Lecture 10:
Analyzing videos and tracking bees

MCDB/BCHM 4312/5312

Download the file

`hw8_twobees.zip`

and extract the image to your MATLAB folder

Learning objectives

- The nearest-neighbor algorithm

- Reading movies
  - Multi-page TIFFs
  - for loops

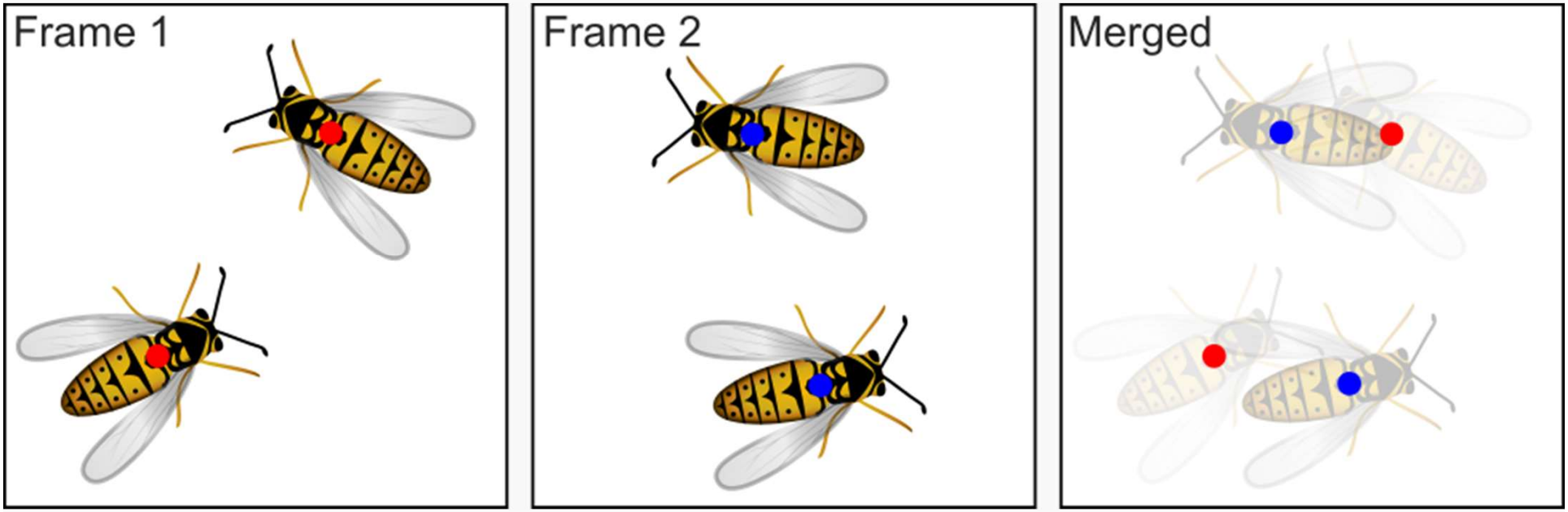
- Segmenting the bees
  - L*a*b colorspace for segmenting objects by color

- if statement

Tracking bees

The tracking problem

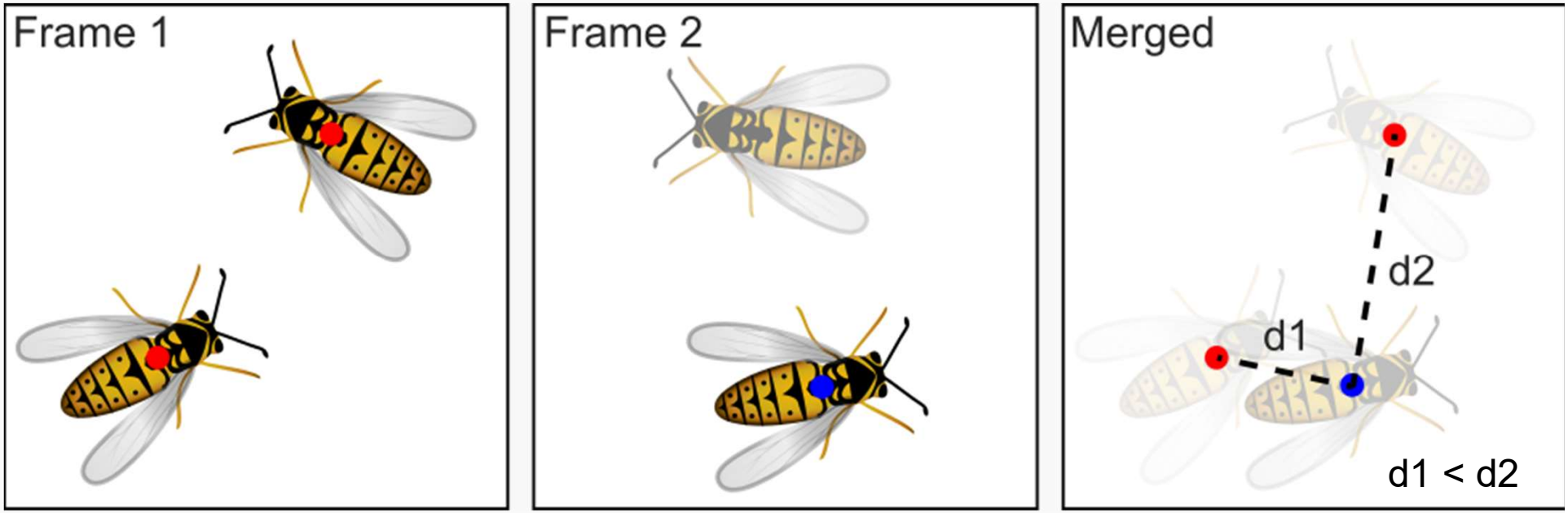
We want to follow the same object through every frame of the movie



