Announcements

Regular 9am LKSC209 lecture next week. Costumes encouraged

Lecture 05: Plotting II & Interactive Programs

NENS 230: Analysis Techniques in Neuroscience

Lecture 05 Outline

Questions?

Plotting a matrix with image and imagesc

Manipulating real images as matrices

Making movies

Interactive Programs

Assignment 5 Overview

Lecture 05 Outline

Questions?

Plotting a matrix with image and imagesc

Manipulating real images as matrices

Making movies

Interactive Programs

Assignment 5 Overview

Lecture 05 Outline

Questions?

Plotting a matrix with image and imagesc

Manipulating real images as matrices

Making movies

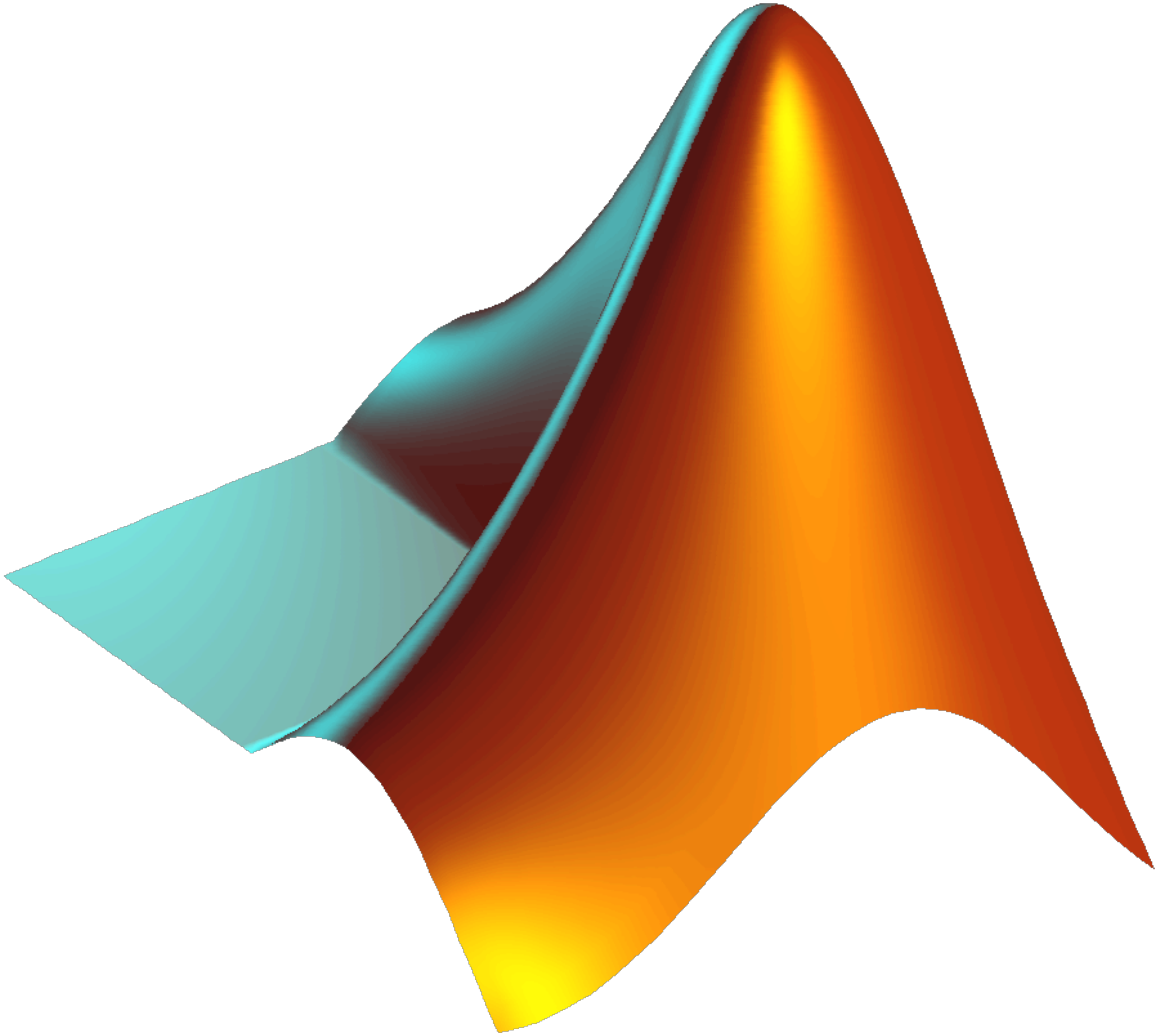
Interactive Programs

Assignment 5 Overview

imagesc and the colormap

`h = imagesc( matrix)` makes a rectangular **image object**, where each element of the matrix is represented by one tile of the image

Demo 1: imagesc of a matrix



imagesc and the colormap

`h = imagesc( matrix)` makes a rectangular **image object**, where each element of the matrix is represented by one tile of the image

The **colorbar,** created with `cbarh = colorbax(  ‘peer’ , axisHandle)` shows the mapping between element value and display color

This mapping from element value to color is defined by the axis’ **colormap**

imagesc scales the colormap such that the smallest and largest elements map to the colormap endpoints. Watch out for outlier values that throw off your colormap

