Image Processing and Imaging
Image Segmentation

Christof Kauba
Fachbereich Computerwissenschaften
Universität Salzburg

Wintersemester 2020/21

UNIVERSITÄT SALZBURG

PARIS LODRON UNIVERSITY SALZBURG

Outline

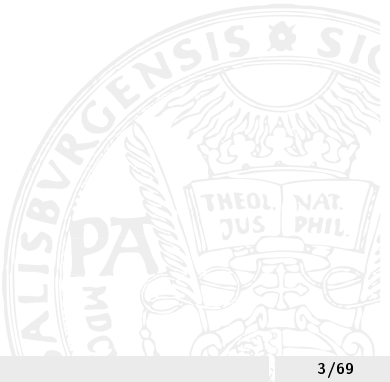


Image Segmentation Overview (1)

- One of the most important stages in the image analysis process
- Main aim is to identify parts of the image exhibiting a high correlation with real world objects or areas
- Key to success is to grasp (at least parts of) the semantics of the image content

Complete Segmentation:

- A set of regions / objects with empty intersection which uniquely correspond to real objects
- In many cases expert knowledge applied in artificial intelligence methods is required
- Some applications which are simple enough to generate good results with simpler approaches:
  - Text with letters and numbers
  - Objects in front of uniform background and high contrast in general

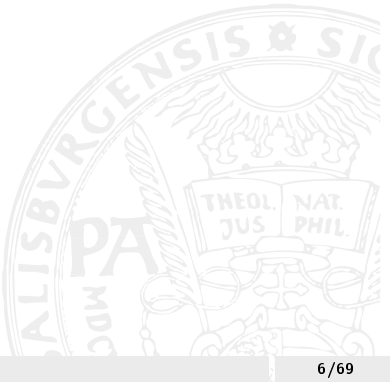
Partial Segmentation:

- Regions which are homogeneous with respect to some criterion are identified
- They do not necessarily correspond to objects of the scene
- In many cases, over-segmentation occurs, i.e. to many regions are found

- Segmentation techniques can be grouped into three classes:
  1. Techniques which use global information about the image or parts of it (e.g. the histogram)
  2. Edge-based techniques try to generate closed edge-chains as object borders
  3. Region-based techniques try to generate homogeneous regions

  - 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



Thresholding

- Thresholding is the simplest segmentation technique
- Due to its high speed it is still of importance
- An intensity constant or threshold is determined to separate objects and background
- Input image is transformed into a binary segmented output image as follows:

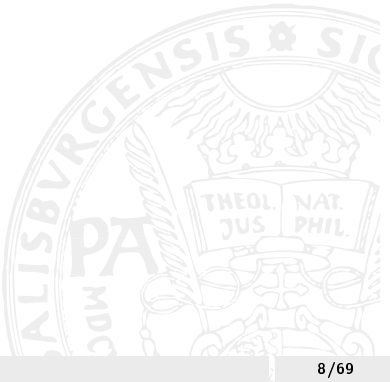
$$g(i,j) = \begin{cases} 1 & f(i,j) \geq T \\ 0 & f(i,j) < T \end{cases}$$

- Thresholding is a suited approach in case objects are not joined and object gray-scale is different from background gray-scale

Selection of the Threshold:

- Is a central question
- Using a fixed threshold is successful in rare cases only
- Variable thresholding is often more successful ($T = T(f, f_c)$ with f_c for the image part considered); varies the threshold value as a function of changing image part characteristic

Outline



Band Thresholding:

- Gray-scales in a specific gray-scale range / band are determined to be object pixels (as opposed to background or non-interest pixels)
- For example, segmentation of cells or cell parts in microscopic images, known gray-scales caused by known material density in CT-images

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

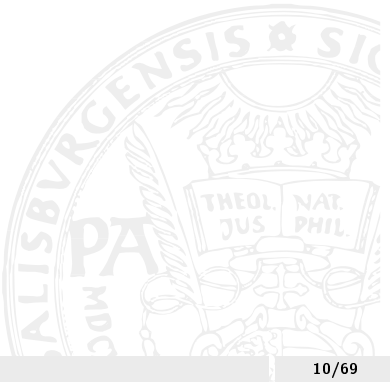
Multi Thresholding: employs several thresholds and the output image is not binary

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

Semi Thresholding: aim is to remove background but to keep gray-scale 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



Text/Letters Example:

- In a text, it is known that letters cover a certain share of the text image pixels ($1/p$ of the image area)
- Straightforward to select the threshold by analysing the histogram (cumulative distribution function), such that $1/p$ of the image is $\leq T$ (is denoted "p-tile thresholding")
- However, in most cases we do not have such information

Threshold selection is done by analysing the histogram:

- Objects with uniform gray-scale which is different from the (uniform) gray-scale of the background:
  - Histogram is *bimodal*
  - Threshold is selected to be the minimal value between the two extrema
- In case of *multimodal* histograms:
  - Thresholds need / can be selected between two corresponding maxima
  - Either, different segmentation results are obtained
  - Or multithresholding has to be applied

Threshold Selection Problem

- How can we decide if a histogram is bimodal or multimodal?
- Usually, bimodal techniques take the largest two maxima and set T at a (the) minimum in between ("mode method")
- In order to avoid that two close local maxima are selected, a minimal distance in terms of gray-scales should be enforced
- Alternatively, the histogram can be smoothed
- Attention: A bimodal histogramm does not guarantee a correct segmentation
- E.g. 50% B/W pixel mixed or concentrated on one image side result in an identical bimodal histogram

Further Strategies to Select Thresholds - Local Neighbourhood and Optimal

Local Neighbourhoods:

- Consideration of local neighbourhoods when generating the histogram
- E.g. by assigning weights to pixels in order to suppress pixels with large gradient
- In this case the histogram does not contain pixels of the border curve but only object and background pixels

Optimal Thresholding:

