MCDB/BCHM 4312 & 5312 – Quantitative Optical Imaging

Lecture 29:

Image analysis tips

Lecturer: Jian Wei Tay

Date: 2021-10-29

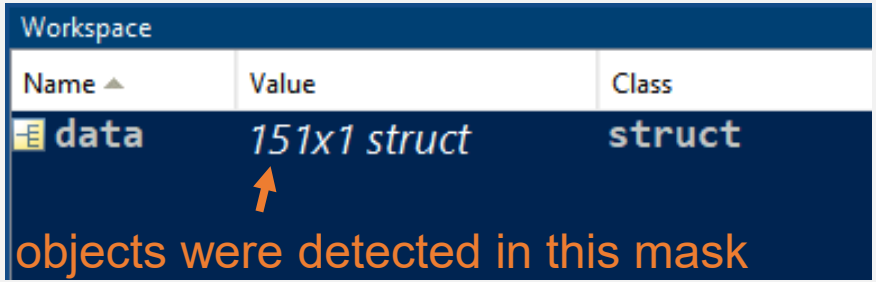
Some tips for image analysis

- Displaying numbers on an image to show `regionprops` labels
- Clarification on mask vs labels on `regionprops`
- Refining the mask after running watershed

regionprops

- We've seen how to use `regionprops` to measure object properties (e.g. size, area, etc).

- The output of `regionprops` is a multi-element struct, where each element corresponds to a different object:

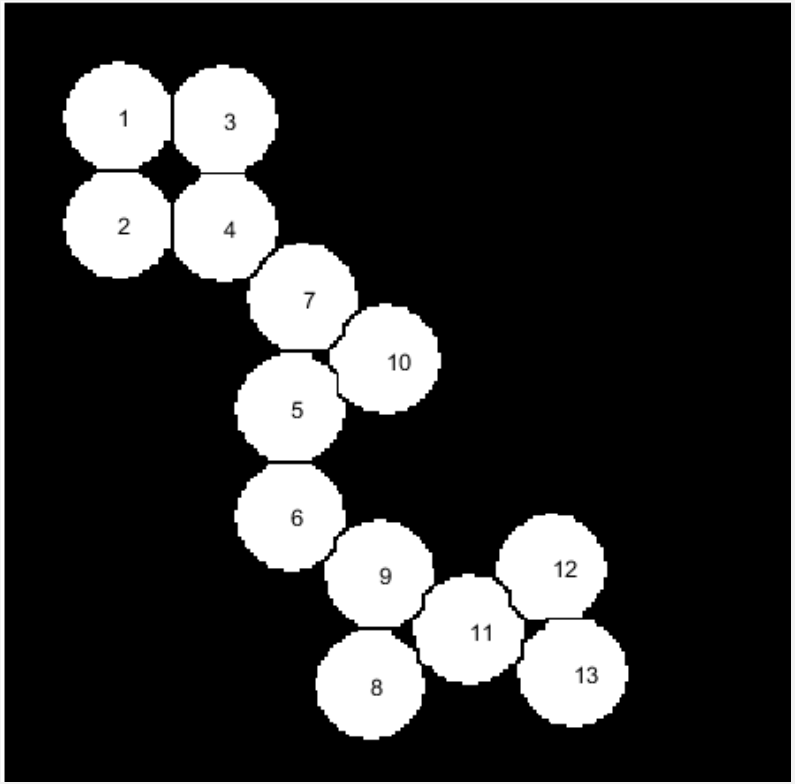


Labeling an image to show the `regionprops` order

- The question is which element in the output of `regionprops` corresponds to which object in the image?

- We can plot numbers using the centroid to find out

Example



- Making a plot like this is useful when trying to identify a single object
- `data(11).Area` will return the area of the circle labeled 11 in the image on the left

Example

- Using the output of the watershed algorithm on `circles.png`

```
data = regionprops(mask, 'Centroid');
imshow(mask)
hold on
for ii = 1:numel(data)
    text(data(ii).Centroid(1), data(ii).Centroid(2),
int2str(ii))
end
hold off
```

Note: `numel` returns the number of elements in a variable (see Lecture 16)

Some notes on the code

- The centroid data is returned as a 1x2 vector containing the coordinates [x, y]

- So `data(ii).Centroid(1)` returns the x-coordinate of object `ii`

- The function `int2str(M)` rounds the elements of `M` to integer, then converts the number into a string

