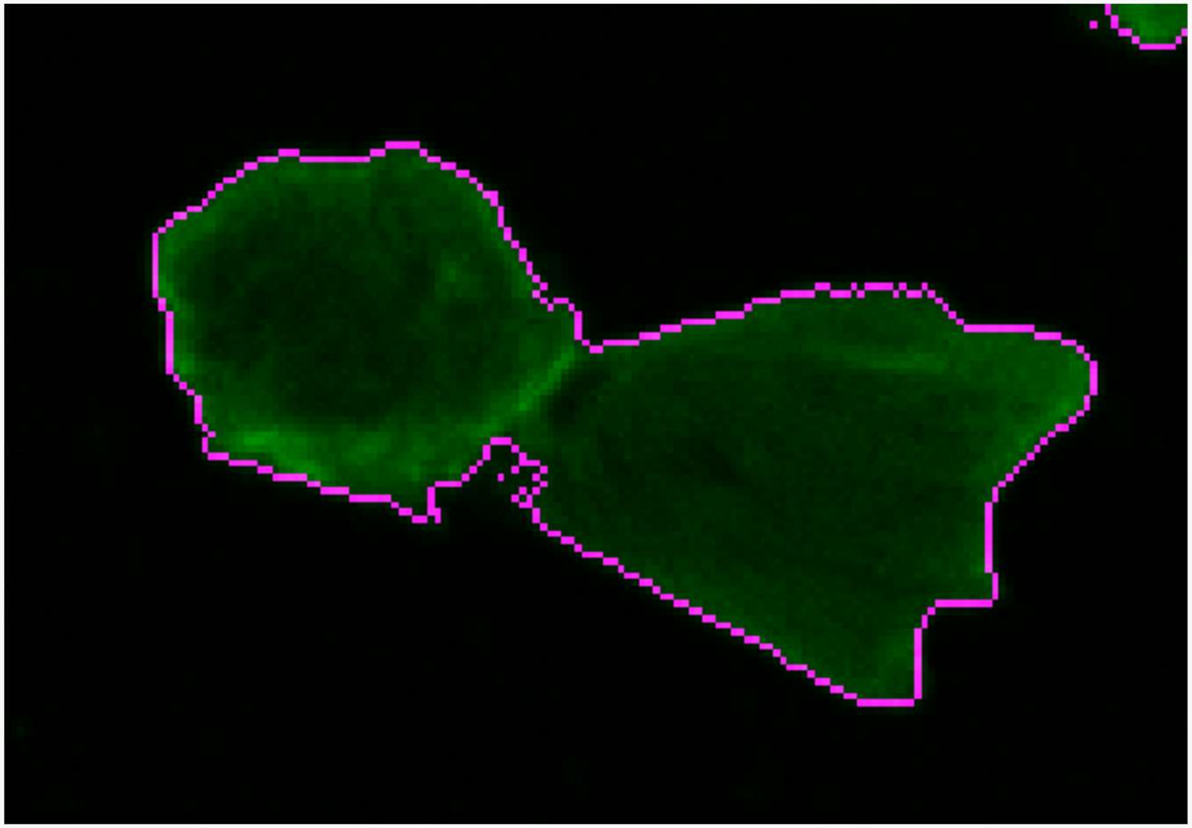
Lecture 22:
Separating clusters of objects using watershedding

University of Colorado Boulder

MCDB/BCHM 4312/5312
Fall 2020

Connected objects are counted as one



Logical indexing

A logical array is used to index elements in a matrix

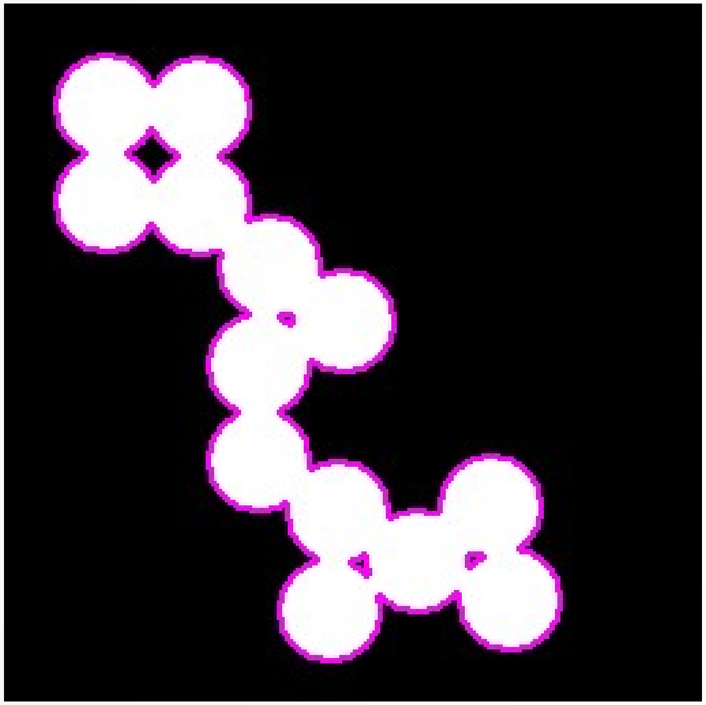
Logical indexing

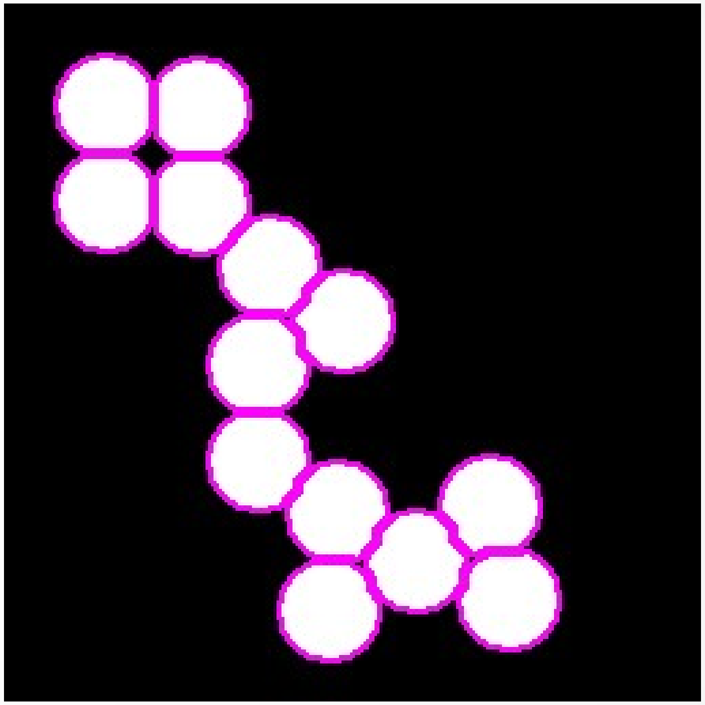
Index must have same size and number of elements as matrix

Example

```
%Create a 4x4 matrix of random numbers
A = rand(4);
L = A > 0.5;  %Create a logical array
A(L)
```

