L9 (Image and Video Segmentation)

Image Segmentation

There are two types of segmentation tasks we work with in IVMSP:

- spatial segmentation (usual region-based segmentation)
- temporal segmentation(shot detection)

Requirements

- connectivity: each segment consists of connected image points
- completeness: union of all segments yields complete image
- homogeneity: each segment is homogeneous under given criterion
- closeness: combining two segments gives inhomogeneous region

The requirement of **homogeneity** in segmentation ensure that each segment region consists of pixels that are similar to each other according to a criterion(similar pixel intensity, similar RGB values, similar texture patterns)

Cluster-Based Segmentation

Supervised:

- class prototypes(PDFs on templates) are known
- labeled data
  Unsupervised:
- neither class prototypes nor the number of classes are known beforehand
- groups pixels into clusters based on feature similarity

Unsupervised Thresholding

Goal: find optimal threshold θ that minimizes within-class variance.
Requirements: assumes bimodal distribution (foreground vs background)

$$arg \min_{\theta} \sigma^2_{wcv}(\theta) = w_F(\theta)\sigma^2_F(\theta) + w_B(\theta)\sigma^2_B(\theta)$$

where $w_F = \frac{N_F(\theta)}{N}$ and $w_B(\theta) = \frac{N_B(\theta)}{N}$ such that $w_F(\theta) + w_B(\theta) = 1$

Problems:

- requires exhaustive search
- computing variances can be expensive

Solution: maximize between-class variances instead:

$$\sigma^2_{bcv}(\theta) = \sigma^2 - \sigma^2_{wcv}(\theta)$$

Since the total variance is constant, the effect of changing the threshold is merely to move the contributions of the two terms back and forth. So, minimizing the within-class variance is the same as maximizing the between-class variance. The nice thing about this is that we can compute the quantities in recursively as we run through the range of t values.
Between-class variance is given by:

$$\sigma^2_{bcv}(\theta) = w_F(\theta)w_B(\theta)(\mu_F(\theta) - \mu_B(\theta))^2$$

Otsu's Thresholding

The above procedure is known as Otsu's thresholding. It is formulated as a recursive computation of N_B, N_F, and μ_F, mu_B:

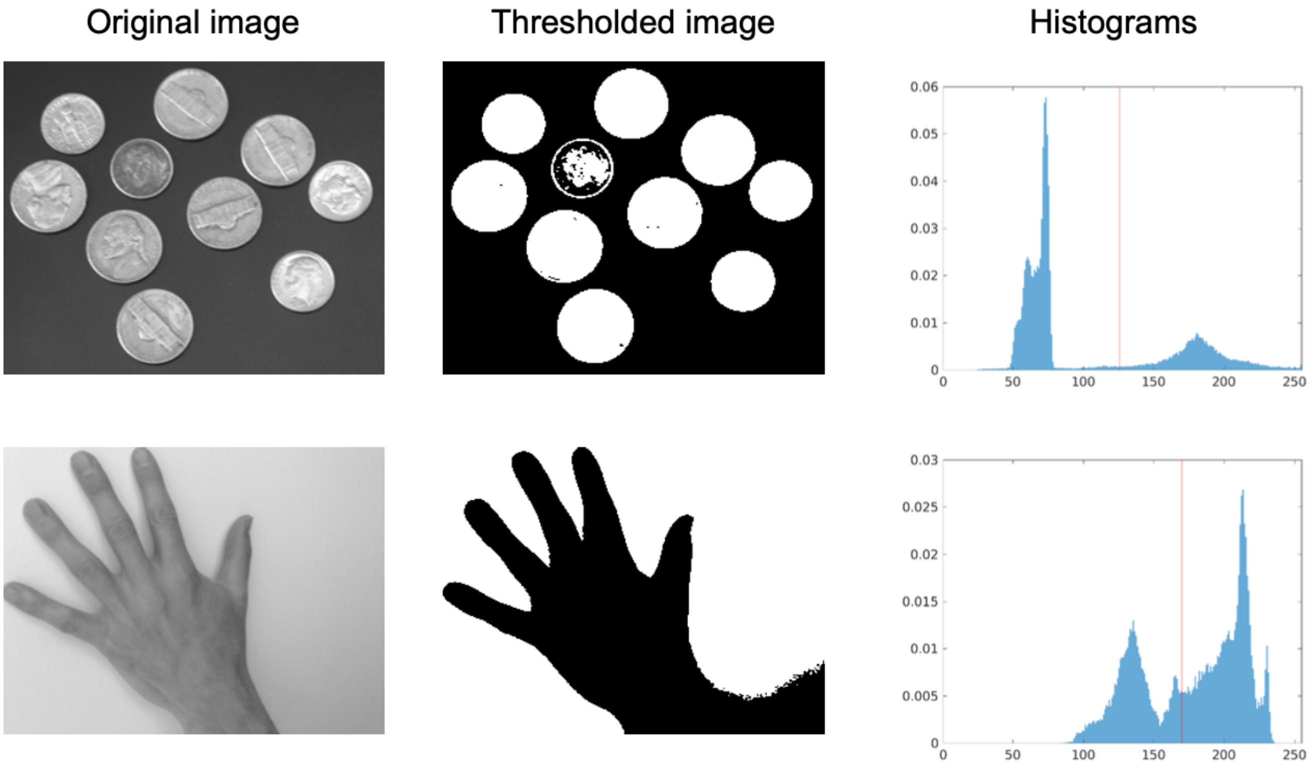
$$N_F(\theta+1) = N_F(\theta) - n_\theta$$

