Basic Image Processing Algorithms

Lecture 8.

PPKE-ITK
Image and Video Segmentation
Previously on... Basic Image Processing

Previous topics:
- Color Spaces, dithering
- 2D convolution, Canny edge detector
- Hough transformation & Image Enhancement
- Fourier analysis
- Texture analysis
- Image recovery
- Segmentation: Otsu, K-means and Morphology

Remaining topics:
- Markov Random Fields, Marked Point Processes
- Mean shift
- Descriptors: SIFT, HOG, Local Binary Patterns
- Video processing
- Machine Learning
- Deep Learning

Classical era mainly from ’60s-’80s (with exceptions)

Modern era mainly from ’00s-’10s (with exceptions)
Recap: Morphological Operations

Limitations: distortion of object shapes

Foreground Mask of MoG ($T=20$)

Closing

Opening
Beyond morphology based approaches?

- Pixel-by-pixel classification: image based knowledge
- Morphology to obtain homogeneous regions: prior knowledge

Pixel-by-pixel classification → Morphology based region refinement → Homogeneous, but often distorted shapes (especially on the object boundaries)

Pixel-by-pixel descriptors → Joint decision considering both factors → Homogeneous shapes with accurate boundaries?
Markov Random Fields in Image Segmentation

- Segmentation as pixel labeling
- Probabilistic approach
  - Segmentation as MAP estimation
  - Markov Random Field (MRF)
  - Gibbs distribution & Energy function
- Classical energy minimization
  - Simulated Annealing
  - Markov Chain Monte Carlo (MCMC) sampling
- Example MRF model & Demo
- Parameter estimation (EM)
Markov Random Fields in Image Segmentation

Main principle

- Mapping the image to a graph
  - nodes are assigned to the different pixels, and the edges connect pixels which are in interaction

- Segmentation as pixel labeling:
  - each pixel gets a class-label from a task-dependent label set $\Lambda$

- Inverse problem formulation:
  - Instead of finding a direct algorithm to find the optimal labeling, we construct a (pseudo-)probability function which assigns a likelihood value to each possible global segmentation, then an optimization process attempts to find the labeling with the highest confidence

- What does the probability function depend on?
  - local feature vectors at each pixel (color, texture etc)
    - classes in $\Lambda$ are as stochastic processes, described by different feature distributions
    - label consistency (soft) constraints between neighboring pixels
      - e.g. for preferring smooth segmentation map we penalize if two neighboring nodes have different labels
Segmentation as a Pixel Labelling Task

- Extract features from the input image
  - Each pixel $s$ in the image has a feature vector $\vec{f}_s$
  - For the whole image, we have:
    $$f = \{\vec{f}_s : s \in S\}$$
- Define the set of labels $\Lambda$
  - Each pixel $s$ is assigned a label $\omega_s \in \Lambda$
  - For the whole image, we have:
    $$\omega = \{\omega_s : s \in S\}$$
  - $\Omega$: set of all possible $\omega$ labelings (i.e. $\omega \in \Omega$)
- For an $N \times M$ image, there are $|\Omega| = |\Lambda|^{NM}$ possible global labelings.
  - Which one is the right segmentation?

Define a **probability measure** on the set of all possible labelings and select the most likely one.

\( P(\omega|f) \) measures the probability of a labelling, given the observed feature \( f \).

Our goal is to find an optimal labeling \( \hat{\omega} \) which **maximizes** \( P(\omega|f) \).

This is called the **Maximum a Posteriori (MAP)** estimate:

\[
\hat{\omega} = \arg\max_{\omega \in \Omega} P(\omega|f)
\]
By Bayes Theorem, we have

$$P(\omega|f) = \frac{P(f|\omega)P(\omega)}{P(f)} \propto P(f|\omega)P(\omega)$$

- $P(f)$ is constant
  - it does not depend on the actual labeling!
- We need to define $P(f|\omega)$ and $P(\omega)$ in our model

We will use **Markov Random Fields**
Why MRF Modelization?

