MCDB/BCHM 4312 & 5312 – Quantitative Optical Imaging

Lecture 16:

Identifying objects by intensity thresholding

Lecturer: Jian Wei Tay

Date: 29 September 2021

Learning objectives

- The logical data type
- Identifying objects with intensity thresholds
- Measuring object data using `regionprops`

The logical data type

- The logical data type contains binary values

- Elements can be either `true` (1) or `false` (0)

Declaring a variable with logical data

- `true` or `false`

```
>> A = true
>> A = true(1, 10)
```

`true`, `false`, and `logical` are functions

- Using `logical` to convert an existing matrix

```
>> B = [0 1 2 3 4]
>> logical(B)
```

Comparing matrix values

- To identify bright objects in an image, we can compare the value of each element in the image data to a threshold

```
I > threshold
```

Comparison operators

Operator	Description
==	Exactly equal
~=	Not equal to
>	**Greater than**
>=	Greater than or equal to
<	**Less than**
<=	Less than or equal to
~	Not (inverts `logical` value)

Practice

- What is the output of the command below?

```
L = 10 > 20
```

A. `true (1)`
B. `false (0)`

Practice

- What is the output of the command below?

```
L = [1 2 3 4 5] > 3
```

A. `true`

B. `false`

C. `[0 0 0 1 1]`

D. `[1 1 0 0 0]`

Practice

- What is the output of the command below?

```
L = [1 2 3 4 5] <= 2
```

A. true
B. [1 1 0 0 0]
C. [0 0 0 1 1]
D. [1 0 0 0 0]

The not operator ~

- The ~ operator inverts the values of a logical matrix

- Example:

```
>> L = [1 0 1 1 0];
>> ~L
ans =

  1×5 logical array

   0   1   0   0   1
```

Practice

- What is the output of the command below?

```
L = ~([1 2 3 4 5] <= 2)
```

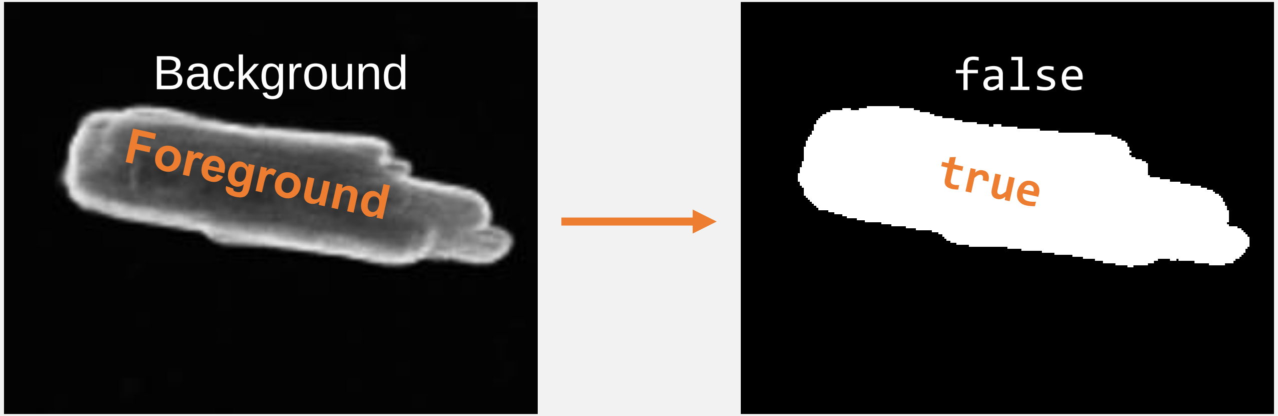
A. [1 1 0 0 0]
B. [1 0 0 0 0]
C. [0 0 1 1 1]
D. [1 0 1 1 0]

Questions?

