Lecture 30:

Analyzing multichannel and time-lapse images

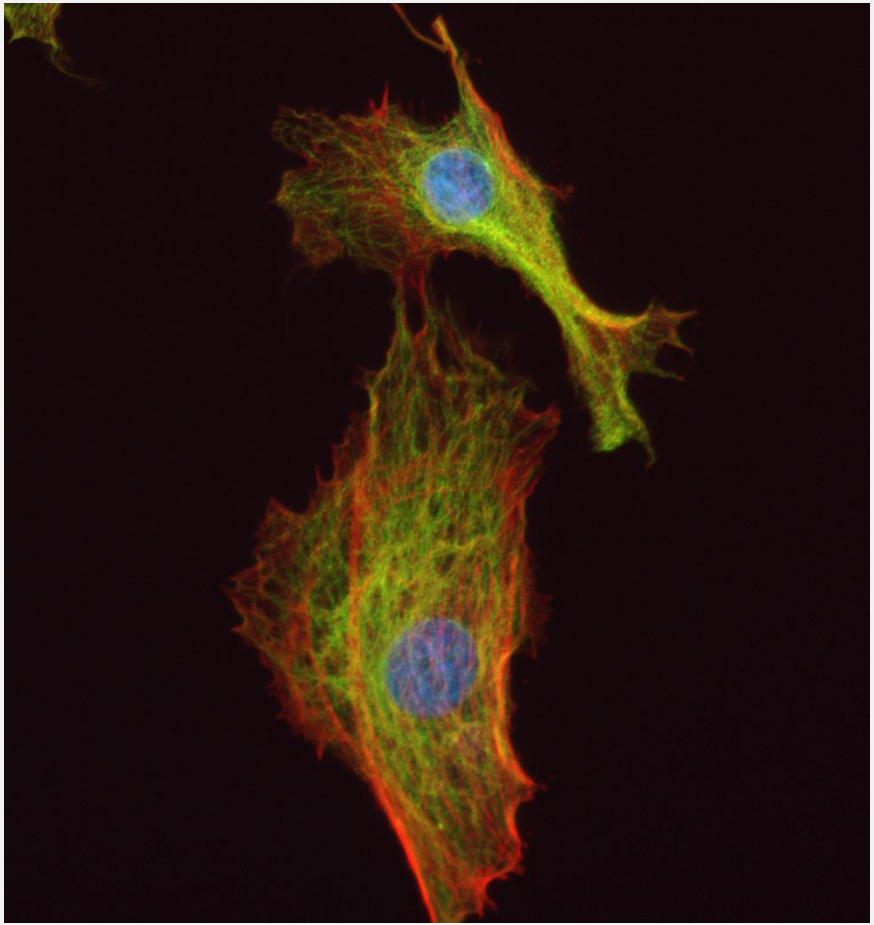
Lecturer: Jian Wei Tay

Date: 5 November 2021

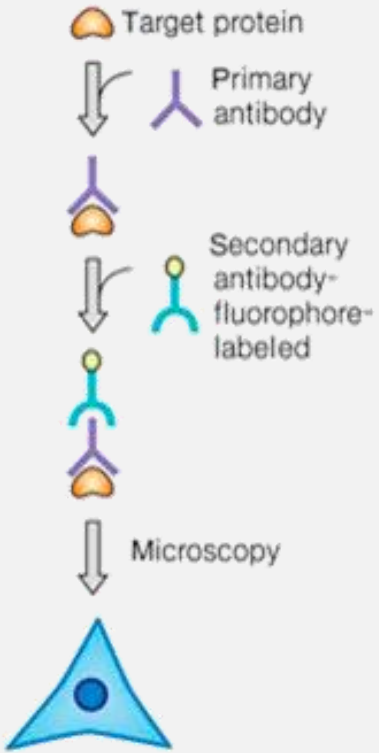
Learning objectives

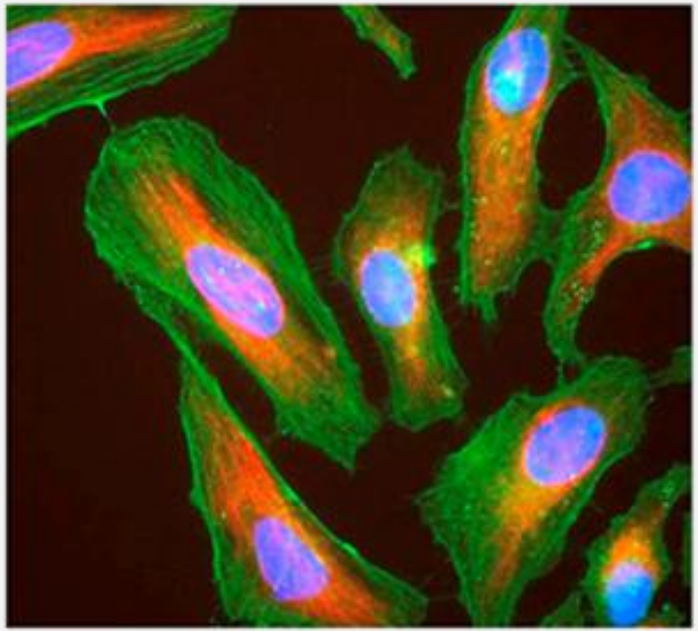
- How to analyze multichannel images
- Considerations for image analysis
- Time-lapse images
- Analyzing time-lapse images using for loops
- Strategies to analyze large images

