Fakultas Informatika
School of Computing
Telkom University

CII4F3 / Pengolahan Citra Digital

Image Segmentation and Registration

Dr. Eng. Ir. Wikky Fawwaz Al Maki, S.T., M.Eng.

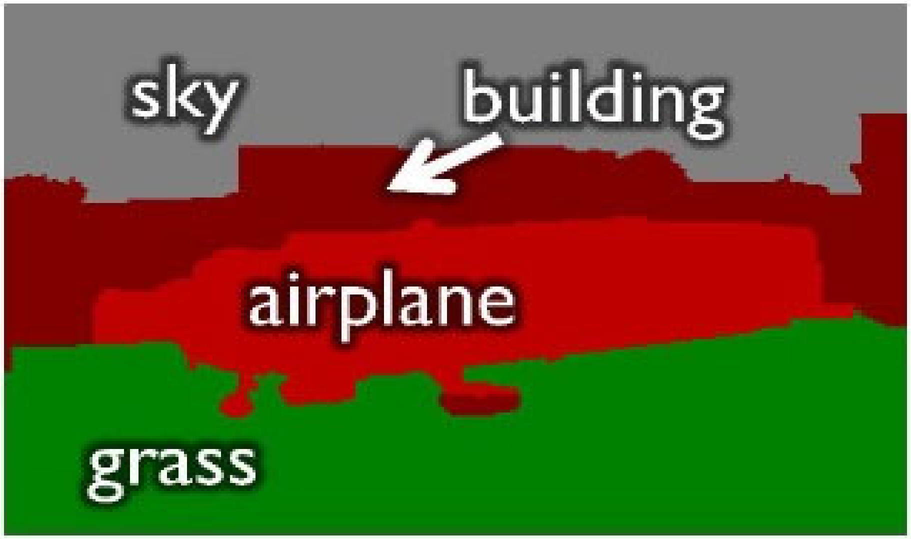
Intelligent Computing and Multimedia (ICM)

Introduction

- Image Segmentation
  - What is it?
  - Tresholding
  - K-means clustering
  - Canny edge detection
  - Graph cut
  - Region growing
  - Level set
- Image Registration
  - What is it?
  - Main components
  - Feature-based
  - Intensity-based
  - Examples

IMAGE SEGMENTATION

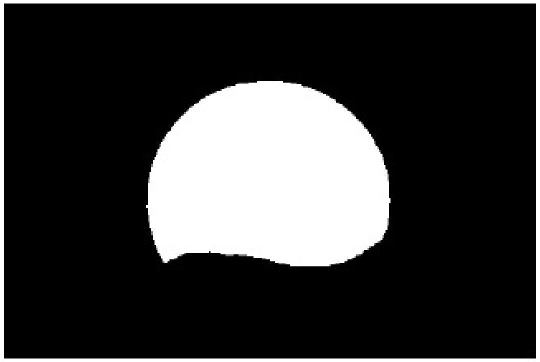
Image Segmentation

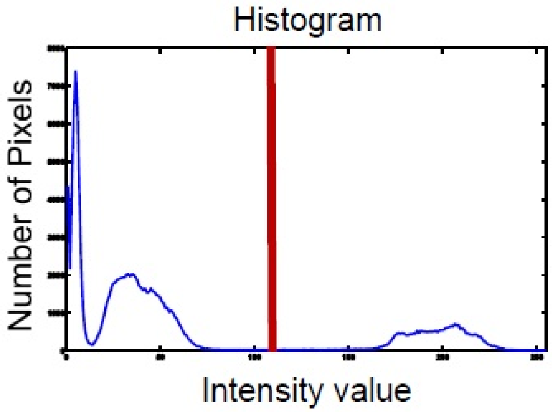


Aim of Image Segmentation

- Partition image into a set of regions which
  - are visual distinct and
  - share certain visual properties
    - Intensity
    - Colour
    - Texture
- To simplify representation for easier analysis

Segmentation via Thresholding



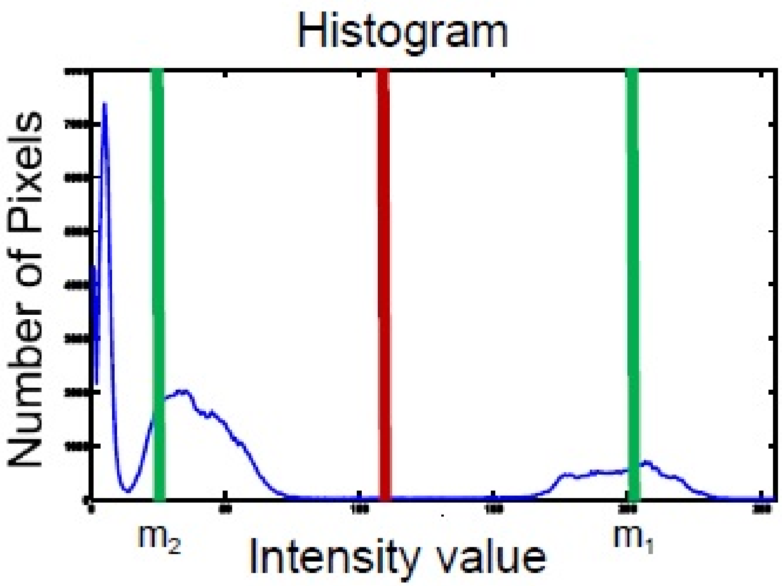


- Value above threshold: Object
- Value below threshold: Background

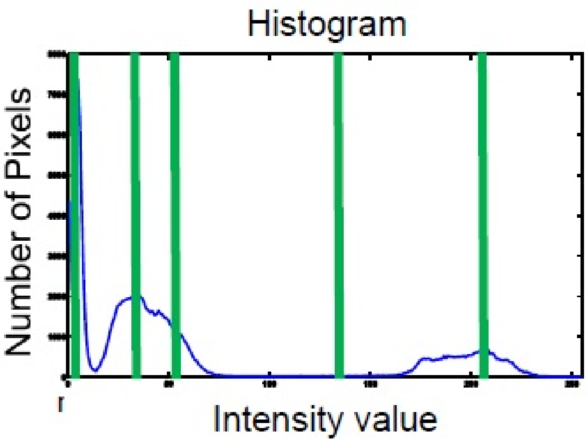
Automatic Thresholding – Two Classes

- Algorithm :

