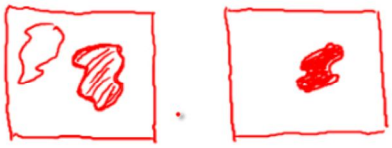
Introduction

- Task

  Divide an image into constituent "meaningful" regions

- Challenges

  Image data ambiguity, noise, integration of high level information, lack of ground truth



- Approaches based on

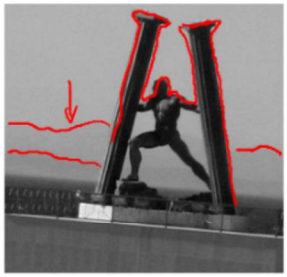
  intensity discontinuity, intensity similarity, morphology

Agenda

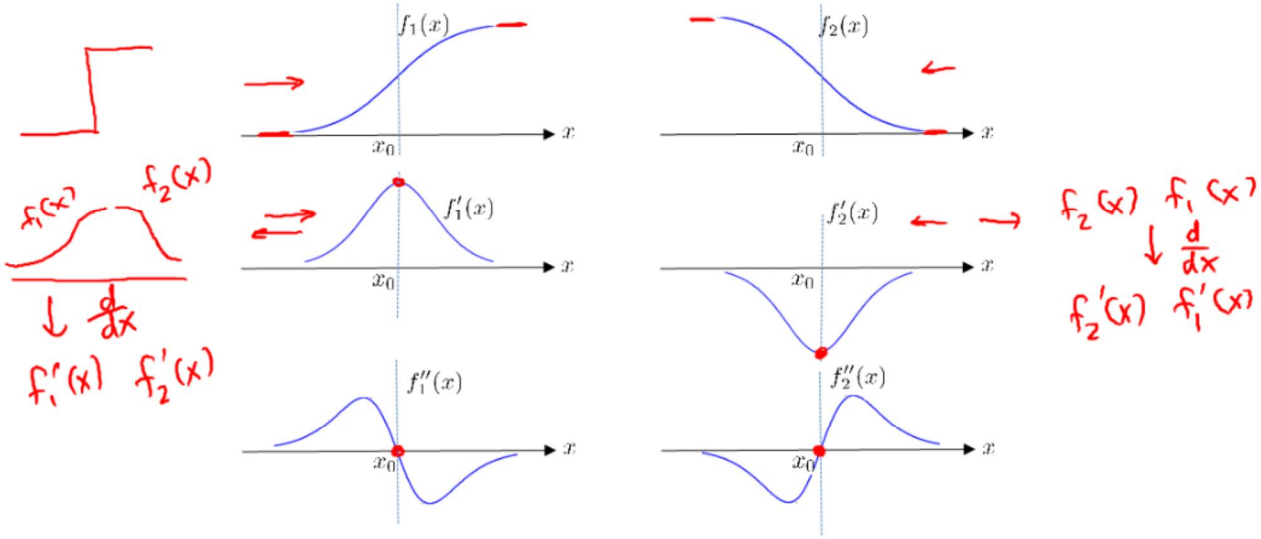
- Methods based on intensity discontinuity
  - Edge-based segmentation
- Methods based on intensity similarity
  - Thresholding
  - Region growing
  - Region splitting and merging
- Watersheds and K-means algorithms
- Advanced methods

Edge Detection

- Gradient-Based Approaches
  - Sobel
  - Prewitt
  - Roberts
- Laplacian-Based Approaches
  - Marr-Hildreth algorithm
- Canny Algorithm



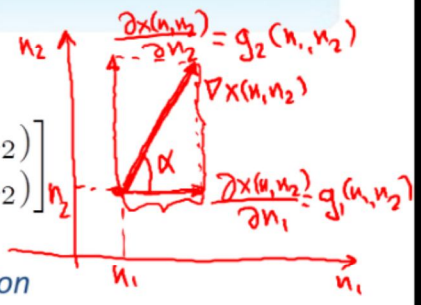
1st and 2nd Order Derivatives



The Image Gradient

Gradient operator

$$g_x(n_1,n_2) = \nabla x(n_1,n_2) = \begin{bmatrix} \frac{\partial x(n_1,n_2)}{\partial n_1} \\ \frac{\partial x(n_1,n_2)}{\partial n_2} \end{bmatrix} = \begin{bmatrix} g_1(n_1,n_2) \\ g_2(n_1,n_2) \end{bmatrix}$$



The Gradient magnitude is commonly used for edge detection

$$M(n_1,n_2) = |g_x(n_1,n_2)| = \left(\left(\frac{\partial x(n_1,n_2)}{\partial n_1} \right)^2 + \left(\frac{\partial x(n_1,n_2)}{\partial n_2} \right)^2 \right)^{\frac{1}{2}}$$

The direction of the gradient vector is given by

$$\alpha(n_1,n_2) = \tan^{-1}\left[\frac{g_2(n_1,n_2)}{g_1(n_1,n_2)} \right]$$

Gradient Evaluation

- How do we find the gradient?
- To approximate partial derivatives we use neighborhood differences.

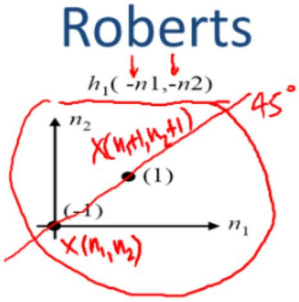
$$\frac{\partial x(n_1,n_2)}{\partial n_1} = g_1(n_1,n_2) = x(n_1,n_2) * *h_1(n_1,n_2) \leftarrow ?$$

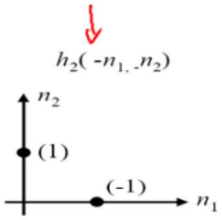
$$g_2(n_1,n_2) = x(n_1,n_2) * *h_2(n_1,n_2) \leftarrow ?$$

$$M(n_1,n_2) = |g(n_1,n_2)| = \sqrt{g_1(n_1,n_2)^2 + g_2(n_1,n_2)^2}$$

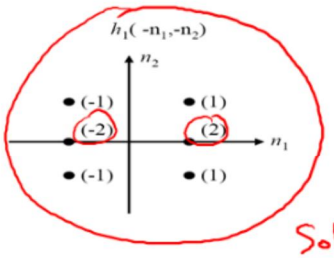
Gradient Kernels

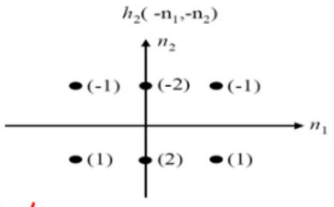
Roberts



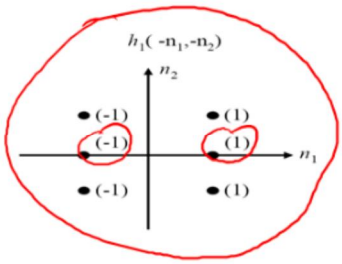


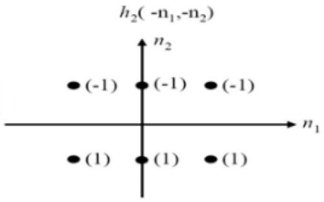
Sobel

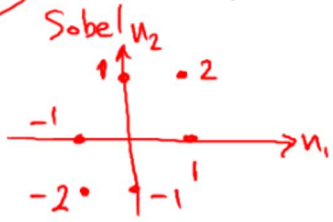




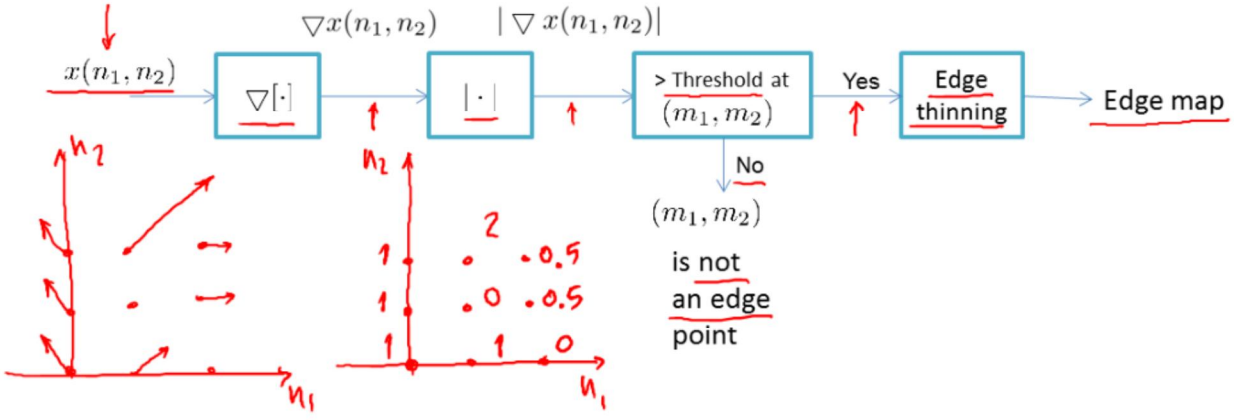
Prewitt







System for Edge Detection



Example

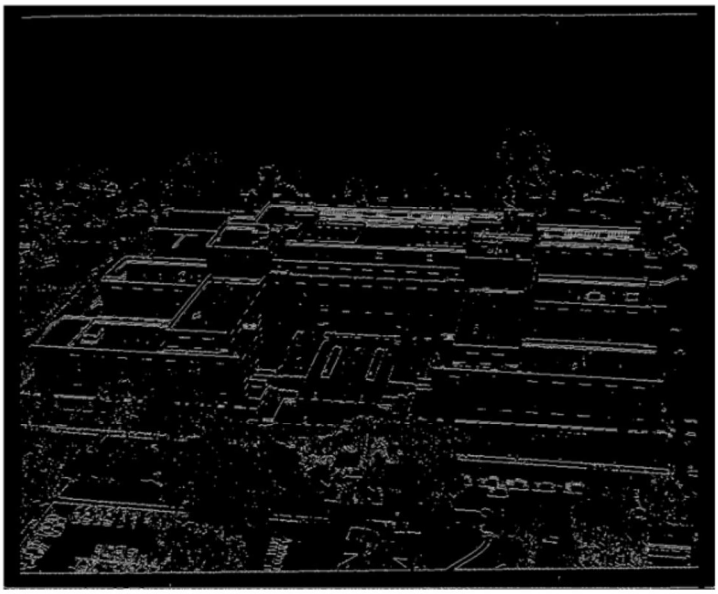
Technological Institute
Northwestern University
1942

