MATLAB Review – Fundamentals and Image Processing

LABORATORY SESSIONS

MATLAB FUNDAMENTALS

MATLAB fundamentals

Creation and use of variables

- Run these instructions and discuss their results:

```
» a = 5
» b = 10
» (a + b ) /2
» c = (a + b ) /2
» d = (a + b ) /2;
```

(Pay attention to the semicolon at the end of instruction)

Notes:

- Matlab distinguishes between small or capital letters in variable names
- Variable names cannot contain spaces
- Up/down arrows recall previous/next instructions making easier to repeat
- *help/doc* displays help about a command

MATLAB fundamentals

Creation and use of variables

Useful commands regarding variables:

- **whos:** lists all the variables in the current workspace, together with information about their size, bytes, class, etc.
- **save:** stores all variables from the current workspace in a MATLAB formatted binary file (MAT-file).
- **load:** loads the variables from a MAT-file into the current workspace
- **clear:** removes all variables from the workspace.

➢ Run these instructions and look at the results of *whos* command:

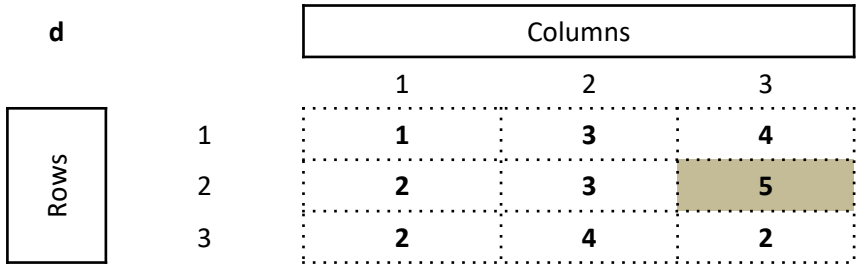
```
» save session1.mat
» clear
» whos
» load session1.mat
» whos
```

MATLAB fundamentals

Creation of vectors and matrices

- Run these instructions and look at the results of *whos* command:

```
» clear
» a = 0.5
» b = [0 2 1]
» c = [4 ; 2 ; 5]
» d = [1 3 4 ; 2 3 5 ; 2 4 2]
» e = [b b]
» f = [b ; b]
» whos
```



Matrix indexing

MATLAB fundamentals

Matrix indexing

- Run these instructions regarding different ways of matrix indexing:

```
» a(1)
» b(1)
» c(3)
» d(2,2)
» d(1,3)
» e(5)
» d(3,b(2))
» c(b(3))
```

```
» d(:,2)
» a = d(:,2)
» a(1)
» d(3,:)
» d(1:2,2:3)
» d(2:3,1:2)
» b(-1)
» d(b(1),1)
```

MATLAB fundamentals

Creation of functions

- Create a new *.m file and copy this code:

```
function result = addition(a,b)

result = a + b;
```

- Save the file as addition.m in your working directory

- The first line defines the output variables(s), the input variables(s) and their order.
- The function does not see all the workspace, just the the input variables(s).
- The second line adds up the input variables and stores the result in the output variable *result*
- The function will be called typing the file's name with the input variable(s) inside parenthesis

MATLAB fundamentals

Creation of functions

- Call the function, for example like this:

```
» clear
» term1 = 3;
» addition(term1,4)
» b = addition(term1,term1/3)
```

- Add some comments after the first line:

```
function result = addition(a,b)
% result = addition(a,b)
% This function returns the sum of the two input variables
% % Code written by: Author
% Session 1 of TDIV-Laboratory
resultado = a + b;
```

- Type *help addition* and see the result

MATLAB fundamentals

Review of Matlab functions

Files management:

Nombre	Descripción	Ejemplo
save	Stores workspace's variables in a file	» save session.mat
load	Load variables of a file in the workspace	» load session.mat
cd	Displays the current working directory. Change the current working directory.	» cd » cd d:\laboratory\tdiv\session1
imread	Reads a grayscale or color image from a file and returns it as a matrix.	[image, palette] = imread('image.bmp');

Matrices:

Nombre	Descripción	Ejemplo
size	Returns the matrix size.	» [height, width] = size(matrix);
zeros	Creates a matrix of zeros with the specified dimensions.	» b = zeros(2,2) » h = zeros(3,4,3)
ones	Creates a matrix of ones with the specified dimensions.	» a = ones(2,2) » d = ones(3,4,3)

MATLAB fundamentals

Review of Matlab functions

Operations:

Nombre	Descripción	Ejemplo
+	Adds two variables. They must have the same dimensions unless one of them is a scalar	» A+B » D = [2 1;3 1] + [5 7;3 3]
-	Subtracts two variables (same restrictions as +)	» A-B » D = [2 1;3 1] - [5 7;3 3]
*	Matrix or scalar product.	» A*B » [2 1;3 1] * [5 7;3 3]
/	Matrix or scalar division	» A*B » [2 1;3 1] * [5 7;3 3]
.*	Element-by-element multiplication of two matrices with the same dimensions.	» A.*B » [2 1;3 1] .* [5 7;3 3]
./	Element-by-element division of two matrices with the same dimensions.	» A/B » [2 1;3 1] .* [5 7;3 3]
^	Power.	» A^2 » [2 1;3 1] ^3
.^	Element-by-element power of two matrices. They must have the same dimensions unless one of them is a scalar.	» A.^2 » [2 1;3 1] .^3

