# IMAGE PROCESSING

# SEGMENTATION

Chakravarthy Bhagvati
Dept. of Computer & Information Sciences
University of Hyderabad

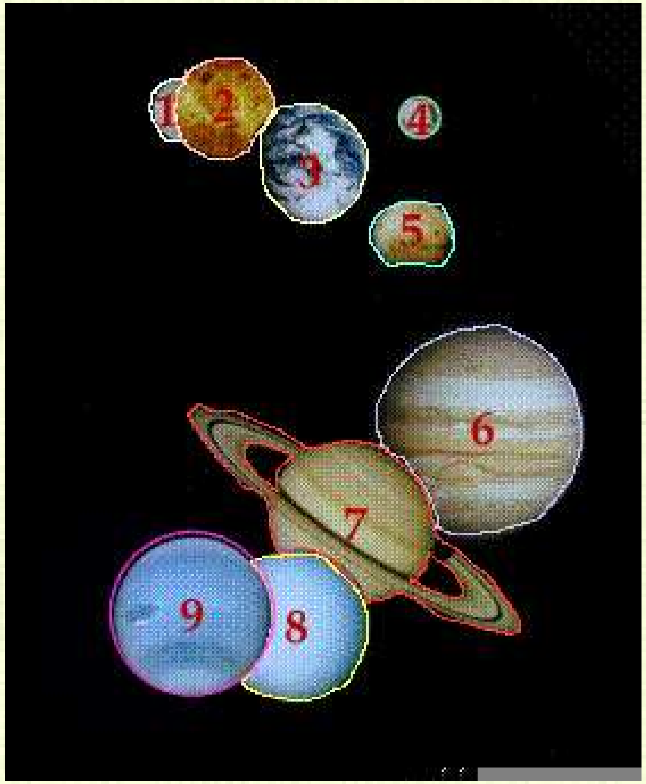
Email: chakcs@uohyd.ernet.in

13 September 2009


- Introduction
- Basic Techniques
  - thresholding and ranging
  - edge detection
  - connected components
- Region Based Techniques
  - region growing
  - region splitting
  - region split-and-merge
- Morphological Techniques
  - conditional dilation
  - closing and watersheds
- Clustering Techniques
  - k-means
  - nearest neighbours
  - neural networks
- Graph Based Techniques
  - spanning tree
  - normalized cuts
- Transform Domain Techniques
  - fourier techniques
  - wavelets
  - Gabor filters
- Robust Techniques
- Conclusion

**OVERVIEW**

- Segmentation: subdividing an image into its constituent parts
- What are the constituent parts?
  - Objects
  - Region containing pixels of similar properties
  - Contiguous regions perceived by humans




Segmentation is one of the most important steps in image processing
(and one of the most difficult !)

Segmentation splits an image $I$ into regions or subsets $R_i$ with the following properties

- $I = \bigcup_{i=1}^{N} R_i$: all pixels belong to some segment or the other
- $R_i \cap R_j = \phi, i \neq j$: regions do not overlap
- Each $R_i$ is a connected component, i.e., there exists a path that lies entirely within $R_i$ between every pair of pixels

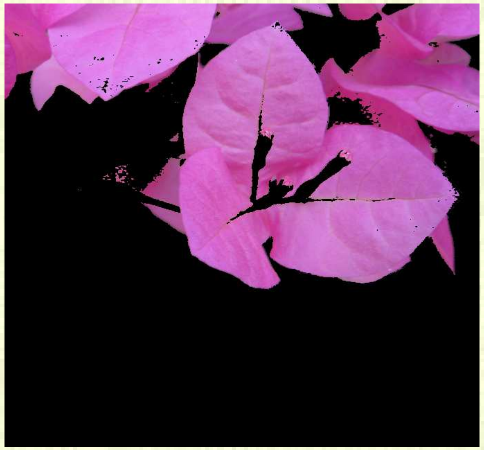
Some important low-level cues

- discontinuity in brightness/colour
- similarity in brightness/colour
- edge densities, gray-level distributions, . . .

