CAP 4453
Robot Vision
Dr. Gonzalo Vaca-Castaño
gonzalo.vacacastano@ucf.edu
Administrative details

• Allow grader to review your homework:

• Homework 1 review

• Any issues with hw2 ?
Credits

• Some slides comes directly from:
  • Yogesh S Rawat (UCF)
  • Noah Snavely (Cornell)
  • Ioannis (Yannis) Gkioulekas (CMU)
  • Mubarak Shah (UCF)
  • S. Seitz
  • James Tompkin
  • Ulas Bagci
  • L. Lazebnik
Short Review from last class
Last 2 classes

- Gradient operators
  - Prewit
  - Sobel
- Marr-Hildreth (Laplacian of Gaussian)
- Canny (Gradient of Gaussian)
Regions ↔ Boundaries
Robot Vision

7. Segmentation I
Outline

• Image segmentation basics

• Thresholding based
  • Binarization
  • Otsu

• Region based
  • Merging
  • Splitting

• Clustering based
  • K-means (SLIC)
Outline

• **Image segmentation basics**
  • Thresholding based
    • Binarization
    • Otsu
  • Region based
    • Merging
    • Splitting
  • Clustering based
    • K-means (SLIC)
Image segmentation

• Partition an image into a collection of set of pixels
  • Meaningful regions (coherent objects)
  • Linear structures (line, curve, ...)
  • Shapes (circles, ellipses, ...)

CAP4453
Image segmentation

• Content base image retrieval
• Machine vision
• Medical imaging applications
• Object detection (face detection, ..)
• 3D reconstruction
• Object/motion tracking
• ...
Image segmentation

• In computer vision, image segmentation is one of the oldest and most widely studied problems
  • Early techniques -> region splitting or merging
  • Recent techniques -> Energy minimization, hybrid methods, and deep learning
Image segmentation methods

- Thresholding
- Region Based methods (region growing, ..)
- Clustering (K-means, meanshift)
- Graph-based methods (graph-cut, random walk, ..)
- Shape based methods (level set, active contours)
- Energy minimization methods (MRF, ..)
- Machine Learning based methods
Image segmentation

- Image segmentation partitions an image into regions called segments.
Image segmentation

- Image segmentation partitions an image into regions called segments.

- Image segmentation creates segments of connected pixels by analyzing some similarity criteria:
  - intensity, color, texture, histogram, features
Outline

• Image segmentation basics

• Thresholding based
  • Binarization
  • Otsu

• Region based
  • Merging
  • Splitting

• Clustering based
  • K-means (SLIC)
Image binarization

- Image binarization applies often just one global threshold $T$ for mapping a scalar image $I$ into a binary image.
Image binarization

• Image binarization applies often just one global threshold $T$ for mapping a scalar image $I$ into a binary image

$$J(x, y) = \begin{cases} 
0 & \text{if } I(x, y) < T \\
1 & \text{otherwise.}
\end{cases}$$
Image binarization

• Image binarization applies often just one global threshold $T$ for mapping a scalar image $I$ into a binary image

$$J(x, y) = \begin{cases} 0 & \text{if } I(x, y) < T \\ 1 & \text{otherwise.} \end{cases}$$

• The global threshold can be identified by an optimization strategy aiming at creating “large” connected regions and at reducing the number of small-sized regions, called artifacts.
Image binarization

- Thresholding: Most frequently employed method for determining threshold is based on histogram analysis of intensity levels

- Peak on the left of the histogram corresponds to dark objects
- Peak on the right of the histogram corresponds to brighter objects

Difficulties
1. The valley may be so broad that it is difficult to locate a significant minimum
2. Number of minima due to type of details in the image
3. Noise
4. No visible valley
5. Histogram may be multi-modal
Thresholding examples
Thresholding examples

Threshold Too Low

Threshold Too High
Thresholding examples
Thresholding examples
Outline

• Image segmentation basics
• Thresholding based
  • Binarization
  • Otsu
• Region based
  • Merging
  • Splitting
• Clustering based
  • K-means (SLIC)
Variance

\[ \text{Var}(X) = E[(X - \mu)^2] = \sigma^2_X \]

where \( \mu \) is the expected value. That is,

\[ \mu = \sum_{i=1}^{n} p_i x_i, \quad p_i = \frac{\text{#values in } i}{\text{sum all values (area under curve)}} \]
Variance

\[ \text{Var}(X) = \mathbb{E}[(X - \mu)^2] = \sigma_X^2 \]