MATLAB fundamentals

Review of Matlab functions

Indexing:

Nombre	Descripción	Ejemplo
(x,y,z)	Reference a single element of a matrix.	» A(2) % Vector » A(2,1) % Matrix » A(4,1,6) % 3D Matrix » A(1,6,2,7) % 4D Matrix
(:, :)	Reference a set of rows/columns of a matrix.	» A(2:3,1:4) % Reference all the elements from row 2 to 3 and from column 1 to 4. » A(1,2:5) % Reference the elements from columns 2 to 5 belonging to row 1.
(:, n) *(n, :)*	Reference the whole column/row of a matrix. It may be also used for matrices of more than two dimensions.	» A(:,1) % Reference all the elements of column 1 » A(3,:) % Reference all the columns of row 3.

MATLAB fundamentals

Review of Matlab functions

Loops and conditions:

Nombre	Descripción	Ejemplo

for	Loop for executing the statements a number of times	for i = 1:n,
		  for j = 1:n,
		  A(i,j) = 1/(i+j-1);
		  end
		end

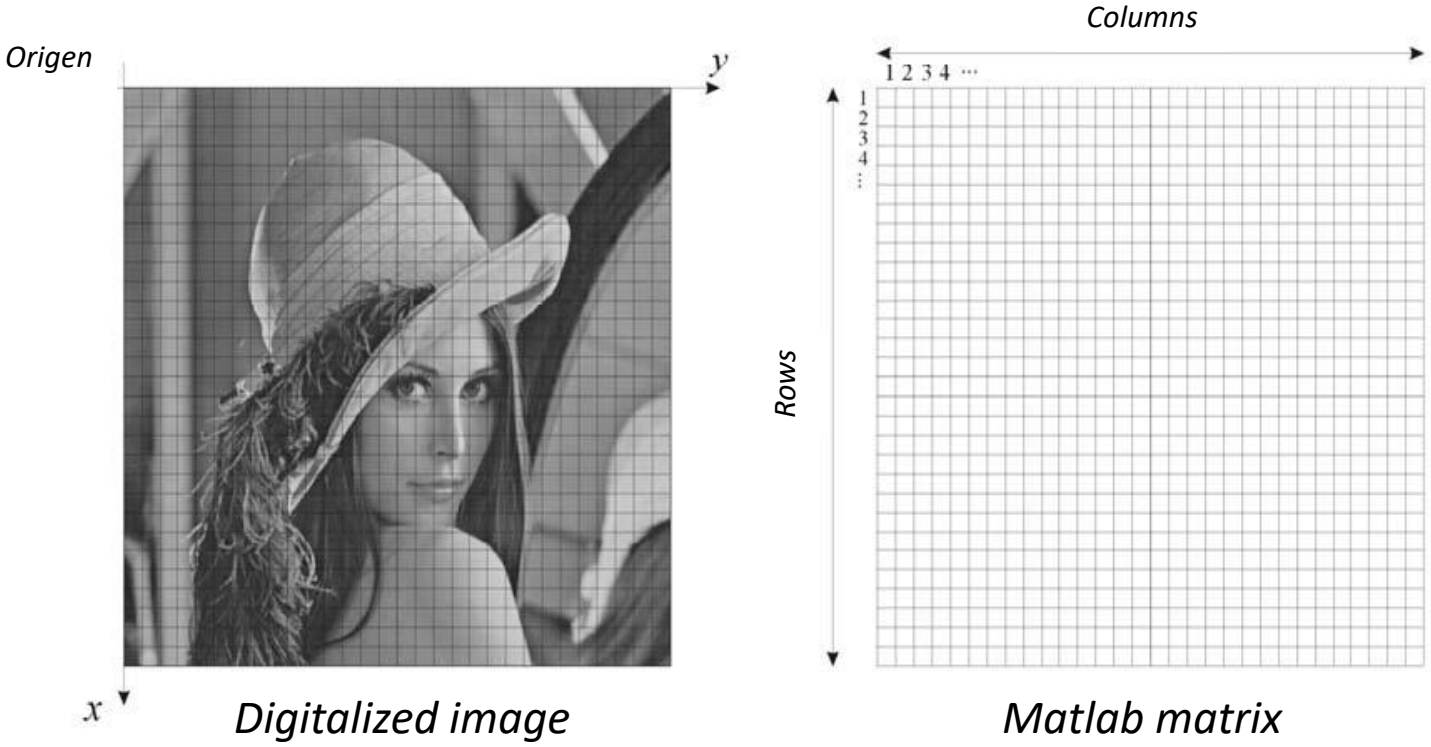
while	Repeatedly execute statements while condition is true	cond=1; while cond==1 command; End

if
else
elseif
	Execute statements if condition is true	if i == j
		  A(i,j) = 2;
		elseif abs(i-j) == 1
		  A(i,j) = -1;
		else
		  A(i,j) = 0;
		end

IMAGE PROCESSING WITH MATLAB

MATLAB for Image Processing

Digital image in Matlab



- Each pixel corresponds to one entry of the Matlab matrix, which represents its bright level.

