Introduction to Medical Imaging
(5XSA0)
Module 5
Segmentation

Jungong Han, Dirk Farin, Sveta Zinger
(s.zinger@tue.nl)

Outline
- Introduction
- Color Segmentation
  - region-growing
  - region-merging
  - watershed
- Background Subtraction
  - background generation
  - change detection
- Model-based Object Detection
  - graph model

Applications of Object Segmentation
- Surveillance
  - detect people in restricted areas
  - detect abnormal behavior (motion pattern)
- Video/Image Analysis
- Video Editing
- Intelligent Video Databases
  - object classification
- Sports Analysis
- Object-Oriented Video-Coding (MPEG-4)

Definition of Object Segmentation
- The object definition and the segmentation algorithm to use depends on the context!
- There is no such thing as “a general segmentation algorithm”
- Possible definitions:
  - regions with uniform color → color segmentation
  - regions with uniform texture → texture segmentation
  - regions with uniform motion → motion segmentation

Definition of Object Segmentation
- It’s your turn. Segment this image!

Definition of Object Segmentation
- Central question: what is an object???
- Semantic problems of object definition:
  - shadows
  - occlusions
  - reflections
  - object status change (parking cars)
  - small movements (waving trees)
  - hierarchical objects (man in car)
Color Segmentation: Example (1)

* Example medical application:
  - count number of blood cells

input image  
binarized image  separate connected cells (requires further model knowledge)

Color Segmentation: Example (2)

* Preprocessing for special applications
  - colors help to detect objects and markers

input image  segmentation

Images taken from CMVision realtime color segmentation library

Color Segmentation: Example (3)

* Color segmentation on natural images:

input image  segmentation

Texture Segmentation

* Group regions based on texture features.

* Usual approach:
  - extract texture descriptors (e.g., Gabor filter coefficients)
  - cluster similar descriptors (comparable to color segmentation)

Background Subtraction

* Assume that a pure background image is known
  - detect changes between input image and background image (Change Detection)

Manual Segmentation

* Interactive segmentation
  - user controls the segmentation process
  - computer determines accurate object boundaries

* Edge based techniques
  - intelligent scissors

* Region based techniques
  - marker based watershed

Selecting Objects with Freehand Sketches
Evaluating the Segmentation Result

- oversegmentation
  - result has more regions than expected
- undersegmentation
  - result has less regions than expected

Point detection

- Look for a point that is different from its neighborhood
- Apply an isolating mask to calculate:
  \[ R = w_1z_1 + w_2z_2 + \ldots + w_9z_9 \]
- A point is detected at the center of the mask if \(|R| \geq T\)

where \(T\) is a threshold

Line detection

- Masks responding to lines of different orientations:
  - Thresholded absolute values of \(-45^\circ\) detector
  - Absolute values of \(-45^\circ\) detector (zoomed)

Outline

- Introduction
- Color Segmentation
  - region-growing
  - region-merging
  - watershed
- Background Subtraction
  - background generation
  - change detection
- Model-based Object Detection
  - graph model

Color Segmentation: Region Growing (1)

- Start a new region with a seed pixel
- Consider neighboring pixels
  - if their color is similar to the mean region color, add pixel to the region
  - continue growing until no more pixels can be added to the region

Color Segmentation: Region Growing (2)

- Implementation
  - input: seed \((x_s,y_s)\), image \(I(x,y)\)
  - insert \((x_s,y_s)\) into queue of positions \(Q\)
  - region color \(C = I(x_s,y_s)\)
  - while \(Q\) not empty
    - extract \((x,y)\) from queue
    - if \(||I(x,y) - C|| < \tau\)
      - add neighbors of \((x,y)\) into \(Q\)
      - update region color \(C\) with \(I(x,y)\)
Color Segmentation: Region Growing (3)

* Regions have already been defined
\[ R = \bigcup_{i=1}^{S} R_i \quad R_i \cap R_j = \emptyset \quad i \neq j \]