## WHAT IS SEGMENTATION?

- The simplest technique — retain pixels having gray levels within a specified range; make all others *black* — separates objects based on brightnesses (or colours)
  - e.g., black text on white paper
  - also, bougainvilla flowers from green leaves (colour ranging)

Original

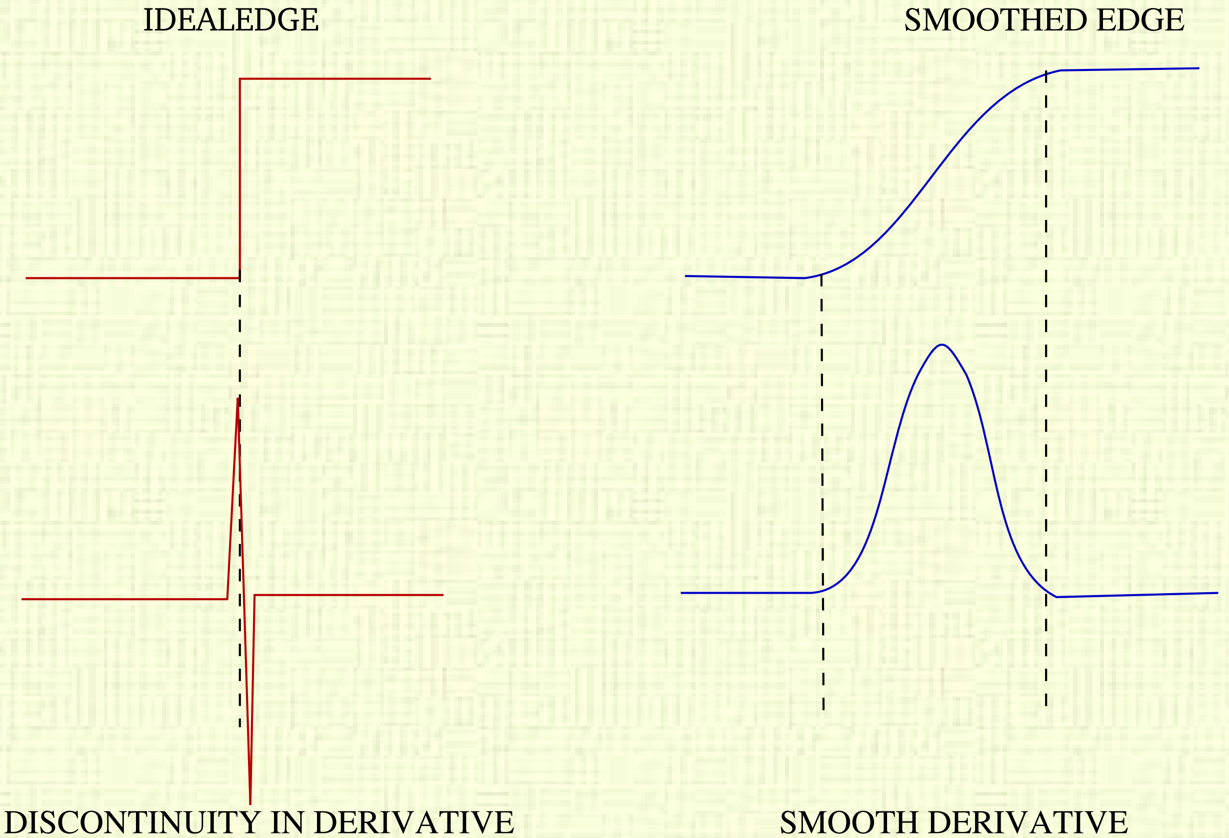


Flowers (Hue $\approx 310^\circ$)

Leaves (Subtract Flowers)

- Detection of discontinuities in intensity function
  - can be discontinuity in any feature value, e.g., frequencies, texture, disparity or depth, etc.
- Already familiar with gradient operators such as Robert's, Sobel's, Prewitt's, etc.
- Let us look at one of the best — Canny's algorithm
  - proposed by John F. Canny (MIT) in 1986
  - extended Marr-Hildreth edge detector by adding a hysteresis component
- Canny's is normally the benchmark for evaluating edge detectors today

- Apply Gaussian filter on original image ($I_0$): makes intensity function continuous and differentiable; gradient operator becomes well-behaved