○ In real images, regions are often **homogenous**; neighboring pixels usually have similar properties (intensity, color, texture, ...) → prior neighborhood constraints vs. noisy pixel level descriptors

○ **Markov Random Field (MRF)** is a probabilistic model which captures such contextual constraints
  - Well studied, strong theoretical background
  - Allows Monte-Carlo Markov Chain (MCMC) sampling of the (hidden) underlying structure → **Simulated Annealing**
  - Fast and exact solution for certain type of models → **Graph cut**

[Kolmogorov]
To give a formal definition for Markov Random Fields, we need some basic building blocks:

- Observation Field and (hidden) Labeling Field
- Pixels and their Neighbors
- Cliques and Clique Potentials
- Energy function
- Gibbs Distribution
Recap: Discrete Markov Chains: discrete time, discrete state stochastic processes

- Given: set of possible states \( S_1, S_2, \ldots S_N \)
- \( q_t \): state at time \( t \), \( t = 1, \ldots T \)
- Observed state sequence: \( q_1, q_2, \ldots q_T \)
- Markov property:

\[
P(q_t = S_j | q_{t-1} = S_i) = P(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \ldots q_1 = S_l)
\]

- Conditional probability of the current state only depends on the previous state (i.e. only neighboring states interact – in time)

Markov Random Fields: instead of temporal neighboring states, we consider the spatially neighboring pixels

- Pixel labels are not independent, however, direct dependence is only considered between the spatial neighbors
Definition – Neighbors

- For each pixel, we can define some surrounding pixels as its neighbors.
- Example: 1st order neighbors and 2nd order neighbors
Definition – MRF

The labeling field $X$ can be modeled as a Markov Random Field (MRF) if

1. For all $\omega \in \Omega$: $P(X = \omega) > 0$
2. For every $s \in S$ and $\omega \in \Omega$:

$$P(\omega_s | \omega_r, r \neq s) = P(\omega_s | \omega_r, r \in N_s)$$

- $N_s$ denotes the neighbors of pixel $s$
The **Hammersley-Clifford Theorem** states that a random field is a MRF if and only if $P(\omega)$ follows a **Gibbs distribution**.

$$P(\omega) = \frac{1}{Z} \exp(-U(\omega)) = \frac{1}{Z} \exp\left( - \sum_{c \in C} V_c(\omega) \right)$$

- where $Z = \sum_{\omega \in \Omega} \exp(-U(\omega))$ is a normalization constant

**Practical consequence:**
- probability functions of MRFs have a special form: they can be factorized into small terms called **clique potentials**, which can be locally calculated on the graph
- this property makes possible to design the **probability function** in a modular way, and enables using efficient iterative optimization techniques
- Technical note: instead of maximizing this probability function we usually minimize the minus logarithm of it which is called **energy function**
Definition – Clique

- The H-C theorem provides us an easy way of defining MRF models via *clique potentials*.
- A subset $C \subseteq S$ is called a *clique* if every pair of pixels in this subset are neighbors.
- A clique containing $n$ pixels is called $n^{th}$ order clique, denoted by $C_n$.
- The set of cliques in an image is denoted by

$$C = C_1 \cup C_2 \cup \ldots \cup C_K$$

![Diagram showing singleton and doubleton cliques](image)
For each clique $c$ in the image, we can assign a value $V_c(\omega)$ which is called **clique potential** of $c$, where $\omega$ is the configuration of the labeling field.

The sum of potentials of all cliques gives us the energy $U(\omega)$ of the configuration $\omega$.

\[
U(\omega) = \sum_{c \in C} V_c(\omega) = \\
= \sum_{i \in C_1} V_{C_1}(\omega_i) + \sum_{(i,j) \in C_2} V_{C_2}(\omega_i, \omega_j) + \cdots
\]
Segmentation of grayscale images: A simple MRF model

- Construct a segmentation model where regions are formed by spatial clusters of pixels with similar intensity:

![Diagram showing the process of constructing a segmentation model involving model parameters, an input image, an MRF segmentation model, and the resulting segmentation.]
Pixel labels (or classes) are represented by Gaussian distributions

\[
P(f_s | \omega_s) = \frac{1}{\sqrt{2\pi}\sigma_{\omega_s}} \exp \left( - \frac{(f_s - \mu_{\omega_s})^2}{2\sigma_{\omega_s}^2} \right)
\]