1. Choose initial threshold (e.g. randomly)
2. Segment the image into object and background
3. Compute mean intensity of object (m_1) and background (m_2)
4. Set new threshold to $(m_1+m_2)/2$
5. Repeat from 2. until no change



Segmentation via K-Means Clustering



Use K means → get K classes
1) K initial means (m_k)
2) Assign each pixel p to cluster k
   - $\mathrm{argmin}_k\ |\mathrm{intensity}(p) - m_k|$
3) Recalculate cluster mean
4) Repeat from 2) until convergence

K-Means Clustering for Color Images

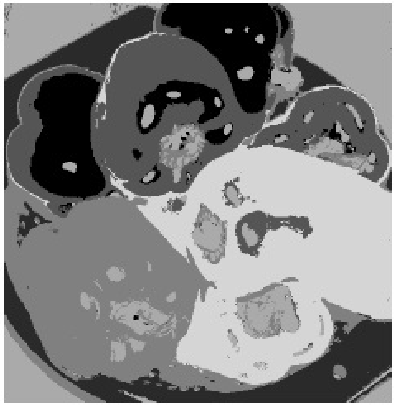
- So far K-means clustering in 1D (intensities)
- Same process for nD by using vector distances
  - e.g. color images (RGB described as 3D-vector)

R G B

K-Means Clustering – Number of Clusters

10 clusters

K-Means Clustering - Initialization



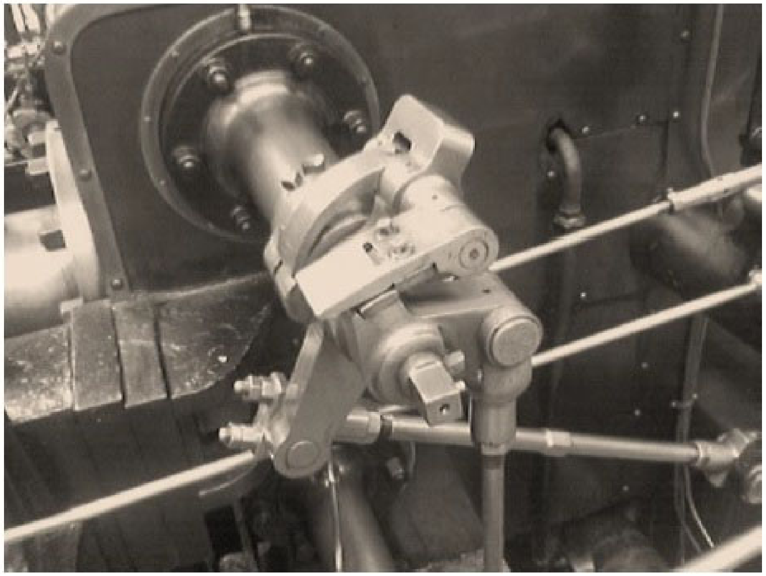
K-Means Clustering - Summary

- Result depends on initialization
- Number of clusters is very important
- No spatial considerations

Edge Detection

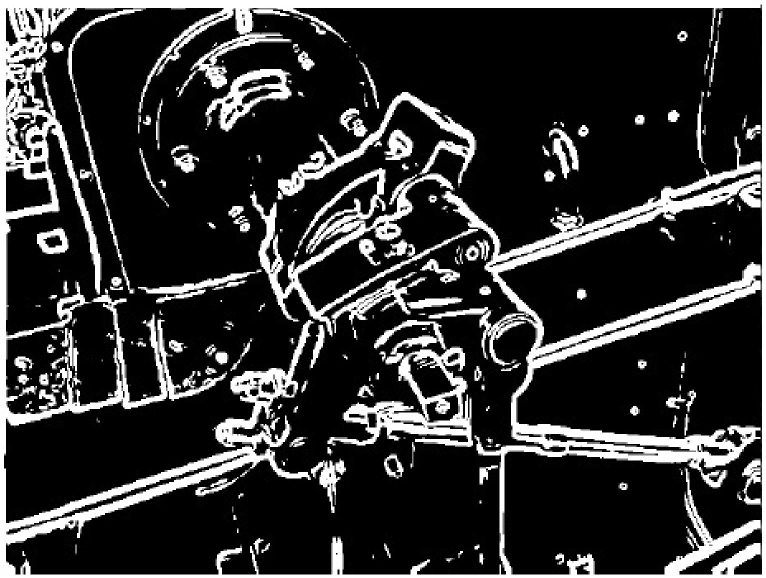
Why Edge Detection ?

Edge Detection



Task:
Segment the image by finding relevant edges

Edge Detection Task

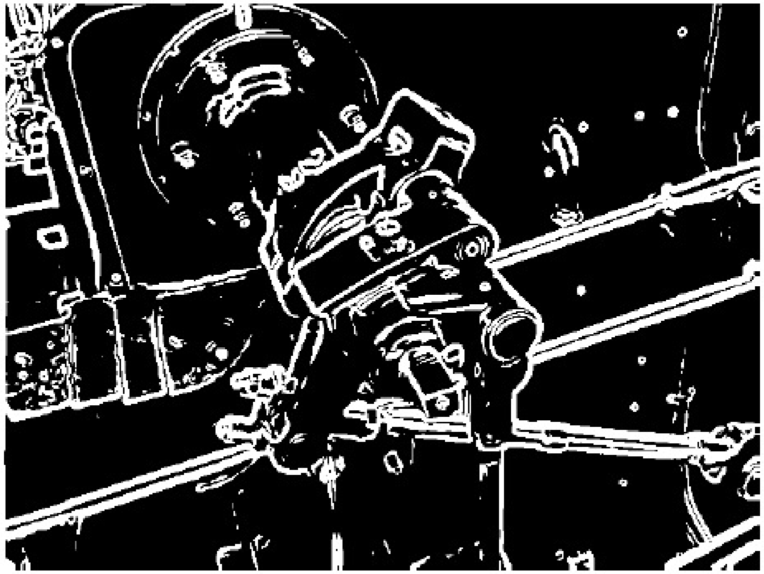


- **Task:**
  Segment the image by finding relevant edges
- **Simple Way:**
  - Smooth the image
  - Calculate magnitude of image gradient
  - Threshold