MATLAB for Image Processing

Read and Write Image Data from Files

Read and write image data from files, get information about contents of image files

Functions

imread	Read image from graphics file
imwrite	Write image to graphics file
imfinfo	Information about graphics file

Basic Display

View image data, view multi-frame images (movies), set display preferences

Functions

imshow	Display image
montage	Display multiple image frames as rectangular montage
immovie	Make movie from multiframe image
implay	Play movies, videos, or image sequences

MATLAB for Image Processing

- How to read and represent an image:

'football.jpg': 8-bit RGB image

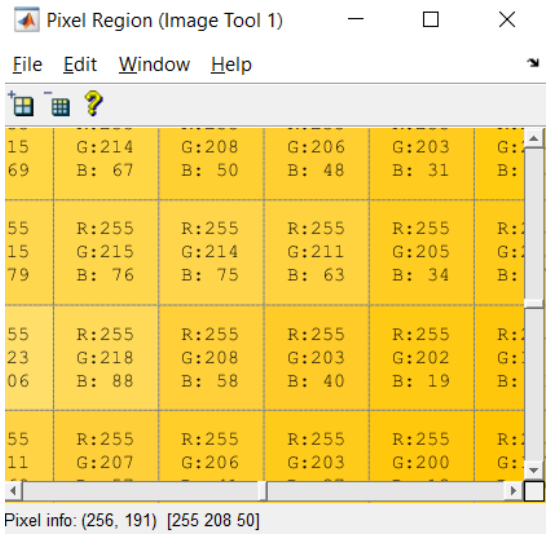
```
» image1 = imfinfo('football.jpg'); % obtains image information
» image1 = imread('football.jpg'); % reads an image file and stores it in a variable
» [rows, columns] = size(image1); % obtains the size of the image
» whos % information of created variables
» imshow(image1); % displays image stored in a variable
```

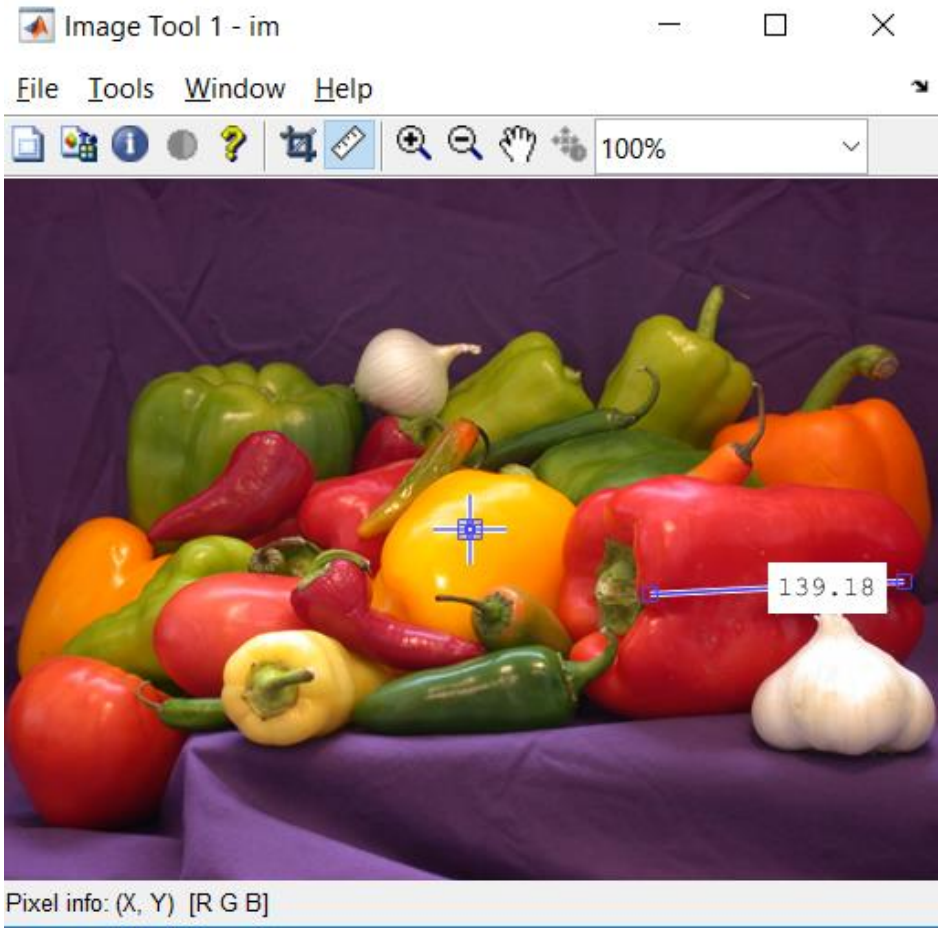
- Repeat the previous commands with the following images:
  - 'cameraman.tif': 8-bit grayscale image
  - 'trees.tif': indexed image
  - 'text.png': binary image

MATLAB for Image Processing

Interactive Exploration with the Image Viewer App

imtool





View image, measure distances, crop image, inspect pixel values, …

MATLAB for Image Processing

Contrast Adjustment

Contrast adjustment, histogram equalization, decorrelation stretching

Functions

imadjust	Adjust image intensity values or colormap
imadjustn	Adjust intensity values in N-D volumetric image
imcontrast	Adjust Contrast tool
imsharpen	Sharpen image using unsharp masking
imflatfield	2-D image flat-field correction
imlocalbrighten	Brighten low-light image
imreducehaze	Reduce atmospheric haze
locallapfilt	Fast local Laplacian filtering of images
localcontrast	Edge-aware local contrast manipulation of images
localtonemap	Render HDR image for viewing while enhancing local contrast
histeq	Enhance contrast using histogram equalization
adapthisteq	Contrast-limited adaptive histogram equalization (CLAHE)
imhistmatch	Adjust histogram of 2-D image to match histogram of reference image