Let the resultant smoothed image be $I_g$

## CANNY'S EDGE DETECTOR

- Find horizontal $(I_h)$ and vertical gradients $(I_v)$ at each pixel

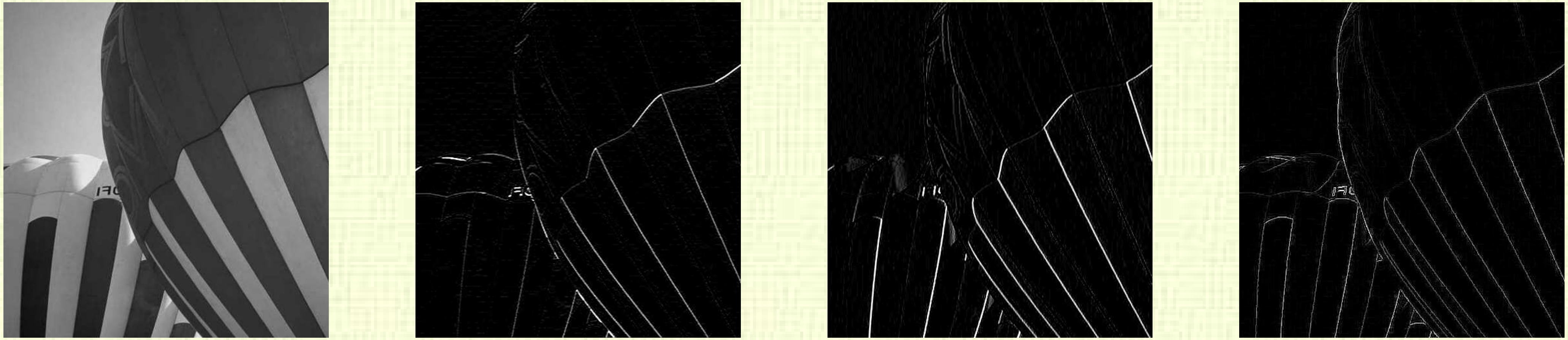
$$\begin{aligned} I_h &= \frac{\partial I_g}{\partial x} \\ &= \frac{\partial}{\partial x}\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}\right) \\ &= -\frac{x}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \frac{x}{\sigma^2} I_g \end{aligned}$$

Similarly, $I_v = \dfrac{\partial I_g}{\partial y} = \dfrac{y}{\sigma^2} I_g$. These are known as DoG operators

- Compute magnitude $(M = \sqrt{I_h^2 + I_v^2})$

orientation $\left(\theta = \arctan\left(\dfrac{I_v}{I_h}\right)\right)$

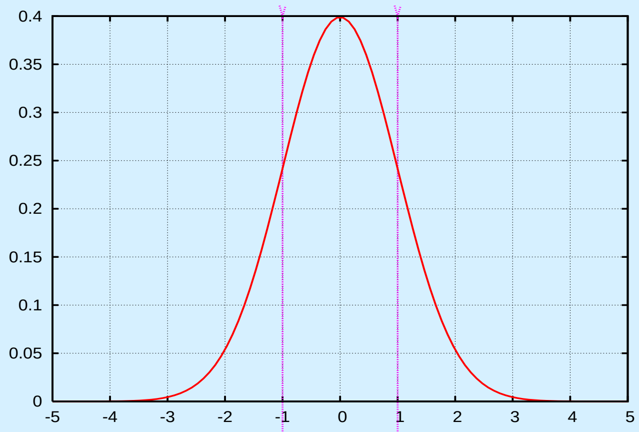
- Find the second derivative (i.e., Laplacian of Gaussian or LoG) $I_L$
- Find the Zero-Crossings in $I_L$; they are the edge locations



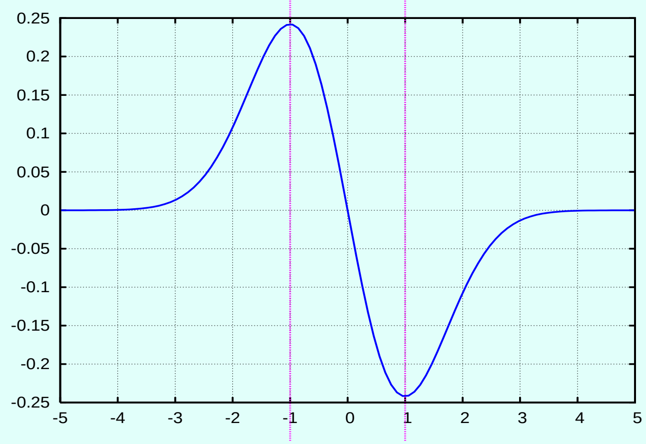
Original $\quad I_h \quad I_v \quad I_L$

