively, two neighbouring edges should point in similar directions if they belong to the same edge chain.

Example of "connection criterion":

Connect x_i with x_j if and only if the following conditions are met ...

- $(\phi(x_i) - \phi(x_j)) \ mod \ 2\pi < \pi/2$
- $s(x_i) > T$
- $s(x_j) > T$

where T is some threshold.

Completing Edge Chains using Graph Search (4)

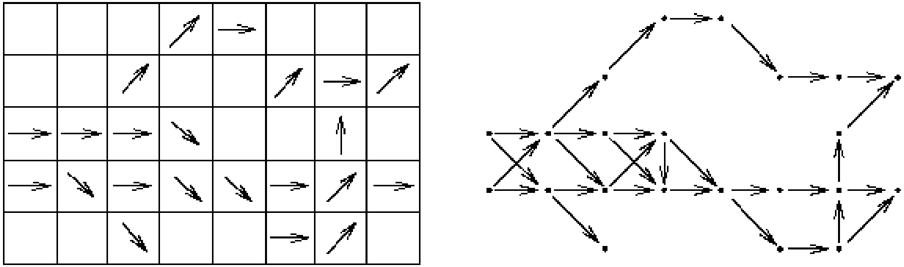


Figure: Graph representation of an edge image

Completing Edge Chains using Graph Search (5)

To generate a path in a graph from x_A to x_B one can apply the well-known technique of **heuristic search (A algorithm)**.

Suppose that ...

- x_A and x_B is the begin- and end-point of the edge chain.
- we have a method ("expension technique") for generating the successor nodes of a given node.
- we have an cost function $f(x_i)$ which estimates the cost for the path between x_A and x_B passing through x_i.
- the **cost function** has to be **monotonic increasing** with respect to path length (non-negative costs only)

Completing Edge Chains using Graph Search (6)

W can think of the cost function $f(x)$ to have two additive components: $g(x_i)$ and $h(x_i)$

$$f(x_i) = g(x_i) + h(x_i)$$

$g(x_i)$ ...	Costs from x_A to x_i
	Sum of the costs of the nodes in the path connecting x_A and x_i
$h(x_i)$ ...	**Estimated** costs from x_i to x_B

Note: We denote the **true costs** from x_i and x_B as $h^*(x)$. These costs are not known in practice and estimated by $h(x_i)$.

Completing Edge Chains using Graph Search (7)

Nielson's A-Algorithm:

1. Expand x_A (put all successors into an OPEN-List with pointers back to x_A); compute costs for all nodes.
2. If OPEN-List is empty, the algorithm failed.
   Otherwise: determine x_i in the list with the lowest costs $f(x_i)$ and remove this node. If $x_i = x_B$ follow the pointers to identify the best path and stop.
3. No "stop" occured in 2. Expand x_i and put the successors into the OPEN-List with pointers back to x_i; compute their costs; go to 2.

Note that ...

- It is important to include a strategy against the occurrence of loops in the algorithm

Completing Edge Chains using Graph Search (8)

The estimation of $h(x_i)$ of $h^*(x_i)$ has significant influence to the behaviour of the search process. The search can be accelerated - less precise - or search can degenerate to full search dependent on how $h(x_i)$ is estimated.

Variants:

$h(x_i) = 0$	: No heuristics is included and the search degenerates into a breadth-first search; heuristic methods do not guarantee to find the optimal result but they are faster.
$h(x_i) > h^*(x_i)$	: The algorithm is fast, but the minimal cost result cannot be guaranteed
$h(x_i) = h^*(x_i)$	: The search is able to identify the path with lowest costs while using a minimal number of expanded nodes; In general, we find that the number of expandend nodes is the smaller, the closer $h(x_i)$ is to $h^*(x_i)$.
$h(x_i) \leq h^*(x_i)$	: The search identifies the path with lowest costs, however, for each part of the path we find that actual costs are larger than estimated costs.

Completing Edge Chains using Graph Search (9)

Applicable cost functions:

- Edge Magnitude: high magnitude $\rightarrow$ reliable edge $\rightarrow$ small costs (e.g. direct costs: difference to largest edge magnitude in the image).
- Curvature: Difference of edge orientations
- Distance to end-point of edge chain
- Distance to known or estimated or conjectured position of the edge chain

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Completing Edge Chains using Dynamic Programming (1)

Dynamic Programming

Background: Bellmann's principle of optimality – independent of the path leading to node E, there is an optimal path connecting E and the end point.