How do we know which object is which?

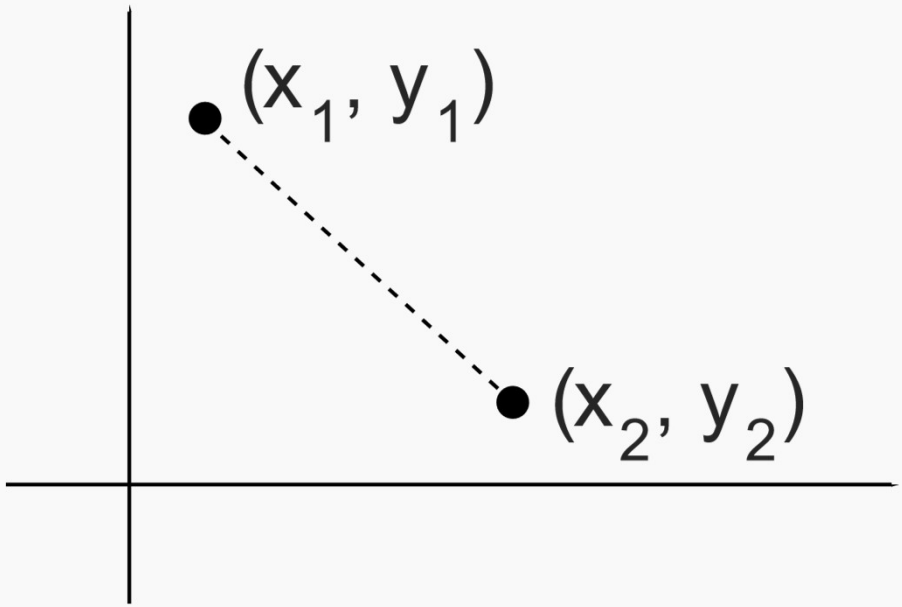
The nearest-neighbor algorithm

- Measure the distance of an object in a frame with every other object in the previous frame
- Link objects with the shortest distance (the nearest-neighbor) - “connect the dots”



Important assumption: The frame rate of the acquisition has to be slow enough that the objects do not move too much between frames

