CITY
UNIVERSITY OF LONDON
EST 1894

Computer Vision
INM460/IN3060

Lecture 4
Image segmentation

Dr Giacomo Tarroni
Slides credits: Giacomo Tarroni, Sepehr Jalali

Recap from the previous lecture

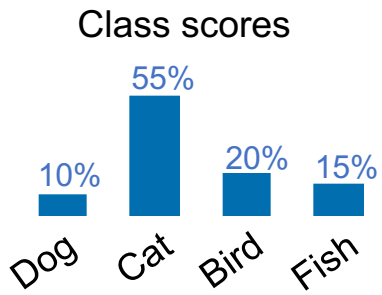
- Image filtering
- Linear filtering:
  - Convolution vs cross-correlation
  - Common filters:
  - Moving average
  - Gaussian
  - Sharpening
- Non-linear filtering: median filter
- Edge detection
  - Gradient-based
  - Laplacian-based
  - Non-maximum suppression
- Common CV tasks

Common CV tasks

Image classification

- Identify the class of the object shown in the image
- Often (but not necessarily) based on class scores estimation

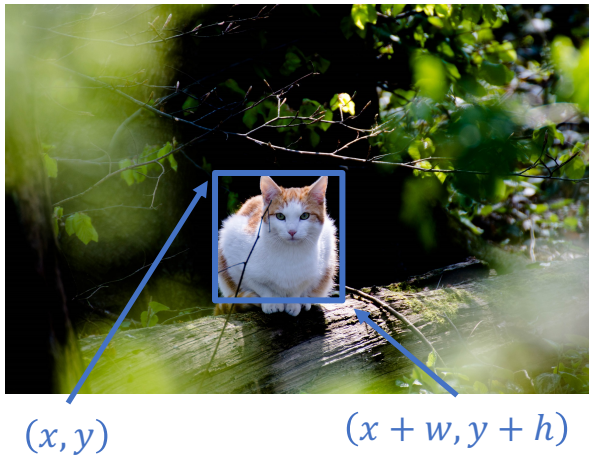
Class label: “Cat”



Common CV tasks

Image classification & localization

- Identify the class and the location of the object (regression of bounding box coordinates)
- Assume only one object per image



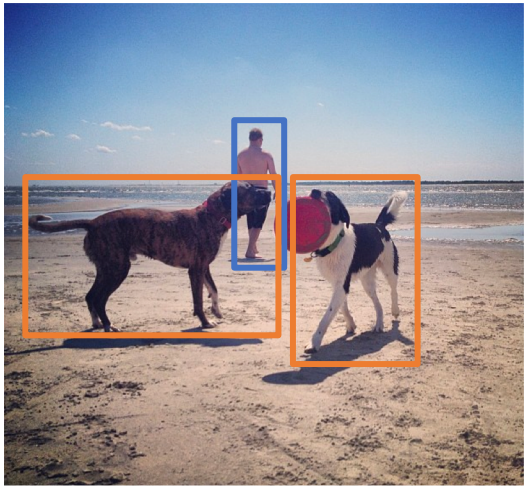
Class. & local.

- Class label: "Cat"
- Bounding box coordinates: (x, y, w, h)

Common CV tasks

Object detection

- Identify the class and the location of the objects in the image
- Assume **more than one object** per image



Object detector

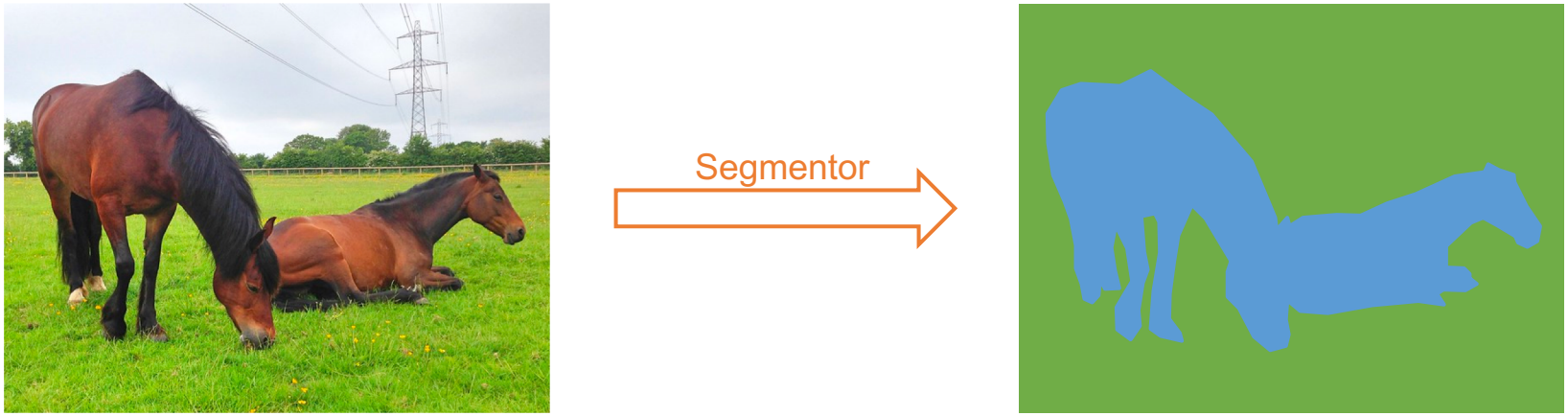
For each object in the image:

- Class label: "Dog"
- Bounding box coordinates: (x, y, w, h)

Common CV tasks

Semantic segmentation

- Assign a label to each pixel

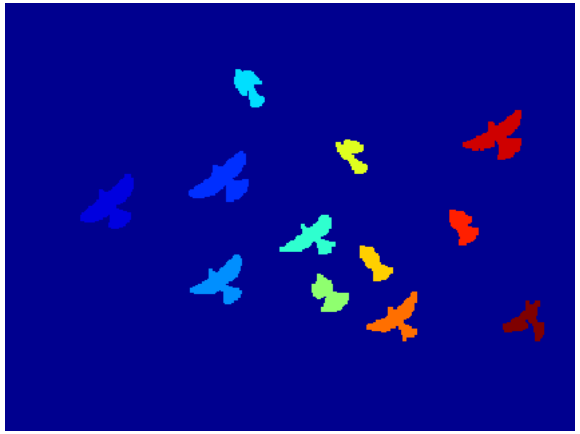


Common CV tasks

Instance segmentation

- Assign a label to each pixel
- **Distinguish** between different **instances**
- Combination of object detection and semantic segmentation

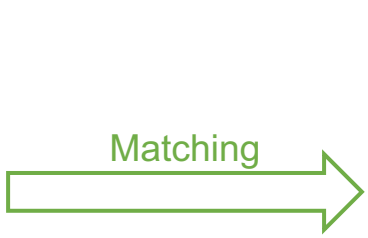
Instance segm.



Common CV tasks

Image matching/alignment/registration

- Identify the (rigid? affine? non-rigid?) transformation which aligns two or more images
- Used for multi-modal comparison, image stitching, etc.



M. Brown. Automatic panoramic image stitching using invariant features, IJCV 2007

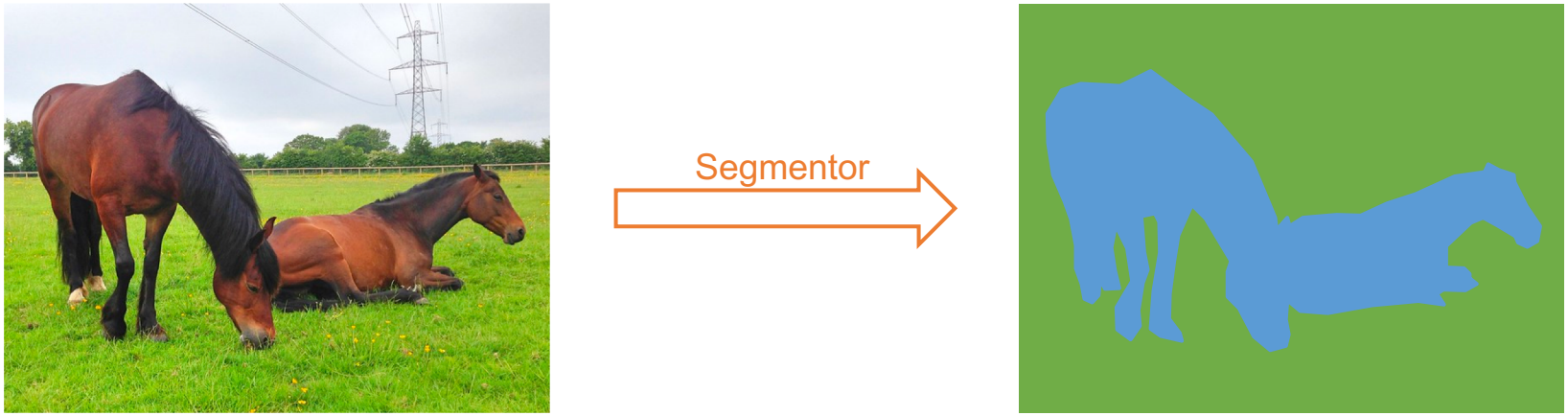
Overview of today's lecture

Image segmentation:

- Intensity-based methods
  - Thresholding
  - K-means clustering
- Watershed
- Active contours
  - Snakes
  - Level-sets
- Graph cuts
- Superpixels
- Quantifying results: how to tell if we achieved a good segmentation

Semantic segmentation

- Image segmentation (also semantic segmentation) is the process of partitioning a digital image into multiple segments to distinguish different objects or structures
- It does not distinguish between different instances



- Assign a label to **each pixel**
- One label per **class** (e.g. 0 is “background”, 1 is “horse”)

Applications

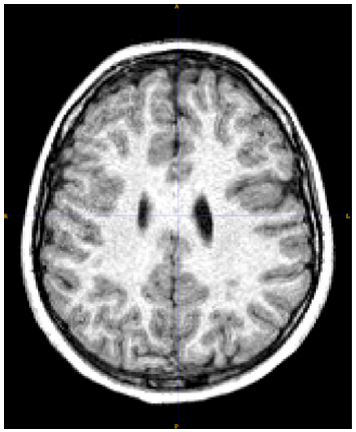
Image segmentation allows to simplify the representation of an image into something easier to analyse. Some example of applications:

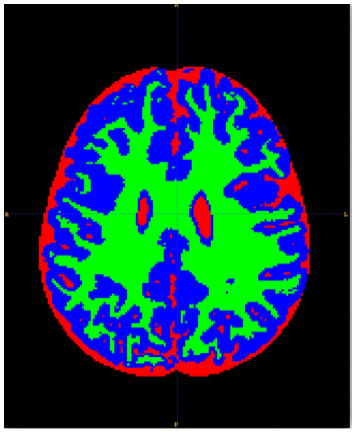
- Image composition

Credits: https://en.wikipedia.org/wiki/Chroma_key

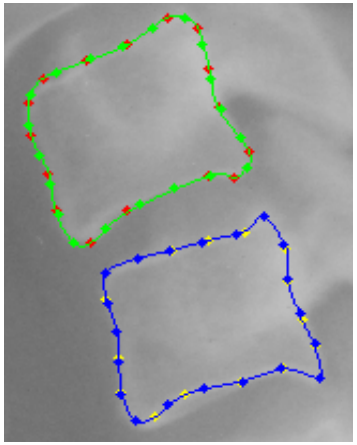
Applications

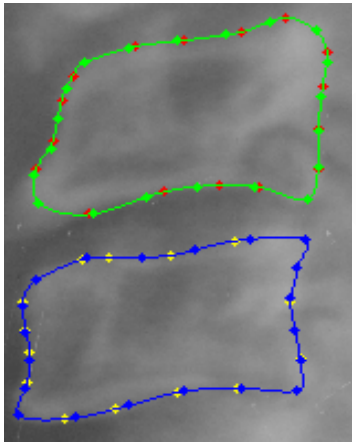
- Measurements from images

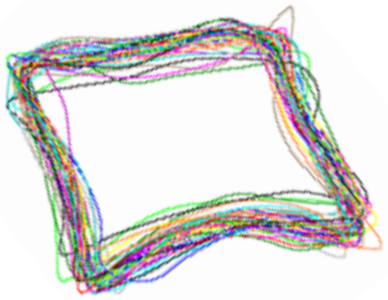




Extraction of volume/surface metrics for different tissue types from brain MRI



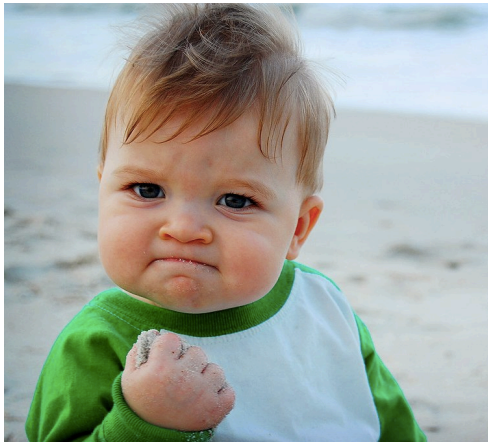




Shape analysis for vertebrae

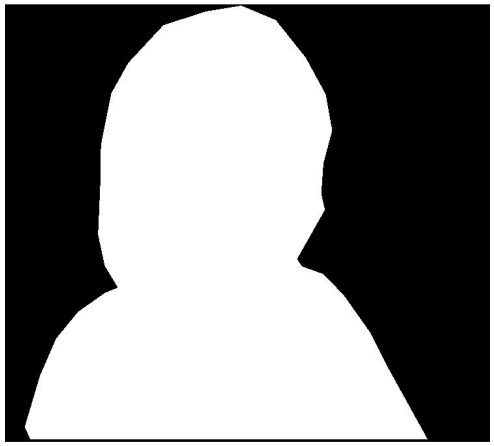
Segmentation vs edge detection

- Edge detection:
  - It aims at identifying edges in an image
  - The output is an **edge map** (potentially with partial/open contours)
  - Usually there is **no specification of objects of interest**
- Image segmentation:
  - It aims at identifying **specific objects/structures** in an image
  - The output is a **segmentation mask** (each pixel as a label)
  - A segmentation mask can be transformed to show the object's contours, while generating a segmentation mask from an edge map is usually non trivial