Example

$** h_1(n_1, n_2)$
=
Sobel

Technological Institute
Northwestern University
1942

Example



Technological Institute
Northwestern University
1942

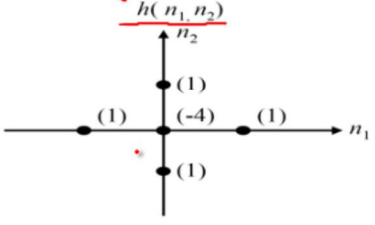
Calculation of the Laplacian

$$\nabla^2 x(n_1,n_2) = \frac{\partial^2 x(n_1,n_2)}{\partial n_1^2} + \frac{\partial^2 x(n_1,n_2)}{\partial n_2^2}$$

$$L(n_1,n_2) = x(n_1,n_2) * *h_1(n_1,n_2) * *h_1'(n_1,n_2) + x(n_1,n_2) * *h_2(n_1,n_2) * *h_2'(n_1,n_2) +$$

$$L(n_1,n_2) = x(n_1,n_2) * *\big(h_1(n_1,n_2) * *h_1'(n_1,n_2) + h_2(n_1,n_2) * *h_2'(n_1,n_2)\big)$$

$$L(n_1,n_2) = x(n_1,n_2) * *h(n_1,n_2)$$



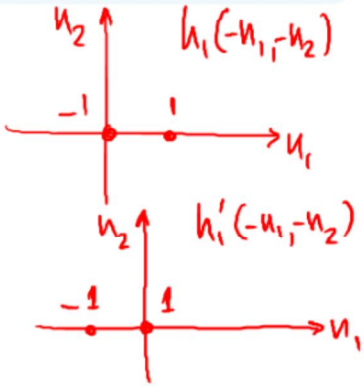
Calculation of the Laplacian

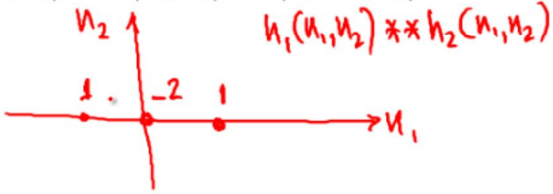
- Horizontal second derivative

$$\frac{\partial x(n_1,n_2)}{\partial n_1} = g_1(n_1,n_2) = x(n_1+1,n_2) - x(n_1,n_2)$$

$$\frac{\partial^2 x(n_1,n_2)}{\partial n_1^2} = \frac{\partial}{\partial n_1}\left(\frac{\partial x(n_1,n_2}{\partial n_1}\right) = g_1(n_1,n_2) - g_1(n_1-1,n_2)$$

$$= x(n_1+1,n_2) - 2x(n_1,n_2) + x(n_1-1,n_2)$$





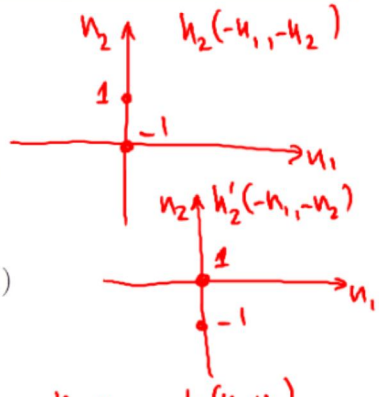
Calculation of the Laplacian

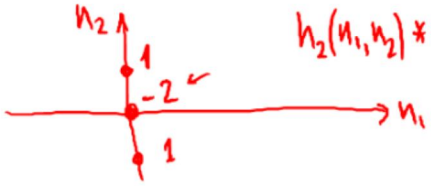
- Vertical second derivative

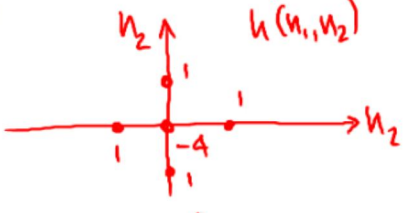
$$\frac{\partial x(n_1,n_2)}{\partial n_2} = g_2(n_1,n_2) = x(n_1,n_2+1) - x(n_1,n_2)$$

$$\frac{\partial^2 x(n_1,n_2)}{\partial n_2^2} = \frac{\partial}{\partial n_2}\left(\frac{\partial x(n_1,n_2}{\partial n_2}\right) = g_2(n_1,n_2) - g_2(n_1,n_2-1)$$

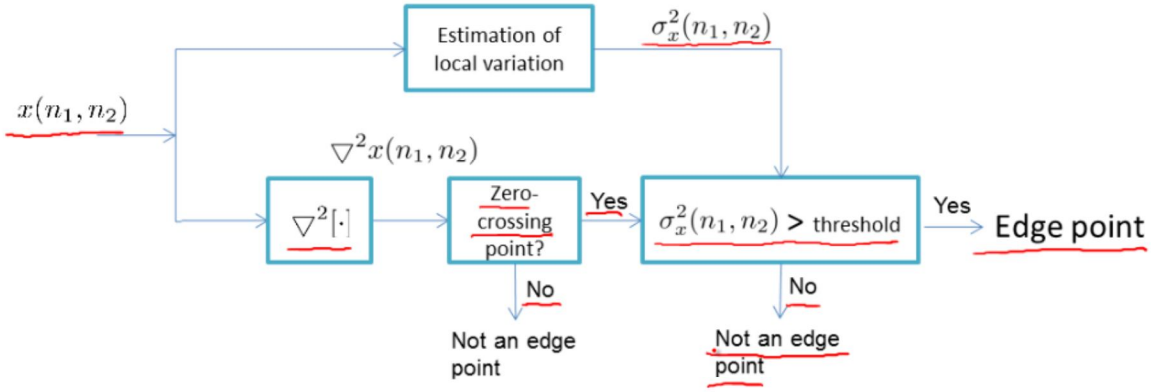
$$= x(n_1,n_2+1) - 2x(n_1,n_2) + x(n_1,n_2-1)$$



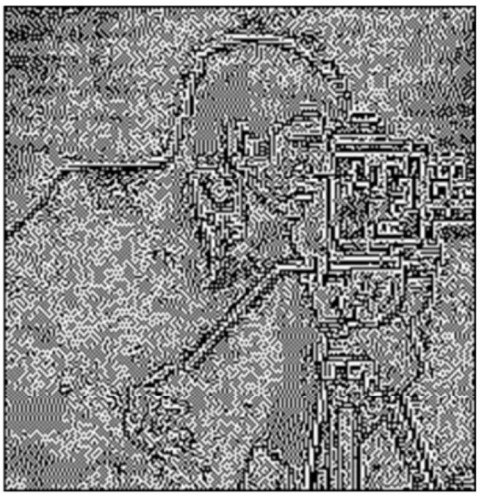




Laplacian-Based Edge Detection System



Example



Laplacian of Gaussian (LOG) Edge Detector

- Intensity changes occur at different scales
- Image is band-limited before edge detection
- Gaussian LPF

$$h(x,y) = e^{-\frac{(x^2+y^2)}{2\pi\sigma^2}}$$

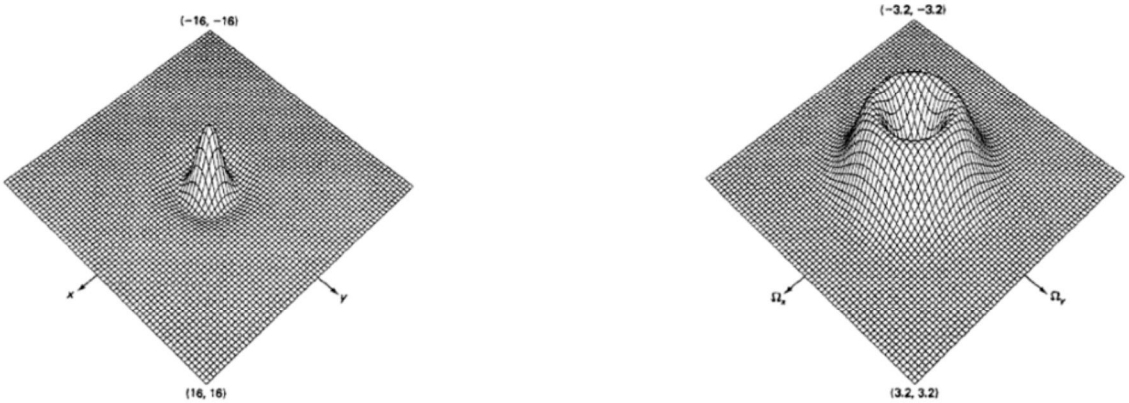
- Taking the Laplacian

$$\nabla^2 \left(f(x,y) * *h(x,y)\right) = f(x,y) * * \left[\nabla^2 h(x,y)\right]$$