How `regionprops` orders objects in a mask

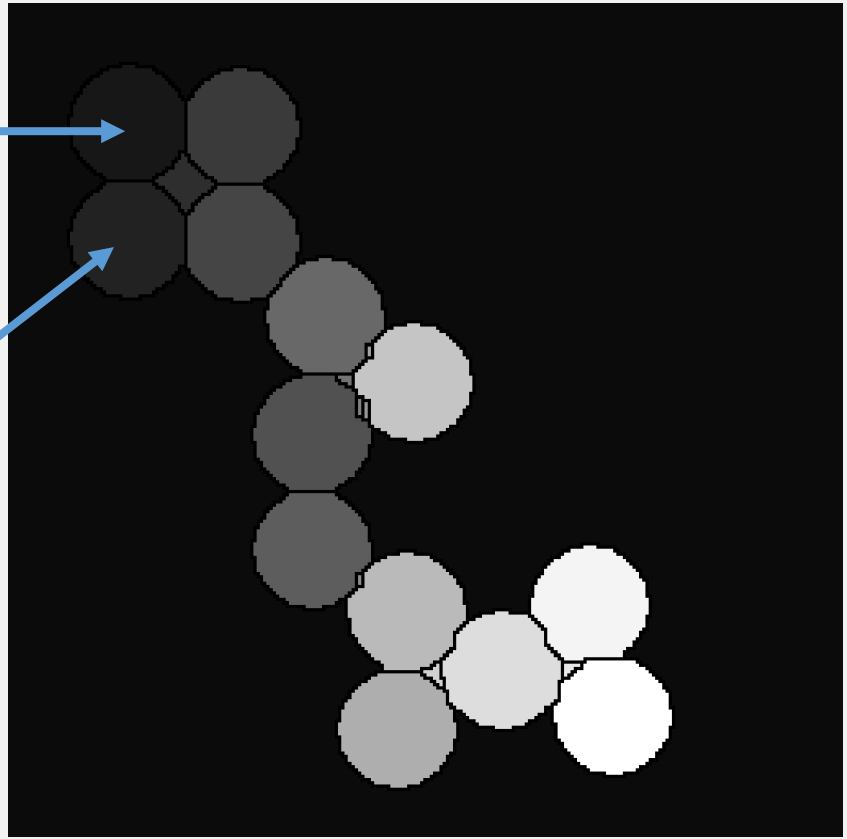
- A mask is a binary image (i.e., you get this by doing intensity thresholding `I > 2000`, or by using `imbinarize`)
- When a mask is used as the first input to `regionprops`, the algorithm looks for connected true regions in the mask
- The search pattern that `regionprops` uses is roughly from top left to bottom right of the image

How `regionprops` orders objects in a labeled image

- In a labeled image, the pixels belonging to the same objects are labeled with the same number
- You get a labeled image from the watershed algorithm

Every pixel in this circle has a value of 2

Every pixel in this circle has a value of 3



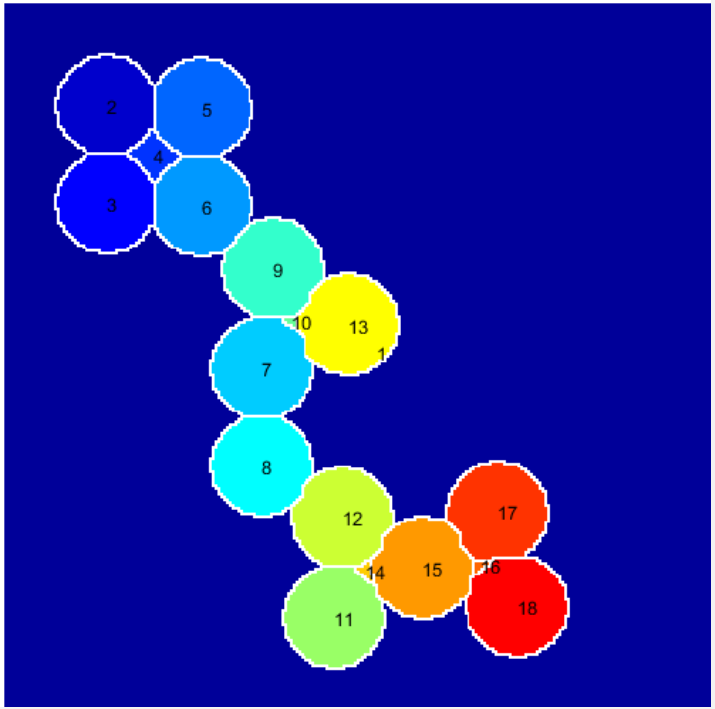
The watershed algorithm will also label the background since it thinks of it as a basin. In this example, the background has a value of 1. Note that the background is not always 1, so you might want to plot numbers on the image to see what it ends up as.