Canny Edge Detection

- Aim for “optimal” edge detection algorithm
- Good detection
  - find all relevant edges
- Good localization
  - find edge at the right location
- Minimal response
  - find only relevant edges

Canny Edge Detection



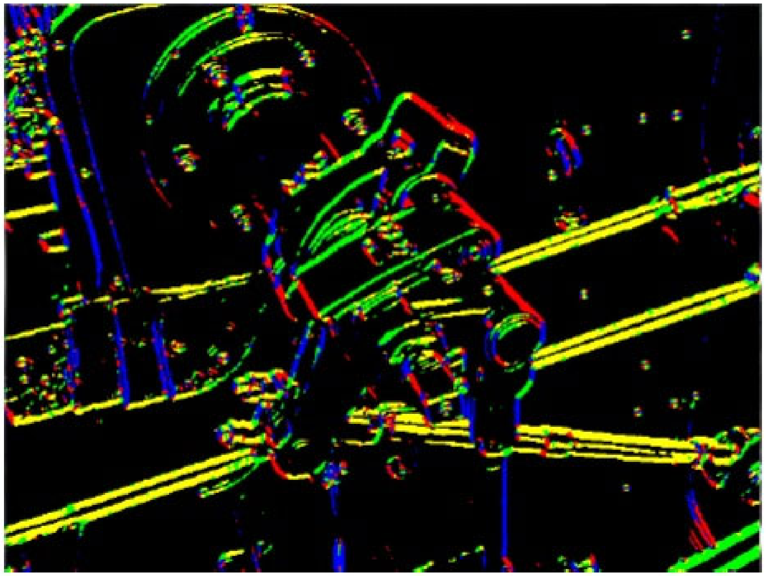
Optimal edges?

good detection: yes

good localization: NO

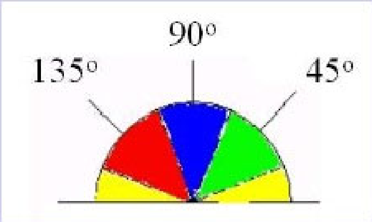
minimal response: yes/no

Canny Edge Detection – Edge Direction

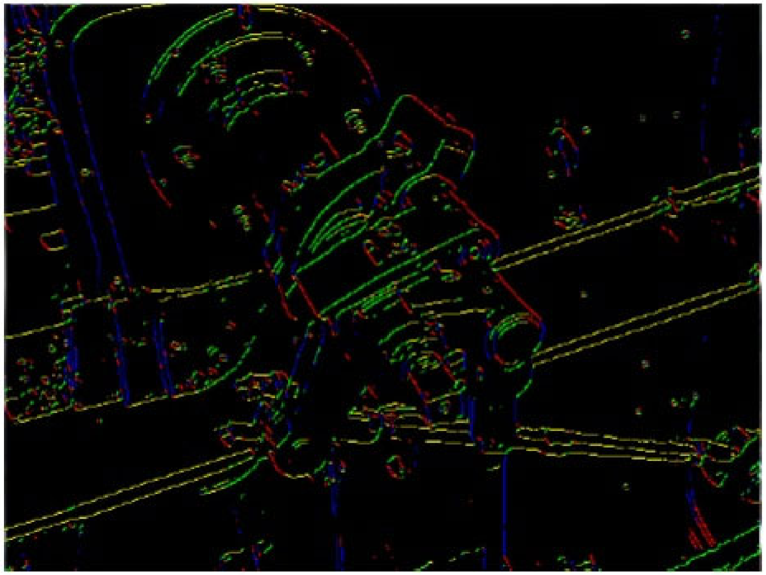


Improve localization

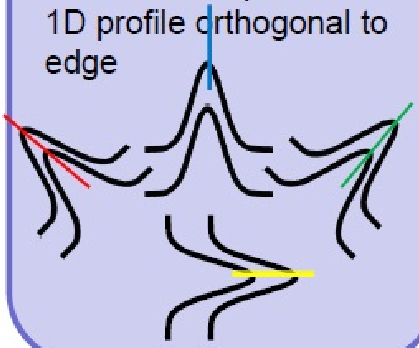
Classify according to gradient direction



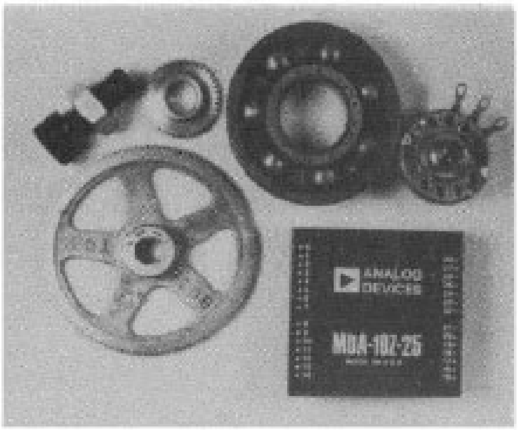
Canny Edge Detection – Thinning



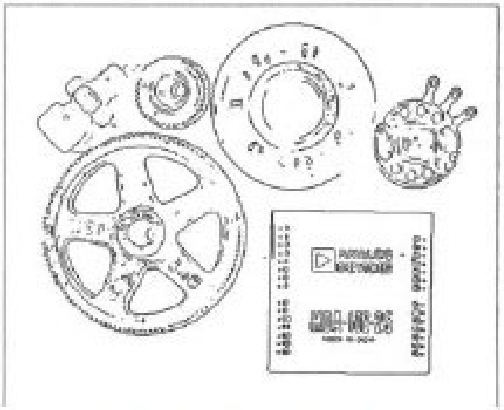
Non-maximum suppression
Thin edge in gradient direction: i.e. keep maximum response on 1D profile orthogonal to edge



Canny Edge Detection – Thresholding



(a)