Multichannel images

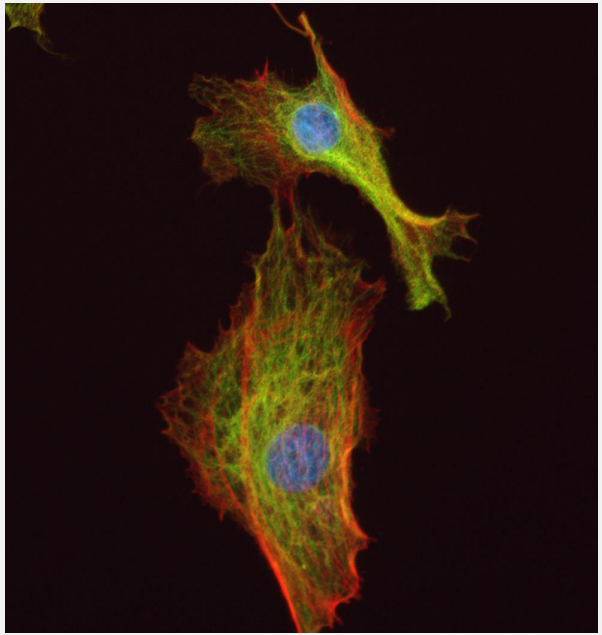


Fluorescence can be used to label cell structures

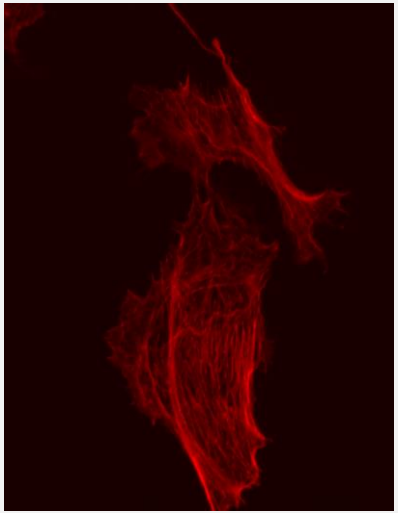




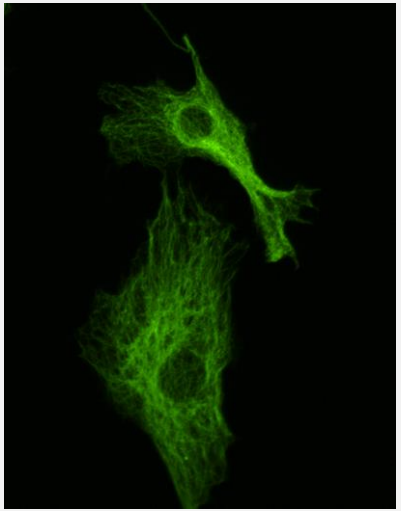
These channels are images captured in series



