Industrial Computer Vision

- Image Segmentation

6th lecture, 2022.10.12

Lecturer: Youngbae Hwang

Mid-term Project

- 현업에서 컴퓨터비전 관련 문제를 정하고 관련 데이터베이스를 수집한 후에 이에 대한 초기 프로젝트를 진행함
- 오늘 강의까지 배운 내용들을 활용해서 키보드/마우스 입력, 영상의 **화질을 개선**하거나 영상에서 **edge**, **boundary**를 추출, 영상 **Segmentation** 하는 알고리즘을 적용
- 다다음주(10/26) 강의시간에 각자 5page 내외로 발표
- 발표자료와 소스코드를 e-campus를 통해 제출

Contents

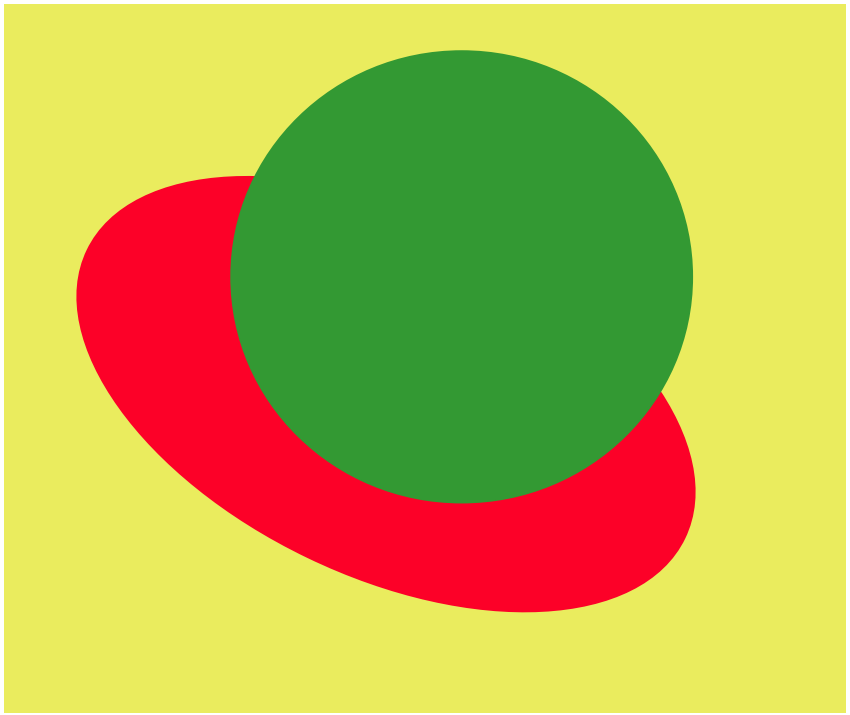
- Region-based growing
- Watershed
- Image Segmentation using K-means clustering
- GrabCut

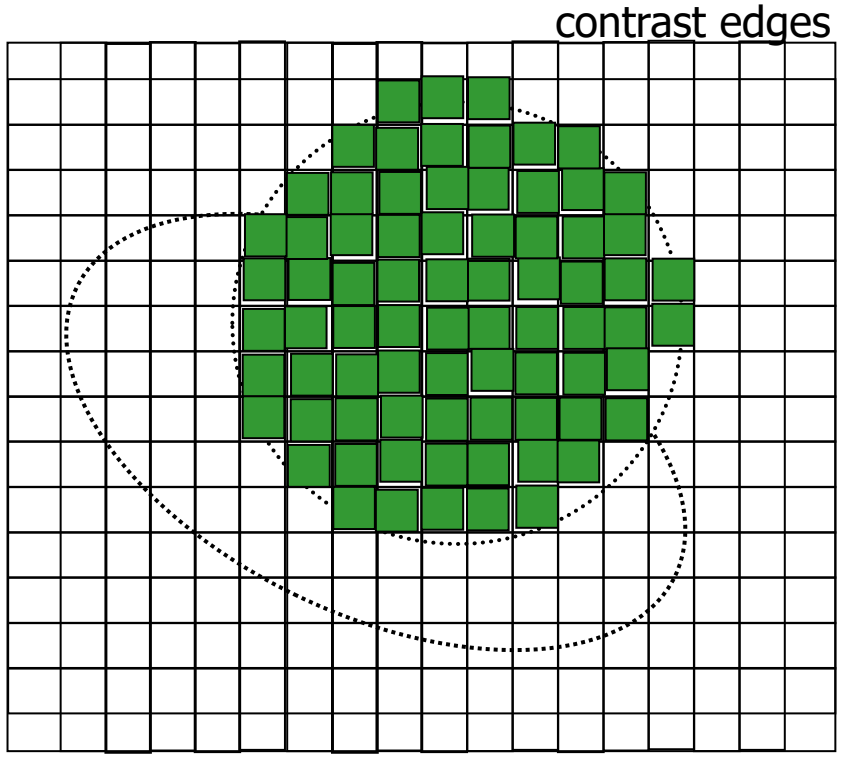
Region-Based Segmentation

- Edges and thresholds sometimes do not give good results for segmentation.
- Region-based segmentation is based on the connectivity of similar pixels in a region.
  - Each region must be uniform.
  - Connectivity of the pixels within the region is very important.
- There are two main approaches to region-based segmentation: region growing and region splitting.

Region growing

- Method stops at contrast edges





- Start with initial set of pixels K (initial seed(s))
- Add to pixels p in K their neighbors q if $|Ip-Iq| < T$
- Repeat until nothing changes

Region-Based Segmentation: Basic Formulation

- Let R represent the entire image region.
- Segmentation is a process that partitions R into subregions, R1,R2,…,Rn, such that

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

(b) R_i is a connected region, $i = 1,2,\ldots,n$

(c) $R_i \cap R_j = \phi$ for all i and $j, i \neq j$

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

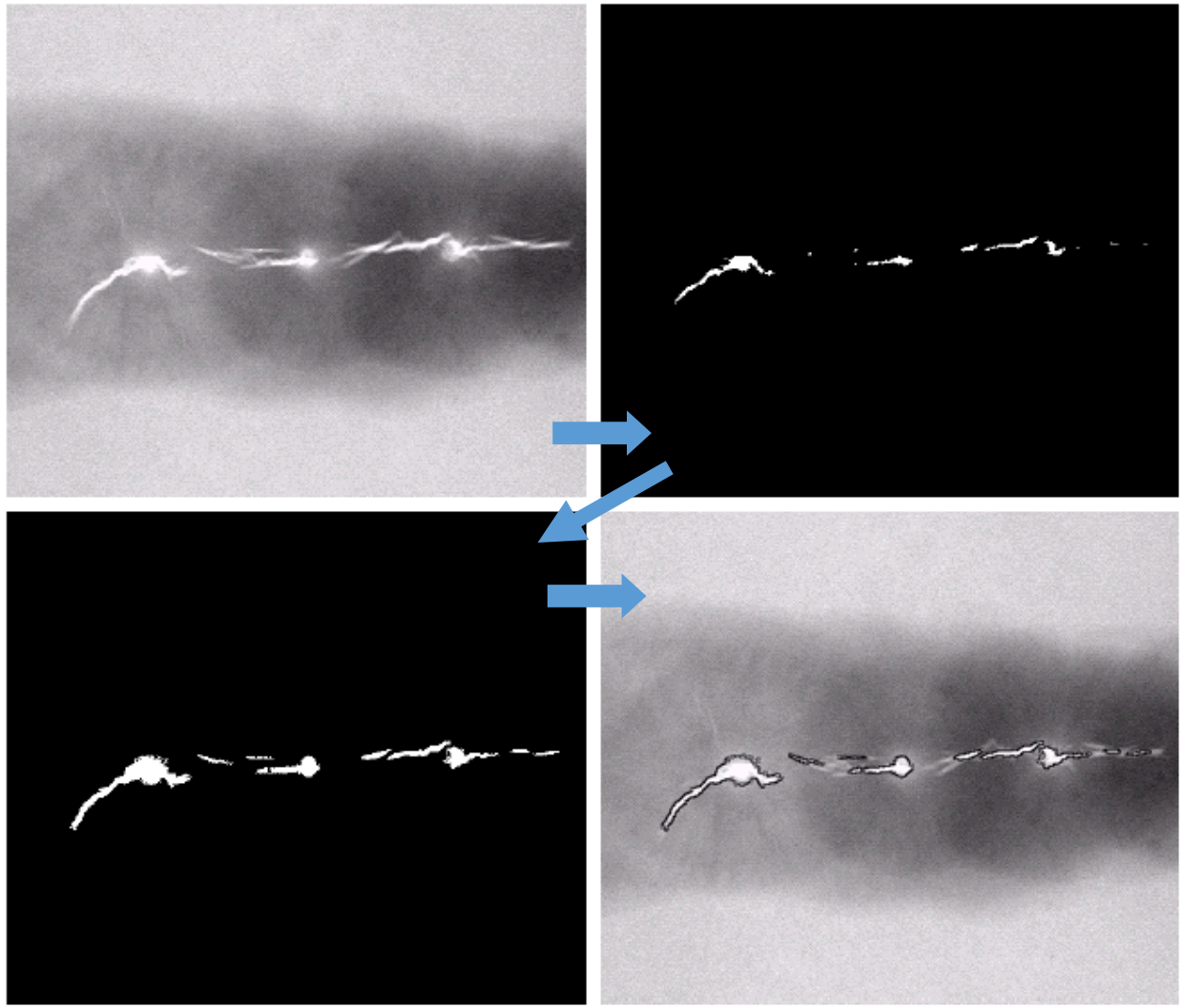
(e) $P(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and R_j

- where $P(R_k)$: a logical predicate defined over the points in set R_k
- For example: $P(R_k)$=TRUE if all pixels in R_k have the same gray level.

Region-Based Segmentation: Region Growing

a b
c d

FIGURE 10.40
(a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).



Region-Based Segmentation: Region Growing

- Fig. 10.41 shows the histogram of Fig. 10.40 (a). It is difficult to segment the defects by thresholding methods. (Applying region growing methods are bet ter in this case.)

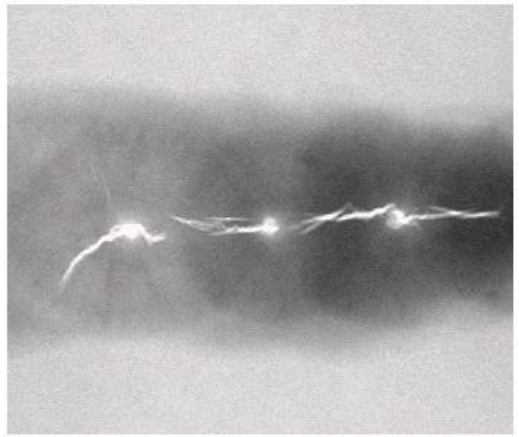


Figure 10.40(a)

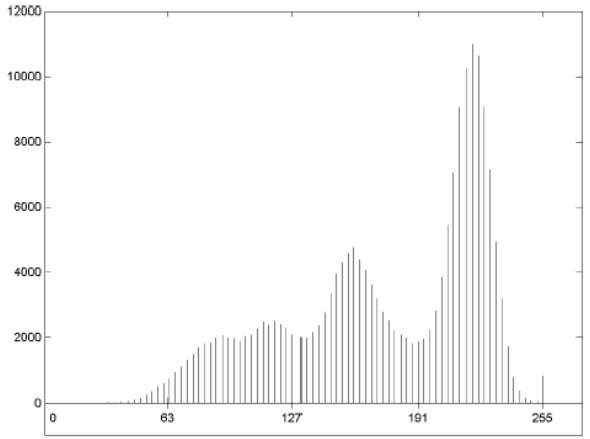
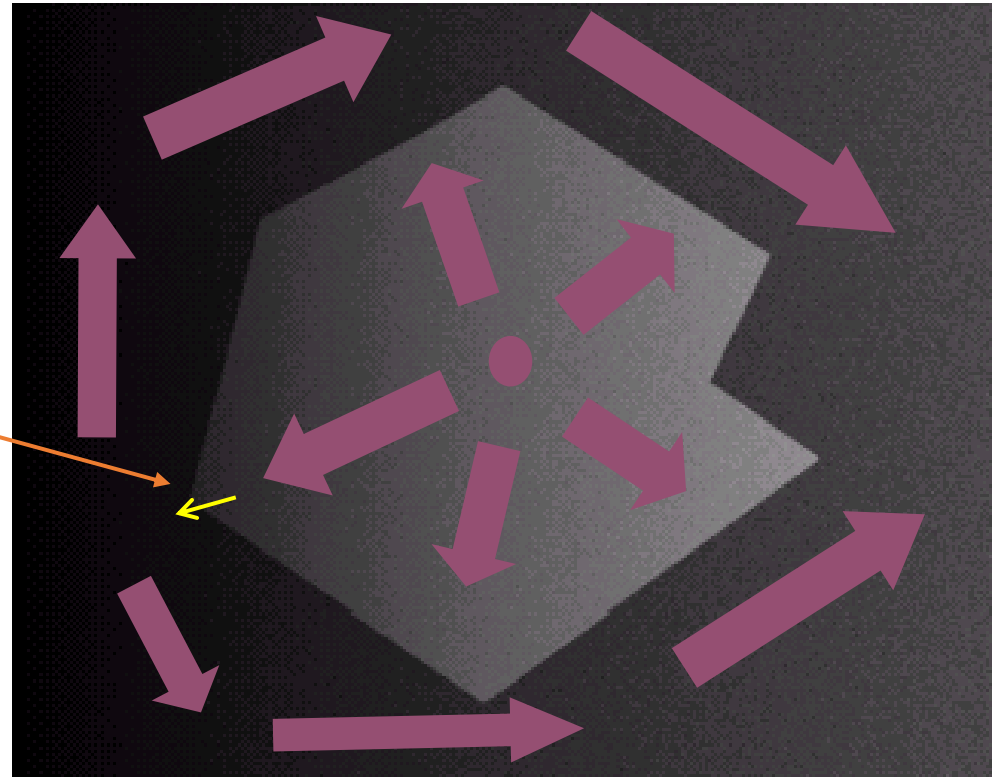


Figure 10.41

What can go wrong with region growing ?