- Histogram is approximated by a weighted sum of probability densities
- Threshold is selected to be the minimal probability between the maxima of the distributions
- **Problem**: Estimation of the parameters of the distributions and identification of the type of the distribution (normal distribution is often assumed)

Concept of Optimal Thresholding

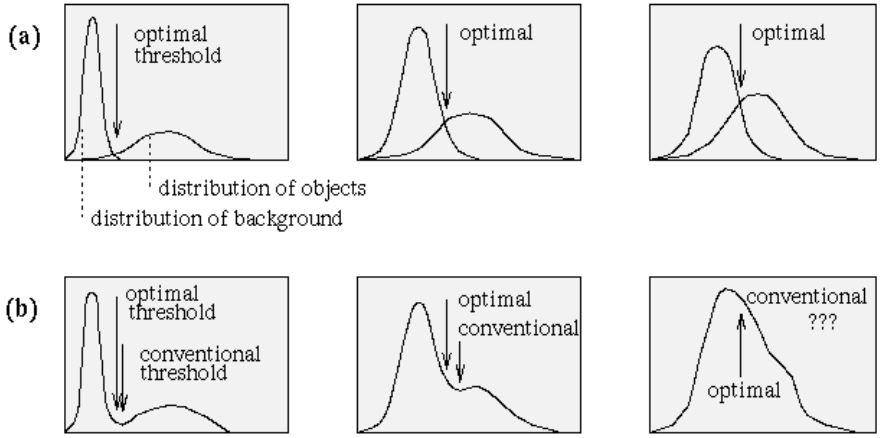


Figure: Concept of optimal thresholding, first row represents the individual distribution, second row is the combined distribution.

Example of Optimal Thresholding - Histogram

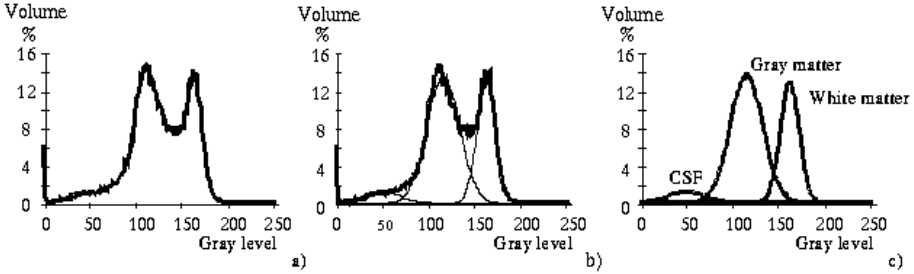
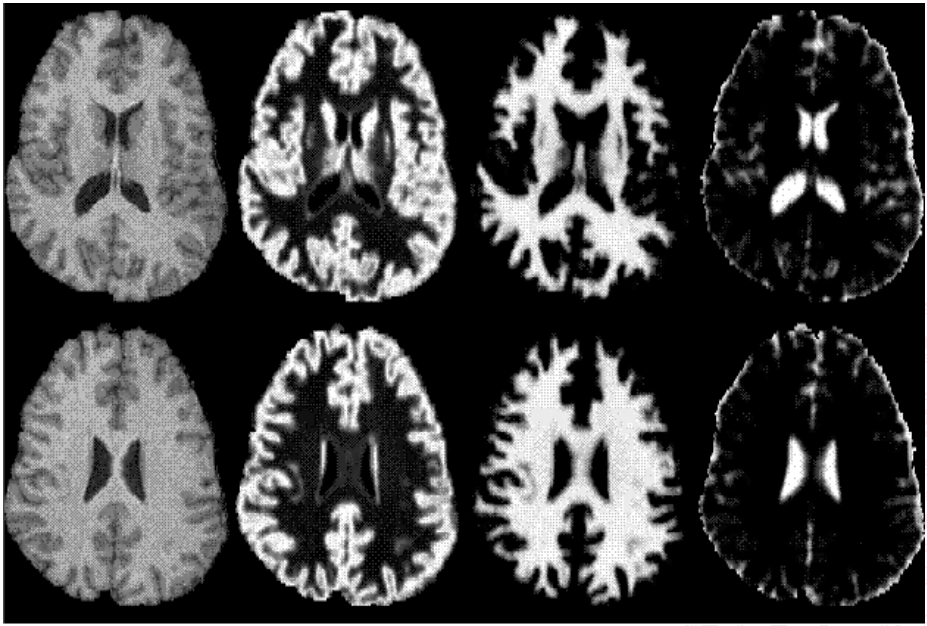


Figure: Example for optimal thresholding: Histogram

Example of Optimal Thresholding - Result



Iterative Threshold Selection:

- Assume the existence of regions with two dominant gray-scales
- Corners of the image contain background pixels

- In iteration t we compute the averages μ_B^t (background) and μ_O^t (object)
- Segmentation in step t is defined to be (background and foreground regions change depending on the threshold T):

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

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

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

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

- If $T^{t+1} = T^t$, stop.

Otsu Thresholding:

- Assumes a bimodal histogram and selects threshold by histogram analysis
- Once the histogram with probability $P(i)$ in bin i has been established, it is fast
- Basic idea is to find the threshold that minimises the weighted intra-class variance:

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

where $q_O(t) = \sum_{i=1}^{t} P(i)$ and $q_B(t) = \sum_{i=t+1}^{I} P(i)$ and the class means defined as $\mu_O(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_O(t)}$ and for background by analogy.