LOG

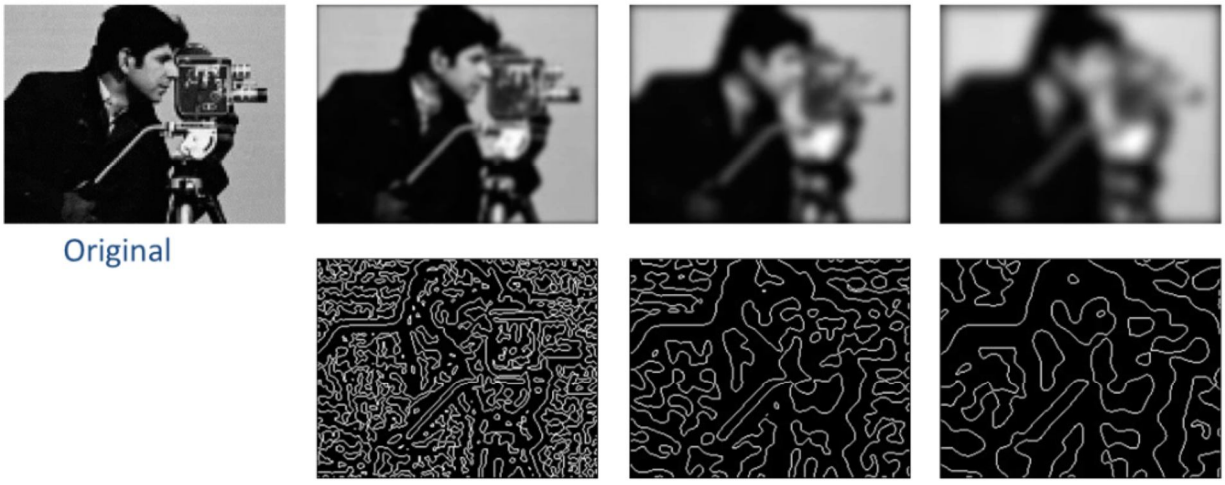
Laplacian of Gaussian (LOG) Edge Detector

$$\nabla^2 h(x,y) = \frac{e^{-\frac{(x^2+y^2)}{2\pi\sigma^2}}}{(\pi\sigma^2)^2}(x^2+y^2-2\pi\sigma^2) \qquad -FT\{\nabla^2 h(x,y)\} = 2\pi^2\sigma^2 \cdot (\Omega_x^2+\Omega_y^2) \cdot e^{-\pi\sigma^2\frac{(\Omega_x^2+\Omega_y^2)}{2}}$$



J. S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice Hall, 1990.

Examples



The Canny Edge Detector

- ✓ Smooth the image with a Gaussian filter (σ)
- ✓ Compute the gradient magnitude and angle images
- Apply non-maximal suppression to the gradient magnitude image
- Use double thresholding (T_H, T_L)

→ J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, No. 6, pp. 679–698, 1986.

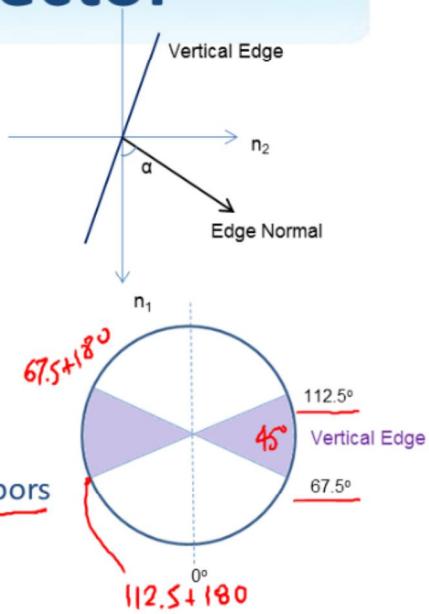
The Canny Edge Detector

Non-maximal suppression

Let d_1, d_2, d_3, d_4 denote the four basic edge directions: horizontal, -45°, +45°, vertical.

At every point (n_1,n_2) in $\alpha(n_1,n_2)$:

1. Find the direction d_k that is closest to $\alpha(n_1,n_2)$;
2. If the value of $M(n_1,n_2)$ is less than at least one of its neighbors along d_k, ignore/suppress it; otherwise, keep it.



The Canny Edge Detector

Double thresholding ($T_H > T_L$)

Pixels with values greater than T_H are "strong" edge pixels

Pixels with values smaller than T_L are "weak" edge pixels

Edge Linking

Locate a pixel p belonging to a "strong" edge;

Find "weak" pixels that are connected to p;

append these "weak" pixels to p.

Example

Sobel

LOG

Canny

The Hough Transform

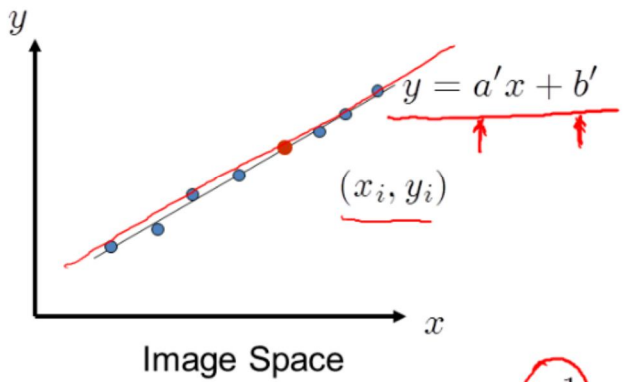
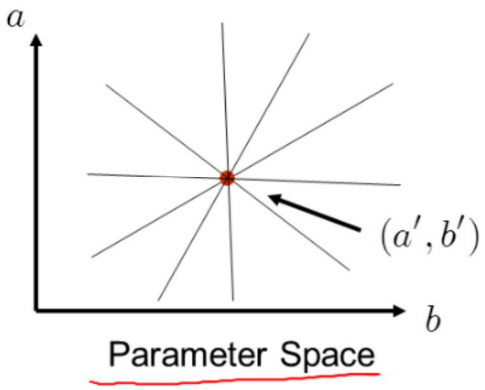


Image Space

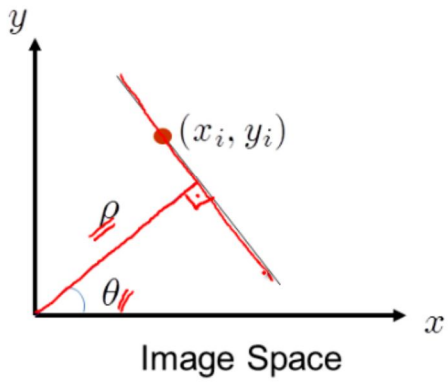


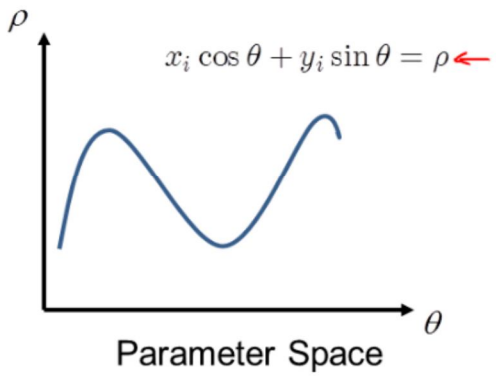
Parameter Space

$$a = -\frac{1}{x}b + \frac{y}{x}$$

slope a-intercept

The Hough Transform

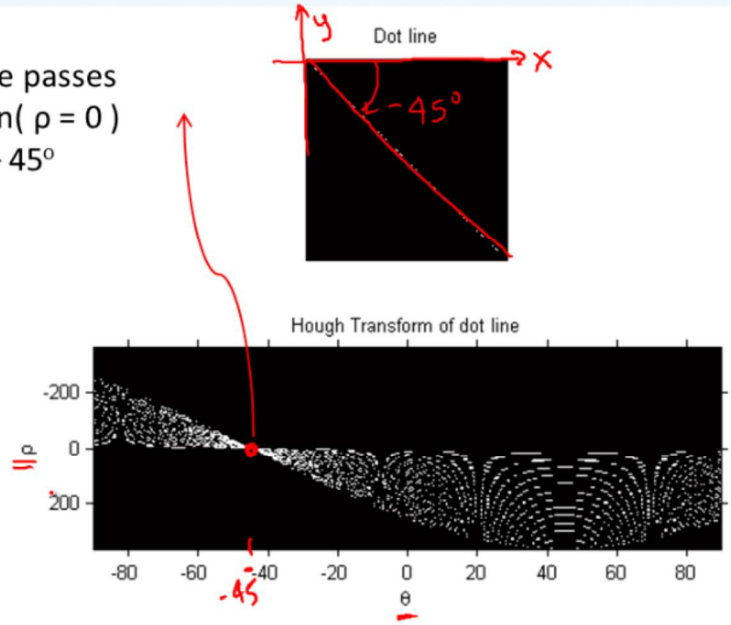




$$x\cos\theta + y\sin\theta = \rho$$

The Hough Transform

The dominant line passes through the origin(ρ = 0) and oriented at – 45°



The Hough Transform

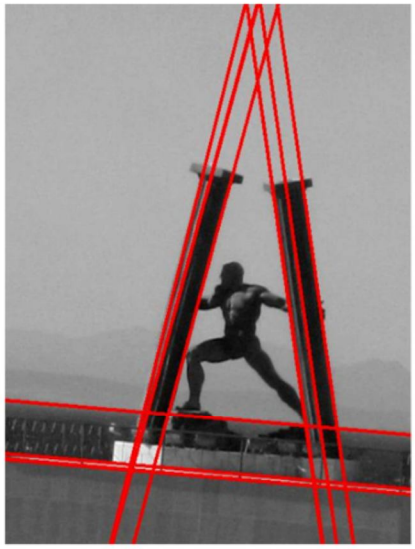
The Procedure:

- Obtain a binary edge image using any techniques (such as the ones discussed);
- Quantify the ρΘ-plane;
- Examine the number of intersects at each cell in the ρΘ-plane;
- Bridge the gap based on continuity*

**The gap in a line associated with a given cell is bridged if the length of the gap is less than a threshold.*

Example

$\rho\theta - \text{plane}$



Thresholding

- Two class case

$$x(n_1, n_2) = \begin{cases} \text{object} & x(n_1, n_2) \geq T \\ \text{background} & x(n_1, n_2) < T \end{cases}$$

- Approach can be easily applied when multiple thresholds are available
- The key of the thresholding process is the choice of the threshold value(s)

Example

Threshold T = 0

$$x(n_1, n_2) = \begin{cases} 1 & x(n_1, n_2) \geq T \\ 0 & x(n_1, n_2) < T \end{cases}$$

Example

Threshold T = 10

$$x(n_1, n_2) = \begin{cases} 1 & x(n_1, n_2) \geq T \\ 0 & x(n_1, n_2) < T \end{cases}$$

Example

Threshold T = 25

$$x(n_1, n_2) = \begin{cases} 1 & x(n_1, n_2) \geq T \\ 0 & x(n_1, n_2) < T \end{cases}$$

Example

Threshold T = 50

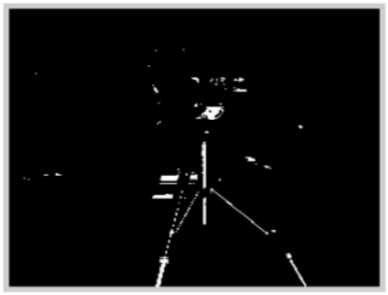
$$x(n_1, n_2) = \begin{cases} 1 & x(n_1, n_2) \geq T \\ 0 & x(n_1, n_2) < T \end{cases}$$

Example

Threshold T = 150

$$x(n_1, n_2) = \begin{cases} 1 & x(n_1, n_2) \geq T \\ 0 & x(n_1, n_2) < T \end{cases}$$

Example



Threshold T = 250

$$x(n_1, n_2) = \begin{cases} 1 & x(n_1, n_2) \geq T \\ 0 & x(n_1, n_2) < T \end{cases}$$

Otsu's Method

- ✓ Automatically finds the optimal global thresholding maximizing the between-class variance
- ✓ Only the histogram of the image is used

N. Otsu, "A threshold selection method from gray-level histograms", *Automatica*, 11(285-296):23–27, 1975.

Otsu's Method

- Compute the normalized histogram of the input image. Denote the components of the histogram by $p_i,\ i = 0, 1, 2, ..., L-1$
- Suppose a threshold is selected $k,\ 0 < k < L-1$
- C_1 is the set of pixels with levels $[0, 1, 2, \cdots, k]$
- C_2 is the set of pixels with levels $[k+1 \cdots, L-1]$

- Obtain the value of the threshold which maximizes the *between class variance*

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

Otsu's Method

- $P_1(k)$ is the probability of the set C_1 occurring

$$P_1(k) = \sum_{i=0}^{k} p_i, \quad k = 0, 1, 2, ..., L-1$$

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k), \quad k = 0, 1, 2, ..., L-1$$

- $m_1(k)$ and $m_2(k)$ are the mean intensities of C_1 and C_2, respectively
- The mean intensity up to level k is given by

$$m(k) = \sum_{i=0}^{k} i p_i, k = 0, 1, 2, ..., L-1$$

- Global intensity mean $\quad m_G = \sum_{i=0}^{L-1} i p_i$

Otsu's Method

- Substituting the expressions from the previous slide we obtain

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}, \quad k = 0, 1, 2, ..., L-1$$

- Only two parameters have to be computed for all values of k
- The Otsu threshold k^* is the value of k which maximizes the *between-class variance* above.

Otsu's Method Example

Region-Based Segmentation

- Basic Formulation: Let R represent the entire image region. Segmentation is the process of partitioning R into subregions $R_1, R_2, \cdots, R_n$ so that

$$\bigcup_{i=1}^{n} R_i = R$$

R_i is a connected region, $i = 1, 2, \cdots, n$

$R_i \bigcap R_j = \emptyset$ for all i and $j, i \neq j$

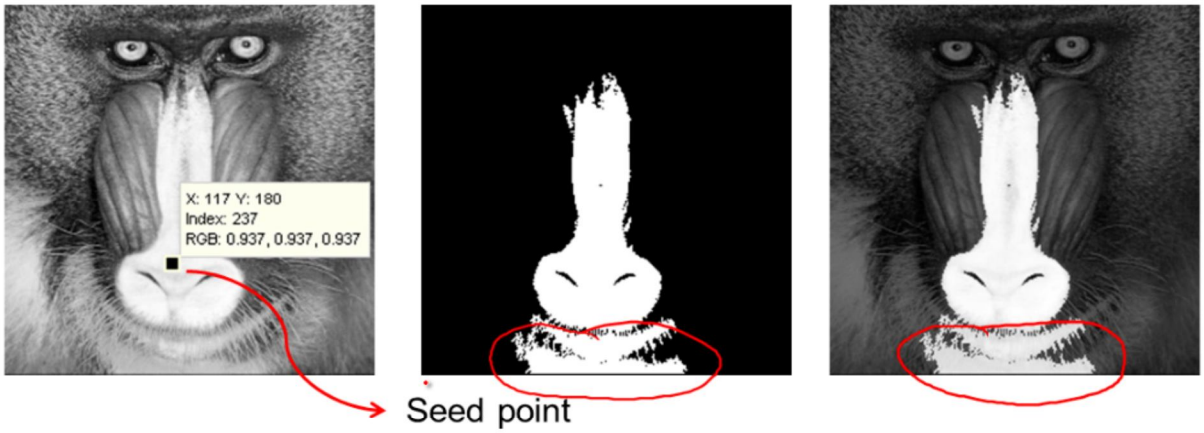
$P(R_i) = \text{TRUE}$ for $i = 1, 2, \cdots, n$

$P(R_i \cup R_j) = \text{FALSE}$ for $i = 1, 2, \cdots, n$

Region Growing

- A set of "seed" points is selected
  - Based on prior information
  - Based on properties computed at each pixel
- Regions grow from "seed" points by appending to each seed those neighboring pixels that have predefined properties similar to the seed
  - Similarities in color, moments, texture, etc.

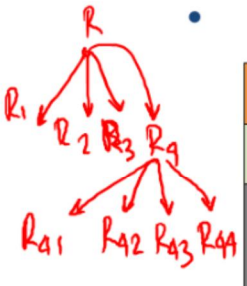
Region Growing Example

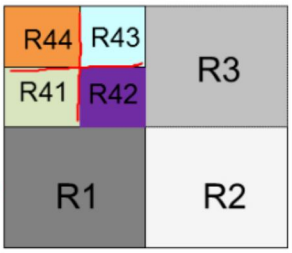


$$S(x(n_1, n_2), x(m_1, m_2)) = exp(-|x(n_1, n_2) - x(m_1, m_2)|^2)$$

Region Splitting and Merging

- Splitting





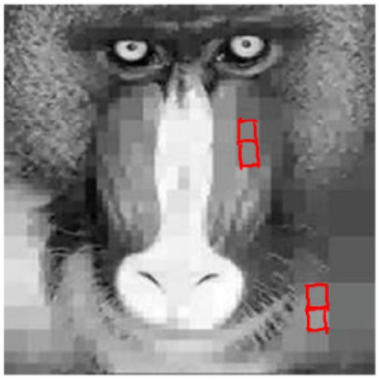
R represents the entire image
Select a predicate function *P*
Divide the image into smaller quadrants if
P(R) = FALSE
A minimum quad-region size is selected.

