SCSV3213

FUNDAMENTAL OF IMAGE PROCESSING

SEM2- S0182019

IMAGE SEGMENTATION

Lecture

Dr. Md Sah Hj Salam

Acknowledgements

Most of the slide are taken and modified from other resources including books and slides from lectures from others universities especially from Oge Marques. It is rearranged to suit the syllabus of the course.

What will we learn?

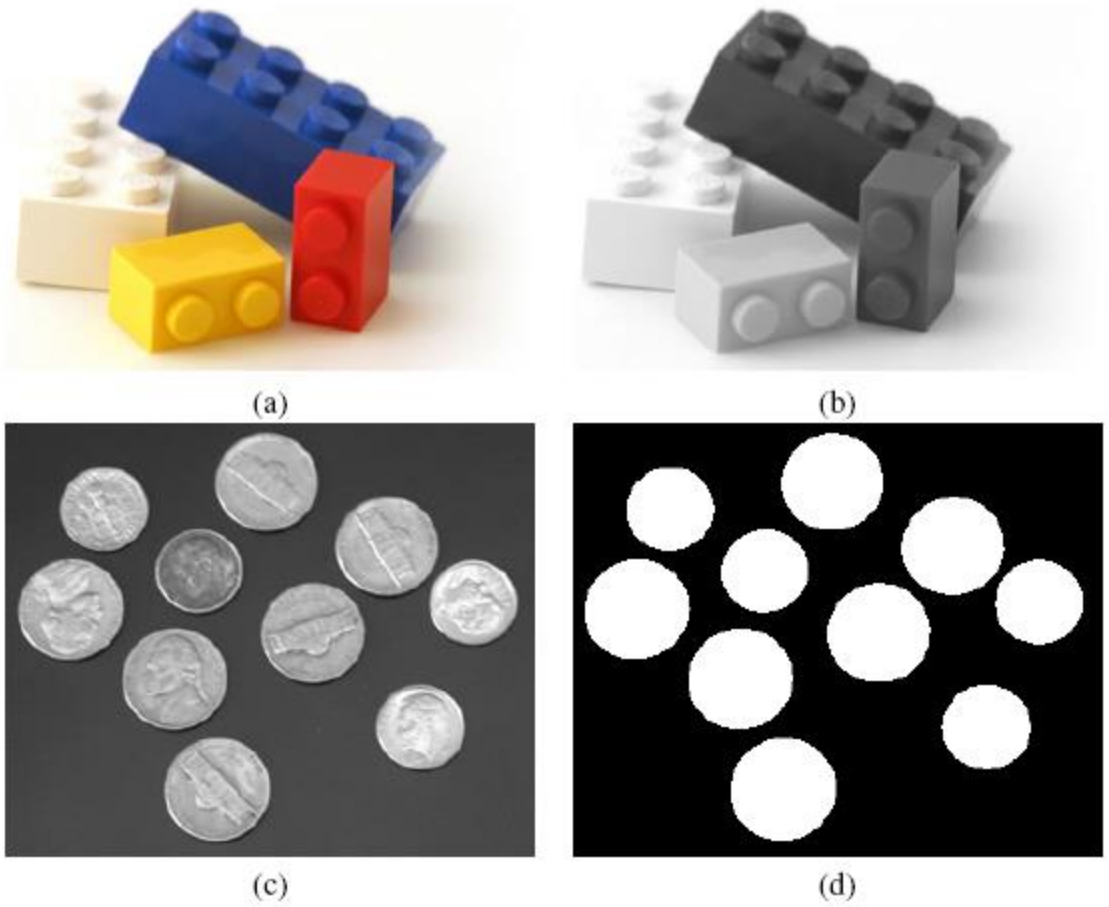
- What is image segmentation and why is it relevant?
- What is image thresholding and how is it implemented in MATLAB?
- What are the most commonly used image segmentation techniques and how do they work?

Introduction

- Segmentation: the process of partitioning an image into a set of non-overlapping regions whose union is the entire image.
  - These regions should ideally correspond to objects and their meaningful parts, and background.
- Most image segmentation algorithms are based on one of two basic properties that can be extracted from pixel values -- discontinuity and similarity -- or a combination of them.
- Segmentation is one of the most crucial tasks in image processing and computer vision.
  - And it's still an open research problem.