$$N_B(\theta+1) = N_B(\theta) + n_\theta$$

$$\mu_F(\theta+1) = \frac{\mu_F N_F(\theta) - \theta n_\theta}{N_F(\theta+1)}$$

$$\mu_B(\theta+1) = \frac{\mu_B N_B(\theta) - \theta n_\theta}{N_B(\theta+1)}$$

where n_θ is the θ^{th} bin of image histogram(number of pixels with luminance equal to θ)
As θ increases, pixels from background class move to the foreground class.
The histogram X-axis show pixel intensity value (0-255) for grayscale images
Y-axis shows the normalized frequency/probability of each intensity value occurring in the image:



Supervised Thresholding

Suppose we know the number of clusters K with cluster centroids given:

$$c^{(k)} = [c^{(0)}, c^{(1)}, \dots, c^{(K-1)}]$$

For each pixel (m, n), we have L features. Each pixel is assigned the **best fitting** cluster S according to:

$$S[m,n] = \arg\min_k \sum_{l=1}^{L} |f_l[m,n] - c_l^{(k)}|^P = \arg\min_k ||f[m,n] - c^{(h)}||_P$$

If the norm is Euclidean ($P = 2$), we have **nearest neighbor classification**

Chroma Keying

It's a hard-thresholding technique that separates foreground from background based on a predefined **color key**. It's a classification problem where pixels are categorized into foreground or background based on their distance from the chosen key color in a color space (HSV).
Instead of thresholding based on luminance, chroma keying uses predetermined color range for threshold.

K-means Clustering

Remember supervised classification based on known cluster centroids? Now, K-means is all about segmentation with **unknown** cluster centroids.
We only know the **number of clusters**.
Steps:

1. Calculate pixel-cluster distance for each pixel and assign the pixel to closest cluster center
2. Re-compute cluster centers based on the new assignment:

$$c_{new}^{k} = \frac{1}{N_k} \sum_{(m,n) \in \text{cluster k}} f[m,n] \quad k = 0, 1, \ldots, K-1$$

3. Go back to step 1 until centroids do not improve within an epsilon ring.

Application

K-means is important for image quantization for several key reasons:

1. Reduced color/intensity space: instead of using all 256 gray levels (in 8-bit images), K-means reduces them to K representative values
2. Each pixel is assigned to its nearest neighbor, effectively reducing the number of unique intensity values

Bayesian Classification

Goal: minimize Bayes risk $R(\hat{s})$
Cost function is given by $C(s, \hat{s})$. Feature probability $P(f)$ is class-independent
Objective:

$$\arg\min_{S} R(\hat{S}|f) = \arg\min[\int_{S} C(\hat{S}, S) P(S|f) dS$$

The cost function is given by :$C(\hat{S}, S) = 1 - \delta(\hat{S}, S)$ where δ is the Kronecker delta function. This means there's a cost of 1 if when the prediction $\hat{S}$ is wrong, and 0 when correct.

Bayes risk: $R_{MAP}(\hat{S}|f) = 1 - P(\hat{S}|f)$
The expected cost/risk of making prediction $\hat{S}$ given feature f.
Posterior probability: probability of class S given observed feature f. It is usually unknown directly but can be calculated using Bayes' rule:

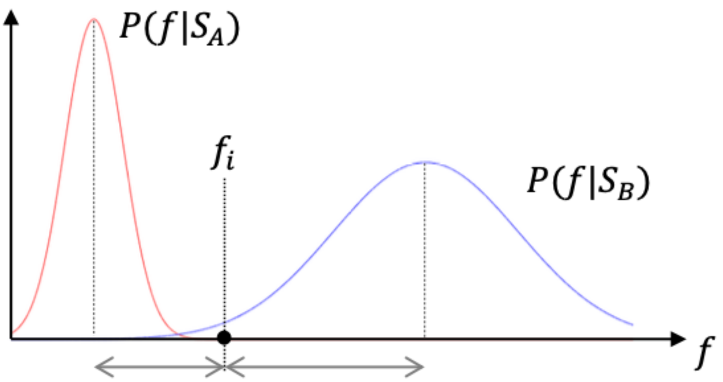
$$P(S|f) = \frac{P(f|S)P(S)}{P(f)}$$

MAP classification: choose the class S that maximizes the posterior probability. This gives the most probable class given the observed features and prior knowledge.
The key advantage of MAP is that it incorporates prior knowledge $P(S)$ about class

distributions, making it more robust than MLE when you have reliable prior information.