Original image

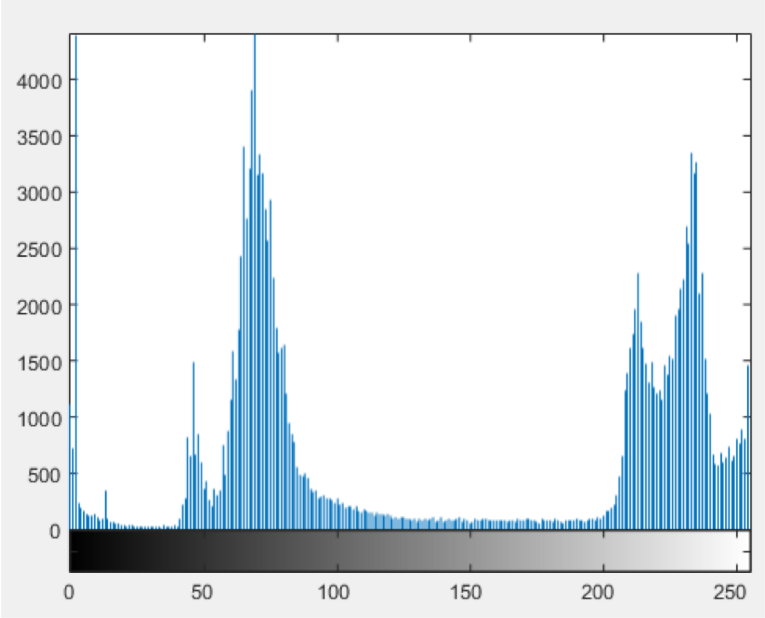
Edge map



Segmentation mask

Intensity-based methods

- The easiest approaches to image segmentation
- They make use of **pixel intensity or colour**: they consider only information included in the histogram
- **No spatial information** is considered
- Examples:
  - Thresholding
  - K-means clustering



Thresholding

- Thresholding classifies each pixel of the image into either background (0 in the produced segmentation mask B) or foreground (1) based on its **intensity being above** (or below) **a given threshold**
- Built-in Matlab function: `imbinarize`, `imquantize`

$$B(x,y) = \begin{cases} 1, & I(x,y) < T \\ 0, & \text{otherwise} \end{cases}$$

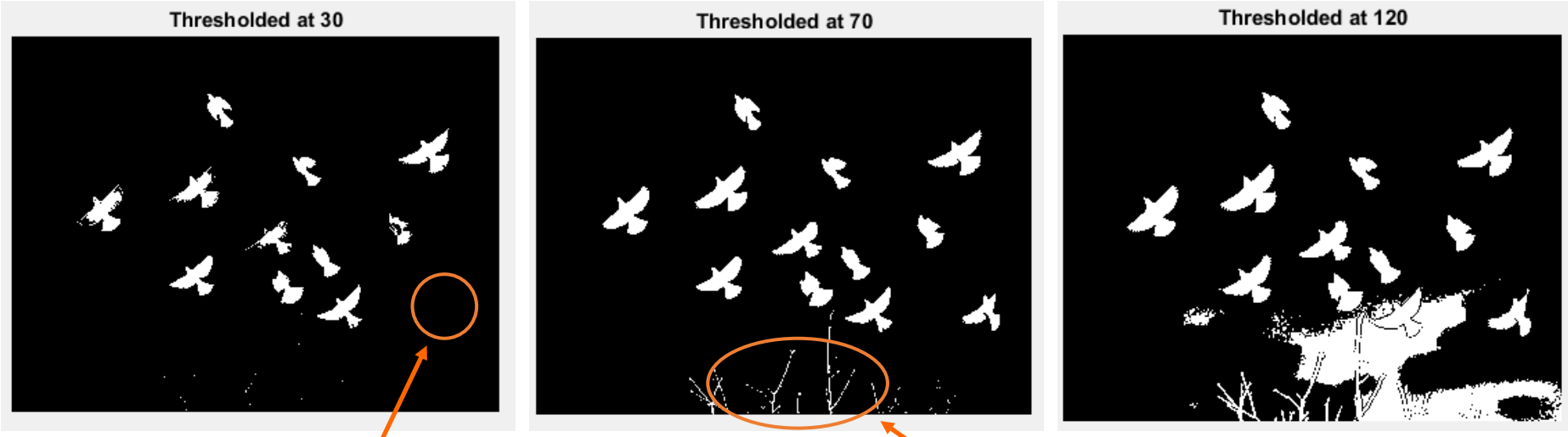
Original image I

Segmentation mask B

```
I = rgb2gray(imread('birds.png'));
B = I < 30; imshow(B);
```

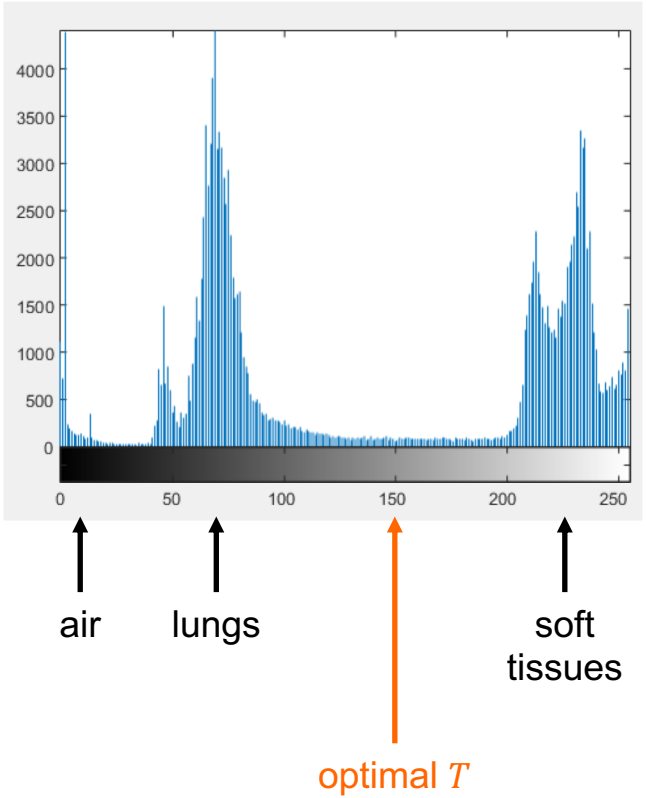
Thresholding

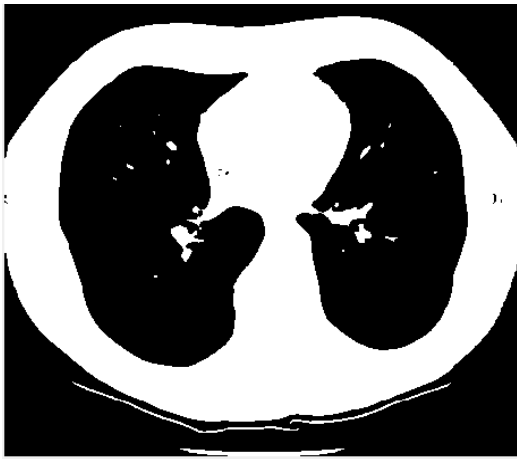
- The choice of the threshold can have a dramatic effect on the result
- It is usually unlikely that a single threshold will correctly separate foreground (object of interest) from background. Some limiting factors:
  - Variable lighting through the scene
  - Similar colours between foreground and background
- A further refinement process is often needed (see morphological operators)



Thresholding

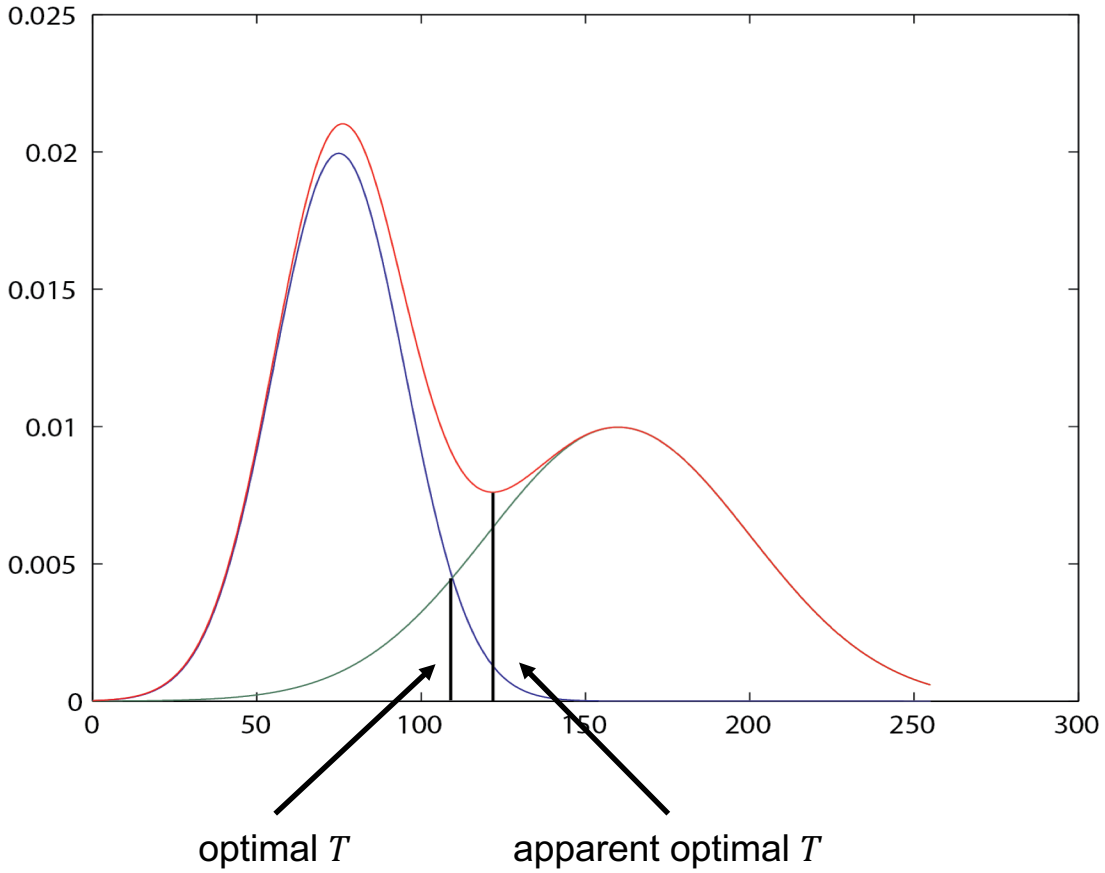
In many cases, the choice for an optimal threshold requires a careful study of the histogram as well as previous knowledge about the image properties





Automatic thresholding

Automatic thresholding methods try to automatically identify an optimal thresholding simply analysing the image histogram



Otsu's thresholding

- Otsu [1] proposed a method for automatic thresholding
- His method consists in **moving a threshold** along the normalized histogram (pdf of I) and in computing the **intra-class variance** σ^2_{intra} at each possible step (**exhaustive search**). The threshold T^* yielding the **lowest** σ^2_{intra} will be selected as optimal:

$$T^* = \underset{T}{\operatorname{argmin}}\, \sigma^2_{intra}(T) \qquad \sigma^2_{intra}(T) = w_f(T)\sigma^2_f(T) + w_b(T)\sigma^2_b(T)$$

with foreground/background variances:

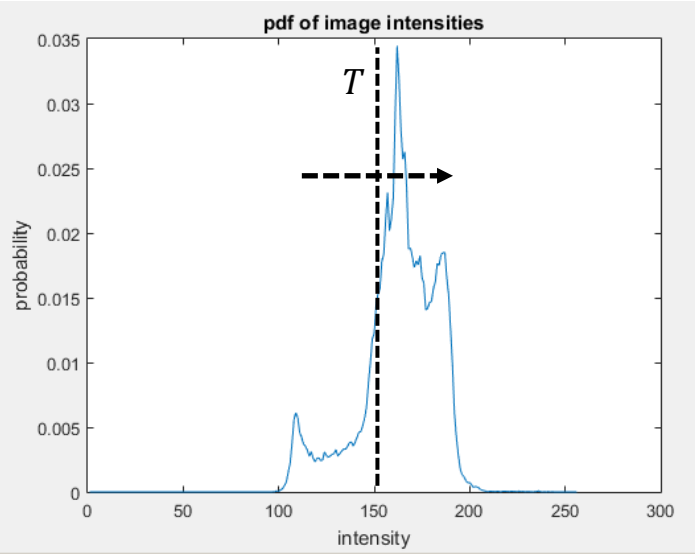
$$\sigma^2_f(T) = \frac{1}{N_f}\sum_{i=1}^{N_f}\left(I_i - \mu_f(T)\right)^2$$

$$\sigma^2_b(T) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(I_i - \mu_b(T)\right)^2$$