Clique potentials

- **Singleton**: proportional to the likelihood of features given \( \omega \): \( \log P(f | \omega) \)
- **Doubleton**: favors similar labels at neighboring pixels – *smoothness prior*

\[
V_{C_2}(i, j) = \beta \delta(\omega_i, \omega_j) = \begin{cases} 
-\beta & \text{if } \omega_i = \omega_j \\
+\beta & \text{if } \omega_i \neq \omega_j
\end{cases}
\]

- as \( \beta \) increases, regions become more homogenous
Model parameters

- Doubleton potential $\beta$
  - less dependent on the input $\rightarrow$
    - can be fixed a priori
- Number of labels $|\Lambda|$
  - Problem dependent $\rightarrow$
    - usually given by the user or
    - inferred from some higher level knowledge
- Each label $\lambda \in \Lambda$ is represented by a Gaussian distribution $N(\mu_\lambda, \sigma_\lambda)$:
  - estimated from the input image
The class statistics (mean and variance) can be estimated via the **empirical mean and variance**:

\[
\forall \lambda \in \Lambda: \quad \mu_\lambda = \frac{1}{|S_\lambda|} \sum_{s \in S_\lambda} f_s
\]

\[
\sigma^2_\lambda = \frac{1}{|S_\lambda|} \sum_{s \in S_\lambda} (f_s - \mu_\lambda)^2
\]

- where \( S_\lambda \) denotes the set of pixels in the training set of class \( \lambda \)
- a training set consists in a representative region selected by the user
Now we can define the energy function of our MRF model:

\[
U(\omega) = \sum_s \left( \log(\sqrt{2\pi}\sigma_{\omega_s}) + \frac{(f_s - \mu_{\omega_s})^2}{2\sigma_{\omega_s}^2} \right) + \sum_{s,r} \beta \delta(\omega_s, \omega_r)
\]

Recall:

\[
P(\omega) = \frac{1}{Z} \exp(-U(\omega)) = \frac{1}{Z} \exp \left( -\sum_{c \in C} V_c(\omega) \right)
\]

Hence:

\[
\hat{\omega}^{MAP} = \arg\max_{\omega \in \Omega} P(\omega|f) = \arg\min_{\omega \in \Omega} U(\omega)
\]
Problem reduced to the minimization of a non-convex energy function

- Many local minima

Gradient descent?

- Works only if we have a good initial segmentation

Simulated Annealing

- Always works (at least in theory)
ICM (Iterated Conditional Mode)
~Gradient descent approach [Besag86]

1. Start at a „good” initial configuration $\omega^0$ and set $k = 0$.
2. For each configuration which differs at most in one element from the current configuration $\omega^k$ (they are denoted by $\mathcal{N}_{\omega^k}$), compute the energy $U(\eta)$ ($\eta \in \mathcal{N}_{\omega^k}$).
3. From the configurations $\mathcal{N}_{\omega^k}$, select the one which has the minimal energy:

$$\omega^{k+1} = \arg\min_{\eta \in \mathcal{N}_{\omega^k}} U(\eta)$$

4. Goto Step 2, with $k = k + 1$ until convergence obtained (for example the energy change is less than a certain threshold).
ICM (Iterated Conditional Mode)
ICM for mage segmentation models

1. Start at a „good“ initial segmentation $\omega^0$ and set $k = 0$.
2. For each segmentation which differs at most in one pixel’s label (pixel s) from the current segmentation $\omega^k$ (they are denoted by $\mathcal{N}_{\omega^k}$), compute the energy:
   \[
   \Delta U(\eta) = U(\eta) - U(\omega^k) \quad (\eta \in \mathcal{N}_{\omega^k}).
   \]
3. From the configurations $\mathcal{N}_{\omega^k}$, select the one which has the minimal energy:
   \[
   \omega^{k+1} = \arg\min_{\eta \in \mathcal{N}_{\omega^k}} \Delta U(\eta)
   \]
4. Goto Step 2, with $k = k + 1$ until convergence obtained (for example the energy change is less than a certain threshold).
Per-pixel Maximum a Posteriori (MAP) estimate:

\[
\omega_s^0 = \arg\min_{\lambda \in \Lambda} \left( \log(\sqrt{2\pi}\sigma_\lambda) + \frac{(f_s - \mu_\lambda)^2}{2\sigma^2_\lambda} \right)
\]
ICM optimization steps
Simulated Annealing: accept a move even if energy increases (with certain probability)

Can get stuck in local minima!