- Merging

Two adjacent regions R_j and R_k are merged if
$P(R_j \cup R_k)$ = TRUE

Region Splitting and Merging

Original Baboon Image



Region splitting

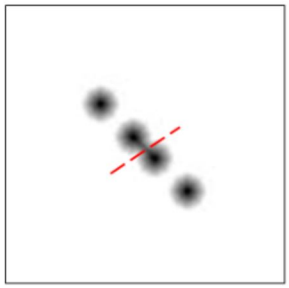
Region splitting and merging

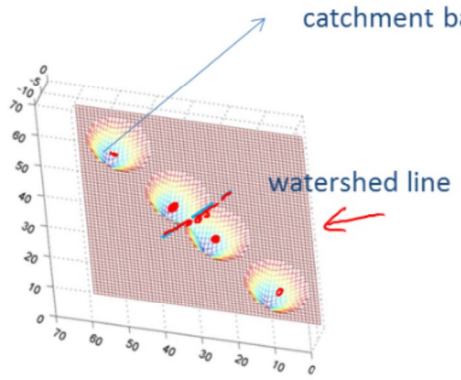
Region Splitting and Merging

2X2

Watersheds

- Watersheds segmentation envisions the grayscale image as a topological surface





Watersheds

Three types of points

- Points belonging to a ***regional minimum***
- **Catchment basin** of a regional minimum
  Points at which a drop of water will certainly fall to a single minimum
- Ridge lines / **Watershed lines**
  Points at which a drop of water will be equally likely to fall to more than one minimum

Goal of the algorithm

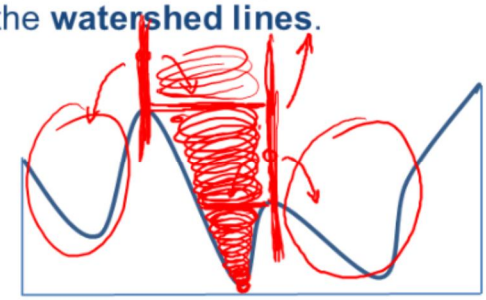
- Find the **watershed lines**

Watersheds

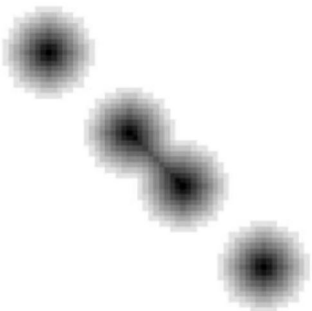
The concept:

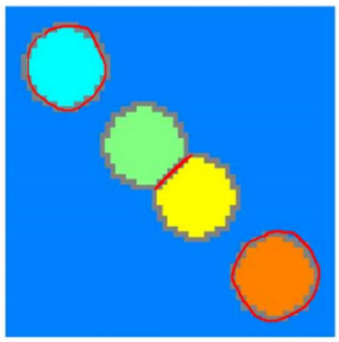
Suppose that a hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate.

When rising water in distinct catchment basins is about to merge, a dam is built to prevent merging. These dam boundaries correspond to the **watershed lines**.

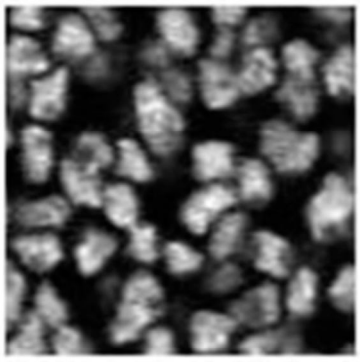


Example





Watersheds Example



Watersheds

A solution to the resulting over-segmentation is to use markers to specify the only allowed regional minima.

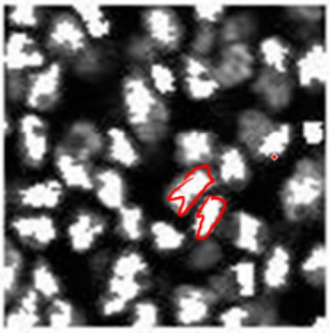
Internal markers:

- each one corresponds to one object
- surrounded by points of higher "altitude"
- points in region form a connected component
- all points in connected components have the same intensity

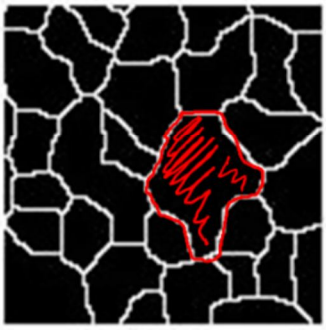
External markers:

- partition the image into regions where the regional minima are allowed to locate

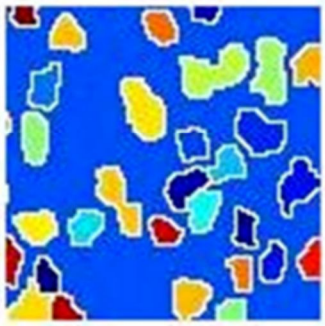
Watersheds Example



Internal markers overlaid on the original image



External markers



Watersheds with markers

K-means Clustering

Clustering means grouping similar data together

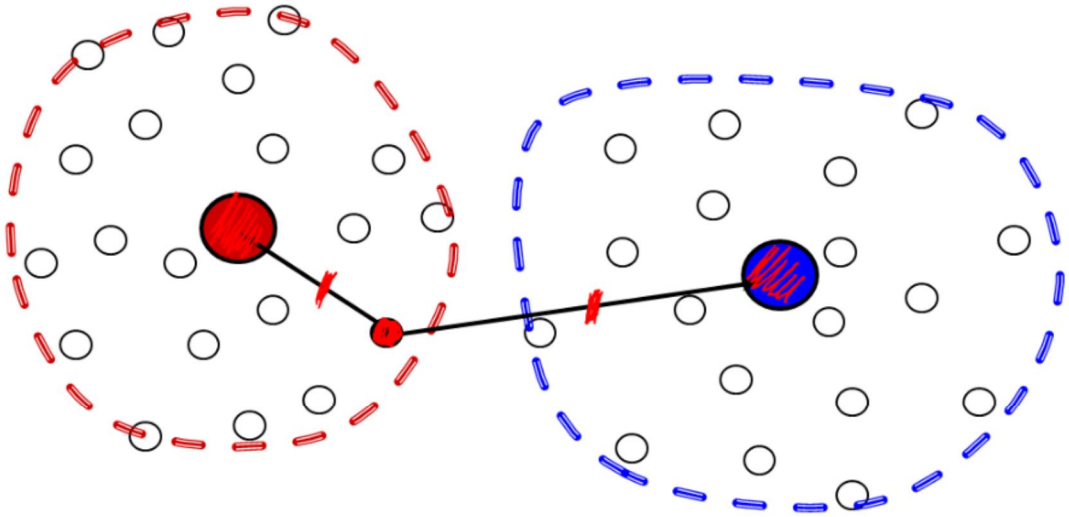
From a set of data points or observations (all numerical), K-means attempts to classify all data points into K clusters

K clusters may represent K segments of an image

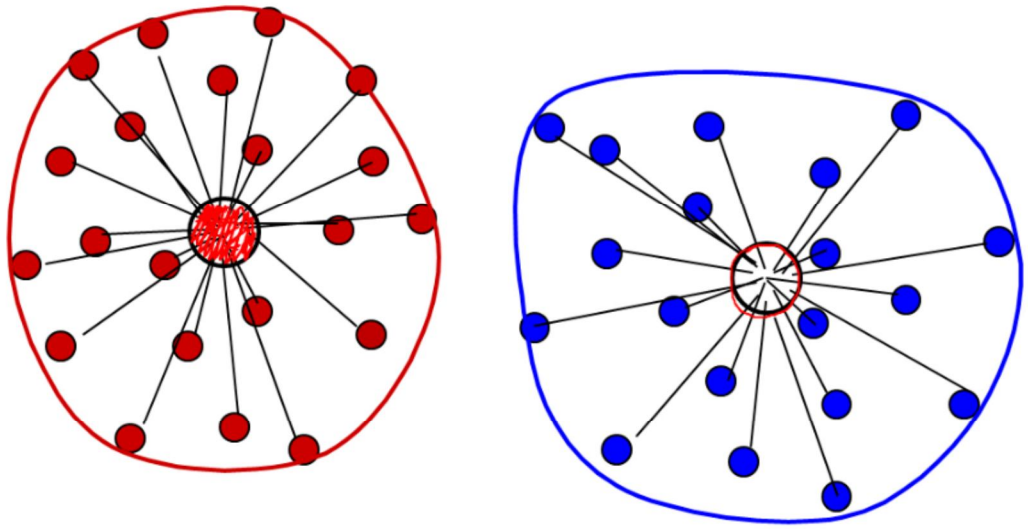
"K" stands for number of clusters, it is a user input to the algorithm

The algorithm is iterative in nature and not confined to image segmentation

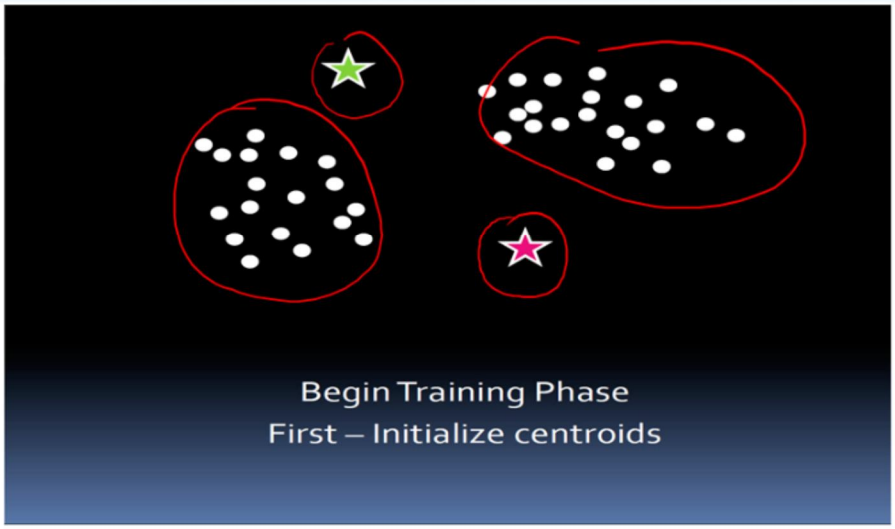
Sub-problem I: Clustering by distance to known centers