Application: Intensity thresholding

- Goal is to identify bright objects in a fluorescent microscope image by labeling each foreground/object pixel as true and everything else as false

Image (`uint16`) Mask (`logical`)



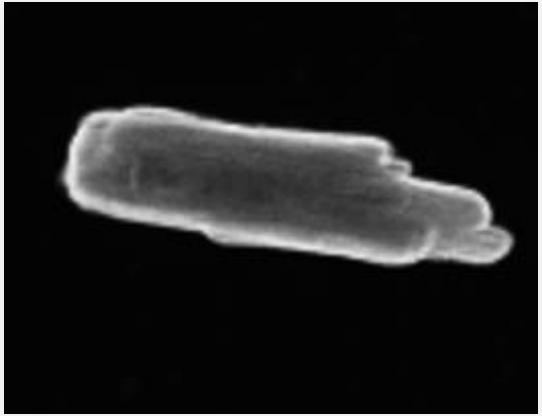
The labeled image is called a <u>mask</u>

Intensity thresholding

- Basic idea is to use a comparison operator to set a threshold value

```
I > threshold
```

- All pixels greater than the threshold will be labeled true and identified as an object



Practice

- Read in the image `l11_moreCardiomyocytes.tif`
- Make a mask that identifies the cells in the image using intensity thresholding

Example

```
I = imread('l11_moreCardiomyocytes.tif’);

mask = I > 1000;
```

To visualize the mask

- Use `bwperim` to find the perimeter of the mask

```
P = bwperim(mask);
```

- Use `imshowpair` to display the perimeter on the original image

```
imshowpair(P, I)
```

Image in first argument displayed in green, second argument in magenta

Practice

- Update your code to use `bwperim` and `imshowpair` to display the mask

Example

```
I = imread('l11_moreCardiomyocytes.tif’);

mask = I > 1000;

P = bwperim(mask);

imshowpair(P, I);
```

Finding the correct threshold

- To find the "best" threshold:
  - Test different values ("trial-and-error")
  - Use the data tips tool to estimate
  - Take a line profile using `improfile`
  - Look at intensity histogram (next week)

How do you know if mask is accurate?

- Check manually by eye (e.g., using `bwperim` and `imshowpair`)
- Compare object size with known values from other measurements
- Manually label objects by hand (ground truth) and compare with labels generated by thresholding

Measuring cell properties with regionprops

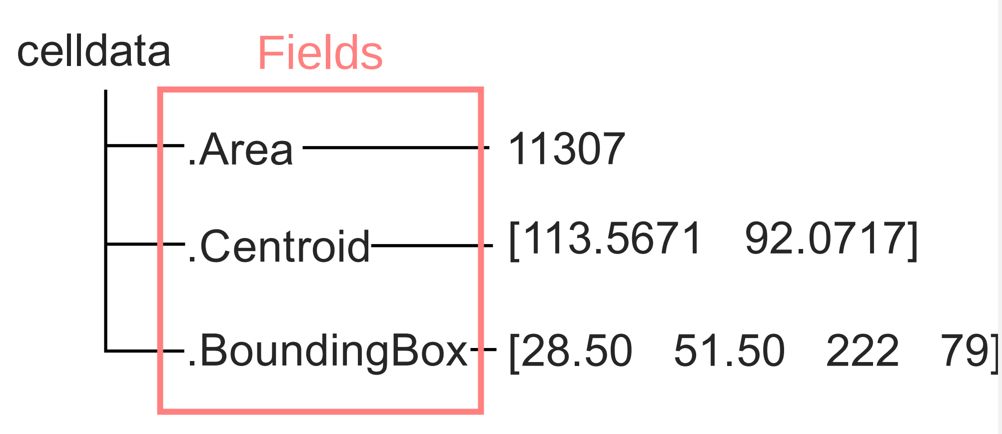
- Once you have the mask, you can measure data using the function `regionprops`

```
celldata = regionprops(mask)
```

- The output of `regionprops` is a `struct` or structured array

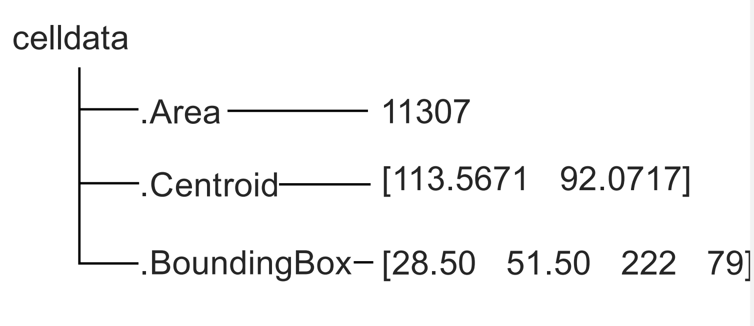
Structured Arrays (`struct`)

- `struct` is a MATLAB data type
- Data is stored in named **fields**



Unlike a matrix or array, a `struct` can hold different data types of different sizes in each field

Accessing data from a struct



Remember that length values are in pixels. You must multiply with the image pixel size to convert to real units.