Slide adopted from C. Rother ICCV’09 tutorial: http://research.microsoft.com/
Simulated Annealing
Modified Metropolis Dynamics (MMD)

1. Set $k = 0$ and initialize $\omega$ randomly. Choose a sufficiently high initial temperature $T = T_0$.
2. Construct a trial perturbation $\eta$ from the current configuration $\omega$ such that $\eta$ differs only in one element from $\omega$.
3. \textbf{(Metropolis criteria)} Compute $\Delta U = U(\eta) - U(\omega)$ and accept $\eta$ if $\Delta U < 0$ else accept with probability $\exp(-\Delta U / T)$ (analogy with thermodynamics):

$$\omega = \begin{cases} 
\eta & \text{if } \Delta U \leq 0 \\
\eta & \text{if } \Delta U > 0 \text{ and } \xi < \exp(-\Delta U / T) \\
\omega & \text{otherwise}
\end{cases}$$

where $\xi$ is a uniform random number in $[0, 1]$.
4. Decrease the temperature $T = T_{k+1}$ and goto step 2 with $k = k + 1$ until the system is frozen.
Temperature Schedule

- **In theory**: should be logarithmic – **in practice**: exponential schedule is reasonable
- **Initial temperature**: set it to a relatively low value (~4) → faster execution
  - must be high enough to allow random jumps at the beginning!
- **Schedule**: $T_{k+1} = c \cdot T_k$, $k = 0, 1, 2, ...$ (e.g. $c = 0.95$).
- **Stopping criteria**:
  - Fixed number of iterations
  - Energy change is less than a thresholds
MMD segmentation

- Starting MMD: random label map!
ICM vs MMD

ICM result

MMD result
MRF Summary

- Design your model carefully
  - Optimization is just a tool, do not expect a good segmentation from a wrong model
- What about other than graylevel features?
  - Extension to color is relatively straightforward
What color features?
We adopt the CIE-L* u* v* color space because it is perceptually uniform.

- Recap from earlier slides: similarly to CIE-L* a* b*, color difference can be measured here by Euclidean distance of two color vectors.

We convert each pixel from RGB space to CIEL* u* v* space

- We have 3 color feature images

L*

u*

v*
Pixel labels (or classes) are represented by three-variate Gaussian distributions

\[
P(f_s | \omega_s) = \frac{1}{\sqrt{2\pi} |\Sigma_{\omega_s}|} \exp \left( -\frac{1}{2} (f_s - \mu_{\omega_s})\Sigma_{\omega_s}^{-1} (f_s - \mu_{\omega_s})^T \right)
\]

Clique potentials

- **Singleton**: proportional to the likelihood of features given \( \omega \): \( \log P(f | \omega) \)
- **Doubleton**: favors similar labels at neighboring pixels – *smoothness prior*

\[
V_{c_2}(i,j) = \beta \delta(\omega_i, \omega_j) = \begin{cases} 
-\beta & \text{if } \omega_i = \omega_j \\
+\beta & \text{if } \omega_i \neq \omega_j
\end{cases}
\]

- as \( \beta \) increases, regions become more homogenous
Segmentation examples

gray level based segmentation

color image segmentation
MPP: extension of MRFs to objects

- **Markov Random Fields (MRF)**
  - able to simultaneously embed a data model – reflecting the knowledge on the image – and prior constraints, such as the spatial smoothness of the solution through a graph based image representation 😊
  - incorporate contextual properties in a flexible way, 😊
  - **limited** in modeling **geometric** information 😞
    - for example, they do not allow setting constrains on the shape of the segmentation regions without leading to prohibitive complexity,