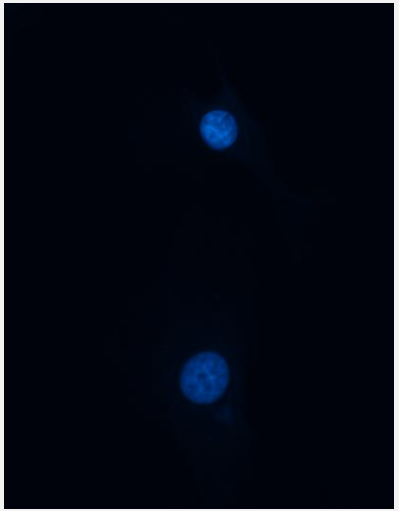
Multichannel image



Actin

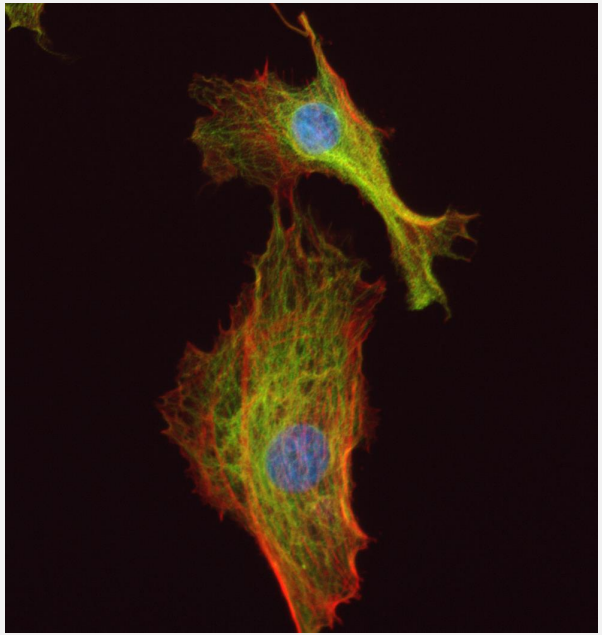


Microtubules

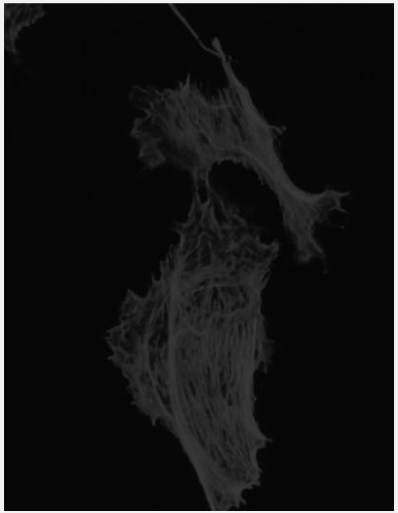


Nuclei

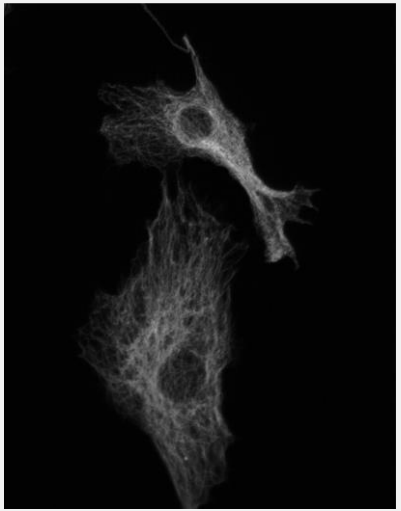
Images are actually grayscale (color is added)