- Identify two thresholds $T_1$ and $T_2$ ($T_1 > T_2$)
  - if edge magnitude at a pixel identified as a zero-crossing in $I_L$ is $> T_1$, mark it as an edge
  - if edge magnitude is $< T_2$, mark it as non-edge

## CANNY'S EDGE DETECTOR . . .


Gaussian Filter




DoG Filter

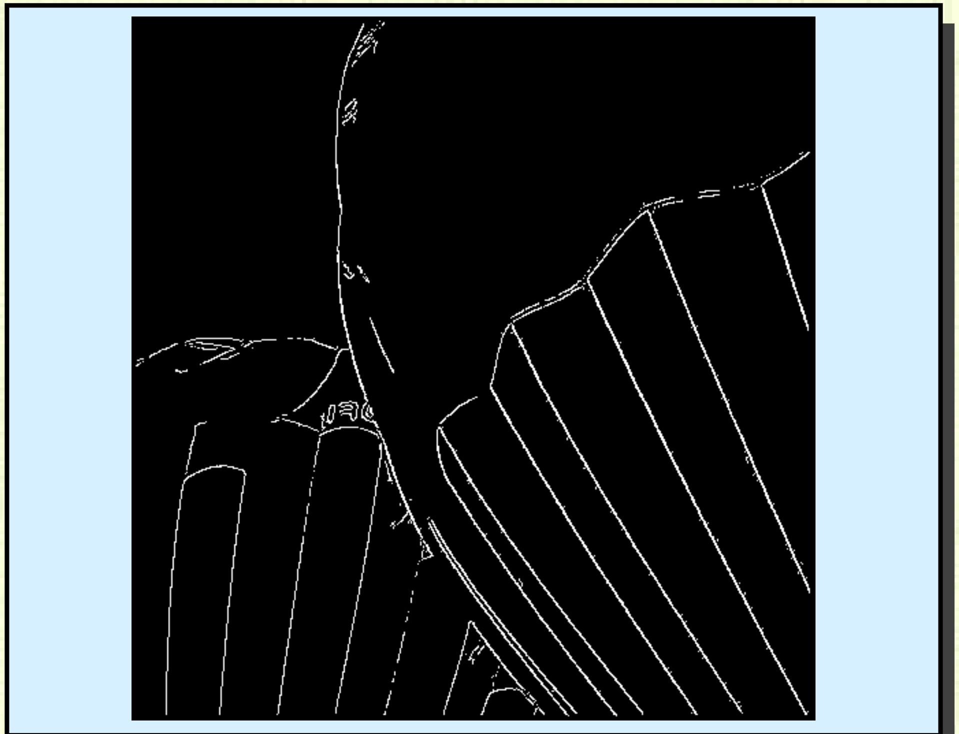
- DoG filter has two humps at $\pm\sigma$
- DoG filter gives peak response at edge; hard to detect in noise
- LoG filter is called Mexican Hat
- LoG filter gives zero response at edge


LoG Filter

# HOW DO DOGs and LOGs LOOK LIKE?


- Edge Linking: For every pixel marked as an edge
  - ○ travel forward along the edge orientation $\theta$ for a distance $= \sigma$
  - ○ mark as an edge pixel if any pixel has an edge magnitude $T_1 > m > T_2$ along the way

