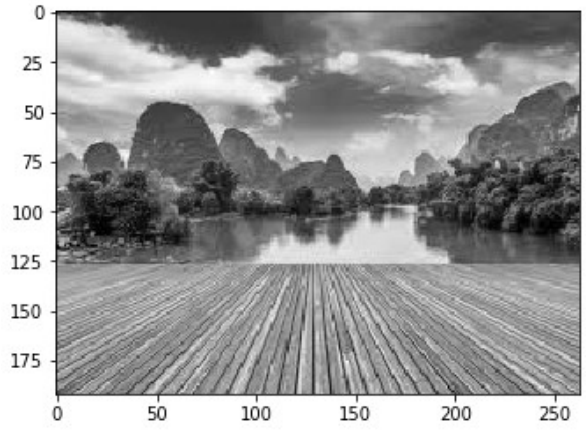
Image segmentation

The goal of image segmentation is to cluster pixels into salient image regions.

The simplest method of image segmentation is called the thresholding method.



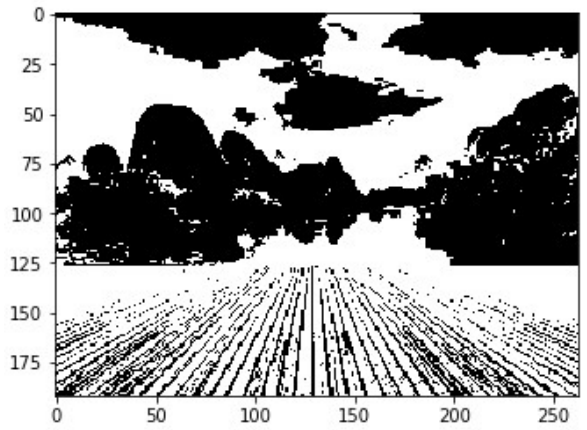


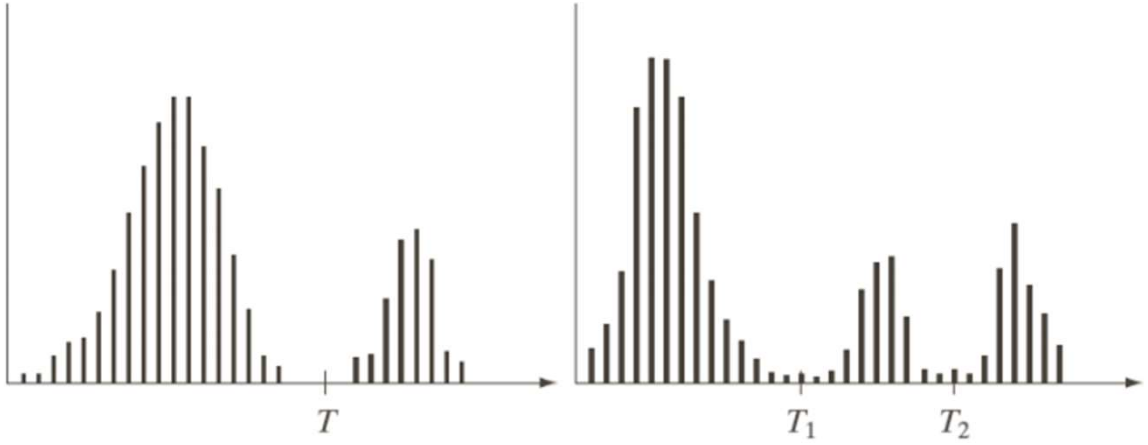
Image segmentation

The pixels are partitioned depending on their intensity value.

Variable thresholding, if T can change over the image.

$$g(x, y) = \begin{cases} a, & \text{if } f(x, y) > T_2 \\ b, & \text{if } T_1 < f(x, y) \leq T_2 \\ c, & \text{if } f(x, y) \leq T_1 \end{cases}$$

Peaks and valleys of the image histogram can help in choosing the value for the threshold.