MATLAB for Image Processing

Image Filtering

Convolution and correlation, predefined and custom filters, nonlinear filtering, edge-preserving filters

Functions

imfilter	N-D filtering of multidimensional images
fspecial	Create predefined 2-D filter
fspecial3	Create predefined 3-D filter
roifilt2	Filter region of interest (ROI) in image
nlfilter	General sliding-neighborhood operations
imgaussfilt	2-D Gaussian filtering of images
imgaussfilt3	3-D Gaussian filtering of 3-D images
wiener2	2-D adaptive noise-removal filtering
medfilt2	2-D median filtering
medfilt3	3-D median filtering
imbilatfilt	Bilateral filtering of images with Gaussian kernels
imnlmfilt	Non-local means filtering of image

MATLAB for Image Processing

Image Arithmetic

Add, subtract, multiply, and divide images

Functions

`imabsdiff`	Absolute difference of two images
`imadd`	Add two images or add constant to image
`imapplymatrix`	Linear combination of color channels
`imcomplement`	Complement image
`imdivide`	Divide one image into another or divide image by constant
`imlincomb`	Linear combination of images
`immultiply`	Multiply two images or multiply image by constant
`imsubtract`	Subtract one image from another or subtract constant from image

MATLAB for Image Processing

Geometric Transformations

Resize, rotate, and crop images; perform geometric transformation of multidimensional arrays

Functions

imcrop	Crop image
imcrop3	Crop 3-D image
imresize	Resize image
imresize3	Resize 3-D volumetric intensity image
imrotate	Rotate image
imrotate3	Rotate 3-D volumetric grayscale image
imtranslate	Translate image

imwarp	Apply geometric transformation to image
affineOutputView	Create output view for warping images
fitgeotrans	Fit geometric transformation to control point pairs
findbounds	Find output bounds for spatial transformation
fliptform	Flip input and output roles of spatial transformation structure
tformfwd	Apply forward spatial transformation
tforminv	Apply inverse spatial transformation

MATLAB for Image Processing

Read and Write Image Data from Files

Read and write image data from files, get information about contents of image files

imread	Read image from graphics file
imwrite	Write image to graphics file
imfinfo	Information about graphics file

Image Type Conversion

Convert between the image types, such as RGB (truecolor), binary, grayscale, and indexed.

rgb2gray	Convert RGB image or colormap to grayscale
rgb2ind	Convert RGB image to indexed image
ind2rgb	Convert indexed image to RGB image
label2rgb	Convert label matrix into RGB image
hsv2rgb	Convert HSV colors to RGB
rgb2hsv	Convert RGB colors to HSV
imbinarize	Binarize 2-D grayscale image or 3-D volume by thresholding
adaptthresh	Adaptive image threshold using local first-order statistics
imsegkmeans	K-significa segmentación de imagen basada en clustering