You can specify built-in or custom colormaps using `colormap( axisHandle, yourColorMap )`

To define a variable as a built-in existing colormap, use for example

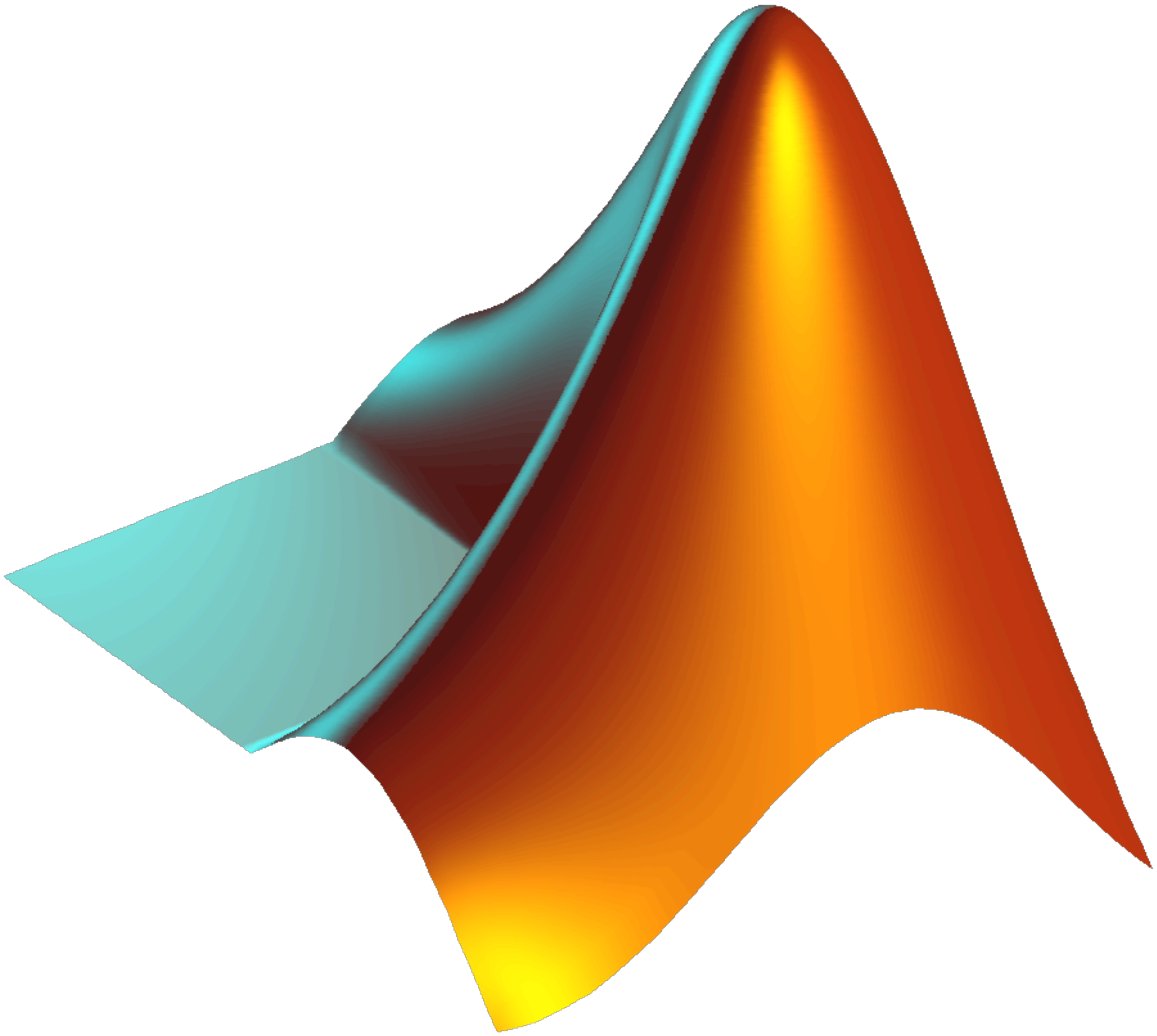
```
cmap = jet( numDiffColors )
```

which returns a numDiffColors x 3 matrix, where each row is an RGB color

image

`h = image( matrix)` is similar to imagesc except that the indexing into the colormap is explicit, i.e. a value of 1 is displayed as first color, 2 as second color, and so on...