$$\hat{S}_{MAP} = \arg\max_S P(S|f) = \arg\max_S [P(f|S)P(S)]$$

Difference between MAP and NN classification
MAP assigns classes based on likelihood while NN considers only the distance to nearest cluster
Example in 1D:



Nearest neighbor assigns class A (it is closer)

MAP assigns class B (it is more likely)

If classes are equally probable where $P(S)$ is constant, then MAP is reduced to Maximum Likelihood(ML) classifier
Disadvantage of classification and clustering methods so far include the fact they only operate on features, ignoring **spatial relation** between pixels.

Region-Based Segmentation

Goal: incorporate knowledge about topological structure of partition.
Region: group of connected pixels with similar properties.
There are two principles:

- similarity: feature differences/variance
- spatial proximity: Euclidean distance, compactness of a region

Basic principle:

- start with seed points
- gradually expand regions by examining neighboring pixels
- add pixels that **meet similarity criteria** to the region
- continue until no more pixels can be added

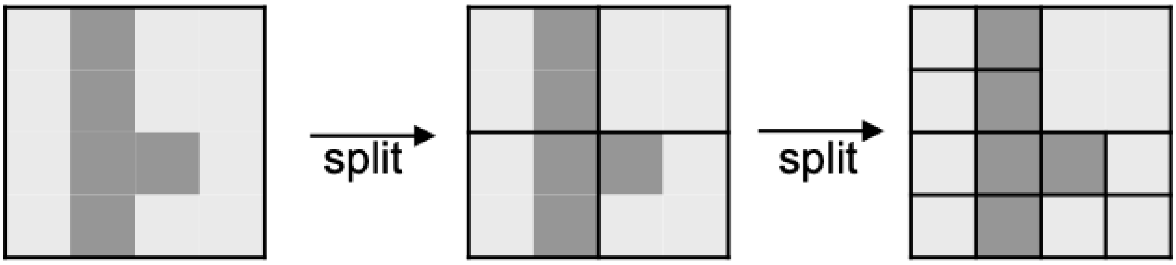
Key components

Seed Point selection

- can be manual or automatic
- critical for segmentation quality
- often chosen based on intensity or location
  There are multiple approaches: single seeded growth, multiple seeded growth, split and merge(region splitting)

Region Splitting:

Split images into disjoint regions by checking each region for homogeneity, if not homogeneous keep splitting
Quad-tree decomposition:



→ top-down segmentation

In top-right, all pixels are homogeneous since they have the same gray levels.
Problems

- how to optimally split a region into homogeneous sub-regions?
- requires knowledge about number of sub-regions and location of region boundaries(by edge detection)

Split&Merge

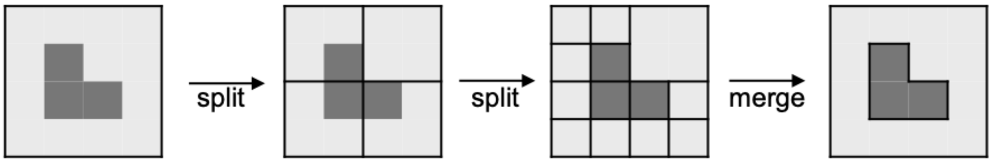
A better option is to use split and merge method which combines agglomerative and divisive region operations

Splitting (e.g. quad-tree)

- Keep splitting until all blocks fulfill homogeneity criterion

Merging

- Merge all neighboring block which are sufficiently similar



Disadvantage: region borders often exhibit "staircase" character

Similarity Criteria

There are multiple similarity measures possible:

- Absolute deviation of mean value: $d(R_i, R_j) = |\mu(R_i) - \mu(R_j)|$ simple to calculate, but does not account for region variances
- Variance coherence: extensible to higher order statistics
- Likelihood ratio: consider region size N

Temporal Segmentation of Video

Scene Cut assumption: assumes that scene transitions in videos are abrupt/discontinuous.
Shot detection: detect scene cuts and segment video into a number of temporally **consistent** video sequences(shots)
Method: analyze sum of absolute histogram differences between subsequent video frames. A scene cut is detected if the sum exceeds hard threshold T_H or remains above soft threshold T_s for a certain number of frames.