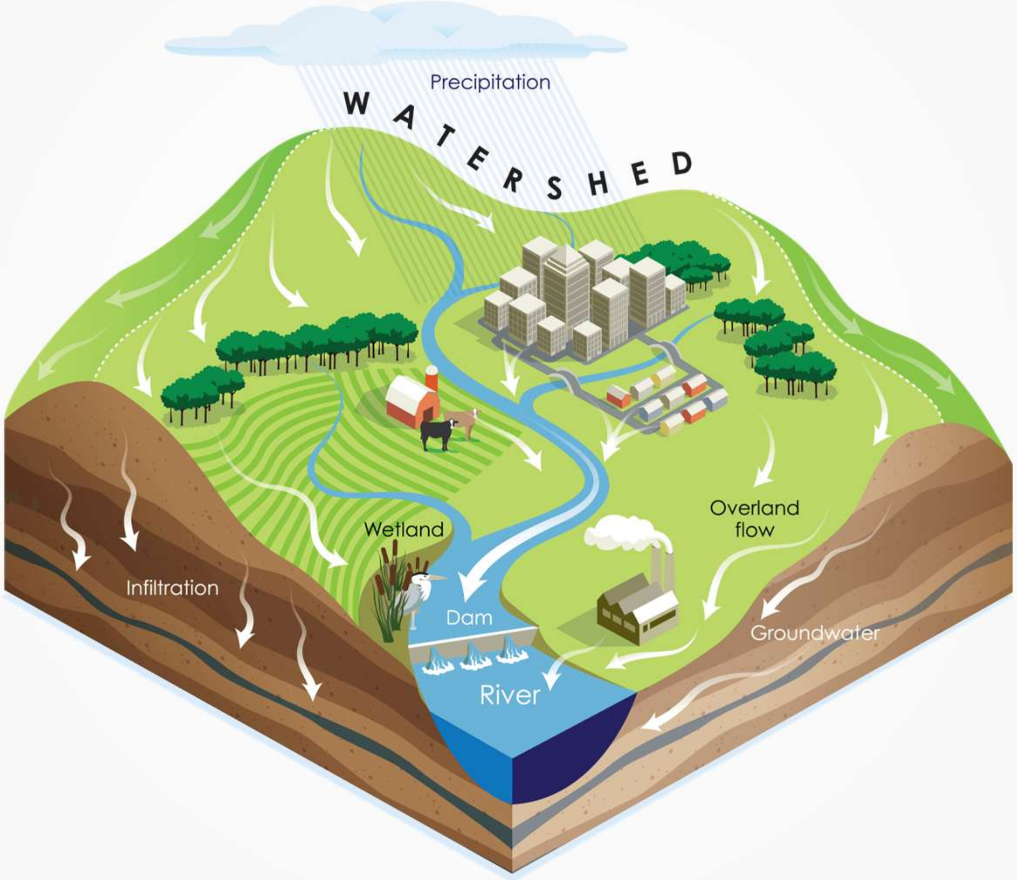
The problem



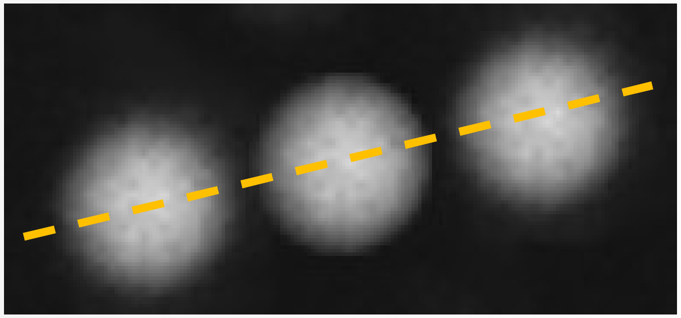


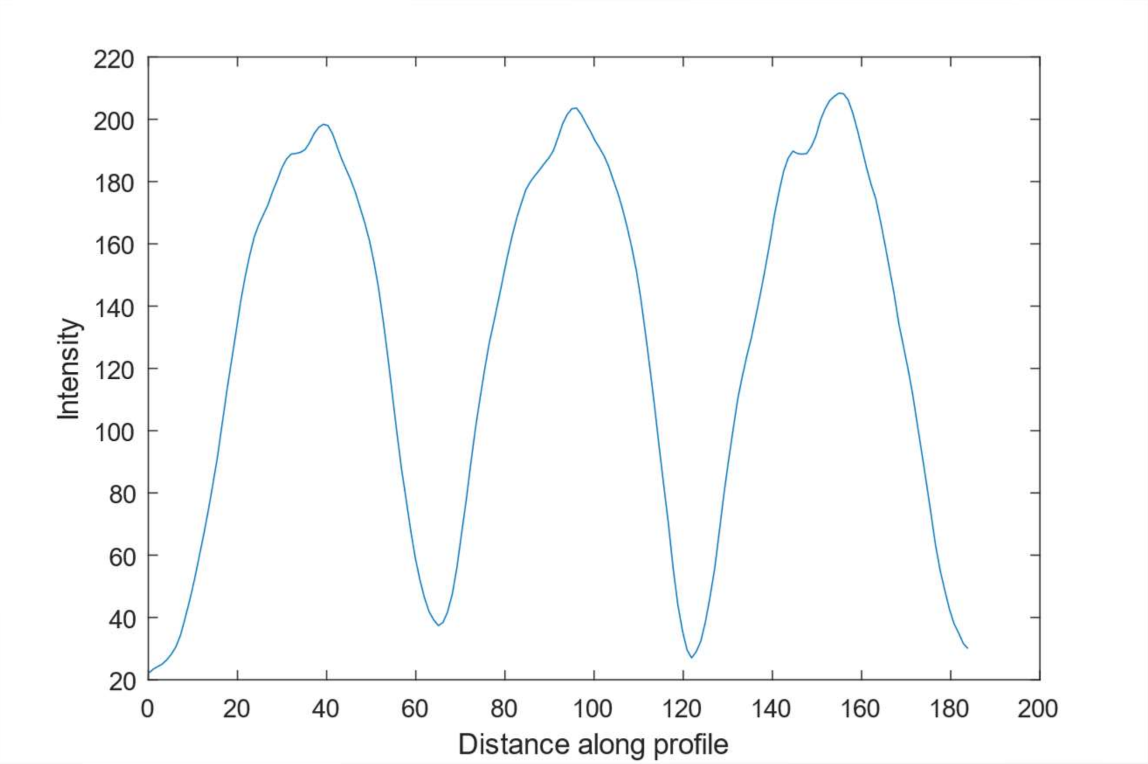
Desired output

The watershed algorithm

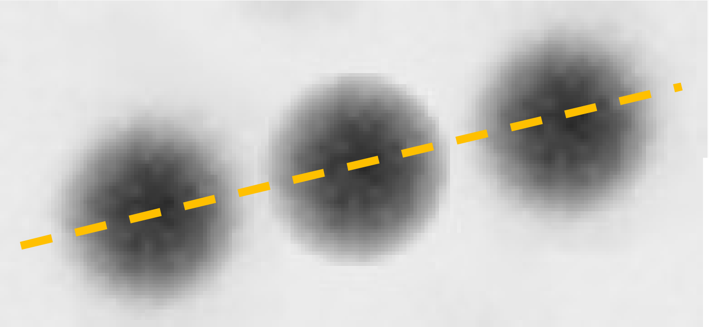


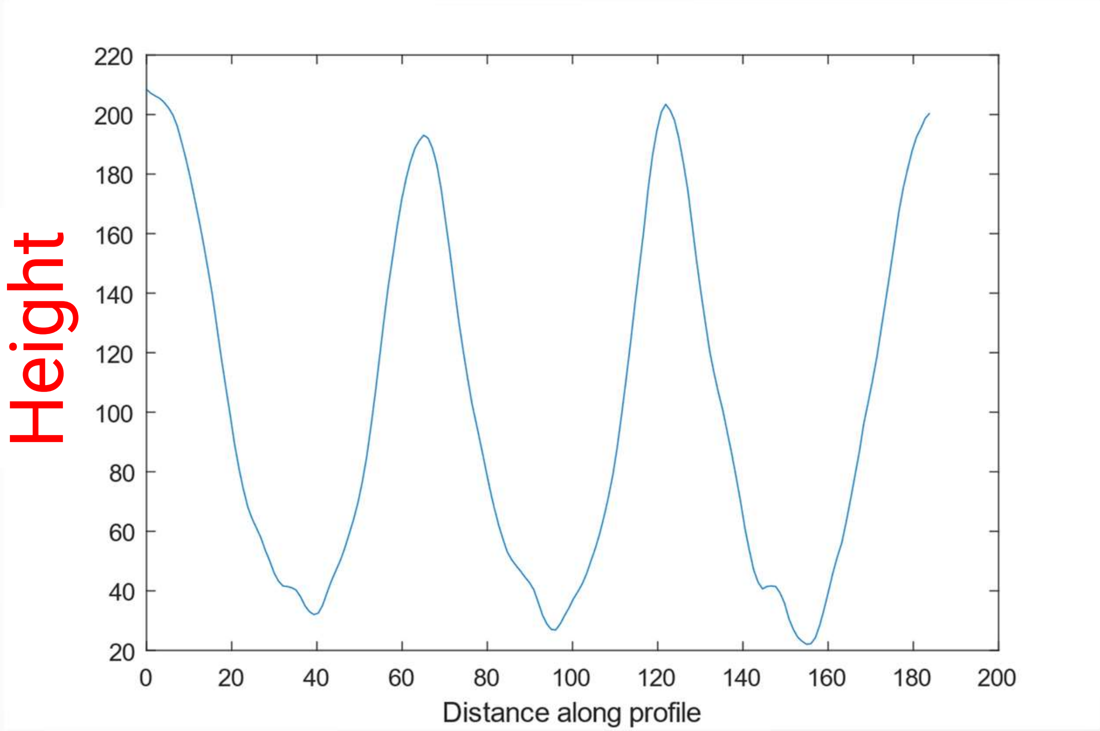