Mean int. values

with N_f and N_b num. of pixels in foreground and background respectively, and weights:

$$w_f(T) = N_f/N_{tot} \qquad w_b(T) = N_b/N_{tot}$$

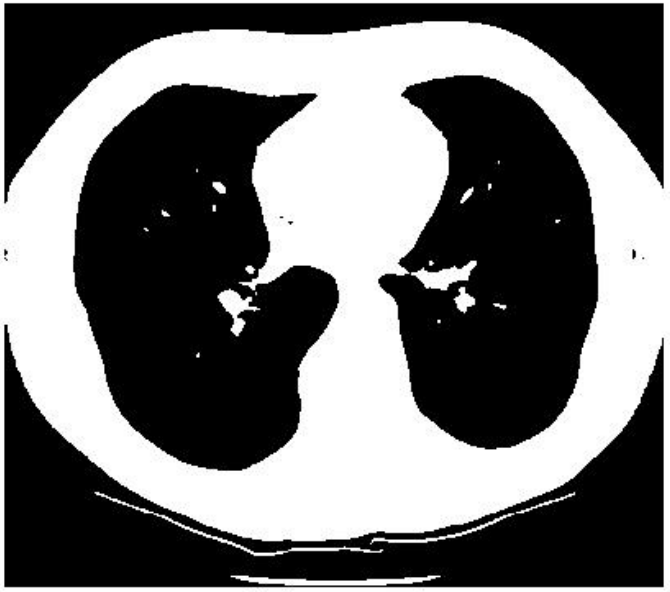


[1] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. TSMC 1979

Otsu's thresholding

- Otsu's method can be extended to **multiple thresholds**
- Built-in Matlab functions: `imbinarize`, `graythresh`, `otsuthresh`, `multitresh`
- For more insights on the functioning of the algorithm: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

```
I = rgb2gray(imread('lung.png'));
T = graythresh(I)*255;
figure, imshow(I>T);
```



Otsu's thresholding
(T = 130)

K-means clustering

- K-means is an **unsupervised clustering algorithm** that can be used for a variety of applications, including image segmentation
- K-means attempts at dividing the data space into clusters. Each cluster is represented by its centre (i.e. its mean), and each data point is associated to the nearest cluster centre based on some distance metric
- The optimal cluster centres are those minimising the intra-class variance:

$$\sigma^2_{intra}(T) = w_f(T)\sigma^2_f(T) + w_b(T)\sigma^2_b(T)$$

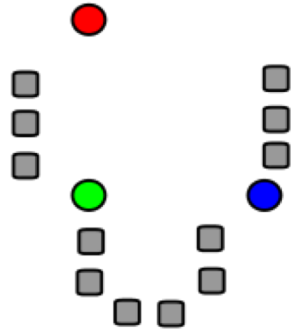
Generalized to K clusters (for single threshold, $K = 2$)

$$= \frac{N_f}{N_{tot}}\frac{1}{N_f}\sum_{i=1}^{N_f}\left(I_i - \mu_f(T)\right)^2 + \frac{N_b}{N_{tot}}\frac{1}{N_b}\sum_{i=1}^{N_b}\left(I_i - \mu_b(T)\right)^2 = \frac{1}{N_{tot}}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\left(I_i - \mu_k(T)\right)^2$$

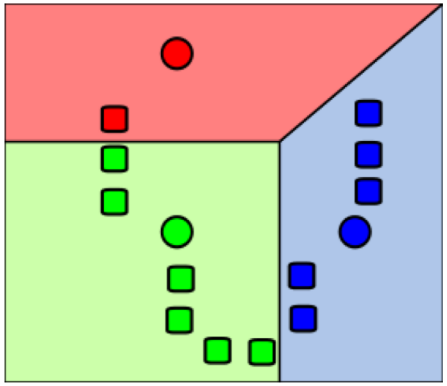
$$T^* = \underset{T}{\operatorname{argmin}}\,\sigma^2_{intra}(T) = \underset{T}{\operatorname{argmin}}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\left(I_i - \mu_k(T)\right)^2$$

K-means clustering

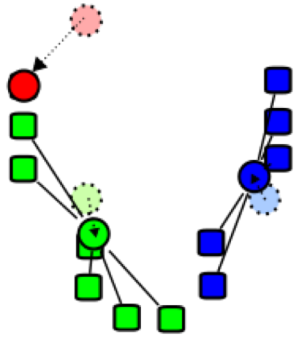
- It minimizes the same function as Otsu's, but is implemented as **iterative refinement** instead of exhaustive search
- It can be used to detect even $K > 2$ clusters (i.e. multiple thresholds)
- It can be extended to more than one input dimension (e.g. colour channels
- Basic algorithm:
  1. Initiate K cluster centres at random positions of the data space
  2. Assign each data point to a cluster based on distance from centres
  3. Update the cluster centres by computing the new means
  4. Repeat steps 2-3 until convergence (centres don't move anymore)



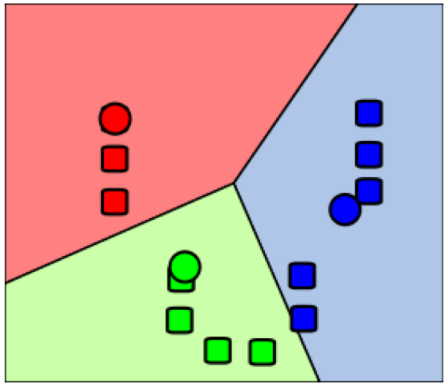
1. Cluster centres initialization



2. Clustering



3. Cluster centres update



2. Clustering

Credits: https://en.wikipedia.org/wiki/K-means_clustering

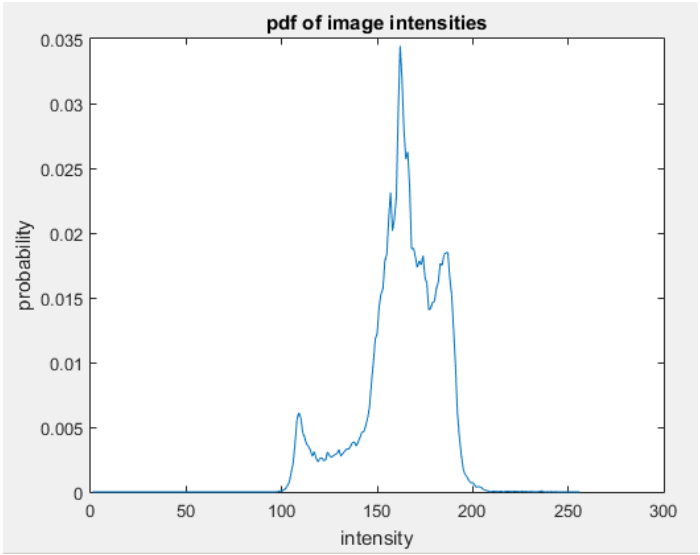
K-means clustering

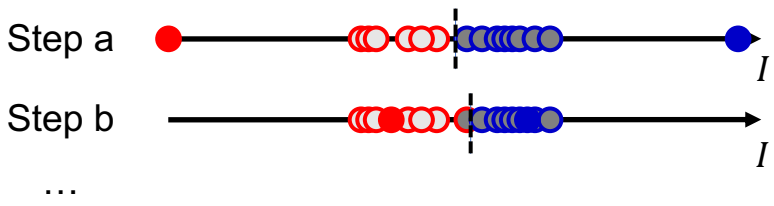
Let's apply K-means ($K = 2$) to a 1D data space: image intensity.
Possible algorithm:

1. Define $\mu_b = 0$ and $\mu_f = 255$
2. Assign each pixel based on distance from its I_i to that of the two means
3. Recalculate the new μ_b and μ_f
4. Repeat steps 2-3 until convergence (i.e. μ_b and μ_f don't change anymore). Then

$$T^* = \frac{(\mu_b + \mu_f)}{2}$$

```
T = 127.5; lastT = 0;
while (abs(T - lastT) > 1)
    lastT = T;
    mf = mean(I(I>T));
    mb = mean(I(I<=T));
    T = 0.5*(mb+mf);
end
```





Built-in Matlab functions: `kmeans`, `imsegkmeans`

K-means clustering

Let's apply K-means ($K = 4$) to a 3D data space: colour channels.
Possible algorithm:

```
% Read image and resize for speed
I = imresize(imread('road_sky.jpg'), 0.25);
imshow(I);
[rows, cols, planes] = size(I);

% Create a matrix R of size N*3 with the
% RGB data. N is the number of pixels in I
R = double(reshape(I, rows*cols, 3));

% Run K-means (K = 4)
[clusterID, clusterCentre] = kmeans(R, 4);

% Reshape back to an image and display
clusterID = reshape(clusterID, rows, cols);
figure; imagesc(clusterID);
```

Original image

K-means clustering

K-means clustering

Limitations:

- K needs to be decided in advance

$K = 2$

$K = 3$

$K = 4$

$K = 5$

K-means clustering

Limitations:

- Due to the random initialization, K-means can generate different results each time (i.e. it gets stuck into local minima)

$K = 4$

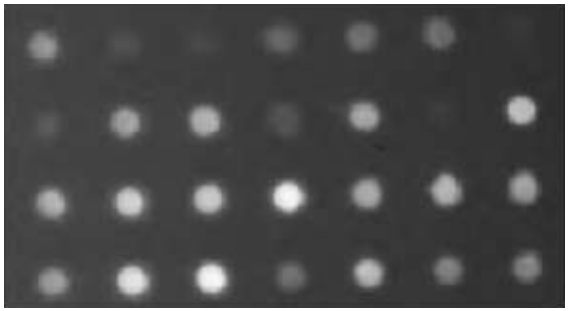
$K = 4$

To have more chances of reaching global optimal solution, run the algorithm more times and average results:

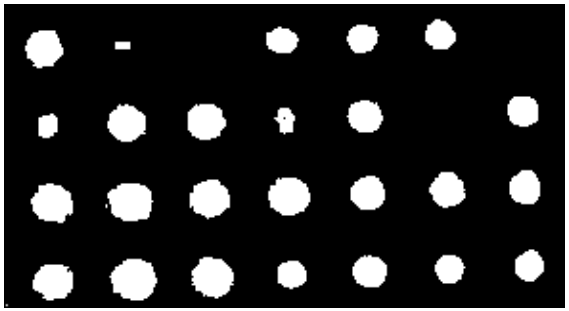
```
[clusterID, clusterCentre] = kmeans(R, 4, 'Replicates', 3);
```

Hysteresis thresholding

- Hysteresis is the lagging of an effect, an inertia
- In thresholding, it means that we recognize objects contours beyond a first (lower) threshold, but we consider them actual objects only if they remain visible beyond a second (higher) threshold
- This can be seen as (backward) “continuations” of high-confidence areas
- Example: segmentation of the brighter blobs

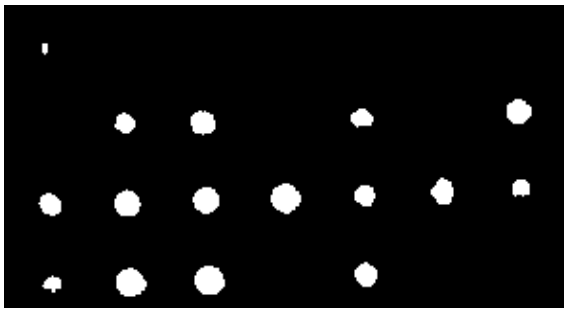


I



$I > T_{low}$

Shapes are (mostly) correct but darker blobs included



$I > T_{high}$

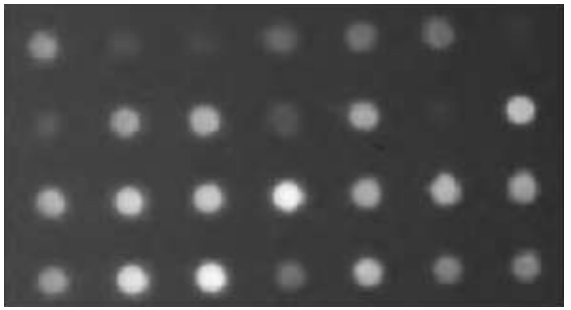
Brighter blobs are identified but shapes are not correct

- Hysteresis thresholding will select brighter blobs using T_{high}, and for those blobs use the shape from T_{low}

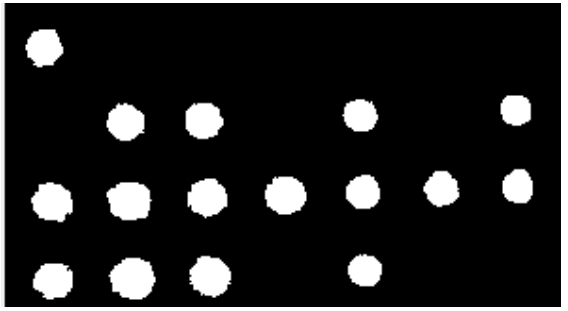
Hysteresis thresholding

- The following code labels the regions detected using the low thresholded image (L) and uses the high thresholded image (H) to select valid regions
- The pixels from L that are part of a valid region (i.e. are connected to H) are retained

```
I = imread('dots.png');
I = double(rgb2gray(I));

% Compute thresholded images (L & H)
Thigh = 150;
Tlow = 80;
H = I > Thigh;
L = I > Tlow;

% Label connected components of L
labels = bwlabel(L);
% Identify labels persisting in H
validLabels = unique(labels(H));
J = ismember(labels, validLabels);

figure; imshow(J);
title('Hysteresis tresholding result');
```