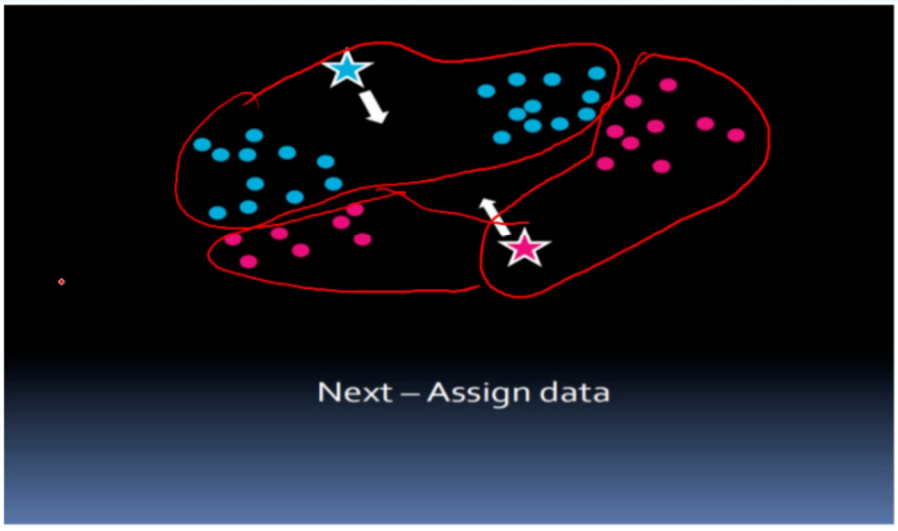
Sub-problem II: Finding new centers from known clustering



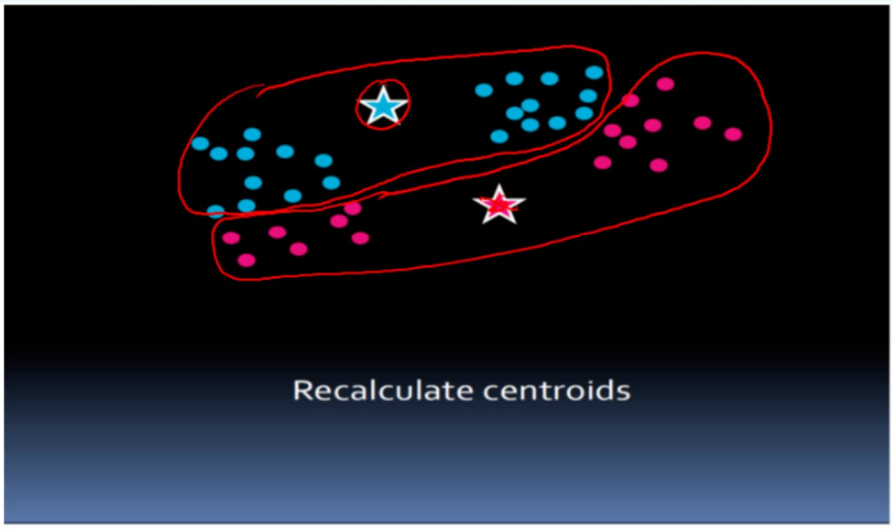
The K-means Clustering



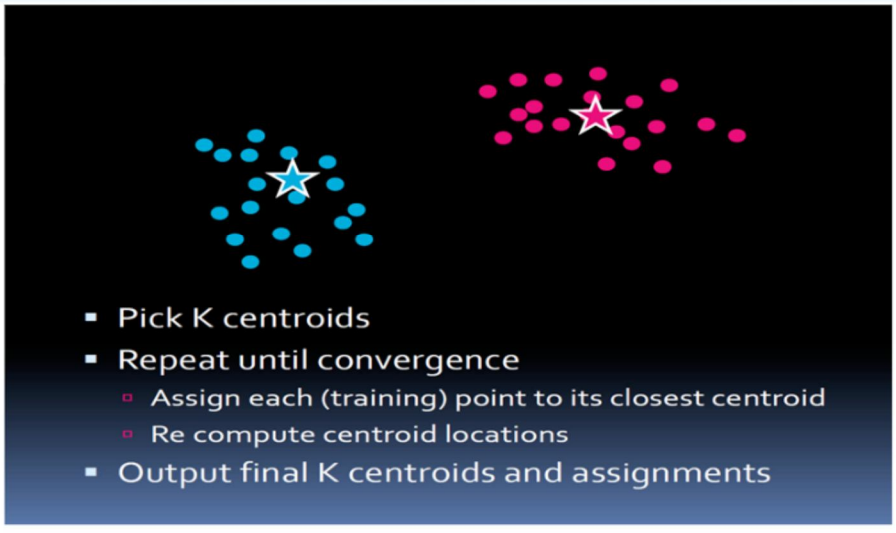
The K-means Clustering



The K-means Clustering

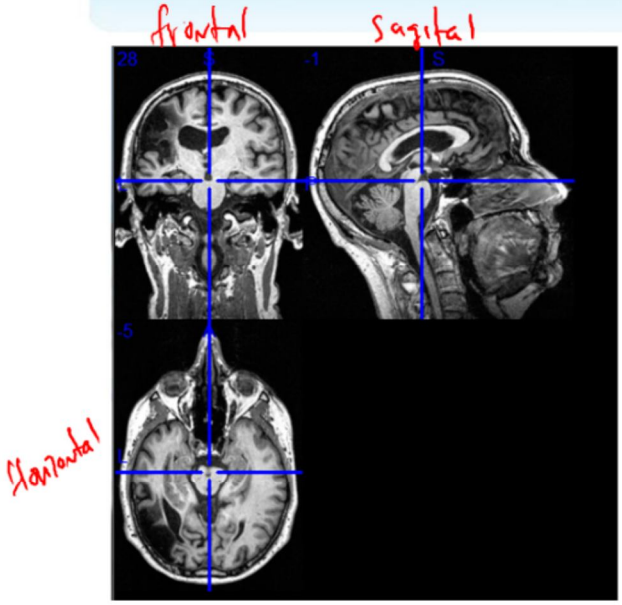


The K-means Clustering



- Pick K centroids
- Repeat until convergence
  - Assign each (training) point to its closest centroid
  - Re compute centroid locations
- Output final K centroids and assignments

The K-means Clustering



$$v(x,y,z) = \begin{bmatrix} I(x,y,z) \\ x \\ y \\ z \end{bmatrix}$$

$$D(v_i, v_j) = \|v_i - v_j\|$$

Prior knowledge:

- Voxel with same intensity should belong to the same group;
- Voxel next to each other should belong to the same group;
- There are three groups: White matter, Grey matter and CSF

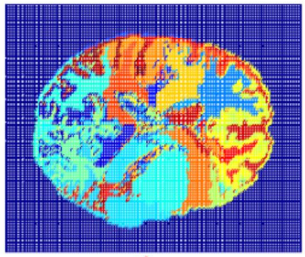
The K-means Clustering

- The definition of vector and distance

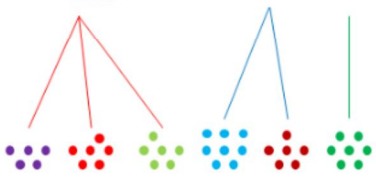
$$v(x,y,z) = \begin{bmatrix} \alpha \cdot I(x,y,z) \\ \beta \cdot x \\ \beta \cdot y \\ \beta \cdot z \end{bmatrix}$$

$$D(v_i, v_j) = \|v_i - v_j\|_1$$

$$D(v_i, v_j) = \|v_i - v_j\|_2$$

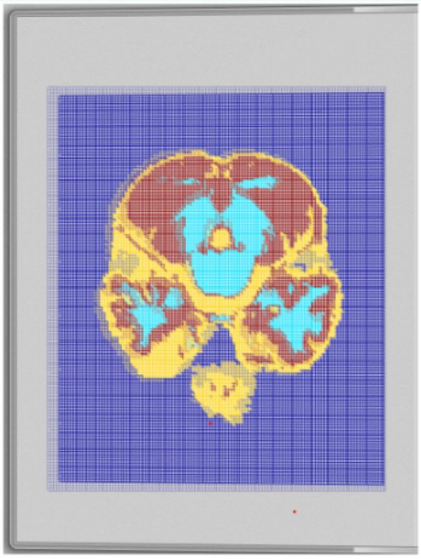


- Means of K-means: Multiple layer of K-means



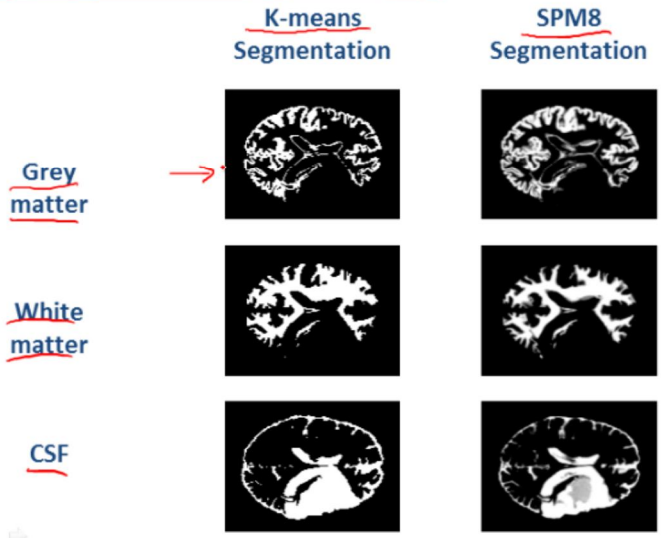
The K-means Clustering

3D segmentation of
A human brain

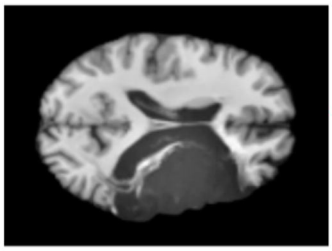


Experimental Results

3D segmentation by K-means on slice 60, compared to SPM8* segmentation result

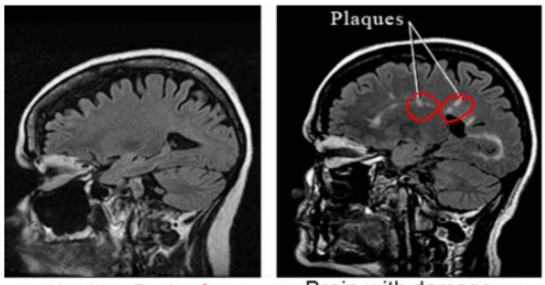


Original Image



*SPM8 is a public toolbox supported by Matlab

Brain Image Segmentation is a very difficult problem; the identification of lesions is even harder