Otsu's method

Find the threshold that minimizes the weighted within-class variance.
within-class variance, defined as a weighted sum of variances of the two classes:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

This turns out to be the same as maximizing the between-class variance

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$$

Step through all possible thresholds t=1, … maximum intensity, compute $\sigma_b^2(t)$

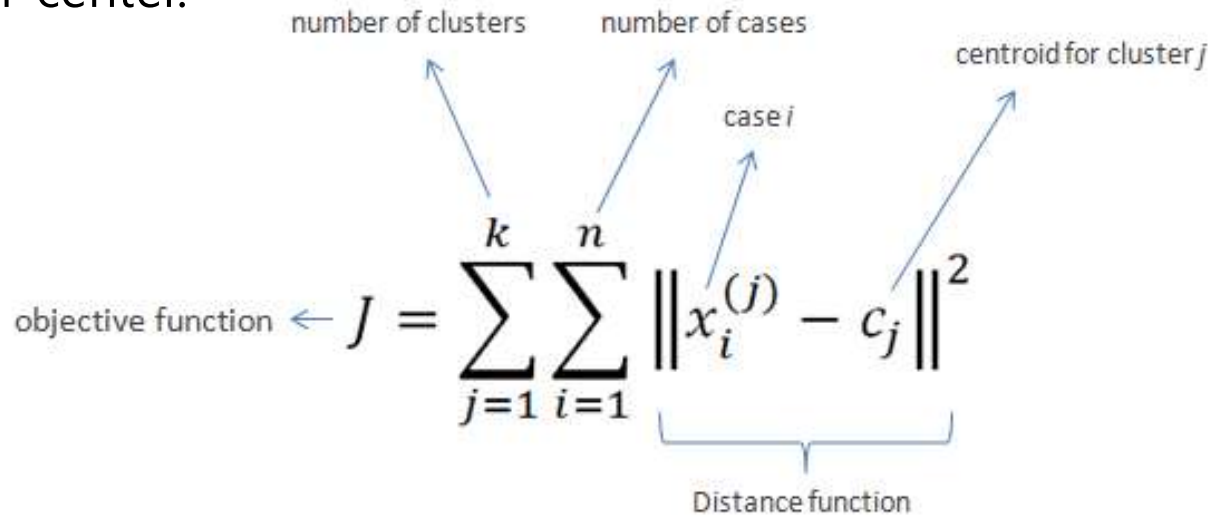
Desired threshold corresponds to the maximum $\sigma_b^2(t)$

K-means clustering

K-Means clustering algorithm is an unsupervised algorithm and it is used to segment the interest area from the background.

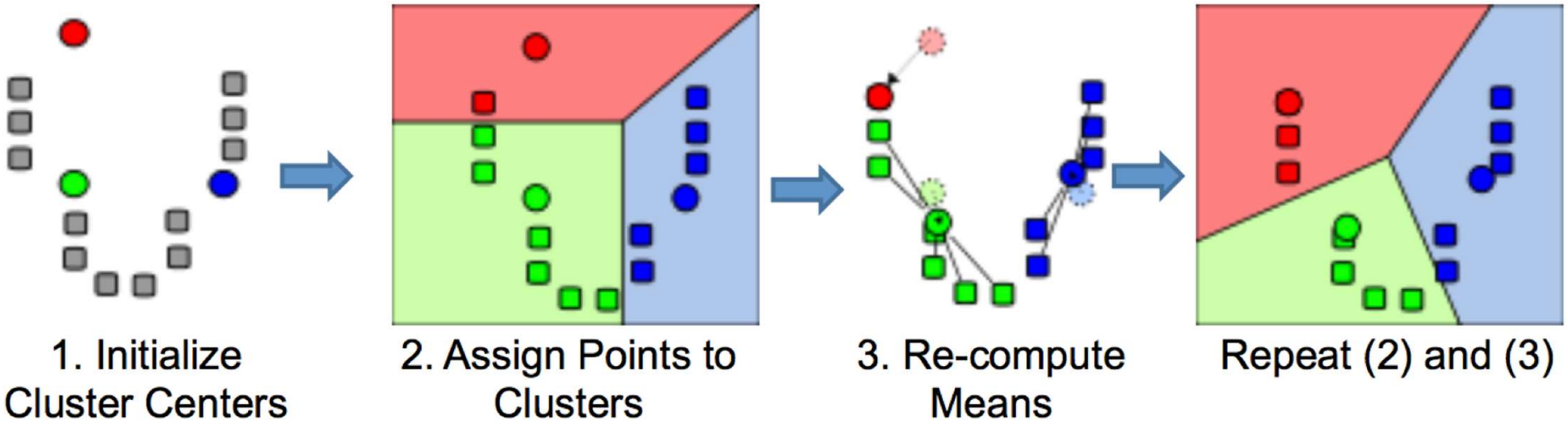
It partitions the given data into K-clusters based on similarity of the data.