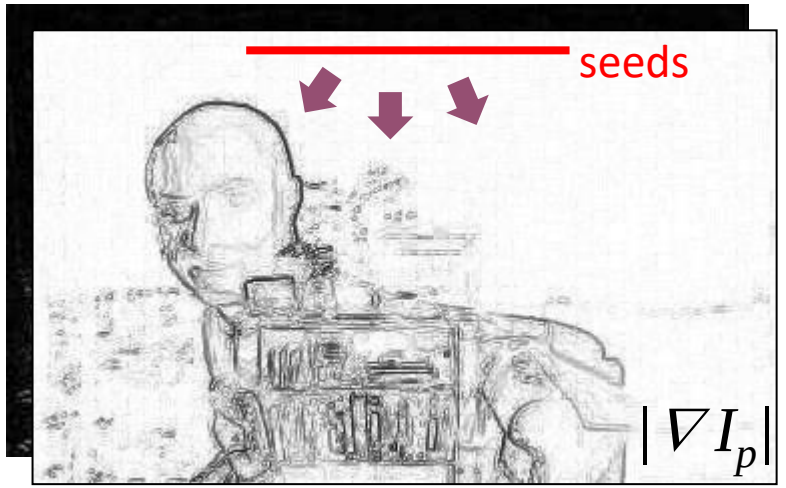
Region growth may “leak” through a single week spot in the boundary

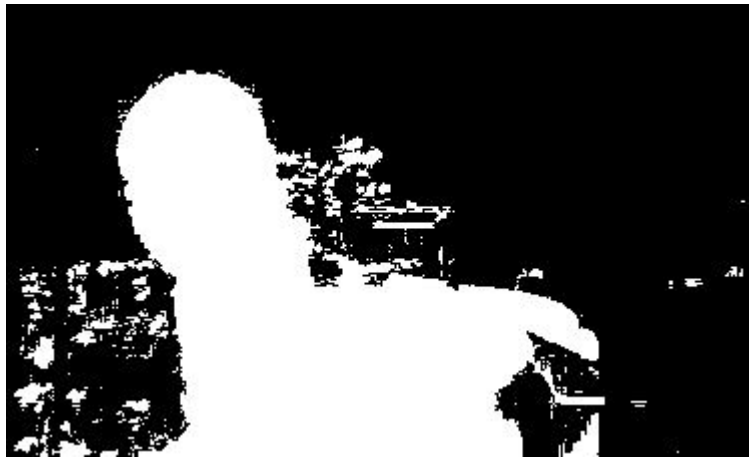


Region growing

-

=





Breadth First Search (seeds) :

$$|\nabla I_p| < T$$

Region growing

See region *leaks* into sky due to a weak boundary between them

Region-Based Segmentation: Region Splitting and Merging

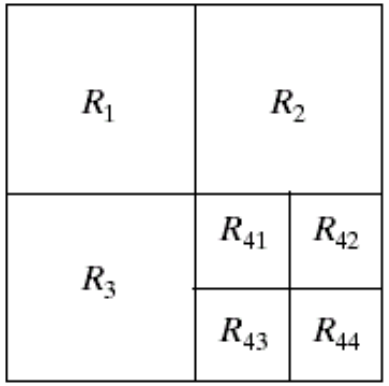
- Region splitting is the opposite of region growing.
  - First there is a large region (possible the entire image).
  - Then a predicate (measurement) is used to determine if the region is uniform.
  - If not, then the method requires that the region be split into two regions.
  - Then each of these two regions is independently tested by the predicate (measuremen t).
  - This procedure continues until all resulting regions are uniform.

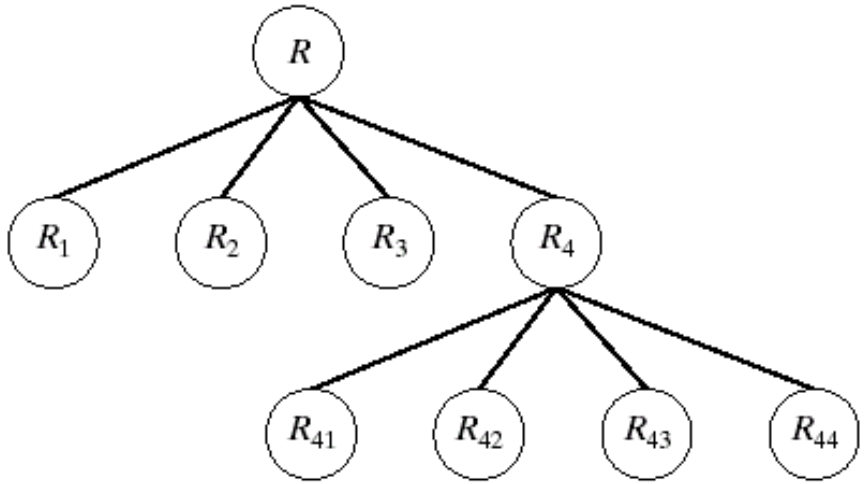
Region-Based Segmentation: Region Splitting

- The main problem with region splitting is determining where to split a region.
- One method to divide a region is to use a quadtree structure.
- Quadtree: a tree in which nodes have exactly four descendants.

a b

FIGURE 10.42
(a) Partitioned image.
(b) Corresponding quadtree.



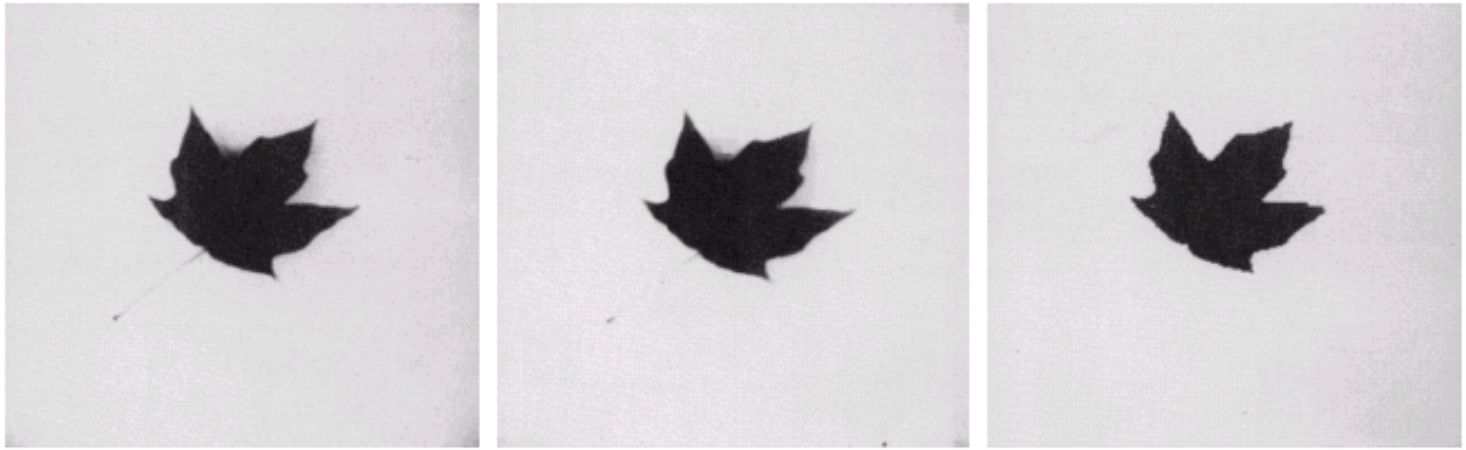


Region-Based Segmentation: Region Splitting and Merging

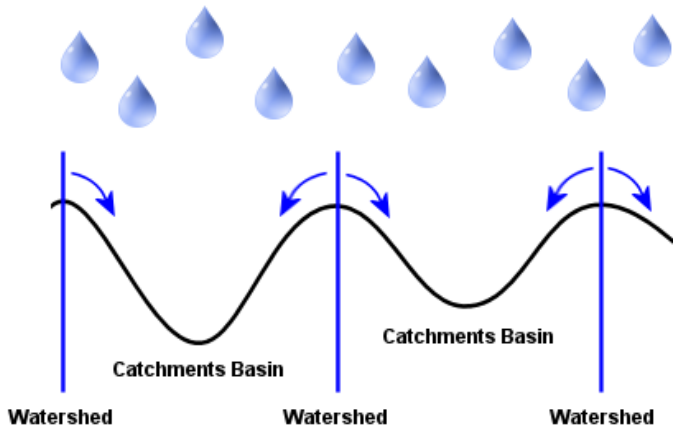
- The split and merge procedure:
  - Split into four disjoint quadrants any region R_i for which $P(R_i)$ = FALSE.
  - Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k)$ = TRUE. (the quadtree structur e may not be preserved)
  - Stop when no further merging or splitting is possible.

a b c

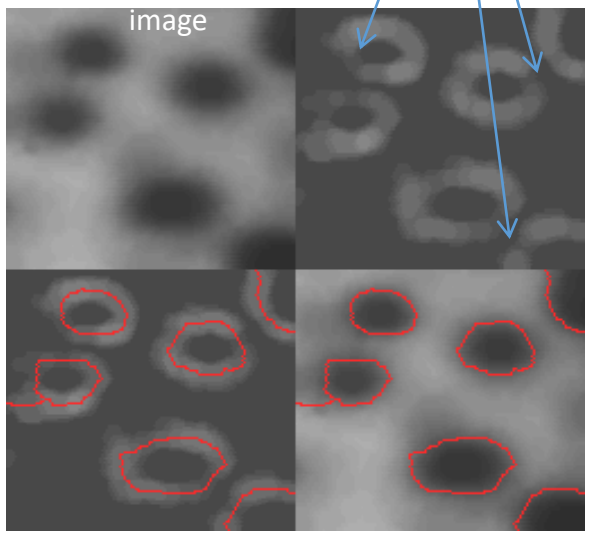
FIGURE 10.43
(a) Original image. (b) Result of split and merge procedure. (c) Result of thresholding (a).



Watersheds



need tricks to build dambs
closing gaps



1. gradient magnitudes

2. find catchmen t basins

3. copy over original image

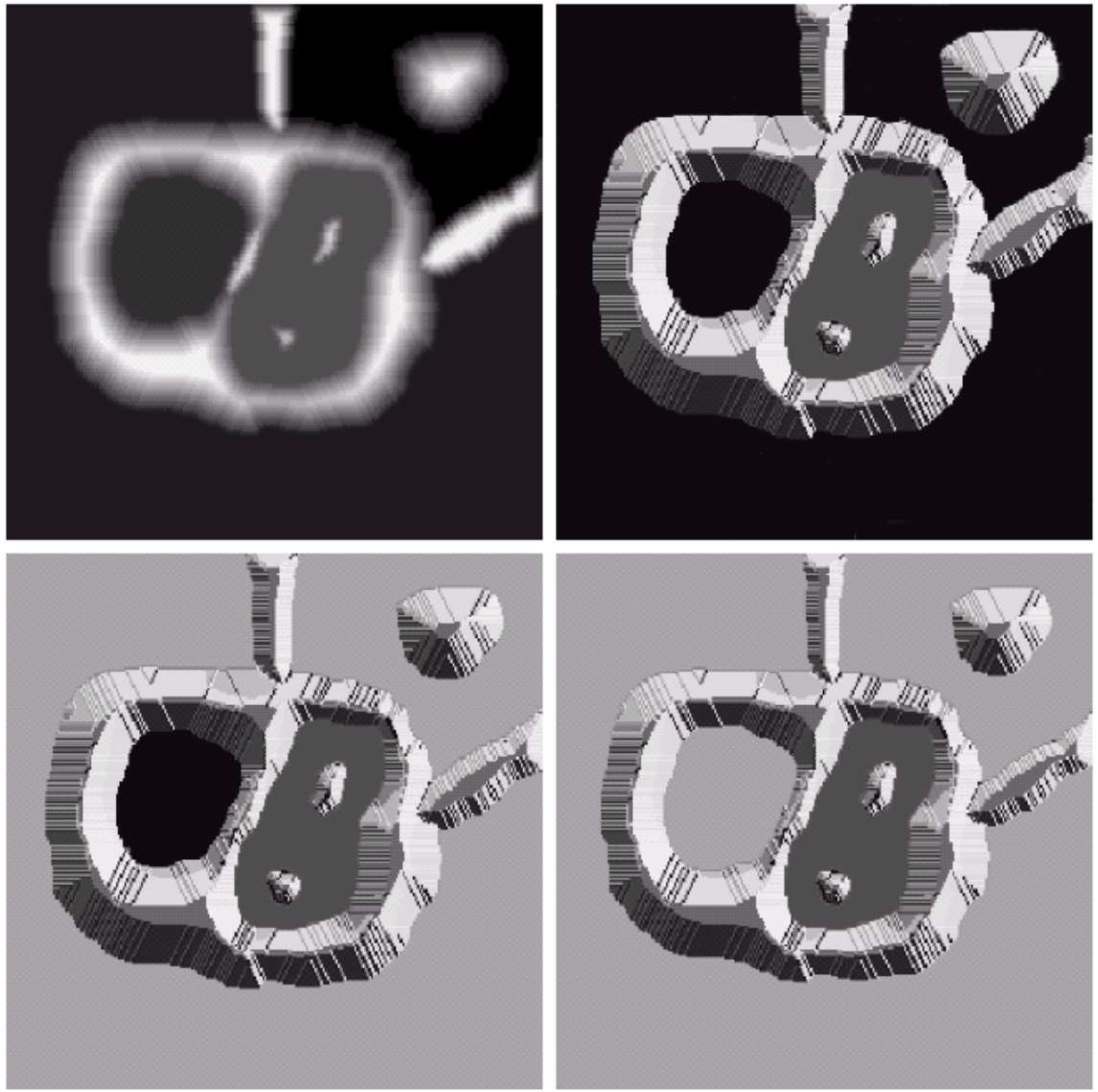
Segmentation by Morphological Watersheds

- The concept of watersheds is based on visualizing an image in three dimensi ons: two spatial coordinates versus gray levels.
- In such a topographic interpretation, we consider three types of points:
  - (a) points belonging to a regional minimum
  - (b) points at which a drop of water would fall with certainty to a single minimum
  - (c) points at which water would be equally likely to fall to more than one such minimu m
- The principal objective of segmentation algorithms based on these concepts is to find the watershed lines.