Introduction

- Segmentation: easy and hard images



Introduction

- There is no underlying theory of image segmentation, only *ad hoc* methods, whose performance is often evaluated indirectly, based on the performance of the larger system to which they belong.

- Image segmentation techniques can vary widely according to:
  - type of image (e.g., binary, gray, color)
  - choice of mathematical framework (e.g., morphology, image statistics, graph theory)
  - type of features (e.g., intensity, color, texture, motion) and
  - approach (e.g., top-down, bottom-up, graph-based).

Introduction

- There is no universally accepted taxonomy for classification of image segmentation algorithms, either.
- In the book, we use three categories:
  - Intensity-based (non-contextual) methods: work based on pixel distributions (i.e., histograms).
    - Example: thresholding.
  - Region-based (contextual) methods: rely on adjacency and connectivity criteria between a pixel and its neighbors.
    - Examples: region growing and split-and-merge.
  - Other methods: these include segmentation based on texture, edges, and motion, among others.

Intensity-based segmentation

- Rely on pixel statistics (histogram properties) to determine which pixels belong to foreground objects and which pixels should be labeled as background.

- The simplest method within this category is *image thresholding*.

Image thresholding

- Def.: The conversion of an image with many gray levels into another image with fewer gray levels, usually only two.
  - This conversion is usually performed by comparing each pixel intensity against a reference value (*threshold*) and replacing the pixel with a value that means 'white' or 'black' depending on the outcome of the comparison.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{otherwise} \end{cases}$$

Image thresholding

- Global thresholding: single value of *T* for the whole image.
  - Steps:
    1. Inspect the image's histogram (using **`imhist`**).
    2. Select an appropriate value for *T*.
    3. Apply the selected value (using **`im2bw`**) to the image.
    4. Inspect the results: if they are acceptable, save resulting image; otherwise, make adjustments and repeat steps 2-4.

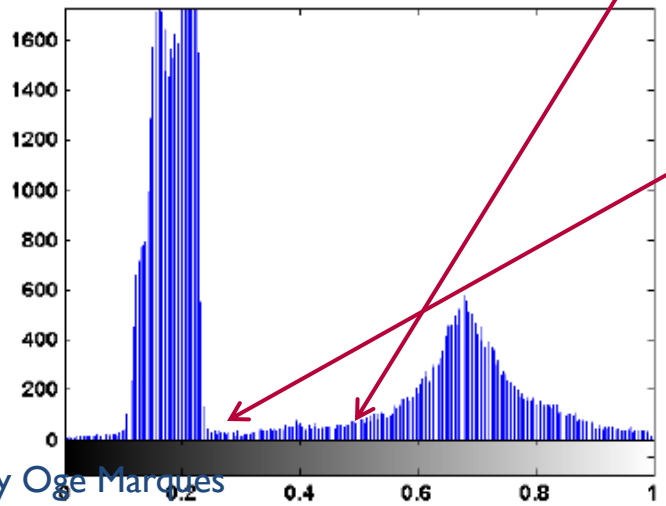
Image thresholding

- Global thresholding: a procedure for selecting the best value of *T* automatically (Gonzalez & Woods, 2008).

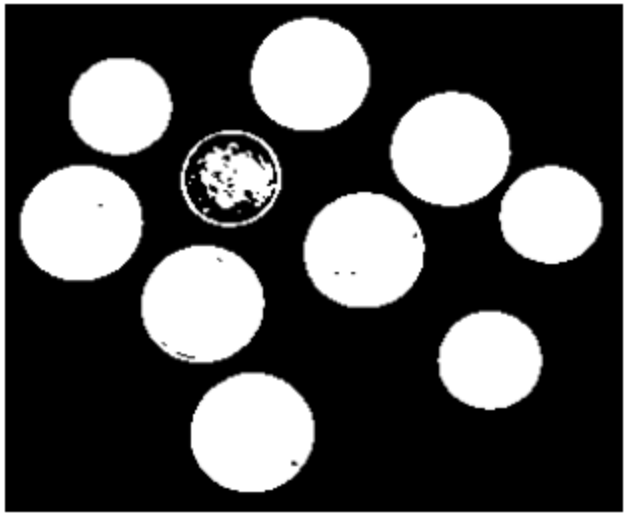
```
Id = im2double(I); % I is a uint8 grayscale image
T = 0.5*(min(Id(:)) + max(Id(:)));
deltaT = 0.01; % convergence criterion
done = false;
while ~done
      g = Id >= T;
      Tnext = 0.5*(mean(Id(g)) + mean(Id(~g)));
      done = abs(T - Tnext) < deltaT;
      T = Tnext;
end
```