Intensity profile of an image



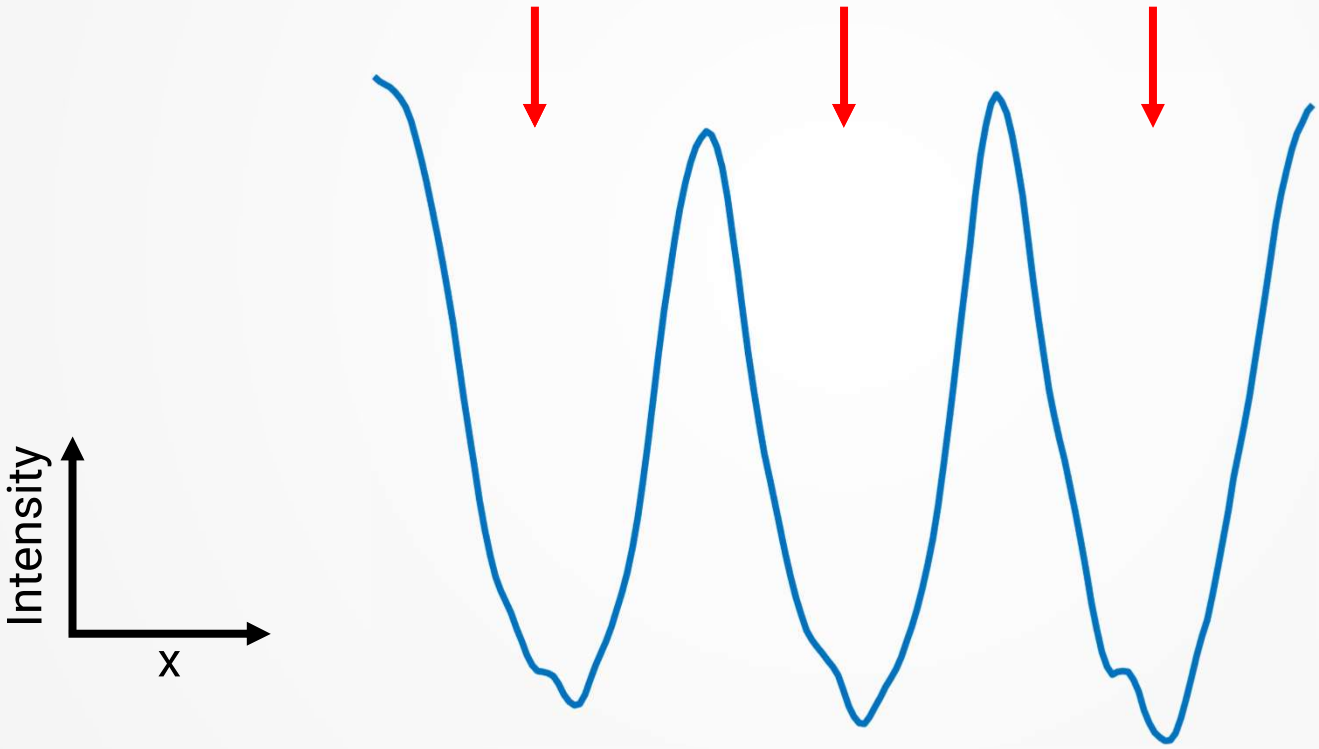


Invert image/profile

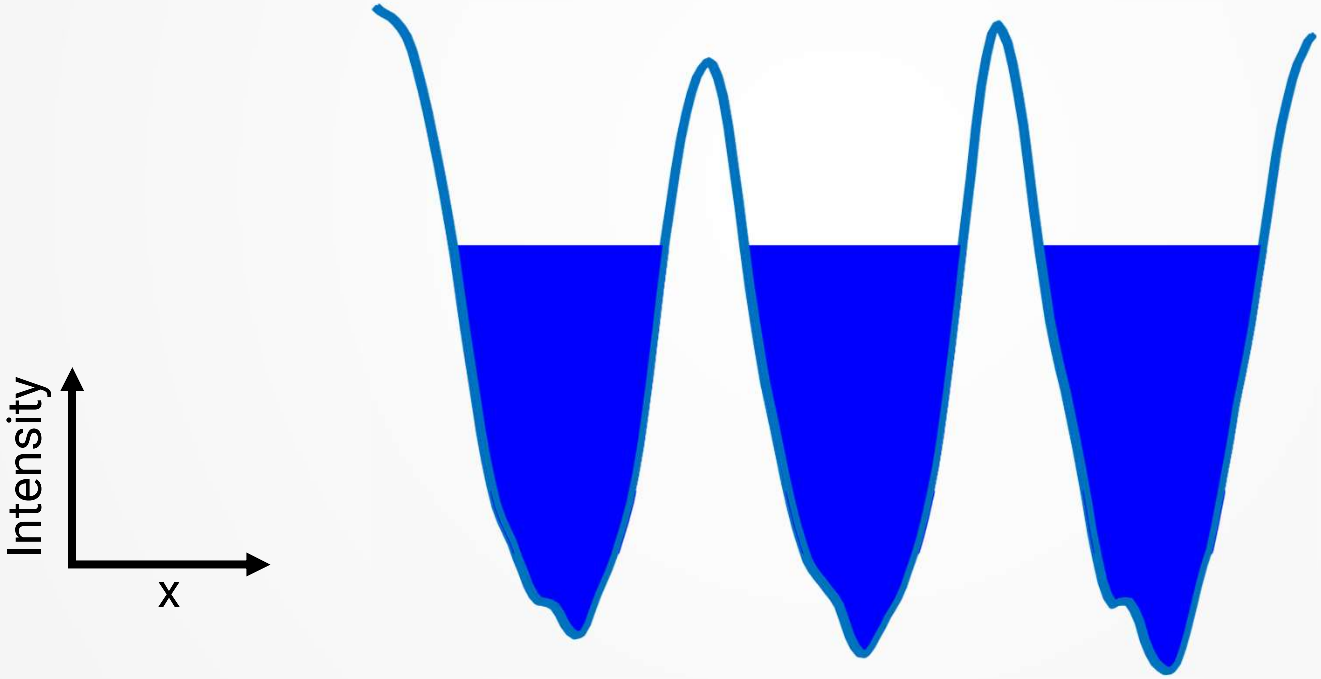




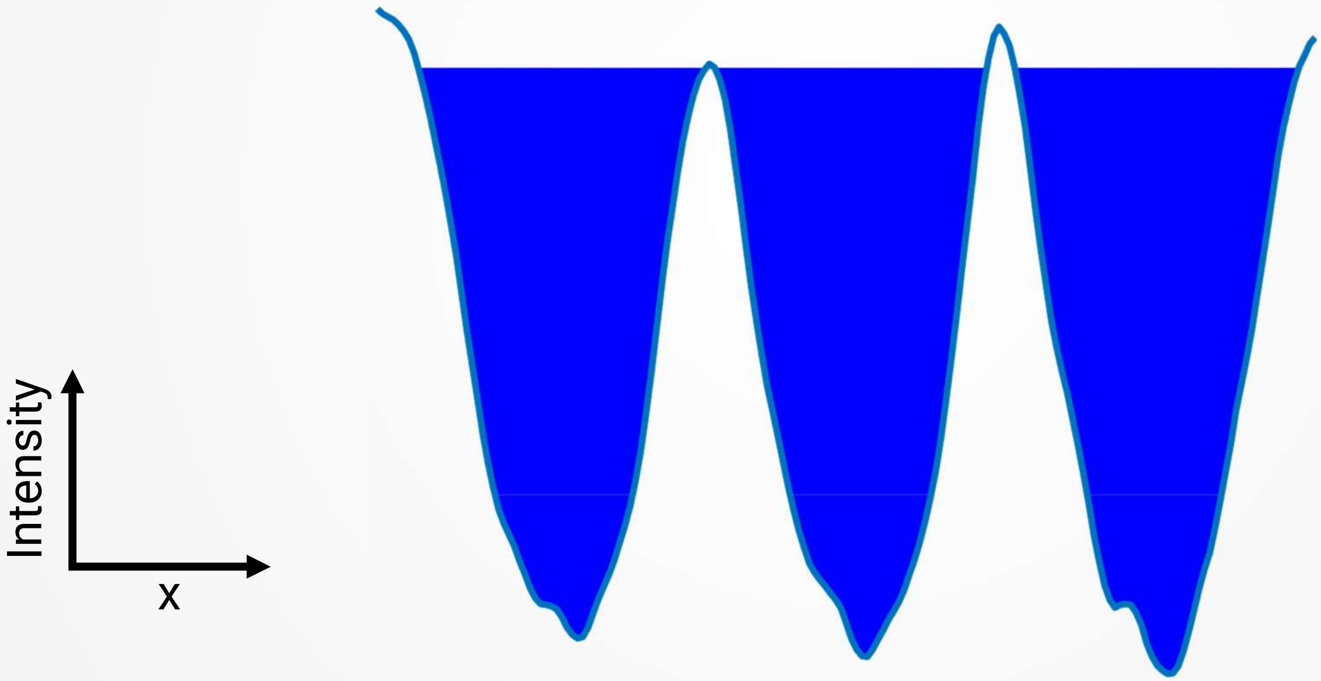
Basins



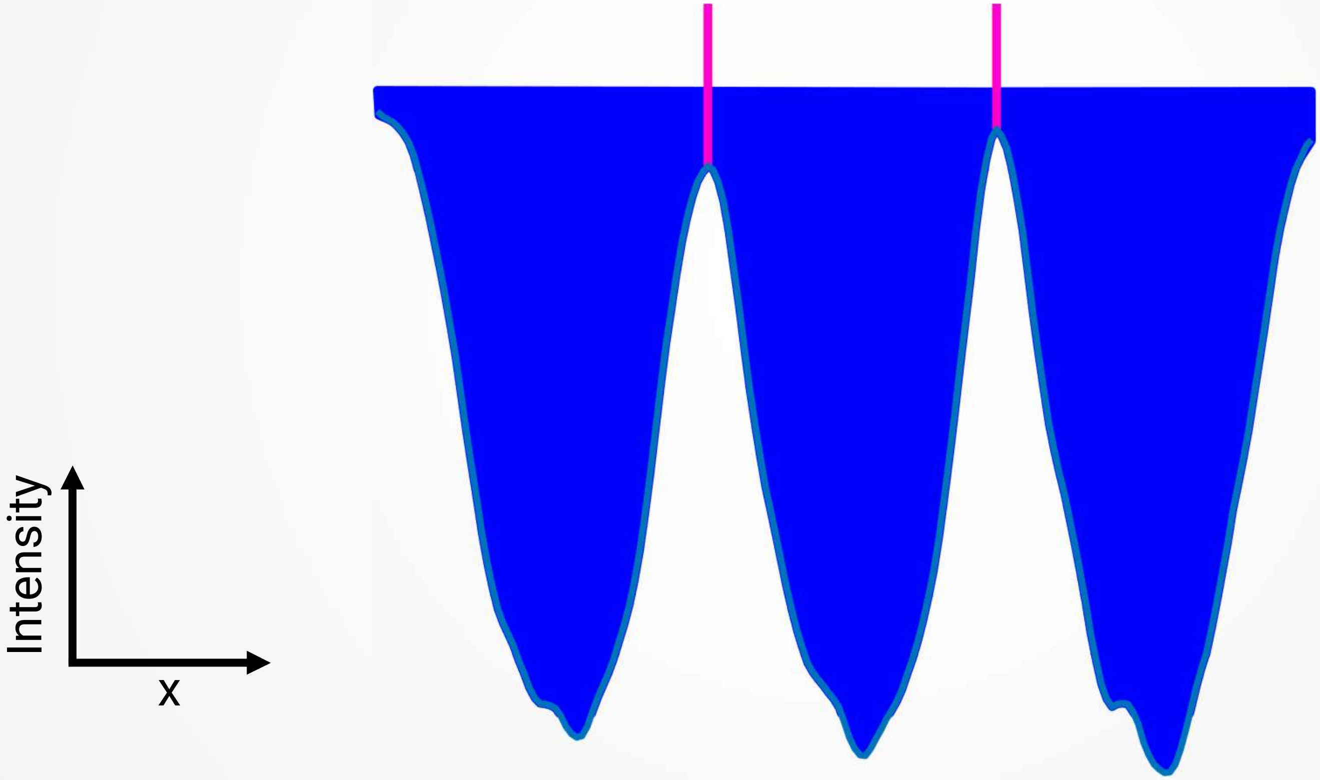
Imagine it is raining and the basins are filling up