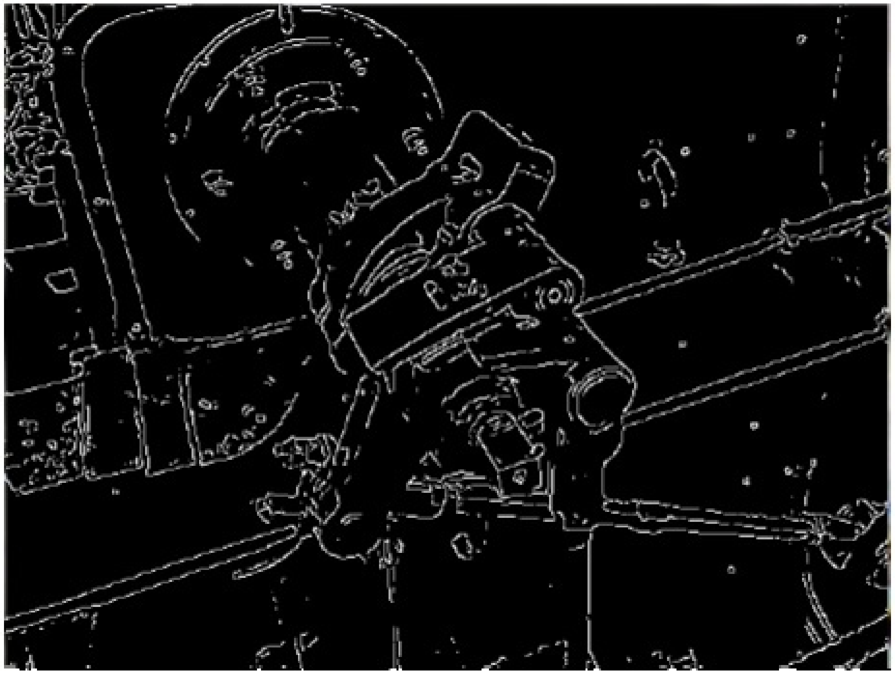


Strong edges

Hysteresis thresholding

- To find relevant edges
- Keep strong edges (response > T_{high})
- Keep weaker edges connected to strong edges (response > T_{low} and connectable to T_{high} pixels)

Canny Edge Detection – Result

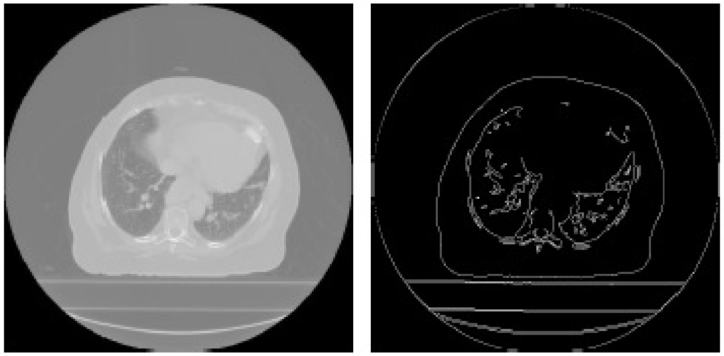


Canny Edge Detection - Steps

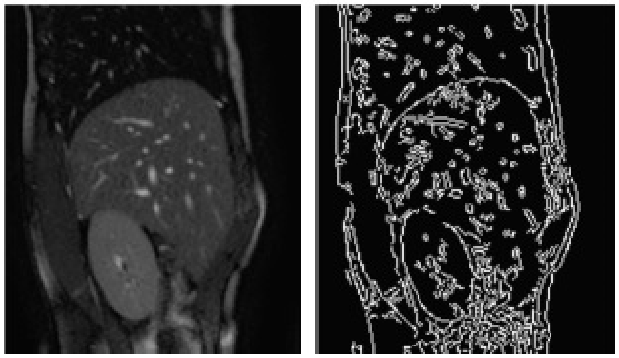
1. Convolve image with Gaussian filter
2. Compute edges and estimate edge direction
3. Find edge locations using non-maximal suppression
4. Trace edges using hysteresis thresholding

Canny Edge Detection - Summary

- Affected by noise
- No automatic threshold selection
- Useful as preprocessing step



Axial slice of lung CT image

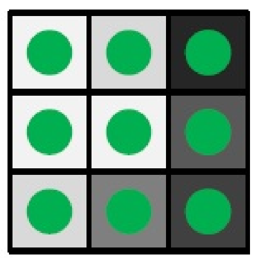


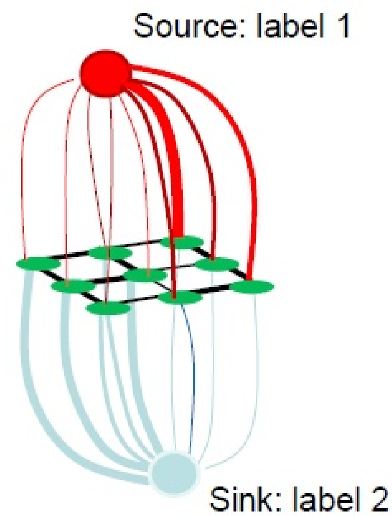
Sagittal slice of liver MR image

Hybrid Methods

- So far methods based on
  - properties of single pixels (e.g. thresholding, K-mean clustering)
  - relationship between neighbouring pixels (e.g. Canny edge detection)
- Combine these!