MATLAB for Image Processing

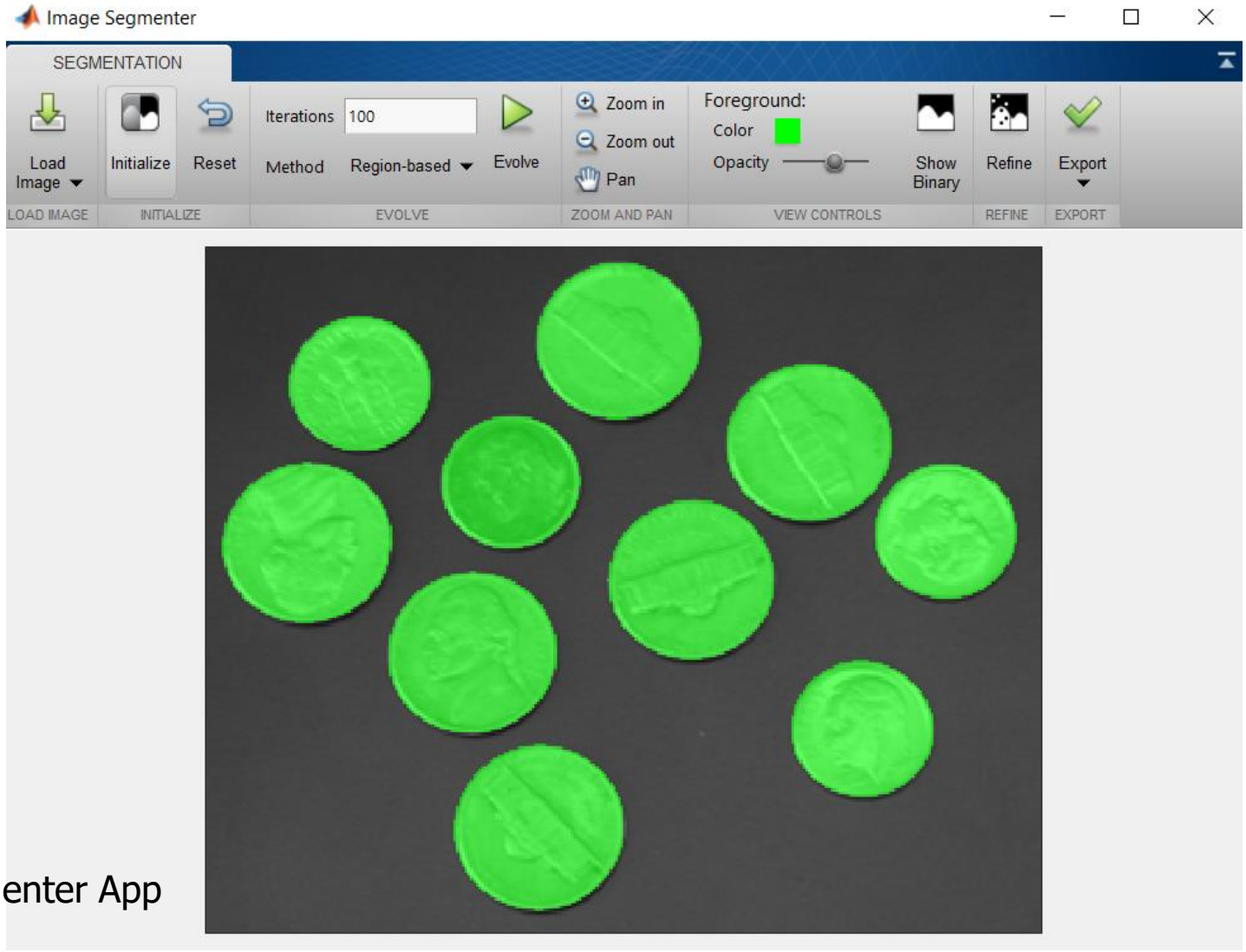
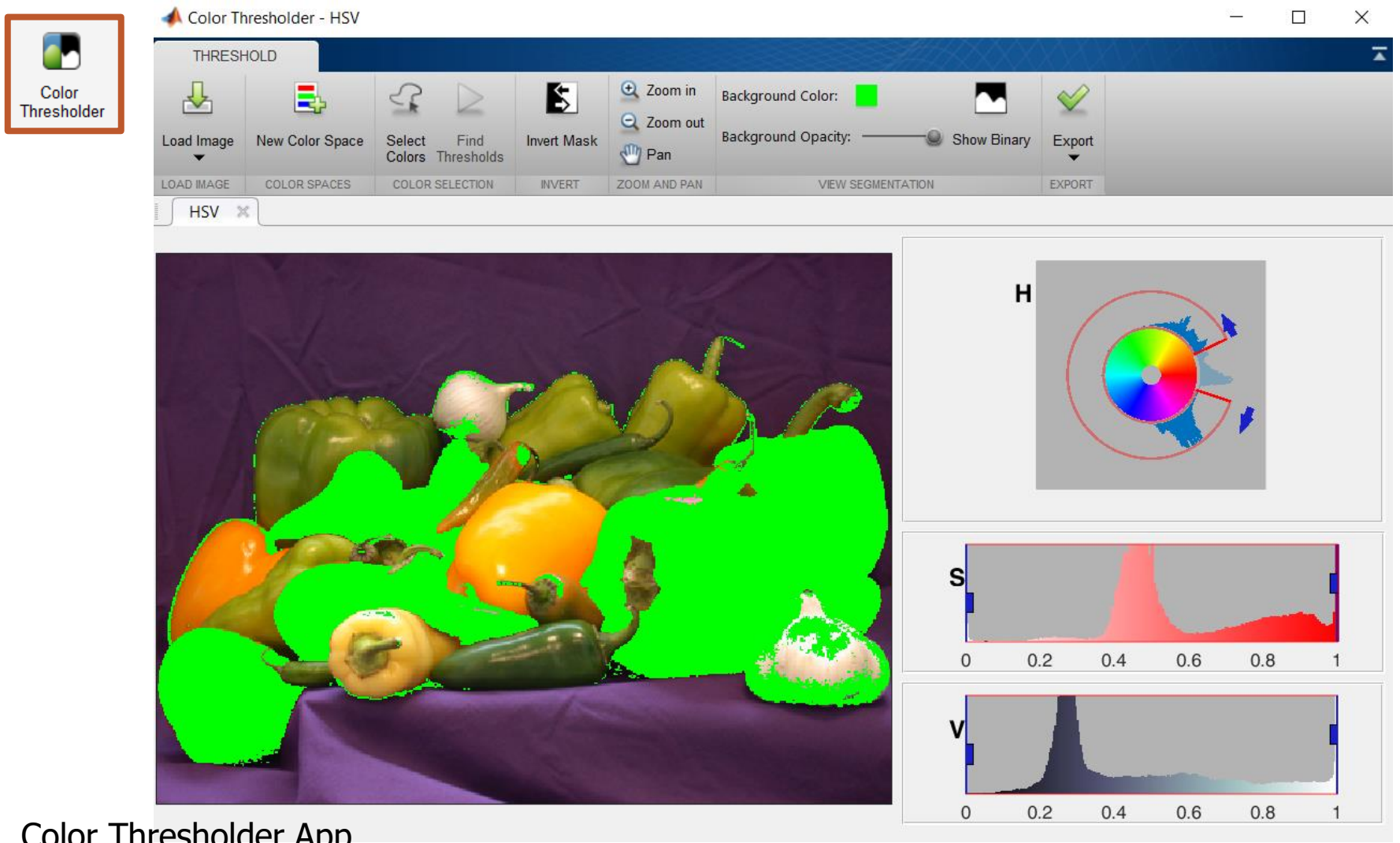


