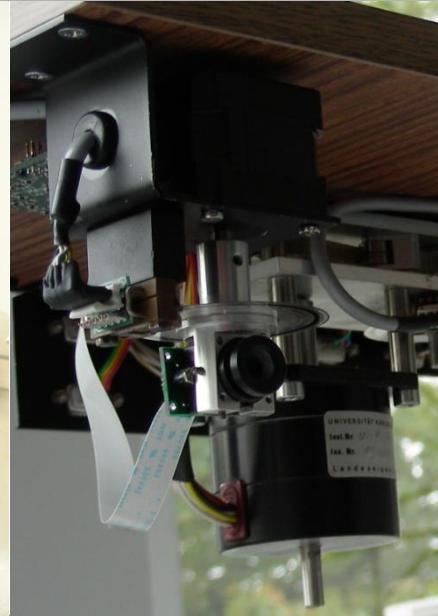
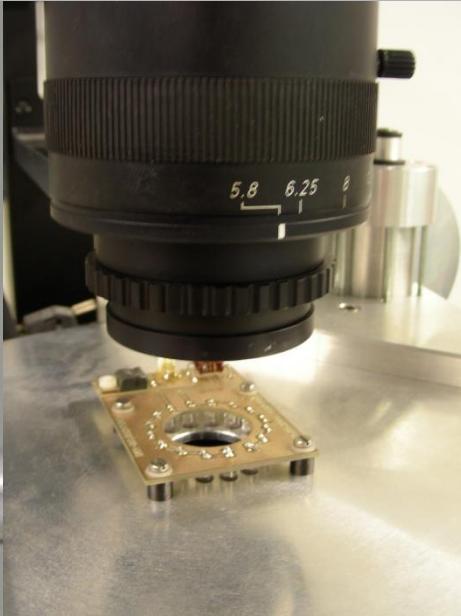


# Machine Vision

## Chapter 6: Segmentation

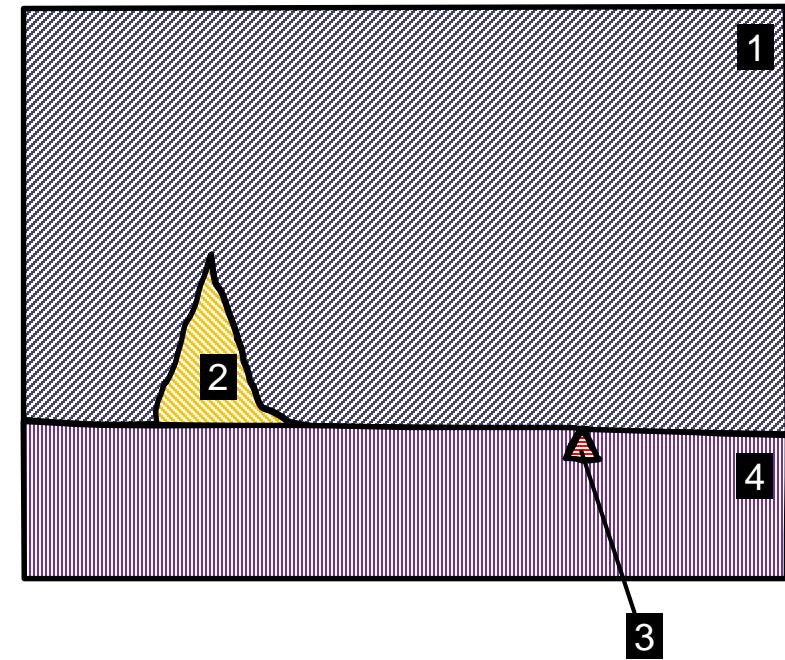
*Dr. Martin Lauer*

Institut für Mess-  
und Regelungstechnik



# Segmentation

- partitioning the image into areas of similar color
  - image driven
  - no semantics for segments
- what we need for segmentation:
  - a criterion that defines which pixels belong to a segment and which don't
  - an algorithm that efficiently subdivides pixels into segments



# Criteria for Segmentation

- criteria for segmentation:
  - *predefined color criterion*
  - *neighborhood criterion*
  - *homogeneity criterion*
  - *connectedness criterion*
  - *spatial criterion*
  - *boundary smoothness criterion*
  - *size criteria*
  - ...

# Criteria for Segmentation

## *– predefined color criterion*

pixel color belongs to a predefined set of “interesting” colors

1. specify which color values are relevant
2. check which pixels are colored

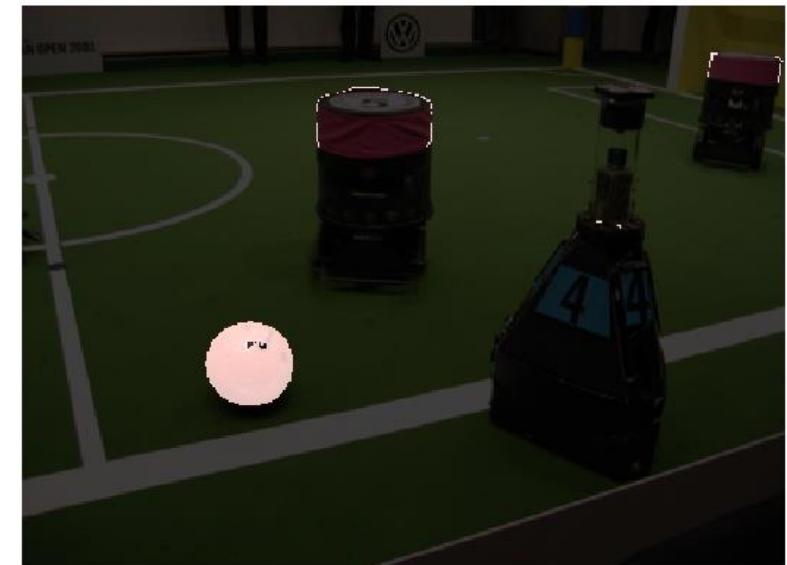
example:

find the orange ball on a soccer robot field

orange pixels are those with HSV values in the interval:  $0^\circ \leq H \leq 24^\circ$ ,  $0.4 \leq S \leq 1$ ,  $0.4 \leq V \leq 1$

advantages and disadvantages:

- very simple, very fast
- can be applied if color of objects is known in advance and color is discriminative
- not applicable if different objects share the same colors
- finding appropriate color specification is often cumbersome



# Criteria for Segmentation

## *– neighborhood criterion*

pixel color is similar to color of neighboring pixels

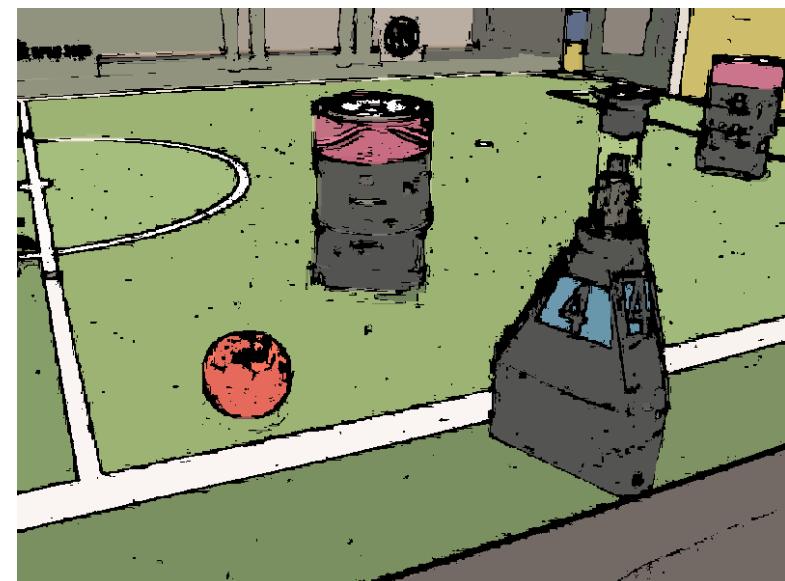
1. specify which colors are similar
2. group all pixels in one segment which have at least one similar neighbor which already belongs to the segment

example:

pixels are neighboring if Euclidean distance of RGB triplets is less than  $7/255$

advantages and disadvantages:

- simple
- objects colors don't need to be known
- object boundaries must be high-contrast, the inside must be low-contrast
- blurry images might lead to undersegmentation, noisy images to oversegmentation



# Criteria for Segmentation

## *– homogeneity criterion*

pixel color is similar to the average color of a segment

1. specify how to compute the average color and decide whether two colors are similar
2. group all pixels in one segment which are similar to the average color of a segment

example:

pixels that are similar to the average ball color

advantages and disadvantages:

- objects colors don't need to be known
- objects must have similar color in all parts
- does not support low frequent color changes
- recurrent definition



# Criteria for Segmentation

## *– connectedness criterion*

all pixels in the same segment must be connected, i.e. between two pixels of the segment there is a path which does not leave the segment

example

advantages and disadvantages:

- criterion is combined with other criteria



# Criteria for Segmentation

## *– spatial criterion*

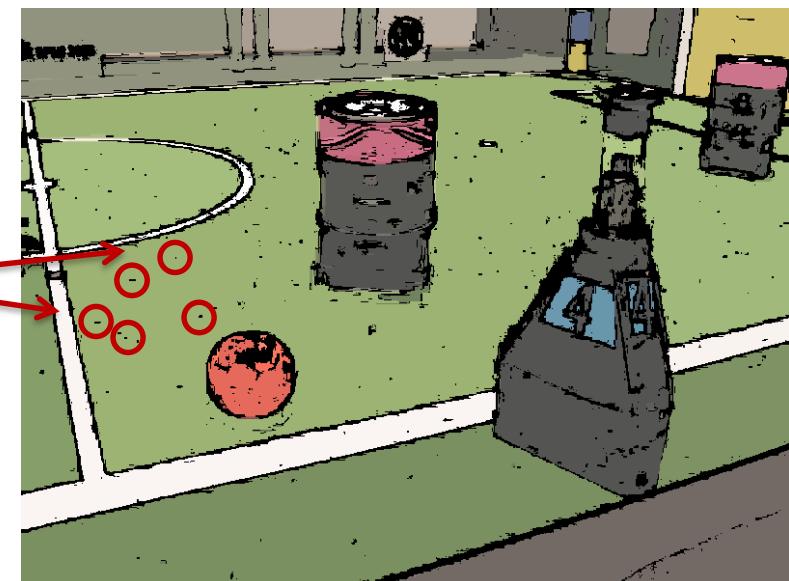
pixels which are surrounded by pixels of another segment should belong to that segment

example

advantages and disadvantages:

- criterion is combined with other criteria
- improves robustness w.r.t. noise

avoid/fill these gaps



# Criteria for Segmentation

## *– boundary smoothness criterion*

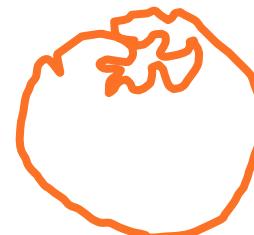
the boundary of segments should be smooth, not ragged.

example

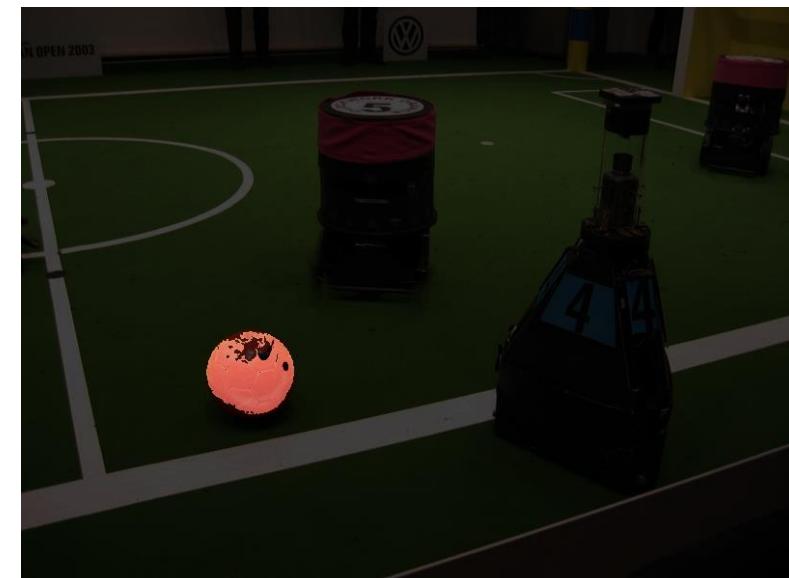
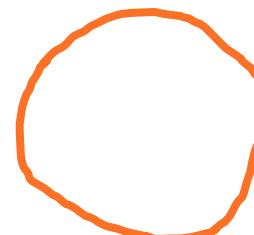
advantages and disadvantages:

- criterion is combined with other criteria
- improves robustness w.r.t. noise

ragged boundary – bad



smooth boundary – better



# Criteria for Segmentation

## – *size criteria*

the size of a segment should be within a range/not too small/not too large

# Segmentation Algorithms

- basic segmentation algorithms:
  - *region growing*
  - *connected components labeling*
  - *k-means and mean-shift algorithm*
  - *morphological operations*
- more elaborated algorithms:
  - *level set methods*
  - *random fields*

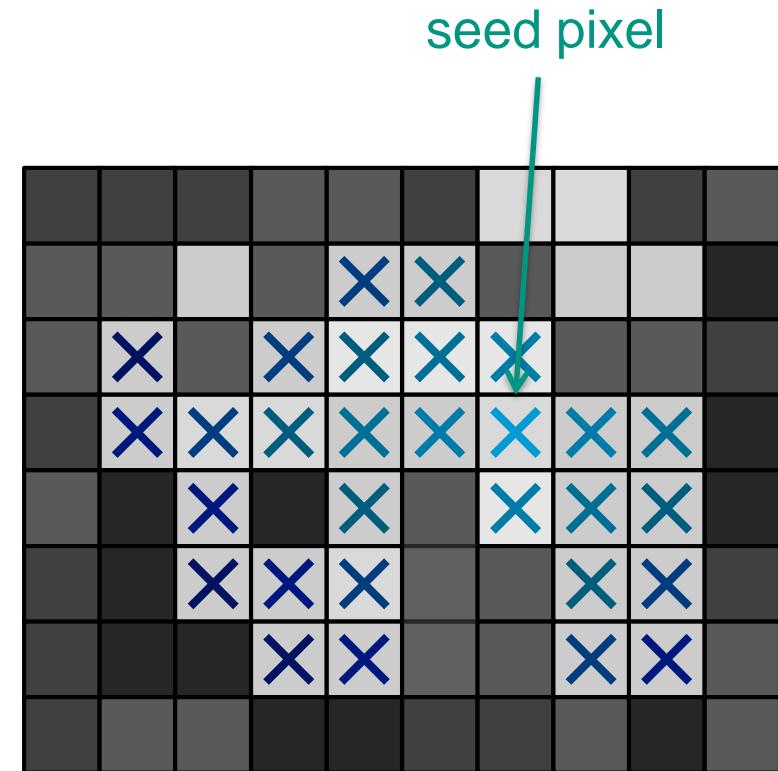
# Region Growing

– key idea:

- start from one/more seed points (seed points must be provided)
- incrementally expand segment until no more pixel can be added
- implements connectedness criterion + homogeneity or neighborhood criterion
- yields single segment

– advantages and disadvantages:

- easy to implement (breadth-first-search)
- requires one or more seed points



no more extension possible

# Region Growing cont.



# Connected Components Labeling (CCL)

– key idea:

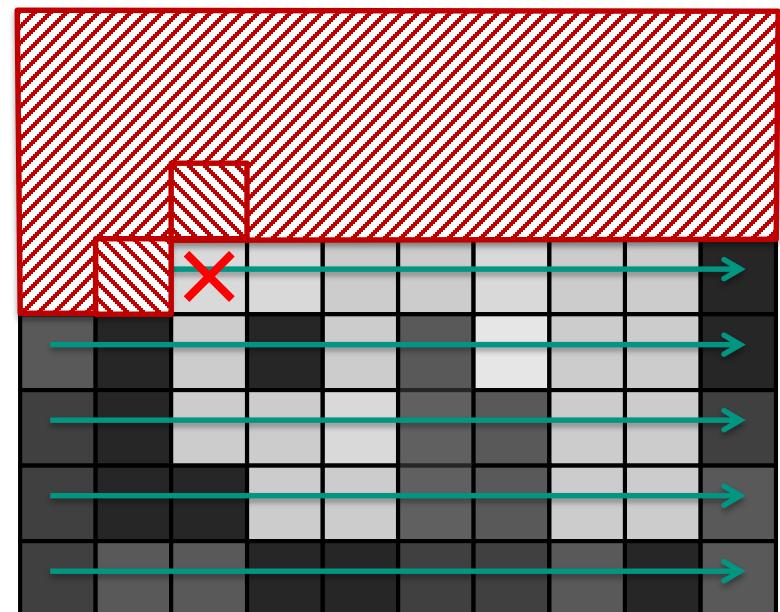
- create a full segmentation of the image
- implements connectedness criterion + neighborhood criterion
- assign each pixel to segment only by determining similarity with two neighboring pixels