Distance between two points



$$\text{distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

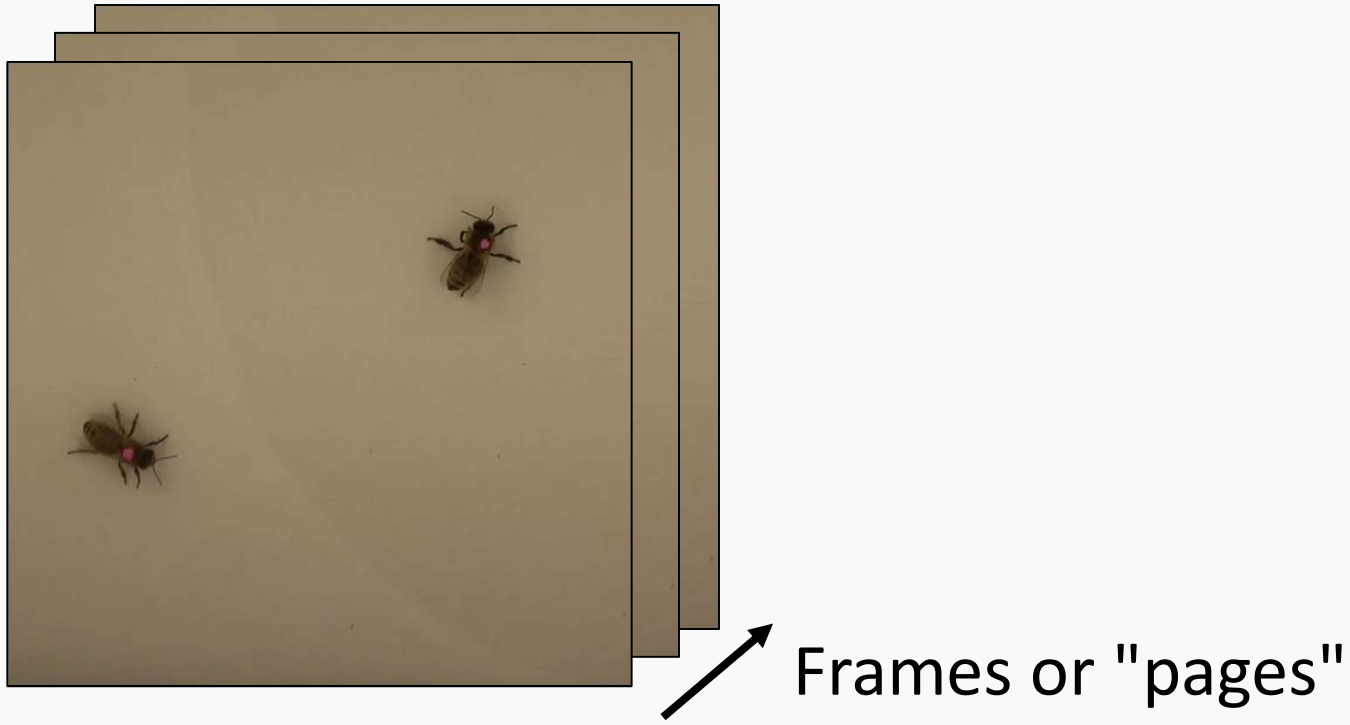
What is the sequence of steps to track the bees in the movie?

1. Read a frame of the movie
2. Segment the bees (or the spot on the bees)
3. Measure the position of the bees
4. Track the bees using the nearest neighbor algorithm
5. Repeat for each frame

1. Read a frame of the movie

- Time-lapse images are commonly saved as **multi-page TIFFs** (or TIFF stack)

- A single TIF file that contains multiple images



Reading multi-page TIFFs

To get number of images in a TIFF file, get the **im**age **f**ile **info**rmation:

```
info = imfinfo('hw8_twobees.tif');

numFrames = numel(info);
```

How many frames are in the movie 'hw8_twobees.tif'?

Reading in a specific image/page

Basic syntax:

```
I = imread('hw8_twobees.tif', page_num);
```

Example: Read in frame 5 and display it

```
I = imread('hw8_twobees.tif', 5);
```

What type of image is this?

a) **RGB image**

b) Grayscale image

c) Binary image

Remember:

- RGB images are width by height by color
- To index the different color channels, use matrix notation e.g. green = I(:, :, 2)

2. Segment the bees

How to segment the bees?

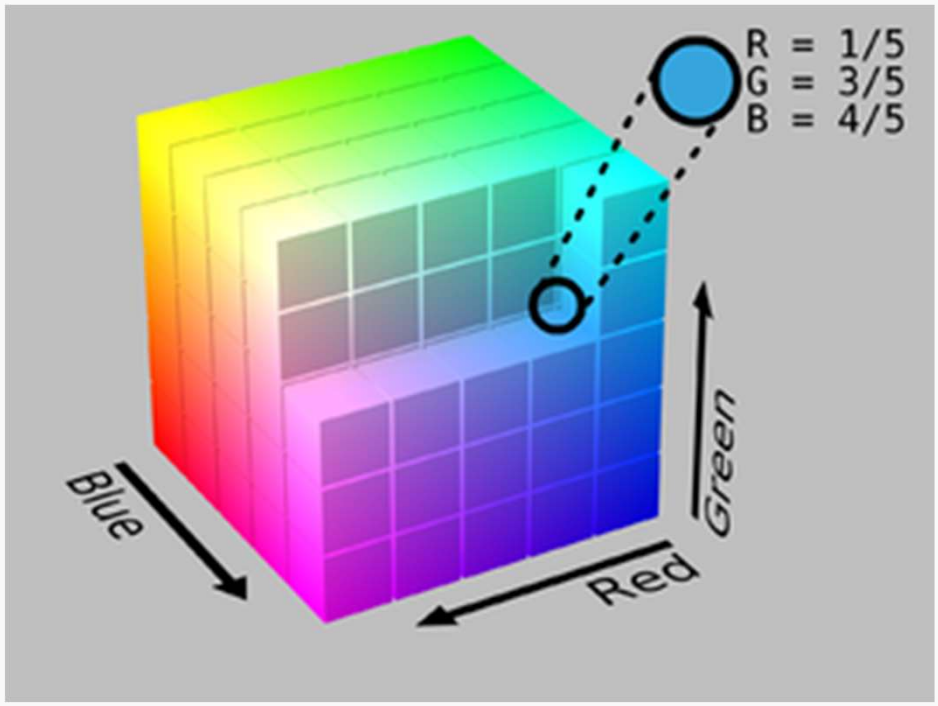
1. You could convert the image to grayscale (e.g. using `rgb2gray` or use an appropriate color channel), then intensity threshold the bees