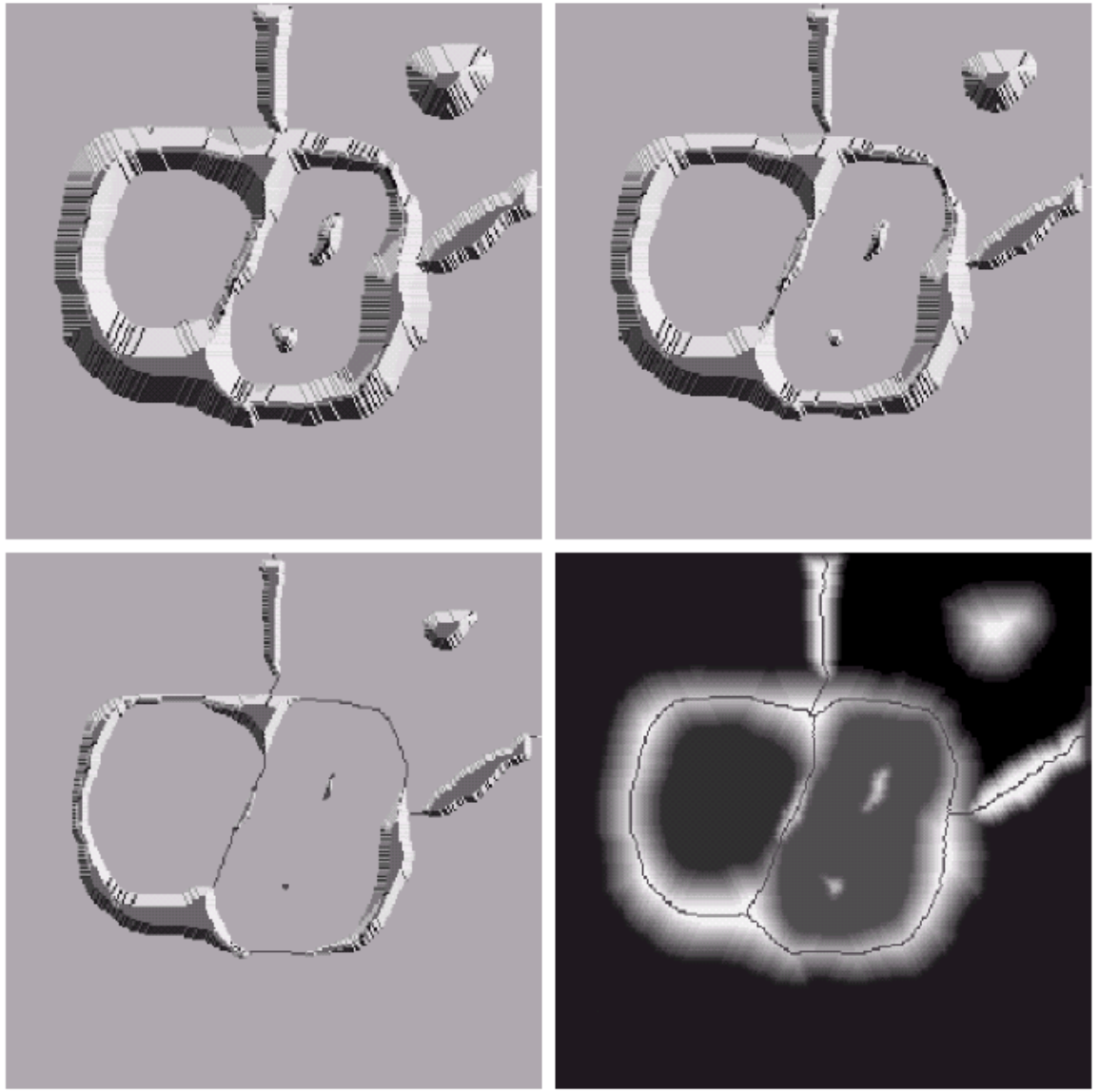
Segmentation by Morphological Watersheds: Example

a b
c d

FIGURE 10.44
(a) Original image.
(b) Topographic view. (c)–(d) Two stages of flooding.



Segmentation by Morphological Watersheds: Example

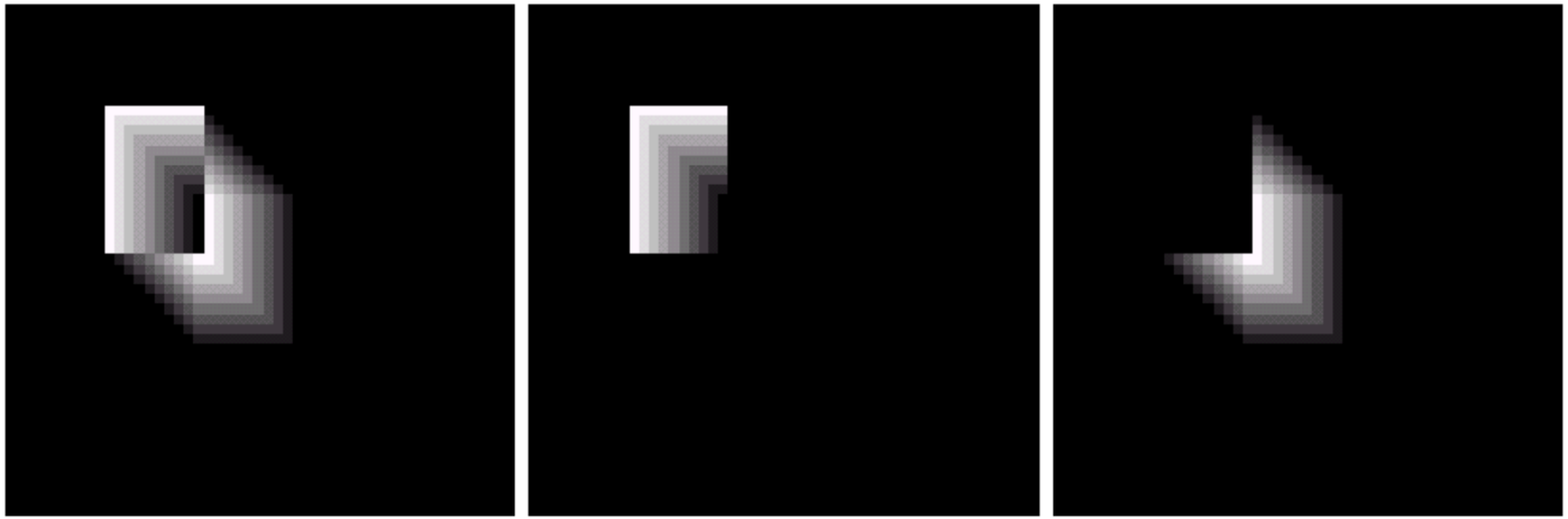


e f
g h

FIGURE 10.44
(Continued)
(e) Result of further flooding. (f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

The Use of Motion in Segmentation

- ADI: accumulative difference image



a b c

FIGURE 10.49 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.

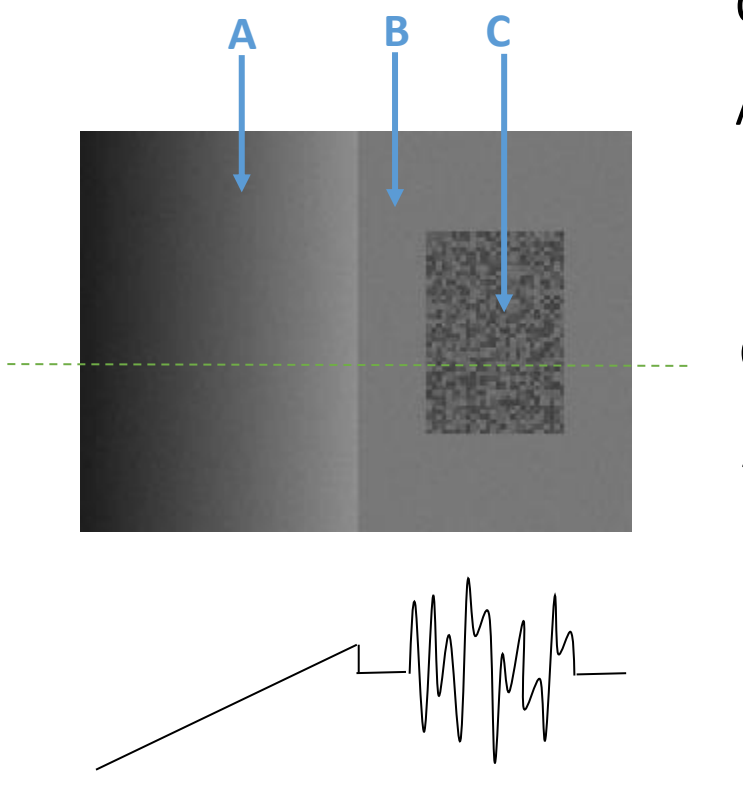
The Use of Motion in Segmentation

a b c

FIGURE 10.50 Building a static reference image. (a) and (b) Two frames in a sequence. (c) Eastbound automobile subtracted from (a) and the background restored from the corresponding area in (b). (Jain and Jain.)

Motivating example

This image has three per ceptually distinct regions



Q: Where would **image thresholding** fail?

A: Region A would be divided in two sets and region C will be split into a large number of arbitrary small subsets

Q: Where would **region growing** fail?

A: Either A and B are merged or region C is split into many small subsets
Also, B and C are merged

Towards better segmentation

Combine color and boundary edge information. How?

- Formulate segmentation quality function (objective or energy)

$$E(S) = C(S) + B(S) + \ldots$$

- Optimize = find the best solution S

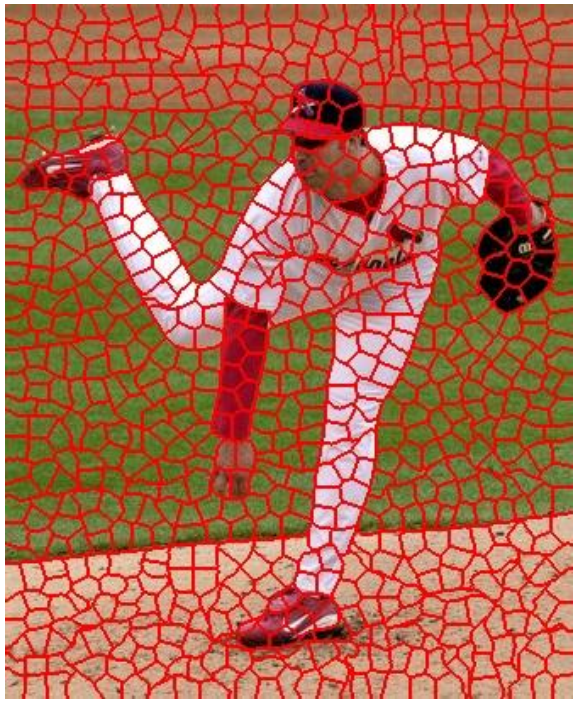
(soon)

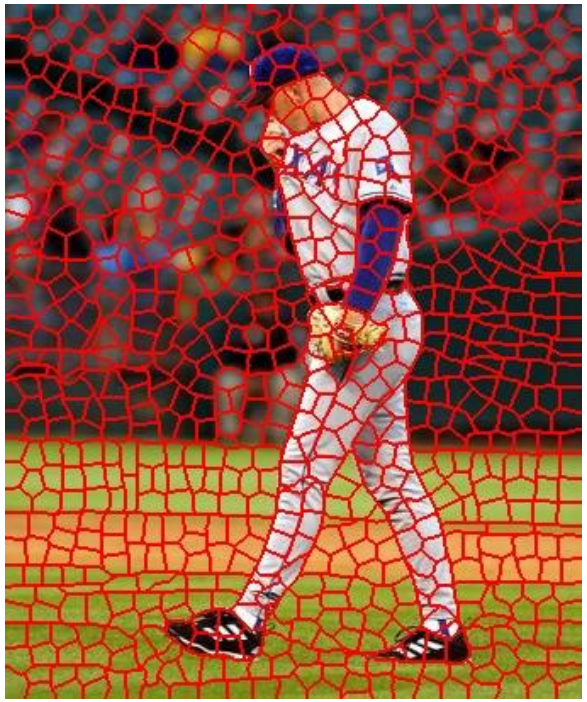
Image segmentation

The goals of segmentation

- Group together similar-looking pixels for efficiency of further processing
  - “Bottom-up” process
  - Unsupervised

“superpixels”



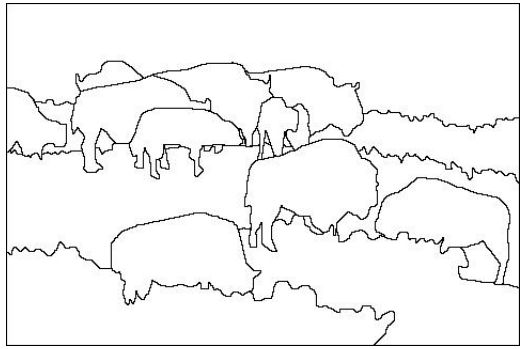


X. Ren and J. Malik. **Learning a classification model for segmentation.** ICCV 2003.

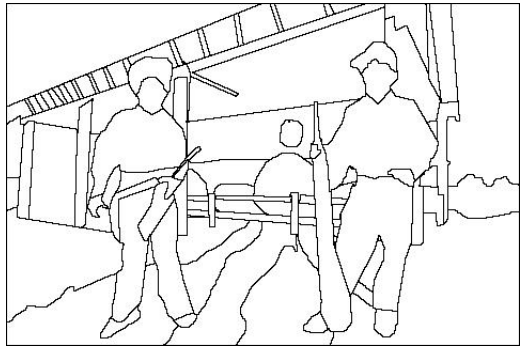
The goals of segmentation

- Separate image into coherent “objects”
  - “Bottom-up” or “top-down” process?
  - Supervised or unsupervised?

image



human segmentation



Berkeley segmentation database:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

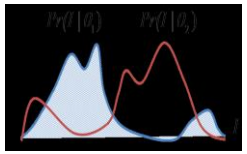
First, more complex decisions in color space

How to move away from manual decision boundaries
(i.e. user-set threshold) in color space ?