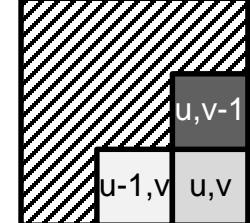
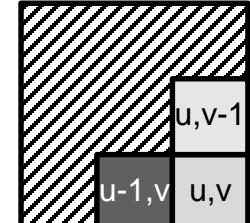
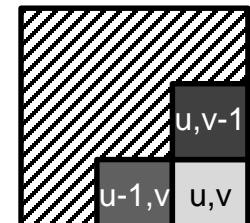
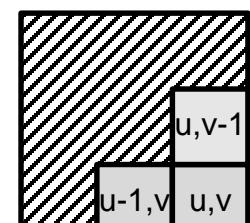
# Connected Components Labeling (CCL)

– procedure:

- we visit pixels row-by-row from the left upper corner to the right lower corner and immediately assign them to a segment
- when we visit a pixel  $(u,v)$  we already visited  $(u-1,v)$  and  $(u,v-1)$
- we compare color( $u,v$ ) with color( $u-1,v$ ), color( $u,v-1$ ). Five cases

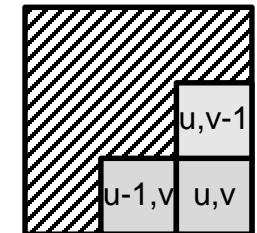


# Connected Components Labeling (CCL)

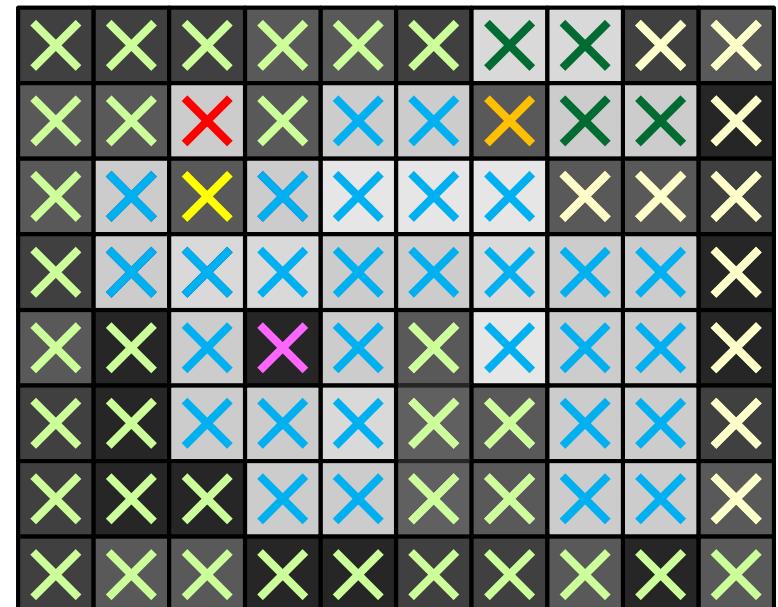
1. pixel colors at  $(u,v)$  and  $(u-1,v)$  are similar  
pixel colors at  $(u,v)$  and  $(u,v-1)$  are dissimilar  
→ pixel  $(u,v)$  and  $(u-1,v)$  belong to the same segment  
→ we assign pixel  $(u,v)$  to the segment of pixel  $(u-1,v)$ 
2. pixel colors at  $(u,v)$  and  $(u-1,v)$  are dissimilar  
pixel colors at  $(u,v)$  and  $(u,v-1)$  are similar  
→ pixel  $(u,v)$  and  $(u,v-1)$  belong to the same segment  
→ we assign pixel  $(u,v)$  to the segment of pixel  $(u,v-1)$ 
3. pixel colors at  $(u,v)$  and  $(u-1,v)$  are dissimilar  
pixel colors at  $(u,v)$  and  $(u,v-1)$  are dissimilar  
→ why should pixel  $(u,v)$  belong to the segments of  $(u-1,v)$  or  $(u,v-1)$ ?  
→ we create a new segment and assign pixel  $(u,v)$  to it
4. pixel colors at  $(u,v)$  and  $(u-1,v)$  are similar  
pixel colors at  $(u,v)$  and  $(u,v-1)$  are similar  
pixels  $(u-1,v)$  and  $(u,v-1)$  belong to the same segment  
→ pixel  $(u,v)$  also belongs to that segment  
→ we assign pixel  $(u,v)$  to that segment

# Connected Components Labeling (CCL)

5. pixel colors at  $(u,v)$  and  $(u-1,v)$  are similar
- pixel colors at  $(u,v)$  and  $(u,v-1)$  are similar
- pixels  $(u-1,v)$  and  $(u,v-1)$  do not belong to the same segment
  - pixel  $(u,v)$  belongs to the segments of both neighbors
  - we merge the two neighboring segments and assign pixel  $(u,v)$  to the merged segment



- Example



# Connected Components Labeling (CCL)



CCL on RGB values  
pixels are similar if  
Euclidean distance in color  
space is below  $10/255$

30 areas with more than  
100 pixels

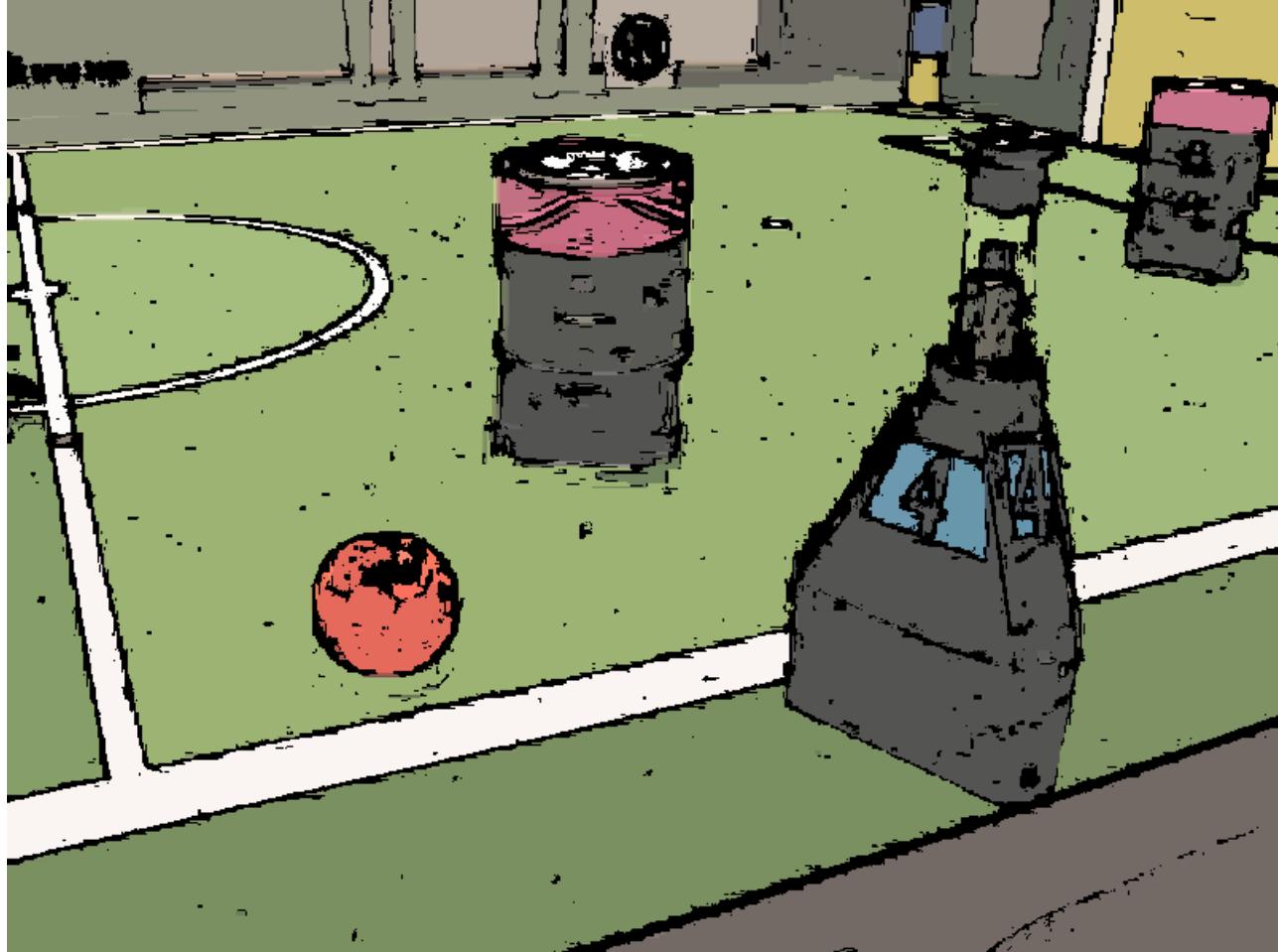


# Connected Components Labeling (CCL)



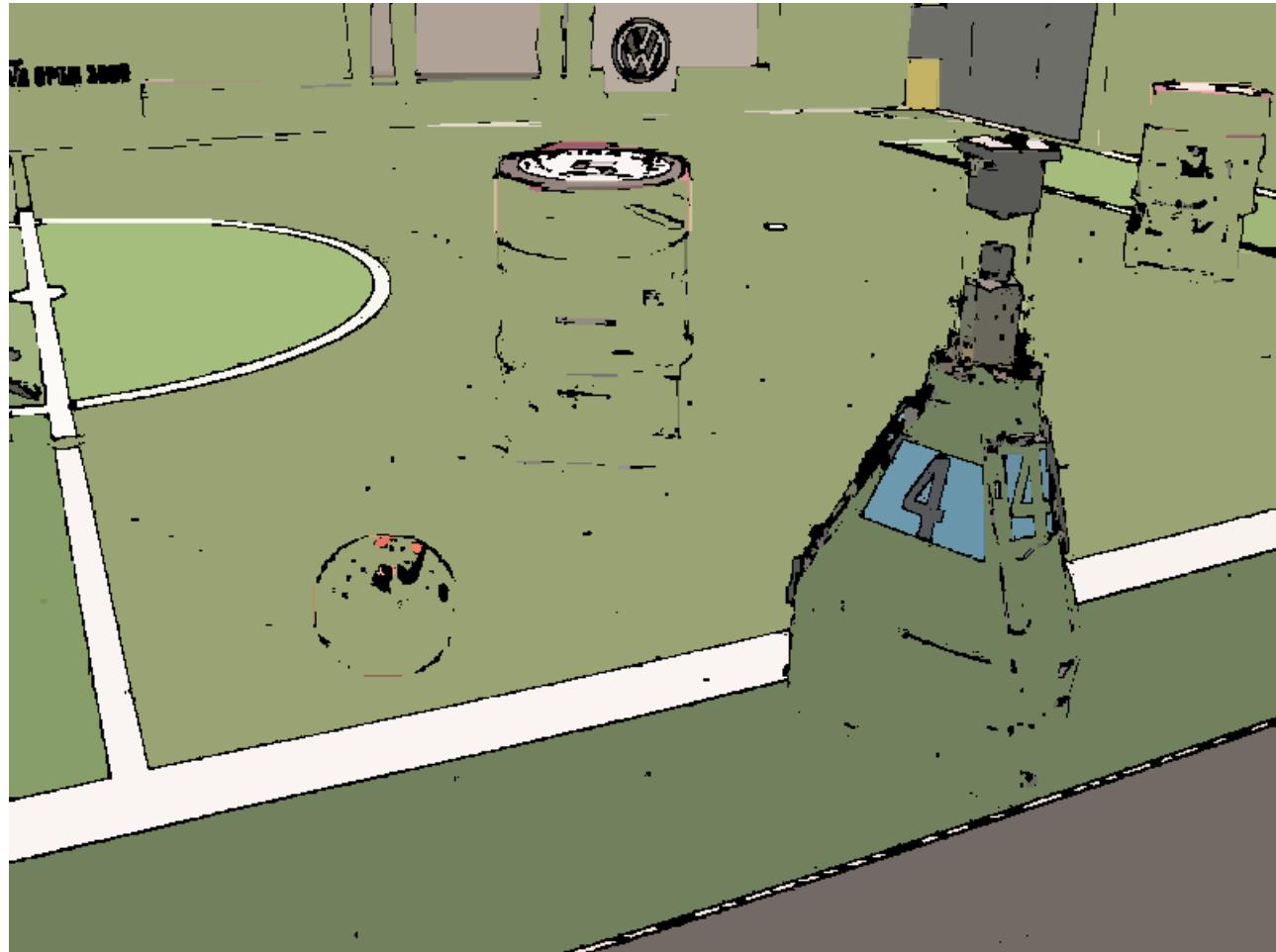
CCL on RGB  
 $\theta=2/255$  (Euclidean distance)  
2418 areas with more than 10 pixels

# Connected Components Labeling (CCL)



CCL on RGB  
 $\theta=7/255$  (Euclidean distance)  
730 areas with more than 10 pixels

# Connected Components Labeling (CCL)



CCL on RGB  
 $\theta=15/255$  (Euclidean distance)  
304 areas with more than 10 pixels

# k-means

## – key idea:

- image is composed out of areas of similar color
- find clusters of color
- assign each pixel to its color cluster
- implements homogeneity criterion
- creates full segmentation

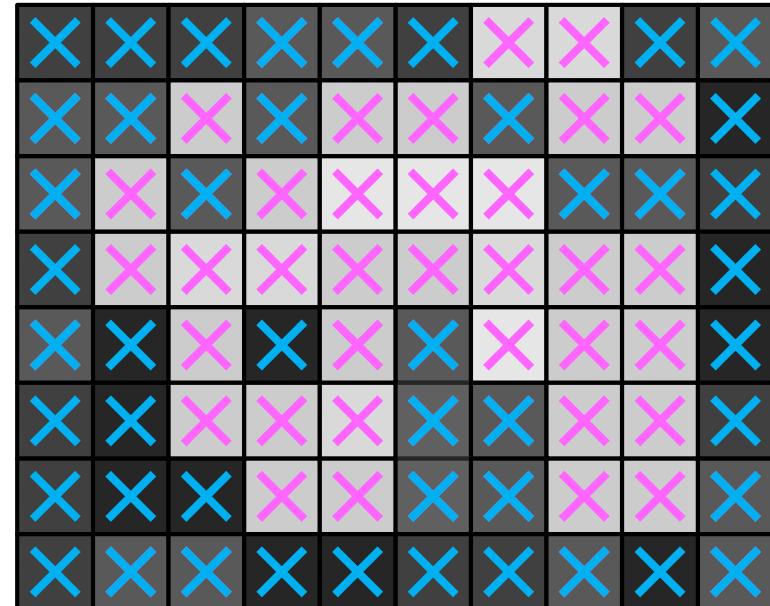


color clusters in the robot soccer picture:

- green
- white
- orange
- black
- magenta
- blue
- yellow
- gray

# k-means

- how can we find color clusters?
  - if we know the number of clusters
    - k-means algorithm
1. initialize  $k$  prototype colors  $c_1, c_2, \dots, c_k$  randomly (e.g. by randomly picking pixels from image)
  2. assign each pixel to the prototype color that is most similar
  3. recalculate prototype colors by averaging over colors of pixel which have been assigned in step 2
  4. repeat steps 2 and 3 until convergence (i.e. the assignments in step 2 do not change any more)



example:  $k=2$

step 1: randomly pick colors  
from two pixels



step 2: assign pixels to  
most similar cluster



step 3: recalculate  
prototype colors



step 2: reassign pixels



step 3: recalculate  
prototype colors

step 2: reassign pixels →  
convergence

# k-means

- Examples:

original image



k=5, iteration=10



# k-means

- Examples:



no light blue prototype

# k-means

- Examples:



suboptimal prototype colors: only  
one prototype for yellow+orange

# Mean-Shift

## – k-means algorithm

- advantage:
  - simple, easy to implement
- disadvantages:
  - number of clusters ( $k$ ) must be known
  - often converges into suboptimal clustering (depending on initial prototype colors)