**Discrete random variable**

If the generator of random variable \( X \) is discrete with probability mass function \( x_1 \rightarrow p_1, x_2 \rightarrow p_2, \ldots, x_n \rightarrow p_n \), then

\[ \text{Var}(X) = \sum_{i=1}^{n} p_i \cdot (x_i - \mu)^2, \]

where \( \mu \) is the expected value. That is,

\[ \mu = \sum_{i=1}^{n} p_i x_i. \]

*[Variance - Wikipedia]*
Otsu thresholding

• Definition: The method uses grey-value histogram of the given image I as input and aims at providing the best threshold (foreground/background)

• Otsu’s algorithm selects a threshold that maximizes the between-class variance $\bar{\sigma}_b^2$ or minimize within-class variance $\sigma_w^2$

• For each threshold $t$ in [0, 255], pixels can be separated into two classes, $C1$ and $C2$; those pixels whose $P_i < t$ are put into $C1$, otherwise into $C2$
• The possibilities of $C1$ and $C2$ separated by $t$, denoted as $W1$ and $W2$, respectively. For example,
  $$W1 = \frac{\text{(#pixels in } C1)}{\text{(total pixels count)}}.$$  
• Given $H$, $W1$, and $W2$, for each $t$, compute the between-class variance $\sigma_b^2$ or within-class variance $\sigma_w^2$ ($\sigma_b^2 \rightarrow \text{red curve}$)
• optimal cut $t^*$ corresponds to $t$ whose $\sigma_b^2$ is maximum or $\sigma_w^2$ is minimum.
Otsu thresholding

• Definition: The method uses grey-value histogram of the given image \( I \) as input and aims at providing the best threshold (foreground/background)

• Otsu’s algorithm selects a threshold that maximizes the between-class variance \( \sigma_b^2 \)

Option 1: maximum of:

\[
\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2
\]

\[
\mu_1(t) = \sum_{i=1}^t \frac{iP(i)}{w_1(t)}
\]

\[
\mu_2(t) = \sum_{i=t+1}^T \frac{iP(i)}{w_2(t)}
\]

\[
w_1(t) = \sum_{i=1}^t P(i)
\]

\[
w_2(t) = \sum_{i=t+1}^T P(i)
\]
Otsu thresholding

• Definition: The method uses grey-value histogram of the given image $I$ as input and aims at providing the best threshold (foreground/background)

• Otsu’s algorithm selects a threshold that maximizes the between-class variance $\sigma_w^2$ or minimize within-class variance $\sigma_w^2$

Option 2: minimum of:

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t)$$

$$w_1(t) = \sum_{i=1}^{t} P(i) \quad P(i) = \frac{n_i}{n}$$

$$w_2(t) = \sum_{i=t+1}^{I} P(i)$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{w_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{w_2(t)}$$
Step by step example

• Find Otsu threshold for this image

• By minimizing within-class variance

Otsu’s method for image thresholding explained and implemented – Muthukrishnan
Step by step example

Otsu’s method for image thresholding explained and implemented – Muthukrishnan
Step by step example

Otsu’s method for image thresholding explained and implemented – Muthukrishnan
Step by step example

Otsu’s method for image thresholding explained and implemented – Muthukrishnan
Step by step example

Otsu’s method for image thresholding explained and implemented – Muthukrishnan

Thresholding in t=100
Step by step example

within-class variance

\[ w_1(t)\sigma^2_1(t) + w_2(t)\sigma^2_2(t) \]

Otsu’s method for image thresholding explained and implemented – Muthukrishnan
Step by step example

Otsu’s method for image thresholding explained and implemented – Muthukrishnan
Step by step (otsu thresholding)

Minimize within-class variance

- The value of variance remains the same from 28 and 120.
- within-class variance is least at $t=28$ or more precisely between 28 to 120.
- Otsu threshold $= 28$. 

\[
\begin{array}{ccc}
T=22, \sigma^2 = 4092.58 & T=23, \sigma^2 = 3667.60 & T=25, \sigma^2 = 2642.35 \\
T=26, \sigma^2 = 2009.93 & T=28, \sigma^2 = 371.55 & T=124, \sigma^2 = 1316.48 \\
\end{array}
\]
Otsu threshold implementation

```python
# Set total number of bins in the histogram
bins_num = 256

# Get the image histogram
hist, bin_edges = np.histogram(image, bins=bins_num)

# Get normalized histogram if it is required
if is_normalized:
    hist = np.divide(hist.ravel(), hist.max())

# Calculate centers of bins
bin_mids = (bin_edges[:-1] + bin_edges[1:]) / 2.

# Iterate over all thresholds (indices) and get the probabilities w1(t), w2(t)
weight1 = np.cumsum(hist)
weight2 = np.cumsum(hist[:-1])[:-1]

# Get the class means mu0(t)
mean1 = np.cumsum(hist * bin_mids) / weight1
# Get the class means mu(t)
mean2 = (np.cumsum(hist * bin_mids)[:-1]) / weight2[:-1]

# Calculate inter-class variance
inter_class_variance = weight1[:-1] * weight2[1:] * (mean1[:-1] - mean2[1:]) ** 2

# Maximize the inter_class_variance function
val = np.argmax(inter_class_variance)

# Thresholding result
threshold = bin_mids[val]
print("Otsu's algorithm implementation thresholding result: ", threshold)
```
Otsu threshold implementation