- **Marked Point Processes (MPP)**
  - efficient extension of MRFs, as they work with objects as variables instead of pixels, considering that the number of variables (i.e. number of objects) is also unknown. 😃
  - MPPs embed prior constraints and data models within the same density, therefore similarly to MRFs, algorithms for model optimization, and parameter estimation are available. 😊
Object extraction: bottom-up vs. inverse approaches

- Bottom-up approach: extracting *primitives* (roof blobs, edges, corners etc.) and constructing the buildings from them

- **Advantage:**
  - can be relatively fast

- **Drawbacks:**
  - construction of appropriate primitive filter may be difficult/inaccurate (above a simple color filter is used)
  - challenging to incorporate a priori information (e.g. shape, size)
  - challenging to model object interactions
Object extraction: bottom-up vs. inverse approaches

- Inverse approach:
  - assigning a fitness value to each possible configuration

- Advantages:
  - complex object appearance models can be used
  - easy to incorporate a priori information (e.g. only rectangles) easy to model object interactions (e.g. penalize intersection, favour similar orientation)

- Drawbacks:
  - high computational need (search in a high dimension population space)
  - local maxima or false maxima of the fitness function
Robust extraction of **object configurations** based on probabilistic description

- **Point process**: arbitrary number of pointwise entities in the 2-D (or 3-D) space, \( K \) – e.g. object centers (buildings etc.)
  \[ \vec{p} = \{p_1, p_2, \ldots p_n(\vec{p})\} \]

- **Markers**: geometric description of the shapes (e.g. for rectangles: width, height, orientation)
  \[ M = \left[ -\frac{\pi}{2}, \frac{\pi}{2} \right] \times \left[ L_{\min}, L_{\max} \right] \times \left[ l_{\min}, l_{\max} \right] \]

- **Marked object** = point + markers
  \[ u_i \in \mathcal{H} = K \times M \]
MPP definitions

- $\mathcal{H}$: space of $u$ objects
  - $\sim$ neighborhood relation in $\mathcal{H}$: $\mathcal{N}_\omega(u) = \{v \in \omega | u \sim v\}$
- $\Omega$: configuration space
  \[ \Omega = \bigcup_{n=0}^{\infty} \Omega^n, \quad \Omega^n = \{\{u_1, u_2, ..., u_n\} \in \mathcal{H}^n\} \]
- Gibbs distribution on the configuration space:
  \[ P_\mathcal{F}(\omega) = P(\omega | \mathcal{F}) = \frac{1}{Z} \exp(-\Phi_\mathcal{F}(\omega)) \]
  - with normalizing constant:
  \[ Z = \sum_{\omega \in \Omega} \exp(-\Phi_\mathcal{F}(\omega)) \]
MPP definitions

- Configuration energy function:

\[ \Phi_F(\omega) = \sum_{u \in \omega} A(u) + \sum_{u \sim v} I(u, v) \]

Data terms: how the individual objects fit the observed image

Interaction terms: penalize/favour various interactions between objects (e.g. non-overlapping, parallel alignment)
Optional material, not discussed at the lecture

Optimization
Multiple birth and death algorithm in nutshell

- Start with an empty population
- Iterate object birth and death steps till convergence is obtained in the global population
  - *Birth*: we generate multiple random objects and add them to the population.
  - *Death*: we remove some probably incorrect objects

Multiple birth and death algorithm - details

1. **Initialization**: calculate a $P_b(.) : S \rightarrow \mathbb{R}$ birth map using the $\mathcal{F}$ input data, which assigns to each pixel $s$ a pseudo probability value $P_b(s)$ estimating how likely $s$ is an object center (without domain info a uniform map can be used).

2. **Main program**: initialize the inverse temperature parameter $\beta = \beta_0$ and the discretization step $\delta = \delta_0$ and alternate birth and death steps.