some standard solutions:

1. use known probability appearance models, if available
   (leads to likelihood ratio tests, as shown earlier)

$$\log \frac{P_1(I_p)}{P_0(I_p)} \geq 0$$



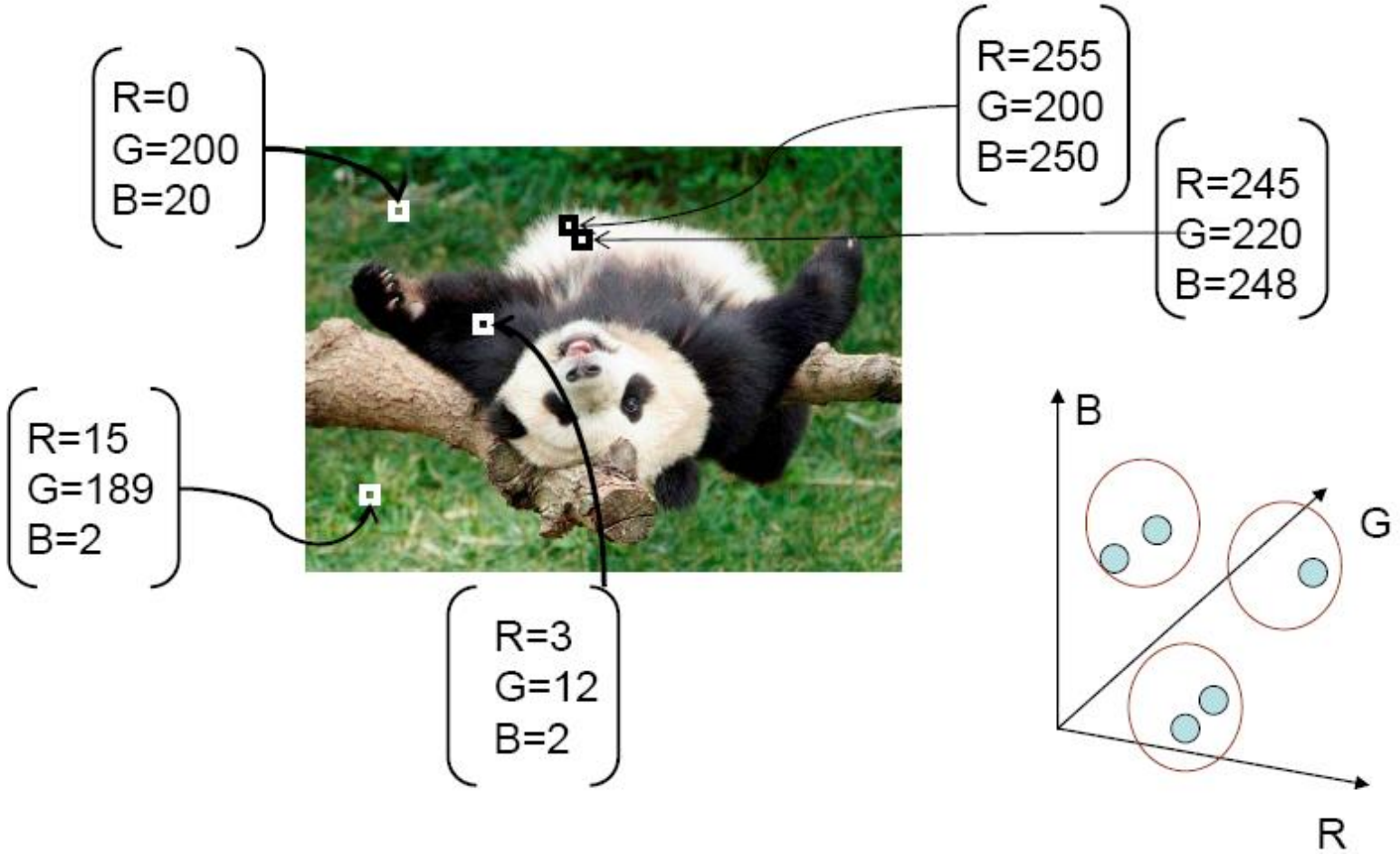
yields complex decision boundaries

2. automatic *clustering* methods

   a. K-means, GMM (parametric)
   b. mean shift, medoid-shift (non-parametric)
   c. kernel-K-means, normalized cuts (non-parametric)

Segmentation as clustering

- Cluster similar pixels (features) together



Source: K. Grauman

Segmentation as clustering

- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
  - Clusters don’t have to be spatially coherent

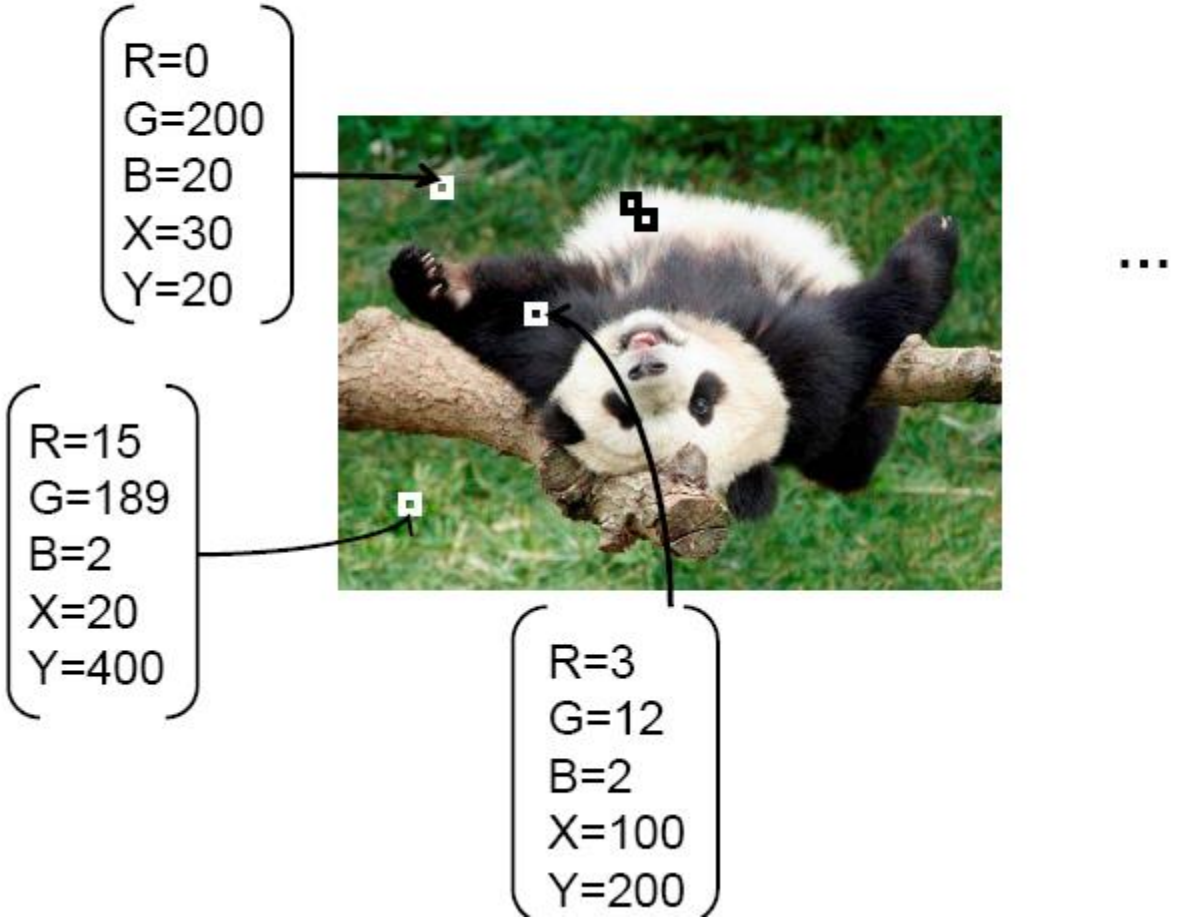
Image

Intensity-based clusters

Color-based clusters

Segmentation as clustering

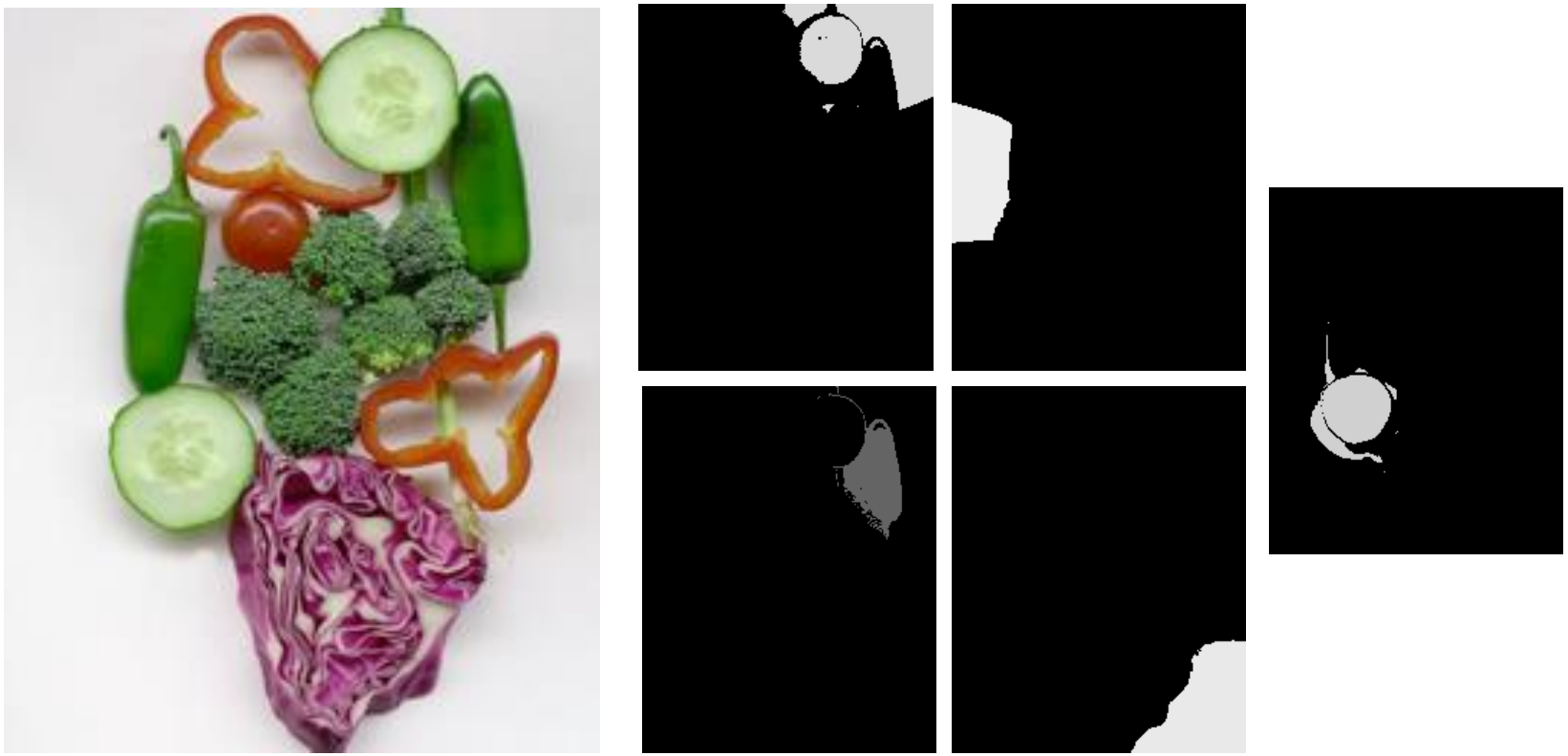
- Cluster similar pixels (features) together



Source: K. Grauman

Segmentation as clustering

- Clustering based on (r,g,b,x,y) values enforces more spatial coherence

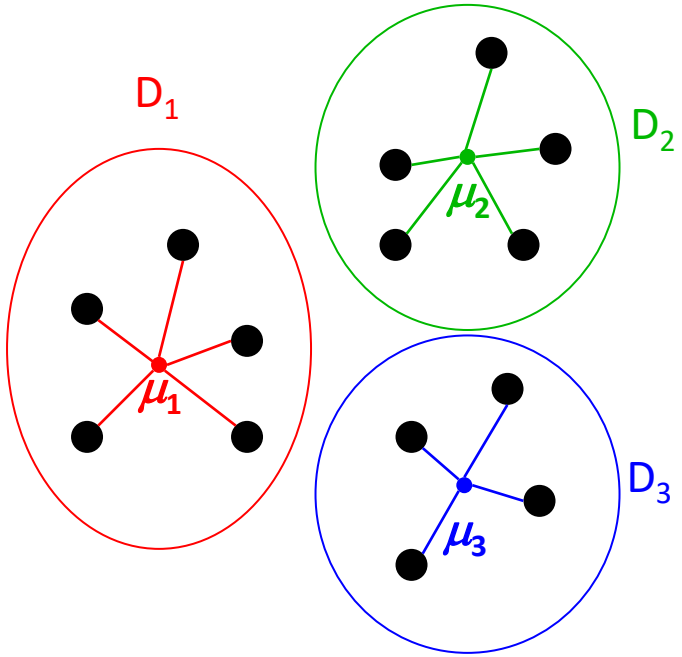


K-means Clustering: Objective Function

- Probably the most popular clustering algorithm
  - assumes the number of clusters is given - k
  - optimizes (approximately) the following objective function for variables Di and μi