Graph Cut

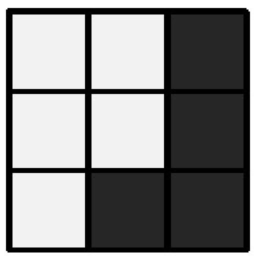




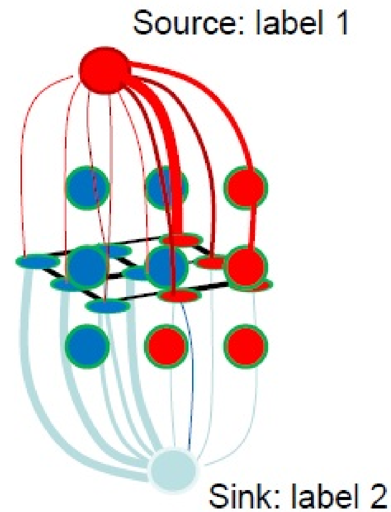
$$E(y) = \sum_{p \in P} E_d(y_p) +$$

Unary term: cost to
not assign label y_p to pixel p

Graph Cut



Cut graph such that cost *E(y)* is minimal

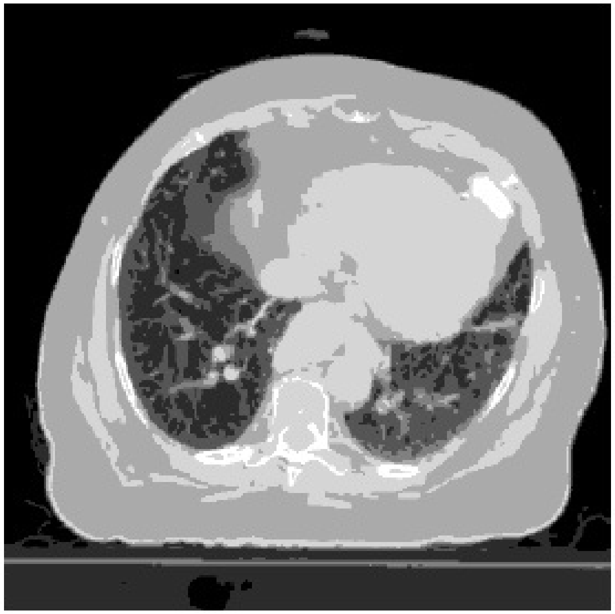


$$E(y) = \sum_{p \in P} E_d(y_p) + \lambda \sum_{p \in P, q \in N_2(p)} E_s(y_p, y_q)$$

Unary Term: cost to not assign label y_p to pixel p

Binary Term: cost to assign label y_p to p and y_q to q

Examples: K-Means refined by Graph Cut



Graph cut

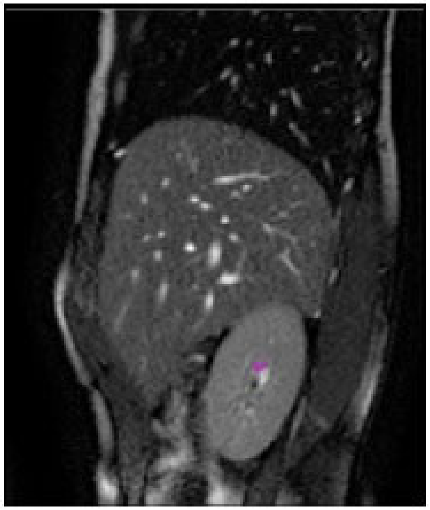
Graph cut

Graph Cut - Summary

- Hybrid method: intensity and edge costs
  - Provides method to solve such a problem
  - Global optimum
- But high memory usage

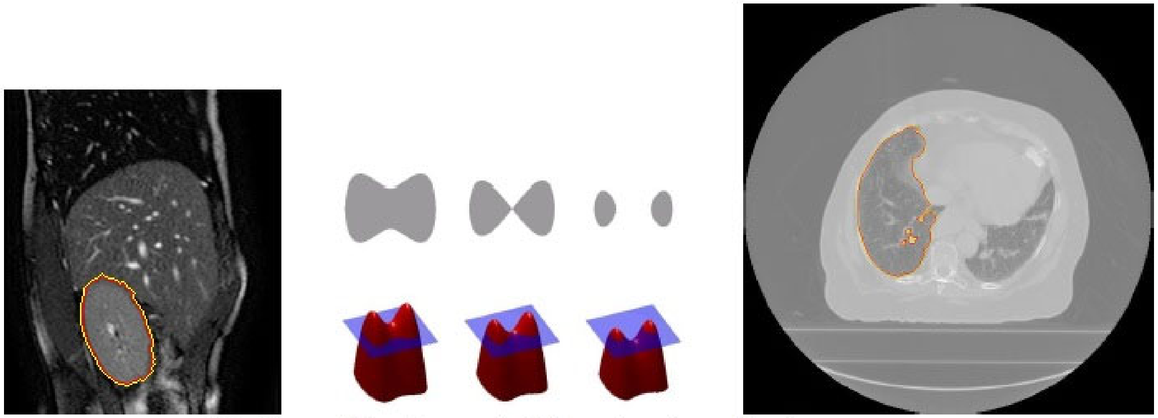
Region Growing

- Region based perspective
  - From a (manually selected) seed (pixel or region)
  - Expand boundary to enclose homogenous region (e.g. allowed intensities within range of mean±delta)
  - Leakage – when to stop?



Level Set

- Want smooth boundary enclosing homogenous regions
- Hybrid method: intensities and curvature of boundary



http://en.wikipedia.org/wiki/Level_set_method

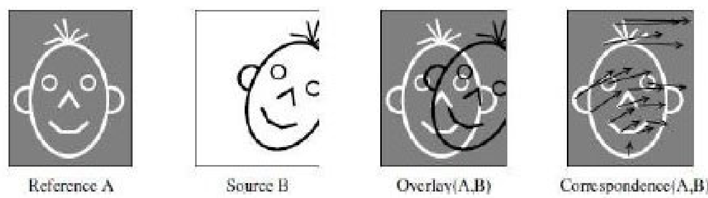
Top-down Methods

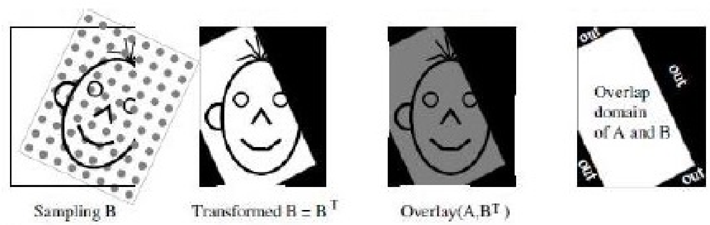
- So far all methods were bottom-up
-
- Can we use prior knowledge?
  - What objects are expected in the images?
  - E.g. searching for certain shapes and appearances
  - Important especially for noisy, low-contrast, low-resolution images
  - Involves image registration → next