Demo 2: image of a matrix



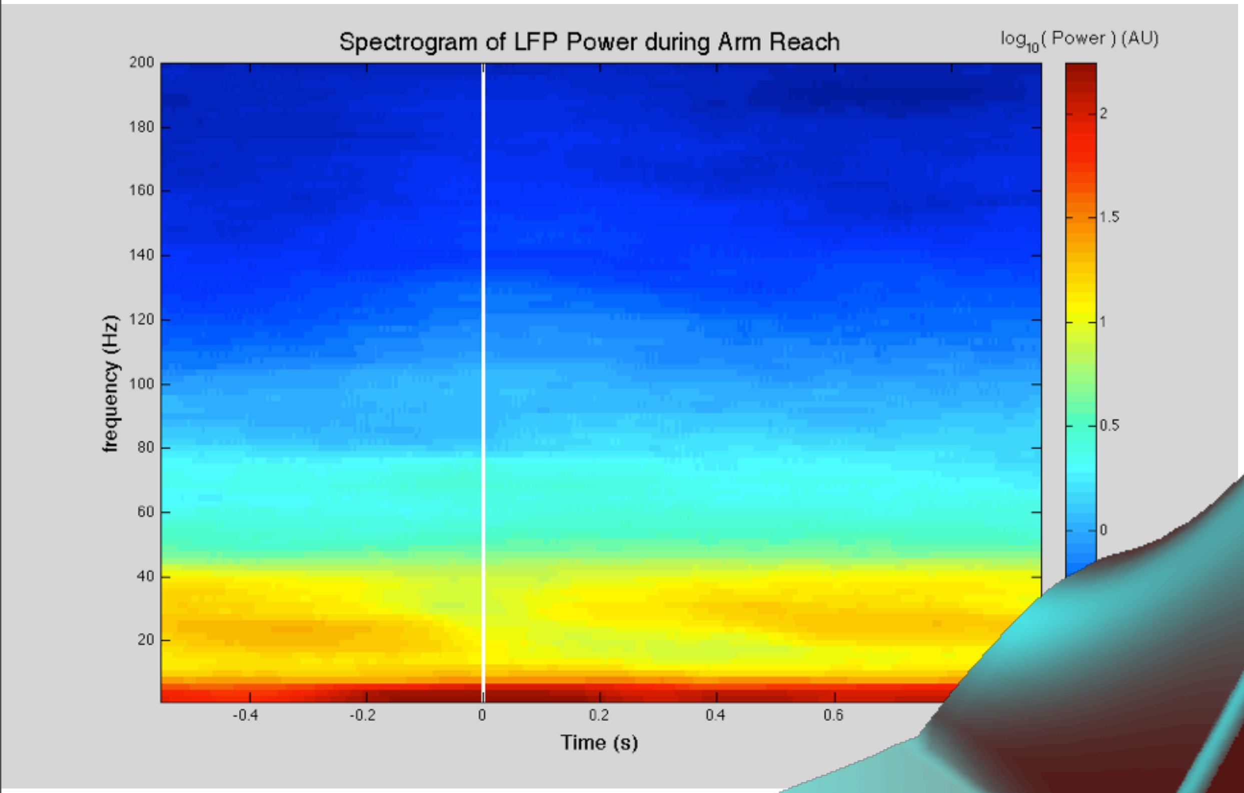
image

`h = image( matrix)` is similar to imagesc except that the indexing into the colormap is explicit, i.e. a value of 1 is displayed as first color, 2 as second color, and so on...

To just visualize abstract data in a matrix you generally want to use `imagesc( )`

When you want very tight control of color, use `image( )`. This is typically for a “real” image such as a photograph, microscope capture, etc.

Demo 3: Spectrogram using imagesc



More on imagesc and image

Both of these functions can be called with the xTick and yTick values specified as follows:

```
h = imagesc( xTicks, yTicks, matrix, Param1, Value1, ...)
```

An image object exists in an axis like any other graphics object. Thus, you can add things like lines, patches, and text annotations to this axis

Lecture 05 Outline

Questions?