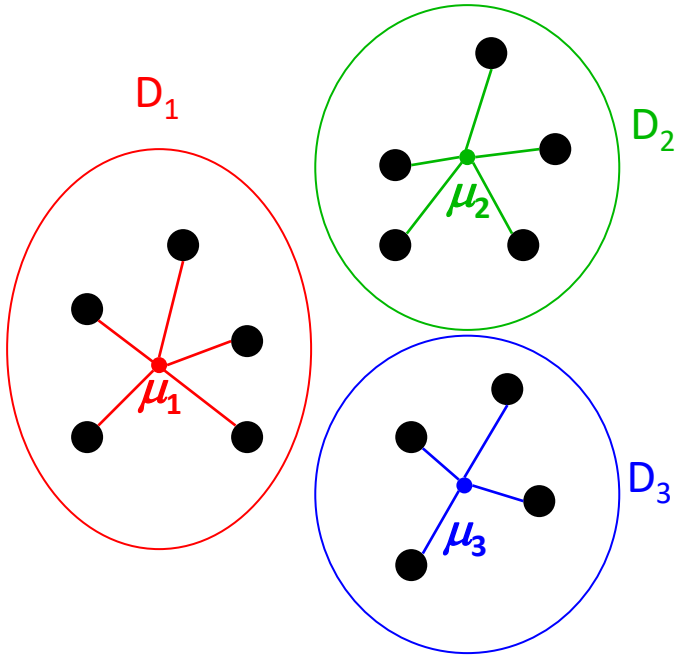
$$E_k = SSE = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - \mu_i\|^2$$

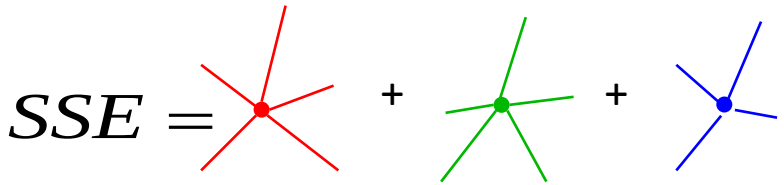
sum of squared errors from cluster center μ_i



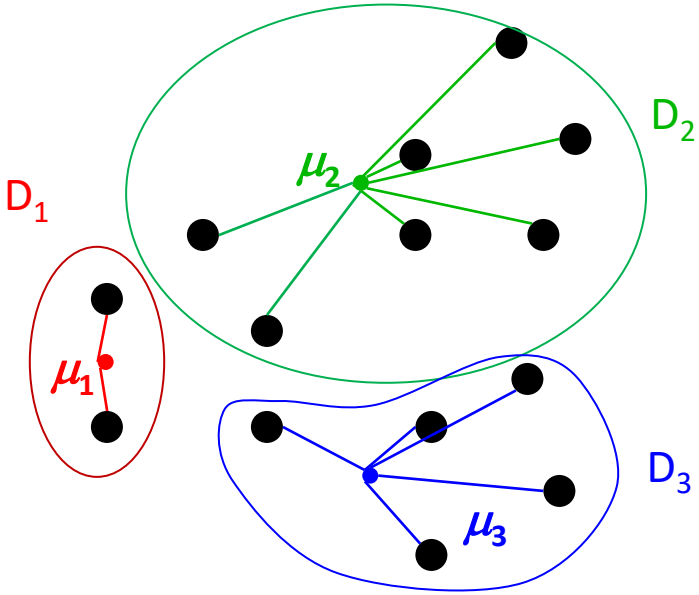
$$SSE = \quad + \quad +$$

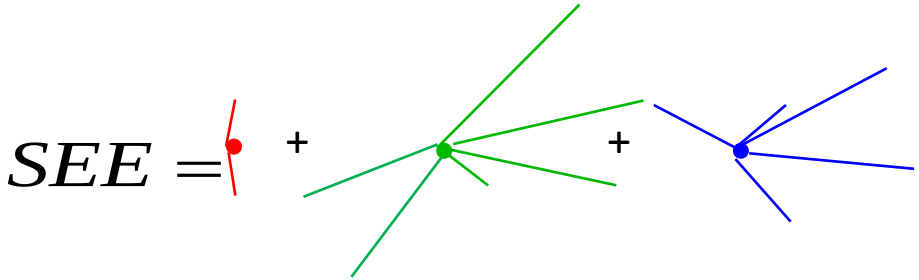
K-means Clustering: Objective Function





Good (tight) clustering
smaller value of SSE

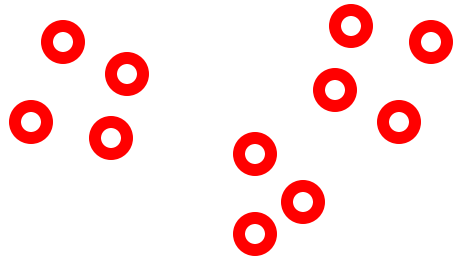




Bad (loose) clustering
larger value of *SSE*

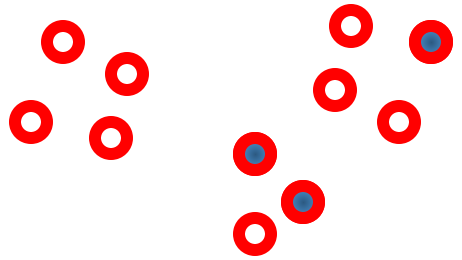
K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly



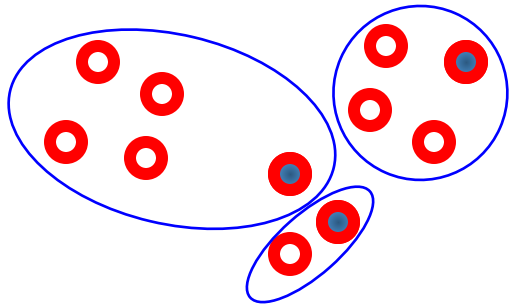
K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly



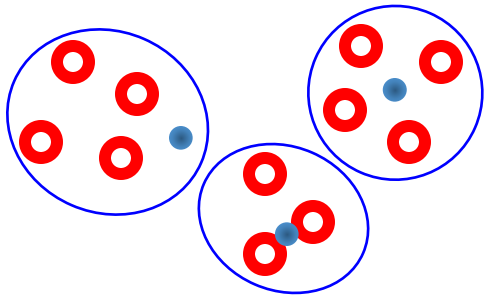
K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center
- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$



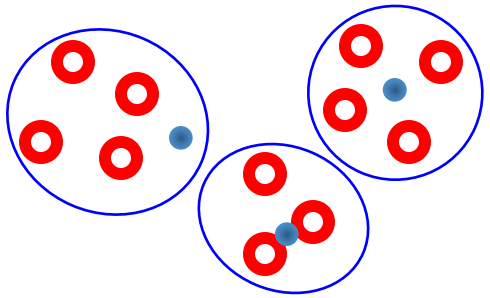
K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center
- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$
  2. re-assign each sample to the closest mean



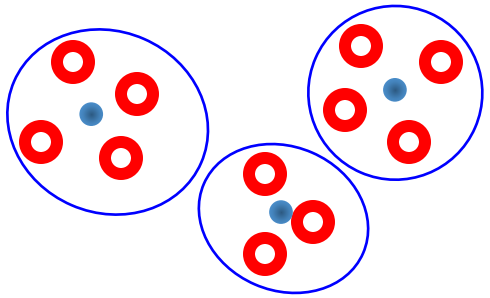
K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center
- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$
  2. re-assign each sample to the closest mean
- Iterate until clusters stop changing



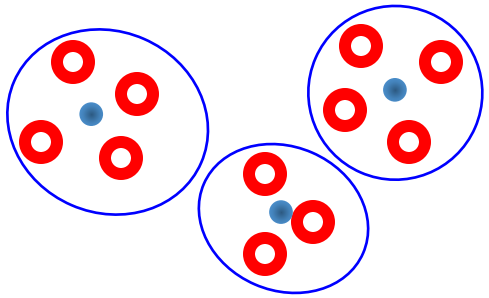
K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center
- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$
  2. re-assign each sample to the closest mean
- Iterate until clusters stop changing



K-means Clustering: Algorithm

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center



- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$
  2. re-assign each sample to the closest mean
- Iterate until clusters stop changing

  - This procedure decreases the value of the objective function

$$E_k(D, \mu) = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - \mu_i\|^2$$

optimization variables

$$D = (D_1, \ldots, D_k)$$

$$\mu = (\mu_1, \ldots, \mu_k)$$

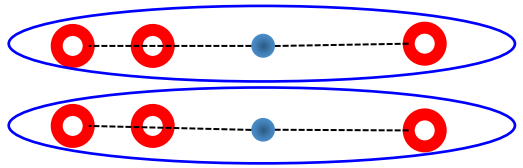
block-coordinate descent: step 1 optimizes μ, step 2 optimizes D

K-means: Approximate Optimization

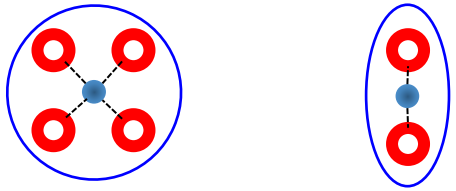
- K-means is fast and often works well in practice
- But can get stuck in a local minimum of objective E_k
  - not surprising, since the problem is NP-hard

initialization

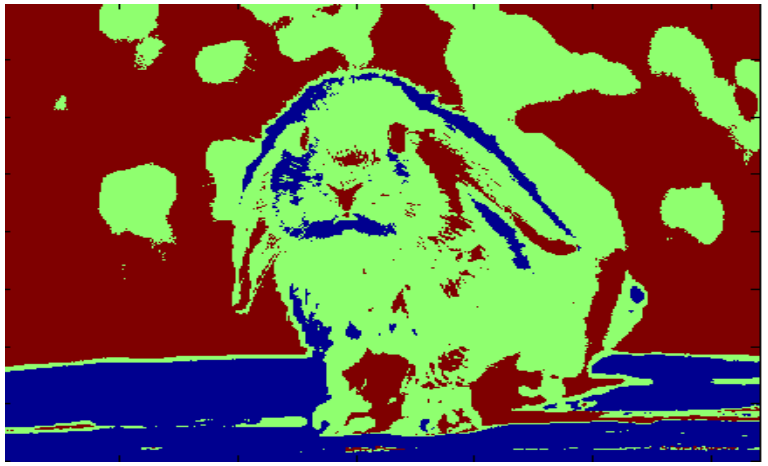
converged to local min



global minimum

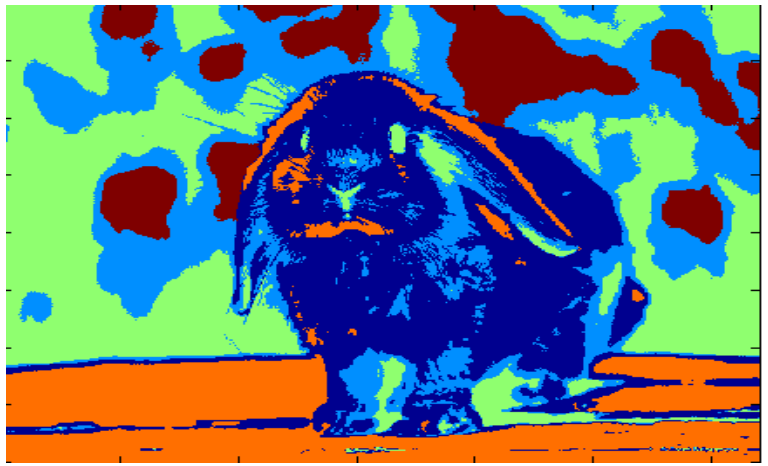


K-means clustering examples: Segmentation?

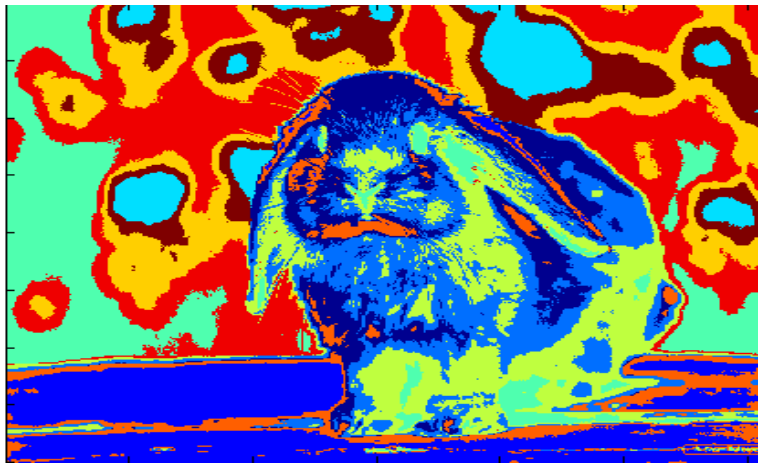


***k* = 3**

(random colors are used to better show segments/clusters)