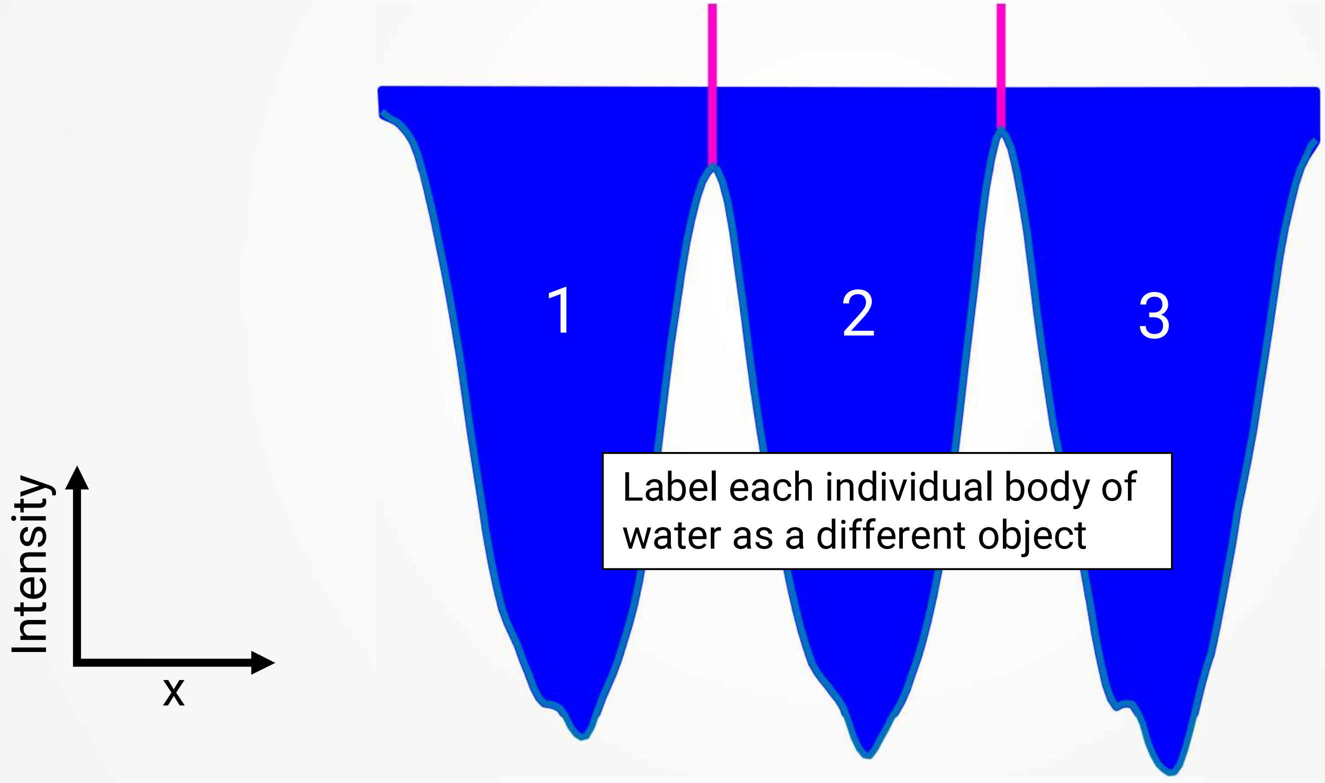
Imagine it is raining and the basins are filling up



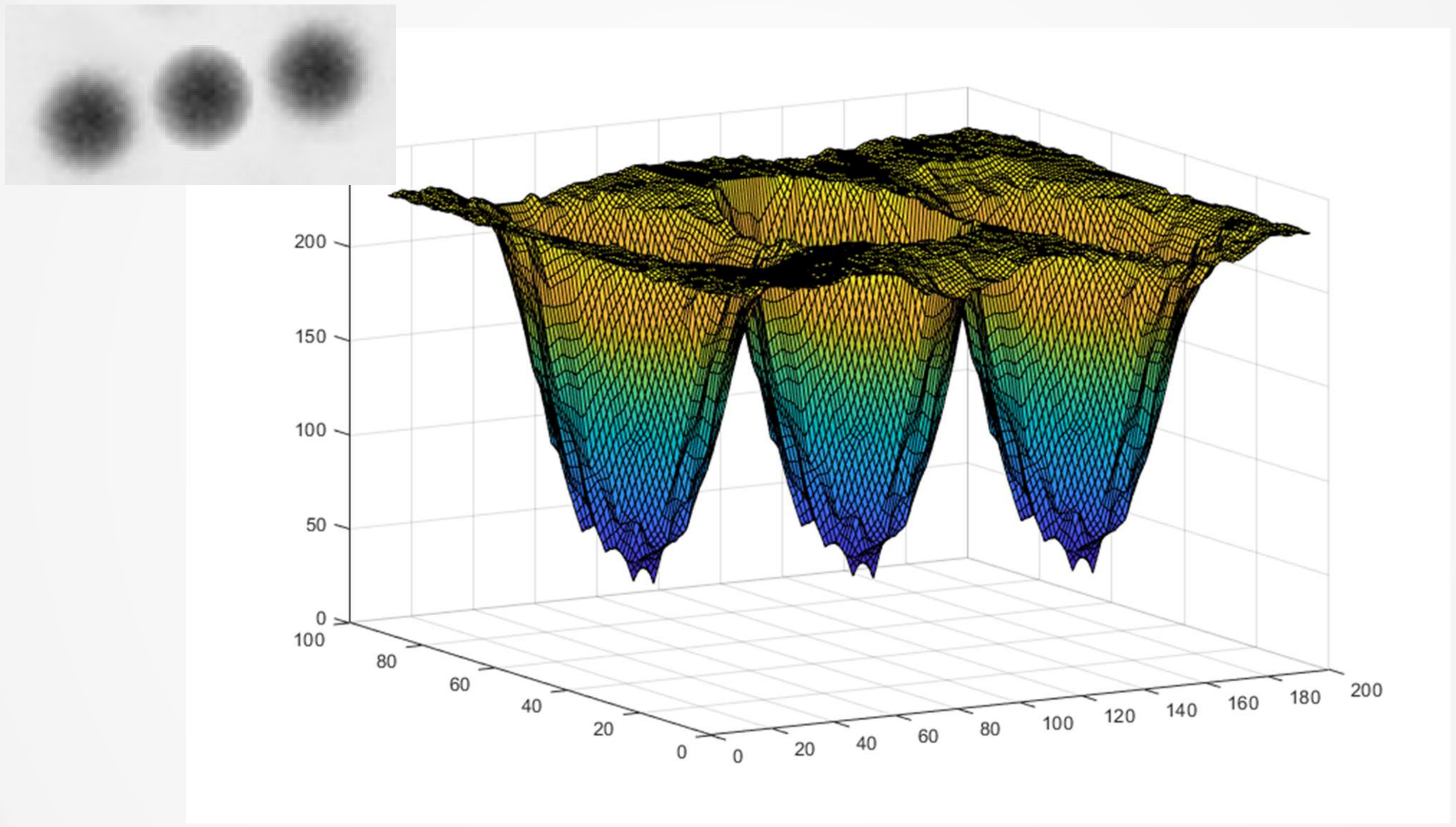
Mark locations where the bodies of water touch
(ridge lines = edge of an object)



Mark locations where the bodies of water touch
(ridge lines = edge of an object)