3. **Convergence test**: if the process has not converged, increase the inverse temperature $\beta$ and decrease the discretization step $\delta$ by a geometric scheme and go back to the birth step. The convergence is obtained when all the objects added during the birth step, and only these ones, have been killed during the death step.
Multiple birth and death algorithm - details

- **Birth step**: for each pixel \( s \in S \), if there is no object with center \( s \) in the current configuration \( \omega \), choose birth with probability \( \delta P_b(s) \). If birth is chosen in \( s \):
  - generate a new object \( u \) with center \( s \)
  - set the object parameters (marks of \( u \)) and add \( u \) to the current configuration \( \omega \).

- **Death step**: Consider the configuration of objects \( \omega = \{u_1, u_2, ..., u_n\} \) and sort it from the highest to the lowest value of the unary (data) term \( \varphi_Y(u) \). For each object \( u \) taken in this order compute the death rate as follows:

\[
d_\omega(u) = \frac{\delta a_\omega(u)}{1 + \delta a_\omega(u)} \quad \text{where} \quad a_\omega(u) = \exp \left( -\beta \left( \Phi_F(\omega/\{u\}) - \Phi_F(\omega) \right) \right)
\]

and kill \( u \) with probability \( d_\omega(u) \)
Optional material, not discussed at the lecture

Multiple birth and death algorithm
Building detection Results

Tree detection from infrared images
Likelihood vs dataterm
Application to flamingo counting

Estimation of the size of a colony in Camargue:
• 557 detected flamingos

© Ariana team, INRIA Sophia Antipolis, France
Application to flamingo counting

- Estimation of the size of a colony in Turkey (2004):
  - 3682 detected flamingos

© Ariana team, INRIA Sophia Antipolis, France
Some references

Markov random fields in image analysis


Marked Point Processes in image analysis

Watershed algorithm

- A mathematical morphology based approach on image segmentation
Watershed Segmentation

- A grey-level image may be seen as a topographic surface, where the grey level of a pixel is interpreted as its altitude in the surface.
- The goal of the algorithm is to find the „watersheds” that are separating the „catchment basins” from each other.

Concept of the watershed algorithm*

Watershed-Basic Definitions

- $I$: 2D gray level image

- Path $P$ of length $\ell$ between $p$ and $q$ in $I$
  - A $(\ell + 1)$-tuple of pixels $(p_0 = p, p_1, ..., p_\ell = q)$ such that $p_i, p_{i+1}$ are adjacent (4 adjacent, 8 adjacent, or $m$ adjacent)
  - $\ell(P)$: the length of a given path $P$

- Minimum
  - A minimum $M$ of $I$ is a connected plateau of pixels from which it is impossible to reach a point of lower altitude without having to climb
Piercing holes in each regional minimum of $I$

The 3D topography is flooded from below gradually

When the rising water in distinct catchment basins is about to merge, a dam is built to prevent the merging

Instead of working on an image itself, this technique is often applied on its gradient image.
Watershed-Basic Definitions

Three types of points

• Points belonging to a regional minimum
• Catchment basin / watershed of a regional minimum
  • Points at which a drop of water will certainly fall to a single minimum
• Divide lines / Watershed lines
  • Points at which a drop of water will be equally likely to fall to more than one minimum
  • Crest lines on the topographic surface

This technique is to identify all the third type of points for segmentation
Basic Steps

- The dam boundaries correspond to the watershed lines to be extracted by a watershed segmentation algorithm.
- Eventually only constructed dams can be seen from above.
Based on binary morphological dilation

At each step of the algorithm, the binary image in obtained in the following manner:

1. Initially, the set of pixels with minimum gray level are 1, others 0.
2. In each subsequent step, we flood the 3D topography from below and the pixels covered by the rising water are 1s and others 0s. (See previous slides)
The dam is constructed by the points on which the dilation would cause the sets being dilated to merge.