Plotting a matrix with image and imagesc

Manipulating real images as matrices

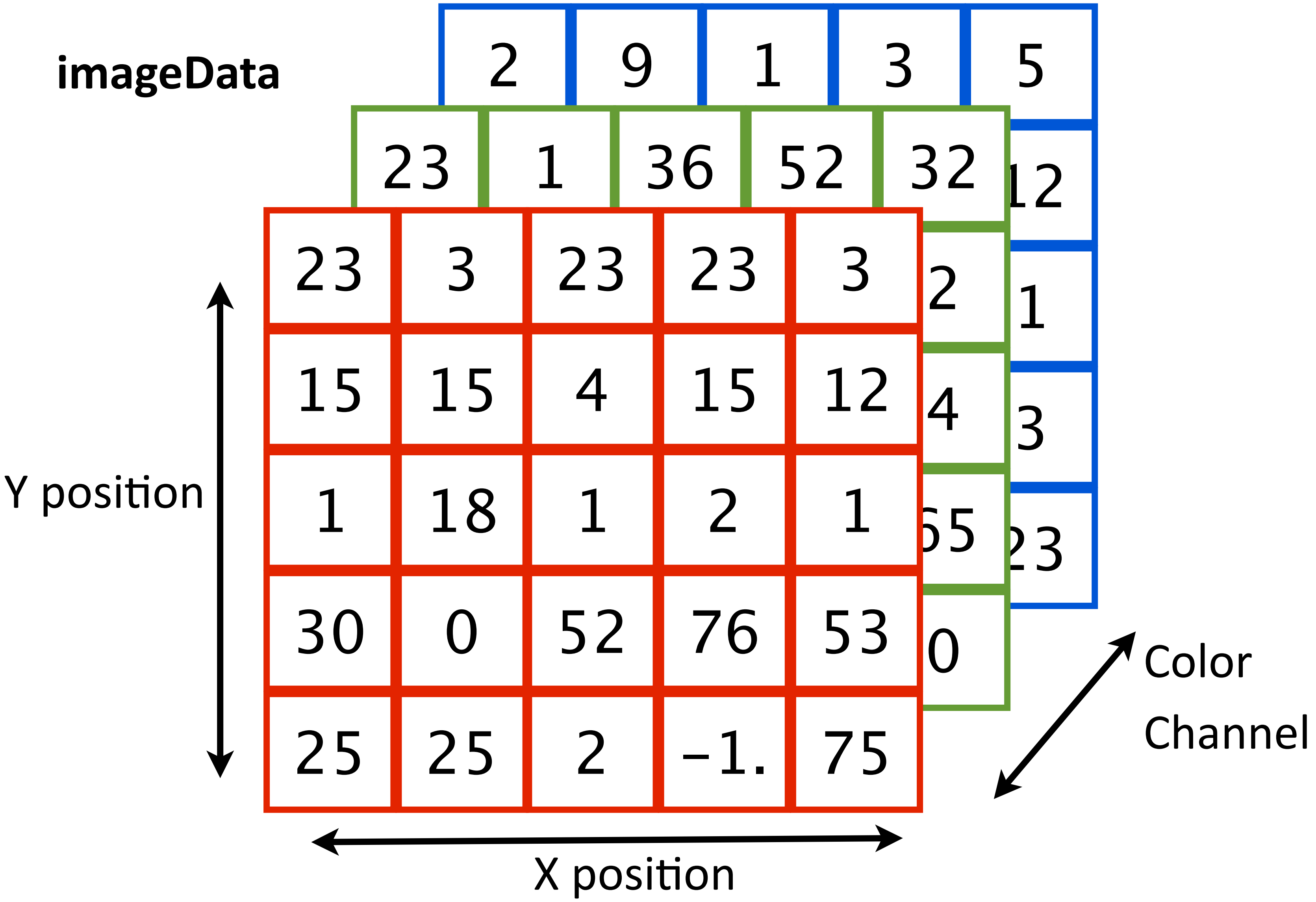
Making movies

Interactive Programs

Assignment 5 Overview

Importing and Manipulating Images

Import an image using `[imageData, colormap] = imread( filename, formatString)`
where **formatString** can be `'jpg'`, `'gif'`, `'tif'`, `'png'`, etc