2. **Use color segmentation to identify the pink dot**

Color spaces are specific ordering of colors

RGB color space

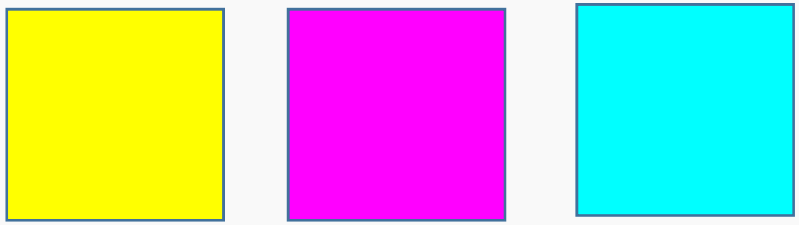
- Colors are defined by the coordinate (red, green, blue)
- 0 → black, 1 → white



Values indicate intensity

(1, 0, 0) (0.5, 0, 0) (0.03, 0, 0)

Get different colors by mixing different amounts of red, green, and blue

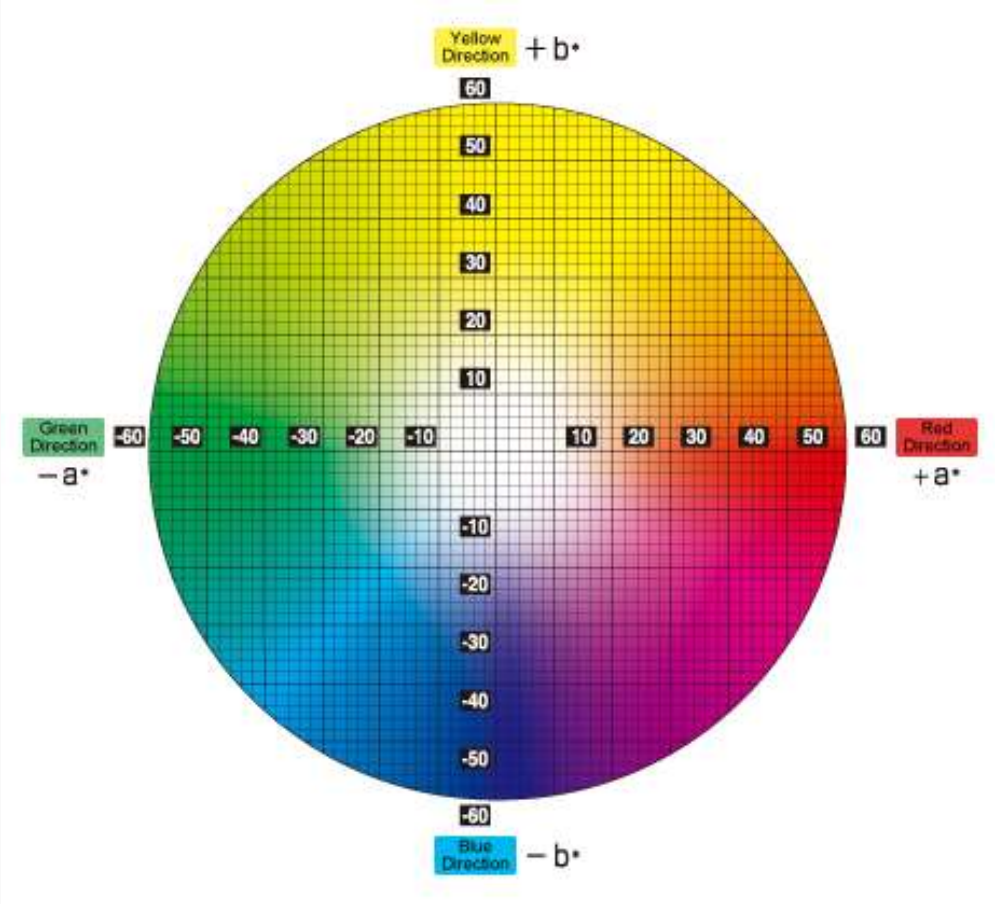


(1, 1, 0) (1, 0, 1) (0, 1, 1)