Original image

Manually
Th = 90

otsu
Th = 127
Otsu threshold implementation

```python
# Applying Otsu's method setting the flag value into cv.THRESH_OTSU.
# Use a bimodal image as an input.
# Optimal threshold value is determined automatically.
otsu_threshold, img_result = cv2.threshold(
    image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU,
)
print("Obtained threshold: ", otsu_threshold)
```

Obtained threshold: 132.0
Otsu thresholding example
The math!

\[ \sigma_{total}^2 = E[(X - E[X])^2] = E[X_{total}^2] - \mu_{total}^2 \]  

(1)

\[ P_i = \frac{n_i}{n_{total}} \quad \text{When } i \leq t \]

\[ P_i^1 = \frac{n_i}{n_1} \]

\[ P_i^2 = \frac{n_i}{n_2} \quad \text{When } t < i < T \]

\[ P_i = \frac{n_1 P_i^1}{n_{total}} = w_1(t) P_i^1 \]

\[ P_i = \frac{n_2 P_i^2}{n_{total}} = w_2(t) P_i^2 \]

\[ E[X_{total}^2] = \sum_{i=1}^{T} P_i x_i^2 = \sum_{i=1}^{t} P_i x_i^2 + \sum_{i=t+1}^{T} P_i x_i^2 = w_1(t) \sum_{i=1}^{t} P_i^1 x_i^2 + w_2(t) \sum_{i=t+1}^{T} P_i^1 x_i^2 = w_1(t) E[X_{1}^2] + w_2(t) E[X_{2}^2] \]  

(2)

\[ \mu_{total}^2 = (w_1(t) \mu_1 + w_2(t) \mu_2)^2 = w_1^2 \mu_1^2 + 2w_1 w_2 \mu_1 \mu_2 + w_2^2 \mu_2^2 = w_1 (1-w_2) \mu_1^2 + 2 w_1 w_2 \mu_1 \mu_2 + w_2 (1-w_1) \mu_2^2 \]  

(3)
The math!

\[ \sigma_{total}^2 = E[(X - E[X])^2] = E[X_{total}^2] - \mu_{total}^2 \quad \text{(1)} \]

\[ E[X_{total}^2] = w_1(t)E[X_1^2] + w_2(t)E[X_2^2] \quad \text{(2)} \]

\[ \mu_{total}^2 = w_1(1-w_2)\mu_1^2 + 2w_1w_2\mu_1\mu_2 + w_2(1-w_1)\mu_2^2 \quad \text{(3)} \]

\[ \sigma_{total}^2 = w_1E[X_1^2] + w_2E[X_2^2] - [w_1\mu_1^2 + w_1w_2\mu_1^2 + 2w_1w_2\mu_1\mu_2 + w_2\mu_2^2 - w_1w_2\mu_2^2] \]

\[ \sigma_{total}^2 = w_1E[X_1^2] - w_1\mu_1^2 + w_2E[X_2^2] - w_2\mu_2^2 + [-w_1w_2\mu_1^2 - 2w_1w_2\mu_1\mu_2 + w_1w_2\mu_2^2] \]

\[ \sigma_{total}^2 = w_1(t)(E[X_1^2] - \mu_1^2) + w_2(t)(E[X_2^2] - \mu_2^2) + w_1(t)w_2(t)(\mu_2^2 - \mu_1^2) \quad \text{(4)} \]

The math!

\[ \sigma_{total}^2 = E[(X - E[X])^2] = E[X_{total}^2] - \mu_{total}^2 \] (1)

\[ \sigma_{total}^2 = w_1(t)(E[X_1^2] - \mu_1^2) + w_2(t)(E[X_2^2] - \mu_2^2) + w_1(t)w_2(t)(\mu_2^2 - \mu_1^2) \]

\[ \sigma_{total}^2 = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t) + w_1(t)w_2(t)(\mu_2^2(t) - \mu_1^2(t)) \]

within-class variance

between-class variance

fixed

Minimize

Maximize
Questions?