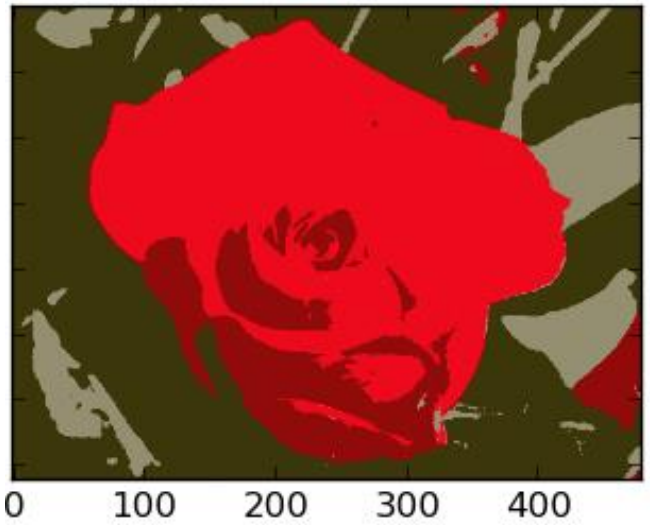


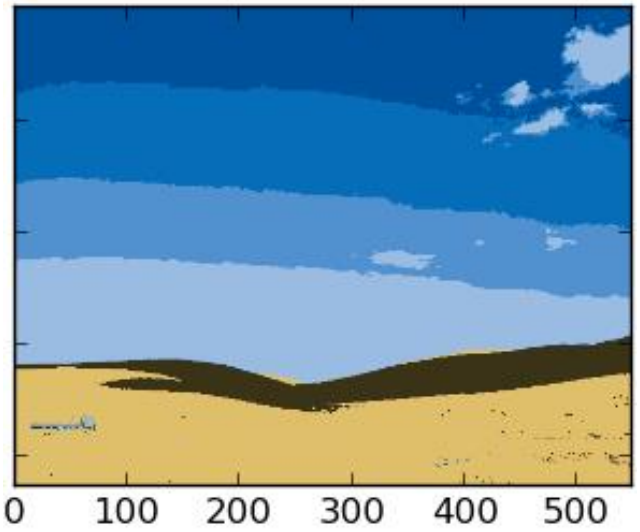
***k* = 5**



***k* = 10**

K-means clustering examples: Color Quantization





NOTE
bias to
equal-size
clusters

K-means clustering examples: Adding XY features

color quantization

RGB features

superpixels

RGBXY features

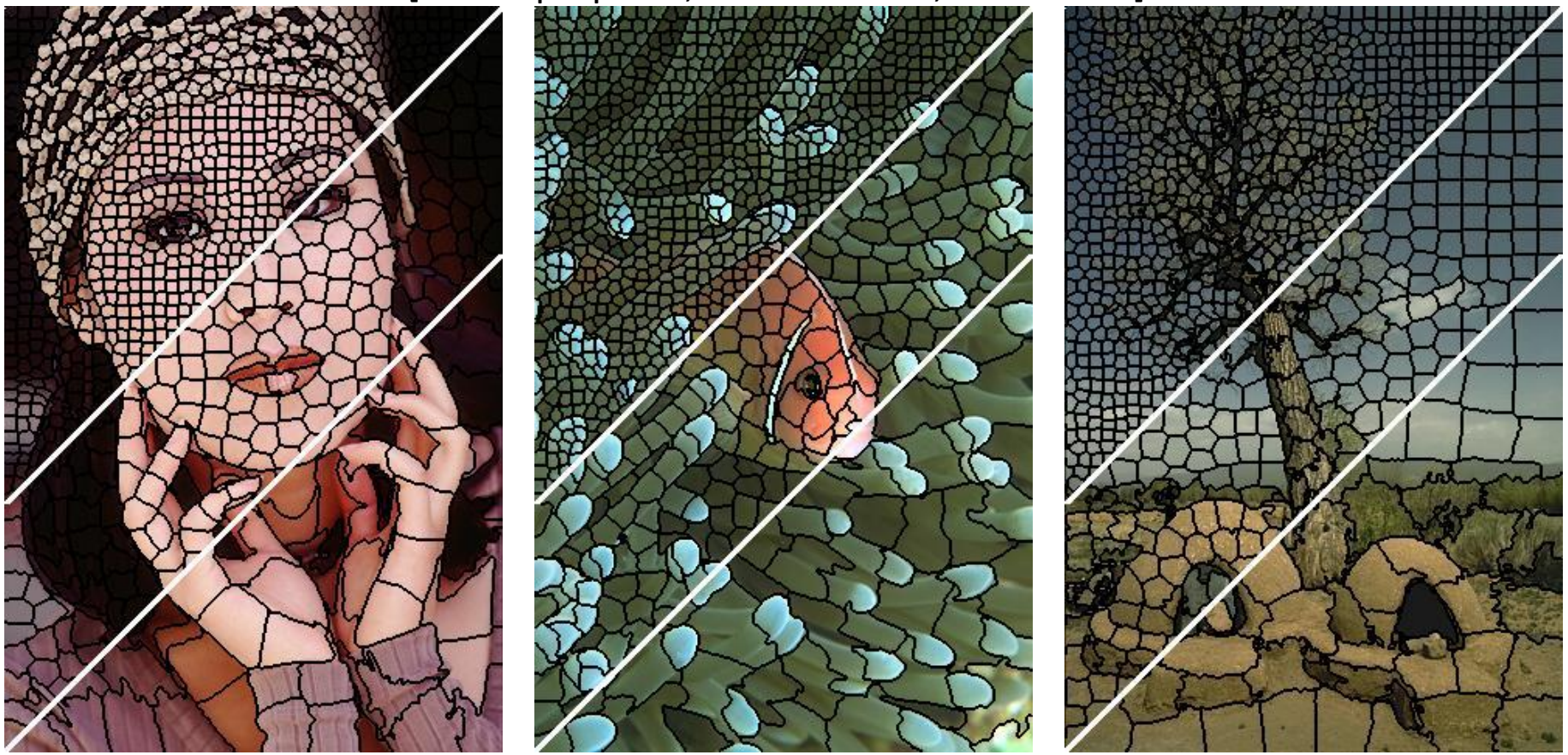
Voronoi cells

XY features only

K-means clustering examples: Superpixels

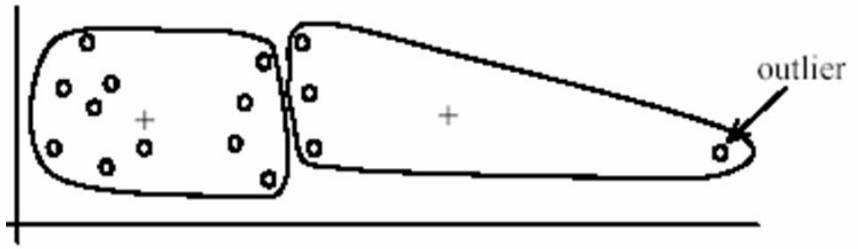
- Apply K-means to RGBXY features

[SLIC superpixels, Achanta et al., PAMI 2011]

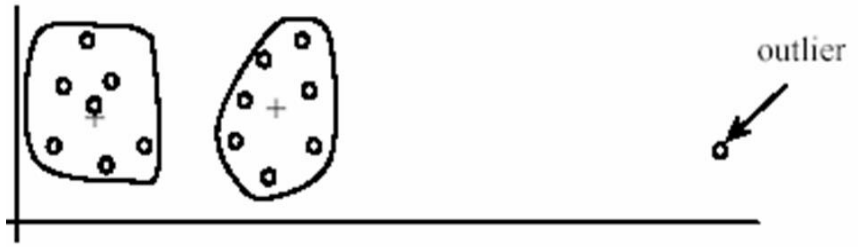


K-Means for segmentation

- Pros
  - Very simple method
  - Converges to a local minimum of the error function
- Cons
  - Memory-intensive
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds “spherical” clusters



(A): Undesirable clusters



(B): Ideal clusters

Problem

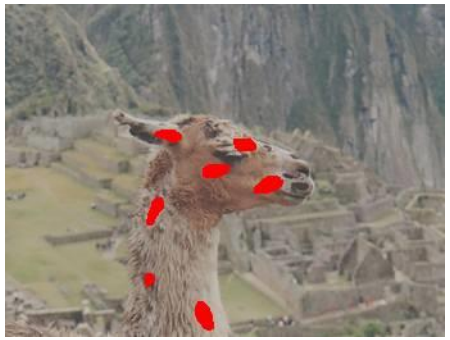
Fast &
Accurate ?

What GrabCut does

User Input

Result

Magic Wand
(198?)



Regions

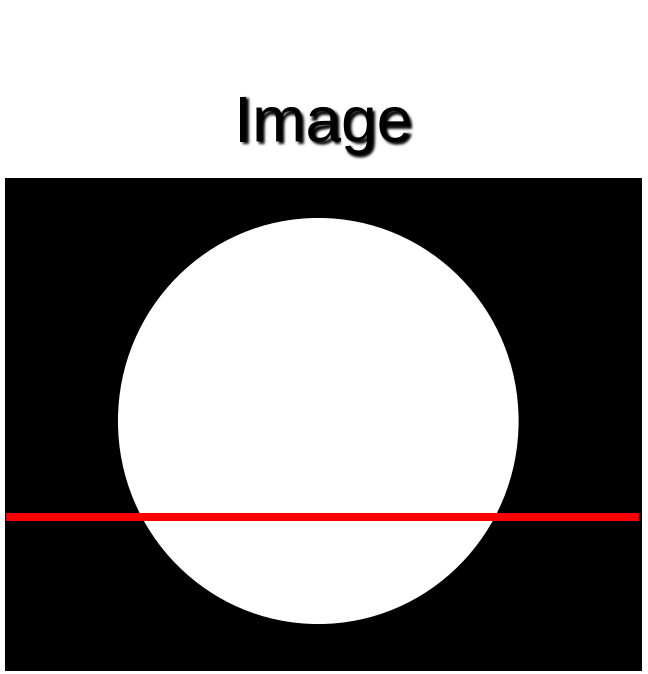
Intelligent Scissors
Mortensen and Barrett (1995)

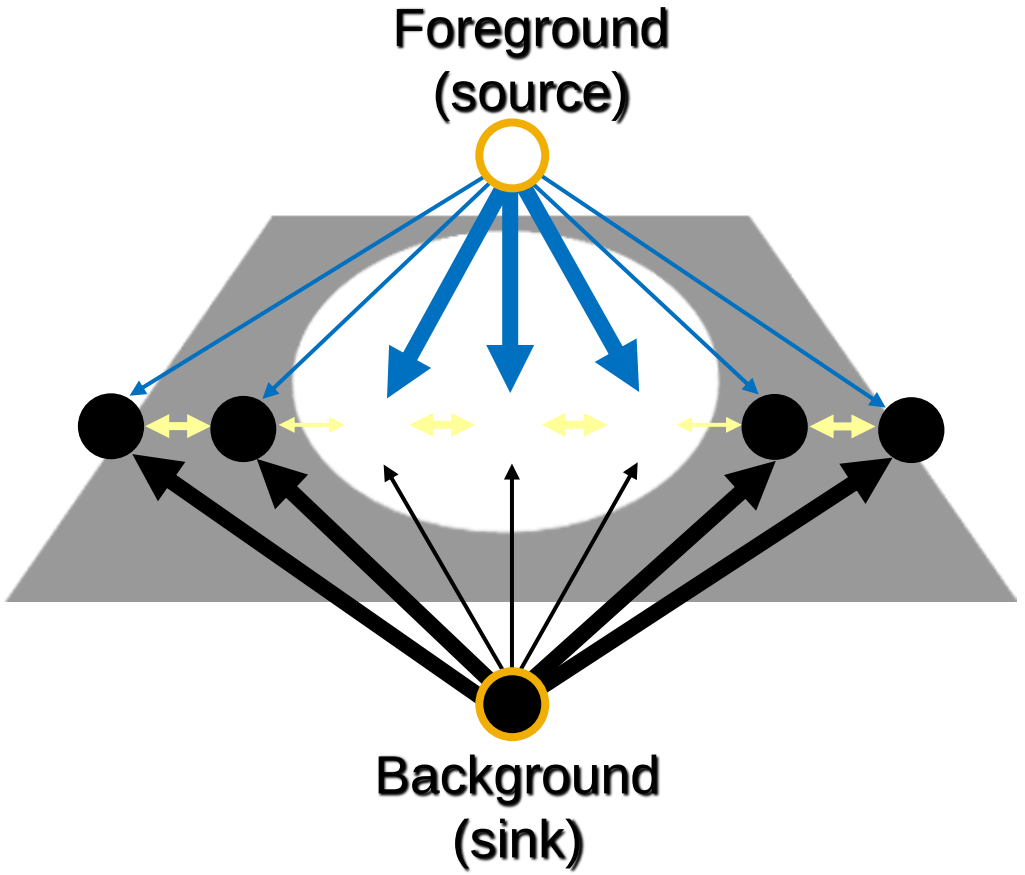
Boundary

GrabCut

Regions & Boundary

Graph Cuts: Boykov and Jolly (2001)





Cut: separating source and sink; Energy: collection of edges

Min Cut: Global minimal enegry in polynomial time

Iterated Graph Cut

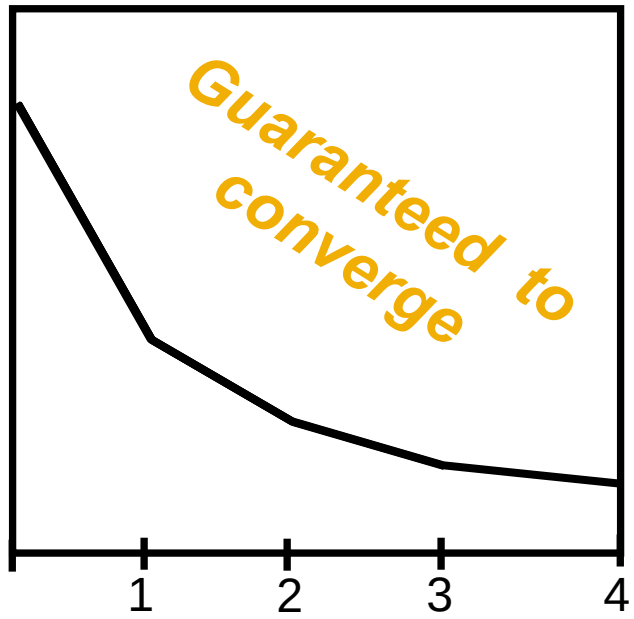
User Initialisation

K-means for learning colour distributions

Graph cuts to infer the segmentation

Iterated Graph Cuts

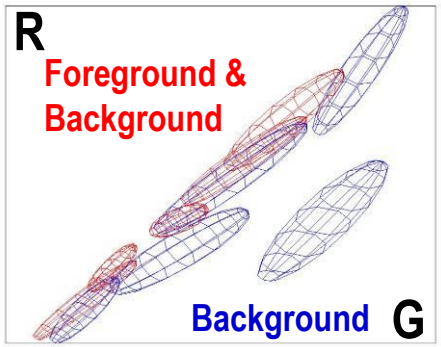
Result



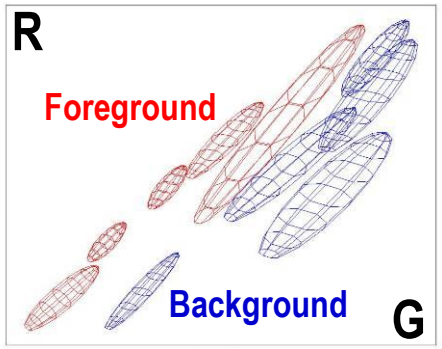
Energy after each Iteration

Colour Model

- Gaussian Mixture Model (typically 5-8 components)



Iterated
graph cut

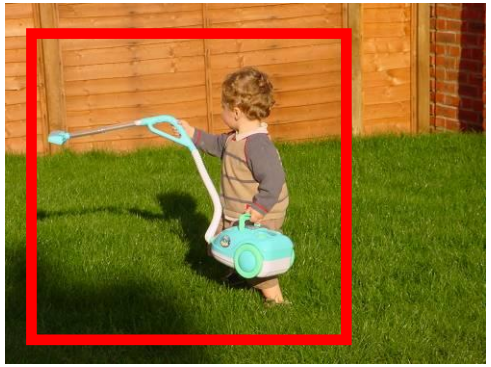


Coherence Model

An object is a coherent set of pixels:

Blake et al. (2004): Learn jointly

Moderately straightforward examples






… GrabCut completes automatically

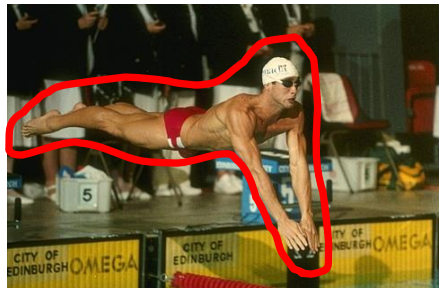
Difficult Examples

- Camouflage &
- Low Contrast

Fine structure

No telepathy

Initial Rectangle

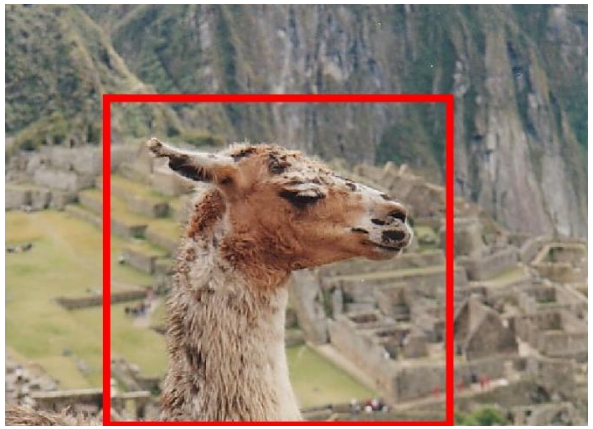
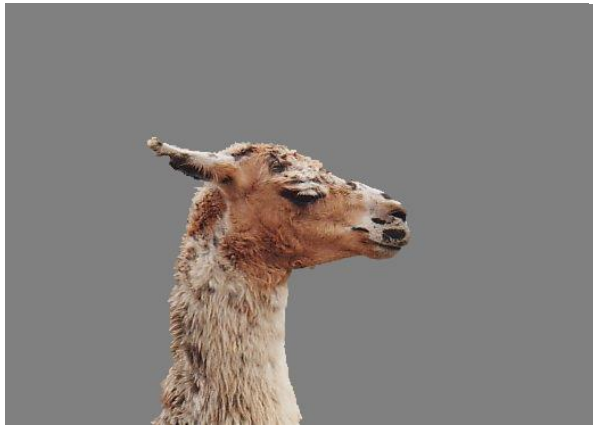
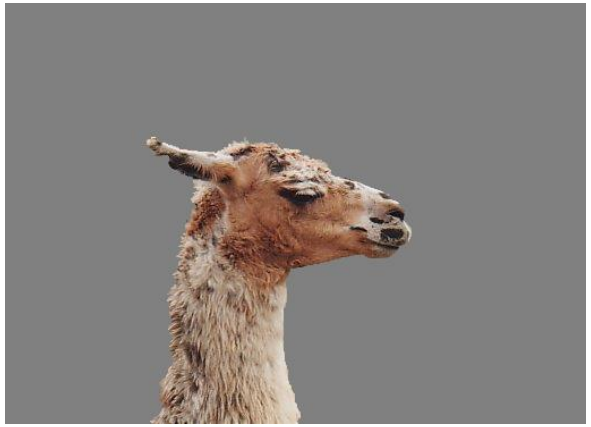


Initial Result

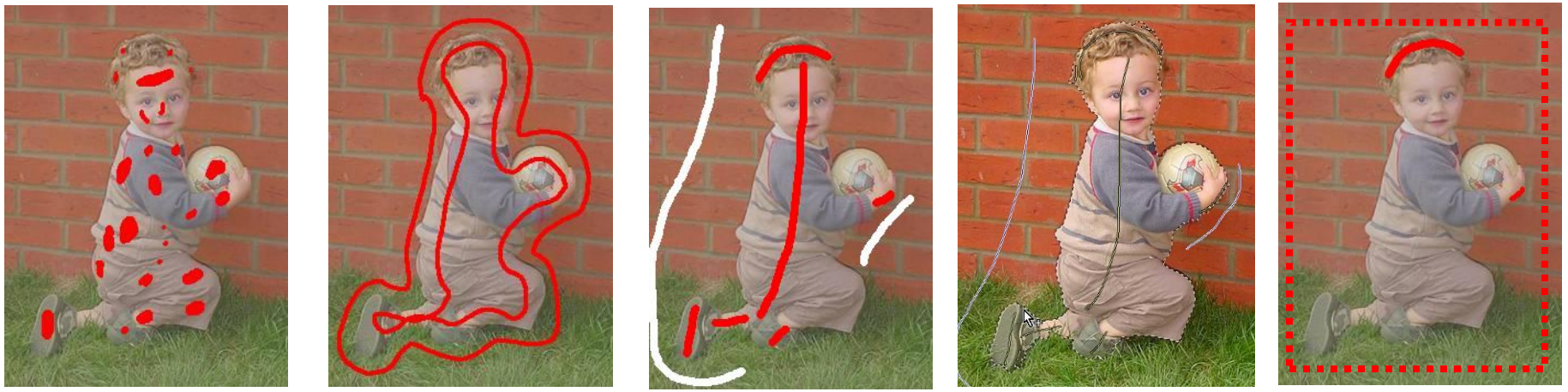
Evaluation – Labelled Database

Available online: http://research.microsoft.com/vision/cambridge/segmentation/

Comparison

	Boykov and Jolly (2001)	GrabCut
User Input		
Result		
	Error Rate: 0.72%	**Error Rate: 0.72%**

Summary



Magic Wand (198?)

Intelligent Scissors Mortensen and Barrett (1995)

Graph Cuts Boykov and Jolly (2001)

LazySnapping Li et al. (2004)

GrabCut Rother et al. (2004)

Image Segmentation using K-means algorithm

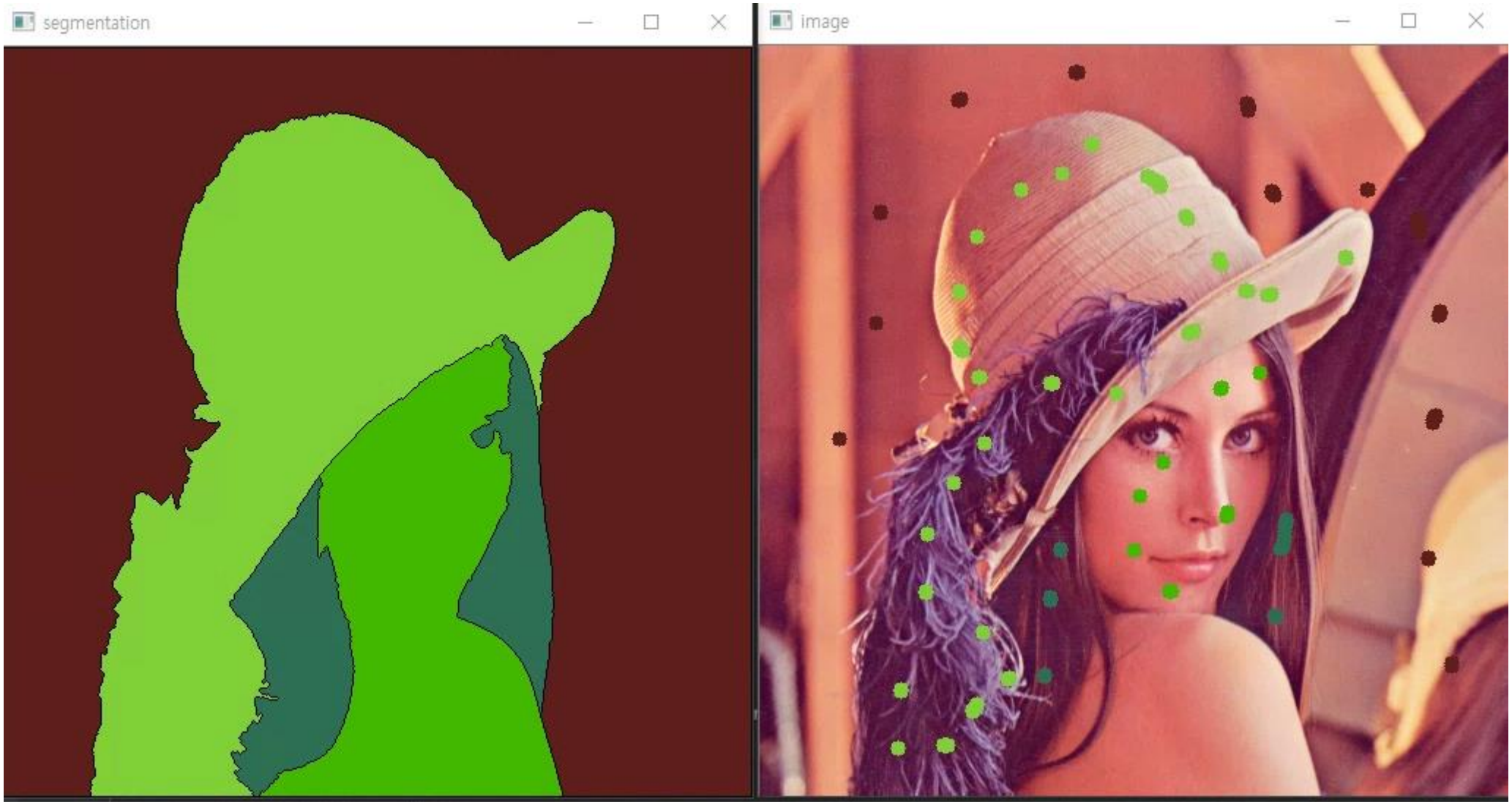
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('../data/Lena.png').astype(np.float32) / 255.
image_lab = cv2.cvtColor(image, cv2.COLOR_BGR2Lab)

data = image_lab.reshape((-1, 3))

num_classes = 8
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 50, 0.1)
_, labels, centers = cv2.kmeans(data, num_classes, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

segmented_lab = centers[labels.flatten()].reshape(image.shape)
segmented = cv2.cvtColor(segmented_lab, cv2.COLOR_Lab2RGB)

plt.subplot(121)
plt.axis('off')
plt.title('original')
plt.imshow(image[:, :, [2, 1, 0]])
plt.subplot(122)
plt.axis('off')
plt.title('segmented')
plt.imshow(segmented)
plt.show()
```

Image Segmentation using K-means algorithm

Image Segmentation using Watershed

```python
import cv2
import numpy as np
from random import randint

img = cv2.imread('../data/Lena.png')
show_img = np.copy(img)

seeds = np.full(img.shape[0:2], 0, np.int32)
segmentation = np.full(img.shape, 0, np.uint8)

n_seeds = 9

colors = []
for m in range(n_seeds):
    colors.append((255 * m / n_seeds, randint(0, 255), randint(0, 255)))

mouse_pressed = False
current_seed = 1
seeds_updated = False

def mouse_callback(event, x, y, flags, param):
    global mouse_pressed, seeds_updated

    if event == cv2.EVENT_LBUTTONDOWN:
        mouse_pressed = True
        cv2.circle(seeds, (x, y), 5, (current_seed), cv2.FILLED)
        cv2.circle(show_img, (x, y), 5, colors[current_seed - 1], cv2.FILLED)
        seeds_updated = True

    elif event == cv2.EVENT_MOUSEMOVE:
        if mouse_pressed:
            cv2.circle(seeds, (x, y), 5, (current_seed), cv2.FILLED)
            cv2.circle(show_img, (x, y), 5, colors[current_seed - 1], cv2.FILLED)
            seeds_updated = True
    elif event == cv2.EVENT_LBUTTONUP:
        mouse_pressed = False

cv2.namedWindow('image')
cv2.setMouseCallback('image', mouse_callback)

while True:
    cv2.imshow('segmentation', segmentation)
    cv2.imshow('image', show_img)

    k = cv2.waitKey(1)

    if k == 27:
        break
    elif k == ord('c'):
        show_img = np.copy(img)
        seeds = np.full(img.shape[0:2], 0, np.int32)
        segmentation = np.full(img.shape, 0, np.uint8)
    elif k > 0 and chr(k).isdigit():
        n = int(chr(k))
        if 1 <= n <= n_seeds and not mouse_pressed:
            current_seed = n

    if seeds_updated and not mouse_pressed:
        seeds_copy = np.copy(seeds)
        cv2.watershed(img, seeds_copy)
        segmentation = np.full(img.shape, 0, np.uint8)
        for m in range(n_seeds):
            segmentation[seeds_copy == (m + 1)] = colors[m]

        seeds_updated = False

cv2.destroyAllWindows()
```

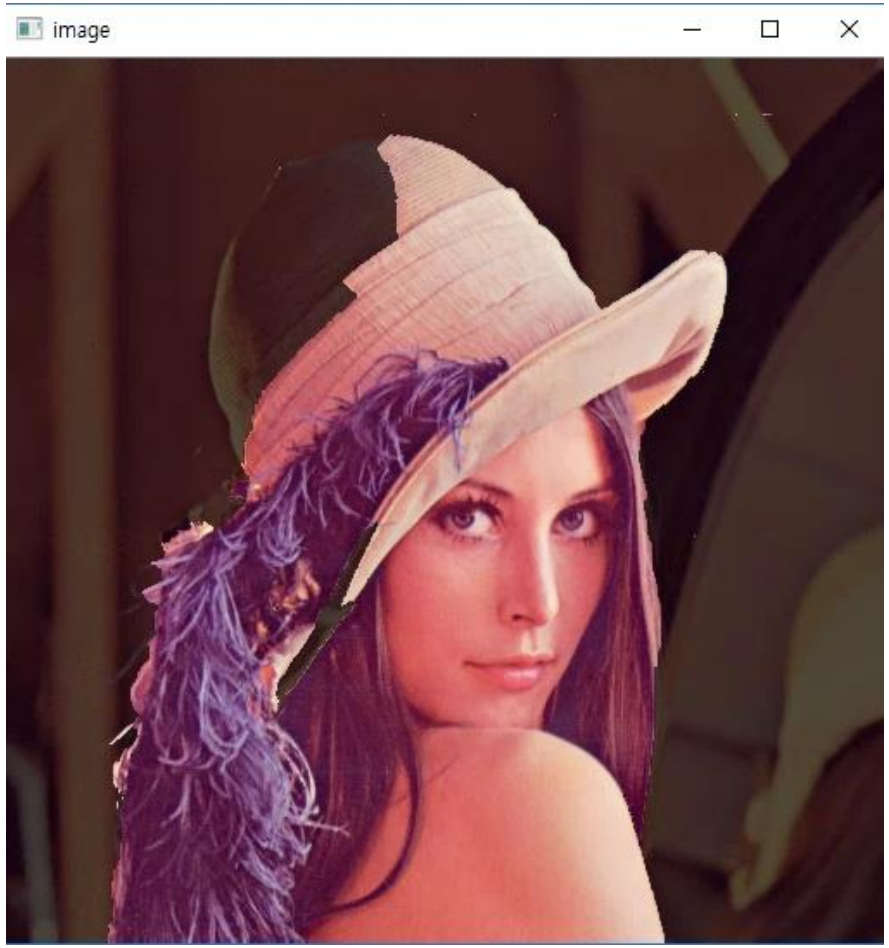
Image Segmentation using Watershed



Foreground segmentation using GrabCut