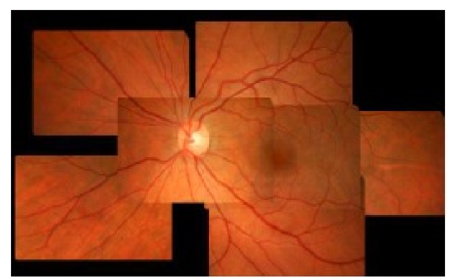
IMAGE REGISTRATION

Image Registration

- Aim: to establish spatial correspondences

- Result: motion vect  between images

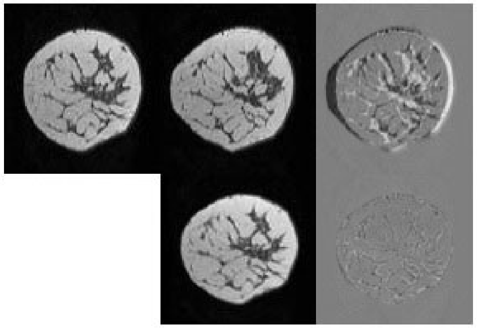




Main Component: Optimization Criteria

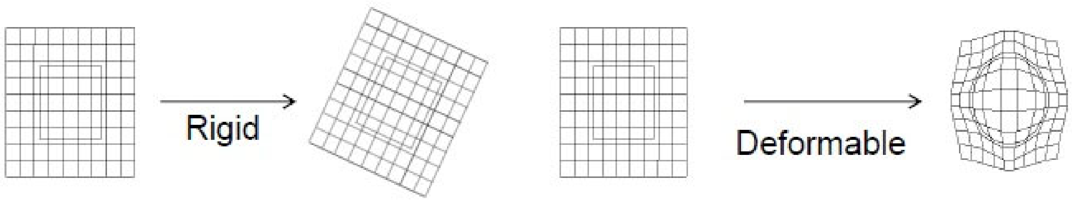
- Feature-based
  - Find feature candidates
  - Match features (minimize feature difference)
  - Estimate spatial transformation

- Intensity-based
  - Transform source image such that similarity between images is maximized



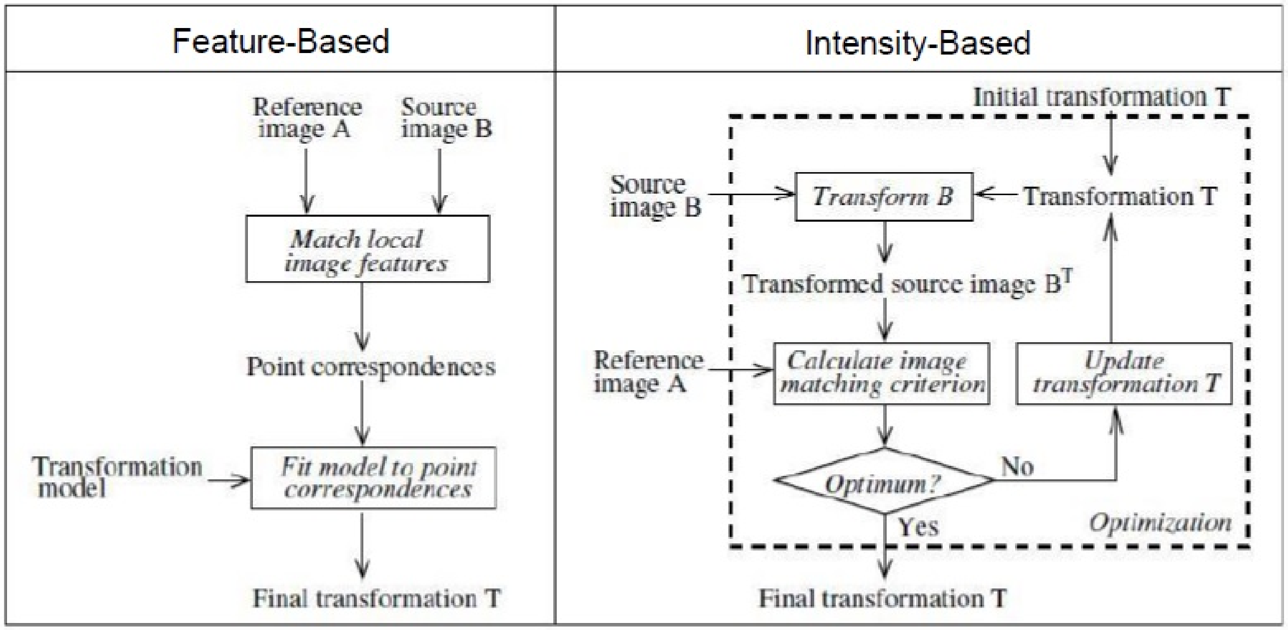
Main Component: Spatial Transformation

- What spatial transformation is expected?
  - E.g. rigid transformation for bones
  - Helps to constrain problem



- Defines interpolation function
  - From sparse correspondences (e.g. at features)
  - To dense displacement field

Image Registration Approaches



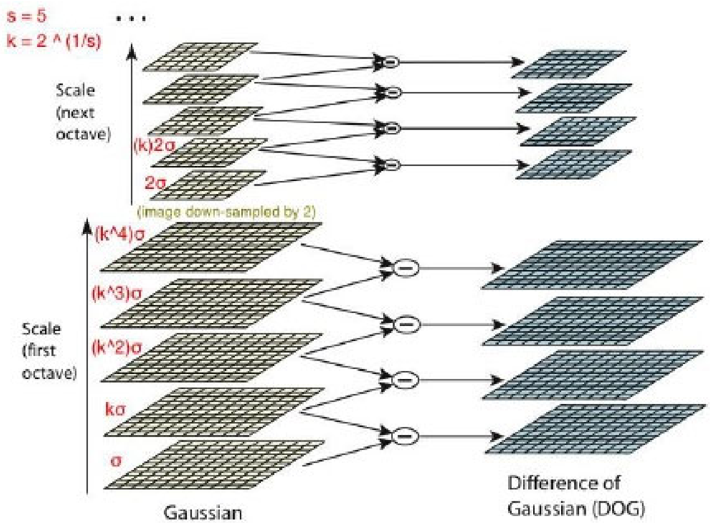
Landmarks - SIFT Features

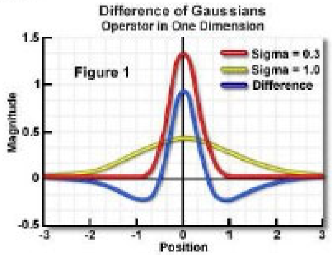
- **SIFT** = Scale Invariant Feature Transform
- **Rotation** and **scale** invariant