The objective of K-Means clustering is to minimize the sum of squared distances between all points and the cluster center.



K-means clustering

1. Initialize K clusters randomly.
2. Assign each pixel to the closest center.
3. Update cluster centers by computing the average of the pixels in the cluster.
4. Repeat steps 2 and 3 until no pixels change cluster centers.

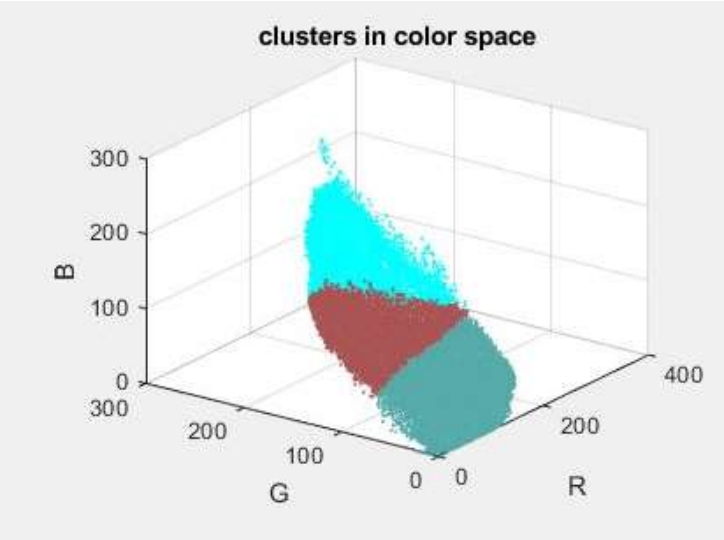


K-means clustering

K-means finds K cluster centers that are a good representation of the data.