Selecting similar colors is difficult

→ Have to specify a cuboid (3D rectangle)

L*a*b color space



L = Luminosity (not shown on this plot)
L* = 0 black, L* = 100 white

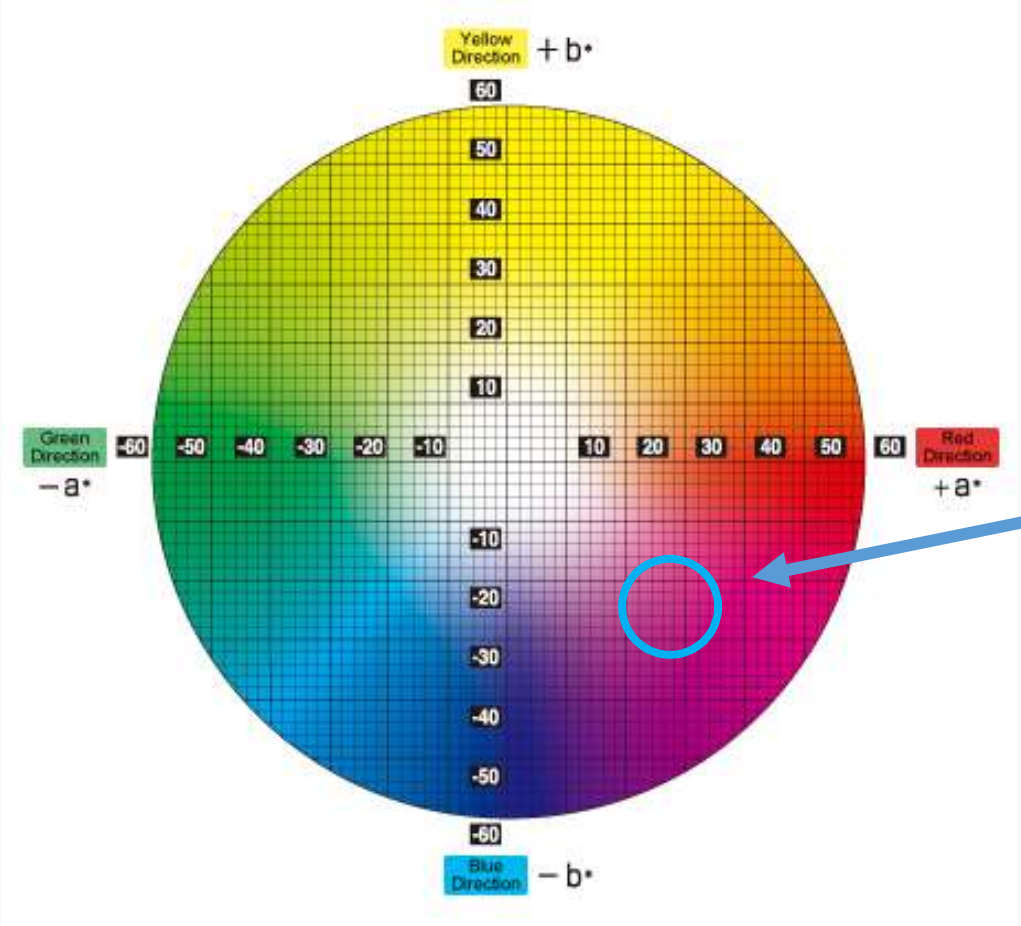
a* - is the green-red component
-ve a* = green
+ve a* = red

b* - the blue-yellow component
-ve b* = blue
+ve b* = yellow

Color is specified by the a* and b* coordinate

L*a*b color space

- Similar colors are grouped around each other in a*-b* coordinates



Color segmentation in MATLAB

1. Pick the color of the tag on bee in RGB.

2. Convert the color into L*a*b* coordinates.

3. Convert the whole image into L*a*b* colorspace.

4. Select pixels in a circle centered on the a*b* coordinates from 2.

1. Get the color of the tag on bee

```
imshow(I)
```

Use data tip, get color

```
I chose [134, 69, 82] as the spot color
```

2. Convert RGB spot color to L*a*b* color

Convert the RGB color into a 1 x 1 x 3 matrix:

```
spotRGB = cat(3, 134, 69, 82);
```