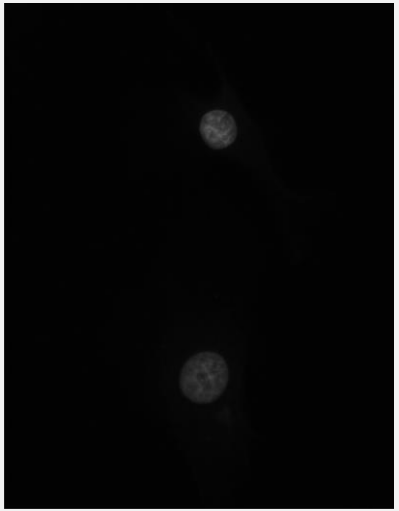
Multichannel image



Actin

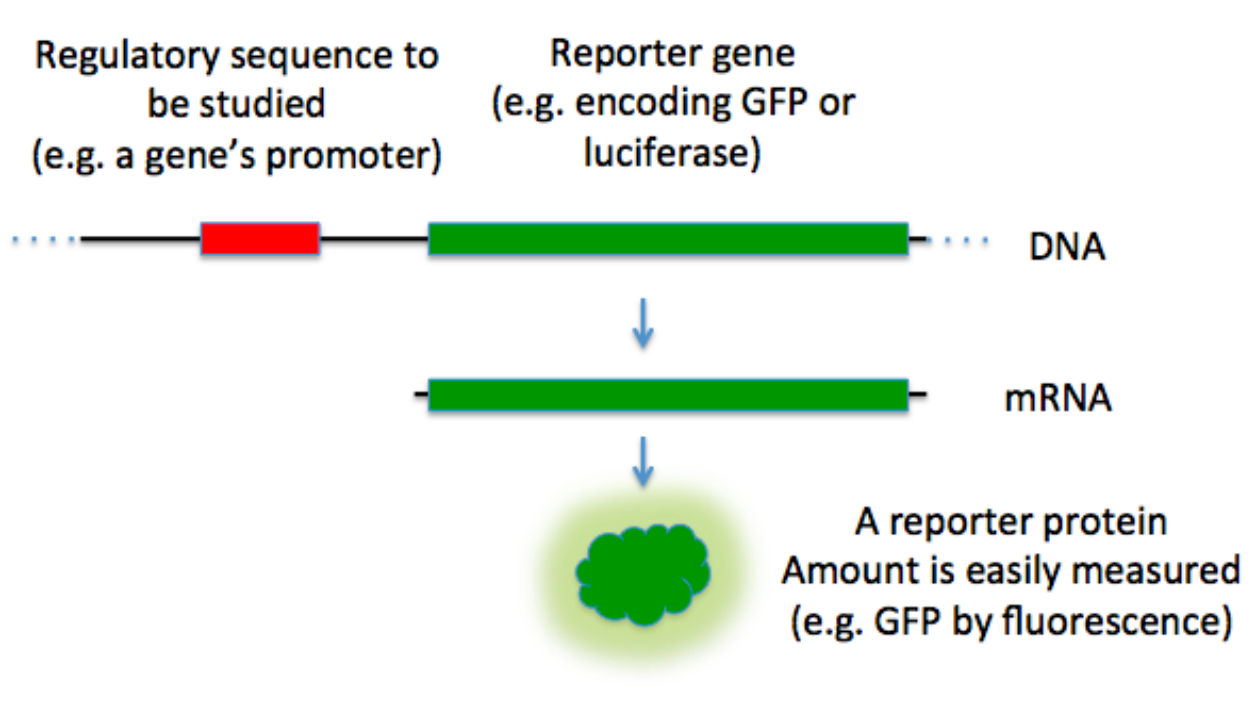


Microtubules

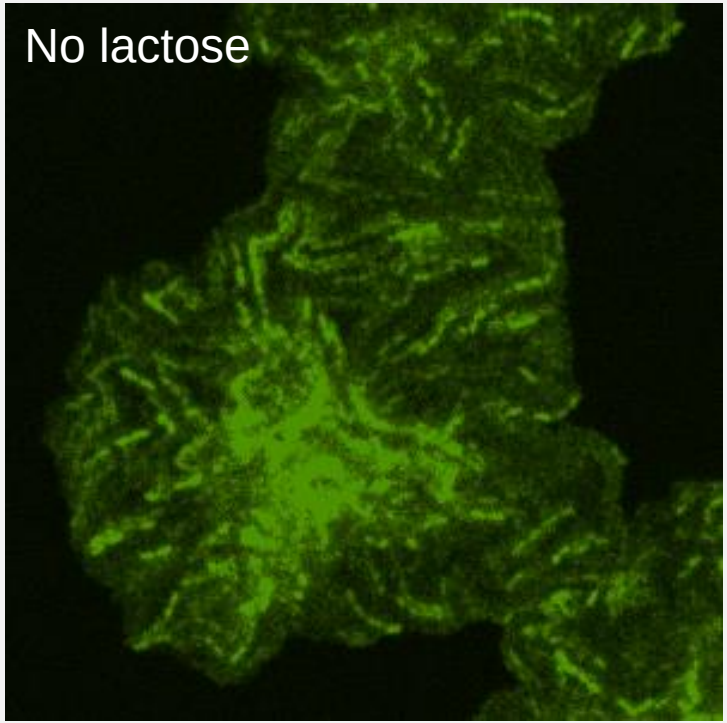


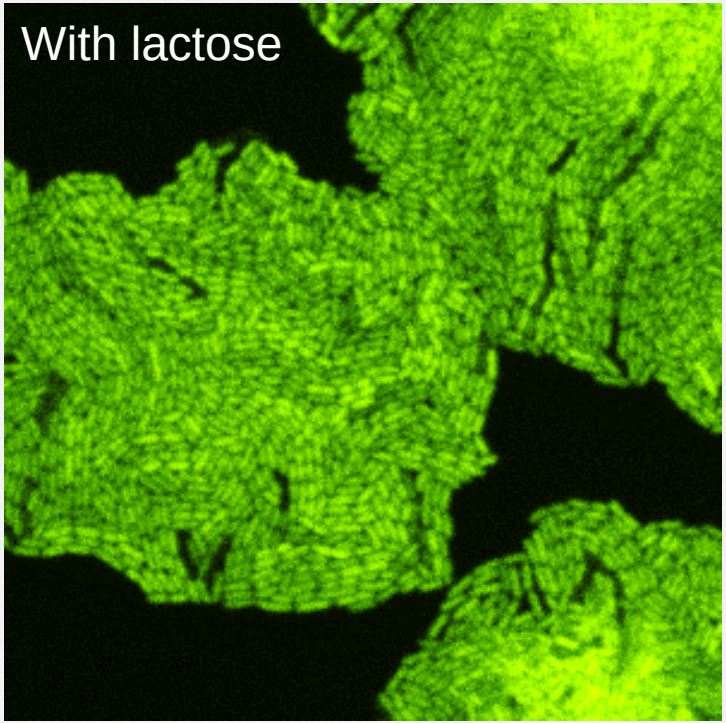
Nuclei

Fluorescent reporters can be used to quantify activity



lacZ (beta-galactosidase) expression reporter in E. coli

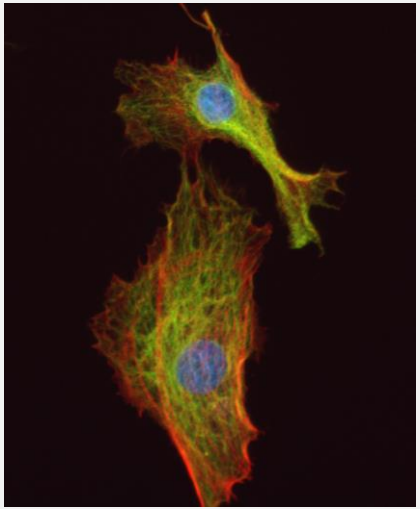




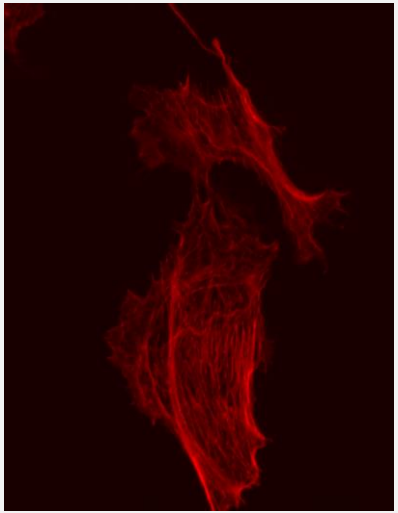
Tay & Kralj

Think about image analysis when planning an experiment

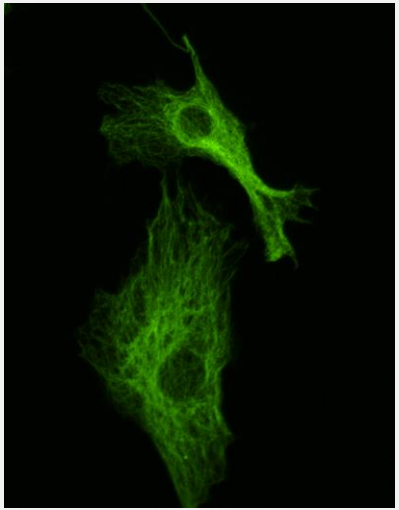
Which channel would you use to count these cells?