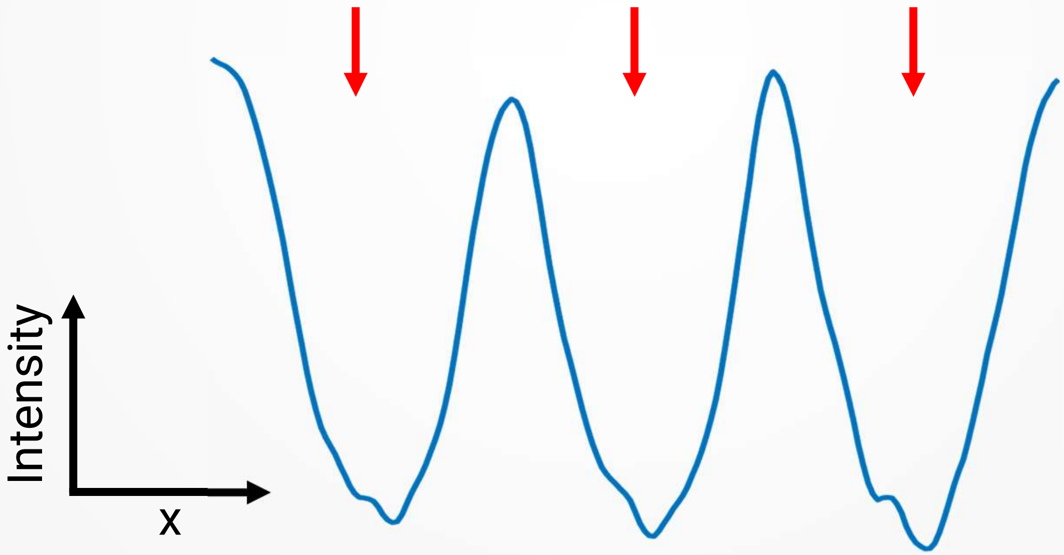


Label each individual body of water as a different object


200
150
100
50
0
100
80
60
40
20
0
0
20
40
60
80
100
120
140
160
180
200

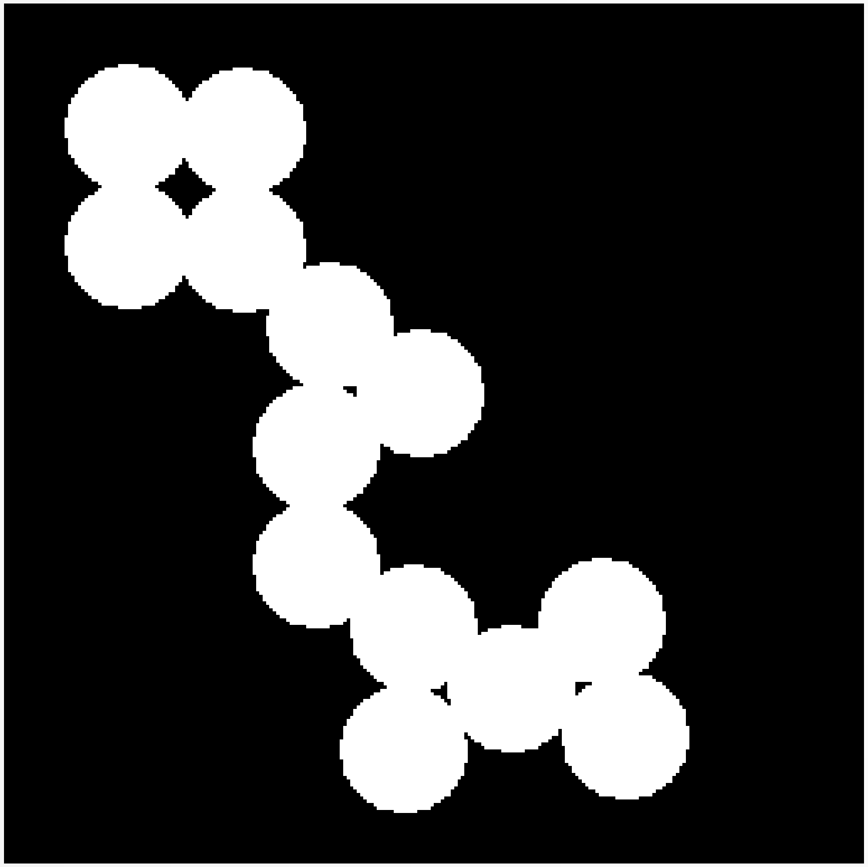
Requirements for the watershed algorithm

- Each object in the input image must be a "basin"
- The center of each object should be near the bottom of the basin



Steps for performing the watershed transform

1. Segment objects of interest
2. Convert the mask into an intensity profile using the distance transform
3. Refine the distance transform
4. Run the watershed algorithm
5. Update the original mask

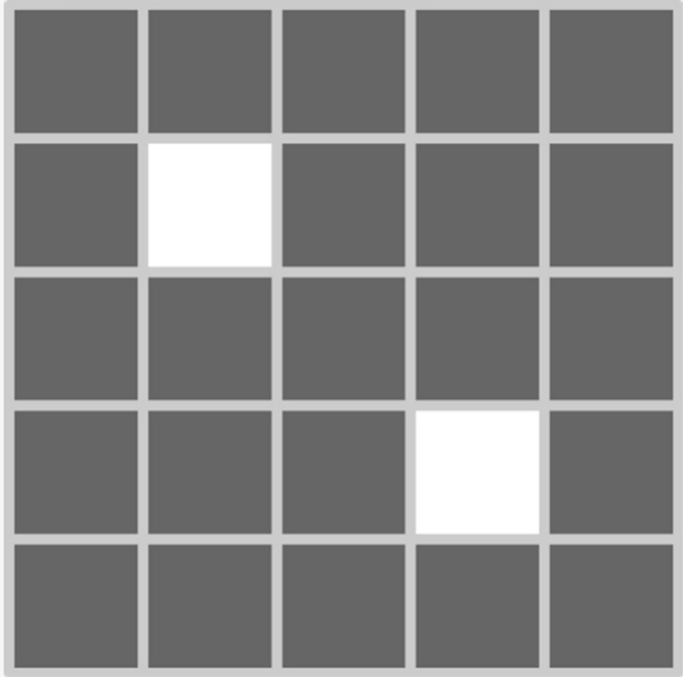


After segmentation, we have a mask

How do we transform this into a suitable input for the watershed algorithm?

The distance transform

computes the distance between each pixel and the nearest nonzero pixel



Input

1.41	1	1.41	2.24	3.16
1	0	1	2	2.24
1.41	1	1.41	1	1.41
2.24	2	1	0	1
3.16	2.24	1.41	1	1.41

Distance transform

dd = bwdist(M)

M = logical array (mask)

dd = distance transform (double)

Task

- Read in the image 'circles.png'
- Calculate the distance transform

```
dd = bwdist(M)
```

Task

- Read in the image 'circles.png'
- Calculate the distance transform
- Display the distance transform as a 3D surface

```
mesh(dd)
```

L = watershed(dd)

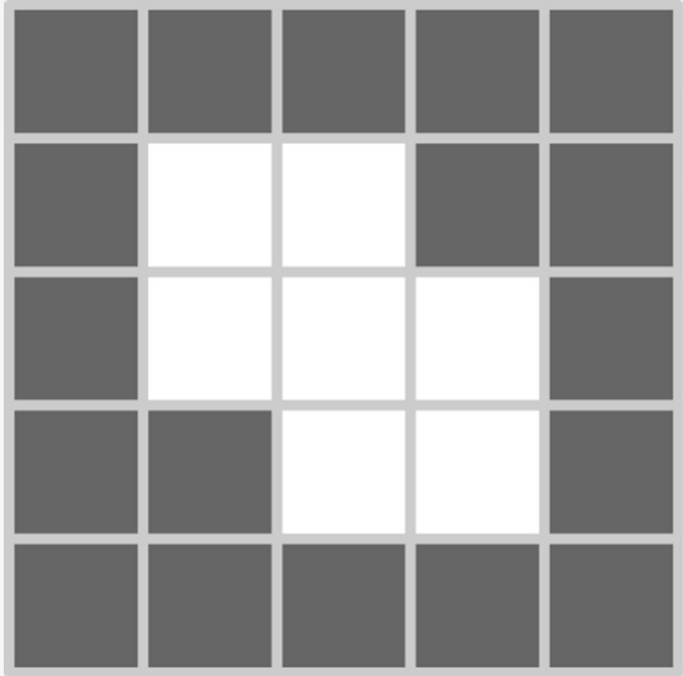
dd = distance transform

L = labeled image

Task

- Read in the image 'circles.png'
- Calculate the distance transform
- Try running the watershed algorithm then display the results

```
L = watershed(dd)
```



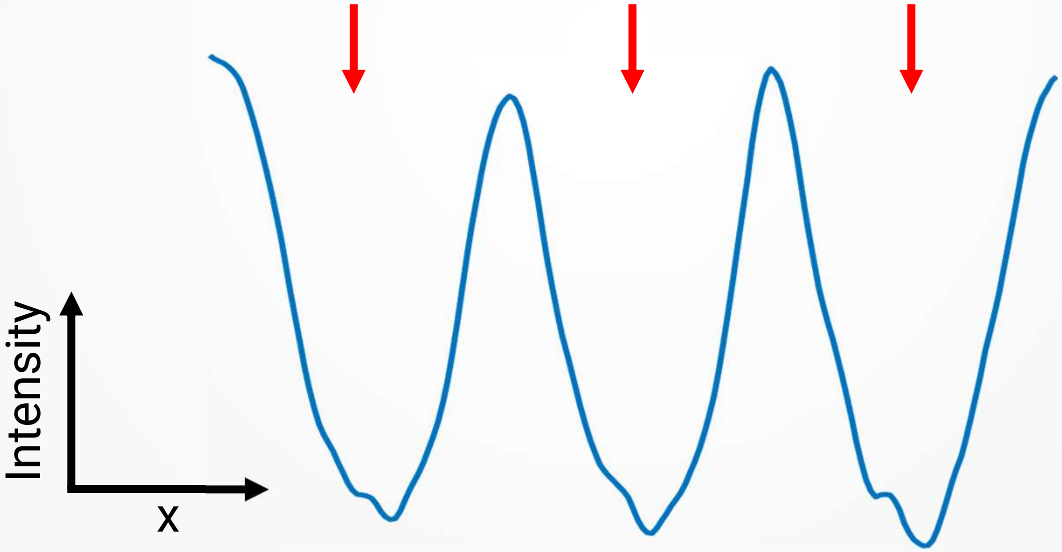
Input

1.41	1	1	1.41	2.24
1	0	0	1	1.41
1	0	0	0	1
1.41	1	0	0	1
2.24	1.41	1	1	1.41

Distance transform

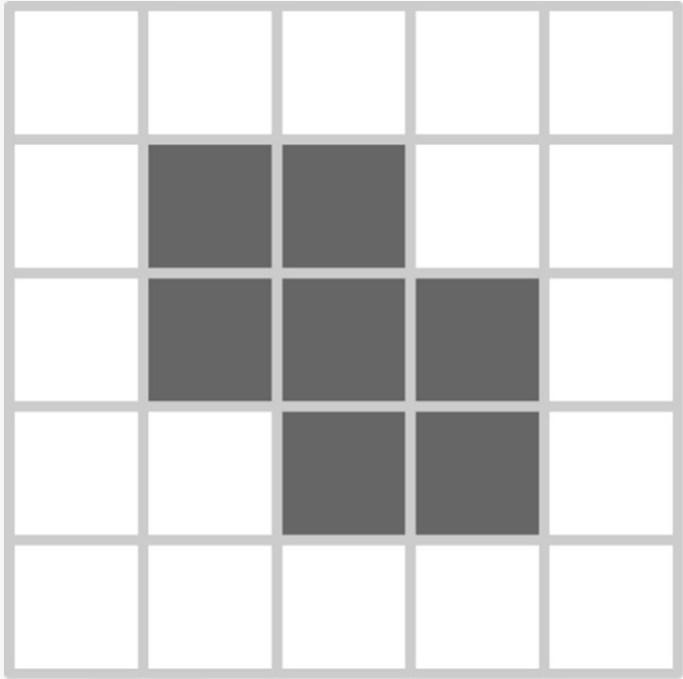
Requirements for the watershed algorithm