- Finally, the individual class variances are given as:

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

(background variance by analogy)

- Now, we could actually stop here
- All we need to do is just run through the full range of t values [1,256] and pick the value that minimizes σ^2_{intra}

Further Strategies to Select Thresholds - Otsu (2)

- Relationship between the intra-class and inter-class variances can be exploited to generate a recursion relation that permits a much faster calculation:
- For any given threshold, the total variance is the sum of the weighted intra-class variances and the inter-class variance (which is the sum of weighted squared distances between the class means and the overall mean)

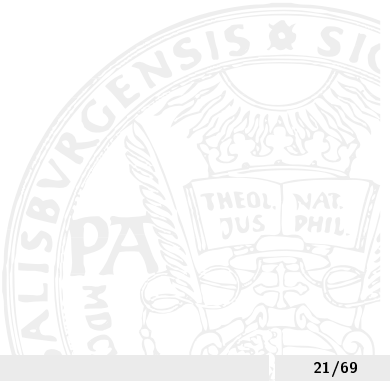
$$\sigma^2 = \sigma^2_{intra}(t) + \sigma^2_{inter}(t) = \sigma^2_{intra}(t) + q_O(t)(1 - q_O(t))(\mu_O(t) - \mu_B(t))2$$

- Since the overall variance does not depend on a threshold, minimizing the intra-class variance is the same as maximizing inter-class variance
- $\sigma^2_{inter}(t)$ can be computed more efficiently as compared to $\sigma^2_{intra}(t)$ by a recursion when running through the range of t values:
  - Initialisation: $q_O(1) = P(1)$, $\mu_O = 0$
  - Recursion:
    1. $q_O(t+1) = q_O(t) + P(t+1)$
    2. $\mu_O(t+1) = \frac{q_O(t)\mu_O(t) + (t+1)P(t+1)}{q_O(t+1)}$
    3. $\mu_B(t+1) = \frac{\mu - q_O(t+1)\mu_O(t+1)}{1 - q_O(t+1)}$

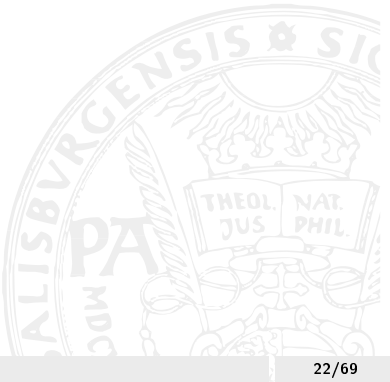
Thresholding in Hierarchical Data Structures:

- Uses a pyramidal data structure
- Regions are identified in the low resolution image and are further refined in the higher resolutions
- Two variants are possible:
  1. Segmentation in the low-resolution image using thresholding; in the next higher resolution pixels close to the region border are re-assigned to object(s) or background (depending on a similarity criterion), this is done successively until the highest resolution.
  2. A *significant pixel detector* is applied in the lowest resolution image and determines pixels different to their neighbourhood; usually done with 3 x 3 masks which indicate a central pixel being different from its neighbourhood; such pixels correspond to regions in full resolution; corresponding image part in full resolution is thresholded using 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).
- Advantage of hierarchical techniques is their higher robustness against noise, since the initial segmentations start with a significantly smoothed image.

Outline



Outline



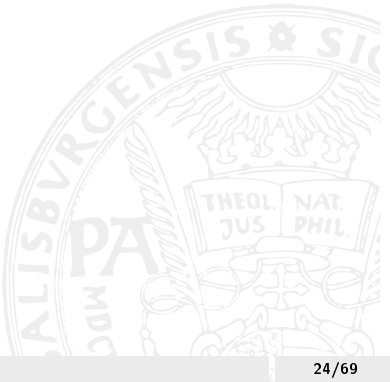
Edge-Based Techniques

- Edge-based segmentation techniques are the oldest ones
- Edge detection does not result in a segmented image
- For this purpose, edge pixels need to be connected to form *edge chains*, which correspond to region borders

Thresholding of Edge Images:

- Small edge values correspond to non-significant gray-scale changes
- These values can be excluded by a simple thresholding procedure
- Further processing can be done by further excluding isolated edge pixels or edge-chains with a length below a certain length threshold

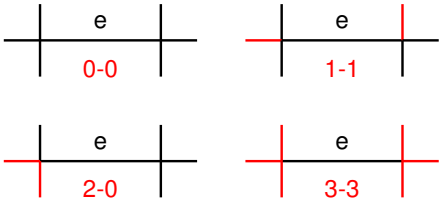
Outline



Edge-Based Techniques - Edge Relaxation (1)

- Edge thresholding is often impacted by noise which results in missing edge-chain parts to form complete region borders
- Edge relaxation assesses edge property in the context of neighbouring pixels (cf. thresholding with hysteresis)
- Under consideration of edge magnitude and edge continuation an iterative process increases or decreases the edge property
- For this algorithm, edges are considered as *crack edges*

- Edge property is investigated at both ends of an edge in all three possible edge continuation directions
- Edge e has a crossing at each side and there are three possible continuation directions
- Each crossing is assessed according to its number of "leaving" edges and the form of the crossing
- Initial edge property $c^1(e)$ is the normalised magnitude of the crack edge
- Edge property converges from iteration to iteration to either 0 or 1



starke Kante
schwache Kante

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 properties

Algorithm (2. and 3. are iterated):

1. Evaluate edge property $c^1(e)$ for all crack edges in the image
2. Determine edge types in the neighbourhood and determine crossing types
3. Update $c^{k+1}(e)$ for for each edge corresponding to its type and $c^k(e)$
4. Stopping criterion (e.g. in case of convergence to 0 or 1)

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; $q \sim 0.1$

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

Update Step:

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

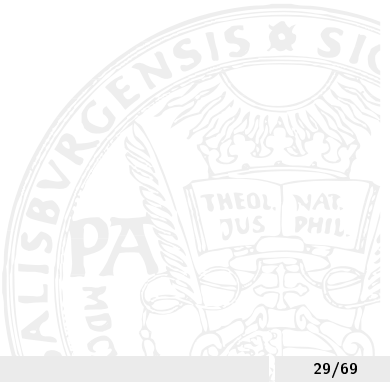
Decreasing edge property : $c^{k+1}(e) = max(0, c^k(e) - \delta)$

- δ is typically selected from 0.1 - 0.3 (for strong and weak modification)
- Simpler if only one value is used
- In the version discussed so far, often only slow convergence is achieved

Improved Update Step:

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

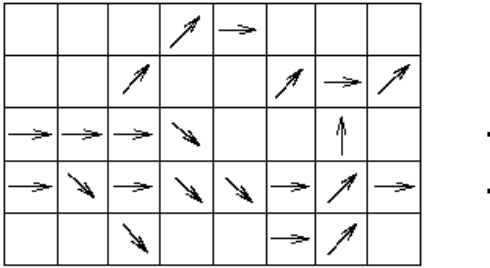
Outline



Edge-Based Techniques - Completing Edge Chains using Graph Search (1)

Initial situation: Begin and end of an edge chain
Graph: set of edges x_i and paths $[x_i, x_j]$ connecting these edges

- We consider oriented and weighted paths and denote the weights as "costs'
- Edge search is transformed into search for an optimal path in a weighted graph (graph problem)
- Search for the best path connecting begin and end of an edge chain
- Assume to have information about edge magnitude (gradient magnitude) $s(x)$ and edge orientation (gradient direction) $\phi(x)$
- Each (edge-)pixel corresponds to a node in the graph, weighted with $s(x)$
- Two edges x_i and x_j can be connected by a path, if $\phi(x_i)$ and $\phi(x_j)$ fit together:
  - x_i has to be one of the three existing neighbours of x_j in the direction $d \in [\phi(x_i) - \pi/4, \phi(x_j) + \pi/4]$ and $s(x_i)$ und $s(x_j) \geq T$



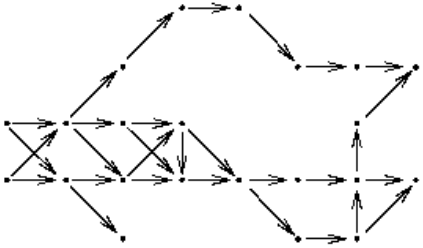


Figure: Graph representation of an edge image

Based on this set-up, classical graph search techniques can be applied:

- Let x_A and x_B be begin and end-point of the edge chain
- Define an expansion technique plus a cost function $f(x_i)$, which allows to conduct an estimation of the costs of a path between x_A and x_B passing through x_i
- Cost function has to be monotonic with respect to path length
- It has to be possible to partition the computation of the function among several path parts:

$g(x_i)$ costs from x_A to x_i
sum of the costs of the nodes in the path connecting x_A and x_i
$h(x_i)$ costs from x_i to x_B (estimation)

Nielson's A-Algorithm (heuristic graph search)

1. expand x_A and 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.

- It is important to include a strategy against the occurrence of loops in the algorithm
- Estimation $\hat{h}(x_i)$ of $h(x_i)$ has significant influence to the behaviour of the search process:
- Search can be accelerated - less precise - or search can degenerate to full search

Variants:

$\hat{h}(x_i) = 0$: 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.

$\hat{h}(x_i) > h(x_i)$: The algorithm is fast, but the minimal cost result cannot be guaranteed

$\hat{h}(x_i) = h(x_i)$: 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 $\hat{h}(x_i)$ is to $h(x_i)$.

$\hat{h}(x_i) \leq h(x_i)$: 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.

Branch and Bound Algorithms : maximal admissible costs are defined and more expensive paths are no longer considered in the search.

Multiresolution Processing : Ideas described so far are applied to low-resolution-image data first, and transferred to higher resolution subsequently (usually in a prediction - correction framework).

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

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 point passes through E, also the partial paths begin point $\rightarrow$ E and E $\rightarrow$ end point have to be optimal.

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.

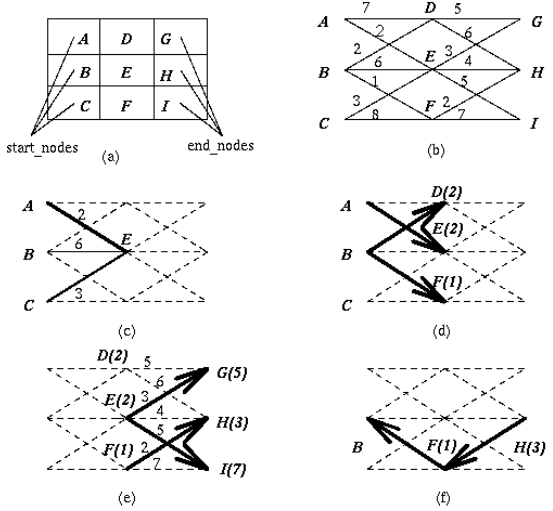
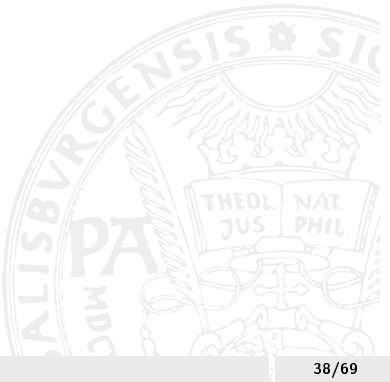


Figure: Example for Dynamic Programming: (a) Edge image (b) Graph with costs (c) admissible paths E, A-E is optimal (d) optimal paths to D,E,F (e) optimal paths to G,H,I (f) Backtracking from H determines lowest-cost path

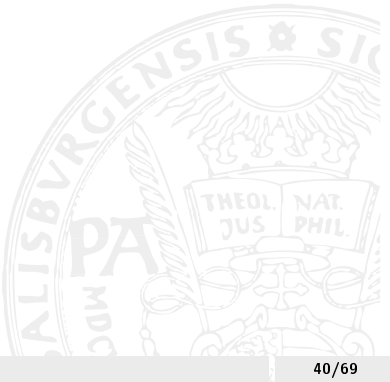
Outline



Active Contours - Snakes / Level Set Segmentation:

- Iteratively adapt a closed curve to image content by optimising energy functionals:
  - External term (attracts the contour toward the closest image edge, obviously gradient information is used)
  - Internal terms (force 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

Outline



Region-Based Techniques

- Primary applied to noisy images since under such conditions edge detection is difficult and results are not reliable
- In circumstances where the major difference among pixels of different objects is not their differing luminance (e.g. different structure - texture), edge-based techniques may fail due to lacking contrast
- Strategy is to partition an image into regions of maximal homogenity:

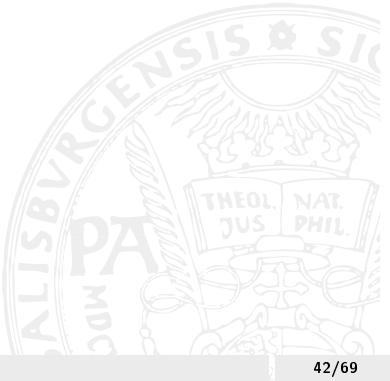
Homogeneity:

- Often defined based on gray-values (e.g. average gray-value, shape of a local histogram) or uses texture properties etc.
- Target regions exhibit the following property (apart from their zero intersection):

$$H(R_i) = \text{True} \qquad i = 1, \ldots, S$$
$$H(R_i \cup R_j) = \text{False} \qquad i \neq j \text{and} R_i \text{ adjacent to } R_j$$

  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

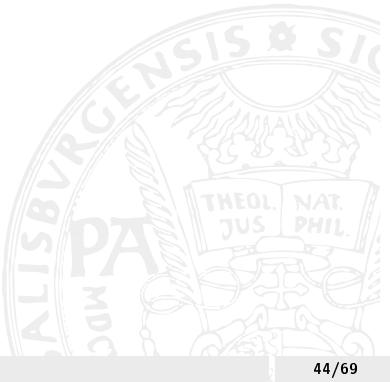


Starting from seed pixels, new (adjacent) candidates are investigated and added to the region, in case homogeneity is maintained:

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
3. Repeat step 2) for the newly added pixels; stop if no more pixels can be added

- Criteria how to define similarity (or homogeneity of the region) include average intensity, variance, colour (histograms), texture, motion, shape, etc.
- Selection of seed points is highly application dependent and can be done similar to marker selection in watershed segmentation
- Simplest choice is to select intensity / colour values corresponding to the highest peaks in the histograms

Outline



Region Merging Algorithm:

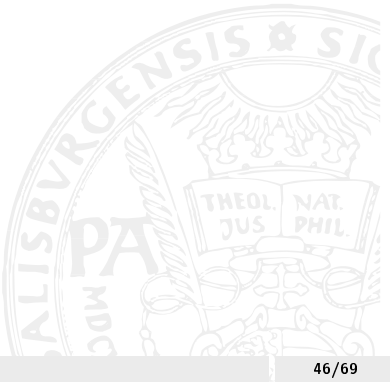
1. Segment image into small regions satisfying the homogeneity requirement
2. Define criteria to merge those regions (*merging*)
3. Apply merging until it is no longer possible

- Different techniques differ in terms of their initial segmentations and the different criteria for homogeneity
- Improvement: 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:

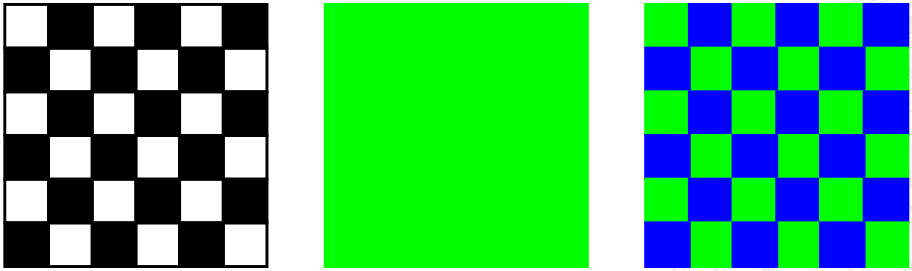
- Contrasting to *Region Merging* we start with the entire image which usually does not satisfy the homogeneity criterion
- Image is split into regions satisfying this constraint

Outline



Remark:

- Merging is not dual to splitting even if the same homogeneity criterion is used
- Figure: Splitting and merging applied to a chess board, where identical average gray-value is used as criterion to rate homogeneity
- Splitting: Criterion is a medium gray for all for image quadrants (so the criterion is fulfilled and no splitting is applied)
- Merging: it is black or white until the size of the chess-board pattern is reached (i.e. we result in black and white regions corresponding to the fields of the board)

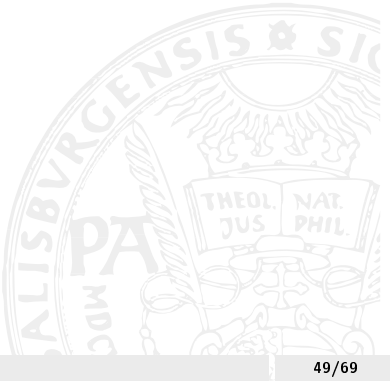


Application:

In many cases a combination of splitting and subsequent merging is applied
Data structure employed is often a quad-tree (*split and merge*):

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

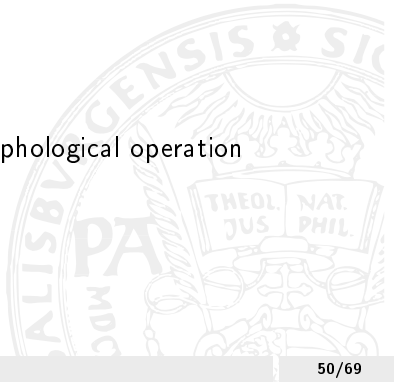


Template Matching:

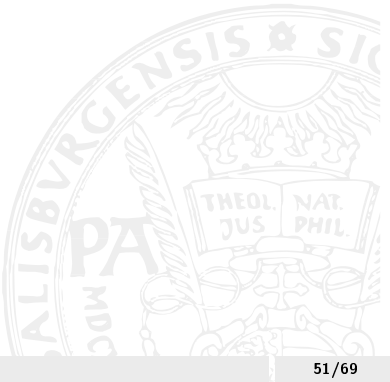
- Known objects are identified by computing the difference to given *templates*
- A template needs to be generated representing the object of interest as well and as general as possible