- Use “dot notation” to access data in a struct

```
variable_name.field_name
```

- Example:

```
>> celldata(1).Area
```

Field names are case-sensitive. With `regionprops`, they all start with an uppercase letter.

Measure different properties by specifying in input

```
celldata = regionprops(mask, ...
      'Area', 'Centroid', 'MajorAxisLength')
```

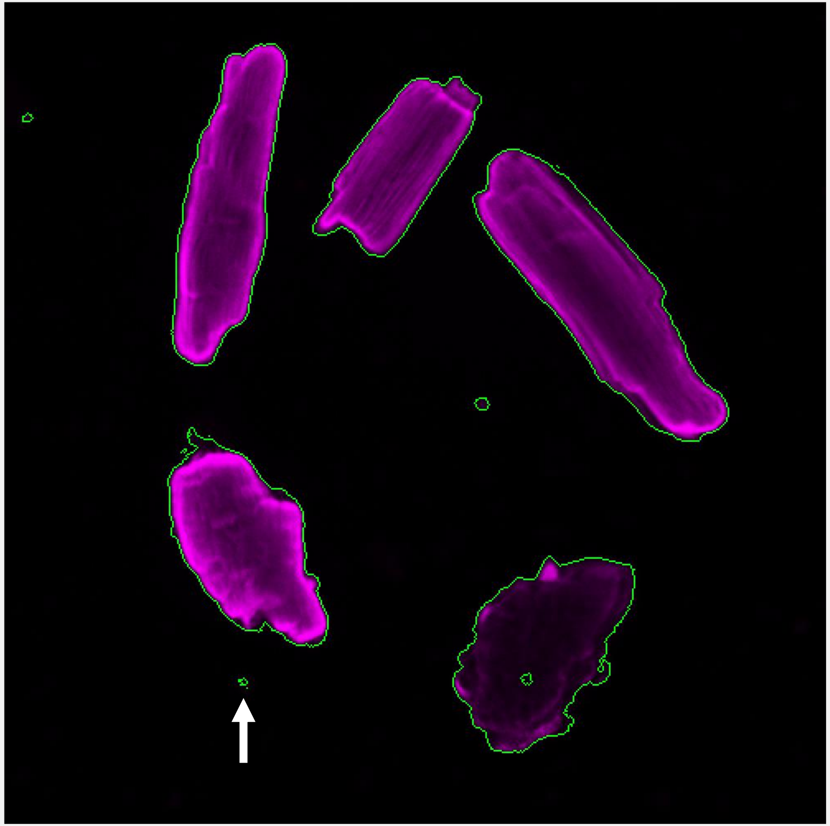
Read the documentation to understand what these properties are

Measuring data with multiple objects

- Each element of the struct output of regionprops refers to a different object
- Every element in the struct will have the same fields
- Use indexing to access data from a specific object:

```
variable_name(index).field_name
```

How many objects were detected?



`regionprops` identifies objects by looking for connected region of trues in the mask.

Need to be careful – a single unconnected pixel (arrow) will be a new object

How many objects were detected?

`regionprops` identifies objects by looking for connected region of trues in the mask.

Need to be careful – a single unconnected pixel (arrow) will be a new object

Removing small objects using `bwareaopen`

- The function `bwareaopen` will remove objects smaller than a threshold size from the mask

- Example:

```
>> mask = bwareaopen(mask, 100)
```

Removes objects < 100 pixels in area from the mask

Number of detected objects = number of elements in struct

```
numCells = numel(celldata)
```

The function `numel` returns the **<u>num</u>**ber of **<u>el</u>**ements in a variable

Analyzing data

- To analyze data from the output of regionprops, we must first concatenate (join) elements together

- Use the function `cat`

```
>> areas = cat(1, celldata.Area)
```

The first argument of cat specifies the dimension to join. In this example, DIM = 1 means to join the data in rows.

Analyzing data

- Once the data is in a matrix, you can use indexing and functions as usual

Practice

- Modify your code to concatenate the Area for each cell into a new matrix
- Compute the mean area of the cells in the image

Example

```
I = imread('l11_moreCardiomyocytes.tif');
mask = I > 1000;

%Remove small objects
mask = bwareaopen(mask, 300);

%Measure cell data
celldata = regionprops(mask);

%Measure mean cell area
areas = cat(1, celldata.Area);  %Concatenate (join) areas into a single matrix
mean(areas)  %Compute the mean area
```