Image Segmenter App

MATLAB for Image Processing



Color Thresholder App

MATLAB for Image Processing

Morphological Operations

Dilate, erode, reconstruct, and perform other morphological operations

Functions

imerode	Erode image
imdilate	Dilate image
imopen	Morphologically open image
imclose	Morphologically close image
imtophat	Top-hat filtering
imbothat	Bottom-hat filtering
imclearborder	Suppress light structures connected to image border
imfill	Fill image regions and holes
bwhitmiss	Binary hit-miss operation
bwmorph	Morphological operations on binary images
bwmorph3	Morphological operations on binary volume
bwperim	Find perimeter of objects in binary image
bwskel	Reduce all objects to lines in 2-D binary image or 3-D binary volume
bwulterode	Ultimate erosion
strel	Morphological structuring element

MATLAB for Image Processing

Object Analysis

Detect edges, circles and lines; trace boundaries; perform quadtree decomposition

Functions

bwboundaries	Trace region boundaries in binary image
bwtraceboundary	Trace object in binary image
visboundaries	Plot region boundaries
imfindcircles	Find circles using circular Hough transform
viscircles	Create circle
edge	Find edges in intensity image
edge3	Find edges in 3-D intensity volume
imgradient	Find gradient magnitude and direction of 2-D image
imgradientxy	Find directional gradients of 2-D image
hough	Hough transform
houghlines	Extract line segments based on Hough transform
houghpeaks	Identify peaks in Hough transform

MATLAB for Image Processing

Region and Image Properties

Get information about the objects in an image

Functions

regionprops	Mida las propiedades de las regiones de imagen
regionprops3	Mida las propiedades de las regiones de imagen volumétrica 3-D
bwarea	Área de objetos en la imagen binaria
bwareaopen	Quite los objetos pequeños de la imagen binaria
bwareafilt	Extraiga objetos de la imagen binaria por tamaño
bwconncomp	Encuentre los componentes conectados en la imagen binaria
bwconvhull	Generar imagen de casco convexo de imagen binaria
bwdist	La transformación de distancia de la imagen binaria
bwdistgeodesic	La transformada de distancia geodésica de imagen binaria
bweuler	El número de Euler de imagen binaria
bwferet	Las propiedades de Measure Feret
bwperim	Encuentre el perímetro de los objetos en la imagen binaria
bwpropfilt	Extraiga objetos de una imagen binaria utilizando propiedades
bwlabel	Etiquetar componentes conectados en imágenes binarias en 2-D

MATLAB for Image Processing

Deep Learning with Images

Train convolutional neural networks from scratch or use pretrained networks to quickly learn new tasks

Apps

Deep Network Designer	Design, visualize, and train deep learning networks

Functions

Train Network

trainingOptions	Options for training deep learning neural network
trainNetwork	Train neural network for deep learning
analyzeNetwork	Analyze deep learning network architecture

Pretrained Networks

squeezenet	SqueezeNet convolutional neural network
googlenet	GoogLeNet convolutional neural network
inceptionv3	Inception-v3 convolutional neural network
densenet201	DenseNet-201 convolutional neural network
mobilenetv2	MobileNet-v2 convolutional neural network

MATLAB for Image Processing

Object Detection Using Features

Detect faces and pedestrians, create customized detectors

Functions

Detectors

ocr	Recognize text using optical character recognition
readAprilTag	Detect and estimate pose for AprilTag in image
readBarcode	Detect and decode 1-D or 2-D barcode in image
acfObjectDetector	Detect objects using aggregate channel features
peopleDetectorACF	Detect people using aggregate channel features
vision.CascadeObjectDetector	Detect objects using the Viola-Jones algorithm
vision.ForegroundDetector	Foreground detection using Gaussian mixture models
vision.PeopleDetector	Detect upright people using HOG features
vision.BlobAnalysis	Properties of connected regions

Create Custom Object Detectors

trainACFObjectDetector	Train ACF object detector
trainCascadeObjectDetector	Train cascade object detector model
trainImageCategoryClassifier	Train an image category classifier

MATLAB for Image Processing

Object Detection using Deep Learning

Perform classification, object detection, transfer learning using convolutional neural networks (CNNs, or ConvNets)

Detect Objects Using Deep Learning Detectors

rcnnObjectDetector	Detect objects using R-CNN deep learning detector
fastRCNNObjectDetector	Detect objects using Fast R-CNN deep learning detector
fasterRCNNObjectDetector	Detect objects using Faster R-CNN deep learning detector
ssdObjectDetector	Detect objects using SSD deep learning detector
yolov2ObjectDetector	Detect objects using YOLO v2 object detector
selectStrongestBbox	Select strongest bounding boxes from overlapping clusters
selectStrongestBboxMulticlass	Select strongest multiclass bounding boxes from overlapping clusters