Original image



Hysteresis thresholding

Distance transform

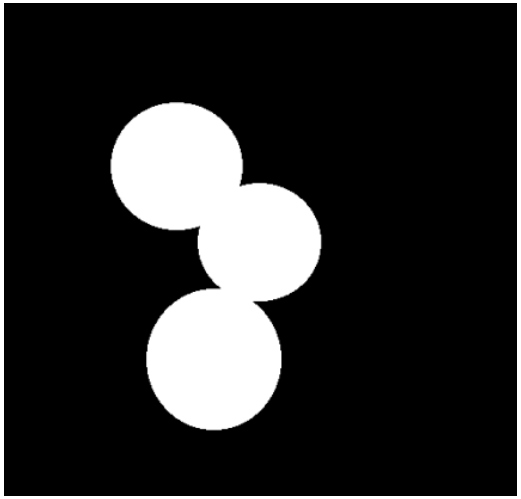
- Image intensity (or colour) is a powerful feature. However, there is a lot of additional contextual information which is **encoded spatially**
- One way to make use of it is through the distance transform: when applied to a binary image, the distance transform returns for each pixel **the distance** (usually Euclidean) **from the closest non-zero pixel**
- Built-in Matlab function: `bwdist`

```
B =
     0     0     0     0     0
     0     1     0     0     0
     0     0     0     0     0
     0     0     0     1     0
     0     0     0     0     0
```

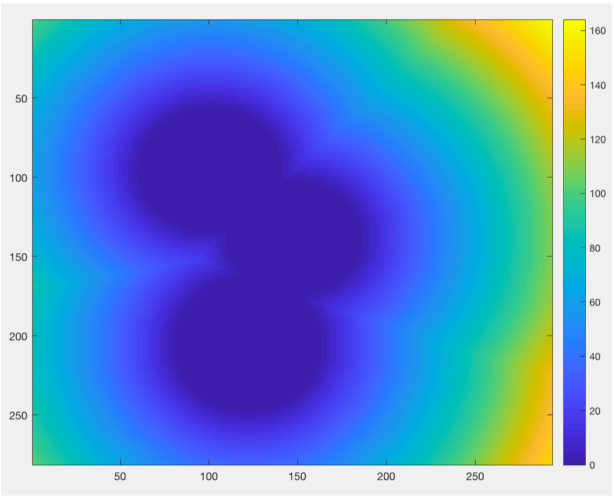
```
D = bwdist(B)

D =
    1.4142    1.0000    1.4142    2.2361    3.1623
    1.0000         0    1.0000    2.0000    2.2361
    1.4142    1.0000    1.4142    1.0000    1.4142
    2.2361    2.0000    1.0000         0    1.0000
    3.1623    2.2361    1.4142    1.0000    1.4142
```

Distance transform

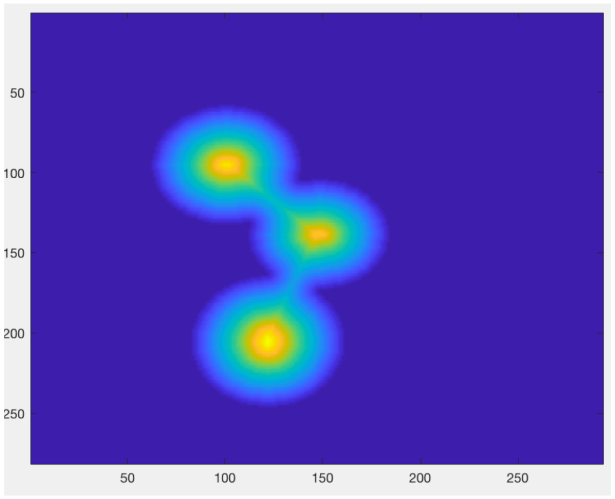
- Let's assume we want to separate three overlapping objects in a binary map
- A visual example of distance maps:



B



bwdist(B)

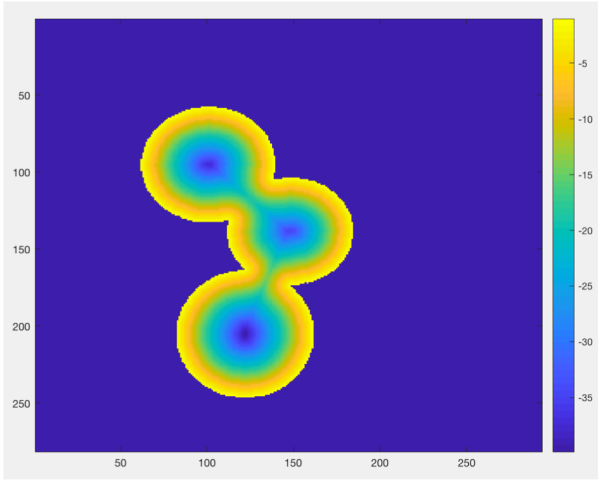


bwdist(~B)

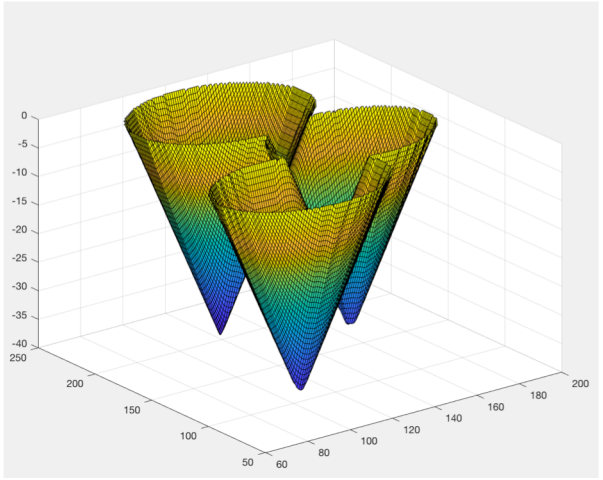
Watershed segmentation

- The watershed algorithm treats the distance map like a topographic map (i.e. the value of each point represents its height) and **finds the lines that run along the tops of ridges**
- Flooding visualization: imagine of placing a water source in **each local minimum** in the topographic map to flood the entire relief from sources, and build barriers when different water sources meet
- The resulting set of barriers constitutes the watershed segmentation

```
D = -bwdist(~B);
D(~B) = -Inf;
```



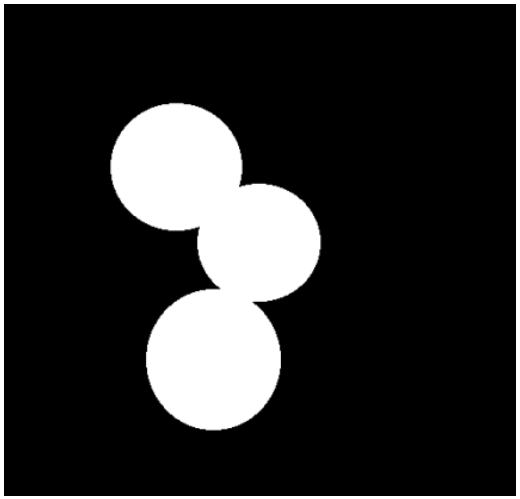
`imagesc(D);`



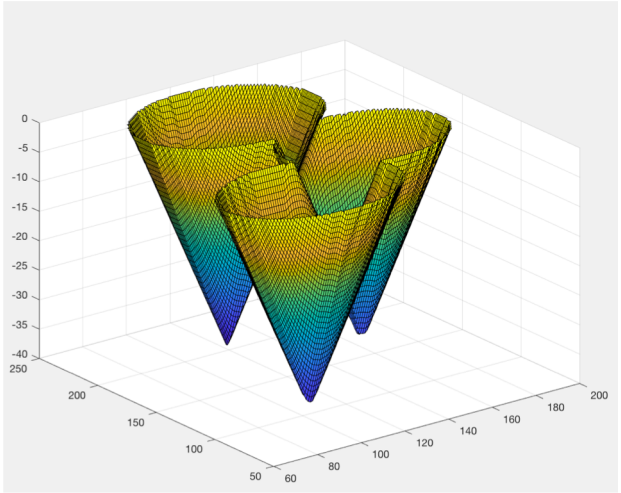
`surf(D);`

Watershed segmentation

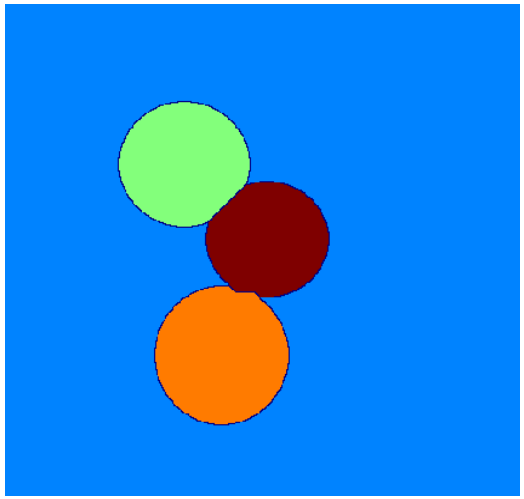
- Watersheds are usually extracted from distance maps of a binary edge maps
- It is capable of detecting separate objects despite them being connected
- Built-in Matlab function: `watershed`



Original image



3D visualization
of distance map

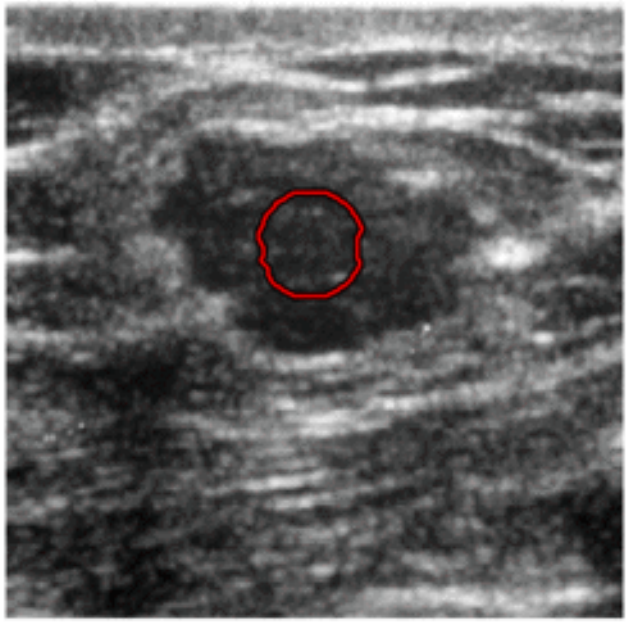


Watershed segmentation

- Marker-controlled watershed: it is possible to manually define "markers/seeds": each seed defines a water source, the others are ignored
- Built-in Matlab functions: `imregionalmin`, `imimposemin`

Active contours

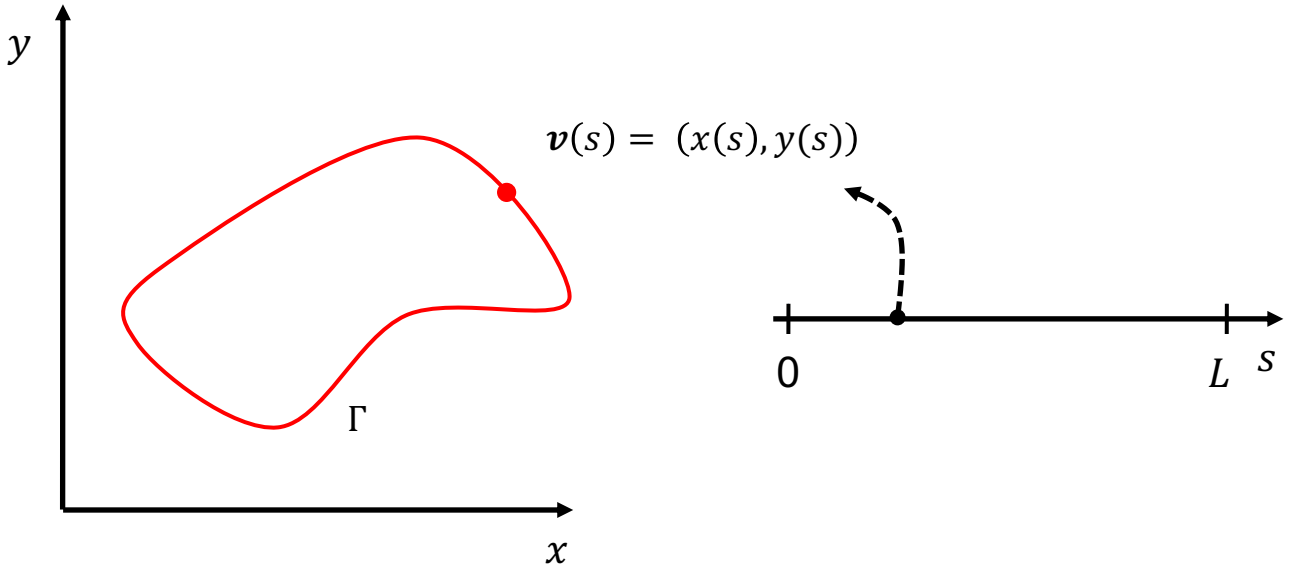
- Active contours are a segmentation approach in which **an initial contour is iteratively deformed** through time towards a goal segmentation mask
- The deformation (called **contour evolution**) is realized by iteratively minimizing an energy term using partial differential equations
- The contour will evolve until it reaches **a minimum of the energy term**



Credits: https://github.com/pmneila/morphsnakes

Snakes

- Kass et al. [1] introduced active contours for the first time as “snakes”
- Explicit representation:
  - The contour Γ is explicitly represented by the function $\Gamma = \boldsymbol{v}(s)$, a parametrization where s is the arch length parameter
  - The contour does not have to be closed
  - At each time step during contour evolution, the representation changes: $\boldsymbol{v}^0(s), \boldsymbol{v}^1(s), \boldsymbol{v}^2(s), \ldots$