For the function to work correctly, the spot color must be the same data type as the original image (uint8):

```
spotLab = rgb2lab(uint8(spotRGB))
```

3. Convert the whole image into L*a*b*

```
labImg = rgb2lab(I);
```

The order of the third dimension of `labImg` is L*, a* and b*

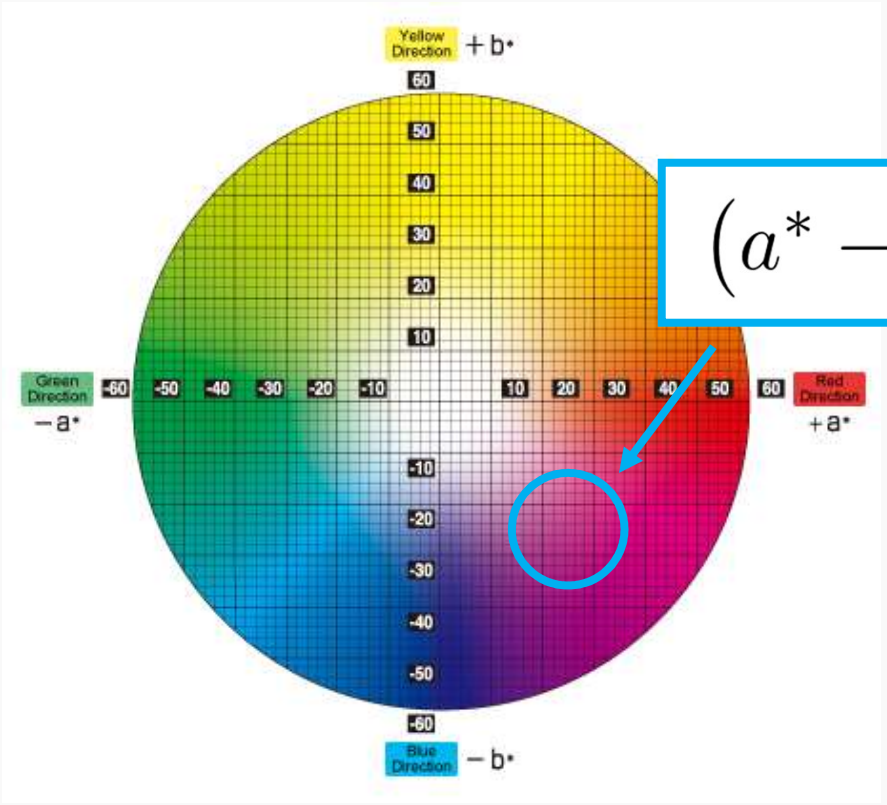
Index the a* and b* coordinates into new variables for convenience

```
aa = labImg(:, :, 2);
bb = labImg(:, :, 3);
```

4. Select pixels with similar colors

(i.e. select a circle around the color we want)

```
mask = (aa - spotLab(2)).^2 + (bb - spotLab(3)).^2 <= 10^2;
```



$$\left(a^* - a^*_{spot}\right)^2 + \left(b^* - b^*_{spot}\right)^2 \leq r^2$$

a^*, b^* = Image

a^*_{spot}, b^*_{spot} = Color of spot

r = radius of circle (larger circle, more colors allowed)

```
I = imread('hw8_beemovie.tif', 5);

spotRGB = cat(3, 134, 69, 82);
spotLab = rgb2lab(uint8(spotRGB));

labImg = rgb2lab(I);
aa = labImg(:, :, 2);
bb = labImg(:, :, 3);

mask = (aa - spotLab(2)).^2 + ...
    (bb - spotLab(3)).^2 <= 10^2;

mask = imopen(mask, strel('disk', 3));
```

3. Measure the position of the bees

How do we measure position using the mask?

```
data = regionprops(mask, 'Centroid')
```

5. Repeat for all frames using a loop

- **For loops** are useful for repeating sections of code

- Basic syntax:

Index variable Index values

```
for idx = 1:10

    disp(idx)

end
```

Statement body
Repeated each loop

For loop

```
for idx = 1:10

    disp(idx)

end
```

Loop 1: `idx` = 1

Loop 2: `idx` = 2

…

Loop 10: `idx` = 10

- During each loop:
  - The index variable `idx` will change to the next value
  - The statements in the body will be carried out

```
1.   finfo = imfinfo('hw8_twobees.tif');
2.   numFrames = numel(finfo);

3.   spotColorRGB = cat(3, 134,69, 82);
4.   spotColorLab = rgb2lab(uint8(spotColorRGB));
```