- Each object in the input image must be a "basin"
- The center of each object should be near the bottom of the basin



Not operator ~

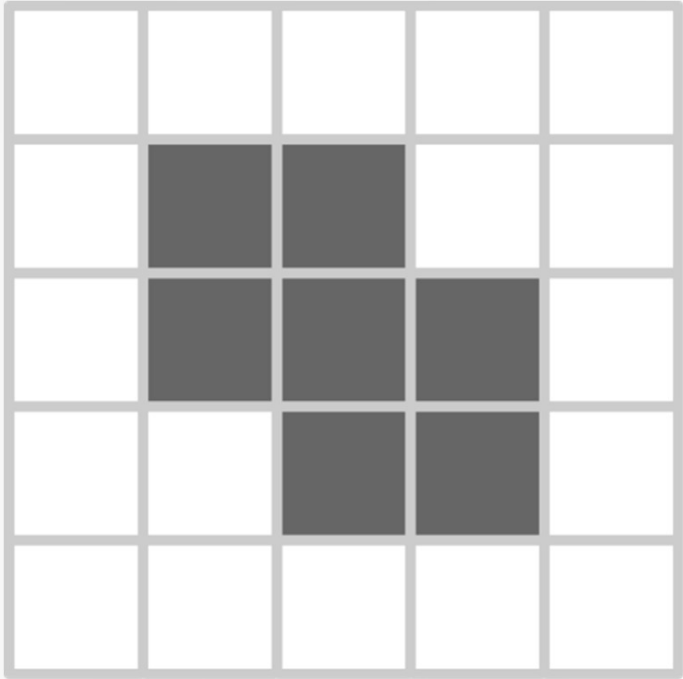
Converts true values to false and vice versa



Input

0	0	0	0	0
0	1	1	0	0
0	1	1.41	1	0
0	0	1	1	0
0	0	0	0	0

Distance transform



Input

0	0	0	0	0
0	-1	-1	0	0
0	-1	-1.41	-1	0
0	0	-1	-1	0
0	0	0	0	0

Distance transform

Summary of distance transform procedure

1. Invert the mask using the not operator (~)
2. Compute the distance transform of the inverted mask
3. Take the negative of the transform

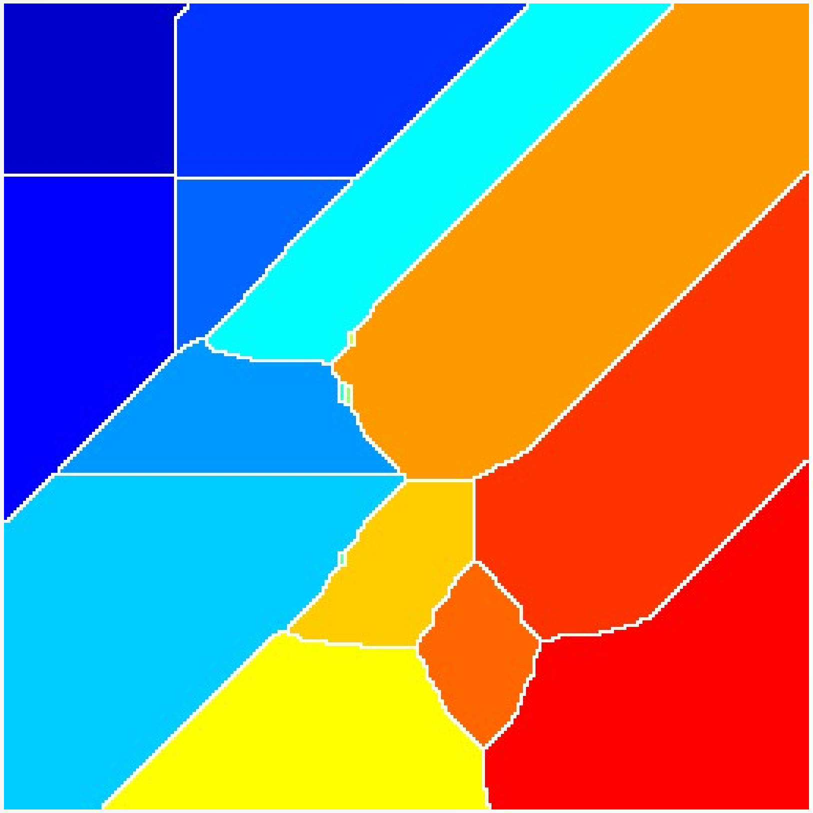
```
dd = -bwdist(~mask);
```

Task

- Read in the image 'circles.png'
- Calculate the distance transform using the inverted mask

```
dd = -bwdist(~mask);
```

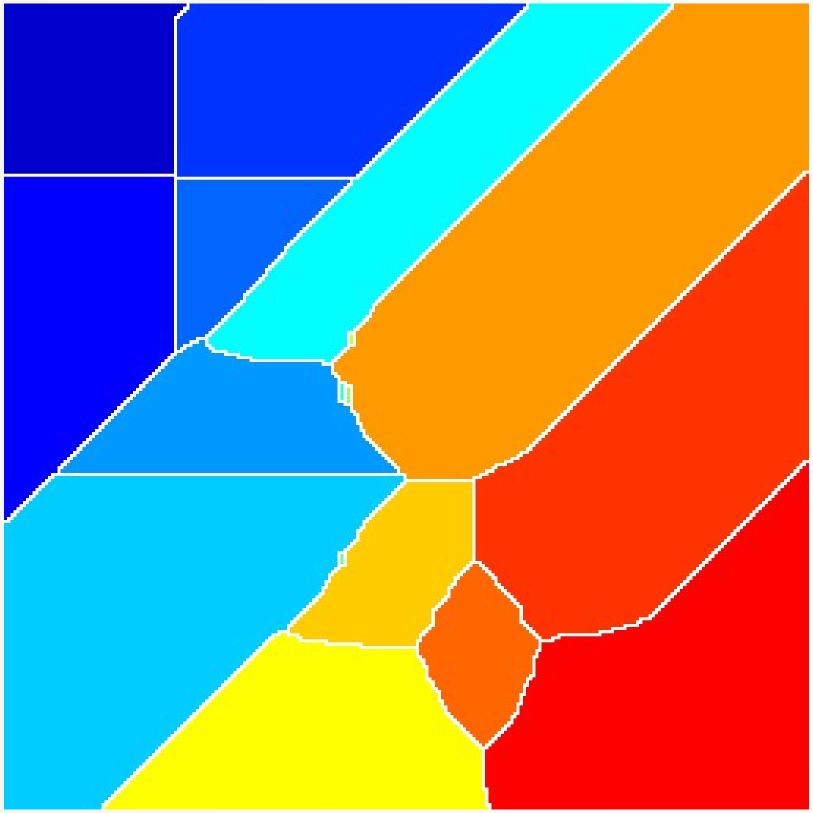
- Try running the watershed algorithm then display the results



Labeled image

You can get a more colorful display using `label2rgb`:

```
imshow(label2rgb(L))
```

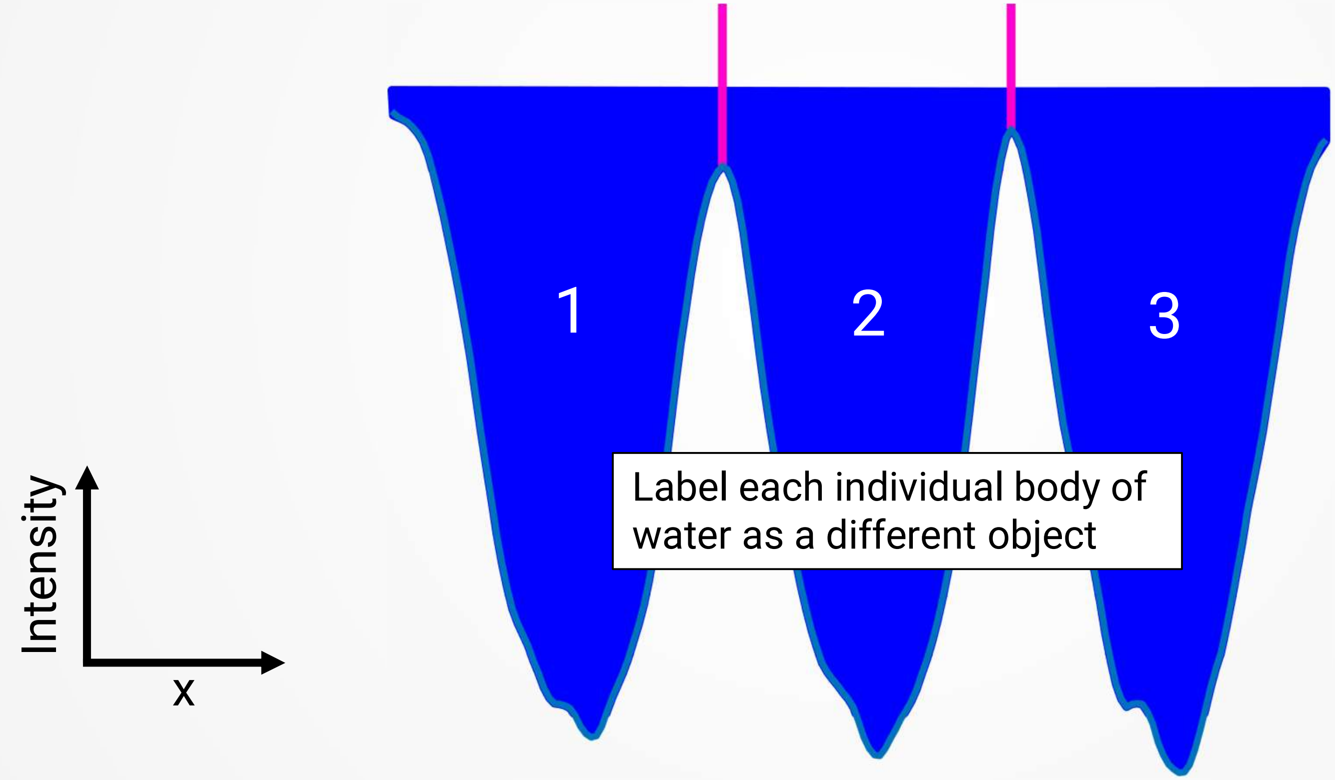


Labeled image

Each different color region indicates a watershed region or "basin"