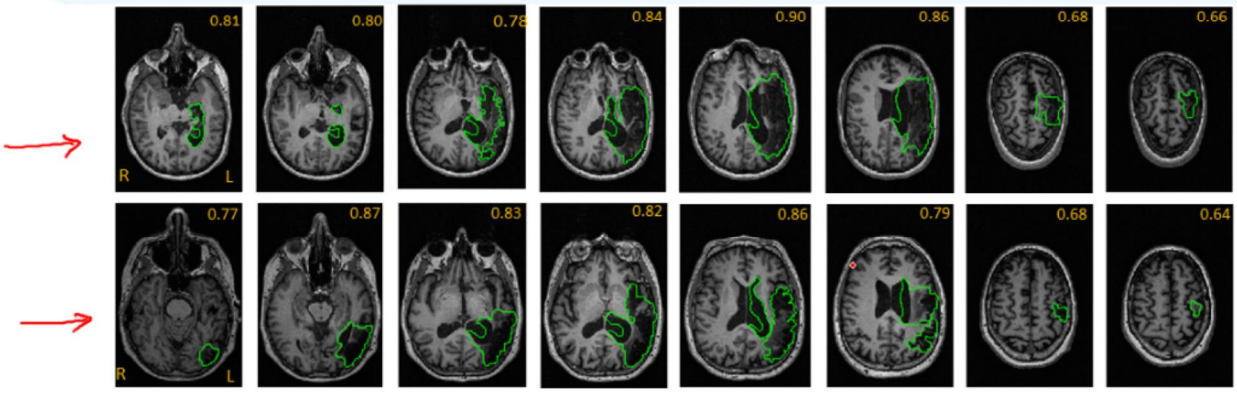
Healthy Brain

Brain with damage (Lesions or plaques)

Current Research Status:

- Trying to identify the stroke lesion and the peri-lesional region to **predict recovery** or **response to therapy**.
- Typically this is drawn by hand which is **time consuming** and **subjective process**.

Segmentation Techniques may Save Lives



- Used proactively, the technique can screen a healthy individual's brain to detect abnormalities.
- Following a disease, however, the same **technique** can assess the patient's response to therapy and predict recovery

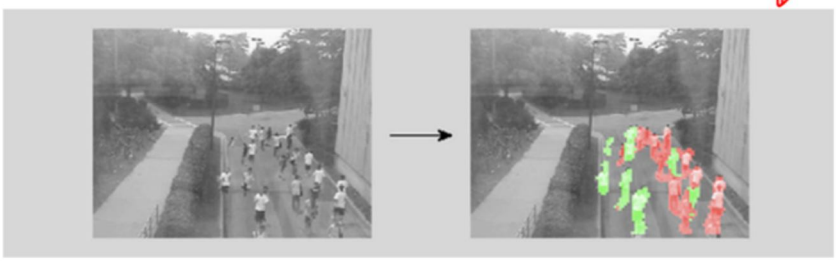
Dice Similarity Index $s = \frac{2|X \cap Y|}{|X| + |Y|}$ [0,1]

Motion-based Segmentation

Motion is a powerful cue used by humans and many other animals to extract objects or regions of interest from a background of irrelevant detail.

Given an image sequence, determine

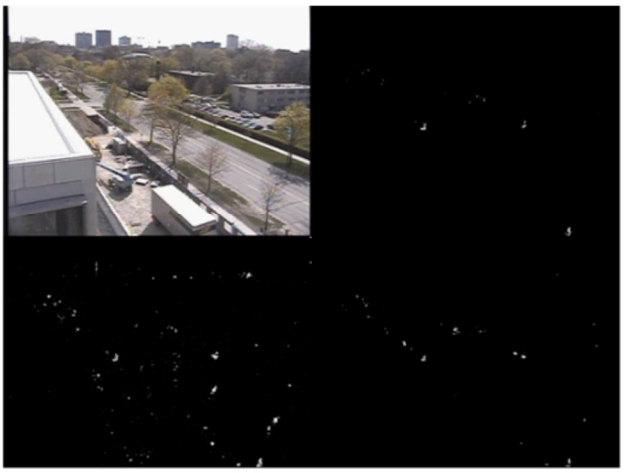
- **Number** of motion models (affine, Euclidean, etc.)
- **Motion model**: affine (2D) or Euclidean (3D)
- **Segmentation**: model to which each pixel belongs



Motion-based Segmentation

- Object detection
  - pedestrian detection
- Tracking
  - vehicle tracking
- Robotics
- Surveillance
- Image and video compression
- Scene reconstruction
- Video manipulation / editing
  - video matting
  - video annotation
  - motion magnification

Example

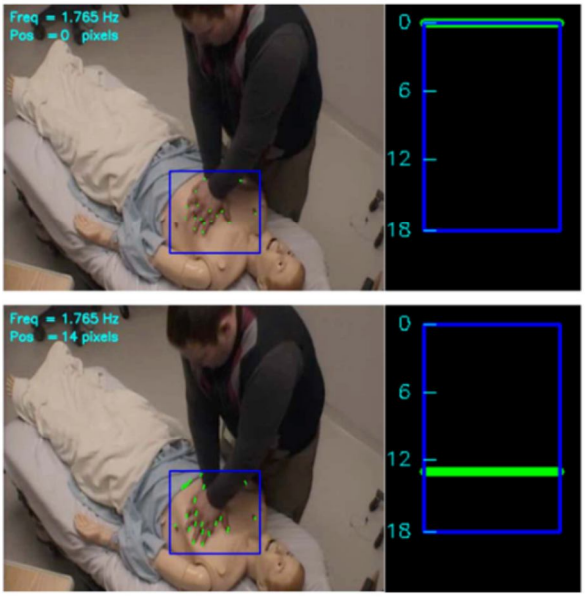


Motion-based Segmentation

Segmenting images based on **common motion**

Points moving together are **grouped together**

Other useful information can be extracted trough the process,
e.g. depth and frequency of motion



Motion-based Segmentation

Basic Approach

- A difference image between t_i and t_j is defined as

$$d_{i,j}(x,y) = \begin{cases} 1 & \text{if } |f(x,y,i) - f(x,y,j)| > T \\ 0 & \text{otherwise} \end{cases}$$

T is a specified threshold.

- Object motions result in 1's in $d_{i,j}$
- Assumptions
  - Images registered spatially
  - Relatively constant Illumination

Motion-based Segmentation

Improvement by Accumulative difference image (ADI)

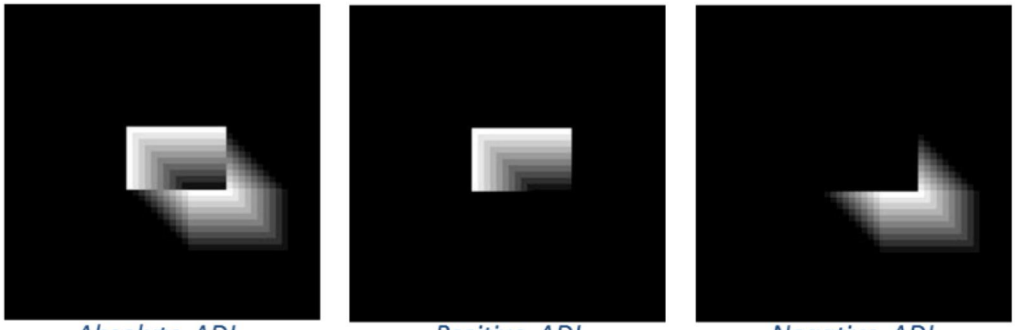
$$A_k(x,y) = \begin{cases} A_{k-1}(x,y) + 1 & \text{if } |R(x,y) - f(x,y,k)| > T \\ A_{k-1}(x,y) & \text{otherwise} \end{cases}$$ *Absolute*

$$P_k(x,y) = \begin{cases} P_{k-1}(x,y) + 1 & \text{if } [R(x,y) - f(x,y,k)] > T \\ P_{k-1}(x,y) & \text{otherwise} \end{cases}$$ *Positive*

$$N_k(x,y) = \begin{cases} N_{k-1}(x,y) + 1 & \text{if } [R(x,y) - f(x,y,k)] < -T \\ N_{k-1}(x,y) & \text{otherwise} \end{cases}$$ *Negative*