Image thresholding

- Global thresholding example



T = 0.4947



T = 0.25

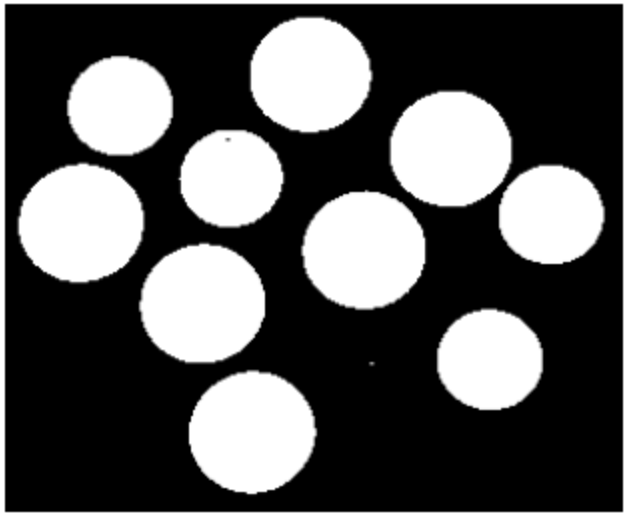
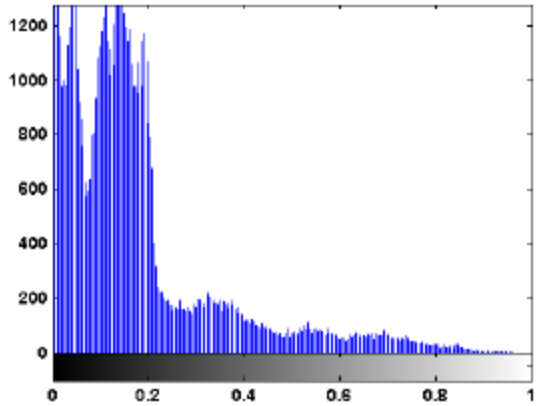


Image thresholding

- Optimal thresholding
  - (Otsu, 1979), implemented in the IPT as: **`graythresh`**.
  - For the previous example, provides the optimal value
    $T = 0.4941$

Image thresholding

- The role of illumination and noise
  - Even easy images may become hard to segment…