* We hope
\[ H(R_i) = \text{TRUE} \quad i = 1, 2, ..., S \]
\[ H(R_i \cup R_j) = \text{FALSE} \quad i \neq j, \quad R_i \text{ adjacent to } R_j \]

Color Segmentation: Region Growing (4)

* Seed pixels can be
  -- placed manually, or
  -- arbitrarily chosen from the unprocessed pixel

Color Segmentation: Region Growing (5)

* Segmentation result is depending on
  -- similarity threshold
  -- choice of seed pixels
  -- order in which seed pixels are processed
  -- algorithm implementation (order in which neighboring pixels are processed)

Color Segmentation: Region Merging (1)

* Alternative approach: Region Merging
  -- Start with a set of regions (obtained by another algorithm)
  -- Consecutively join the two most similar neighboring regions

Color Segmentation: Region Merging (2)

* Data-structure:
  -- neighborhood graph of image regions
  -- each node represents a region
  -- graph edges denote adjacency
  -- they are attributed with the region similarity

Color Segmentation: Region Merging (3)

![Diagram of strong and weak similarity in region merging]
Color Segmentation: Region Merging (4)
* Several merging criteria are possible
* Simple model:
  - describe region color \( l_i \) by average luminance.
  
    \[
    l_i = \frac{1}{r_i} \sum_{(x,y) \in r_i} I(x,y)
    \]

* Merging criterion:
  - mean: difference of mean region luminances
    \[
    \text{mean}(l_i) - \text{mean}(l_j)
    \]

* Difference of mean luminances does not take region size into account

Color Segmentation: Watershed (1)
* The watershed algorithm
* Fast color segmentation algorithm
* Generally gives oversegmented results on natural images
* Well-defined output
* Often used as preprocessing operation
  - quickly convert a pixel image into a region-level description to speed-up further processing
* Variant: watershed with manually placed markers

Color Segmentation: Watershed (2)
* Algorithm principle:
  - apply gradient filter on input image and work on the resulting edge image
  - search for local minima in edge image and initialize a new region for each minimum
  - extend regions like follows:
    Watersheds are built where water from different ‘lakes’ meets

Color Segmentation: Watershed (3)
* input image
* edge profile of input image

Color Segmentation: Watershed (4)
* water begins to rise from local minima
* build watershed when different lakes touch
* water continues to rise; build more watersheds

Color Segmentation: Watershed (5)
* Typical result:
  - Many regions because camera noise generates many local minima
**Color Segmentation: Watershed (6)**

- Noise in flat areas generates too many local minima
- Small noise regions can be avoided by clipping gradients to a minimum value

**Color Segmentation: Watershed (7)**

- Result with clipped gradient strength:

**Color Segmentation: Watershed (8)**

- Variant: manual watershed
- Instead of starting to flood from local minima, start flooding from markers

**Color Segmentation**

- Summary
  - Region Growing
    - easy implementation, low-quality results
  - Region Merging
    - difficult implementation, flexible control of segmentation process, high-quality results
  - Watershed
    - easy and fast implementation, predictable result, severe over-segmentation
    - no threshold to influence result (except gradient clipping)
    - Result only sufficient for specialized applications

**Outline**

- Introduction
- Color Segmentation
  - region-merging
  - watershed
- Background Subtraction
  - background generation
  - change detection
- Model-based Object Detection
  - graph-models

**Thresholding**

- Simple and computationally efficient
- Threshold selection uses intensity information ➔ histogram
- Example: bimodal histogram
**Optimal Thresholding**

- What value of $T$ will give us the best segmentation?
- Gonzalez and Woods:
  - Initialize $T$ (e.g., halfway between min and max)
  - Iteratively set $T = 0.5(\mu_1 + \mu_2)$, where $\mu_1, \mu_2$ the mean values of pixels with value larger or smaller than $T$, respectively
  - Until $T$ converges