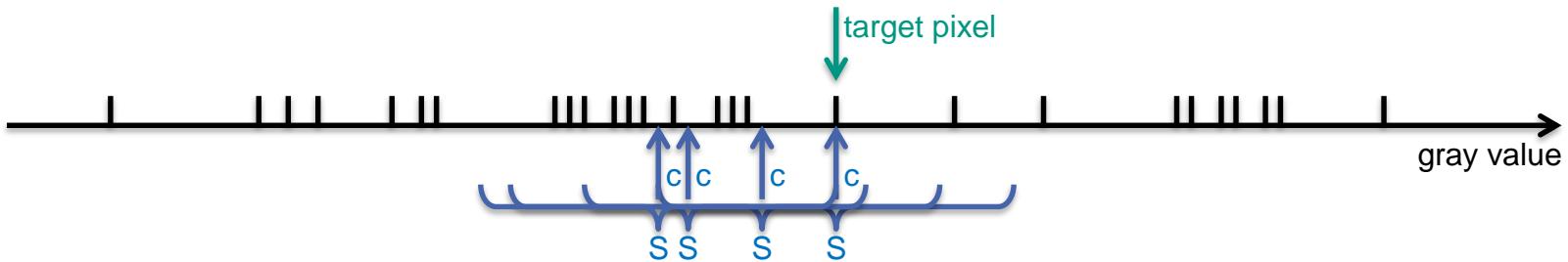
## – improvement for unknown number of clusters → mean-shift

- requires a similarity measure for colors      通过计算像素密度高峰，动态调整像素归属。
- for each pixel  $p$  proceed as follows
  1. determine color of  $p$  and assign it to variable  $c$
  2. find the set  $S$  of all pixels in image which are similar to  $c$
  3. calculate the average color of  $S$  and assign it to variable  $c$  (do not change pixel value of  $p$  in image!)
  4. repeat steps 2 and 3 until convergence (i.e. until  $S$  remains unchanged in step 2)
  5. finally,  $c$  is the prototype color of the segment which the pixel  $p$  belongs to

# Mean-shift

## – example

arranged all pixel colors (gray values) along one axis



- step 1: pick color of target pixel and initialize  $c$
- step 2: find the set of similar pixels  $S$
- step 3: calculate average color of  $S$  and assign it to  $c$
- step 2: recalculate  $S$
- step 3: recalculate average color of  $S$  and assign it to  $c$
- step 2: recalculate  $S$
- step 3: recalculate average color of  $S$  and assign it to  $c$
- step 2: recalculate  $S \rightarrow$  convergence

# Mean-shift

使用较窄的颜色相似性 (narrow sense) 分割的结果

- Examples:



mean shift  
narrow sense of  
color similarity



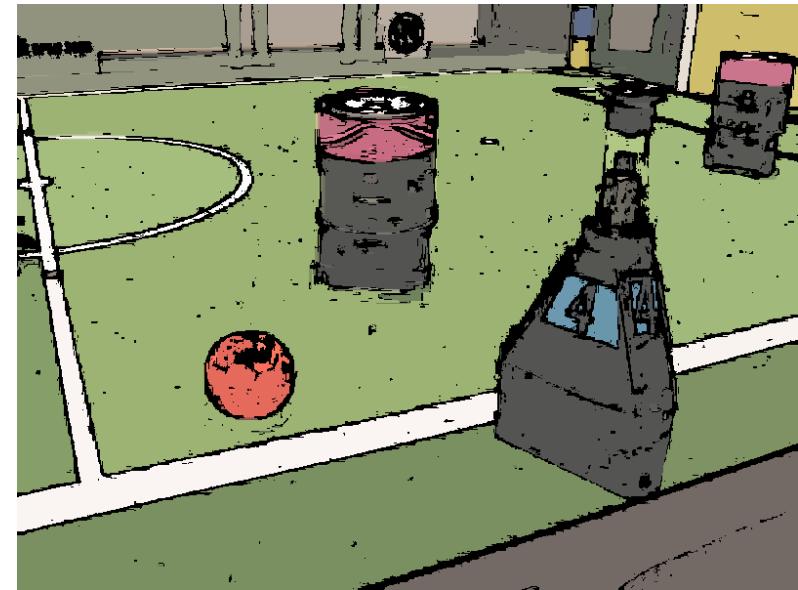
mean shift  
broad sense of  
color similarity



使用较宽的颜色相似性 (broad sense) 分割的结果。

# Morphological Operations

- problems:
  - holes
  - ragged contours
  - gaps
  - tiny areas
- extend/shrink areas
  - *erosion*: shrink area by one pixel
  - *dilation*: extend area by one pixel
- assumption
  - background pixels are encoded with 0
  - foreground pixels are encoded with numbers  $\geq 1$

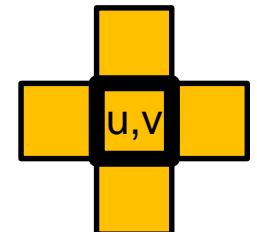


# Morphological Operations cont.

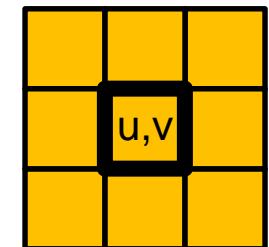
- Erosion:

$$\text{erode}\{g\}(u, v) = \min\{g(u, v), g(u + 1, v), g(u + 1, v + 1), g(u, v + 1), g(u - 1, v + 1), g(u - 1, v), g(u - 1, v - 1), g(u, v - 1), g(u + 1, v - 1)\}$$

*“take the minimal value of neighbors”*



4-neighborhood  
of pixel (u,v)

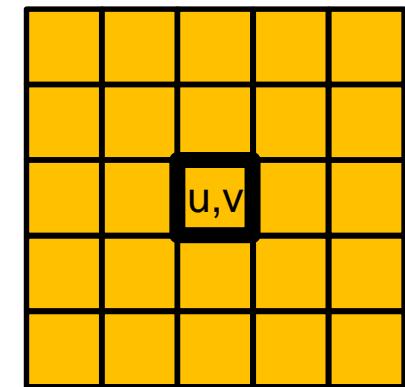


8-neighborhood  
of pixel (u,v)

- Dilation:

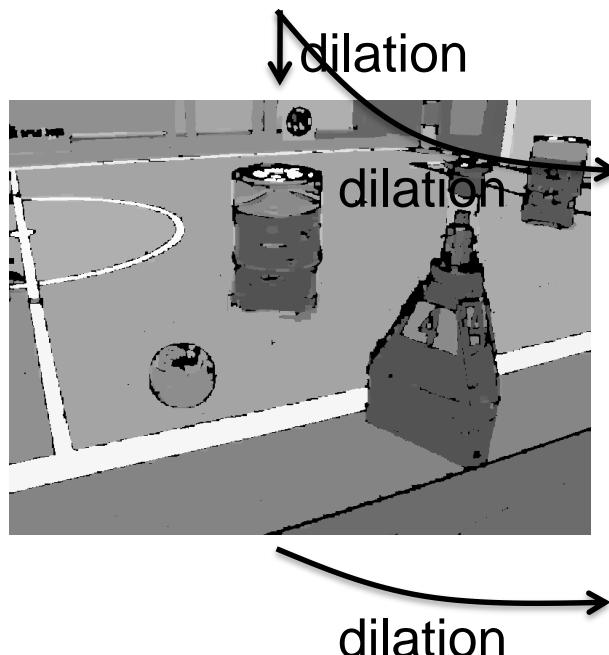
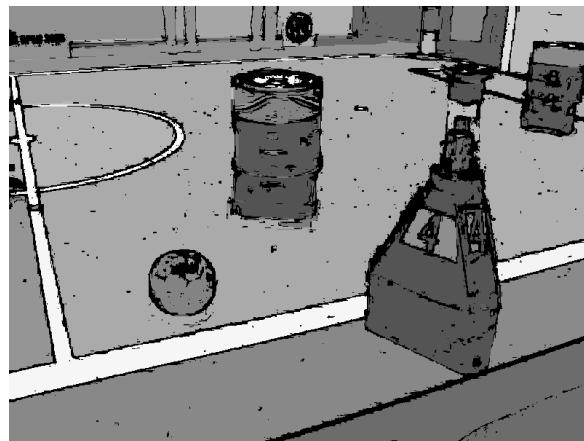
$$\text{dilate}\{g\}(u, v) = \max\{g(u, v), g(u + 1, v), g(u + 1, v + 1), g(u, v + 1), g(u - 1, v + 1), g(u - 1, v), g(u - 1, v - 1), g(u, v - 1), g(u + 1, v - 1)\}$$

*“take the maximal value of neighbors”*



24-neighborhood  
of pixel (u,v)

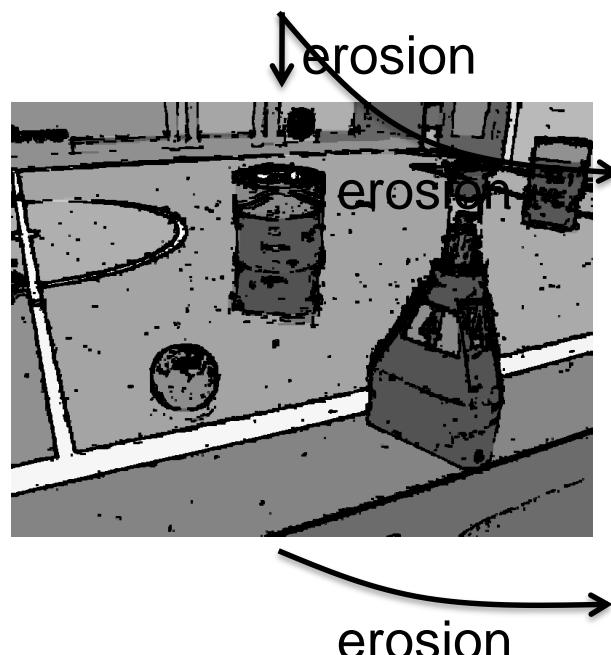
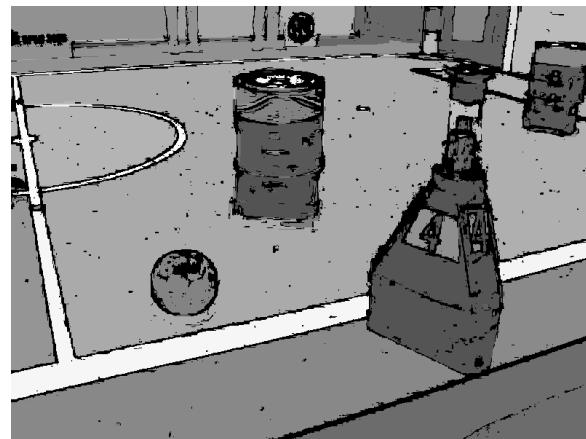
# Morphological Operations cont.



dilation fills holes and gaps



# Morphological Operations cont.

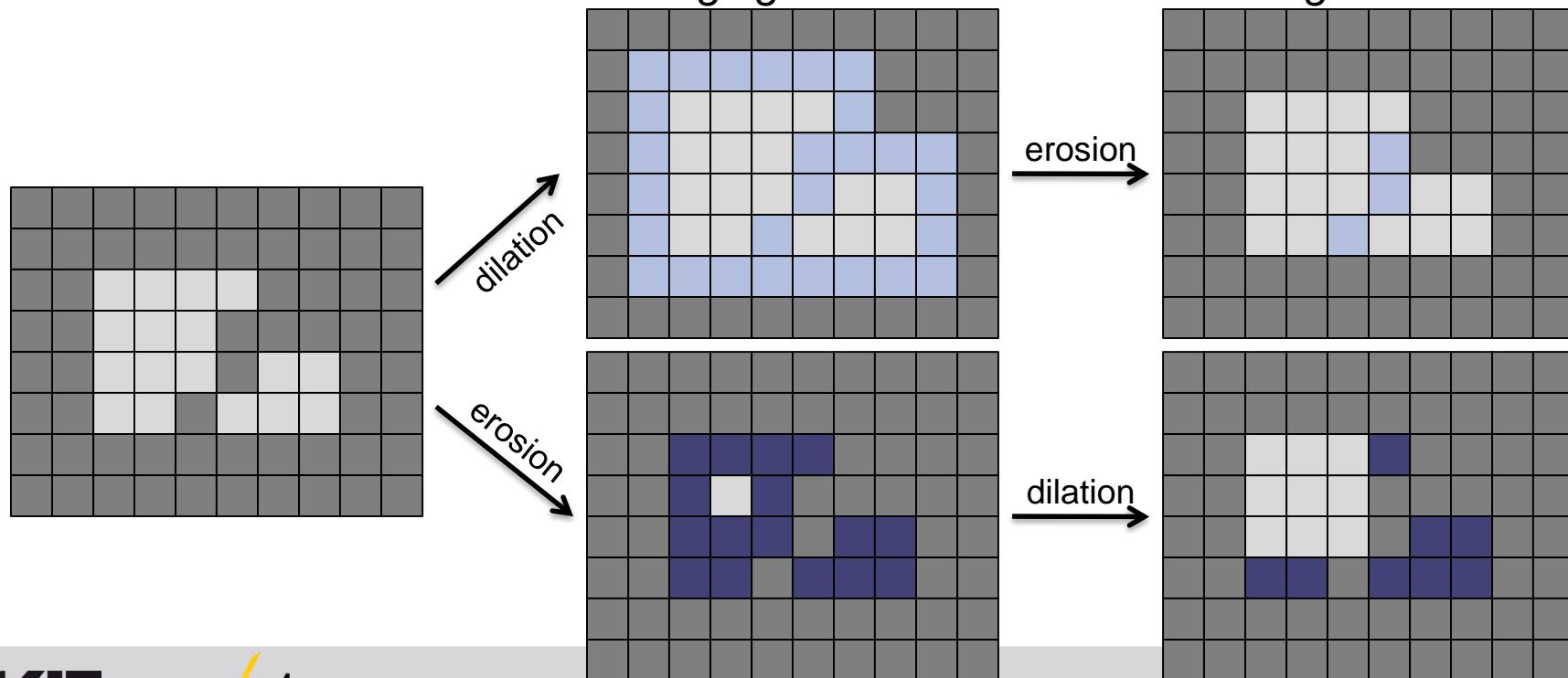


erosion eliminates thin structures



# Morphological Operations cont.

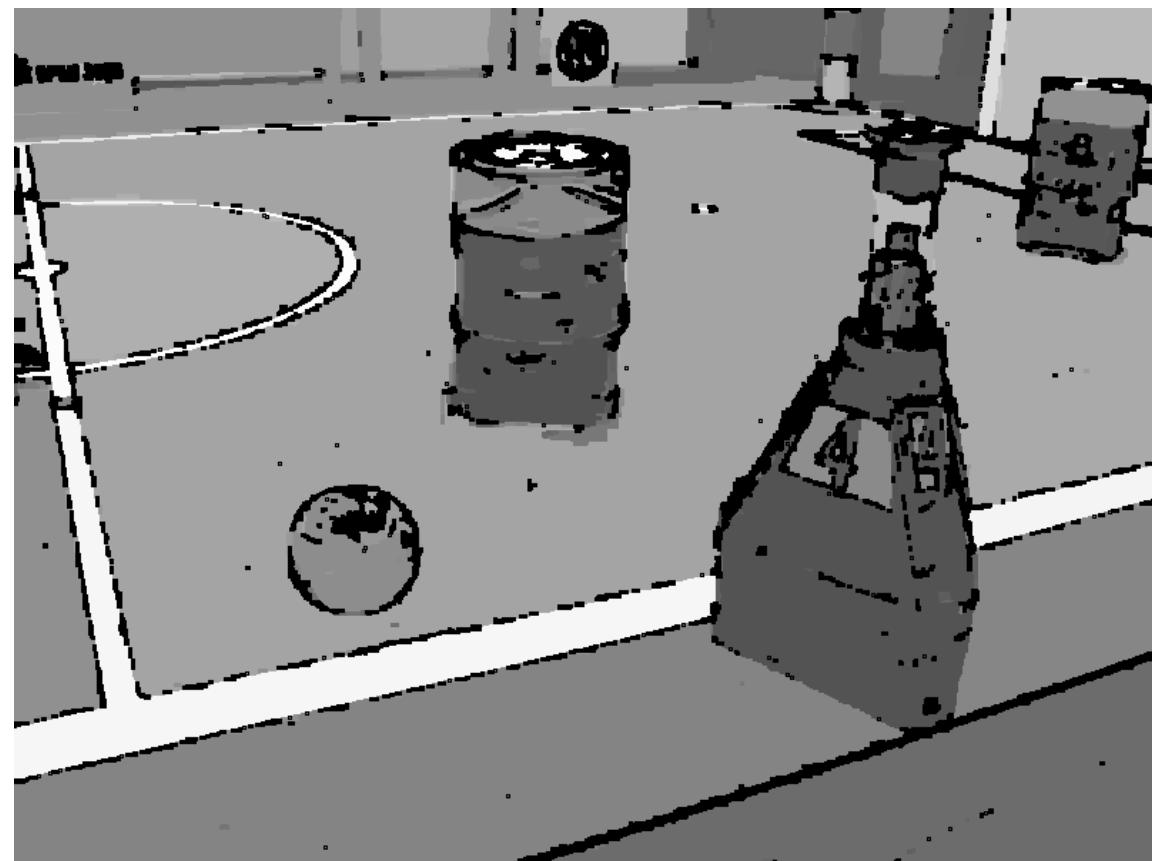
- erosion and dilation can be combined:
  - *closing*: first dilation, then erosion  
fill gaps and holes without changing the overall extension of areas
  - *opening*: first erosion, then dilation  
*remove thin areas without changing the overall extension of large areas*