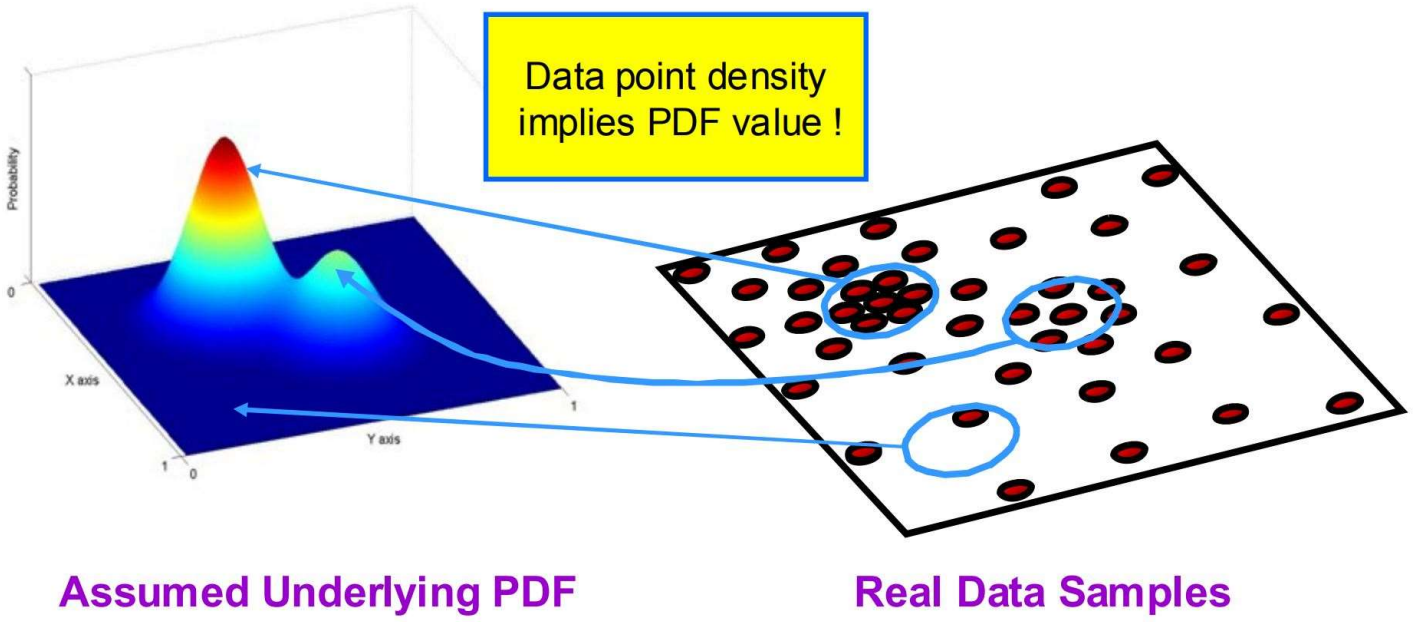
It is prone to being affected by outliers and can be slow in runtime.

It only considers color similarity, not the distance of pixels.



Mean-shift

Find local maxima of the probability density function given by samples.