[1] Micheal Kass et al. Snakes: Active contour models. IJCV 1988

Snakes

- The total energy of the snake, to be minimized, is defined as follows:

$$E_{snake\_global} = \int_0^L E_{snake}(\boldsymbol{v}(s))ds \qquad E_{snake}(\boldsymbol{v}(s)) = E_{int}(\boldsymbol{v}(s)) + E_{image}(\boldsymbol{v}(s))$$

- The energy consists of two terms:
  1. Internal energy (E_{int}): it tries to keep the contour regular and smooth
     - The first part minimizes the length of the contour (which will then tend to shrink: modifications of this term are possible to instead favour a constant length during contour evolution)
     - The second part minimizes the curvature of the contour, thus minimizing the bending of the contour during the evolution)

$$E_{int}(\boldsymbol{v}(s)) = \alpha \left\| \frac{d\boldsymbol{v}(s)}{ds} \right\|^2 + \beta \left\| \frac{d^2\boldsymbol{v}(s)}{ds^2} \right\|^2$$

contour length element

contour curvature element

α and β are user-defined weights

Snakes

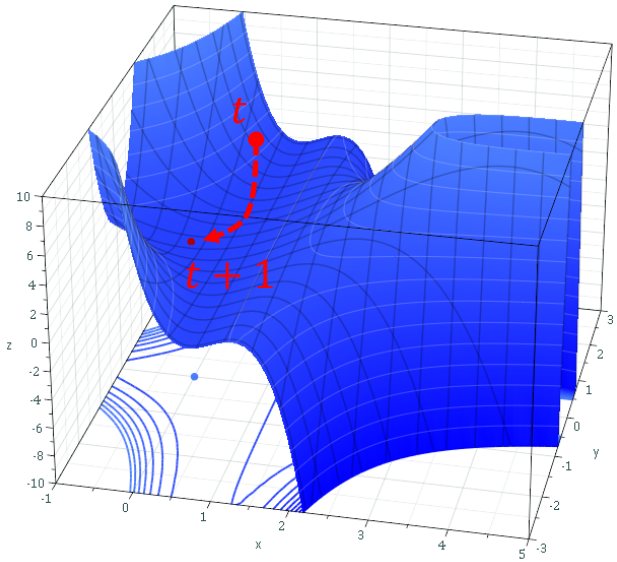
2. Image or external energy (E_{image}): it tries to attract the contour towards the edges in the image (intuitively similar to gravity pull)
   - It is usually implemented using the negative of the (squared) gradient of the image: zero in uniform areas and negative closer to edges

$$E_{image}(\boldsymbol{v}(s)) = -\left\|\nabla I\left(\boldsymbol{v}(s)\right)\right\|^2 = -I_x^2\left(\boldsymbol{v}(s)\right) - I_y^2\left(\boldsymbol{v}(s)\right)$$

- Optimization (i.e. minimization of the snake energy) is carried out through gradient descent:

$$\boldsymbol{v}^{t+1}(s) = \boldsymbol{v}^t(s) - \Delta t \nabla E_{snake\_global}\left(\boldsymbol{v}^t(s)\right)$$

- This defines contour evolution
- Evolution is stopped when points don't move anymore (i.e. an energy minimum is reached)
- Δt is the user-defined time step



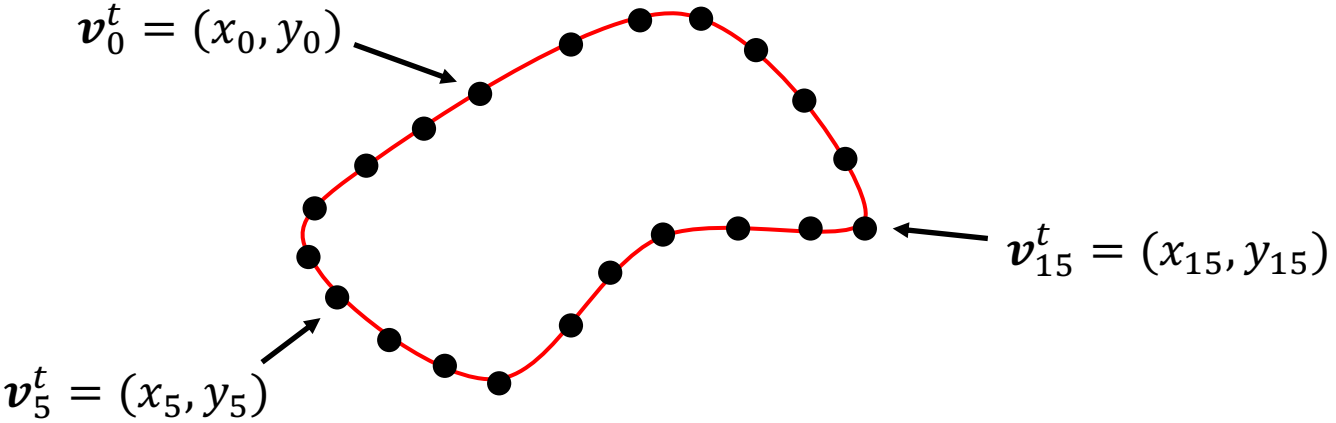
E_{snake_global}

Credits: https://en.wikipedia.org/wiki/Maxima_and_minima

Snakes

- Performing the differentiation of the energy is non trivial
- To solve this, **the contour is discretized**: only N points on the contour are explicitly defined
- The contour can be defined through spline interpolation

$$\boldsymbol{v}^t(s) \rightarrow \boldsymbol{v}_i^t \qquad\qquad i \in [1, N]$$



- The new energy functional reads

$$E_{snake\_global} = \int_0^L E_{snake}(\boldsymbol{v}(s))ds \approx \sum_{i=0}^{N} E_{snake}(\boldsymbol{v}_i)$$

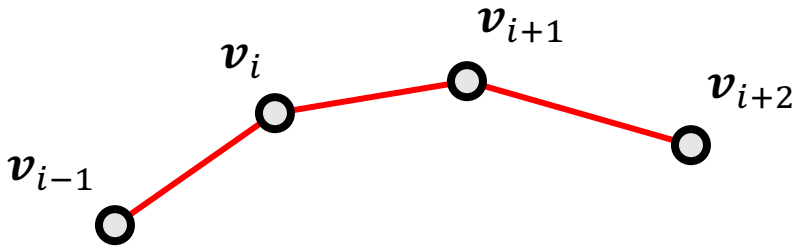
Snakes

- This allows a great simplification of the gradient descent formula:

$$\nabla E_{snake\_global} \approx \nabla \sum_{i=0}^{N} E_{snake}(\boldsymbol{v}_i) = \sum_{i=0}^{N} \nabla E_{snake}(\boldsymbol{v}_i)$$

$$\boldsymbol{v}_i^{t+1} \approx \boldsymbol{v}_i^t - \Delta t \nabla E_{snake}(\boldsymbol{v}_i)$$

- This means that we can approximate the evolution of the contour by simply **moving each point independently** with its own gradient of the energy term
- Simplified formulas for E_{int} can also be adopted:



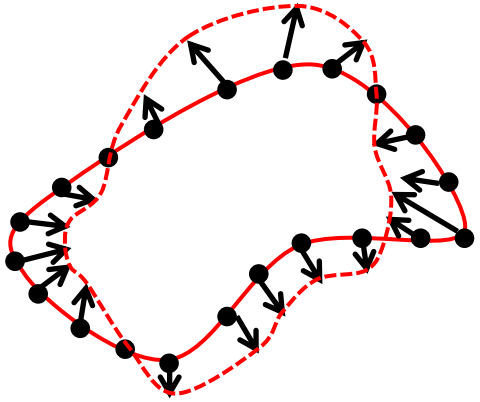
$$\frac{\partial \boldsymbol{v}}{\partial s} \approx \boldsymbol{v}_{i+1} - \boldsymbol{v}_i$$

$$\frac{\partial^2 \boldsymbol{v}}{\partial s^2} \approx -\boldsymbol{v}_{i-1} + 2\boldsymbol{v}_i - \boldsymbol{v}_{i+1}$$

Snakes

Full derivation (study is optional):

$$
\begin{aligned}
E(\mathbf{v}) &= E_{int}(\mathbf{v}(s)) + E_{ext}(\mathbf{v}(s)) \\
&= \alpha \left| \frac{\partial \mathbf{v}}{\partial s} \right|^2 + \beta \left| \frac{\partial^2 \mathbf{v}}{\partial^2 s} \right|^2 - I_x^2 - I_y^2 \\
&= \alpha \left| \mathbf{v}_{i+1} - \mathbf{v}_i \right|^2 + \beta \left| -\mathbf{v}_{i-1} + 2\mathbf{v}_i - \mathbf{v}_{i+1} \right|^2 - I_x^2 - I_y^2 \\
&= \alpha (x_{i+1} - x_i)^2 + \alpha (y_{i+1} - y_i)^2 + \beta (-x_{i-1} + 2x_i - x_{i+1})^2 \\
&\quad + \beta (-y_{i-1} + 2y_i - y_{i+1})^2 - I_x^2 - I_y^2
\end{aligned}
$$



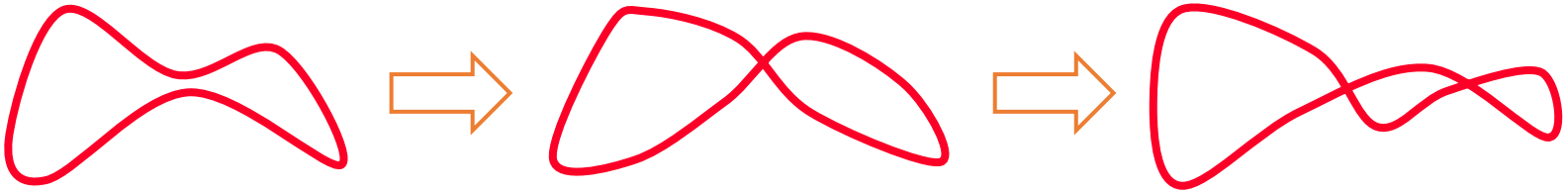
$$
\begin{aligned}
\frac{\partial E(\mathbf{v}(s))}{\partial x} &= -2\alpha(x_{i+1} - x_i) + 2\alpha(x_i - x_{i-1}) \\
&\quad 4\beta(-x_{i-1} + 2x_i - x_{i+1}) - 2\beta(-x_i + 2x_{i+1} - x_{i+2}) - 2\beta(-x_{i-2} + 2x_{i-1} - x_i) \\
&\quad -2I_x I_{xx} - 2I_y I_{xy} \\
&= -2\alpha(x_{i+1} - x_i) + 2\alpha(x_i - x_{i-1}) \\
&\quad +2\beta(x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} + x_{i+2}) - 2I_x I_{xx} - 2I_y I_{xy}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E(\mathbf{v}(s))}{\partial y} &= -2\alpha(y_{i+1} - y_i) + 2\alpha(y_i - y_{i-1}) \\
&\quad 4\beta(-y_{i-1} + 2y_i - y_{i+1}) - 2\beta(-y_i + 2y_{i+1} - y_{i+2}) - 2\beta(-y_{i-2} + 2y_{i-1} - y_i) \\
&\quad -2I_x I_{xy} - 2I_y I_{yy} \\
&= -2\alpha(y_{i+1} - y_i) + 2\alpha(y_i - y_{i-1}) \\
&\quad +2\beta(y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2}) - 2I_x I_{xy} - 2I_y I_{yy}
\end{aligned}
$$

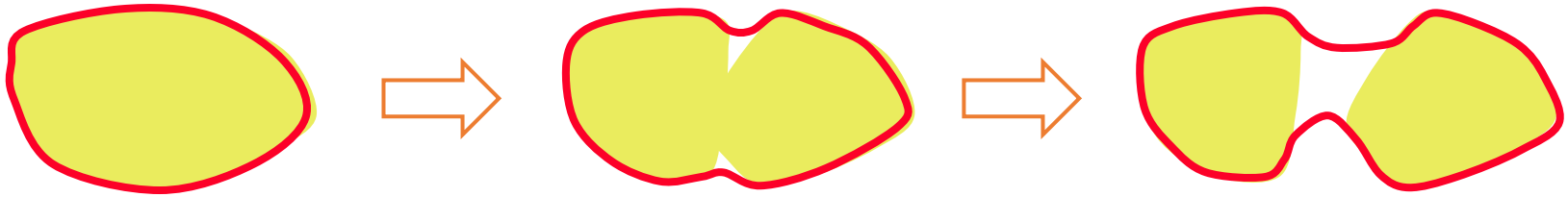
Snakes

Limitations:

- Snakes are **unaware of faraway edges**, since E_{image} is effective only very close to them. The result is thus dependent on the initialisation: initial contour should be close to object or risk to get stuck in a local minimum
- Results depend also on **several parameters** $(N, \Delta t, \alpha, \beta)$: their choice determines the final contour and may produce over-smoothing
- Snakes can self intersect, which can lead to non-well behaved representations potentially critical for evolution:

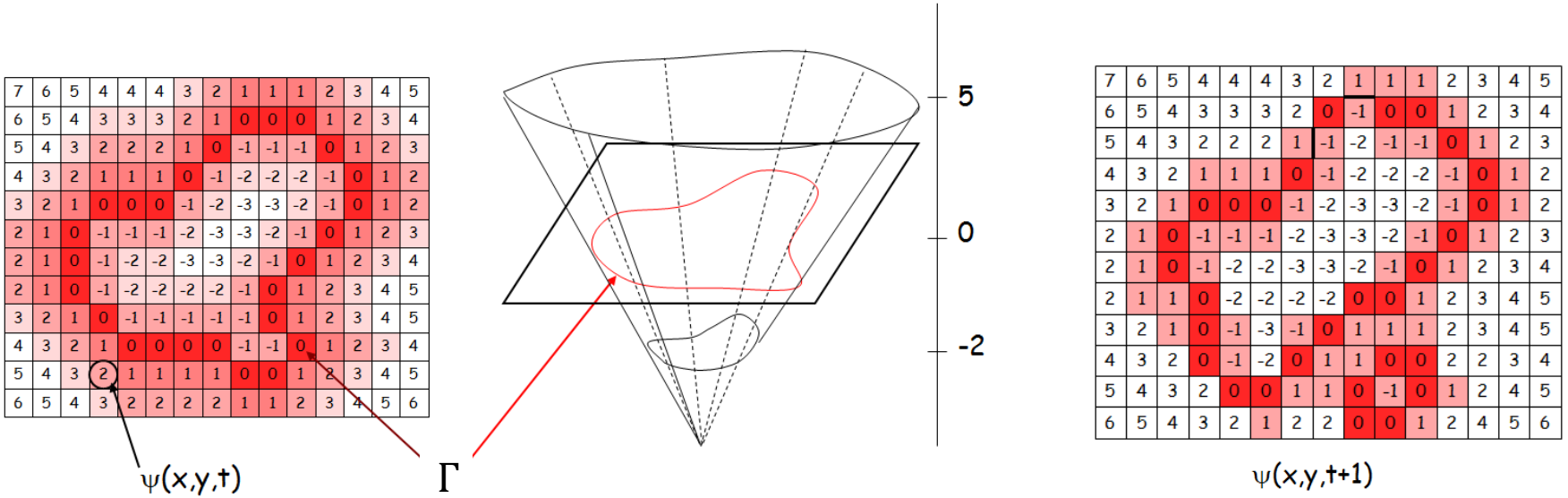


- Snakes poorly handle topological changes:



Level-sets

- Level-sets are an alternative family of active contour models that can better handle intersections and topological changes
- **Implicit representation**:
  - The contour Γ is represented as the zero level-set of a signed distance function ψ (i.e. negative inside the contour and positive outside)
  - In other words, at each time step, the contour is located at the intersection between the surface ψ and the hyperplane $z = 0$
  - The evolution is still defined by iteratively minimizing an energy term, but it is the function ψ that evolves, and not directly the contour



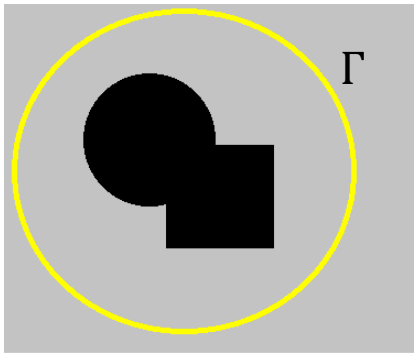
Common level-sets models

- **Geodesic active contours** (Caselles et al. [1]):
  - Similar to the energy formulation of snakes
  - Built-in Matlab function: `activecontour` (method: `'edge'`)
- **Region-based energy model** (Chan & Vese [2]):
  - Contour evolution is not based on edges, but on minimizing the variance of pixel intensity both inside and outside the contour:

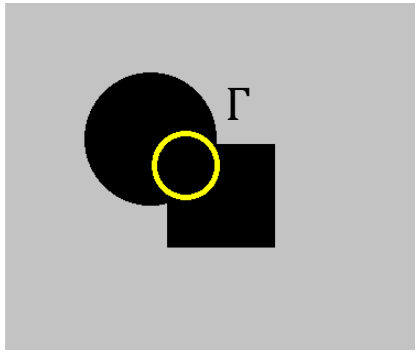
$$E_{image} = \int_{inside\ \Gamma} |I - \mu_i|^2 dxdy + \int_{outside\ \Gamma} |I - \mu_o|^2 dxdy$$

μ_i: mean inside int.
μ_o: mean outside int.

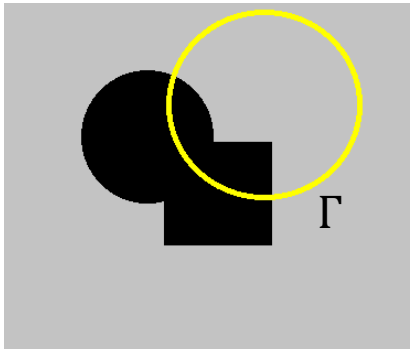
  - Built-in Matlab function: `activecontour` (method: `'Chan-Vese'`)



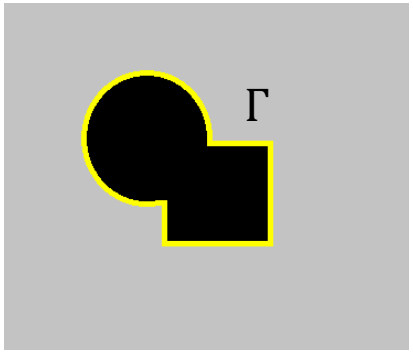
$E_{image} > 0$



$E_{image} > 0$



$E_{image} > 0$



$E_{image} \sim 0$

[1] Vicent Caselles et al. Geodesic Active Contours. IJCV 1997
[2] Tony Chan and Luminita Vese. Active Contours Without Edges. IEEE TIP 2001

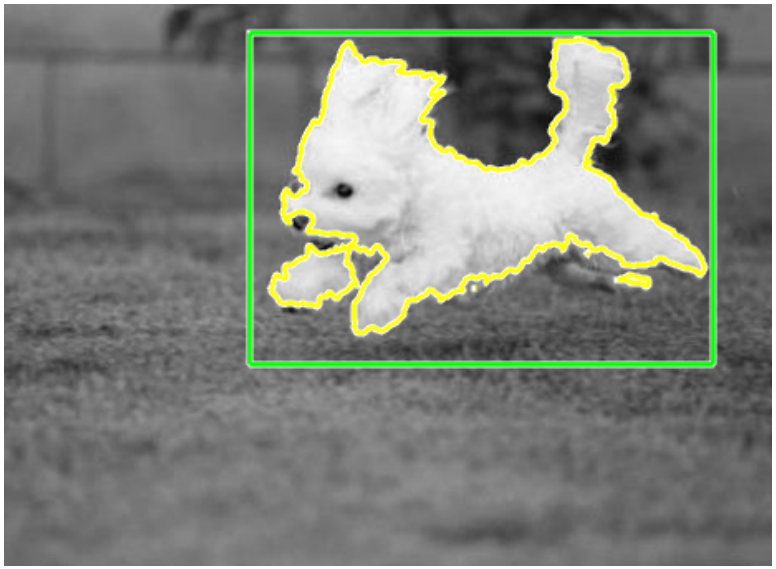
Region-based energy model

$$E_{image} = \int_{inside\ \Gamma} |I - \mu_i|^2 dxdy + \int_{outside\ \Gamma} |I - \mu_o|^2 dxdy$$

- Example #1

```
I = rgb2gray(imread('whitedog1.jpg'));
imshow(I); hold on;

% Definition of initial mask
mask = false(size(I));
mask(20:220, 150:430) = true;
visboundaries(mask,'Color','g');

% Level-set, Chan-Vese model
B = activecontour(I, mask, 250, 'Chan-Vese');
% Display segmentation mask as contour
visboundaries(B,'Color','y');
```

Max iterations



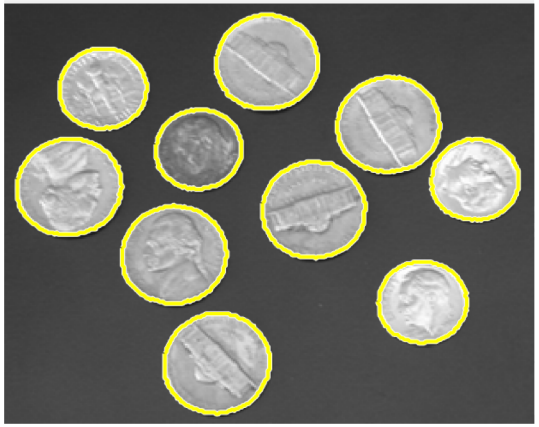
Credits: https://stuffmydoghates.wordpress.com/2008/10/30/cute-little-white-dogs/

Region-based energy model

$$E_{image} = \int_{inside\ \Gamma} |I - \mu_i|^2 dxdy + \int_{outside\ \Gamma} |I - \mu_o|^2 dxdy$$

- Example #2

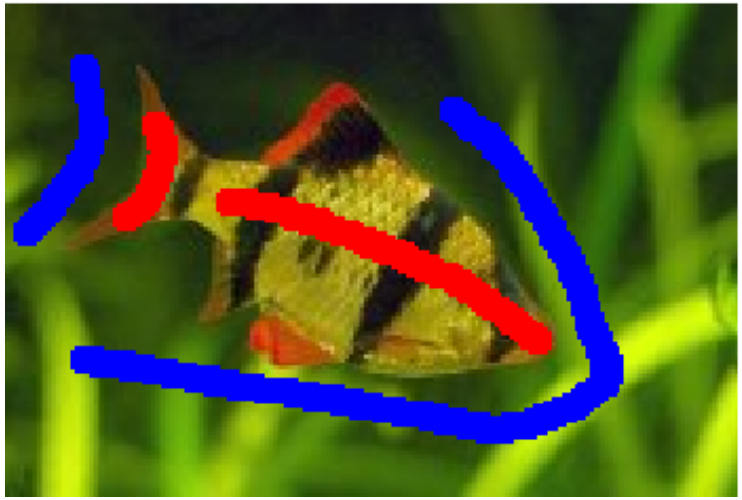
```
I = imread('coins.png');
B = true(size(I));

% Level-set, Chan-Vese model
% For cycle to plot intermediate results
for i = 1:7
    figure;
    imshow(I);
    hold on;
    B = activecontour(I, B, 50, 'Chan-Vese');
    visboundaries(B,'Color','y');
    pause;
end
```



Interactive segmentation

- Interactive methods **require the user to provide "seeds"**, i.e. to annotate a small amount of pixels in the image with the correct label
- The seeds are used to extract information from the image, mainly:
  - Colour/intensity models for the different regions
  - Spatial context

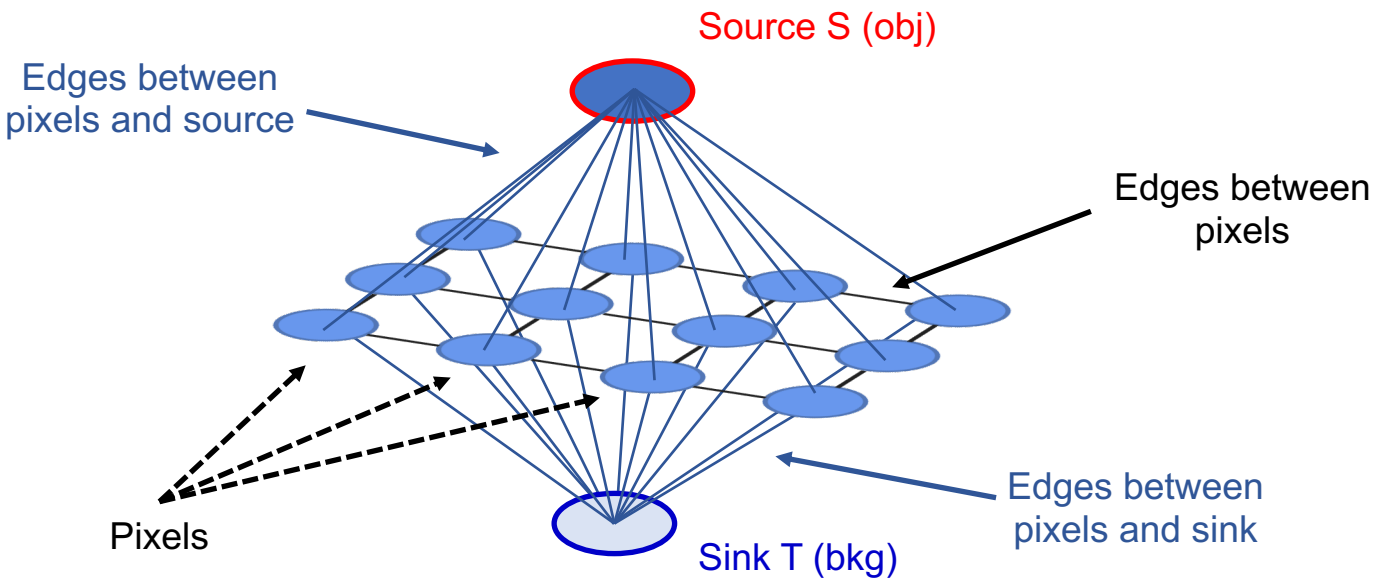
Object
Background



Credits: http://jgmalcolm.com/pubs/research.html

Graph cuts

- Introduced by Boykov & Jolly [1], graph cuts are a segmentation method particularly suitable for this task
- Each pixel in the image is considered a node in an undirected graph
- Two additional “terminal” nodes are included called source (S, associated with the object) and sink (T, associated with the background)
- Each pixel is connected to neighbours and to both S and T through edges with a specific weight, which are assigned based on an energy term