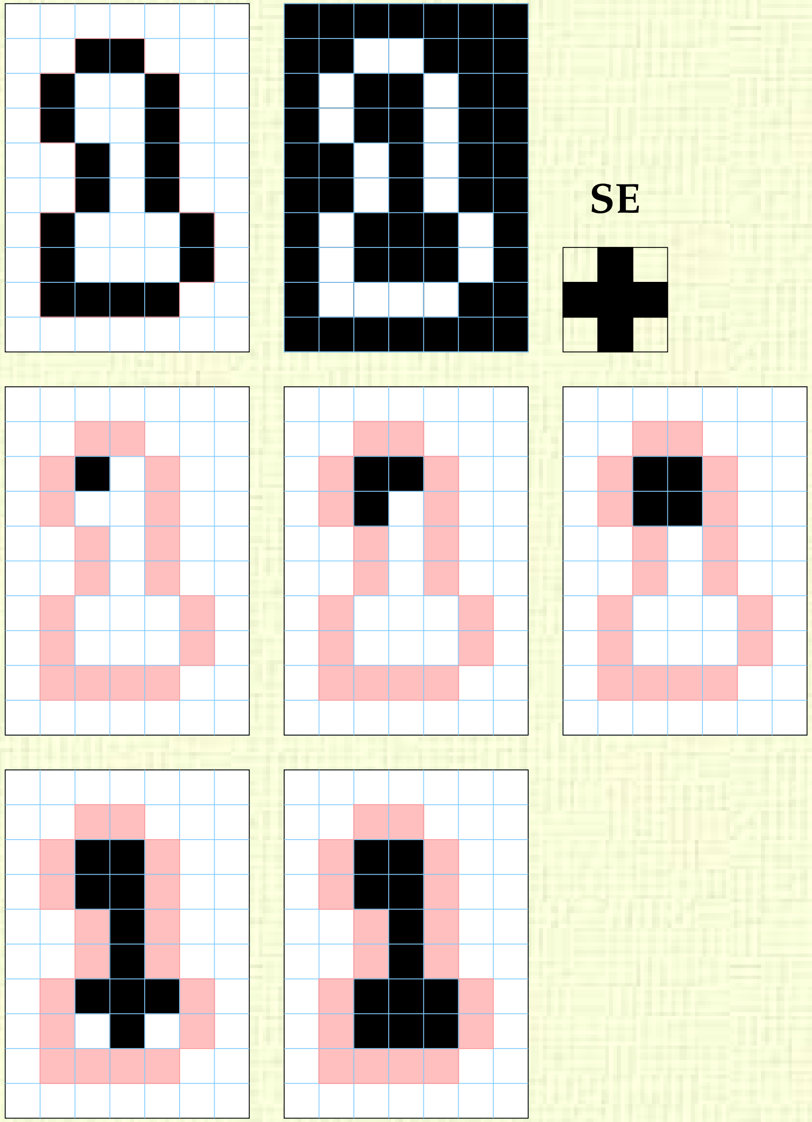
- Non-Max Suppression: For every pixel marked as an edge
  - ○ travel in orthogonal direction for distance $= \sigma$ on both sides
  - ○ find maximum edge magnitude and suppress all pixels with weaker strength
- The end result are ‘clean’ edges that are one pixel wide



BACK TO CANNY’S

- Determine INSIDE and OUTSIDE
- draw an imaginary line from the pixel to the image boundary
- if it intersects edges an odd number of times, it is inside; otherwise it is outside
- Label all the inside pixels uniquely for each region
- Problem: Simple in concept but hard to implement correctly

- Given a set of boundary points, fill the region enclosed by them
- Key idea: conditional dilation
  - Dilated object $A$ is intersected with $A^c$ (prevents from filling entire image)




**Region Filling Algorithm**

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, \ldots$$

$$\text{until } X_k = X_{k-1}$$

$$\text{Finally, Region } = X_K \cup A$$

## REGION FILLING

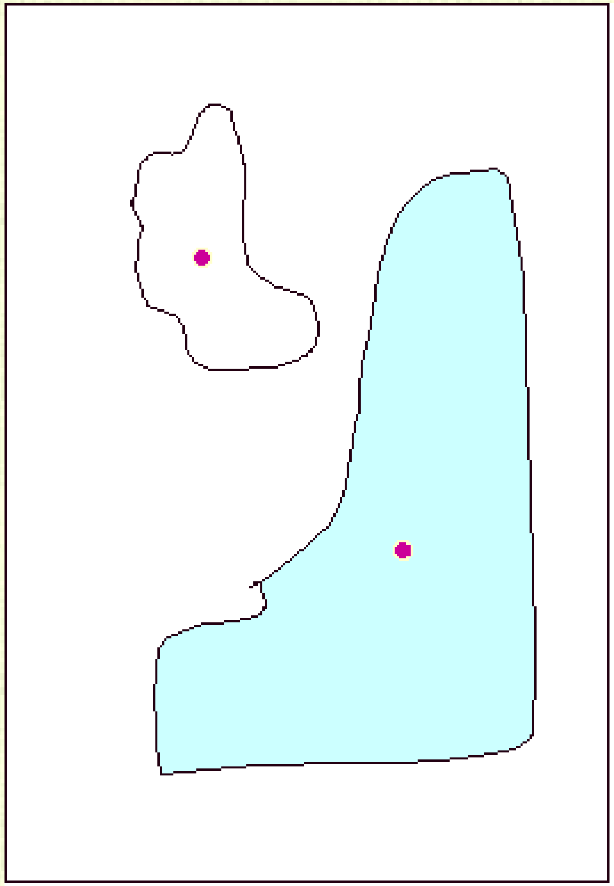
| | | |
|---|---|---|
| $p_1$ | $p_2$ | $p_3$ |
| $p_4$ | $\circ$ | $p_5$ |
| $p_6$ | $p_7$ | $p_8$ |