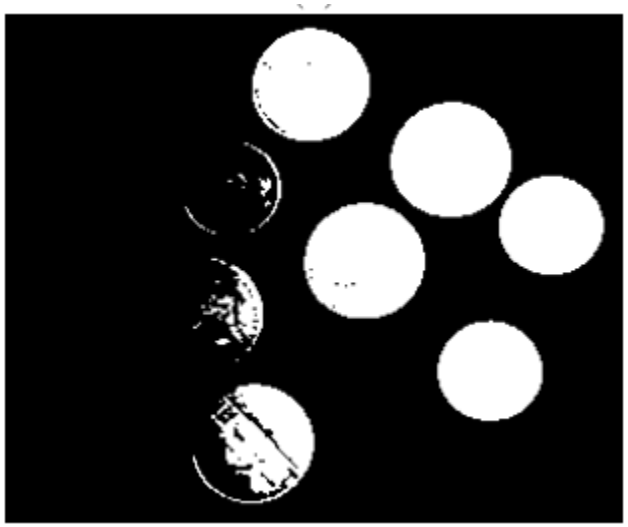
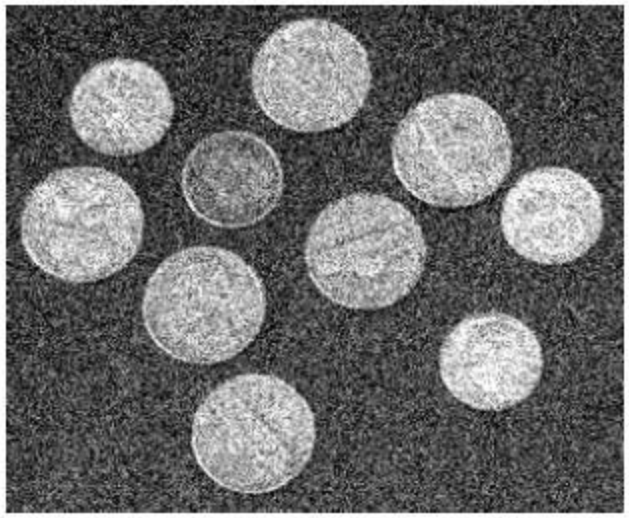
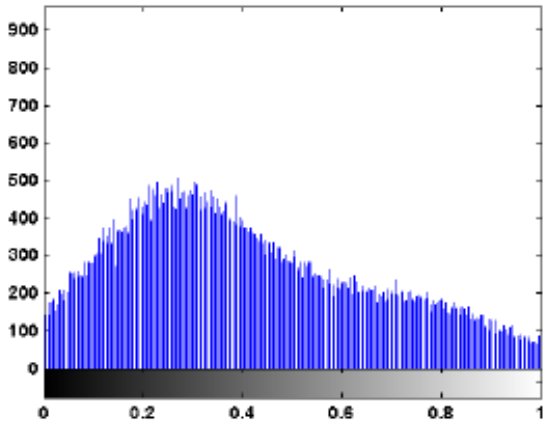


Image thresholding

- The role of illumination and noise
  - Even easy images may become hard to segment…





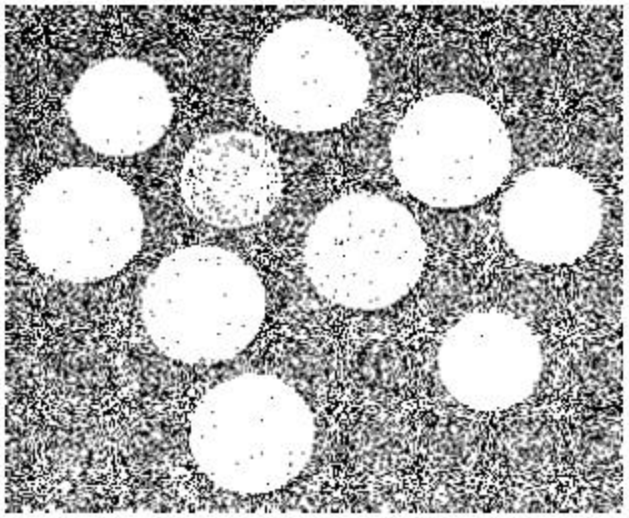
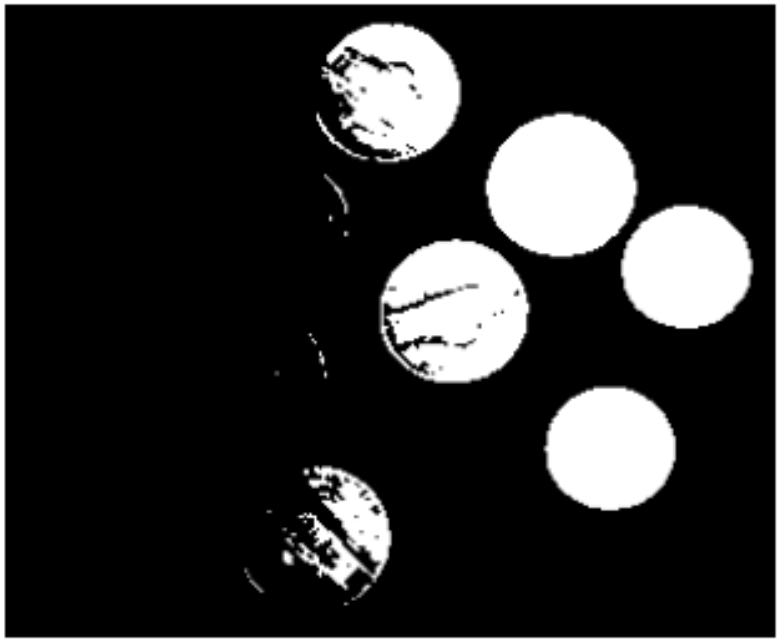