In other words: If the optimal path connecting begin- and end-point passes through E, also the partial paths from begin-point $\rightarrow$ E and E $\rightarrow$ end-point have to be optimal.

Algorithm:

1. Construct the graph and the rating (based on weights) of all partial paths between two layers.
2. In each layer, determine for each node E the lowest-cost partial path connecting the preceding layer to E.
3. Determine the lowest-cost node in the final layer.
4. Backtracking along the identified optimal partial paths identifies the overall optimal path.

Completing Edge Chains using Dynamic Programming (2)

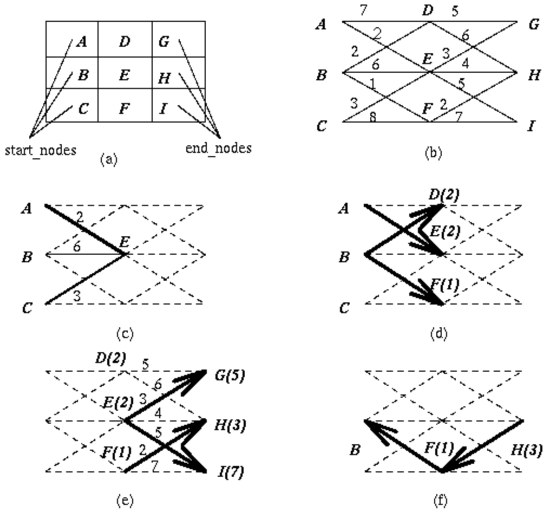


Figure: Example for Dynamic Programming: (a) Edge image (b) Graph with costs (c) admissible paths E, A-E is optimal (d) optimal paths to D,E,F (e) optimal paths to G,H,I (f) Backtracking from H determines lowest-cost path

Graph Search vs. Dynamic Programming

- It has been shown that for finding a path between two points, heuristic search can be more efficient than dynamic programming
- Dynamic programming build paths from **multiple** starting points. If this is required by a particular task, then dynamic programming is probably the better choice.

Outline

1 Introduction

2 Thresholding-based segmentation
- Thresholding Variations
- Threshold Selection
- Optimal Thresholding Techniques

3 Region-based segmentation
- Region Growing
- Region Merging and Splitting
- K-Means Clustering Segmentation
- Mean Shift Segmentation
- Graph Cut Segmentation
- Template Matching and Watershed Segmentation

4 Edge-based segmentation
- Edge relaxation
- Completing Edge Chains using Graph Search
- Completing Edge Chains using Dynamic Programming
- Completing Edge Chains using Active Contours (Snakes)

Completing Edge Chains using Active Contours (Snakes) (1)

Iteratively adapt a closed curve to image content by optimising energy functionals. The curve is modelled by an energy-minimizing spline also called **snake**.

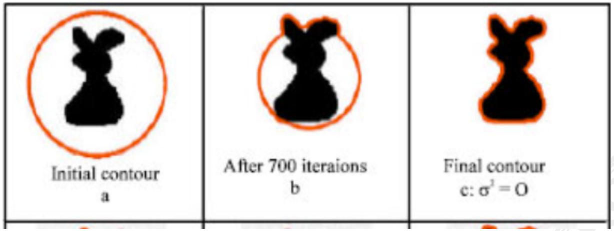


Figure: Example of fitting an active contour

Completing Edge Chains using Active Contours (Snakes) (2)

A spline simply is a function defined piecewise by polynomials.
Our goal is to adopt the parameters of the spline in a way that it fits the borders of the object.

To accomplish this deformation of the spline v is described by an energy function $E(v)$ which consists of two parts:

- External term $E_{ext}(v)$ — Attracts the contour toward the closest image edge, gradient information is used)
- Internal terms $E_{in}(v)$ — Forces the contour to be continuous and smooth)

Important questions:

- How to select the initial curve ?
- How to weight the different terms in the functional?
- More details in the "Medical Imaging" lecture