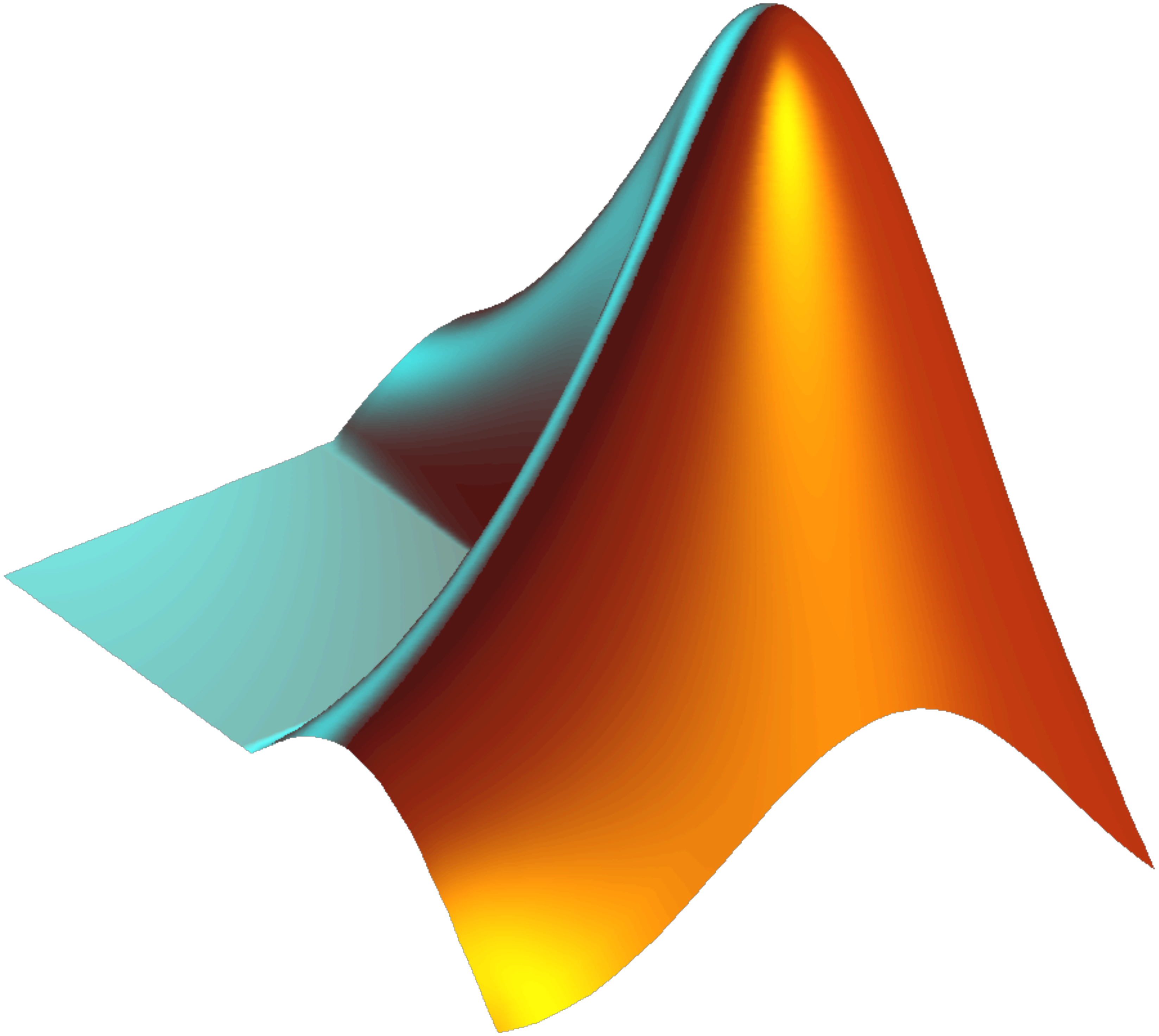


23	3	23	23	3
15	15	4	15	12
1	18	1	2	1
30	0	52	76	53
25	25	2	-1.	75

Images represented as (pixelRow) by (PixelCol) by (RGB) three-dimensional matrix

Each element takes values from 0 (min **intensity**) to 255 (max intensity); smaller values will be treated as 0 and larger values treated as 255.

Demo 4: Manipulating a photograph



Displaying a color channel Image

When visualizing a three-dimensional matrix where 3rd dimension is the color channel, this explicitly determines the color of that pixel and so colormap is ignored.

In this situation, image( ) and imagesc( ) do the same thing and colormap( ) will have no effect