Watershed Segmentation:

- Uses techniques from morphological image processing
- Discussed in the following section as final application of morphological operation



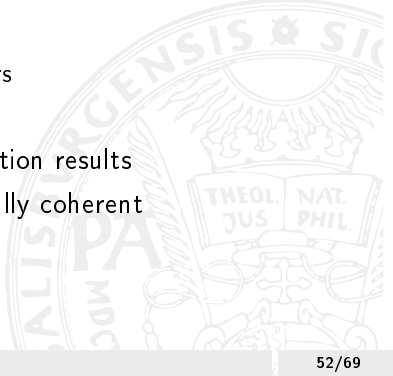
Outline



Region-Based Techniques - K-Means Clustering Segmentation (1)

- Image segmentation can be viewed as a clustering task
- "Similar" pixels are clustered into one region
- This leads to the following 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

- Different representations of pixels lead to different segmentation results
- E.g. if location is not at all employed, clusters are not spatially coherent



K-means Clustering:

- Given a set of pixels $(x_1, \ldots, x_n)$ and represent each pixel by its associated vector
- Aims to partition the n pixels into K sets $(K \leq n)$ $S = \{S_1, S_2, \ldots, S_K\}$ so as to minimize the within-cluster sum of squares (WCSS):

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

with μ_i the mean or centroid of pixels in cluster S_i

- Usually done in two steps:
- **Assignment step**:
  - Each pixel is assigned to the cluster whose mean yields the least WCSS
  - Mathematically/geometrically, this corresponds to a partitioning of the pixels according to the *Voronoi diagram* generated by the means
- **Update step**:
  - New means / centroids in the new clusters are computed

- The optimal solution to the K-means clustering is an NP-hard problem
- In practical applications, heuristic approximations are used
- **Lloyd Algorithm**:
  1. Initialisation: choose k random means $m_1^{(1)} ... m_k^{(1)}$
  2. Assignment step: each data point is assigned to the cluster which variance is inceased least

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

  3. Update step: recompute the cluster means/centres

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

- Lloyd's algorithm has a complexity of $\mathcal{O}(knT)$ with k is the number of clusters, n is the number of data points and T is the number of iterations

Region-Based Techniques - Mean Shift Segmentation (1)

This approach has some severe drawbacks:

- Sensitive to initialisation (how to choose initial μ_i – e.g. random, uniform partitioning – similar to the question how to select seeds in region growing)
- K - the number of image segments has to be chosen a priori
- Sensitive to outliers
- A key limitation of K-means is its cluster model. The concept is based on spherical clusters that are separable in a way so that the mean value converges towards the cluster center

Mean Shift Algorithm:

- A non-parametric clustering technique
- Does not require prior knowledge of the number of clusters
- Does not constrain the shape of the clusters

- Idea is that clusters are places where data points (i.e. pixels) tend to be close together in feature space
- Instead of initialising cluster centers and selecting the number of clusters:
  - Mean shift algorithm seeks *modes* or local maxima of density in the feature space

Actual Mean Shift Procedure:

- Selects a pixel, applies a window in feature space around the pixels' feature
- Subsequently the mean of the pixels' neighbouring feature vectors is computed
- Center of the window is shifted to the position of the mean (centroid)
- Procedure is iterated until the mean does not change its position
- A local maximum – a mode – is found

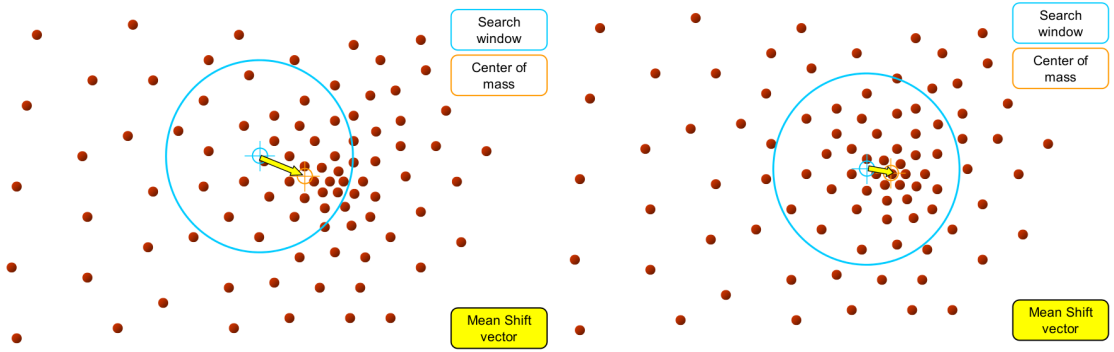


Figure: Mean-shift procedure in feature space

Region-Based Techniques - Mean Shift Segmentation (3)

- 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

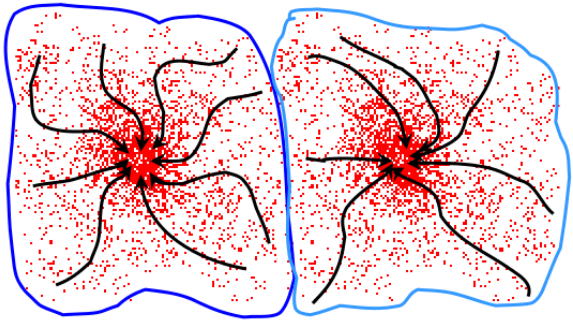


Figure: Constructing basins of attraction

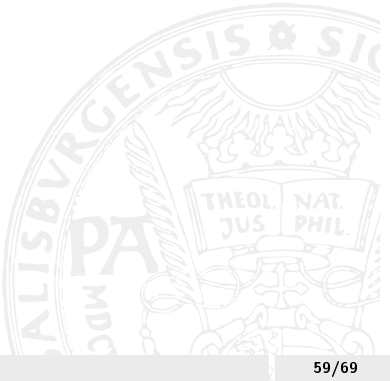
Advantages:

- 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:

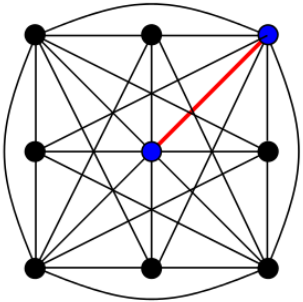
- Output depends on window size
  - The selection of a window size is not trivial
  - Inappropriate window size can cause modes to be merged, or generate additional "shallow" modes
  - Often requires using adaptive window size
- Computationally expensive: $\mathcal{O}(T^2 n)$ with n is the number of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

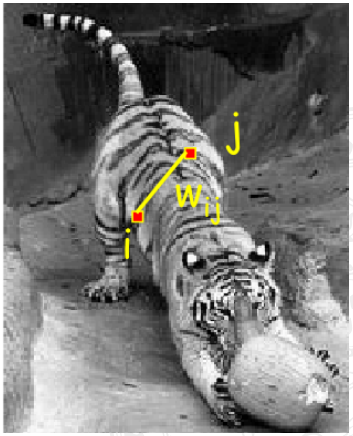
Outline



We represent images as graphs:

- Assigning a vertex to each pixel
- Defining edges between neighbouring pixels
- Weighting edges according to the similarity of pixels (*affinity* of vertices), i.e. distance between representation vectors





In more detail, we define the directed graph $G = (V, E)$ as follows:

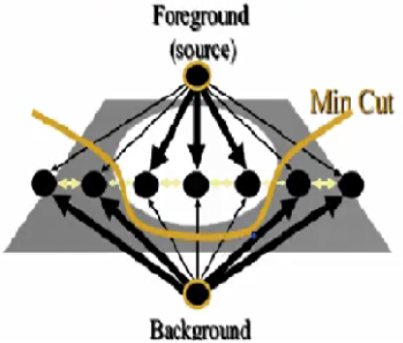
- Each pixel in the image is a vertex V in the graph
- There are special terminal vertices:
  - Source vertex: corresponds to a foreground object
  - Sink vertex: corresponds to a background object, i.e. $|V(G)| = widthxheight + 2$
- There are two types of edges: *terminal* edges and *non-terminal* edges
- There is a directed edge from the terminal node **source** to each non-terminal node in the graph
- Similar, there is a directed edge from each non-terminal node to the other terminal node **sink**
- Each non-terminal node is connected by edges to the nodes corresponding to its neighbouring pixels (4- or 8-neighbourhood)

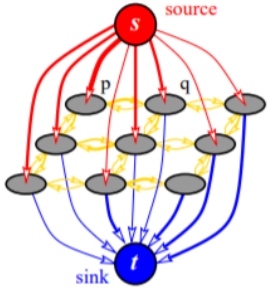
Region-Based Techniques - Graph Cut Segmentation (3)

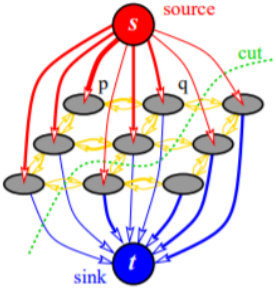
How to compute the *edge weights*:

- For computing the *terminal* edge weights, the feature distribution needs to be estimated at first:
  - We need to define some pixels as foreground and background pixles, e.g. manually
  - Assume that each of the marked/defined pixels has a probability of 1.0 to be in either the foreground or background in the image
  - Probability for all *non-terminal* nodes to either belong to foreground or background has to be computed
- Simplest way to compute those probabilities is to fit a couple of Gaussian distributions to the marked pixels by computing their 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 *pdfs*
- For the *non-terminal* edge weights, the similarities between a pixel node and the nodes corresponding to its neighbouring pixels have to be computed
- E.g. by computing how similar neighborhood pixels are in RGB or any other colour space


Image


Foreground
(source)
Min Cut
Background


source
s
p
q
t
sink


source
s
cut
p
q
t
sink

Region-Based Techniques - Graph Cut Segmentation (5)

- Segmenting an image thus corresponds to cutting this graph into segments
- I.e. removing edges that cross between graph parts corresponding to similar regions
- Obviously, it is easiest to break links that have low affinity
  - Similar pixels should be in the same segments and dissimilar pixels should be in different segments

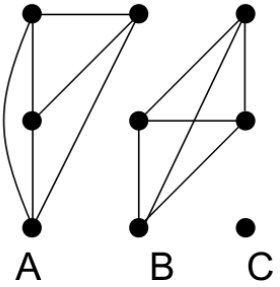


Figure: Segmenting images as graph partitioning

Graph Cut:

- Is the set of edges the removal of which makes a graph disconnected
- Of course a graph cut provides a segmentation

- Cost of a cut is determined by the sum of weights of the cut edges
- Minimum cut (with lowest costs) tends to cut off small, isolated components (see Fig.)
- A problem which is resolved by the **normalised cut**:
  - Here the cost of an edge connecting A and B is normalised by the costs of all edges involving A and by the costs of all edges involving B
  - Minimizing these costs leads to more robust graph cuts