How `regionprops` orders objects in a labeled image

- If given a labeled image, regionprops will return objects based on its label

- So objects labeled with 1 in the input will be in element 1, objects labeled 2 will be in element 2 etc.

Plotting the numbers as shown earlier




```
clearvars
clc

mask = imread('circles.png');
dd = -bwdist(~mask);
dd(~mask) = -Inf;
dd = imhmin(dd, 2);

L = watershed(dd);

data = regionprops(L, 'Centroid');

imshow(label2rgb(L), [])
hold on
for ii = 1:numel(data)
    text(data(ii).Centroid(1), data(ii).Centroid(2),
int2str(ii))
end
hold off
```

Removing the background element

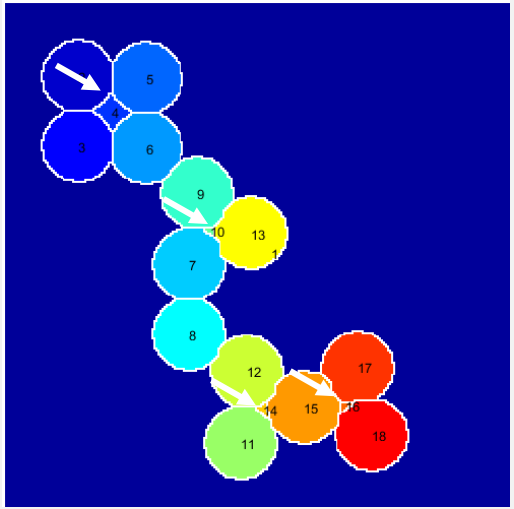
- In the image on the previous slide, you can see that the background is labeled as object 1

- To avoid including this in further analysis, you can use indexing to delete it, e.g.

```
data(1) = []
```

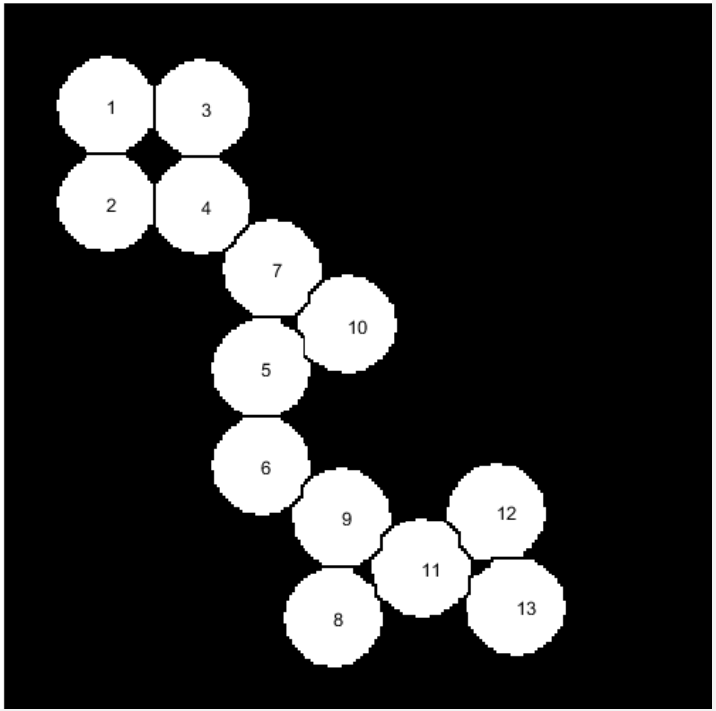
Another way to remove background/small objects

- Another problem is that any separated background regions also get labeled by the watershed algorithm (see regions 4, 10, 14 and 16)



Another way to remove background/small objects

- A simpler option might be to just update the original mask (see previous lecture on logical indexing)




```
clearvars
clc

mask = imread('circles.png');
dd = -bwdist(~mask);
dd(~mask) = -Inf;
dd = imhmin(dd, 2);

L = watershed(dd);

mask(L == 0) = false;

data = regionprops(mask, 'Centroid');

imshow(mask, [])
hold on
for ii = 1:numel(data)
    text(data(ii).Centroid(1), data(ii).Centroid(2),
int2str(ii))
end
hold off
```