- Isolated entries in $d_{i,j}$ resulting from noise are not insignificant
- Although thresholding using connectivity may remove these entries, small or slow-moving objects are also removed.
- Use of ADI ignores changes that occur only sporadically

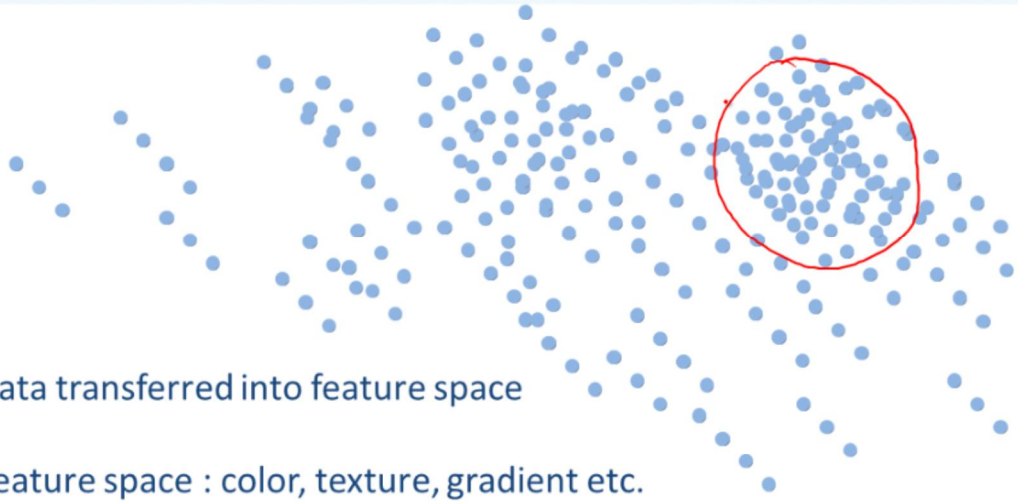
Motion-based Segmentation



Absolute ADI *Positive ADI* *Negative ADI*

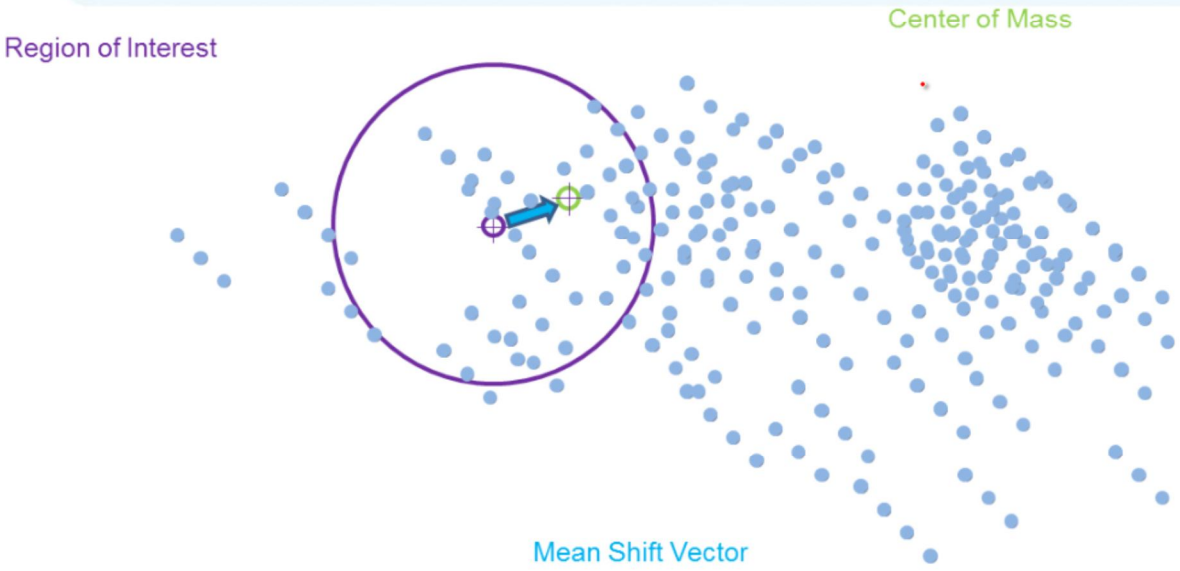
From positive ADI : size of the moving object
location of the moving object in reference image
From absolute ADI / negative ADI : speed and direction of the motion

Mean Shift



- Data transferred into feature space
- Feature space : color, texture, gradient etc.
- Find modes/ local maximum of density in the feature space

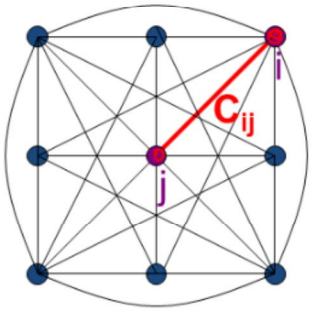
Mean Shift



Mean Shift

Mean Shift

Graph Cut



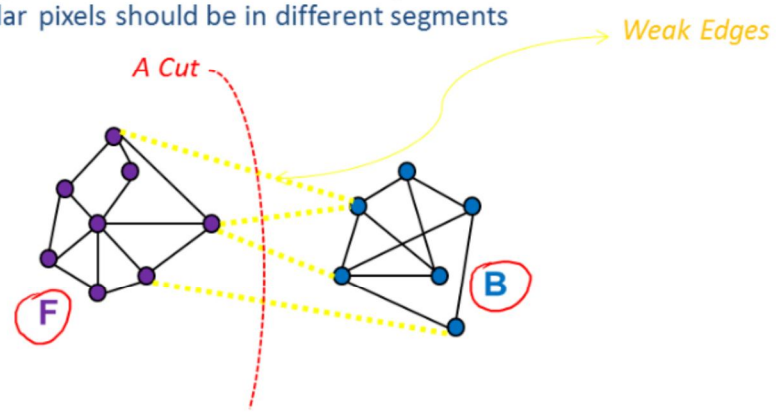
Images as graphs

Pixels correspond to a Vertices (V)
Pairs of pixels correspond to Edges (E)
Similarities between pixels correspond to weights/costs $C_{i,j}$

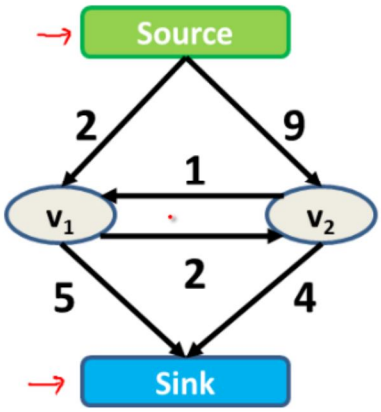
Graph Cut

Intuition : Break Graph into Segments

- Delete Edges that cross between segments
- Easiest to break Edges that have low cost (low similarity)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

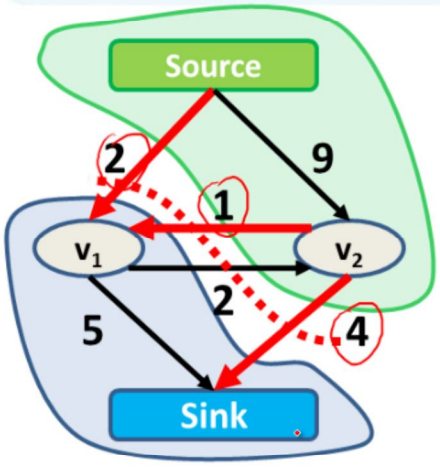


Example: st-Mincut Problem



What is a st-cut?

st-Mincut Problem



2 + 1 + 4 = 7

What is a st-cut?

An st-cut (**S**,**T**) divides the nodes between source and sink.

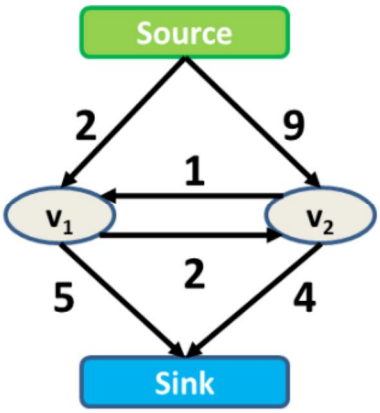
What is the cost of a st-cut?

Sum of cost of all edges going from S to T

What is the st-mincut?

st-cut with the minimum cost

How to compute the st-mincut?



Solve the dual **maximum flow** problem

Compute the maximum flow between Source and Sink

Constraints

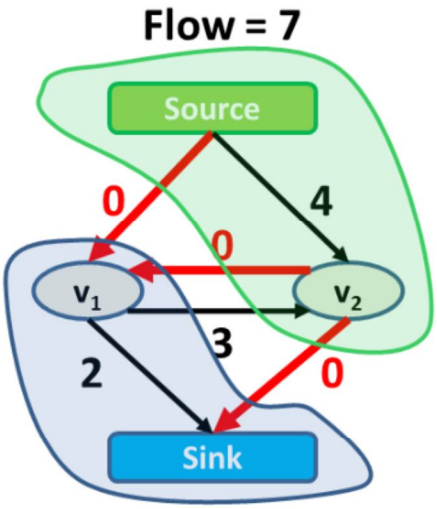
Edges: Flow < Capacity

Nodes: Flow in = Flow out

Min-cut\Max-flow Theorem

In every network, the maximum flow equals the cost of the st-mincut

Maxflow Algorithms



Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity*
2. Push maximum possible flow through this path
3. Repeat until no path can be found

**Algorithms assume non-negative capacity*

Example: using Min Cut