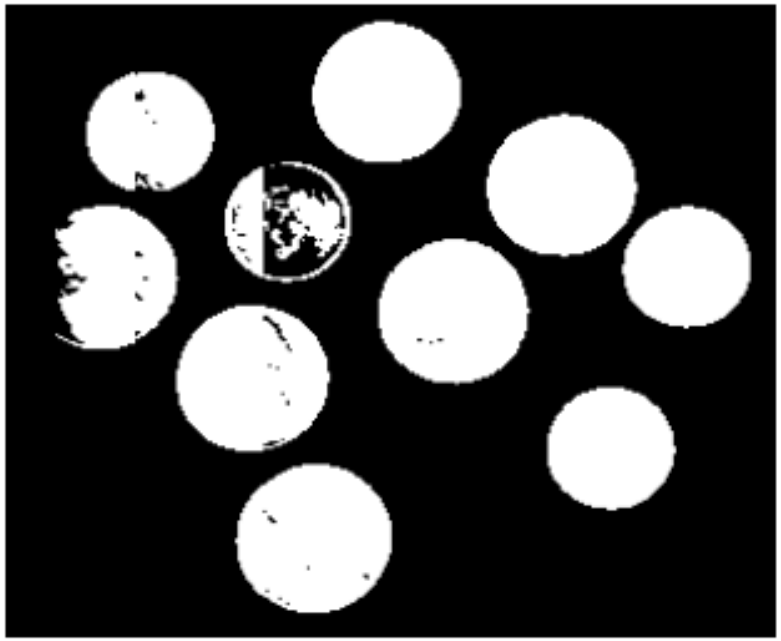
Image thresholding

- Local thresholding
  - Local (adaptive) thresholding uses block processing to threshold blocks of pixels, one block at a time.
    - The size of the block is usually specified by the user.
    - Trade-off:
      - If the blocks that are too small: large computational cost.
      - If the blocks that are too large: results may not be substantially better than the ones obtained with global thresholding.

Image thresholding

- Local thresholding – example





Region-based segmentation

- Based on the fact that a pixel cannot be considered a part of an object or not based solely on its gray value.
- Incorporate measures of connectivity among pixels in order to decide whether those pixels belong to the same region (or object) or not.

Region-based segmentation

- Mathematically, they divide an image *I* into *n* regions R_1, R_2, …, R_n such that:

1. $\bigcup_{i=1}^{n} R_i = I$
2. R_i is a connected region, $i = 1, 2, \cdots, n$.
3. $R_i \cap R_j = \emptyset$ for all i and j, $i \neq j$.
4. $P(R_i)$ = TRUE for $i = 1, 2, \cdots, n$.
5. $P(R_i \cup R_j)$ = FALSE for any adjacent regions R_i and R_j.

where $P(R_i)$ is a logical predicate defined over the points in set R_i and $\emptyset$ is the empty set.

Region-based segmentation

- Logical predicates (also called *homogeneity criteria*) include:
  - Pure uniformity
  - Local mean relative to global mean
  - Local standard deviation relative to global mean
  - Variance
  - Texture

Region-based segmentation

- Region growing – key factors:
  - Choice of **similarity criteria**: for monochrome images, regions are analyzed based on intensity levels and connectivity properties.
  - Selection of **seed points**: these can be determined interactively (if the application allows) or based on a preliminary cluster analysis of the image, used to determine groups of pixels that share similar properties, from which a seed can be chosen.
  - **Stopping rule**: a region should stop growing when there are no further pixels that satisfy the homogeneity and connectivity criteria to be included in that region.