Landmarks - SIFT Features

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Scale Space



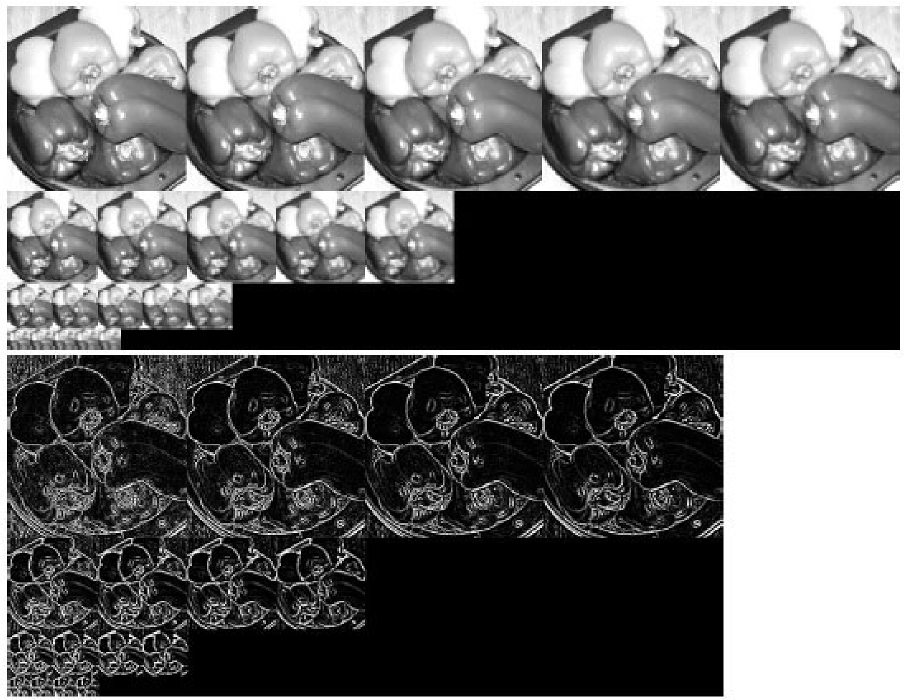


- Convolve image with Gaussian kernel
- Get DoG images
- Downsample by factor of 2
- Repeat

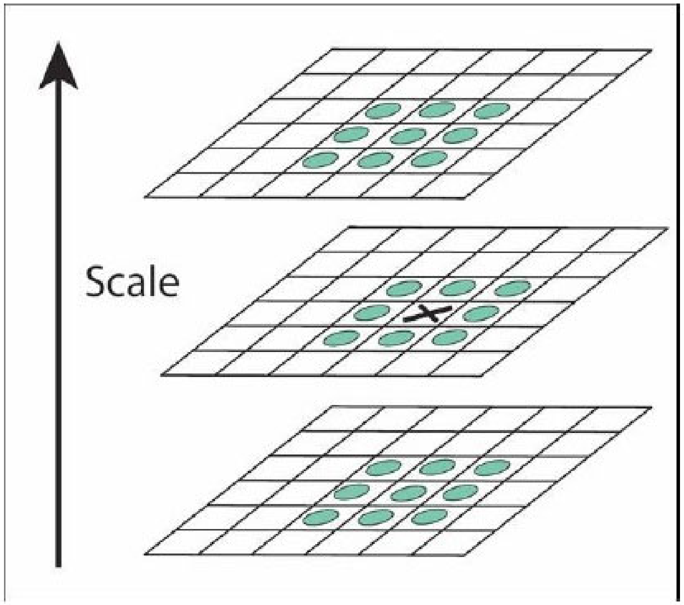
Scale Space

Gaussian blurred images

Difference of Gaussian images



Scale Space Extrema Detection

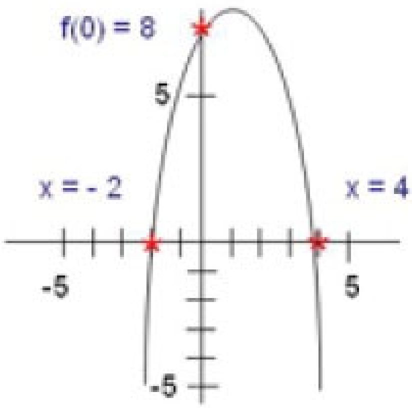


Detect extrema of DoG images:

- Compare pixel to its 26 neighbors
  - In 3x3x3 regions
  - At current and adjacent scales
- Current pixel extrema of all neighbours?

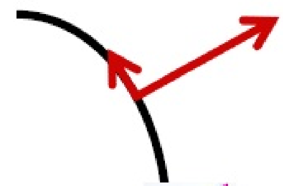
Keypoint Localization

- Where exactly is the extrema?
  - Fit a 3D quadratic function to the local sample points
  - Determine the interpolated location of the extrema



- Reject extrema with low contrast

Keypoint Localization



- Remove edge responses
- DoG function gives strong response at edges
- But location along the edge is poorly determined
  - 1 large principal curvature (across edge)
  - 1 small principal curvature (along edge)
- Eigenvalues (α>β) of Hessian matrix H are proportional to principle curvatures

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

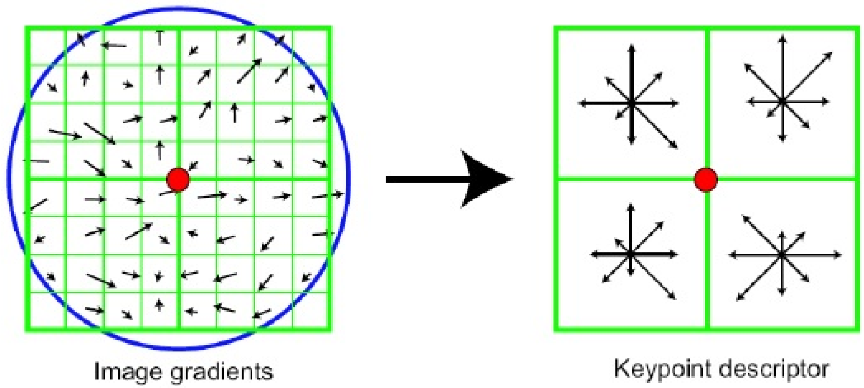
- Reject if ratio of eigenvalues (r = α/β) is large

Orientation Assignment

- Assign one or more orientations to each keypoint
  - Histogram of local image gradient directions in neighborhood
  - Peaks in histograms (>80% of max) define dominant orientations

- Achieves rotation invariance
  - Future operations on images after transformation relative to this orientation, scale, location