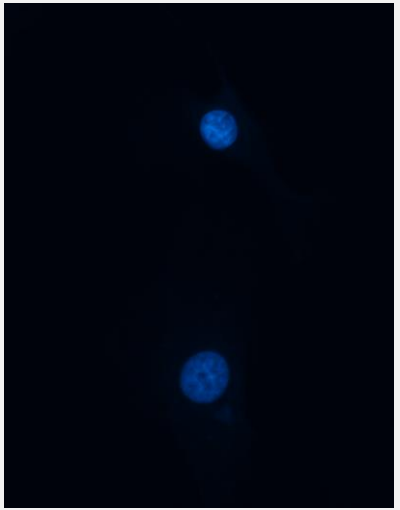
Multichannel image
(A)



Actin
(B)

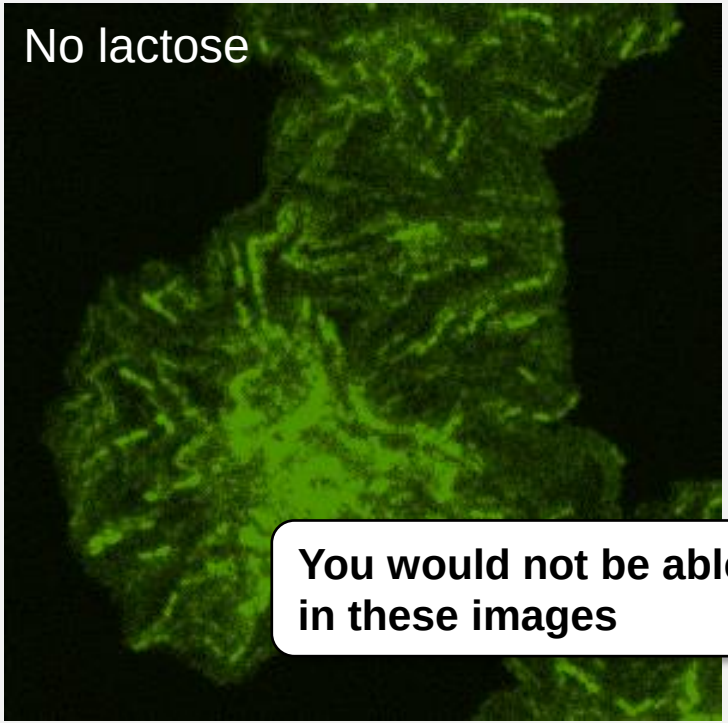


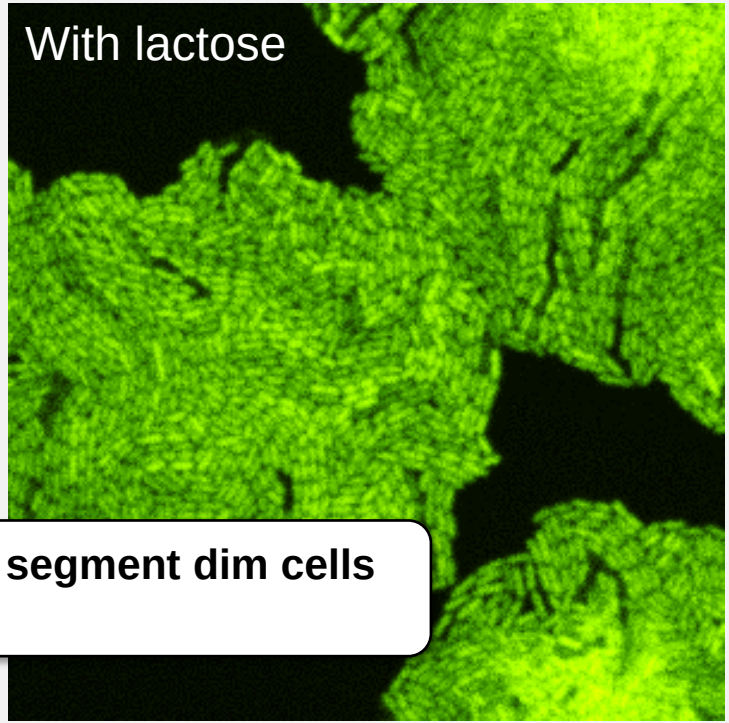
Microtubules
(C)



Nuclei
(D)

How would you quantify the fluorescence intensity in these cells?