Region growing algorithm

```
Let f(x,y) be the input image
Define a set of regions R1, R2, ..., Rn, each consisting of a
          single seed pixel
repeat
   for i = 1 to n do
      for each pixel p at the border of Ri do
         for all neighbors of p do
            Let (x,y) be the neighbor's coordinates
            Let Mi be the mean gray level of pixels in Ri
            if the neighbor is unassigned and
                        |f(x,y) - Mi| <= Delta then
               Add neighbor to Ri
               Update Mi
            end if
         end for
      end for
   end for
until no more pixels can be assigned to regions
```

Region-based segmentation

- Region growing - example

$$P(R_i) = \begin{cases} \text{TRUE} & \text{if } |f(x,y) - \mu_i| \leq \Delta \\ \text{FALSE} & \text{otherwise} \end{cases}$$

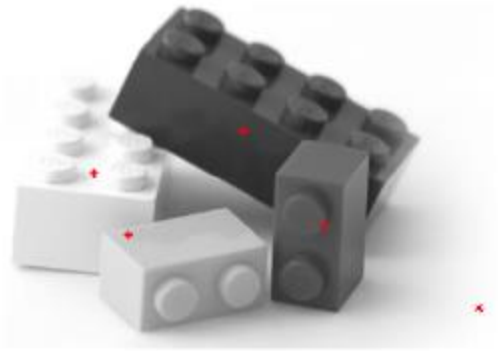
6	7	7	6	5
7	7	8	6	5
5	5	6	7	6
0	1	2	0	1
1	0	0	2	0

6	7	7	6	5
7	7	8	6	5
5	5	6	7	6
0	1	2	0	1
1	0	0	2	0

6	7	7	6	5
7	7	8	6	5
5	5	6	7	6
0	1	2	0	1
1	0	0	2	0

Region-based segmentation

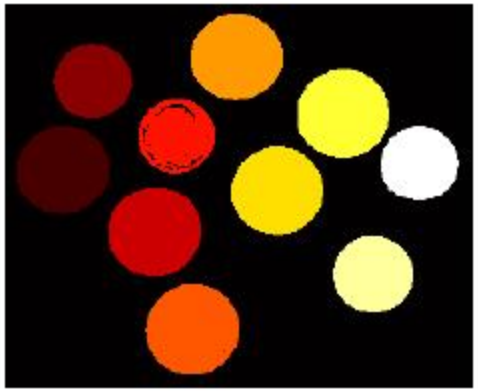
- Region growing: additional examples



(a)

(b)

(c)



(d)

Region-based segmentation

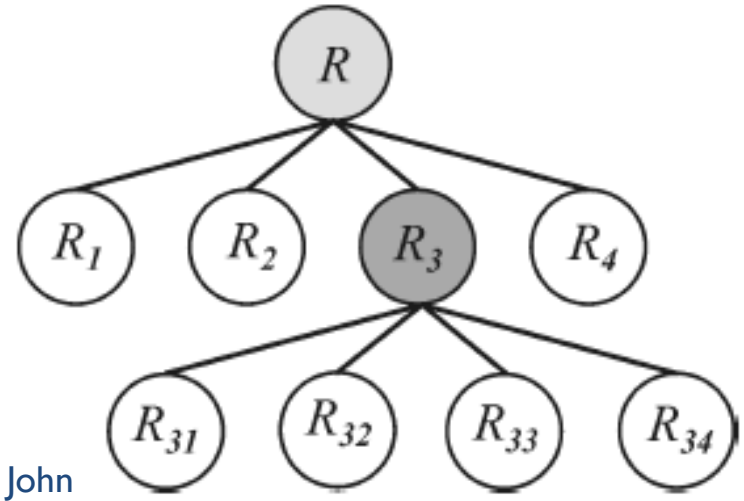
- Region growing – limitations:
  - Not very stable: significantly different results are obtained when switching between 4-connectivity and 8-connectivity criteria.
  - Segmentation results are very sensitive to choice of logical uniformity predicate.
  - The number of seeds provided by the user may not be sufficient to assign every pixel to a region.
  - If two or more seeds that should belong to the same region are incorrectly provided to the algorithm, it will be forced to create distinct regions around them whereas only one region should exist.