[1] Yuri Boykov and Marie-Pierre Jolly. Interactive Graph Cuts. ICCV 2001

Graph cuts

- The energy model, for a possible segmentation mask A, is

$$E(A) = \lambda \sum_{p} R_p(A_p) + \sum_{p,q;\, p \neq q} B_{p,q}(A_p, A_q)$$

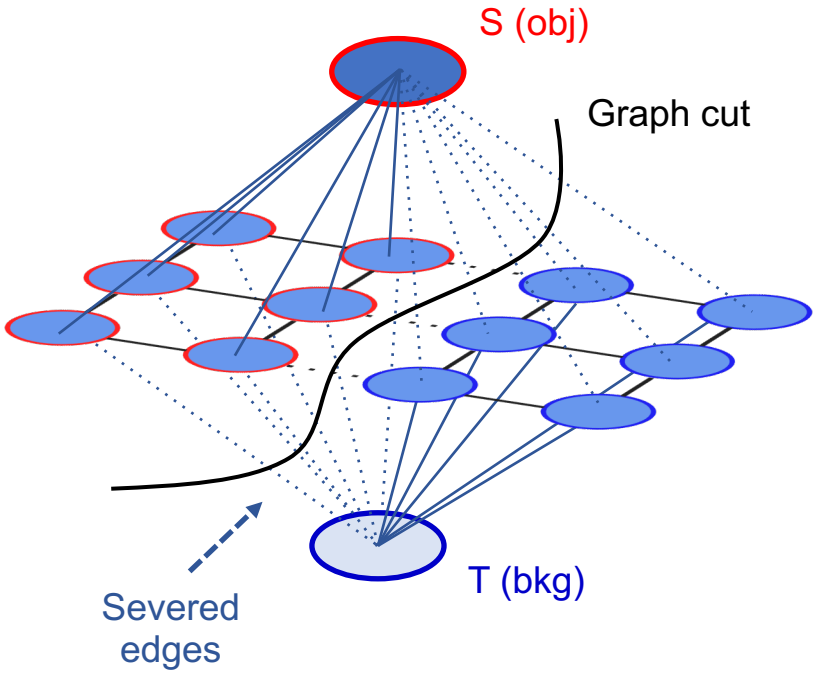
p,q: pixels in the image
A_p: label of pixel p
λ: user-defined weight

- Unary terms R_p:
  - They indicate the cost of assigning the label A_p (obj or bkg) to pixel p
  - A common choice consists in assigning a cost based on the probability of finding that pixel intensity in the histogram of the object (or of the background) as they were estimated in the seeds
- Pairwise terms $B_{p,q}$:
  - They indicate, for each couple of neighbouring pixels, the cost of assigning to them equal or different labels
  - A common choice consists in assigning a cost of 0 if the pixels have the same label, and a cost based on their difference in pixel intensity if they have different labels (high cost for similar intensity values and viceversa)
  - In this way, the pairwise terms act along the boundaries of A penalizing contours between pixels of similar intensity

Graph cuts

- These terms are embedded into the graph as edge weights:
  - Unary terms become edge weights between each pixel and S and T nodes (with the exception of seeded pixels, that get 0 edge weights towards their label and a very big one towards the other one)
  - Pairwise terms become edge weights between neighbouring pixels

- A **graph cut** consists in severing edges so to assign each pixel only to one of the two terminal nodes. The associated cost is the sum of the severed edge weights
- The **globally optimal** graph cut can be computed very efficiently using min-cut/max-flow algorithms, and defines the optimal segmentation
- Built-in Matlab function: use Image Segmenter app

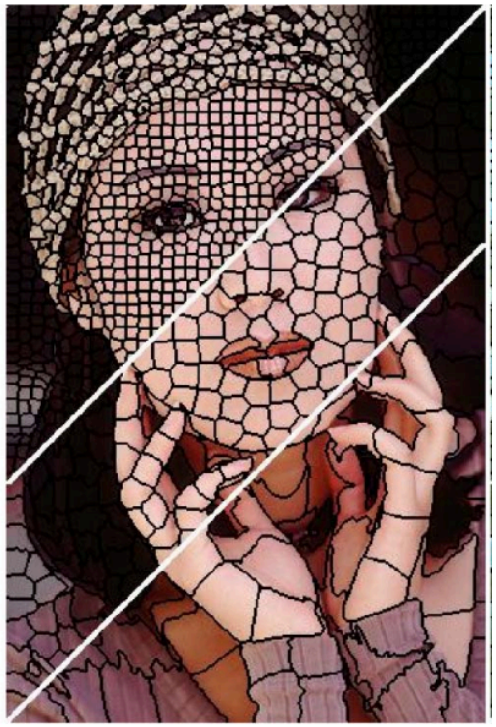


Superpixels

- A superpixel consists of a group of connected pixels with similar properties (e.g. colour), making them perceptually meaningful
- An image can be divided into superpixels, and these can be grouped together based on some algorithm to produce the final segmentation. This can be more efficient than processing each pixel separately
- They can be used as pre-processing for a variety of tasks, reducing computational complexity

Simple Linear Iterative Clustering

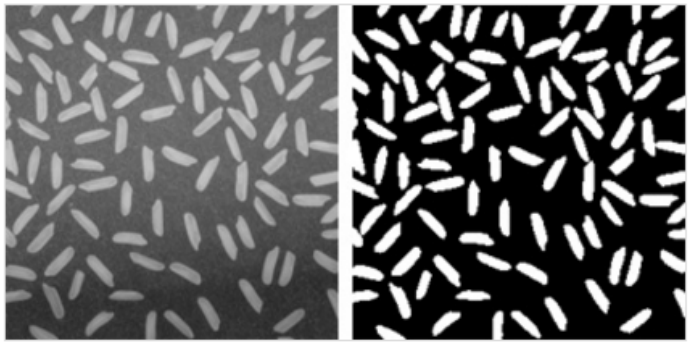
- Achanta et al. [1] have introduced a popular superpixel segmentation technique called Simple Linear Iterative Clustering (SLIC)
- SLIC adapts a K-means clustering approach to efficiently generate superpixels
- Algorithm:
  1. K regularly-spaced superpixel cluster centres are laid onto the image and moved to locations with the lowest (local) gradient position (this is to avoid placing a centre on an edge)
  2. Each pixel in the image is associated to a cluster based on its Euclidean distance from the centre computed in a 5D space defined by the LAB channels and (x, y) pixel positions
  3. New cluster centres are computed as the average position in the 5D space of the pixels of each cluster
  4. The procedure is iterated until convergence
- Built-in Matlab function: `superpixels`

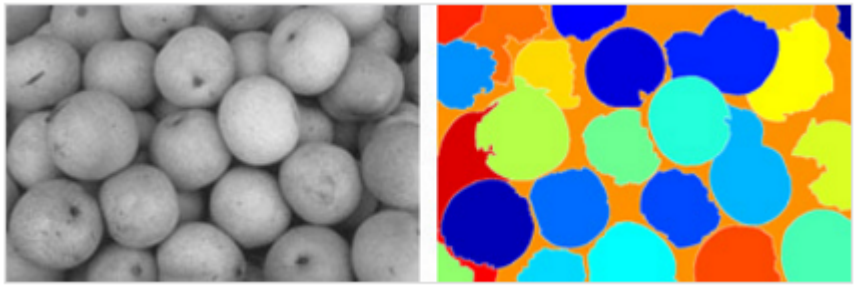


[1] Achanta et al.. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. PAMI 2012

Additional resources

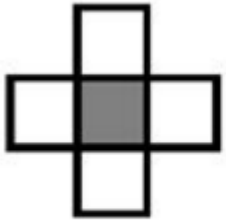
- Graph cuts
  - https://www.youtube.com/watch?v=KXQuR-InVFQ
  - https://www.youtube.com/watch?v=HMGX8HXskKk
  - https://uk.mathworks.com/help/images/segment-image-using-graph-cut.html
- Segmentation in Matlab
  - https://uk.mathworks.com/discovery/image-segmentation.html





Morphological operators

- Morphological operators apply a **non-linear structuring element** to an input binary image (or, in some cases, to a grayscale one)
- The value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbours
- They are usually used to “clean up” the obtained segmentation masks

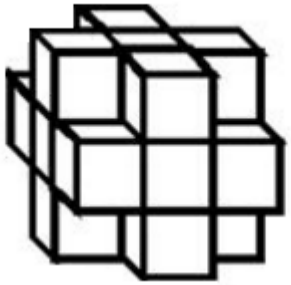


4-neighbourhood

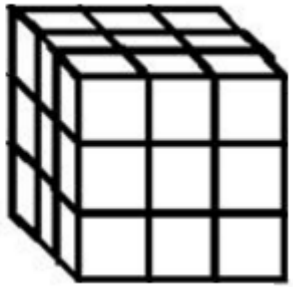
8-neighbourhood

Common 2D neighbourhoods

6-neighbourhood



18-neighbourhood

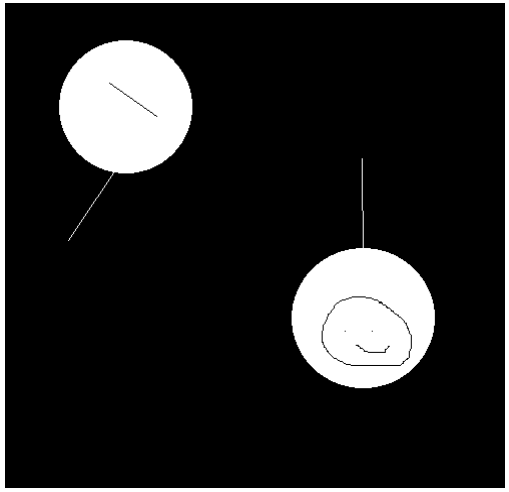


26-neighbourhood

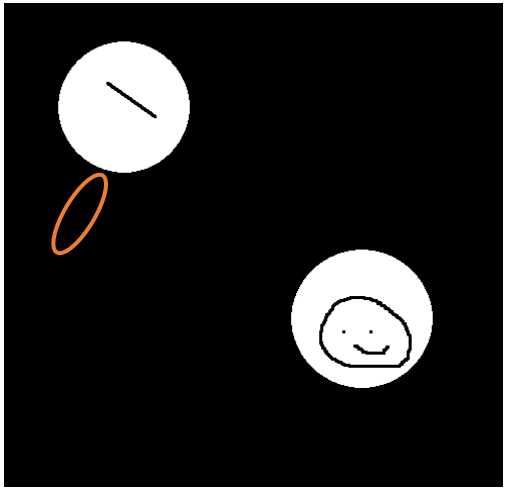
Common 3D neighbourhoods

Common morphological operators

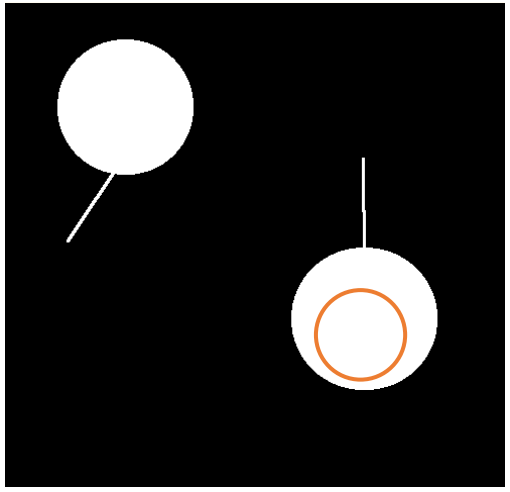
- Erosion (`imerode`):
  - Each pixel is substituted with the **min** intensity of the pixels in the selected neighbourhood
  - It removes islands and small objects
- Dilation (`imdilate`):
  - Each pixel is substituted with the **max** intensity of the pixels in the selected neighbourhood
  - It makes objects more visible and fills in small holes in objects



Original image



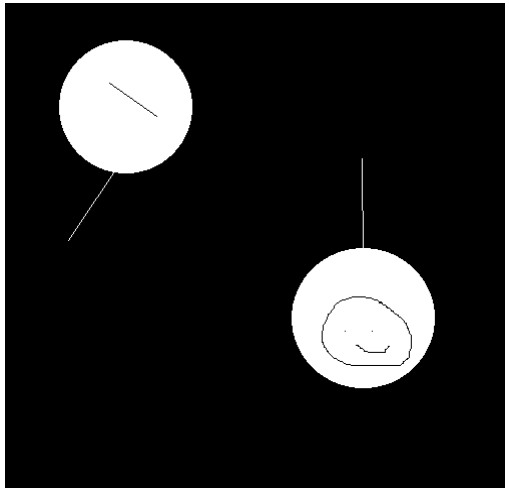
Erosion



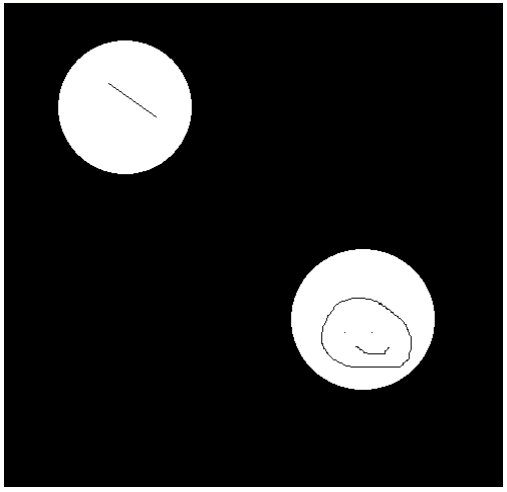
Dilation

Common morphological operators

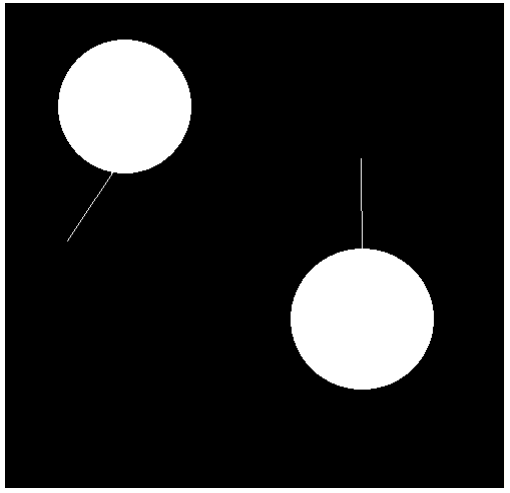
- Opening (`imopen`):
  - Erosion followed by dilation
  - It removes small objects while preserving the shape and size of larger objects
- Closing (`imclose`):
  - Dilation followed by erosion
  - It fills small holes while preserving the shape and size of the objects