Train Object Detectors

trainRCNNObjectDetector	Train an R-CNN deep learning object detector
trainFastRCNNObjectDetector	Train a Fast R-CNN deep learning object detector
trainFasterRCNNObjectDetector	Train a Faster R-CNN deep learning object detector
trainSSDObjectDetector	Train an SSD deep learning object detector
trainYOLOv2ObjectDetector	Train YOLO v2 object detector

MATLAB for Image Processing

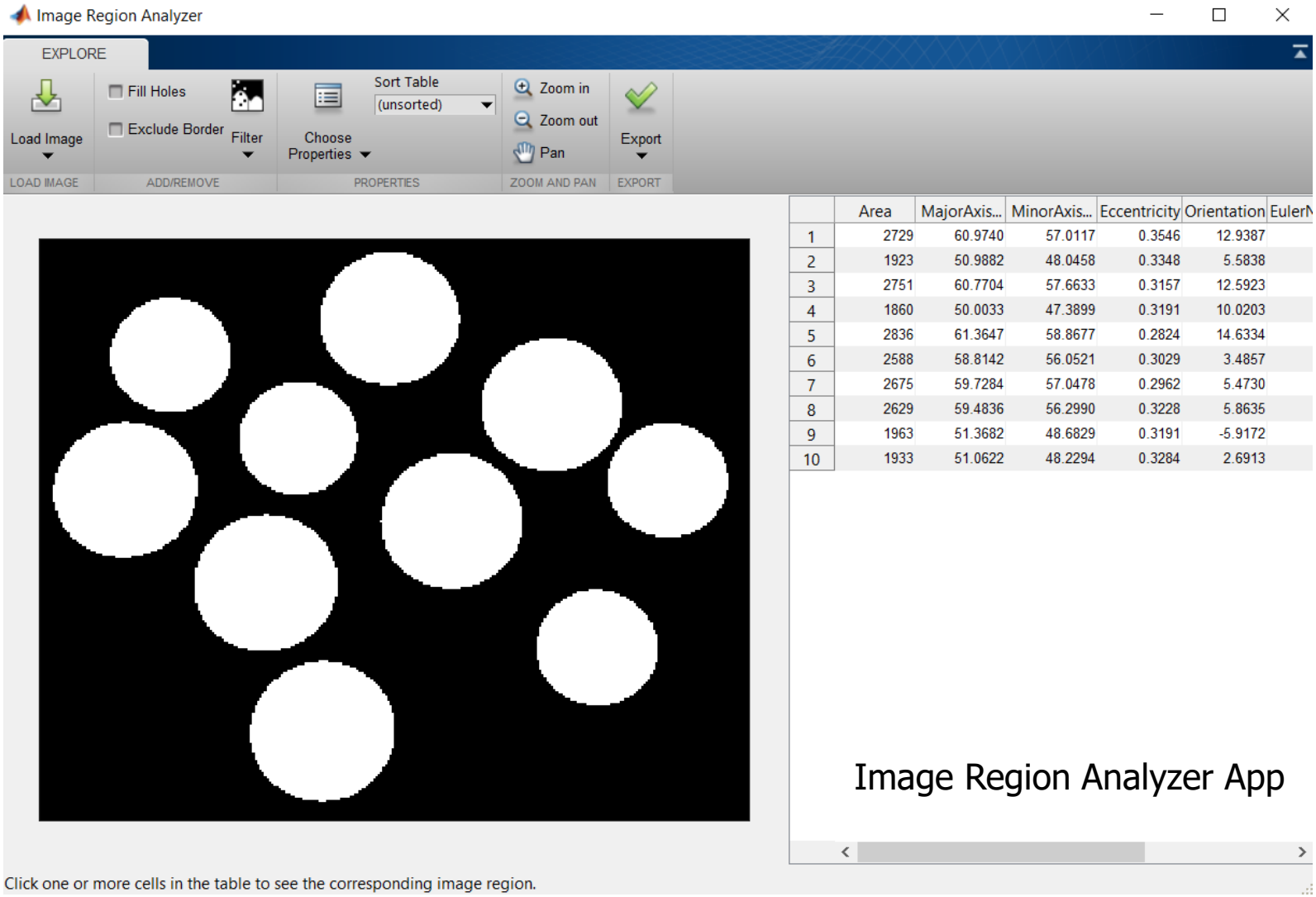


Image Region Analyzer App

MATLAB for Image Processing

Links

- MATLAB Documentation
  https://mathworks.com/help/index.html
- MATLAB Video Tutorials
  https://mathworks.com/videos.html
- MATLAB Examples
  https://mathworks.com/help/examples.html
- MATLAB Image Processing Toolbox
  https://mathworks.com/help/images/index.html
- MATLAB Examples: Image Processing
  https://mathworks.com/help/images/examples.html
- MATLAB Examples: Computer Vision
  https://mathworks.com/help/vision/examples.html

MATLAB for Image Processing

Video Tutorials

- Introduction to MATLAB with Image Processing Toolbox
  https://mathworks.com/videos/introduction-to-matlab-with-image-processing-toolbox-90409.html
- Image Processing Toolbox Overview
  https://mathworks.com/videos/image-processing-toolbox-overview-61214.html
- Image Processing Made Easy
  (eng) https://mathworks.com/videos/image-processing-made-easy-81718.html
  (spa) https://mathworks.com/videos/image-processing-made-easy-100161.html
- Visión Artificial de Manera Sencilla
  https://mathworks.com/videos/computer-vision-made-easy-1485791126907.html
- Image Acquisition and Processing with Matlab
  https://mathworks.com/videos/image-acquisition-and-processing-using-matlab-81586.html
- Procesamiento de Imágenes y Visión Artificial con MATLAB
  https://mathworks.com/videos/image-processing-and-computer-vision-with-matlab-1597884648964.html