# Morphological Operations cont.



closing



# Morphological Operations cont.

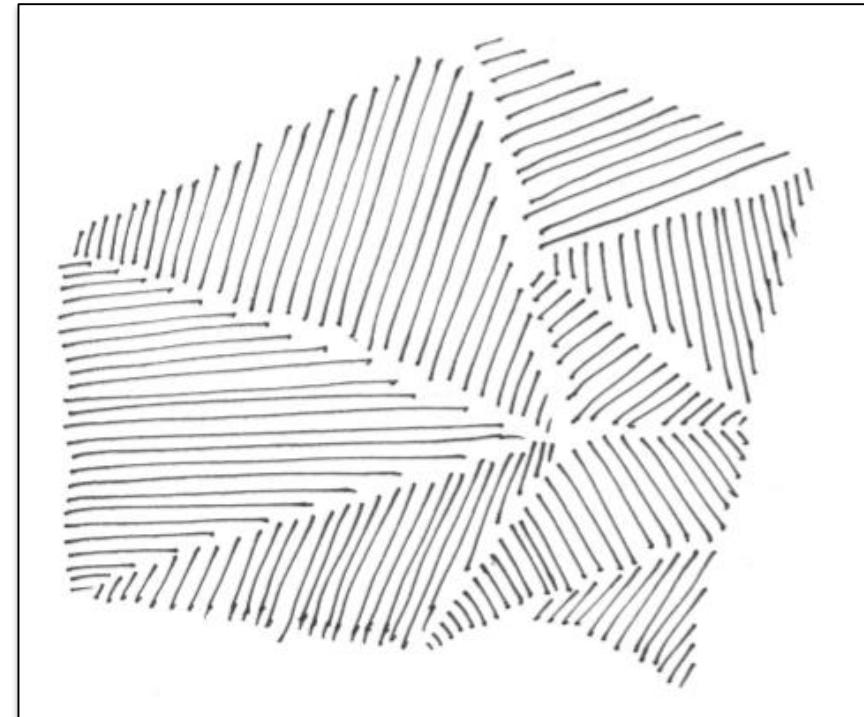


opening



# Generic View on Segmentation

- So far:
  - segmentation was based on color (gray values)
  - different representation of color and different similarity measures
- Question:
  - how can we segment images in which colors are not salient?  
突出的
- Example:
  - segment image into areas of same hatching



# Generic View on Segmentation

- What do we need for image segmentation?
  - for every pixel: a description of the pixel (**image features**)
    - e.g. color
    - e.g. texture information
    - e.g. depth of point (3d scanner/stereo vision)
    - e.g. motion of pixel (optical flow)
    - e.g. features which characterize whether pixel belongs to certain object categories
    - e.g. a combination of those features
  - a **measure of similarity** of different pixels
    - e.g. Euclidean distance between feature vectors
    - e.g. other metric
  - one/more **segmentation criteria**
    - *cf. slide 3*
  - an **efficient algorithm** that implements the segmentation criteria
    - *cf. methods presented on previous slides*



# Generic View on Segmentation

- Example:

- segment image into areas of same hatching
- image features:
  - color and gray level is not salient
  - orientation of lines is salient
  - e.g.
    - calculate gray level gradient
    - determine the dominant gradient direction in local environment around pixel
    - represent direction as 2d vector
    - length of vector is proportional to average gradient length
- criteria and algorithm:
  - neighborhood criterion
  - minimal segment size
  - connected components labeling

input image

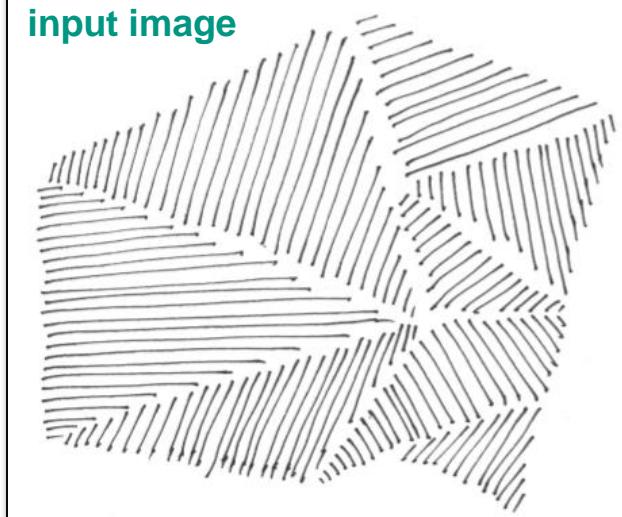
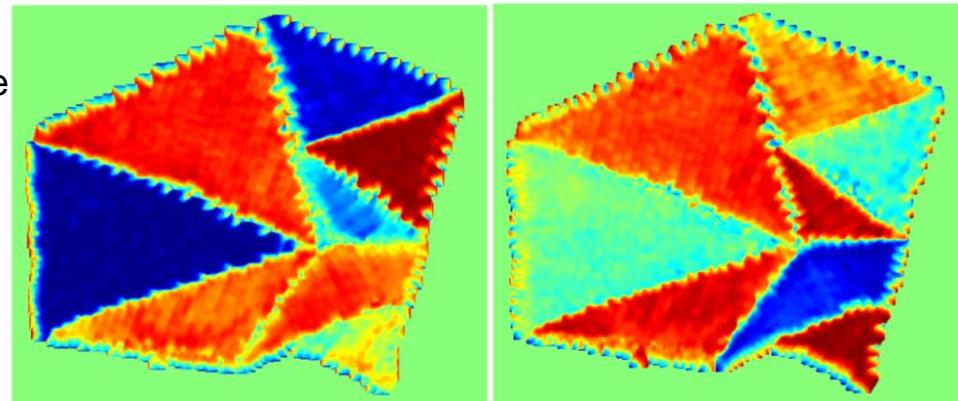


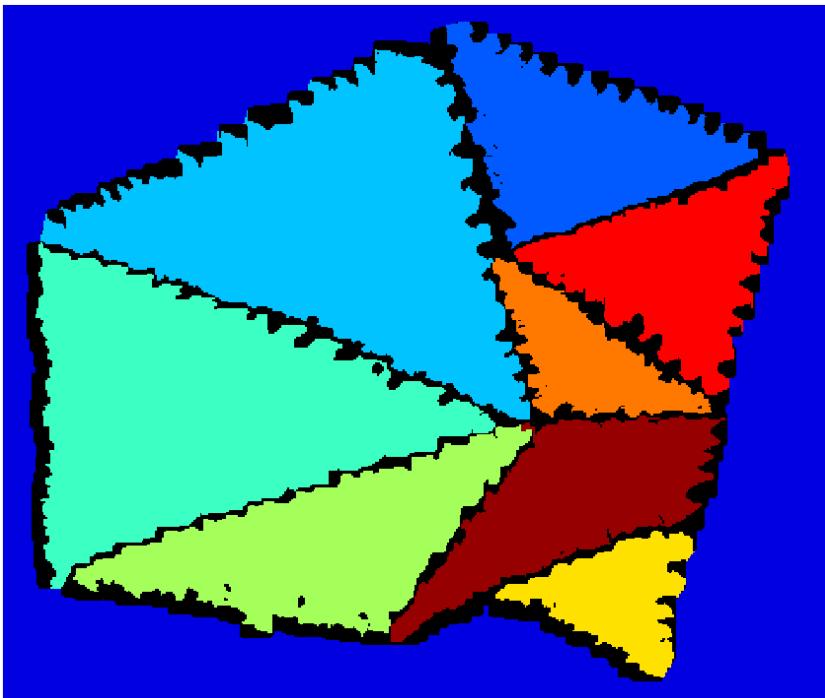
image features (illustrated as color images)



# Generic View on Segmentation

- Example:
  - segment image into areas of same hatching

segmentation result with CCL  
(one color per segment)



input image

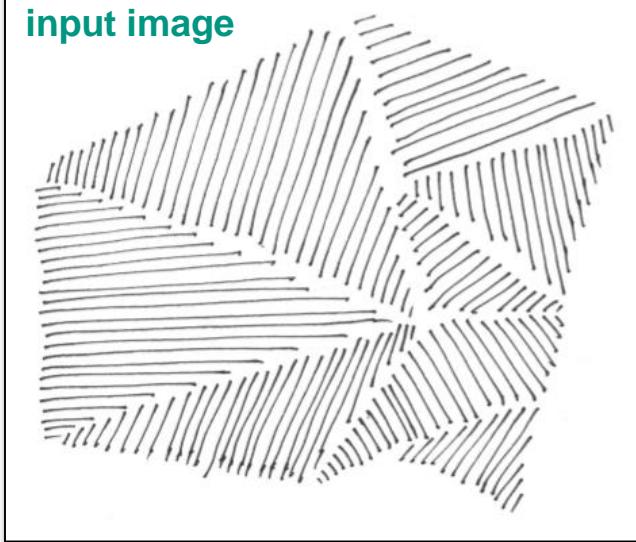
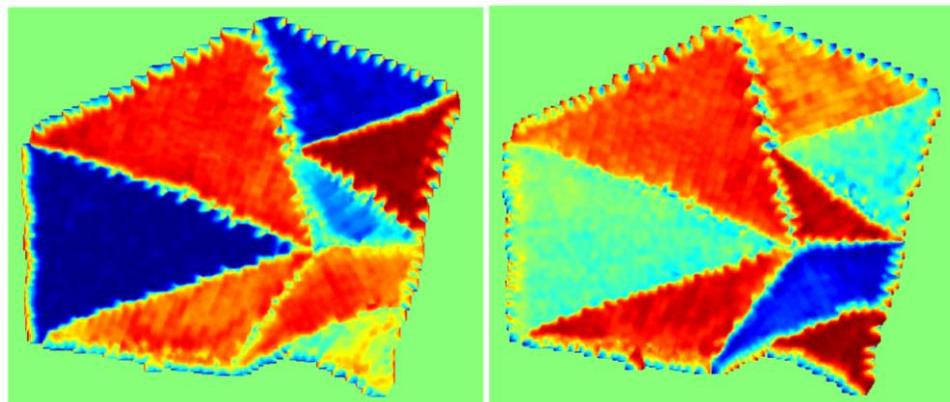


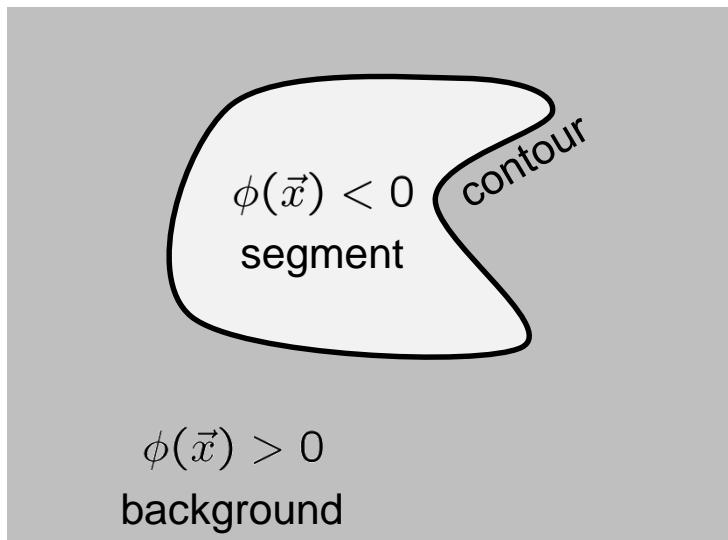
image features (illustrated as color images)



# LEVEL SET METHODS

# Level Set Representation

- two-class-segmentation can be represented by:
  - collection of all pixels that belong to segment
  - indicator function  $\phi(\vec{x}) \begin{cases} < 0 & \text{if pixel } \vec{x} \text{ belongs to segment} \\ > 0 & \text{if pixel } \vec{x} \text{ belongs to background} \end{cases}$
  - contour
  - signed distance function



# Level Set Representation cont.

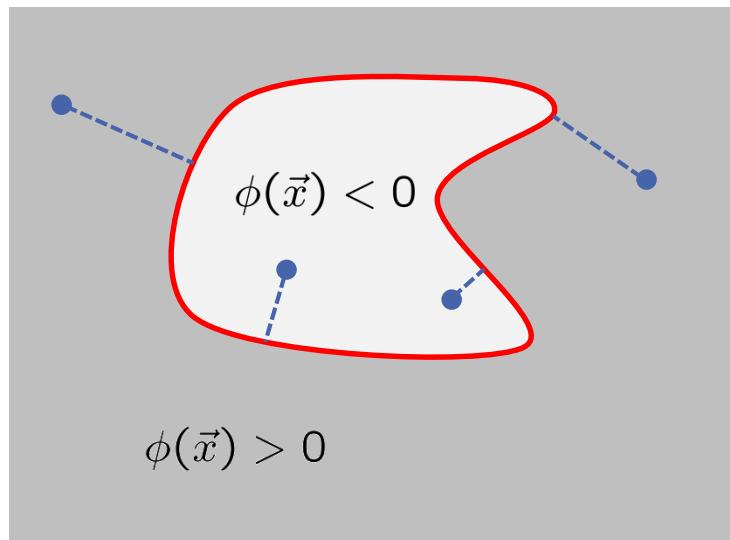
- signed distance function

$$\phi(\vec{x}) \begin{cases} < 0 & \text{if pixel } \vec{x} \text{ belongs to segment} \\ > 0 & \text{if pixel } \vec{x} \text{ belongs to background} \end{cases}$$

$|\phi(\vec{x})| =$  distance of  $\vec{x}$  from contour

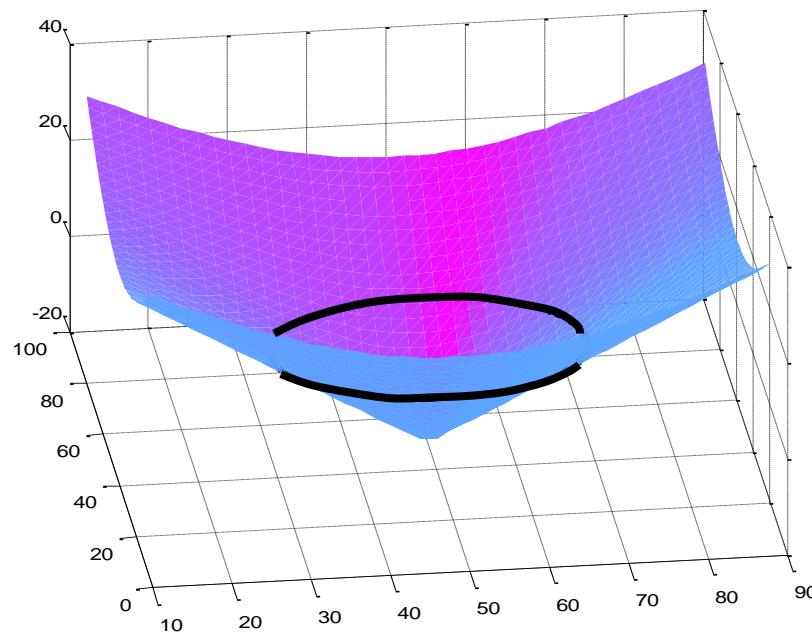
- contourpoints:

$$\phi(\vec{x}) = 0$$

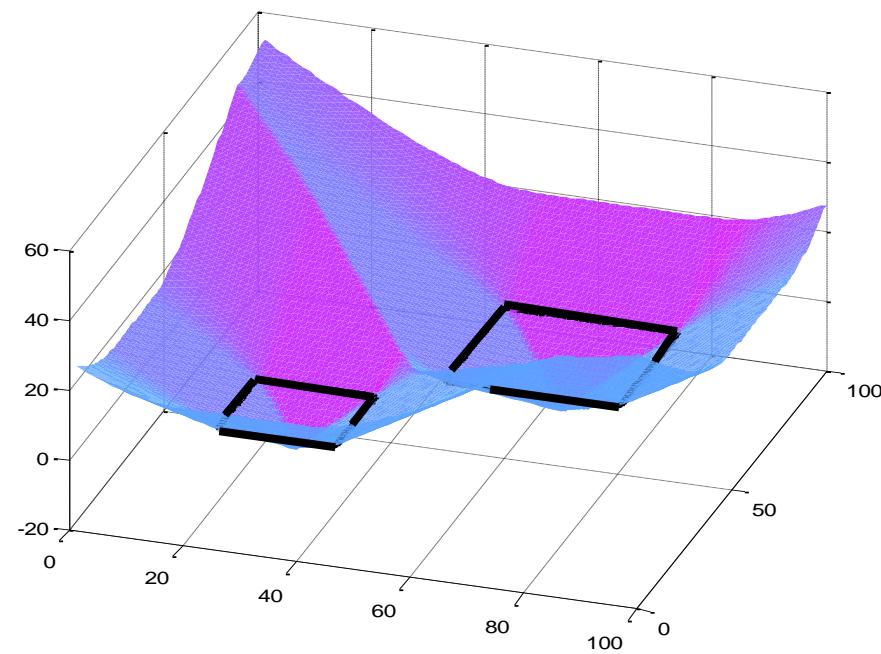


# Level Set Representation cont.

- signed distance function



example: circular contour



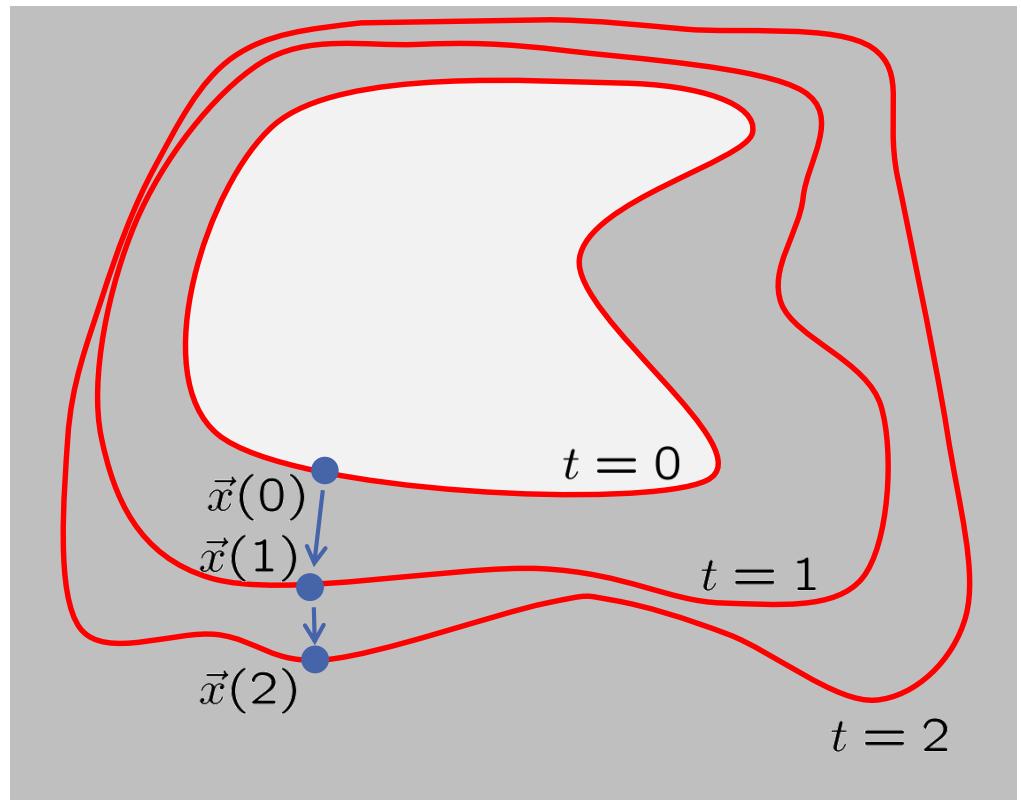
example: two rectangular contours

# Level Set Evolution

- modeling temporal evolution of signed distance function  
 $\phi(\vec{x}, t)$

- tracking a point on the boundary over time  $\vec{x}(t)$

- obviously:  
 $\phi(\vec{x}(t), t) = 0$   
for all  $t$



# Level Set Evolution cont.

- from:

$$0 = \phi(\vec{x}(t), t) \quad \text{for all } t$$

- follows:

$$0 = \frac{d\phi(\vec{x}(t), t)}{dt} = \nabla \phi \cdot \frac{\partial \vec{x}}{\partial t} + \frac{\partial \phi}{\partial t}$$

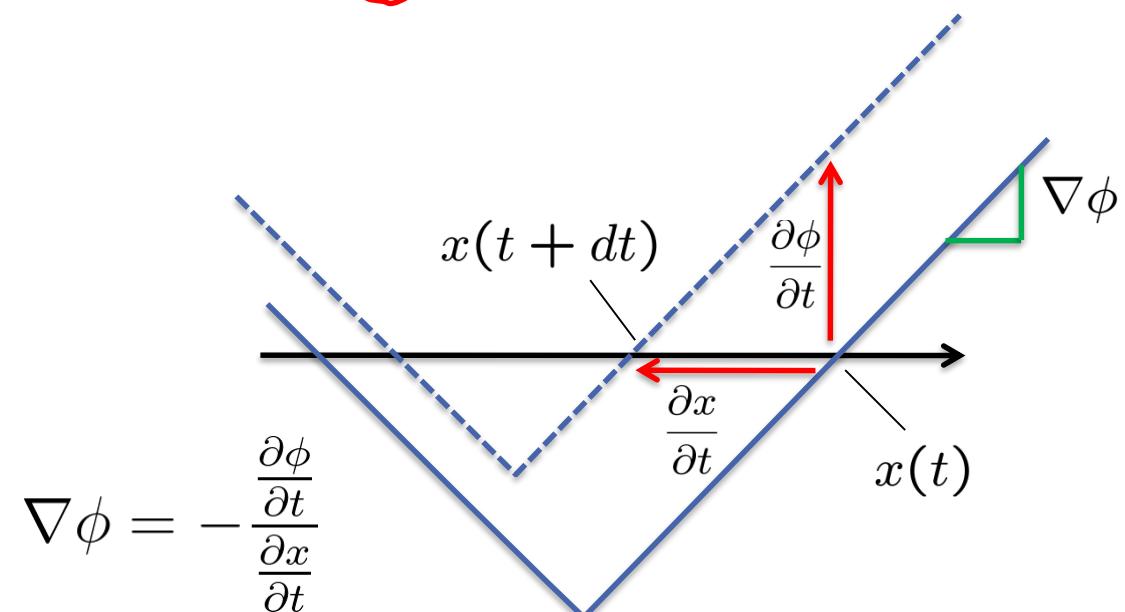
local structure of  $\phi$

movement of a contour point

evolution of  $\phi$

with:

$$\nabla \phi = \left( \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y} \right)$$



# Level Set Evolution cont.

– resolving w.r.t.  $\frac{\partial \phi}{\partial t}$ :

$$\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \frac{\partial \vec{x}}{\partial t}$$

- Basic idea of level set methods:
  - start with initial  $\phi(\cdot, 0)$
  - assume reasonable  $\frac{\partial \vec{x}}{\partial t}$
  - track  $\phi(\cdot, t)$  over time
- Implementation using numerical integration, e.g. Euler's approach (tricky!)

# Expanding and Shrinking

- evolution orthogonal to contour

$$\frac{\partial \vec{x}}{\partial t} = \alpha \cdot \frac{\nabla \phi}{\|\nabla \phi\|}$$

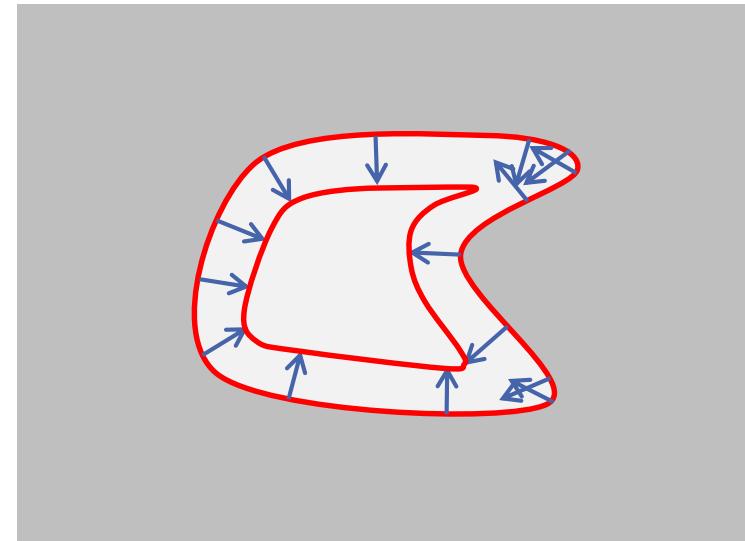
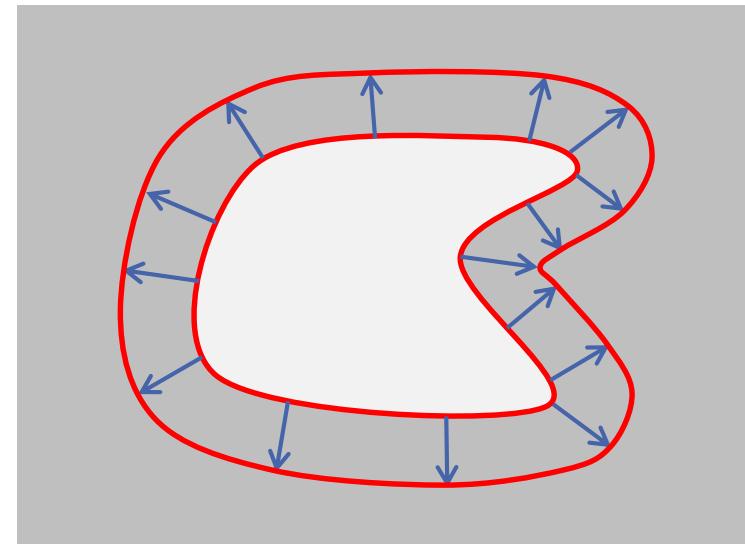
$$\begin{aligned}\frac{\partial \phi}{\partial t} &= -\nabla \phi \cdot \alpha \cdot \frac{\nabla \phi}{\|\nabla \phi\|} \\ &= -\alpha \frac{\|\nabla \phi\|^2}{\|\nabla \phi\|} = -\alpha \|\nabla \phi\|\end{aligned}$$

– 1. case  $\alpha > 0$

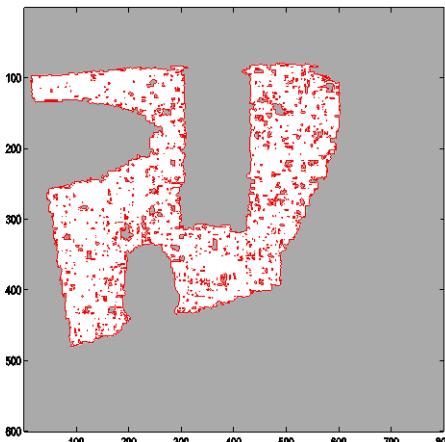
contour expands

– 2. case  $\alpha < 0$

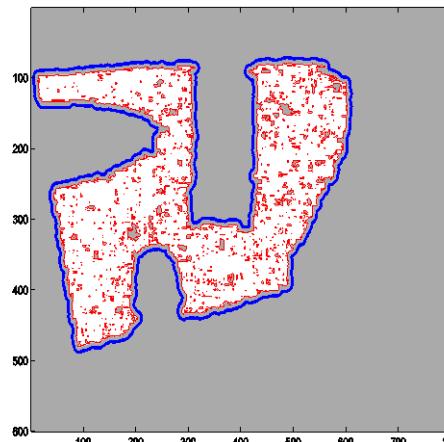
contour shrinks



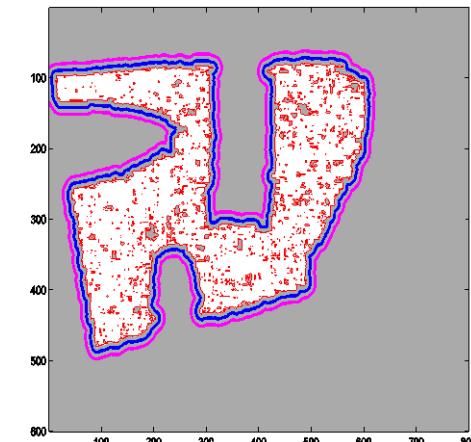
# Expanding and Shrinking cont.



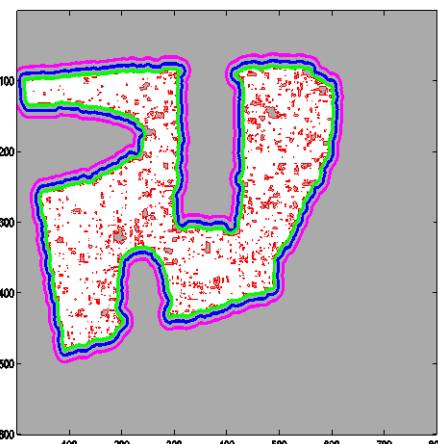
initial contour



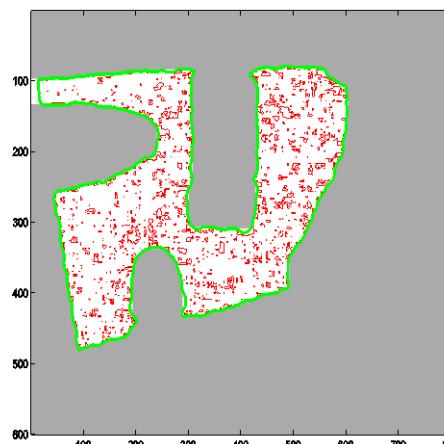
expanding



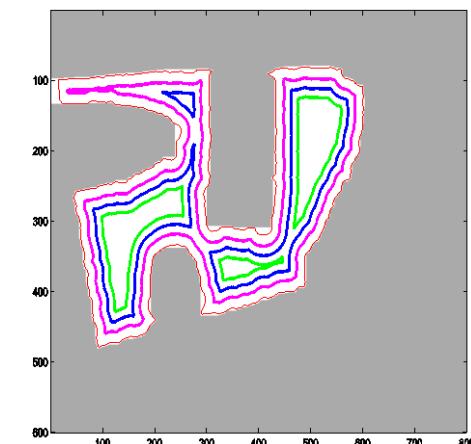
expanding



shrinking



comparison:  
closing operator



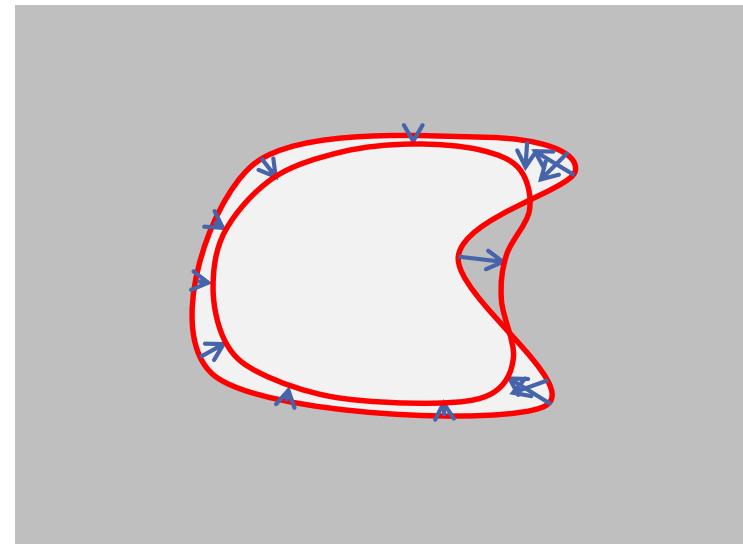
shrinking

# Expanding and Shrinking cont.

- level set evolution can be used to implement morphological operators:
  - dilation = expanding
  - erosion = shrinking
  - closing = shrinking after expanding
  - opening = expanding after shrinking

# Contour Rectification

- making the contour smoother
  - expanding in concave areas
  - shrinking in convex areas
- evolving the level set
  - orthogonal to contour
  - depending on local curvature  $\kappa$

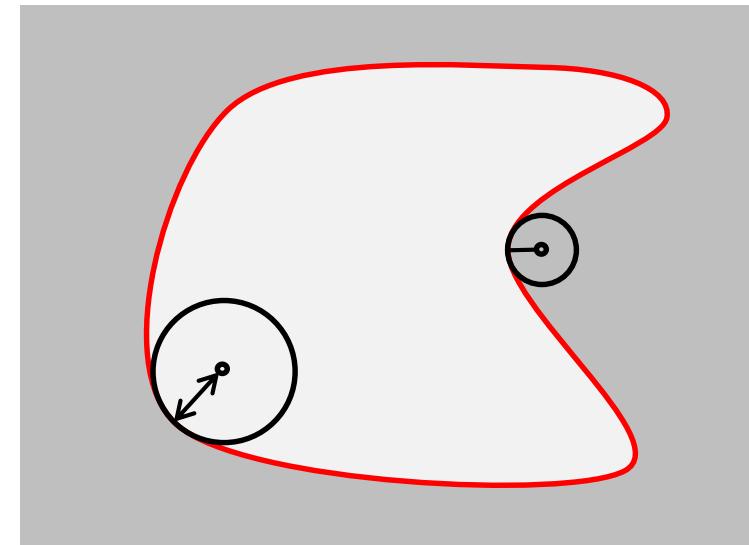


# Contour Rectification cont.

- curvature  $\kappa$ 
  - in convex areas  $\kappa = 1/r$  of circle that locally approximates contour
  - in concave areas:  $\kappa = -1/r$  of circle that locally approximates contour
  - in general:  $\kappa = \nabla \left( \frac{\nabla \phi}{\|\nabla \phi\|} \right)$
- level set update:

$$\frac{\partial \vec{x}}{\partial t} = -\beta \kappa \frac{\nabla \phi}{\|\nabla \phi\|}$$

$$\frac{\partial \phi}{\partial t} = \beta \kappa \|\nabla \phi\|$$



# Contour Rectification cont.

- Example:



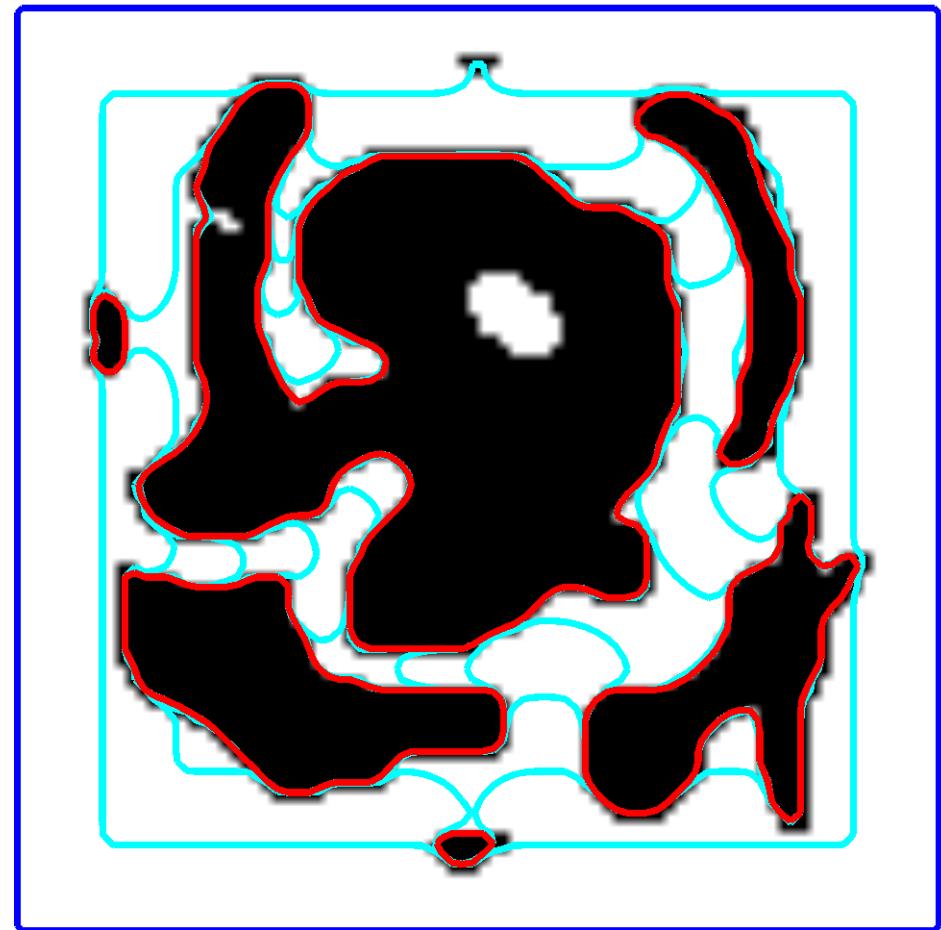
# Image Segmentation with Level Sets

- very simple idea for black/white images:
  - start with a very large contour 轮廓
  - shrink contour at white pixels
  - don't shrink at black pixels
  - contour enwraps black areas

$$\frac{\partial \vec{x}}{\partial t} = \begin{cases} -\gamma \cdot \frac{\nabla \phi}{\|\nabla \phi\|} & \text{if white pixel} \\ 0 & \text{if black pixel} \end{cases}$$

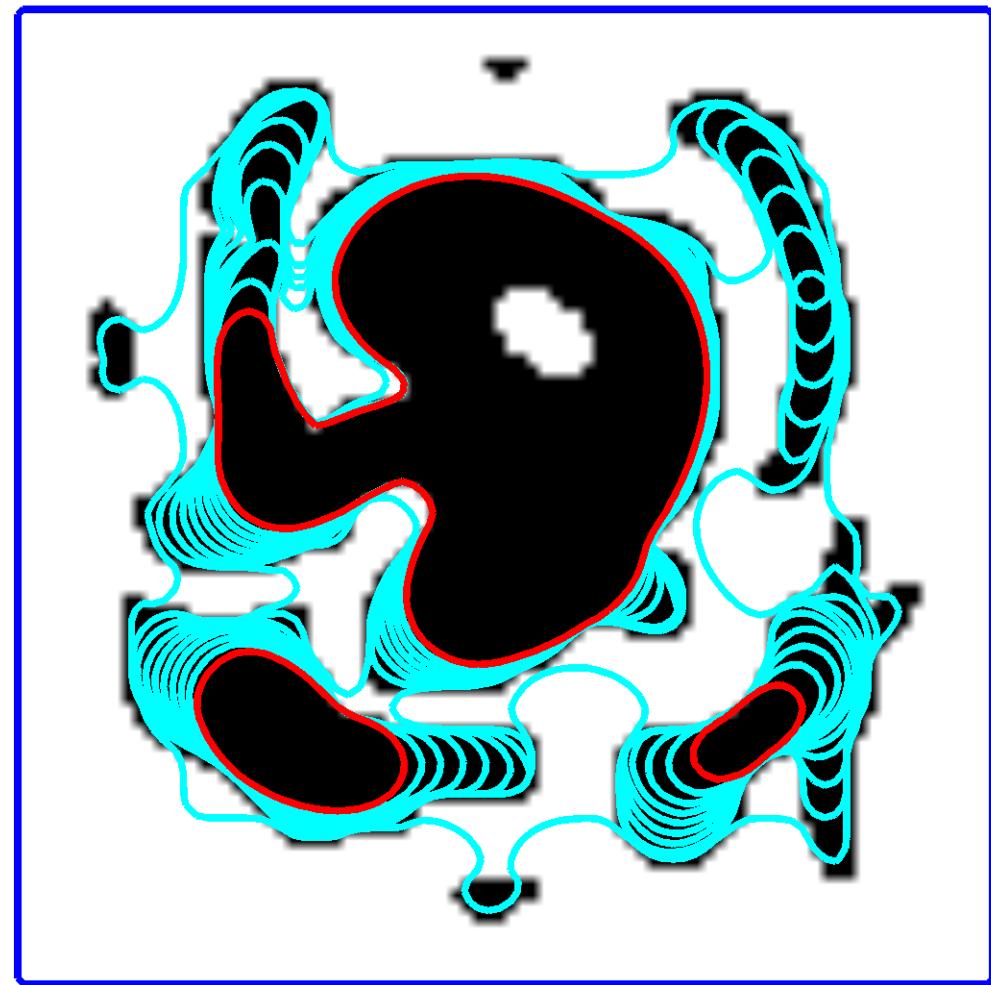
# Image Segmentation with Level Sets cont.

- Example:



# Image Segmentation with Level Sets cont.

- Example:  
combining segmentation  
with contour rectification



# Image Segmentation with Level Sets cont.

- gradient based approach for image segmentation:
    - start with a very large contour
    - shrink contour at pixels with small gradient length
    - don't shrink at pixels with large gradient length (edge pixels)
- contour enwraps areas bordered by edges

$$\frac{\partial \vec{x}}{\partial t} = -\epsilon(g) \cdot \frac{\nabla \phi}{||\nabla \phi||}$$
$$\epsilon(g) = \frac{\gamma}{\gamma + |Gauss * \nabla g|^p}$$

with appropriate  $\gamma > 0, p \geq 1$

$g$  denotes gray level image

# Image Segmentation with Level Sets cont.

- Example:



# Image Segmentation with Level Sets cont.

- Example:



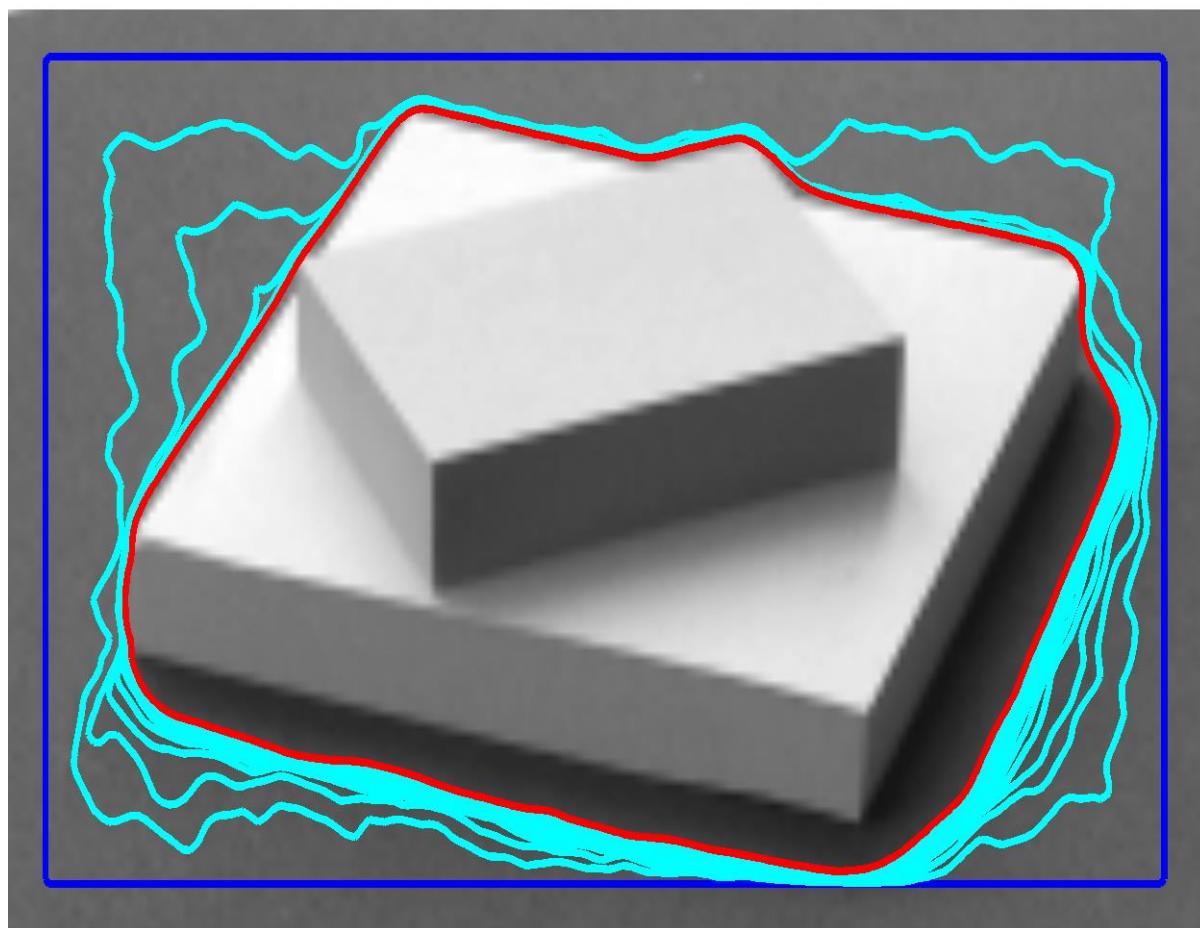
# Image Segmentation with Level Sets cont.

- Example:  
same as before, but with  
contour rectification



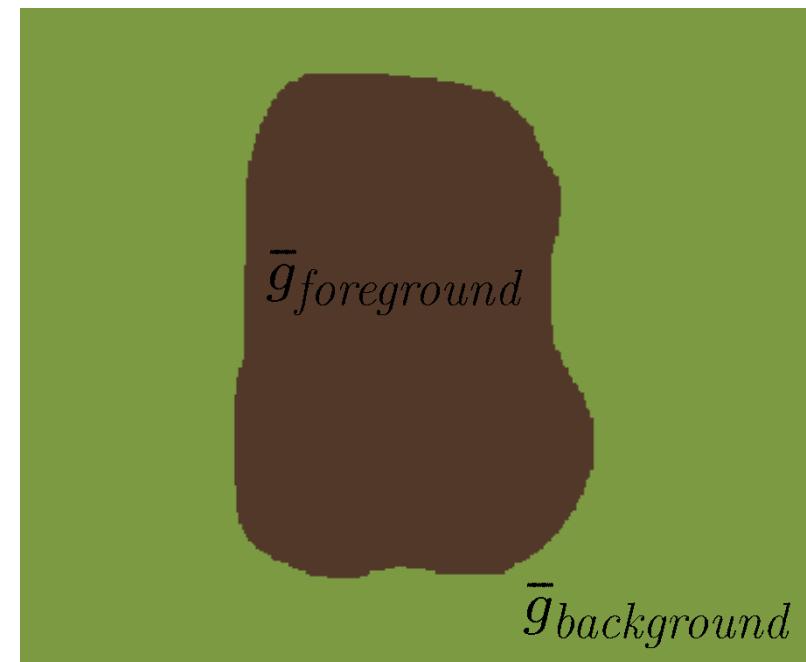
# Image Segmentation with Level Sets cont.

- Example:



# Image Segmentation with Level Sets cont.

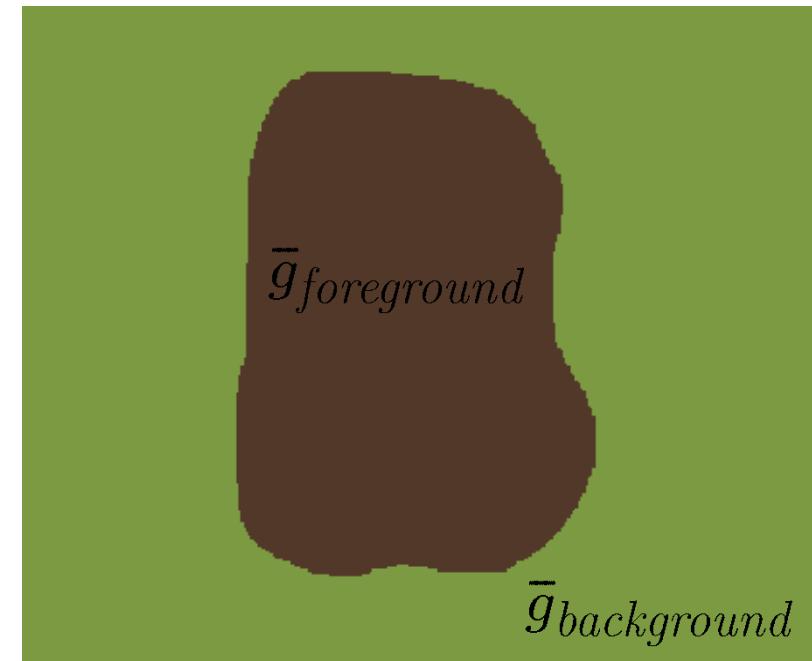
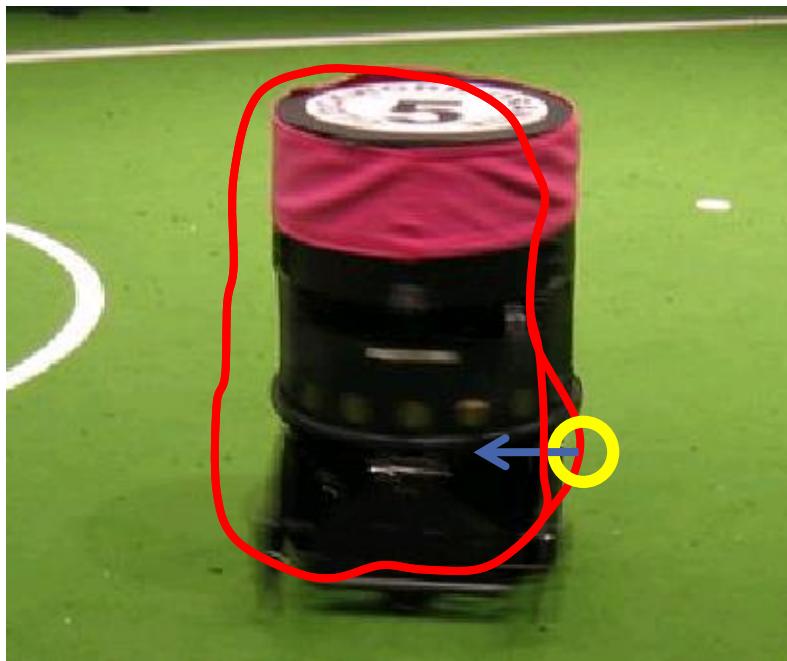
- Mumford-Shah based segmentation
  - idea: pixels should be assigned to the segment with the most similar grey values (color values)  
 $\bar{g}_{foreground}$  : average grey value (color) of pixels in foreground segment  
 $\bar{g}_{background}$  : average grey value (color) of pixels in background segment