These values are constant so they should be outside the loop

Everything below this should be in a for loop

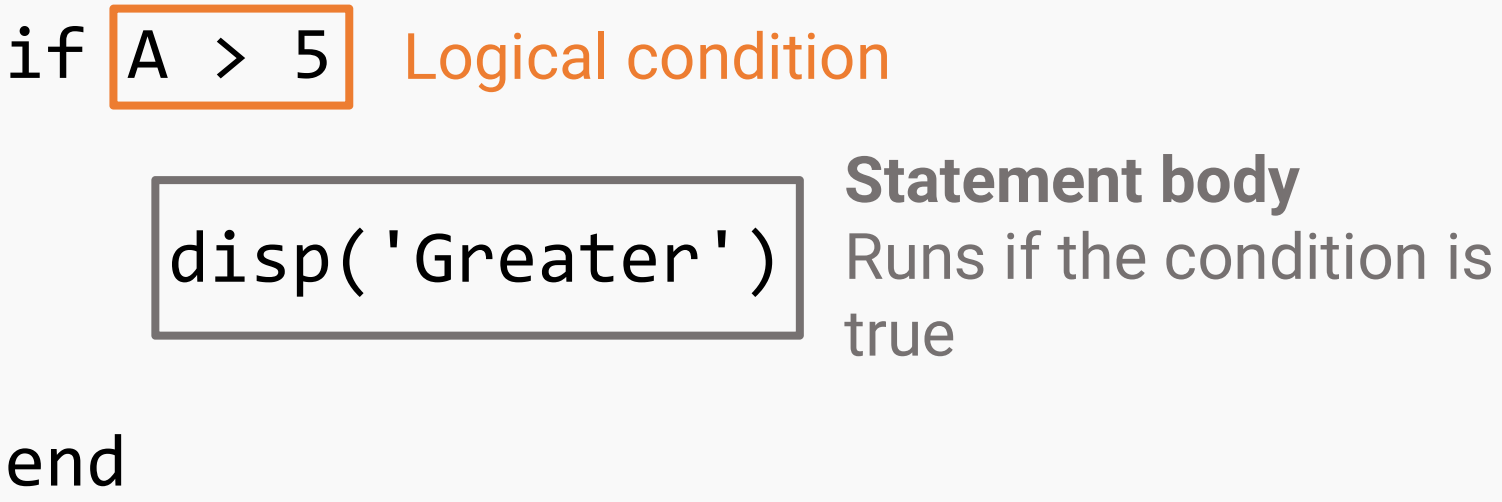
```
5.   currImage = imread('hw8_twobees.tif', 5);
6.   labImg = rgb2lab(currImage);

7.   aa = labImg(:, :, 2);
8.   bb = labImg(:, :, 3);

9.   mask = (aa - spotColorLab(2)).^2 + (bb - spotColorLab(3)).^2 <= 20^2;
10.  mask = imopen(mask, strel('disk', 3));

11.  data = regionprops(mask, 'Centroid');
```

```
1.   finfo = imfinfo('hw8_twobees.tif');
2.   numFrames = numel(finfo);

3.   spotColorRGB = cat(3, 134,69, 82);
4.   spotColorLab = rgb2lab(uint8(spotColorRGB));

5.   for idx = 1:numFrames
6.       currImage = imread('hw8_twobees.tif', idx);
7.       labImg = rgb2lab(currImage);

8.       aa = labImg(:, :, 2);
9.       bb = labImg(:, :, 3);

10.      mask = (aa - spotColorLab(2)).^2 + ...
                    (bb - spotColorLab(3)).^2 <= 20^2;
11.      data = regionprops(mask, 'Centroid');
12.  end
```

Stylistic: Statements in loops are indented by 4 spaces (TAB)
Makes it easier to identify where the loops are

If statements

- if statements allow you to control the execution of code based on a logical condition



If statements

- if statements allow you to control the execution of code based on a logical condition