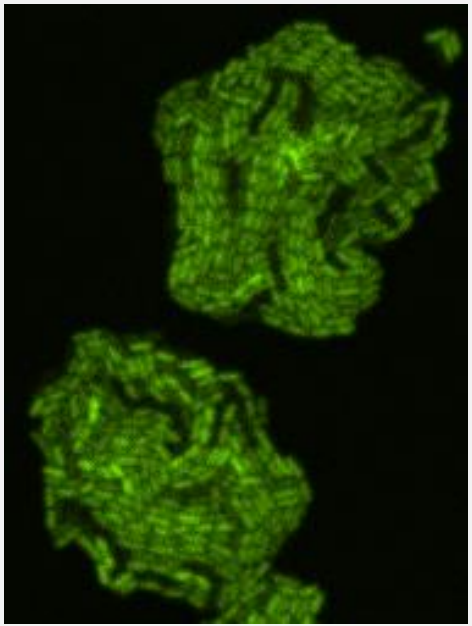




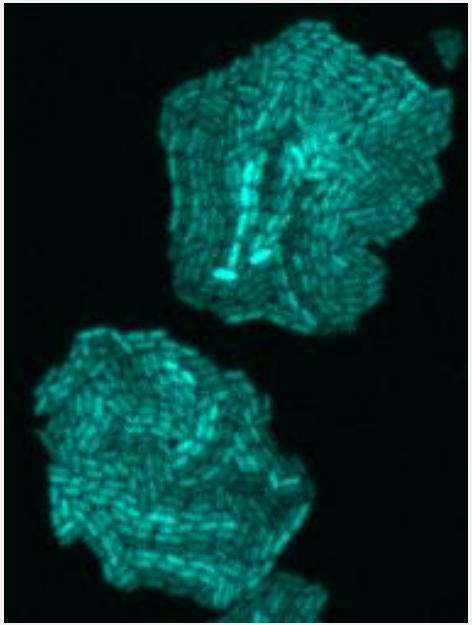
You would not be able to segment dim cells in these images

Tay & Kralj

Might need to include a second fluorophore to mark the cells



GFP – Gene expression reporter



mCherry – Cell marker

Note: Segment/make the mask with the cell marker channel, then use it to measure the intensity of the reporter image

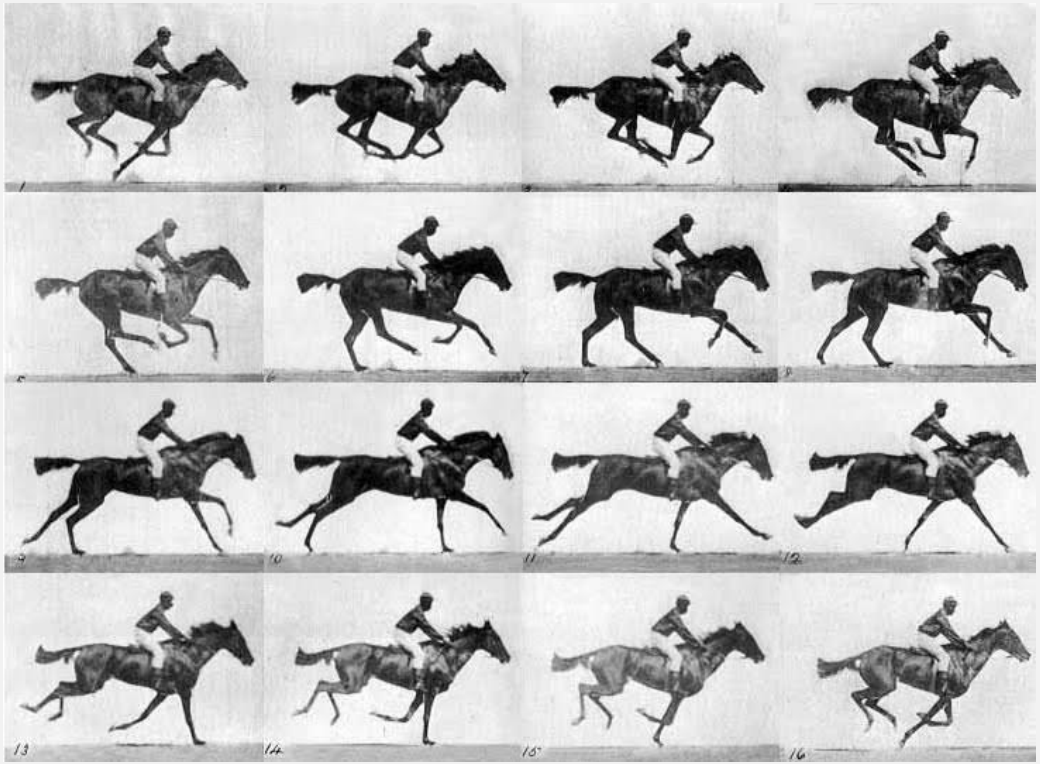
Pseudo-code

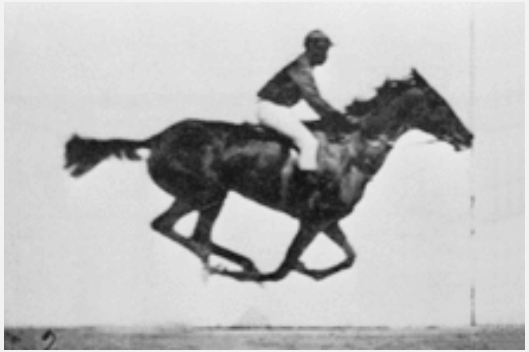
```
mask = imbinarize(cellMarkerImage);

data = regionprops(mask, reporterImage, ...
           'meanIntensity');
```

Note: You will analyze a similar system in the homework this week

Questions?