- Result: one-pixel thick connected path
- Setting the gray level at each point in the resultant path to a value greater than the maximum gray value of the image. Usually max+1
Distance transform operator:

- Input: binary image (showing foreground/background regions)
- Result: a graylevel image, where the graylevel intensities of points inside foreground regions are show the distance to the closest boundary from each point
- Implementation: through morphological operations
- Often used as input of the Watershed transform (instead of the gradient image)

Example 1 - Watershed Transform of Binary Image Using the Distance transform

Distance transform of a binary image is defined by the distance from every pixel to the nearest non-zero valued pixel.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A: Original image                          B: Negative of image A
C: Distance transform of B                 D: Watershed transform of C
Example 1 - Watershed Transform of Binary Image Using the Distance transform

- Segmentation example applying watershed to the inverse distance image using the binary mask
Examples 2 - oversegmentation

(a) Original image
(b) Gradient image of image (a)
(c) Watershed lines obtained from image b (oversegmentation)
   ➔ Each connected region contains one local minimum in the corresponding gradient image
(d) Watershed lines obtained from smoothed image (b)
Simple trick

- Use median filter to reduce number of regions
Simple trick

- Use median filter to reduce number of regions
Object segmentation by watershed algorithm

- Task: segmentation of (possibly touching) objects in front of a background

Electrophoresis image
Watershed Segmentation

- Over-segmentation problem:
  - most times the real watershed transform of the gradient present many catchment basins, each one corresponds to a minimum of the gradient that is produced by small variations, mainly due to noise.

The Use of Markers

- Over-segmentation problem
  - Usually, we cannot overcome it with simple filtering (like median)
  - Use of markers can be a solution
- Internal markers are used to limit the number of regions by specifying the objects of interest
  - Like seeds in region growing method
  - Can be assigned manually or automatically
  - Regions without markers are allowed to be merged (no dam is to be built)
- External markers: pixels where we are confident to belong to the background
  - Watershed lines are typical external markers and they belong to the same (background) region
Watershed Based Image Segmentation

- Use internal markers to obtain watershed lines of the gradient of the image to be segmented.
- Use the obtained watershed lines as external markers.
- Each region defined by the external markers contains a single internal marker and part of the background.
- The problem is reduced to partitioning each region into two parts: object (containing internal markers) and a single background (containing external markers).
Over-segmentation: solution

FIRST STEP: we mark each blob of protein of the original image

Image with a few markers (not all blobs are marked here)
Now we look at the final result of the marking as a topographic surface, but in the flooding process instead of piercing the minima, we only make holes through the components of the marker set that we produced.

This way the flooding will produce as many catchment basins as there are markers in $M$, and the watershed lines of the contours of the objects will be on the crest lines of this topographic surface.
Partitioning each region into two parts: object (containing internal markers) and a single background (containing external markers)

- Global thresholding, region growing, region splitting and merging...
Watershed segmentation example

- Use the Gradient Magnitude as the Segmentation Function
  - The gradient is high at the borders of the objects and low (mostly) inside the objects.

Original image (I)

Gradient magnitude image
Watershed segmentation example

- Obtaining good foreground markers: regional maxima of the morphology enhanced input image

Result of grayscale morphology (M)  
Regional maxima of (M) superimposed on original image (I)
Watershed segmentation example

- Obtaining good background markers
  - Step 1: threshold the morphology enhance image

- Result of grayscale morphology (M)
- T: result of Otsu threshold on M
Watershed segmentation example

- Obtaining good background markers
  - Step 1: threshold the morphology enhance image
  - Step 2: using the watershed transform of the distance transform of T, and then looking for the watershed ridge lines of the result

- T: result of Otsu threshold on M
- Watershed lines (background markers)
Watershed segmentation: Visualization of the results

Superimpose the foreground markers, background markers, and segmented object boundaries.

Segmentation results: display the label matrix as a color image

Matlab tutorial example, with source code:
Summary: Watershed Segmentation

- There are 3 types of pixels:
  - Points belonging to a regional minimum
  - Point belonging to the catchment basin of a regional minimum
  - Points belonging to a watershed line

- The resulted boundaries of the regions are continuous.
- But it is time consuming and has over-segmentation problems.
- The solution to the over-segmentation is to use markers:
  - Internal markers:
    - Each one correspond to one object
    - Surrounded by points with higher altitude
    - Points in a region form a connected component
    - The points of the connected component has the same intensity
  - External markers:
    - Segment the image into regions with one internal marker object and background points.