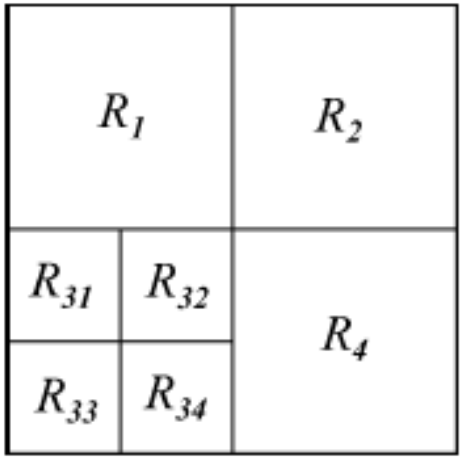
Region-based segmentation

- Region splitting and merging
  - Start from the entire image and partition (***split***) it into smaller subimages until each resulting region is considered homogeneous by some criterion.
  - ***Merge*** two or more adjacent regions into one if they satisfy a homogeneity criterion.

Region-based segmentation

- Region splitting and merging
  - Data structure: *quadtree*, a special type of tree in which each node (except for the leaves) has four children.
  - Each leaf node in the quadtree corresponds to a region in the segmented image





Region-based segmentation

- Region splitting and merging

  – Algorithm

1. Define a logical uniformity predicate $P(R_i)$.
2. Compute $P(R_i)$ for each region.
3. Split into four disjoint quadrants any region R_i for which $P(R_i)$ = FALSE.
4. Repeat steps 2 and 3 until all resulting regions satisfy the uniformity criterion, i.e., $P(R_i)$ = TRUE.
5. Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k)$ = TRUE.
6. Repeat step 5 until no further merging is possible.

Region-based segmentation

- Morphological image segmentation algorithm (watershed)
  - Used to represent regions in a segmented image (equivalent to *catchment basins*) and the boundaries among them (analogous to the *ridge lines*).
  - In MATLAB: **`watershed`**

Region-based segmentation

- The distance transform
  - Computes the distance from every pixel to the nearest nonzero-valued pixel.
  - In MATLAB: **`bwdist`**

```
a =

     0     1     1     0     1
     1     1     1     0     0
     0     0     0     1     0
     0     0     0     0     0
     0     1     0     0     0
```

```
>> b = bwdist(a,'cityblock')

b =

     1     0     0     1     0
     0     0     0     1     1
     1     1     1     0     1
     2     1     2     1     2
     1     0     1     2     3
```