# Image Segmentation with Level Sets cont.

- check for pixels on boundary with grey (color) value  $I$ 
  - pixel more similar to area outside
$$(g - \bar{g}_{foreground})^2 > (g - \bar{g}_{background})^2$$

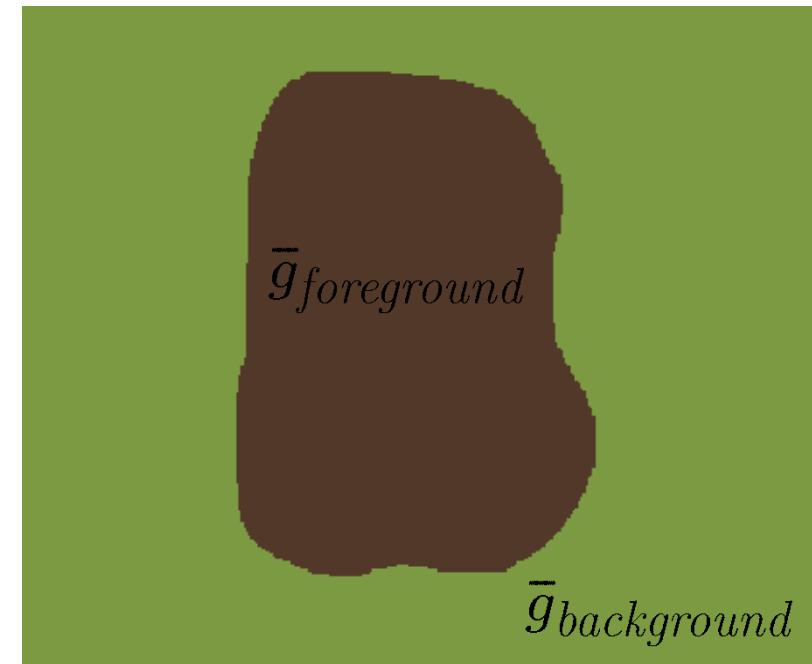
→ shrink contour



# Image Segmentation with Level Sets cont.

- check for pixels on boundary with grey (color) value  $I$ 
  - pixel more similar to area inside
$$(g - \bar{g}_{foreground})^2 < (g - \bar{g}_{background})^2$$

→ expand contour



# Image Segmentation with Level Sets cont.

- Mumford-Shah based segmentation:

$$\frac{\partial \vec{x}}{\partial t} = \frac{\nabla \phi}{\|\nabla \phi\|} \cdot \left( -\beta \kappa - \alpha - \lambda_1(g - \bar{g}_{foreground})^2 + \lambda_2(g - \bar{g}_{background})^2 \right)$$

contour  
rectification

overall preference  
for shrinking/expanding

image  
segmentation

- $\alpha, \beta, \lambda_1, \lambda_2$  can be used to tune approach

# Image Segmentation with Level Sets cont.

- Example:



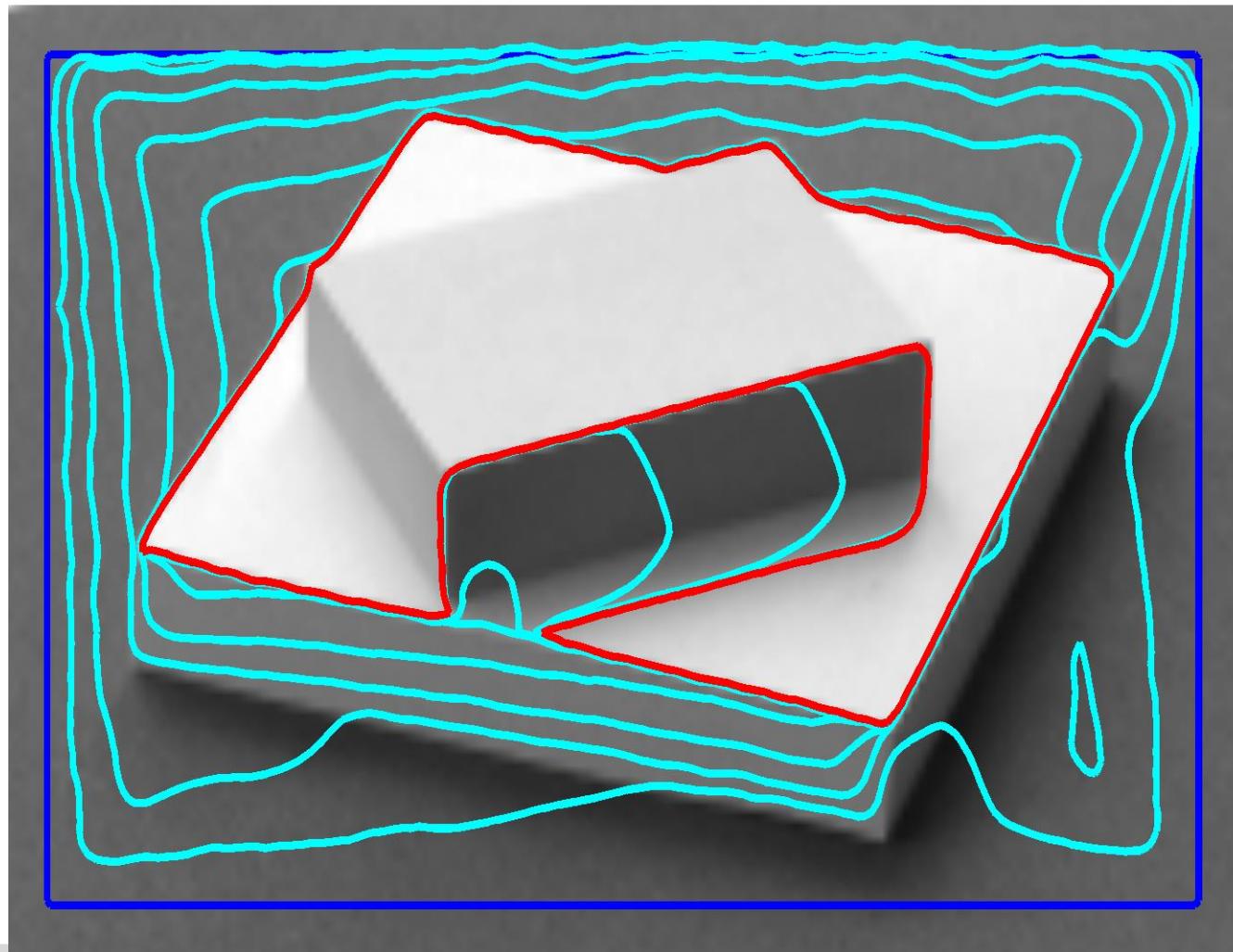
# Image Segmentation with Level Sets cont.

- Example:



# Image Segmentation with Level Sets cont.

- Example:



# RANDOM FIELDS

# Random Fields

- every pixel belongs to one segment. But to which one?



- the segment label of each pixel is seen as a variable
- the feature vector of a pixel is related to its label



- feature vectors of pixels are also seen as variables, however, its value is observed
- the relationship is modeled by **potential functions**

$$\phi_f(l(u, v), f(u, v)) \begin{cases} \text{is small} & \text{if } f(u, v) \text{ supports label } l(u, v) \\ \text{is large} & \text{if } f(u, v) \text{ does not support label } l(u, v) \end{cases}$$

# Random Fields

- labels of neighboring pixels are also related

$$l(u, v) \rightsquigarrow l(u + 1, v)$$

$$l(u, v) \rightsquigarrow l(u, v + 1)$$

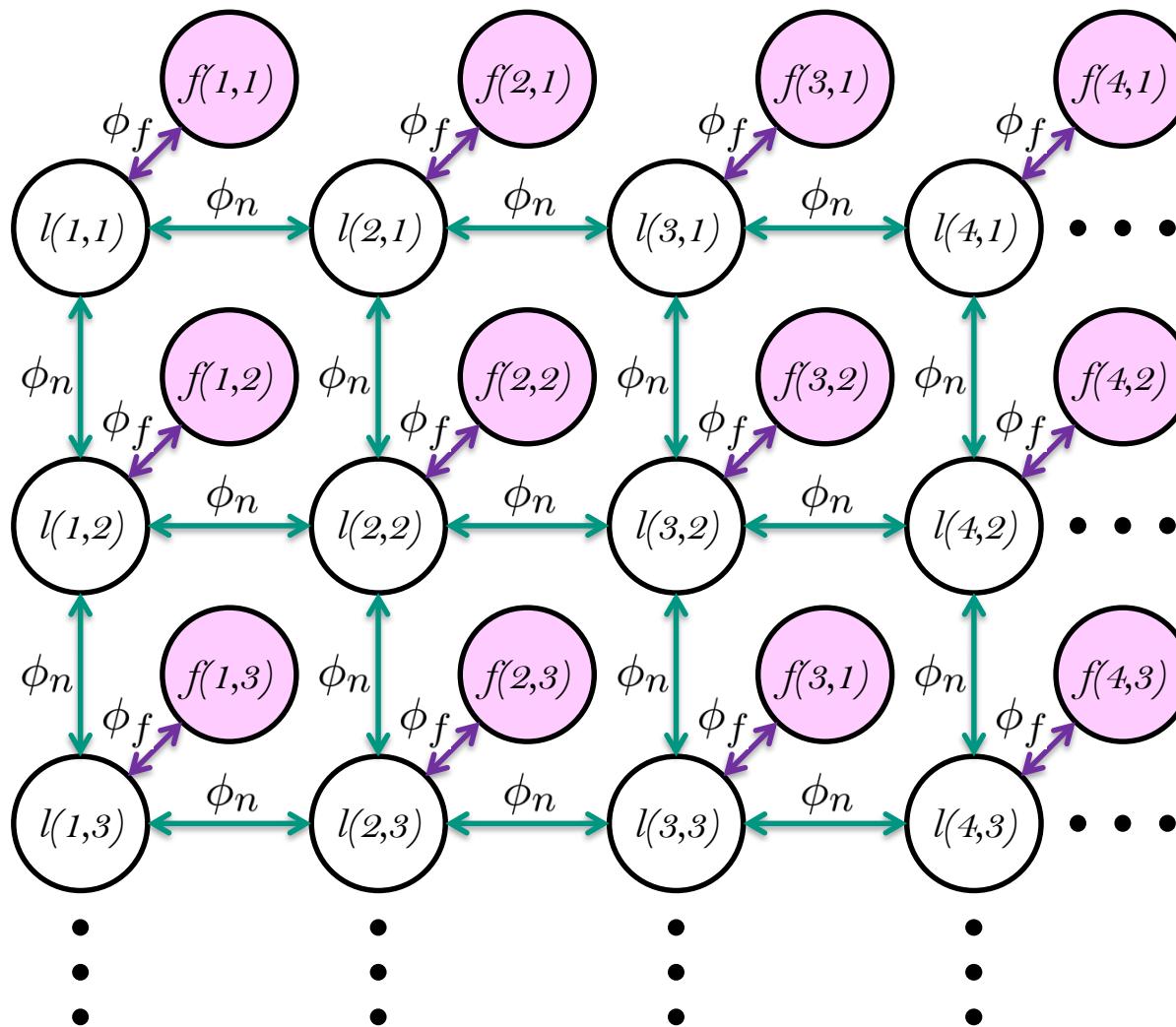
- the relationship is again modeled by potential functions

$$\phi_n(l(u, v), l(u + 1, v))$$

$$\phi_n(l(u, v), l(u, v + 1))$$

$$\phi_n(l(u, v), l(u + 1, v)) \left\{ \begin{array}{ll} \text{is small} & \text{if } l(u, v) \text{ and } l(u + 1, v) \text{ are similar} \\ \text{is large} & \text{if } l(u, v) \text{ and } l(u + 1, v) \text{ are dissimilar} \end{array} \right.$$

# Random Fields



# Random Fields

- Goal:

- find labels  $l(u, v)$  such that the potential functions are minimized

$$\begin{aligned} \underset{l(\cdot, \cdot)}{\text{minimize}} \quad & \alpha_f \cdot \sum_{u, v} \phi_f(l(u, v), f(u, v)) \\ & + \alpha_n \cdot \sum_{u, v} \phi_n(l(u, v), l(u + 1, v)) \\ & + \alpha_n \cdot \sum_{u, v} \phi_n(l(u, v), l(u, v + 1)) \end{aligned}$$

- with weighting factors  $\alpha_f, \alpha_n > 0$
  - solution of optimization problem
    - exact → hard (in general, exceptions exist)
    - approximative

# Random Fields

- Example:
  - extract bright foreground object from dark background

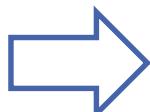
$l = 0$  background

$l = 1$  foreground

$f$  gray value  $0 \leq f \leq 255$

$$\phi_f(l, f) = \left(l - \frac{1}{255}f\right)^2$$

$$\phi_n(l, l') = (l - l')^2$$



- implements segmentation criteria:
- predefined color criterion
  - spatial criterion

# Random Fields

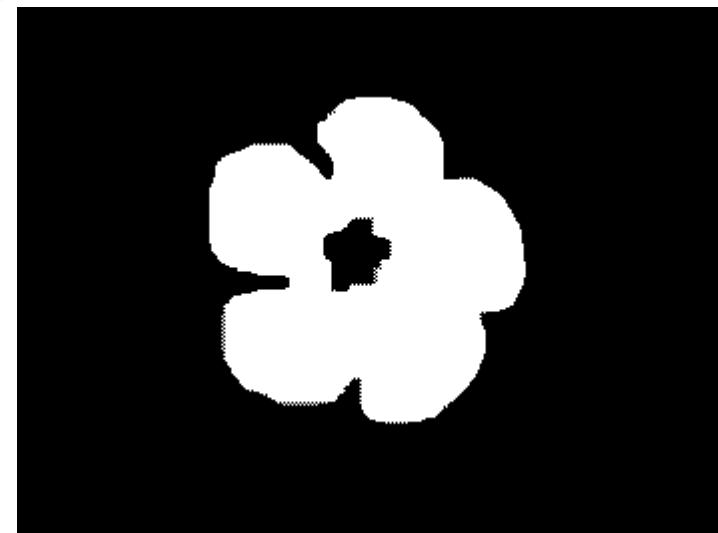
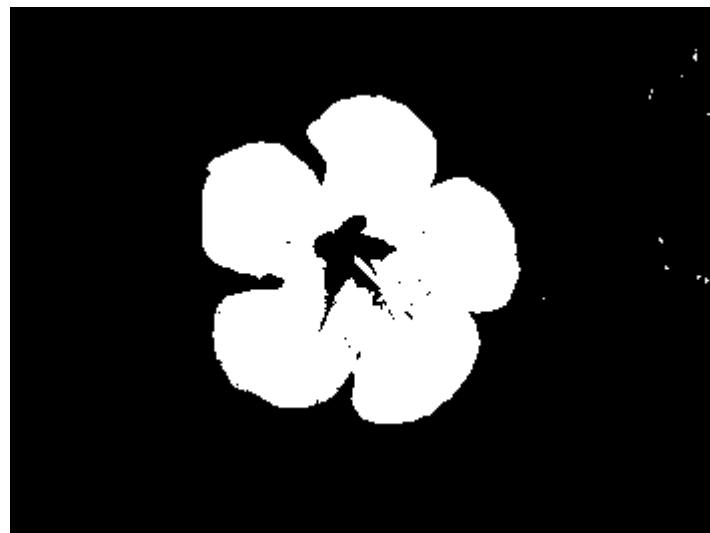
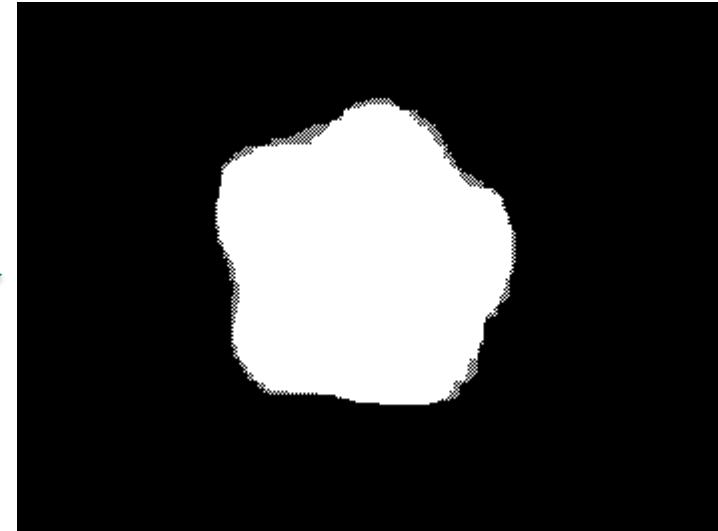