**Bkg. Subtraction: Principle - (1)**

- Assumption: background is static
- Input image is compared with background image
  - if the difference is small: background content
  - if the difference is large: foreground object
  - difference is generally not zero due to noise in the sequences
- For some applications, the background image can be captured separately
- For other applications, it must be synthesized from the input sequence

**Bkg. Subtraction: Principle (2)**

- Background image generation
- Change detection

**Bkg. Sub: Background Generation (1)**

- Temporal median filter
  - good performance can be achieved
  - relatively low computational complexity
- $B(x) = \text{median} (I_1(x), I_2(x), \ldots, I_{N\times K}(x))$

**Bkg. Sub: Change Detection**

- We have
  - current input frame $I_i$
  - background frame $I_b$
- We denote the image color channels as $I_i(x, y, z)$ and the vector combining all channels as $I_i$
- Detect object, e.g., if $r$
- $r$ is a threshold that depends on the noise level
Bkg. Sub: Difference Metrics

- What metric should be used for the image difference?
  - greyscale difference
  - sum of squared differences of RGB channels
  - sum of absolute differences of RGB channels
  - ... in YUV color-space
  - ... something else (L*u*v color-space, non-linear difference functions?)

Distribution of Color Differences (1)

```
Distribution of difference vector d between input color I, and background color I_b
```

Distribution of Color Differences (2)

- Each difference function defines a decision boundary
  - here: Euclidean distance defines sphere
  - inside sphere: background, outside: foreground

Distribution of Color Differences (3)

- Greyscale difference only: Defines slice in Y-dimension
- Better classification could be obtained by integrating color

Distribution of Color Differences (4)

- Euclidean distance in YUV space
- Defines sphere
- However, background distribution has lower variance in U,V dimensions

Background Subtraction

- Summary
  - change detection between input frame and background frame
  - independent pixel classification
    - RGB, YUV color difference
    - greyscale difference
    - sum of squared difference
    - Mahalanobis distance
  - classify groups of pixels
    - joint probability has less overlap
    - results are more robust to noise
Outline

- Introduction
- Color Segmentation
  - region-merging
  - watershed
- Background Subtraction
  - background generation
  - change detection
- Model-based Object Detection
  - graph model

GM Algorithm: Principle (1)

- How can we tell the computer what objects we want to extract?

GM Algorithm: Principle (2)

- The same “object” can show in many deformed appearances
- Objects can be partly occluded

GM Algorithm: Principle (3)

- Idea: Describe objects with graph-based models
- The graph consists of:
  - nodes for the essential object regions
  - nodes have attributes with region features (color, size)
  - edges denote spatial relationships between regions

GM Algorithm: Principle (4)

- Generation of the object model:
  - user takes example image and places markers into essential regions. Apply marker driven watershed
  - user specifies spatial relationships between regions that must be fulfilled

GM Algorithm: Principle (5)

- sample object image
- model graph, region features
- automatic segmentation
- feature extraction
- matching
**Graph-Model: Graph Matching (1)**

- **Graph Matching**
  - model graph \((V_M, E_M)\)
  - segmentation graph \((V_S, E_S)\)
  - mapping
  - cost function
  - determine

**Graph-Model: Graph Matching (2)**

- Matching cost function consists of
  - **Node costs**
    - similarity of regions (color, shape, ...)
  - **Edge cost**
    - relations between regions (distances, size ratios)

**Graph-Model: Result (1)**

- **extension**: 1:N matching

**Graph-Model: Result (3)**

- **1:N matching**

**Graph-Model based Object Detection**

- **Summary**
  - algorithm to detect general objects
  - object is described with attributed graphs (only tree-shaped)
  - manual definition of model graph
  - automatic color segmentation to obtain segmentation graph
  - efficient graph-matching algorithm to detect model graph in segmentation graph

**References (1)**

- **Color segmentation**

- **Background subtraction**
References (2)

- **Intelligent Scissors**

- **Segmentation based on feature clustering**