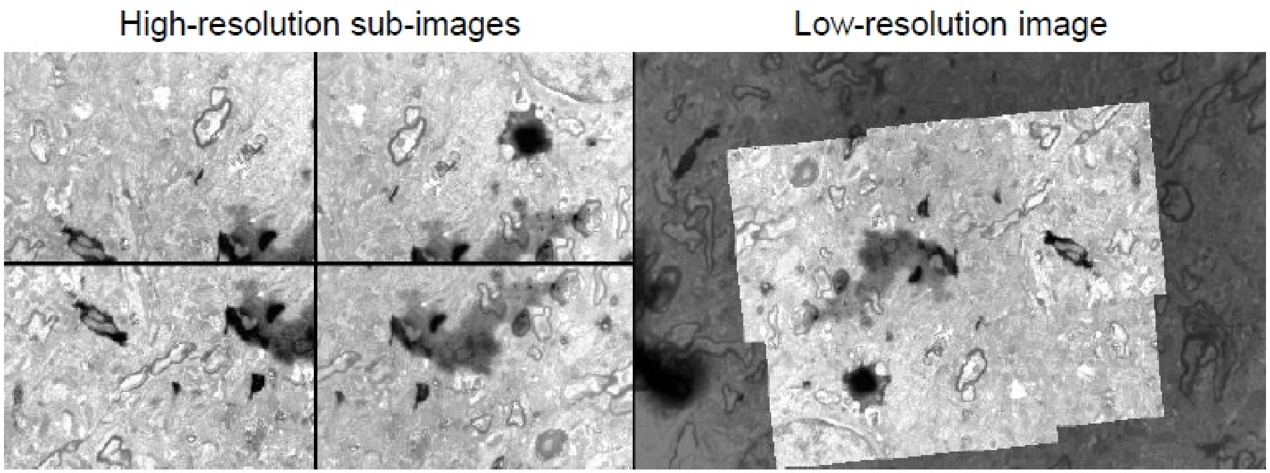
Keypoint Descriptor



- Compute image gradient magnitude and orientation around keypoint
- Surrounding divided into 4x4 subregions
- Accumulate into orientation histograms (8 bins) relative to keypoint orientation
- Keypoint descriptor of length 128 (=16 subregions*8 bins)

- **Correspondences indicated by small distance between key point descriptors**

Stitching Example



1) Detect keypoints
2) Establish correspondences between keypoints
3) Determine transformation

Transformations

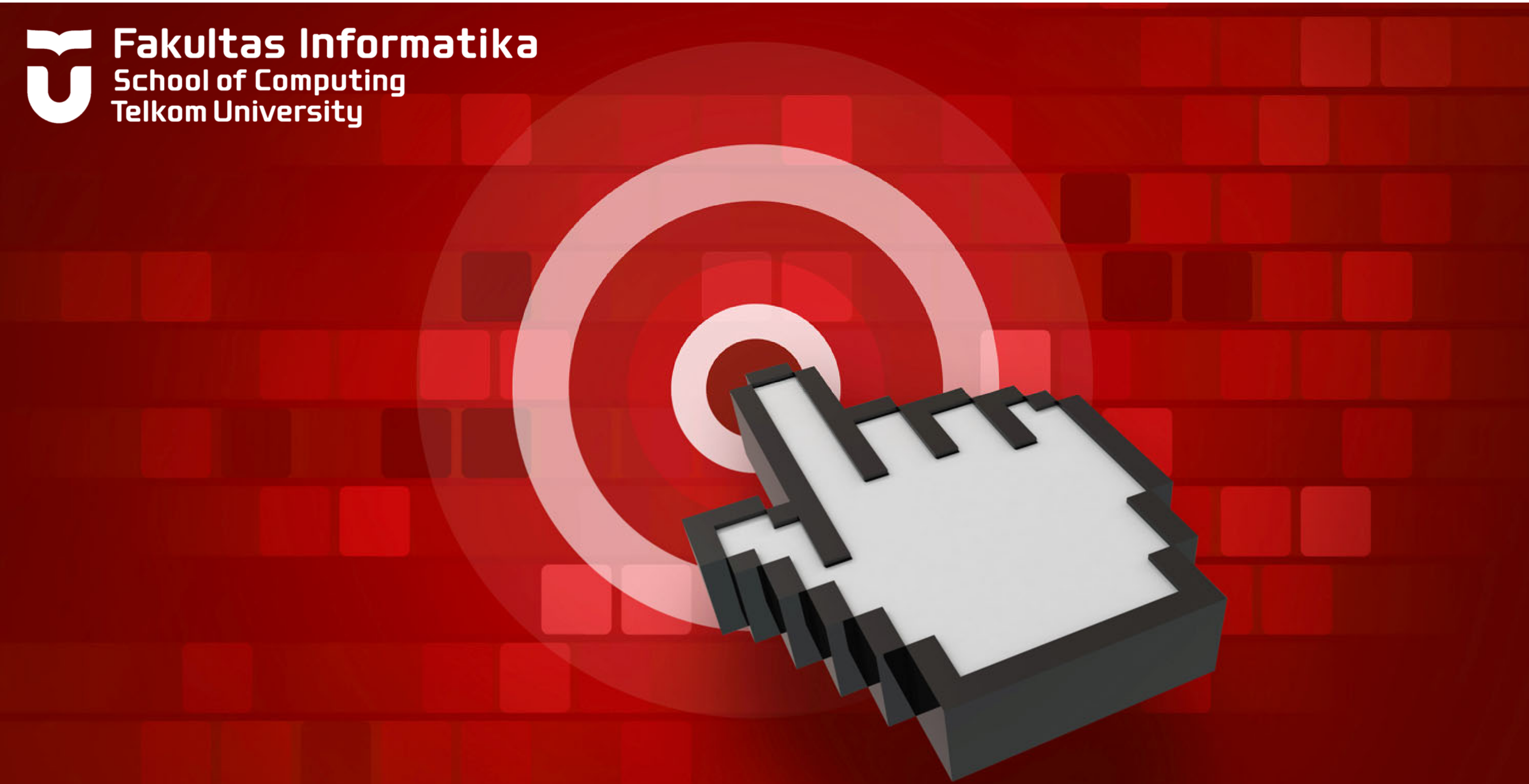
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotation

All 2D affine transformations (translation, rotation, scaling, shearing) can be expressed in a 3x3 transformation matrix **A**, i.e. $\underline{x}'=\mathbf{A}\underline{x}$


Fakultas Informatika
School of Computing
Telkom University

THANK YOU