Some image formats might have a length 4 color channel, e.g. CMYK

The **bit depth** of the image determines the range of each element in specifying intensity of that color channel (so 8-bit color means range of 2^8, e.g 0 to 255)

You can manipulate individual color channels, or individual regions of the image, using indexing.

Lecture 05 Outline

Questions?

Plotting a matrix with image and imagesc

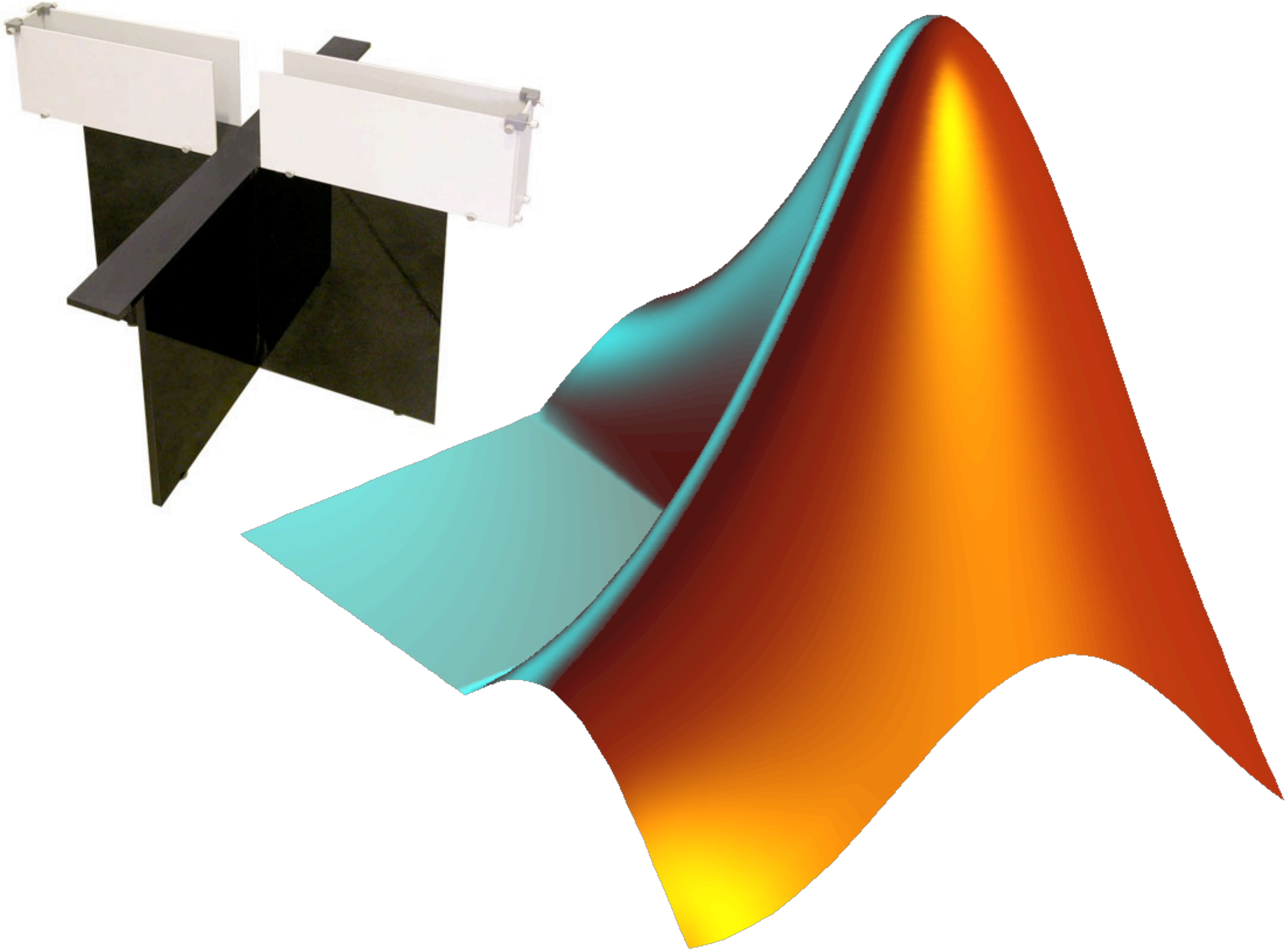
Manipulating real images as matrices

Making movies

Interactive Programs

Assignment 5 Overview

Demo 5: Plus Task Position-Tracking Movie



Movies

You build a movie by saving successive “screenshots” of an axis as a frame

Typically this is a five step process:

1. Create the initial arrangement of the axis
2. Update graphics object properties to make the next frame of the movie
3. Capture the new axis contents using
```
yourMovieFrames(frameIdx) = getframe( axisHandle );
```

Repeat for each frame

4. To test, play the movie within MATLAB using
```
movie( yourMovieFrames, numRepeats, framesPerSecond );
```
5. Export the movie into a .avi using
```
movie2avi( yourMovieFrames, movieFileName,  ‘fps’ , framesPerSecond, ... );
```

On MATLAB Central’s File Exchange, you can find user-created functions to make movies into other formats

You can specify compression parameters to make .avi that are smaller files

Changing the properties of existing graphics objects (e.g. text) is much faster than creating a brand new graphics object/axis/figure for each frame

Lecture 05 Outline

Questions?

Plotting a matrix with image and imagesc

Manipulating real images as matrices

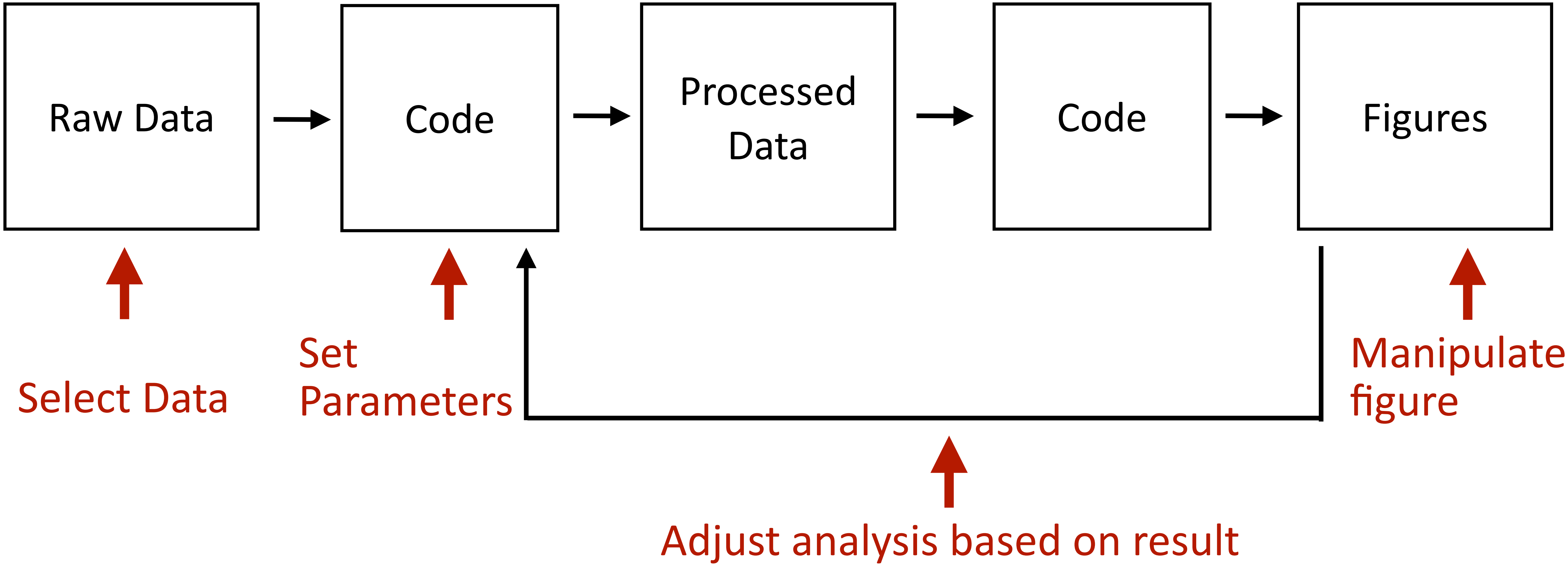
Making movies

Interactive Programs

Assignment 5 Overview

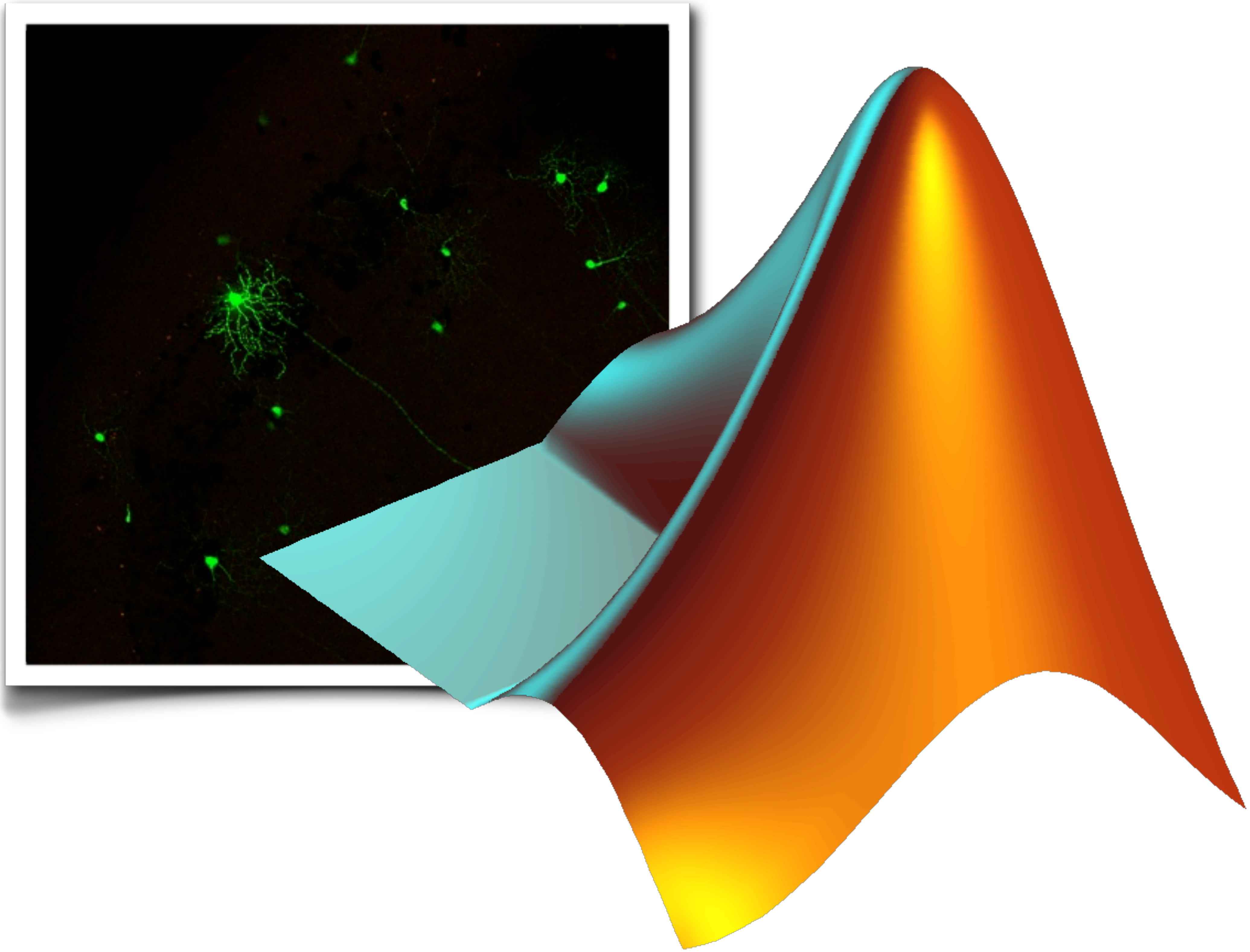
Interactive Programs

Thus far, you've written programs which are run (perhaps with some input parameters) and proceed to termination without user input



You can make more powerful analyses by allowing the user to interact with the program as it runs

Demo 6: Microscopy Image Editor



Interactive functions

`[fileName, pathName] = uigetfile( filter, dialogString )` makes a graphical user interface (**GUI**) that lets the user easily select a file.

- The argument *filter* is a cell array of possible search filters (specified as strings)
- *dialogString* is displayed in the GUI and should instruct the user

`directory = uigetdir( startPath, dialogString )` is analogous for selecting a directory

`choice = menu( dialogString, ‘option1’ , ‘option2’ , ‘option3’ , ...)`
presents a GUI with boxes labelled ‘option1’, ‘option2’, ‘option3’ and returns the index of which option the user clicked

`response = input( promptString )` displays the *promptString* at the **command-line** and records the user’s response. By default, the response is **evaluated**

- Use second argument, `‘s’` if you do not want the user’s response to be evaluated

`response = inputdlg( promptString, dialogTitle )` is a GUI version of input

- The response is not evaluated, and is put inside a cell

Interactive functions (continued)

`[x, y] = ginput( n )` returns the x,y coordinate that user clicks in the current axis *n* times