The white lines are the "ridge lines" – indicates where the mask should be cut to separate objects

Mark locations where the bodies of water touch
(ridge lines = edge of an object)

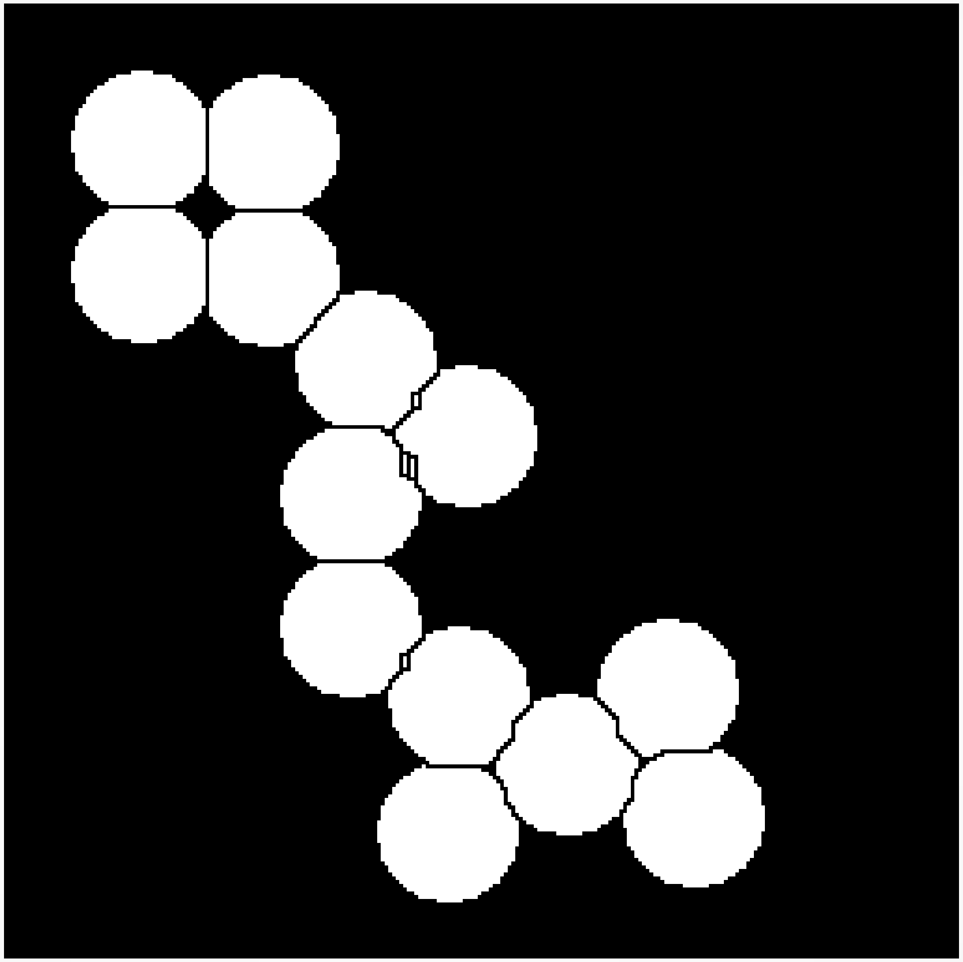


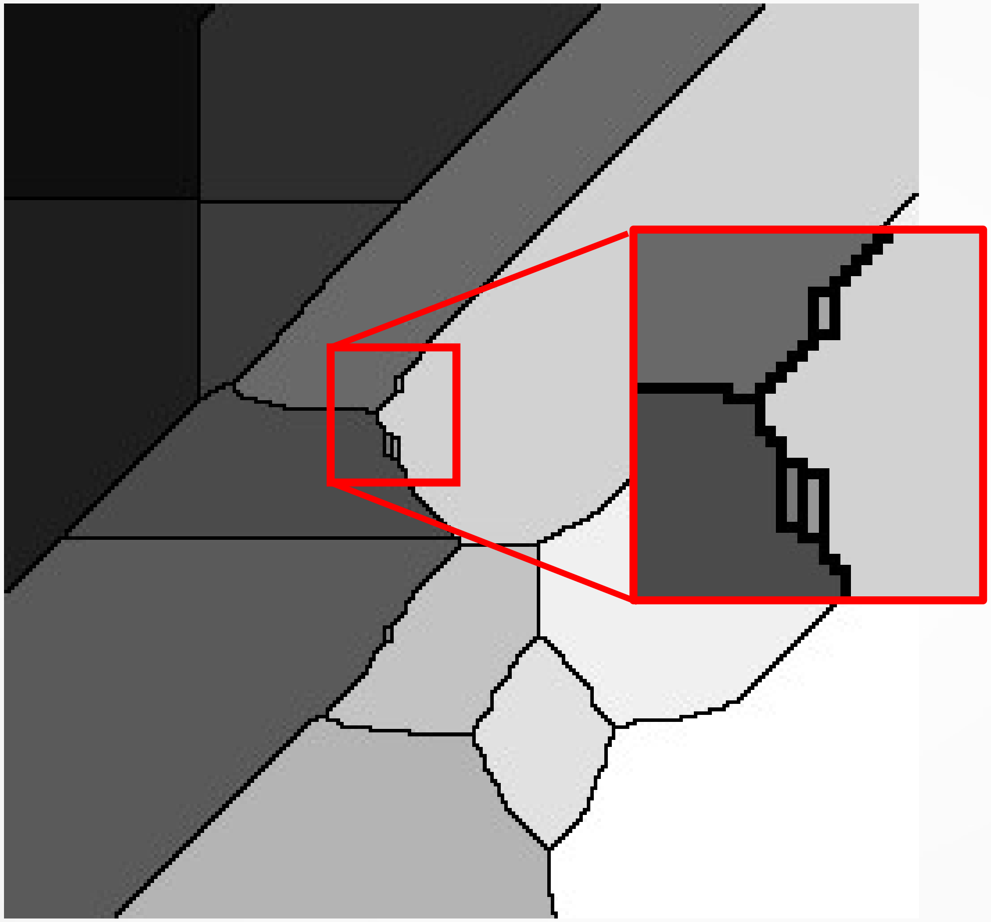
Label each individual body of water as a different object

Task

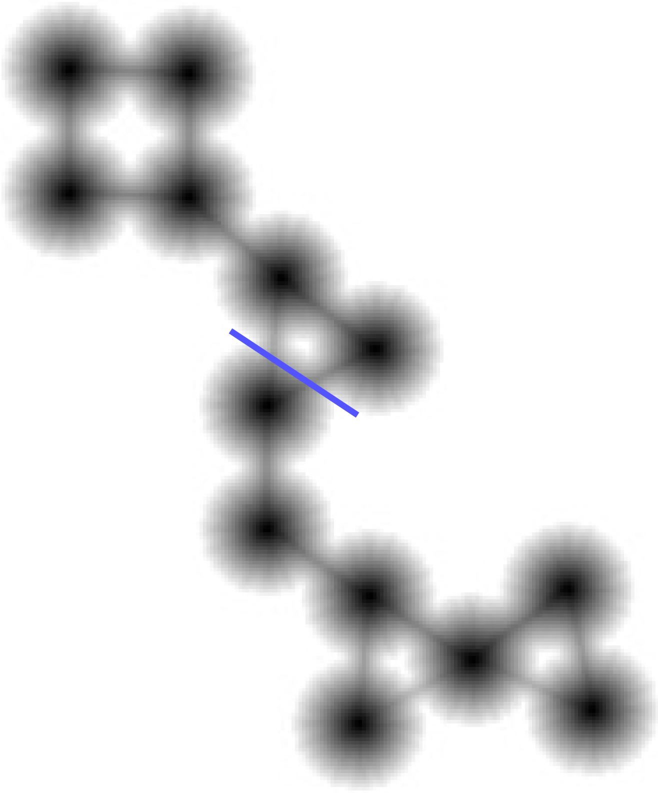
- Read in the image 'circles.png'
- Calculate the distance transform using the inverted mask
- Run the watershed algorithm
- Update the mask with the ridge lines