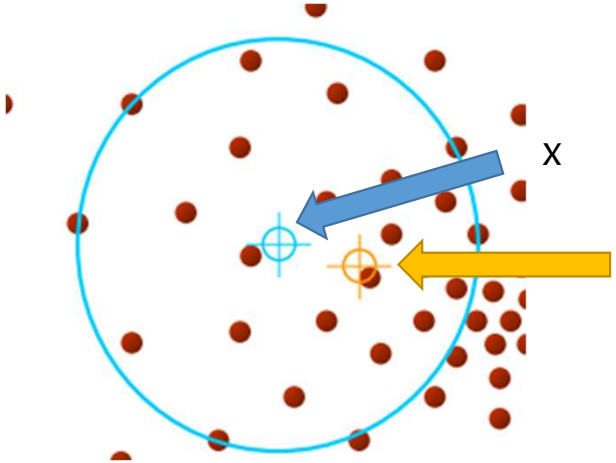


Mean-shift

1. Start with a random region of points.

2. The weighted mean of the density in the window determined by K is

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

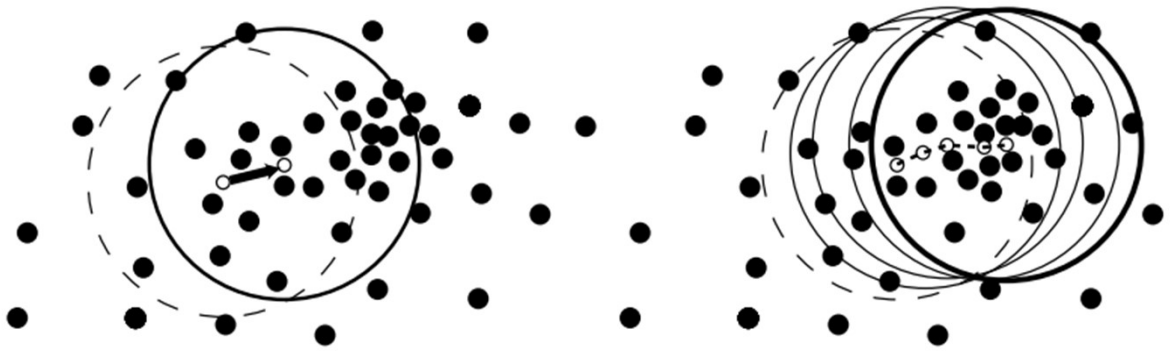


m(x) = weighted mean ≈ center of mass

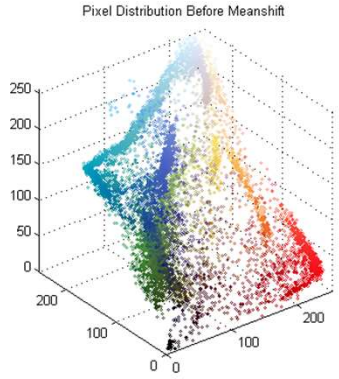
Mean-shift

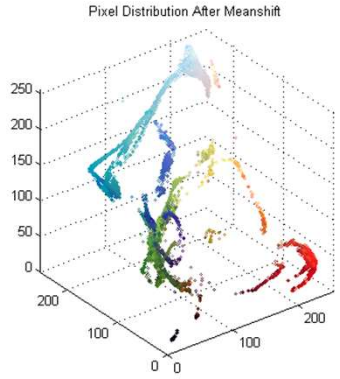
3. Do $x \leftarrow m(x)$, the difference $m(x) - x$ is called mean-shift.

4. Repeat the estimation until $m(x)$ converges.



Mean-shift





Map pixels into 5-dimension(r, g, b, x, y) space.

Contrary to K means, mean shift is robust to outliers.

The output depends only on window size, and mean shift can be computationally expensive.

Graph cut

Map the image into a weighted undirected graph.

Node (vertex) for every pixel

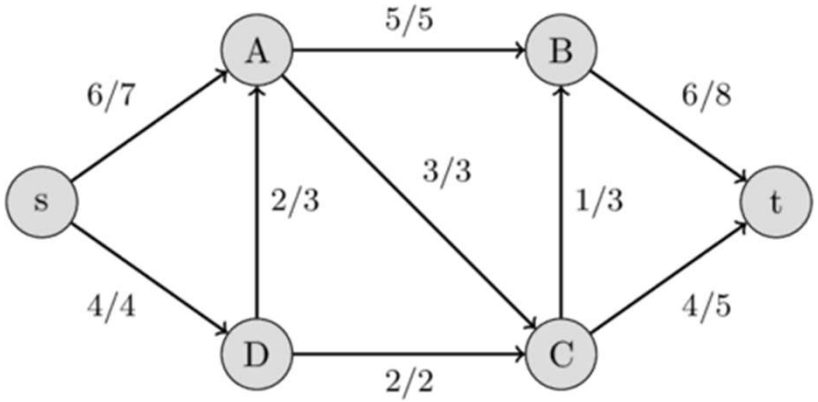
Edge between pairs of pixels

Affinity weight for each edge means pixel similarity

regard the image segmentation problem as the node division

Network Flow

Two vertices in the network flow are designated to be the source vertex **s** and the sink vertex **t**, respectively.



The flow can be sent through some path from s to t.

An **s / t** cut is a partitioning of the vertices into two disjoint subsets such that one contains **s** and the other contains **t**

Network Flow

capacity constraint:

- each edge has a capacity that must be greater than or equal to its flow

flow conservation:

- inflow of the node is equal to its outflow

Network Flow

If we divide the graph by cut, net flow between two graphs
is equivalent to the cut capacity.

We can find the cut which capacity is equivalent to the maximum flow

As stated by the max-flow min-cut theorem, the **maximum amount of flow** is equivalent to the **net flow of the edges in the minimum cut**.