Original image



Opening

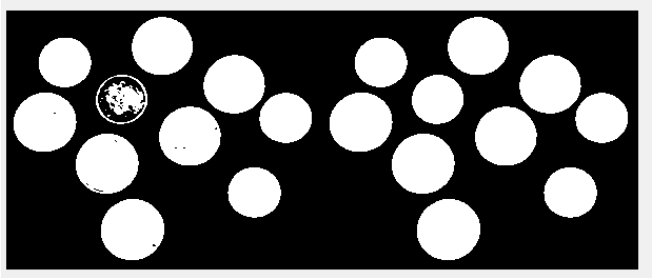


Closing

Common morphological operators

- `imfill`: fill holes in each binary image region

```
I = imread('coins.png');
T = graythresh(I);
J = I > 255*T;
K = imfill(J,'holes');
imshowpair(J,K,'montage’);
```



J K

- `bwareafilt`: filter regions in a binary image based on area (in pixels)

Range

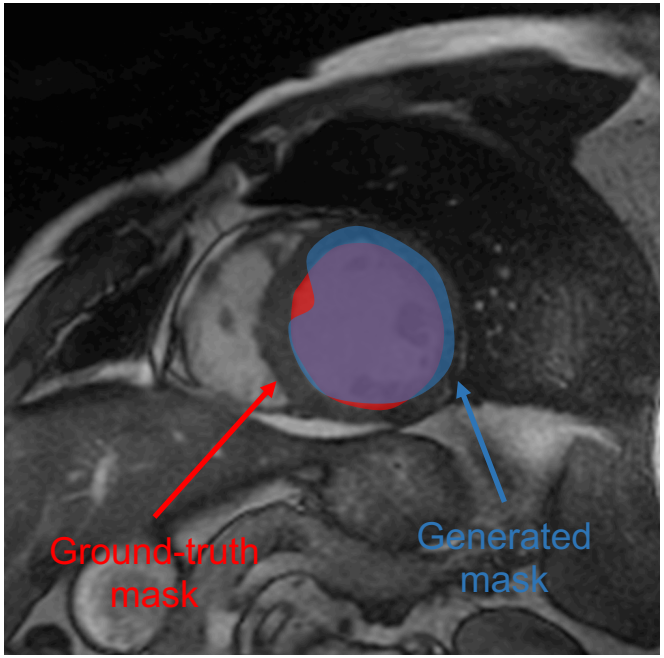
```
I = imread('text.png');
J = bwareafilt(I,[40 50]);
imshowpair(I,J,'montage’);
```

I J

- `regionprops`: get the properties of each region (e.g. area, perimeter,…). Based on this, you can filter regions to keep those within prescribed limits
- https://uk.mathworks.com/help/images/morphological-filtering.html

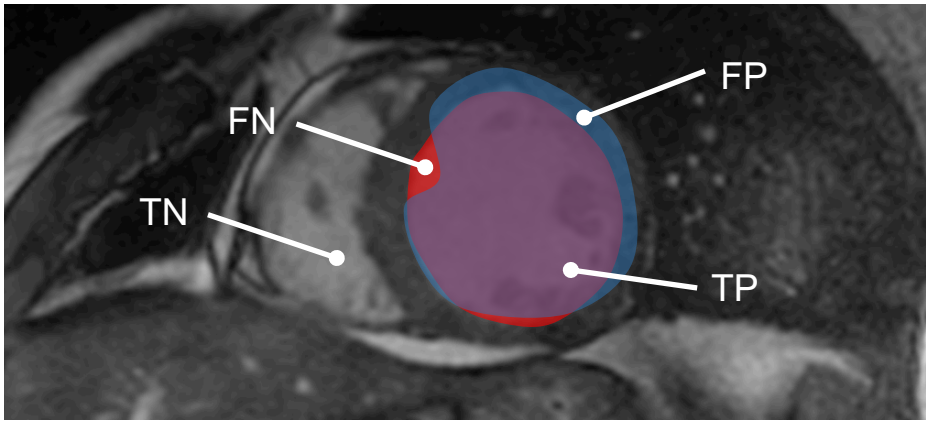
Segmentation metrics

- How do we evaluate the quality of a segmentation?
- In most cases, we will need a **ground-truth segmentation mask** that we can use for reference
- Ground-truth segmentation masks can be produced:
  - By experts annotators (e.g. radiologists looking at medical images)
  - Automatically (e.g. synthetic generation of image datasets)



Classification-based metrics

- Classification-based metrics **treat segmentation as a classification task at pixel level**. They are based on the comparison, for each pixel, between the label of the generated (GS) segmentation mask and that of the ground-truth (GT) one
- For a whole image, in a binary segmentation case, we count:
  - # True positives (TP): # pixels correctly labelled as foreground
  - # True negatives (TN): # pixels correctly labelled as background
  - # False positives (FP): # pixels labelled as foreground in GS which actually were labelled as background in GT
  - # False negatives (FN): # pixels labelled as background in GS which actually were labelled as foreground in GT



Classification-based metrics

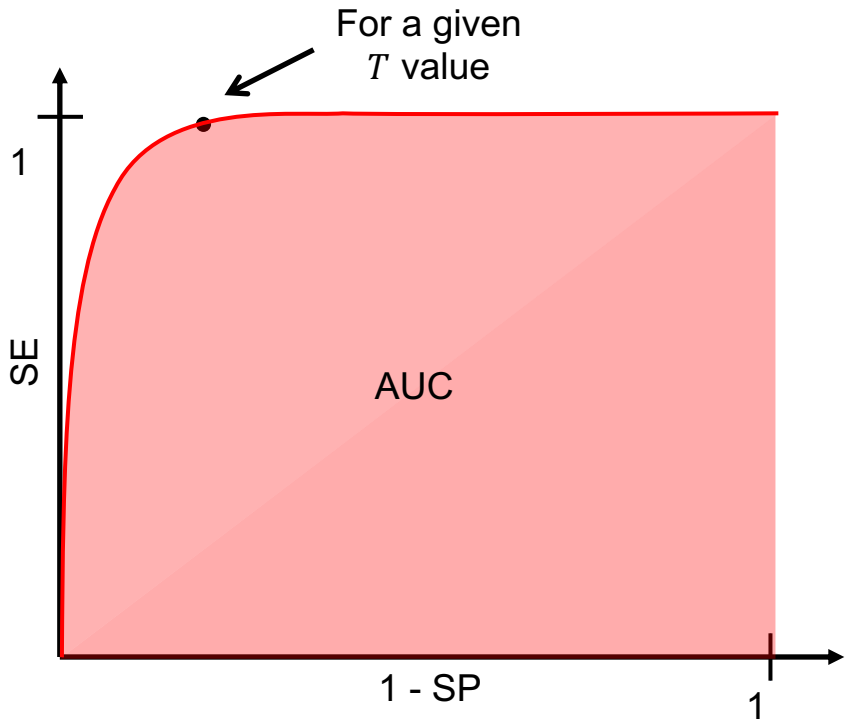
Confusion Matrix		Ground-truth Segm. mask	
		Foreground	Background
Generated Segm. mask	Foreground	TP	FP
	Background	FN	TN

- Main metrics:
  - **Sensitivity** (also recall, true positive rate): SE = TP / (TP+FN)
    "The percentage of GT positives correctly identified"
  - **Specificity** (also true negative rate): SP = TN / (TN+FP)
    "The percentage of GT negatives correctly identified"
  - **Precision**: PR = TP / (TP+FP)
    "The percentage of GS positives which were GT positives"
  - **Accuracy**: ACC = (TP+TN) / (TP+TN+FP+FN)
    "The percentage of correctly identified pixels"
- More metrics/details:
  https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Receiver operating characteristic

- The **receiver operating characteristic** (ROC) curve allows to determine the variation of classification metrics under changes to a given parameter (e.g. a threshold T)
- A sweep of the given parameter T is performed in all its range, the values for SE and SP are measured throughout and finally plotted in a SE vs (1-SP) chart

- The closer to the top-left corner, the better
- Different segmentation methods can be compared by measuring the **Area Under the Curve** (AUC)



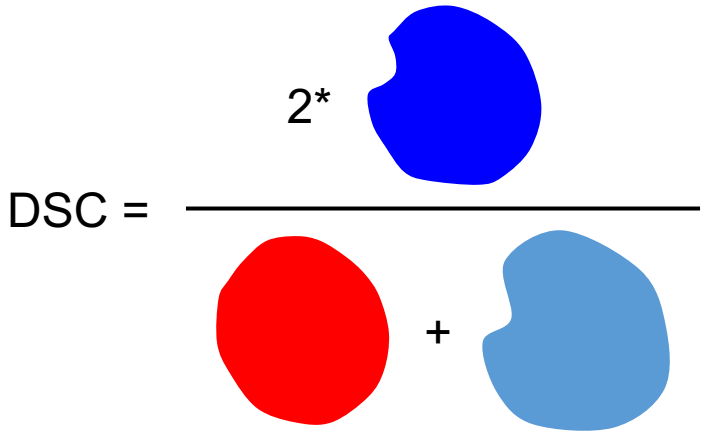
Classification-based metrics

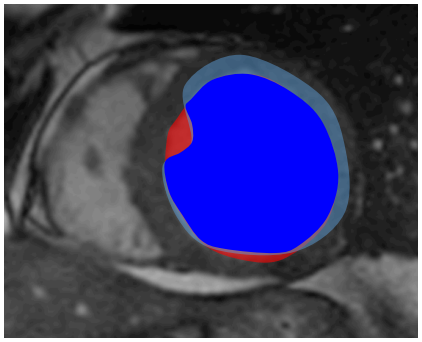
- Ideal case: FP = FN = 0
  ⟶ Confusion matrix becomes diagonal
  ⟶ SE = SP = ACC = 100%

- Limitations of the main classification-based metrics:
  1. Class imbalance:
     if number of TP << TN, then the metrics provide misleading results (e.g. a GS with only background pixels generates a very high ACC)
  2. Lack of geometrical & topological evaluation:
     these metrics don't take into account the differences in spatial distribution between GS and GT

Dice-Sørensen Coefficient

- The Dice-Sørensen Coefficient (DSC) aims at removing the first limitation
- It is the fraction between the area of intersection between GS and GT divided by the area of their union





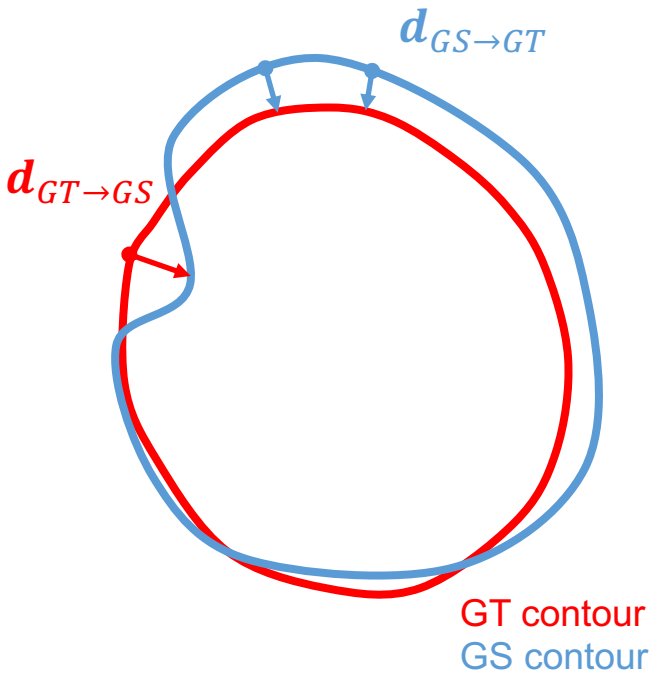
- The closer to 100%, the greater the superposition between GS and GT
- It can also be formulated as DSC = 2TP / (2TP+FP+FN)
  The absence of TN in the formula removes the bias due to class imbalance

Distance-based metrics

- The distance-based metrics aims at removing the second limitation
- They take into account the distances between the contours of GS and GT
- For each of point along GS, the distance to the closest point in GT is recorded. The resulting list is called $\boldsymbol{d}_{GS\rightarrow GT}$
- For each of point along GT, the distance to the closest point in GS is recorded. The resulting list is called $\boldsymbol{d}_{GT\rightarrow GS}$

- **Hausdorff distance** (HD):
  $HD = max[max(\boldsymbol{d}_{GS\rightarrow GT}), max(\boldsymbol{d}_{GT\rightarrow GS})]$
  It measures the maximum minimum distance between the two contours

- **Mean absolute distance** (MAD):
  $MAD = mean[mean(\boldsymbol{d}_{GS\rightarrow GT}), mean(\boldsymbol{d}_{GT\rightarrow GS})]$
  It measures the mean minimum distance between the two contours



Overview of next week's lecture

Image matching:

- Interest point detection
  - Harris detector
  - Scale-adapted Harris detector
  - Harris-Laplace detector
  - Scale-invariant feature transform (SIFT)
- Feature descriptors
  - Basic features
  - SIFT features
  - Speeded-Up Robust Features (SURF)
- Matching algorithm