DIP Notes

Thresholding

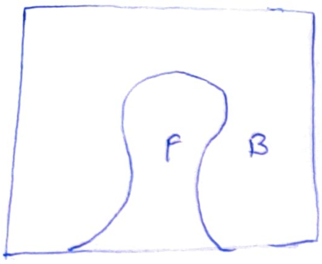
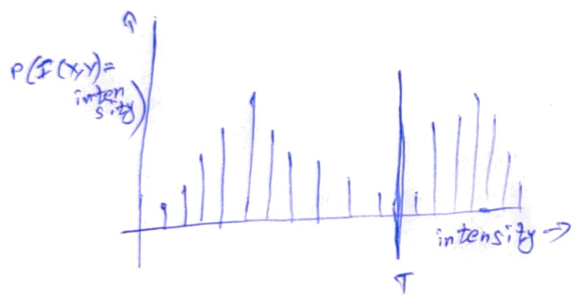


Image segmentation

Easiest method:

$$g(x,y)=\begin{cases}1, & \text{if } I(x,y) > T \quad \text{object (foreground)} \\ 0, & \text{if } I(x,y) \le T \quad \text{background}\end{cases}$$



Histogram

How to choose T?

T(threshold)

Global threshold for all pixels

vs

local / adaptive / pixel dependent threshold

Global Thresholding

assume we use the same T for the whole image.

How to find the "best" T?

classical method: OTSU's Algorithm

idea: Maximize between-class variance.

A good threshold should separate pixels into tight clusters

Image PMF:

p_i = probability that $I(x,y) = i$

$i = 0, 1, \ldots, L-1$

m_G (global mean) $= \sum_{i=0}^{L-1} i\, p_i$

σ_G^2 (global variance) $= \sum_{i=0}^{L-1} (i - m_G)^2 p_i$

Suppose we select a Threshold T.

$$C_1 = \{(x,y) \mid I(x,y) \le k\}$$
$$C_2 = \{(x,y) \mid I(x,y) > k\}$$

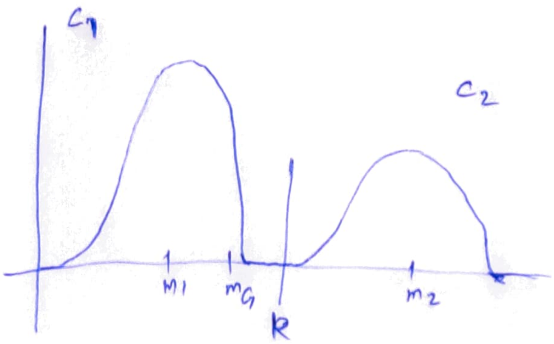
$$P_1 = \sum_{i=0}^{k} P_i \qquad P_2 = \sum_{i=k+1}^{L-1} P_i \qquad P_1 + P_2 = 1$$

Class-conditional mean/variance

$$m_1 = \frac{\sum_{i=0}^{k} i P_i}{P_1} \qquad m_2 = \frac{\sum_{i=k+1}^{L-1} i P_i}{P_2}$$

OTSU's CRITERION: Maximize the between-class variance:

$$\sigma_B^2 = P_1 (m_1 - m_G)^2 + P_2 (m_2 - m_G)^2$$



The ratio σ_B^2 / σ_G^2 is a good measure of separability. Higher is better (more separable).

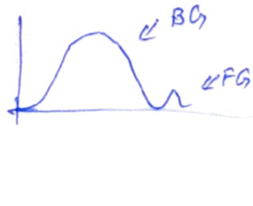
In practice, we just consider all possible k, and choose T as the k that maximizes σ_B^2.

In general, we can extend this to Finding $k-1$ threshold to separate K classes.

$$\sigma_B^2 = \sum_{k=1}^{K} P_k (m_k - m_G)^2$$

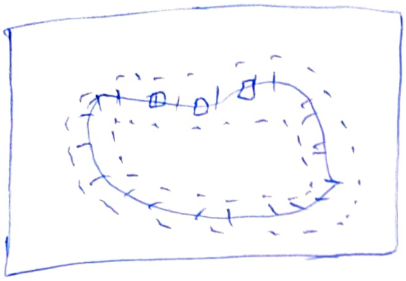
Otsu can fail when:

- no strong peaks in histogram
- object is small w.r.t. background



Remedies:

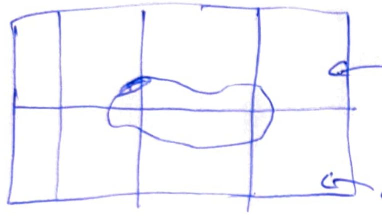
- low-pass filter, then apply otsu
- only consider pixels near edges when computing the threshold.



~~Consider the~~

Only consider the pixels near edges when computing threshold.

Variable/Adaptive Thresholding

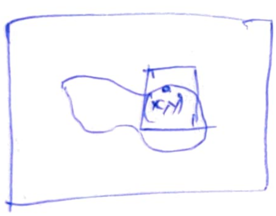


blockproc (in Matlab) - Applies a specied function to each $M \times N$ block of an image

apply otsu to each block

works ok in cases, but choosing block size is tricky & blocking artifacts.

Better: Adapt threshold on a per pixel basis.



At every (x,y), build neighborhood S_{xy}.

Compute μ_{xy}, σ_{xy}.

We could make rules like:

$$g(x,y) = \begin{cases} 1 & I(x,y) > \mu_{xy} + 2\sigma_{xy} \\ 0 & \text{else} \end{cases}$$

$$\text{or } \begin{cases} 1 & I(x,y) > \mu_{xy} \\ 0 & \text{else} \end{cases}$$

$$\text{or } \begin{cases} 1 & |I(x,y) - \mu_{xy}| > 2\sigma_{xy} \\ 0 & \text{else} \end{cases}$$

$$\text{or } \begin{cases} 1 & I(x,y) > \mu_{xy} + \sigma_{xy} \text{ AND } I(x,y) > T_{min}. \\ 0 & \text{else} \end{cases}$$

Can also apply Thresholding to RGB colors:

- Threshold independently on R, G, B, I channels
- combine the channels, eg: $\|I(x,y) - c\| < T$ $\quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \begin{bmatrix} R_c \\ R_G \\ R_B \end{bmatrix}$