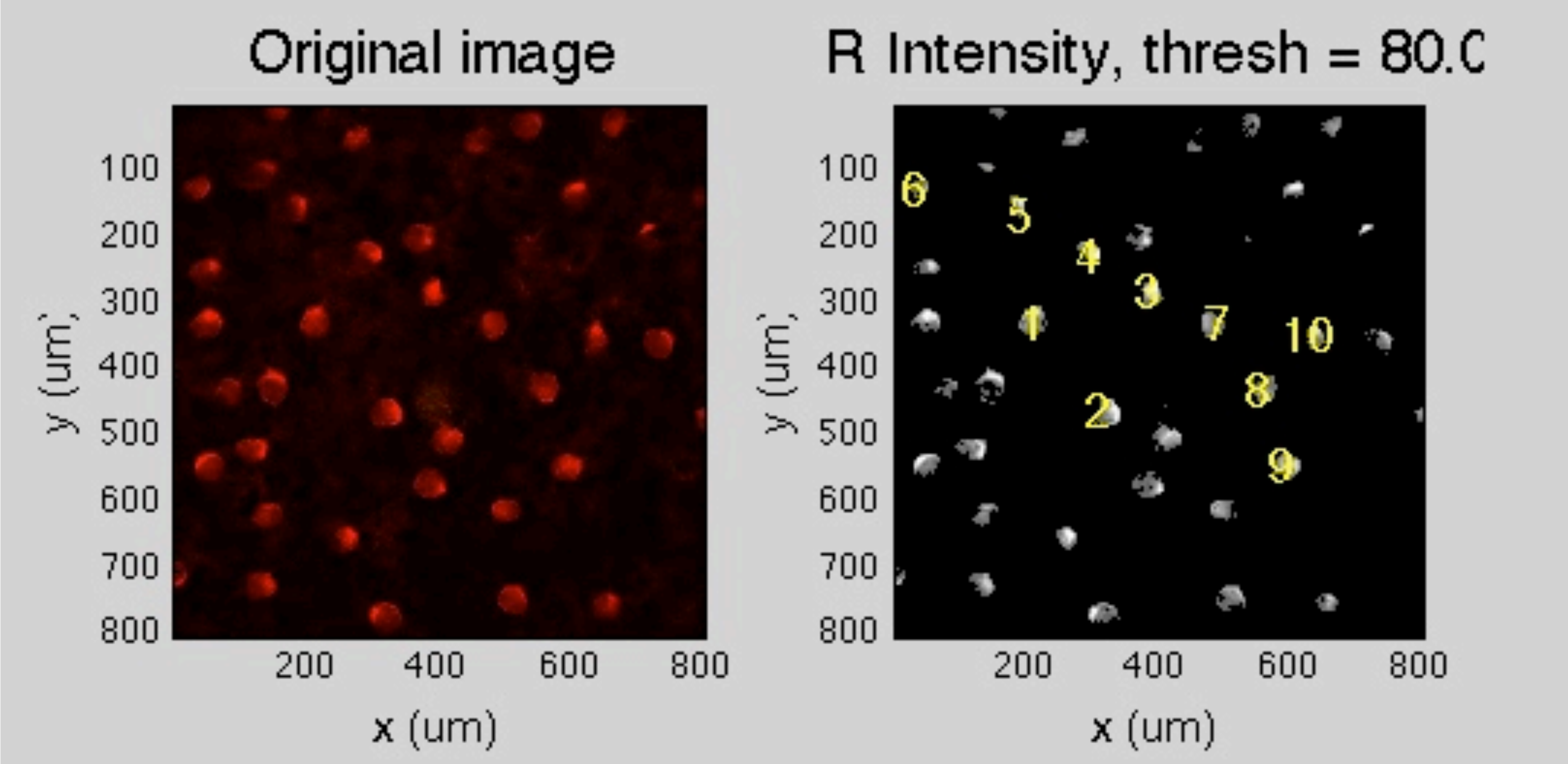
- If n > 1, user can hit ‘Enter’ to terminate early.
- Can also do `[x, y, key] = ginput( n )` to get which button was pressed. For example, key returns 1 for left mouse and 3 for right mouse

`[fileName, path] = uiputfile( filter, dialogString )` presents a GUI for the user to select the name and save directory of a file

- *filter* allows you to suggest a file format. * means wildcard (i.e. anything)

These are just a subset of the many useful interactive functions and GUI-building tools MATLAB has available; look through the Help for other similar functions and optional arguments

Assignment Five: Cell Counting



Dataset consists of a microscope image containing a number of ChAT-stained cells

You will write an interactive program which will be a tool to help you record the location and number of cells in such an image

It will present a single color channel in grayscale alongside the original image, and let you apply an intensity threshold before clicking to 'count' cells

The final output will be a .txt file with the number and location of the counted cells along with the saved image

Lecture 05 Review

Key Concepts

imagesc and **image** both create an **image object** which displays each element of a matrix

The mapping from matrix element value to color is specified by the **colormap**

Colormaps are Nx3 matrices where each row is an RGB color

There are many **built-in colormaps**; you can access and modify these, or make your own

a **colorbar object** is a visual depiction of the colormap of the adjascent axis

imagesc scales the mapping into the colorbar to use the full range of colors for the data

image explicitly maps the value of an element to the nearest index of the colormap

image objects live in an axis which can be manipulated and populated with other objects

You can import existing images of various formats using **imread**

An imported image is just a three-dimensional matrix, (Y) by (X) by (Color)

You can display such images using image(threeDmat) without worrying about colormap

Assemble a **movie** by building an array of matrices each capturing a single **frame**

Movies can be played within MATLAB or exported into a stand-alone format such as .avi

Interactive programs change their behavior depending on run-time user input

uigetfile, **uigetdir** allow user to easily navigate file directories and select file/directories

menu allows you to graphically present a fixed set of choices to the user

input allows you to query the user to enter a response at the command line

inputdlg is similar to input, but with a graphical dialog box

ginput lets the user to specify a location in an axis

uiputfile allows the user to easily specify the directory and name of a file to save

Functions

```
imagesc
image
colorbar
colormap
imread
cat
getframe
movie
movie2avi
uigetfile
uigetdir
menu
input
inputdlg
ginput
uiputfile
```