Time-lapse: Images are captured in series



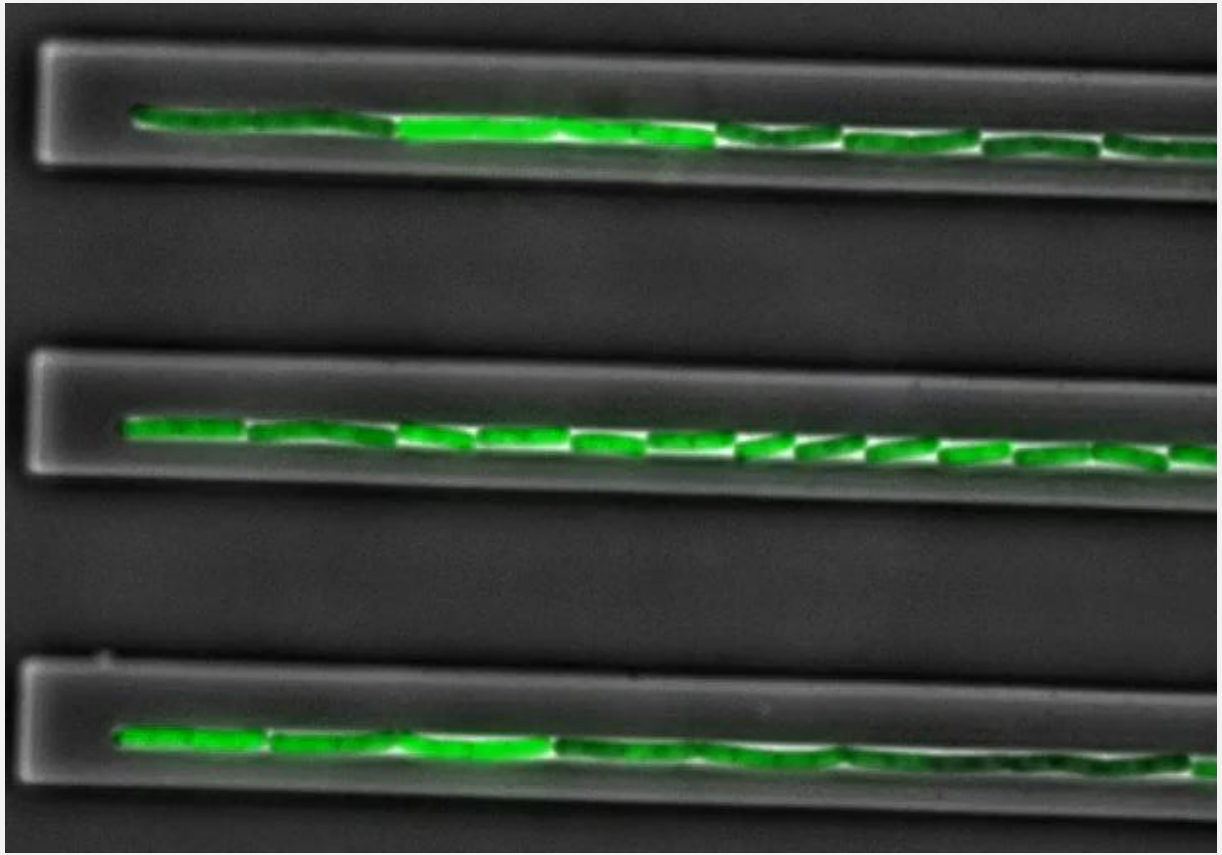


Time-lapse imaging on a microscope

- Images are captured on a microscope at set intervals (e.g., one frame every 10 minutes)

- The microscopes can also capture multiple fluorescence channels per frame (or time-point)

Note: Time-lapse images are also called "movies"

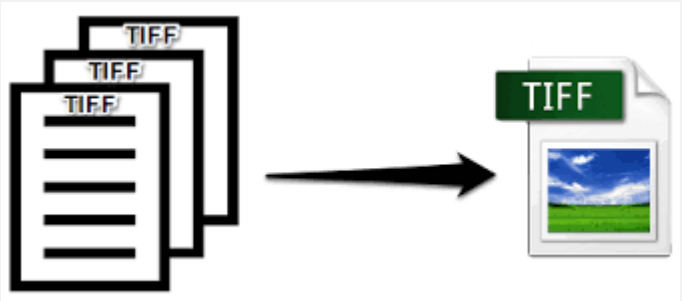


https://elifesciences.org/articles/33099/figures

Time-lapse images

- For this course, we will use *multi-page TIFF* files to store the time-lapse images

- These files store multiple frames, stored in a "stack"



Note: For multichannel time-lapse images, you will be given one TIFF for each channel

We will use the image L29_exampleTL.tif as an example

Getting image file information

- You can use the function `imfinfo` to get information about an image file

```
fileInfo = imfinfo(filename)
```

`fileInfo` is a multi-element struct that contains information about the image (e.g., width, height, bit depth etc.)