features

$$\alpha_f \gg \alpha_n$$

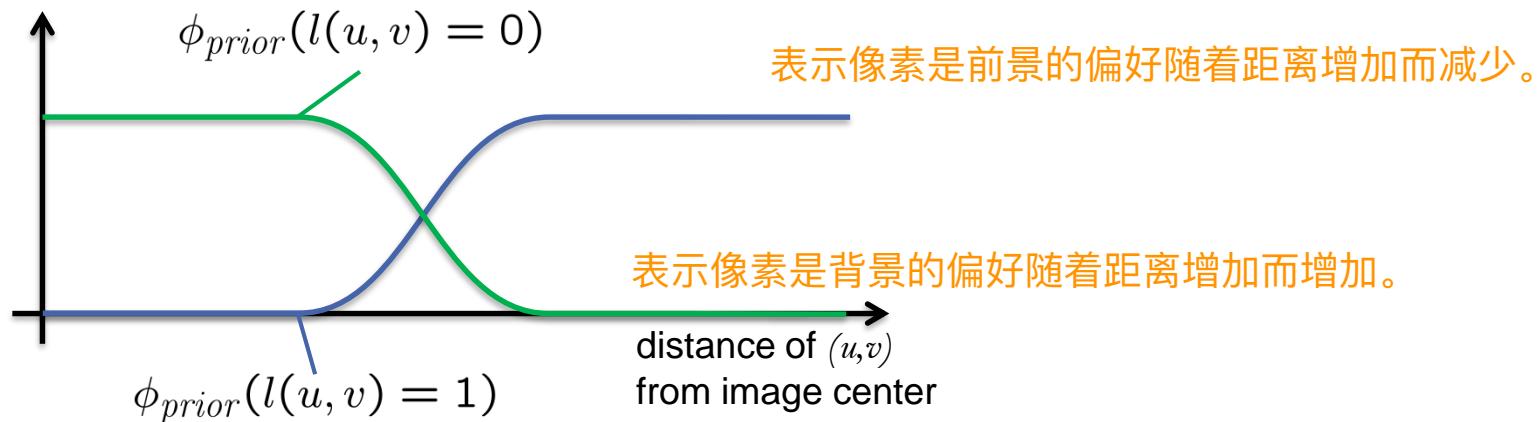
$$\alpha_f \ll \alpha_n$$

$$\alpha_f \approx \alpha_n$$



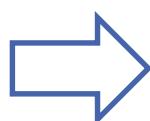
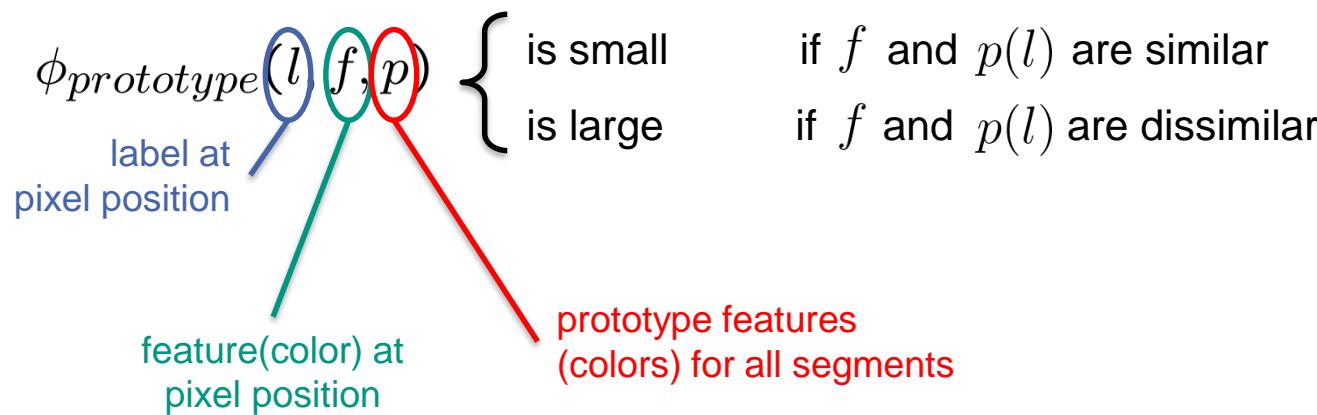
# Random Fields

- Advantage of random field modeling:
  - segmentation problem is formulated as optimization problem
  - potential functions allow to model many segmentation criteria, e.g.
    - seed points  
keep label function constant for seed points
    - general preferences for certain segment labels (a prior)  
→ add unary potential function  $\phi_{prior}(l)$   
e.g. to specify that the foreground object is expected to be in the center of the image



# Random Fields

- prototype segment feature vector. Pixels should be assigned to segment with most similar prototype feature
  - add prototype variables to random field, one for each segment
  - add potential functions that model similarity of prototype feature and pixel feature  $f$



implements homogeneity criterion

# Random Fields

- Example:

- subdivide foreground and background assuming that
  - foreground object is located in the center of the image
  - foreground object and background object have distinctive colors
  - uses pixel colors (e.g. in RGB) as features

对于前景 ( $l=1$ ) :  $\phi_{prior}$  值随着像素远离中心而增加。

$$\phi_{prior}(l(u, v)) = \begin{cases} \max \left\{ \frac{|2u - width|}{width}, \frac{|2v - height|}{height} \right\} & \text{if } l(u, v) = 1 \\ 1 - \max \left\{ \frac{|2u - width|}{width}, \frac{|2v - height|}{height} \right\} & \text{if } l(u, v) = 0 \end{cases}$$

$$\phi_{prototype}(l, f, p) = \|f - p(l)\|^2$$

衡量像素特征<sub>f</sub> (颜色) 与其  
对应分割区域的原型特征<sub>p(l)</sub> 之间的相似性。

$$\phi_n(l, l') = (l - l')^2$$

确保相邻像素的标签一致性，鼓励同一区域内的像素分配相同的标签。

# Random Fields

$$\phi_{prior}(l(u, v)) = \begin{cases} \max \left\{ \frac{|2u - width|}{width}, \frac{|2v - height|}{height} \right\} & \text{if } l(u, v) = 1 \\ 1 - \max \left\{ \frac{|2u - width|}{width}, \frac{|2v - height|}{height} \right\} & \text{if } l(u, v) = 0 \end{cases}$$

$$\phi_{prototype}(l, f, p) = \|f - p(l)\|^2$$

$$\phi_n(l, l') = (l - l')^2$$

加权参数，用于控制不同潜在函数对总能量的贡献。

$$\underset{l(\cdot, \cdot), p(\cdot)}{\text{minimize}} \quad \alpha_{prior} \cdot \sum_{u, v} \phi_{prior}(l(u, v))$$

增大  $\alpha_{prior}$  会更加强调前景靠近图像中心的假设。

$$+ \alpha_f \cdot \sum_{u, v} \phi_{prototype}(l(u, v), f(u, v), p)$$

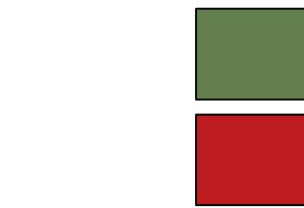
增大  $\alpha_f$  会更倾向于依据颜色特征进行标签分配。

$$+ \alpha_n \cdot \sum_{u, v} \phi_n(l(u, v), l(u + 1, v))$$

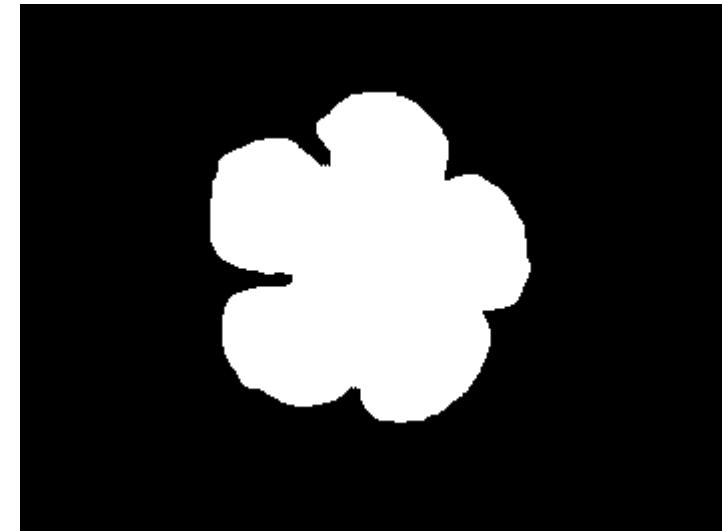
$$+ \alpha_n \cdot \sum_{u, v} \phi_n(l(u, v), l(u, v + 1))$$

增大  $\alpha_n$  会更加注重分割区域的连贯性。

# Random Fields



$\alpha_{prior}$  large



$\alpha_{prior}$  small

background color

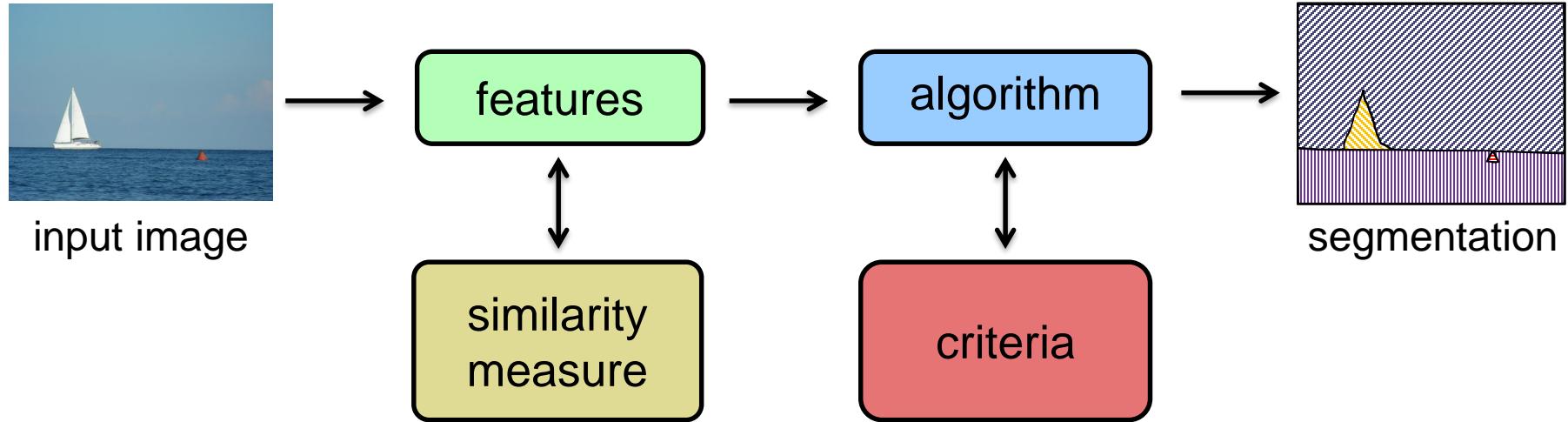


foreground color



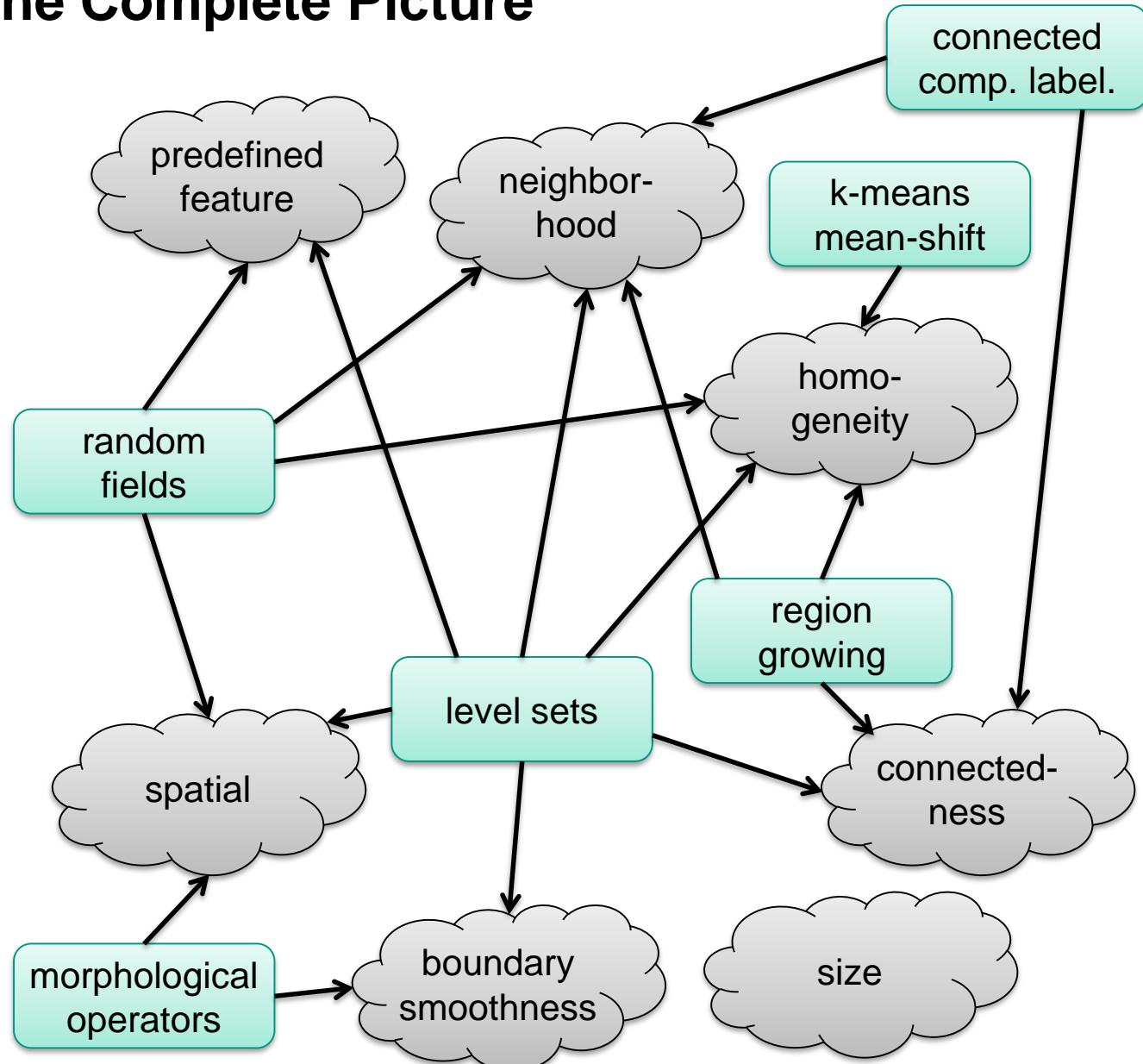
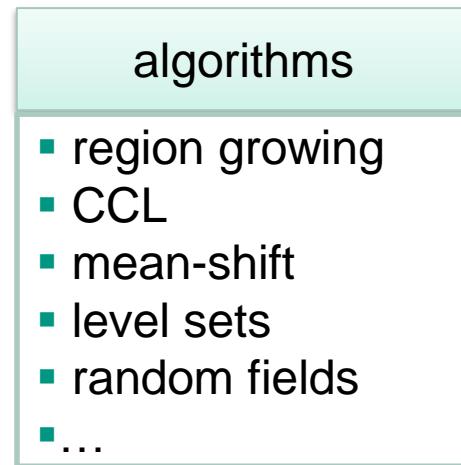
# **SUMMARY: SEGMENTATION**

# Segmentation: the Complete Picture



features	similarity measure	criteria	algorithms
<ul style="list-style-type: none"><li>▪ color</li><li>▪ texture features</li><li>▪ depth</li><li>▪ motion</li><li>▪ ...</li></ul>	<ul style="list-style-type: none"><li>▪ Euclidean dist.</li><li>▪ other metric</li><li>▪ ...</li></ul>	<ul style="list-style-type: none"><li>▪ predefined values</li><li>▪ neighborhood</li><li>▪ homogeneity</li><li>▪ spatial</li><li>▪ size</li><li>▪ ...</li></ul>	<ul style="list-style-type: none"><li>▪ region growing</li><li>▪ CCL</li><li>▪ mean-shift</li><li>▪ level sets</li><li>▪ random fields</li><li>▪ ...</li></ul>

# Segmentation: the Complete Picture



# Segmentation: the Complete Picture

features
▪ color
▪ texture features
▪ depth
▪ motion
▪ ...

similarity measure
▪ Euclidean dist.
▪ other metric
▪ ...

- which features are **salient** and **discriminative**?
  - color
  - texture
  - depth
  - motion
  - ...
- which **representation** is appropriate?
  - color space
  - various texture features
  - histograms
  - ...
- how can we **compare** feature vectors?
  - similarity measures