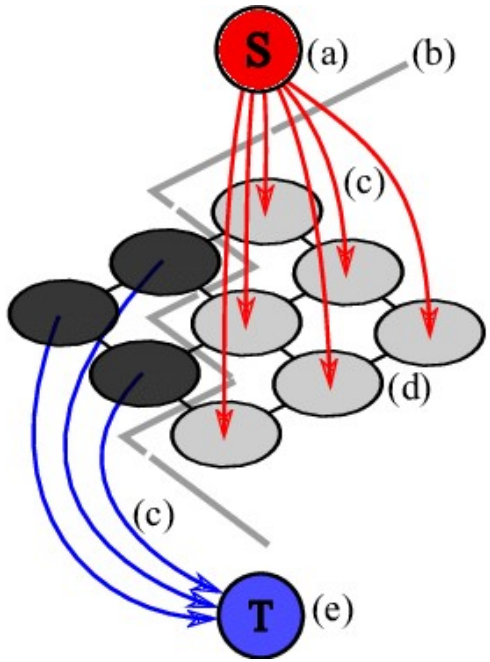
Network Flow

A pixel in the image is a vertex in the graph. These vertices are called pixel vertices.

In addition, there are two extra vertices that serve as the source and the sink.

There are two types of edges:

- The first type connects neighboring pixel vertices in a 4-neighboring system.
- The second type connects the source or sink vertex with the pixel vertices.



Network Flow

The weight of an edge should be big with the two pixels are similar, and small when they are quite different.

Let I_p be the brightness, or intensity, of the pixel vertex p

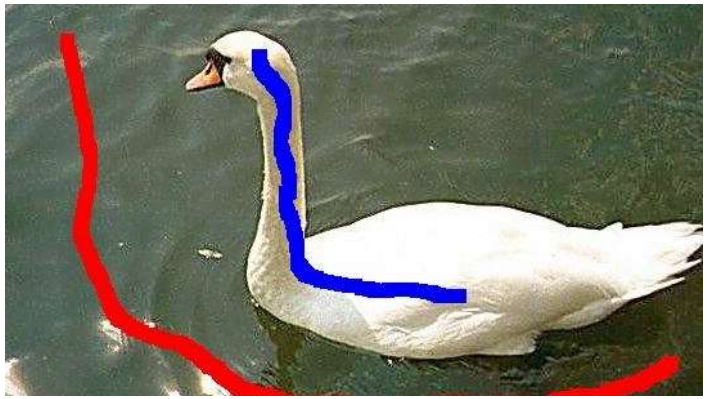
$$B(I_p, I_q) = 100 \cdot \exp\left(\frac{-(I_p - I_q)^2}{2\sigma^2}\right)$$

Network Flow

The user is prompted to highlight at least one pixel vertex as a background pixel and at least one as a foreground pixel. These pixel vertices are called seeds.

For every background seed, an edge is added from the source to the background seed with maximum capacity.

In a similar fashion, edges to the sink are added for every foreground seeds with maximum capacity.



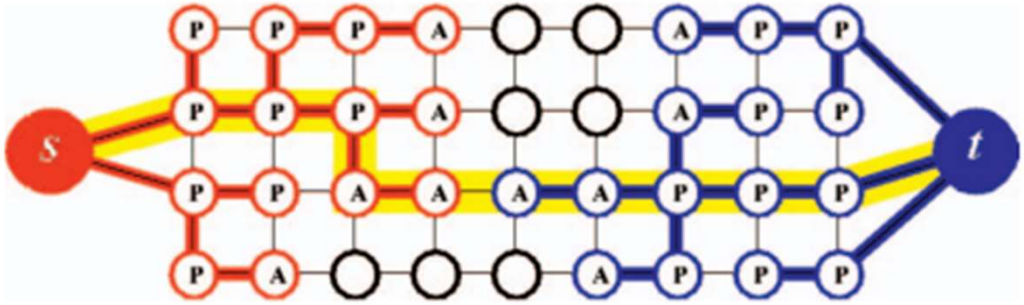
Network Flow

Run a graph cut algorithm to find the minimum cut.

ex) Edmonds-Karp algorithm and push-relabel algorithm.

The active nodes explore adjacent non-saturated edges and acquire new children from a set of free nodes.

This terminates when S and T cannot grow and the trees are separated.



Network Flow