- Very simple algorithm
- Four cases of 8 – connectedness
- First pass: initial labelling of each foreground pixel
  - if none of $p_1 \ldots p_4$ are labelled, assign a new label
  - if any one of $p_1 \ldots p_4$ are labelled, assign same label
  - if more than one of $p_1 \ldots p_4$ are labelled and their labels are identical, assign same label
  - if more than one of $p_1 \ldots p_4$ are labelled and their labels are different, assign any of the labels
    mark different labels of $p_1 \ldots p_4$ as equivalent
- Second pass: renumbering and merging
  renumber equivalent labels on second pass

- Start with seed pixels
- Append to each seed pixel, all its neighbours with similar properties

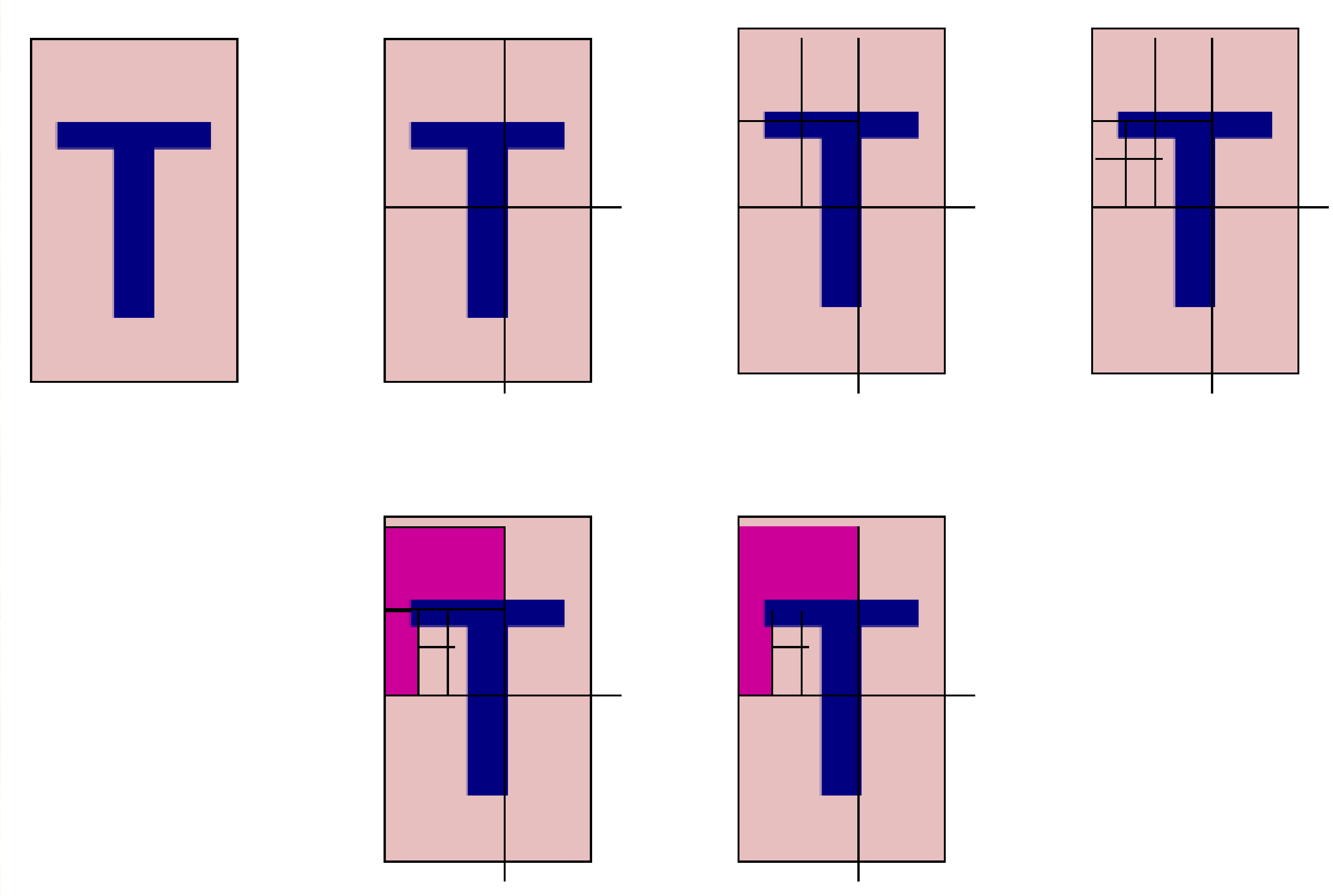
Problems with region growing

- what is a good property?
- how do you determine seed pixels
  - how many
  - where



- Subdivide the image into arbitrary number of sub-regions
  - usually four quadrants
- For each of the sub-regions, check if it is homogeneous (with respect to some property)
- Split Step: If it is not, subdivide the region into four smaller quadrants; repeat Step 2
- Merge Step: For each region, check if any of its adjacent regions have similar properties. If so, merge them

## REGION SPLIT-AND-MERGE



# REGION SPLIT-AND-MERGE. . .

- Basic idea is to group pixels according to some common property
- K-Means is the most popular clustering algorithm
  - Clusters $n$ points into $k$ partitions
  - Inputs: $k$ — the number of clusters
  - Start with $k$ seed points as centroids of $k$ clusters
  - Assign each point to the nearest seed point
  - Recalculate cluster centres after assigning all $n$ points
  - Repeat the above two steps until convergence
- Biggest Problem: knowing $k$
- Variant: Adaptive k-Means

## CLUSTERING TECHNIQUES

Original

6-Clusters

6-Clusters + CC

There are 3535 connected components in the third image!


- The major drawback of K-Means clustering is selecting the number of clusters
- Adaptive K-Means allows us to get an optimal number of clusters