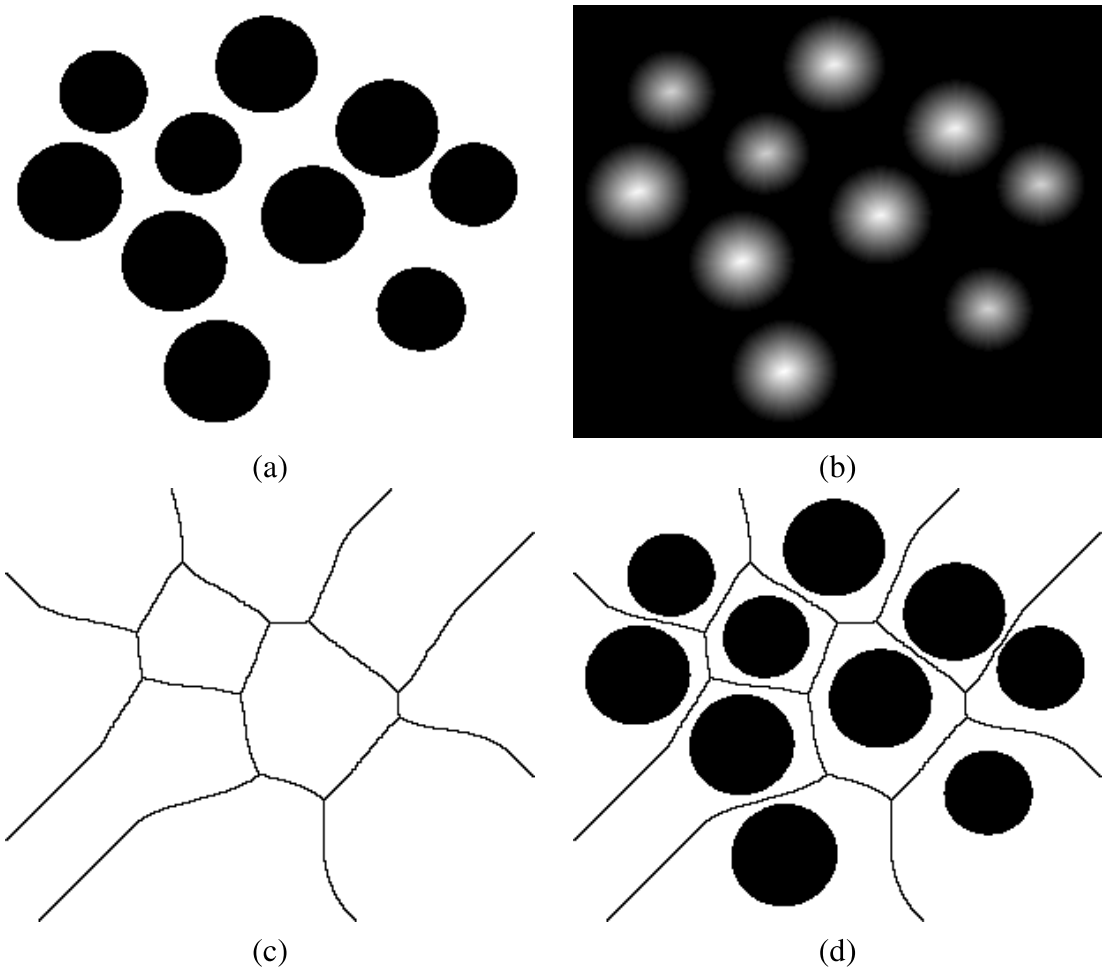
Region-based segmentation

- Morphological image segmentation algorithm (watershed) – Example 15.5

```
J = imread('Figure15_10_a.png');
Jc = ~J;
D = bwdist(Jc);
L = watershed(~D);
w = L==0;
J2 = J & ~w;
J3 = J | w;
imshow(Jc), figure, imshow(~w)
figure, imshow(mat2gray(D))
figure, imshow(~J3)
```

Region-based segmentation

- Morphological image segmentation algorithm (watershed) – Example 15.5



TUTORIAL ON SEGMENTATION

TUTORIAL HEURISTIC THRESHOLDING

- Heuristic thresholding

```
>> I = imread('coins.png');
>> figure, imshow(I), title('original image');
>> figure, imhist(I), title('Histogram of Image');
>> T=85; I_tresh=im2bw(I,(T/255));
>> figure, imshow(I_tresh), title('Threshold image');
```

- Using graythresh( ) function for thresholding

```
>>T2=graythresh(I);
>> I_tresh2=im2bw(I,T2);
>> figure, imshow(I_tresh2), title('Threshold image');
```

TUTORIAL ADAPTIVE THRESHOLDING

- Issue using Global thresholding

```
>>I=imread('gradient_with_text.tif');
>> figure, imshow(I), title('original image');
>> I_gthresh=im2bw(I,graythresh(I));
>> figure, imshow(I_gthresh), title('Global Threshold image');
>> figure, imhist(I), title('Histogram of original Image');
```

- Creating adapt_thresh(x) function for thresholding

```
function y = adapt_thresh(x)
  y=im2bw(x,graythresh(x));
end
```

- Call adapt_thresh(x) function to get adaptive threhold via 10x10 pixel block

```
>>I_thresh=blkproc(I,[10,10],@adapt_thresh);
>> figure,subplot(1,2,1),imshow(I),title('original image');
>> subplot(1,2,2),imshow(I_thresh),title('Adaptive thresholding');
```

TUTORIAL ADAPTIVE THRESHOLDING

- Improving adaptive thresholding. Find std in the are of text and non-text

```
>>std_without_text = std2(I(1:10, 1:10));
>> std_with_text = std2(I(100:110, 100:110));
```

- CHange adapt_thresh(x) function using std

```
function y = adapt_thresh2(x)
if std2(x)<1
   y=ones(size(x,1),size(x,2));
else
  y=im2bw(x,graythresh(x));
end
end
```

- Call adapt_thresh2(x) function to get adaptive threhold via 10x10 pixel block

```
>>I_thresh=blkproc(I,[10,10],@adapt_thresh2);
>> figure,subplot(1,2,1),imshow(I),title('original image');
>> subplot(1,2,2),imshow(I_thresh),title('Adaptive thresholding');
```

End :
SCSV 3213