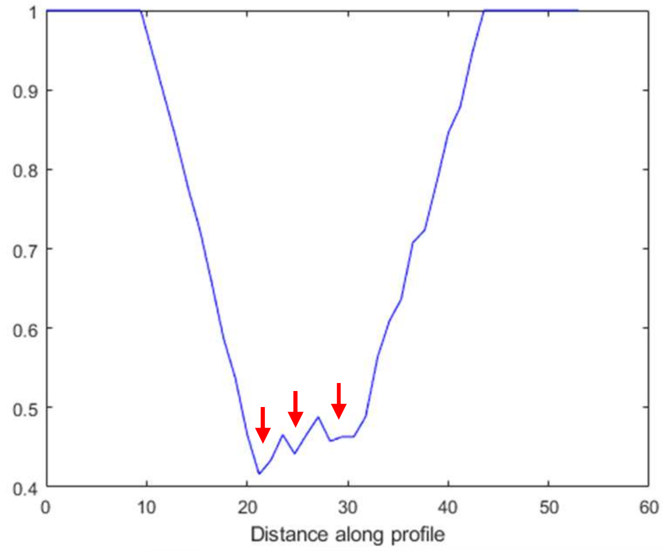
```
M(L == 0) = false;
```





Oversegmentation

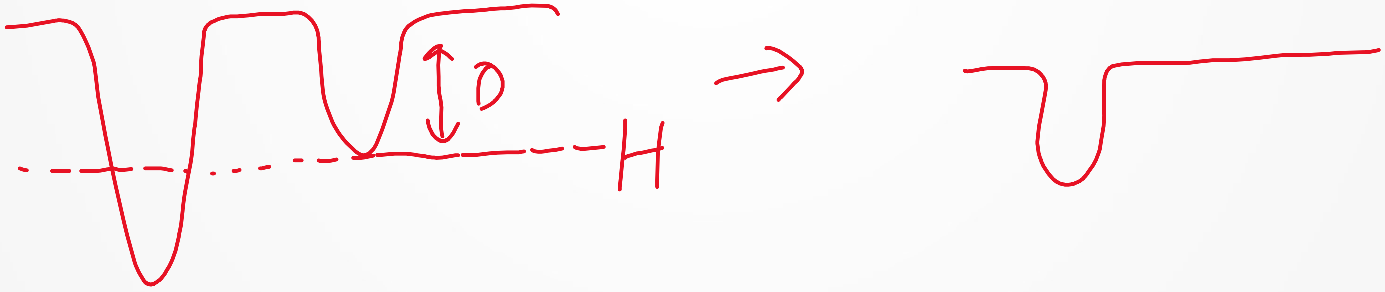




Local minima will create separate regions after watershedding

d2 = imhmin(dd, H)

Suppresses minima in dd whose depth is less than H



Example

```
>> A = [1 1 0.5 1 1 0.3 1 1];
>> imhmin(A, 0.5)

ans =
  Columns 1 through 7
    1.0000    1.0000    1.0000    1.0000
1.0000    0.8000    1.0000
  Column 8
    1.0000
```

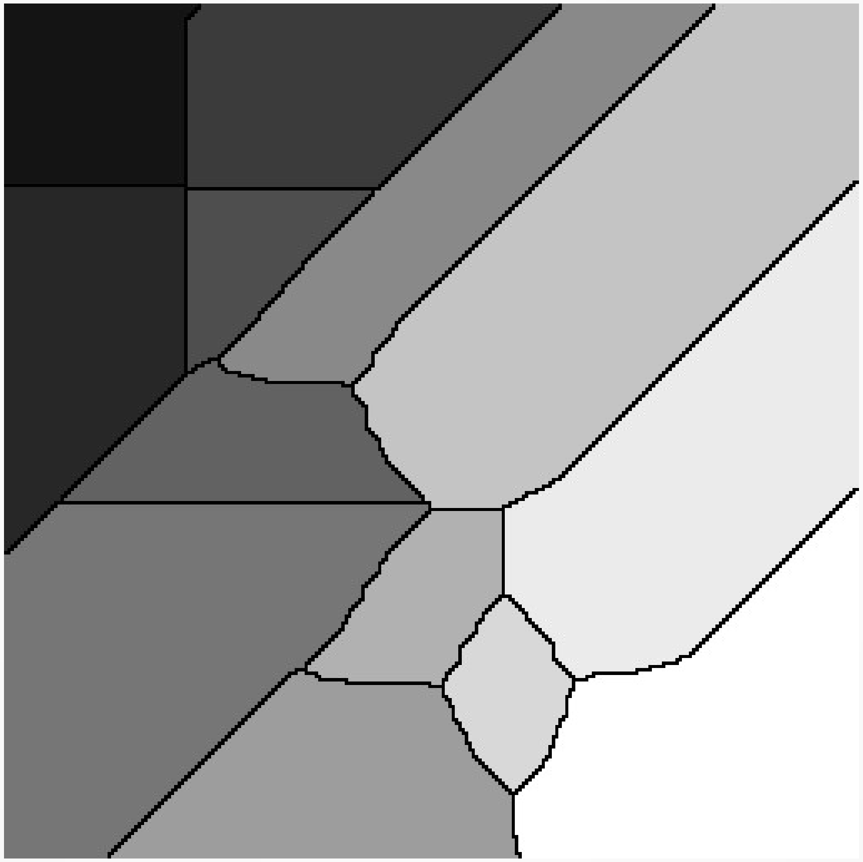
Task

- Read in the image 'circles.png'
- Calculate the distance transform using the inverted mask
- Remove shallow minima from the transform

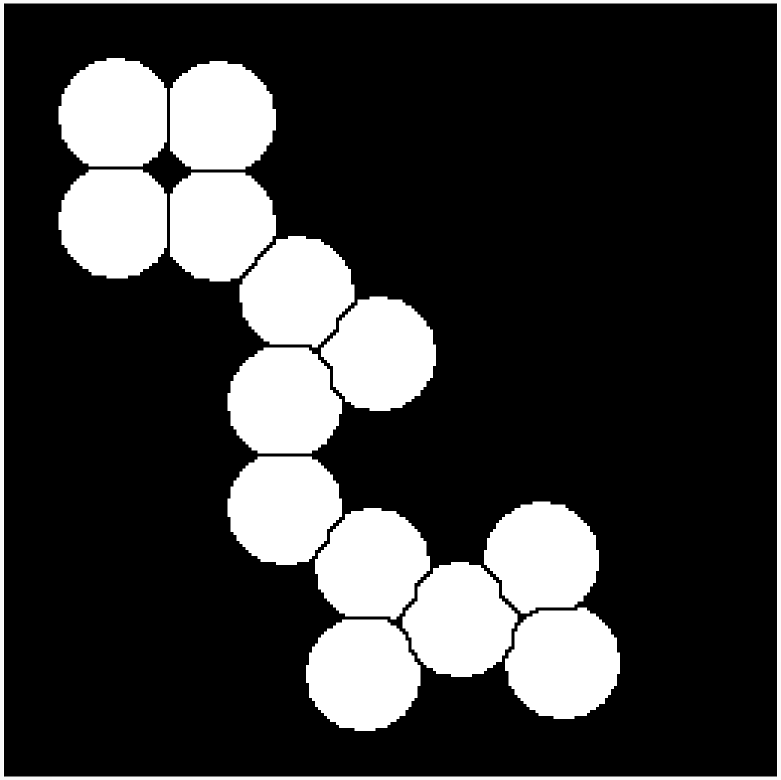
```
dd = imhmin(dd, H)
```

- Run the watershed algorithm
- Update the mask with the ridge lines

Corrected labels

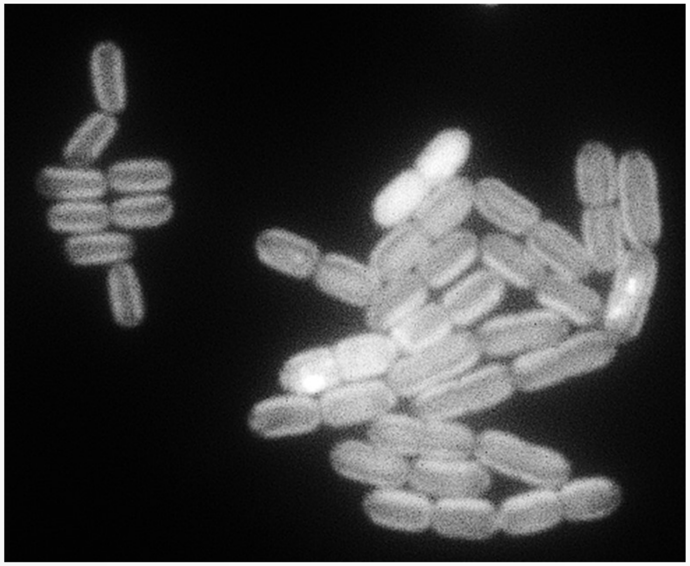


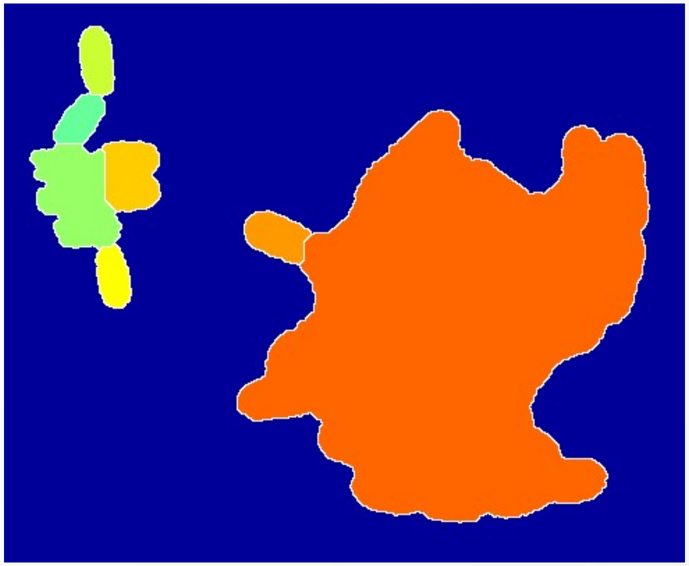
Final mask



Limitations and assumptions of the approach

- Using the distance transform to transform a mask into an image for watershedding works best if objects are circular





References on the marker-controlled watershed (Optional – will not be on tests)

- Yang, Li and Zhoi. IEEE Trans. Circuits and systems 53, 2405 (2006)
- Koyuncu et al. PLOS One 7:e48664 (2012)
- MATLAB Blog

Summary (in code form)

```
M = imread('circles.png');
dd = -bwdist(~M);
d2 = imhmin(dd, 2);
L = watershed(d2);
M(L == 0) = false;
```