- Start K-Means algorithm with $N_c = 2$
- After the clusters converge, measure their goodness using the following criterion
  $G(n) =$ ratio of cluster size and average intercluster distance
- Increment $N_c$ and repeat the above procedure
- Find the $N_c$ that gives the minimum for $G(n)$

## ADAPTIVE K-MEANS

- The most recent and some of the best performing segmentation algorithms view images as weighted graphs
  - every pixel is initially a node in the graph
  - adjacent pixels are connected by edges
  - edge weights measure the dissimilarity between pixels
- Spanning Tree forms the basis for many algorithms
  - split the graph into two by cutting the largest weight in the Minimum Spanning Tree (MST)
  - use the ratios of the largest weights to determine if neighbourhood is uni, bi, or multimodal
- Today, one of the best algorithms is based on Normalized Cuts

- Reference:

  D. P. Pedro, F. Felzenswalb. "Efficient Graph Based Segmentation," *Int. J. of Computer Vision*, **50**(2):167 – 181, 2004.

  Implemented by S. Bhagyalaxmi (MTech 2008)

- Use cluster uniformity and distance to other clusters as a criterion
  - internal difference of a component

    $$C_{int} = \text{max weight in the MST}$$

  - external difference between two components

    $$C_{ext} = \text{min weight edge between the two components}$$

  - Ratio of the above is used for segmentation

Original

6-Clusters + CC

Graph Output

There are 413 connected components in the third image which is a lot better than plain k-means but shows you how much work is left!

k-Means gave more than 5000 components

## A BETTER RESULT:-)


- Segmentation is easy to conceptualize but extremely difficult to achieve in practice
- Segmentation of simple polygonal objects is doable
- We usually under or over-segment — tigers are particularly notorious :-)
- As listed at the beginning there are many strategies for segmentation but nothing works well
- Finally, let us come to an important question
  Is segmentation a low-level or high-level process?