```python
import cv2
import numpy as np

img = cv2.imread('../data/Lena.png', cv2.IMREAD_COLOR)
show_img = np.copy(img)

mouse_pressed = False
y = x = w = h = 0


def mouse_callback(event, _x, _y, flags, param):
    global show_img, x, y, w, h, mouse_pressed

    if event == cv2.EVENT_LBUTTONDOWN:
        mouse_pressed = True
        x, y = _x, _y
        show_img = np.copy(img)

    elif event == cv2.EVENT_MOUSEMOVE:
        if mouse_pressed:
            show_img = np.copy(img)
            cv2.rectangle(show_img, (x, y),
                          (_x, _y), (0, 255, 0), 3)

    elif event == cv2.EVENT_LBUTTONUP:
        mouse_pressed = False
        w, h = _x - x, _y - y


cv2.namedWindow('image')
cv2.setMouseCallback('image', mouse_callback)
while True:
    cv2.imshow('image', show_img)
    k = cv2.waitKey(1)

    if k == ord('a') and not mouse_pressed:
        if w * h > 0:
            break

cv2.destroyAllWindows()

labels = np.zeros(img.shape[:2], np.uint8)
labels, bgdModel, fgdModel = cv2.grabCut(img, labels,
    (x, y, w, h), None, None, 5, cv2.GC_INIT_WITH_RECT)

show_img = np.copy(img)
show_img[(labels == cv2.GC_PR_BGD) | (labels == cv2.GC_BGD)] //= 3

cv2.imshow('image', show_img)
cv2.waitKey()
cv2.destroyAllWindows()

label = cv2.GC_BGD
lbl_clrs = {cv2.GC_BGD: (0, 0, 0), cv2.GC_FGD: (255, 255, 255)}


def mouse_callback(event, x, y, flags, param):
    global mouse_pressed

    if event == cv2.EVENT_LBUTTONDOWN:
        mouse_pressed = True
        cv2.circle(labels, (x, y), 5, label, cv2.FILLED)
        cv2.circle(show_img, (x, y), 5, lbl_clrs[label], cv2.FILLED)
    elif event == cv2.EVENT_MOUSEMOVE:
        if mouse_pressed:
            cv2.circle(labels, (x, y), 5, label, cv2.FILLED)
            cv2.circle(show_img, (x, y), 5, lbl_clrs[label], cv2.FILLED)

    elif event == cv2.EVENT_LBUTTONUP:
        mouse_pressed = False


cv2.namedWindow('image')
cv2.setMouseCallback('image', mouse_callback)

while True:
    cv2.imshow('image', show_img)
    k = cv2.waitKey(1)

    if k == ord('a') and not mouse_pressed:
        break
    elif k == ord('1'):
        label = cv2.GC_FGD - label

cv2.destroyAllWindows()

labels, bgdModel, fgdModel = cv2.grabCut(img, labels, None,
        bgdModel, fgdModel, 5, cv2.GC_INIT_WITH_MASK)

show_img = np.copy(img)
show_img[(labels == cv2.GC_PR_BGD) | (labels == cv2.GC_BGD)] //= 3

cv2.imshow('image', show_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

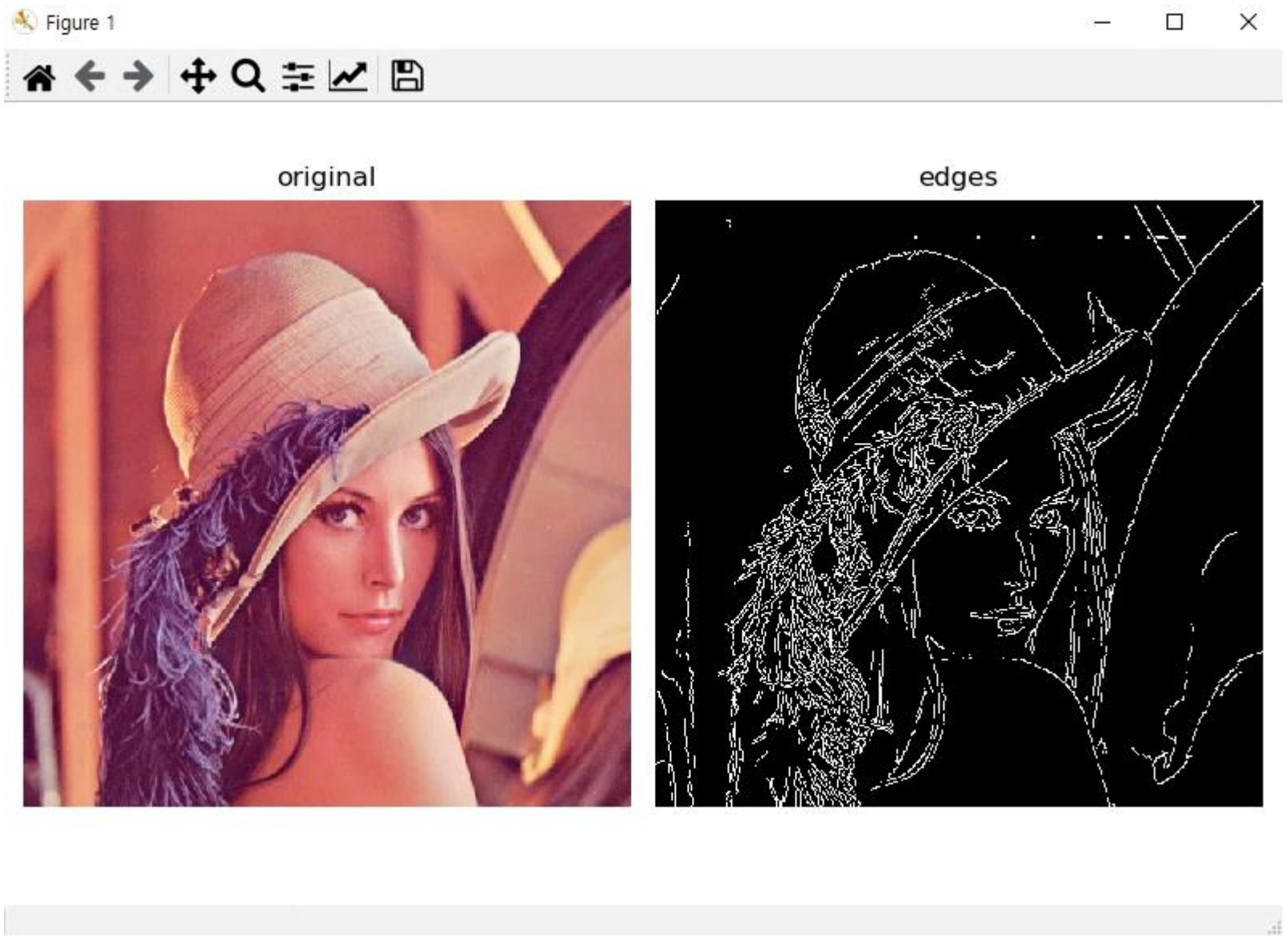
Foreground segmentation using GrabCut



Canny Edge Detection

```python
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('../data/Lena.png')

edges = cv2.Canny(image, 200, 100)

plt.figure(figsize=(8,5))
plt.subplot(121)
plt.axis('off')
plt.title('original')
plt.imshow(image[:,:,[2,1,0]])
plt.subplot(122)
plt.axis('off')
plt.title('edges')
plt.imshow(edges, cmap='gray')
plt.tight_layout()
plt.show()
```

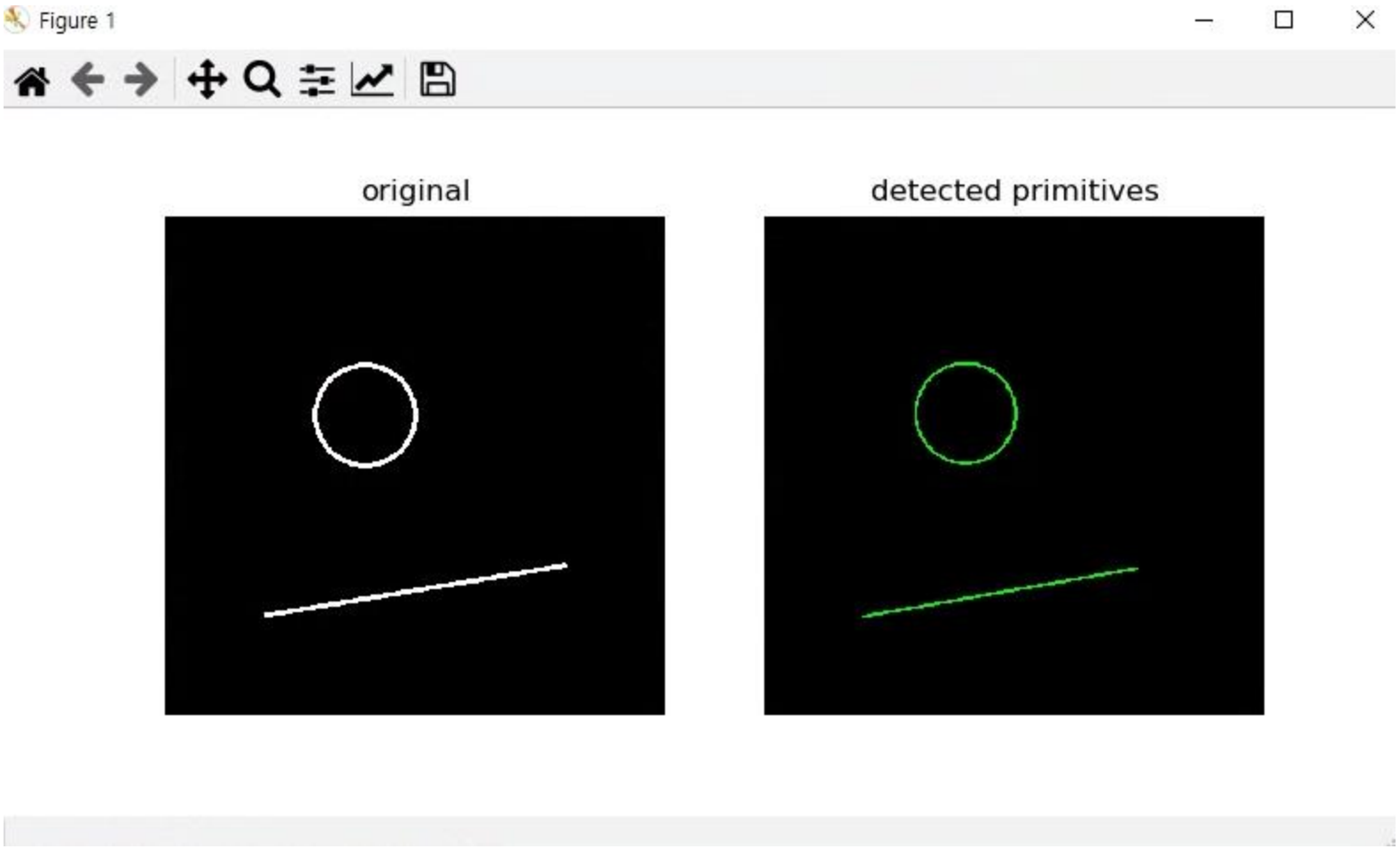
Canny Edge Detection



Line and Circle Detection using Hough Transform

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = np.zeros((500, 500), np.uint8)

cv2.circle(img, (200, 200), 50, 255, 3)
cv2.line(img, (100, 400), (400, 350), 255, 3);

circles = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 1, 15, param1=200, param2=30)[0]

lines = cv2.HoughLinesP(img, 1, np.pi/180, 100, 100, 10)[0]

dbg_img = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)
for x1, y1, x2, y2 in lines:
    print('Detected line: ({} {}) ({} {})'.format(x1, y1, x2, y2))
    cv2.line(dbg_img, (x1, y1), (x2, y2), (0, 255, 0), 2)

for c in circles:
    print('Detected circle: center=({} {}), radius={}'.format(c[0], c[1], c[2]))
    cv2.circle(dbg_img, (c[0], c[1]), c[2], (0, 255, 0), 2)

plt.figure(figsize=(8,4))
plt.subplot(121)
plt.title('original')
plt.axis('off')
plt.imshow(img, cmap='gray')
plt.subplot(122)
plt.title('detected primitives')
plt.axis('off')
plt.imshow(dbg_img)
plt.show()
```

Line and Circle Detection using Hough Transform



실습 문제

- (1) K-means Clustering을 이용해서 (R, G, B, X, Y)의 거리에 따라서 Image Segmentation을 할 수 있도록 코드를 작성하고, (R, G, B)의 거리에 따른 Image Segmentation 과 결과를 비교하시오.

- (2) GrabCut 알고리즘을 통해서 사용자로부터 입력을 통해 원하는 Image Segmentation 결과가 나올 때까지 결과를 수정할 수 있도록 코드를 작성하시오.