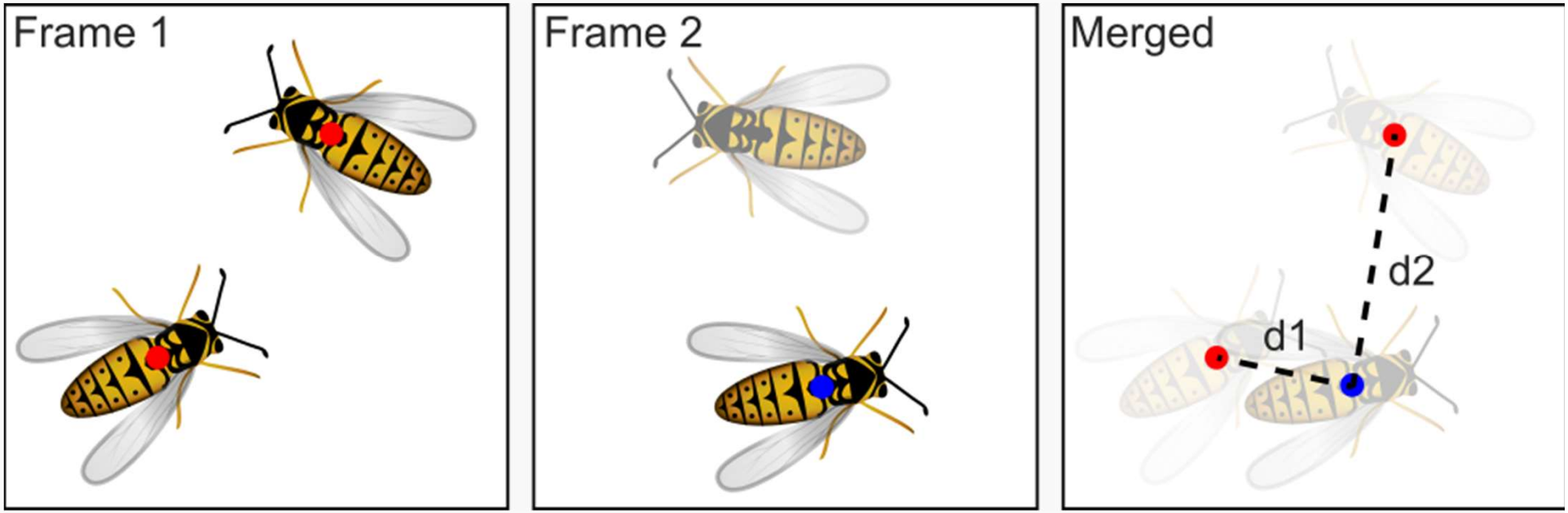
```
if A > 5
    disp('Greater than 5')
elseif A == 5
    disp('Equal to 5')
else

end
```

Use `elseif` to add additional conditions

`else` will run if no other condition was true
MUST BE LAST

4. Nearest neighbor tracking

- Measure the distance of an object in a frame with every other object in the next frame
- Connect objects with the shortest distance (the nearest-neighbor) - “connect the dots”



Setup: Define how you want to store the data

- Initialize two matrices to store the position of each bee outside the for loop:

```
posBee1 = zeros(numFrames, 2);
posBee2 = zeros(numFrames, 2);
```

- The columns are X and Y
- Each row is a new timepoint/frame

posBee1 =

X1	Y1
X2	Y2
…	…
XN	YN

- You can of course define other ways to store the data – this is what I'm using for the example

Different functions to initialize a matrix

- `zeros`
- `ones`
- `nan`

nan = Not a Number

- A value used as a placeholder when you don't want to confuse it for real data e.g. if your real data also contains ones and zeros

Summary of tracking code

```
if idx == 1
     posBee1(1, :) = data(1).Centroid;
     posBee2(1, :) = data(2).Centroid;
else
     dist_to_bee1 = sqrt((posBee1(iT - 1, 1) - data(1).Centroid(1))^2 + ...
          (posBee1(iT - 1, 2) - data(1).Centroid(2))^2);

     dist_to_bee2 = sqrt((posBee2(iT - 1, 1) - data(1).Centroid(1))^2 + ...
          (posBee2(iT - 1, 2) - data(1).Centroid(2))^2);

     if dist_to_bee1 < dist_to_bee2
          posBee1(iT, :) = data(1).Centroid;
          posBee2(iT, :) = data(2).Centroid;
     else
          posBee1(iT, :) = data(2).Centroid;
          posBee2(iT, :) = data(1).Centroid;
     end
end
```

For the first frame, can randomly assign to initialize the data

Calculate the distance

Find the nearest neighbor

This example uses the fact that we know there are only two bees

Otherwise, need to make sure we don't count a bee twice

Summary

- How the nearest neighbor algorithm works conceptually
  - Links objects in the current frame with the closest object in the previous frame
  - Straight-line distance formula

- MATLAB concepts:
  - Reading a multi-page TIFF
  - for loops
  - if statements
  - Pre-allocating a matrix to store data
  - Adding data to a matrix in the for loop

Homework

- Will be uploaded to Canvas after class – put the complete code together and make sure it works

- Please start! If you have questions:

jian.tay@colorado.edu

or

Drop by JSCBB A325

- Try first, but please reach out sooner rather than later if you get stuck