Number of frames in a multi-page TIFF

- The number of frames is the number of elements of the file information struct

```
numFrames = numel(fileInfo)
```

Frame information

- You can get information about a specific frame by indexing

  Example: Get the width of the frame 5

```
width = fileInfo(5).Width
```

Note: Usually this information is the same for all frames in a TIFF-stack

Reading a multi-page TIFF

- To read a multi-page TIFF, we can use imread:

```
I = imread(filename, frame)
```

Example: Read in frame 5

```
I = imread('L29_exampleTL.tif', 5)
```

Questions?

For loops

repeat code a set number of times

Basic syntax

```
for index = vector


end
```

Basic syntax

```
for index = vector
    keywords
end
```

Note: You must close a for loop with a matching `end` or MATLAB will throw an error

Note: Keywords are reserved names in MATLAB. You cannot name variables or files with a name matching a keyword. Use `iskeyword` to see list of keywords.

Basic syntax

```
for index = vector

end
```

Name of index variable

Basic syntax

```
for index = vector

end
```

Values of index

Note: The loop will repeat as many times as the number of elements in the index vector

Example for loop

```
for ii = [1 2 3 4 5]
    disp(ii)
end
```

Note: `disp` displays the value of the variable

- The index variable `ii` changes each time the loop runs (iteration)
  - First iteration: `ii = 1`
  - Second iteration: `ii = 2`
  - …
  - Fifth iteration: `ii = 5`

Note: This loop will run 5 times

Practice

- How many times will the following loop run?

```
for jj = 1:15
    x = jj * 2
end
```

A. 15 times
B. 16 times
C. Once
D. I don't know

Practice

- What is the value of `idx` in the fourth iteration in the following loop?

```
for idx = [10 1 42 2 5 20]
    disp(idx)
end
```

A. 10
B. 42
C. 2
D. 20

How to use for loops to analyze time-lapse images

1. Read in one frame at a time
2. Perform image analysis (e.g., making masks, measuring properties using `regionprops`)
3. Store time-series data in a matrix
4. Repeat for all frames

Homework question – count number of cells in movie

- I have provided a framework to get you started on this question

- This question could be challenging – **please start early**! Email me or schedule office hours if you get stuck.

Questions?

A common "mistake" is trying to load every frame of a time-lapse movie into memory at once

How to use for loops to analyze time-lapse images

1. Read in one frame at a time
2. Perform image analysis (e.g., making masks, measuring properties using `regionprops`)
3. Store time-series data in a matrix
4. Repeat for all frames

Note: Analysis is carried out on a per frame basis

Problem: Running out of memory

- Computers store temporary information in RAM (Random Access Memory)

- Temporary information = data that is only needed when running the current session

- Example: MATLAB variables

How much data can your computer store in memory?

- The maximum size (in memory) that a MATLAB variable can be depends on how much RAM is installed on your computer

- Example:
- 8 GB RAM = 8 GB of total data can be stored… except that operating system, open applications etc. all require RAM as well – typically 50% of RAM is already used up

Time-lapse imaging leads to large datasets

- Example: Say we set the microscope to capture three channels, at a rate of one frame every 10 minutes for 12 hours. How many images in total do we capture?

How much memory does a single image occupy?

- Say each image captured is 2048 x 2048 pixels. The image bit-depth is set to 16-bits. How large in megabytes (MB) is an image? (1 byte = 8 bits)

What is the total size of the time-lapse dataset?

Questions?

(Optional) Functions

- We are running out of time so I'll just throw this here for those who are interested

- I won't test you on material in these slides in the final exam

  Recorded lecture from last year:
  https://www.youtube.com/watch?v=18feMfwnfG4

Functions

are m-files that contain code that takes inputs and returns output(s)

Basic syntax

function <outputs> = <function_name>(<inputs>)

end

Example: Function to compute area of circle

```
function area = circleArea(radius)
    area = pi * radius^2;
end
```

Note: This function will need to be saved in an m-file with a filename that matches the function name, i.e., as `circleArea.m`

You can execute this function just like any other:

```
>> %Compute area of 10 radius circle
>> area = circleArea(10);
```

Functions do not share variables with the workspace

- Back in Lecture 3, I showed you that scripts shared variables with the Workspace
  - i.e., variables declared in a script will show up in the workspace and you can manipulate that variable in the Command Window (and vice versa).

- Functions have their own Workspace that is created when the function runs, and is cleared once it's finished
  - This means that variables declared in a function only exist when the function is running.

Functions do not share variables with the workspace

- Functions have their own Workspace that is created when the function runs, and is cleared once it's finished
  - This means that variables declared in a function only exist when the function is running.
  - This is great because it means that variables in different functions can have the same names (this happens a lot in larger code) and they won't ever interact
  - This is also why functions need to have inputs – the code doesn't have the data otherwise

When you would write a function

- Deciding when to split code out into a function vs keeping it as a long script is kind of an art form
- Basic rule-of-thumb: You should write a function if you plan to reuse the code
  - Example: if you had segmentation code that you want to reuse in other scripts, then making a function makes this easier