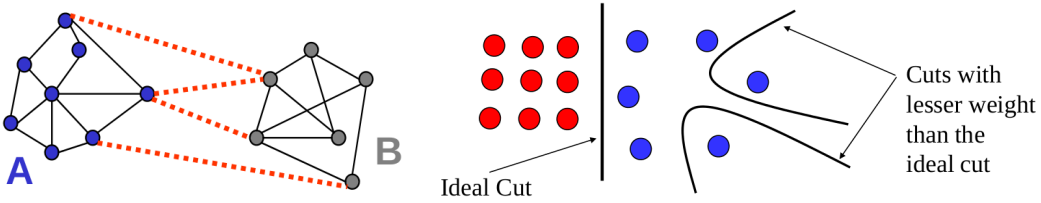


Figure: Problems with minimum cut

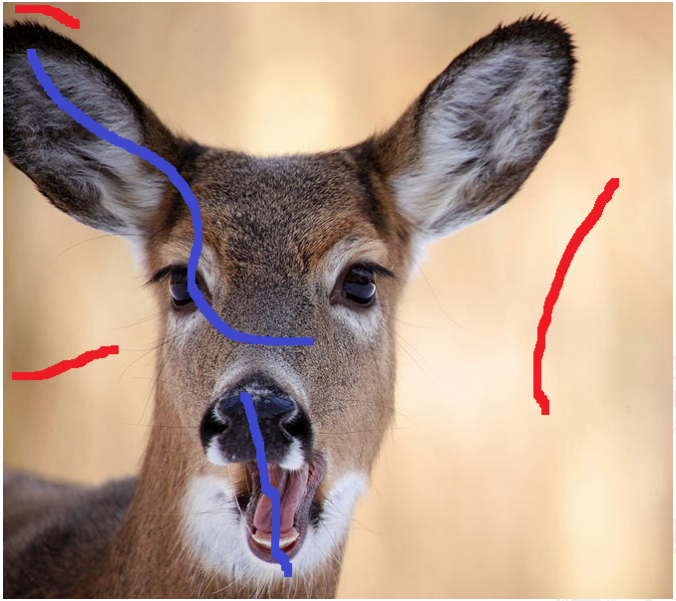


Figure: Graph Cut Example Marked Input Image

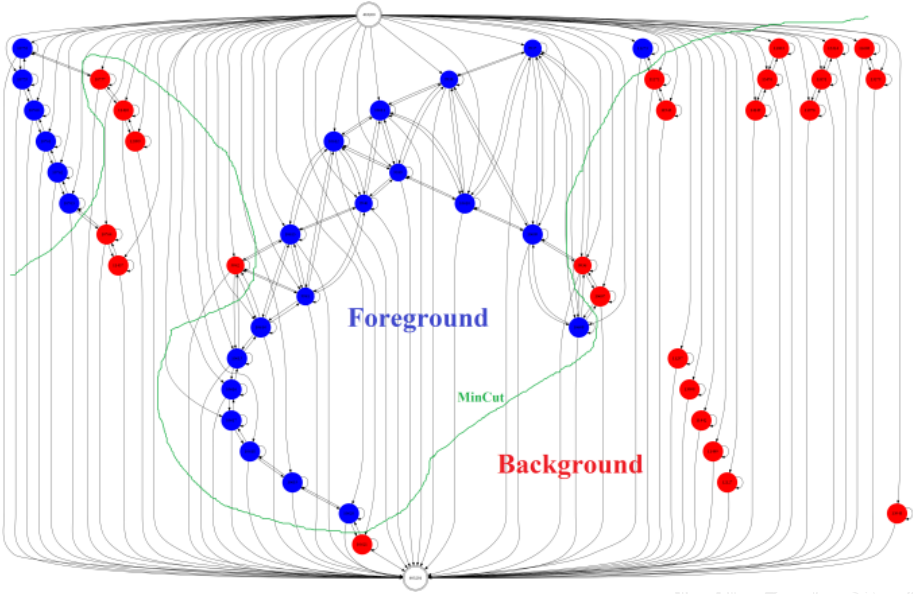


Figure: Graph Cut Example Corresponding Graph

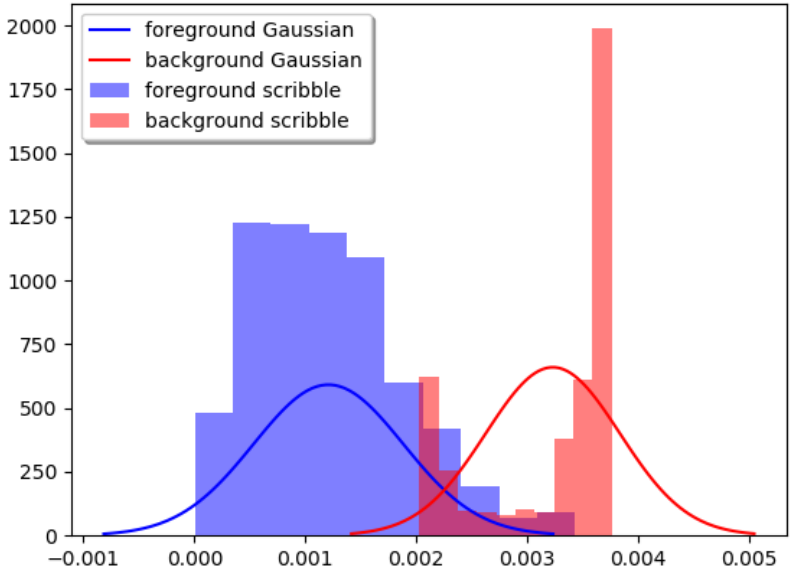


Figure: Graph Cut Example Estimated Probability Densities

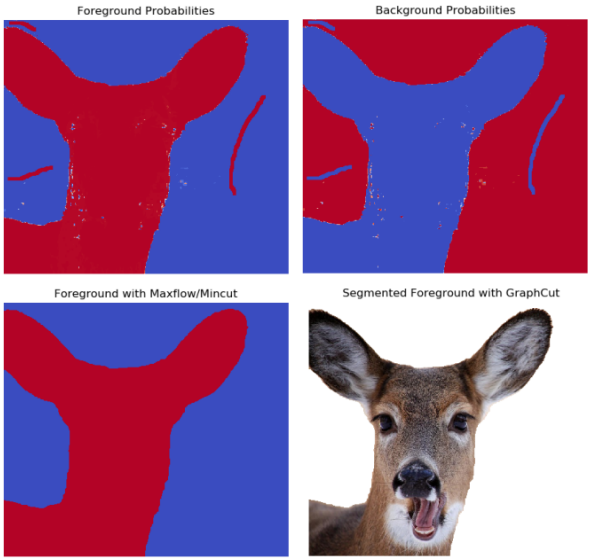


Figure: Graph Cut Example Background/Foreground Probabilities + Result