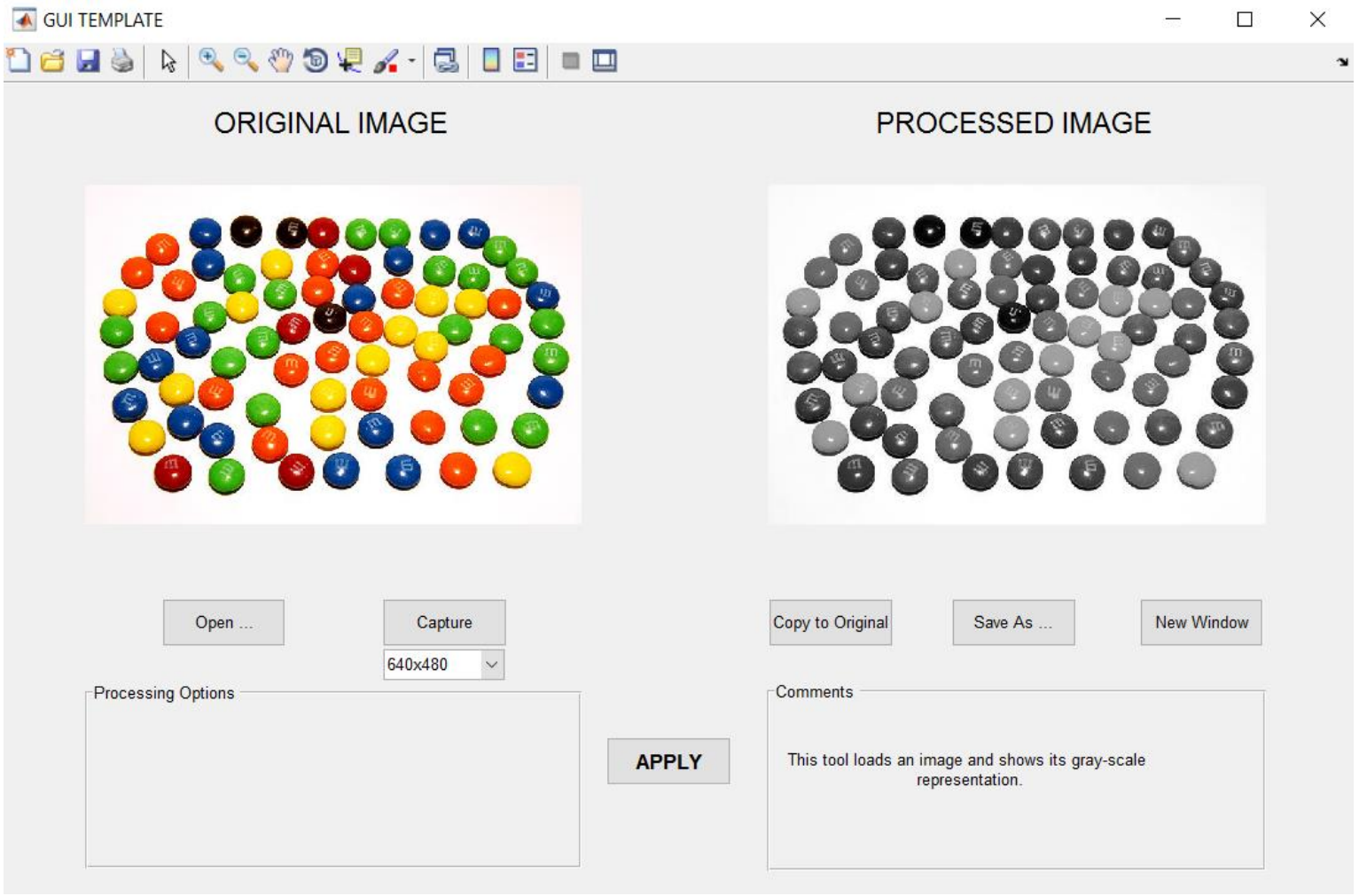
MATLAB for Image Processing

Video Tutorials

- Rapid Development of Image Processing Algorithms with MATLAB
  https://mathworks.com/videos/rapid-development-of-image-processing-algorithms-with-matlab-92910.html
- Video Processing in MATLAB
  https://mathworks.com/videos/video-processing-in-matlab-68745.html
- Computer Vision with MATLAB
  https://mathworks.com/videos/computer-vision-with-matlab-93068.html
- Computer Vision with MATLAB for Object Detection and Tracking
  (eng) https://mathworks.com/videos/computer-vision-with-matlab-for-object-detection-and-tracking-81866.html
  (spa) https://mathworks.com/videos/computer-vision-with-matlab-for-object-detection-and-tracking-82036.html
- Aprendizaje Profundo para Visión Artificial con MATLAB
  https://mathworks.com/videos/deep-learning-for-computer-vision-with-matlab-1540981496452.html

GUI EXAMPLE

GUI EXAMPLE



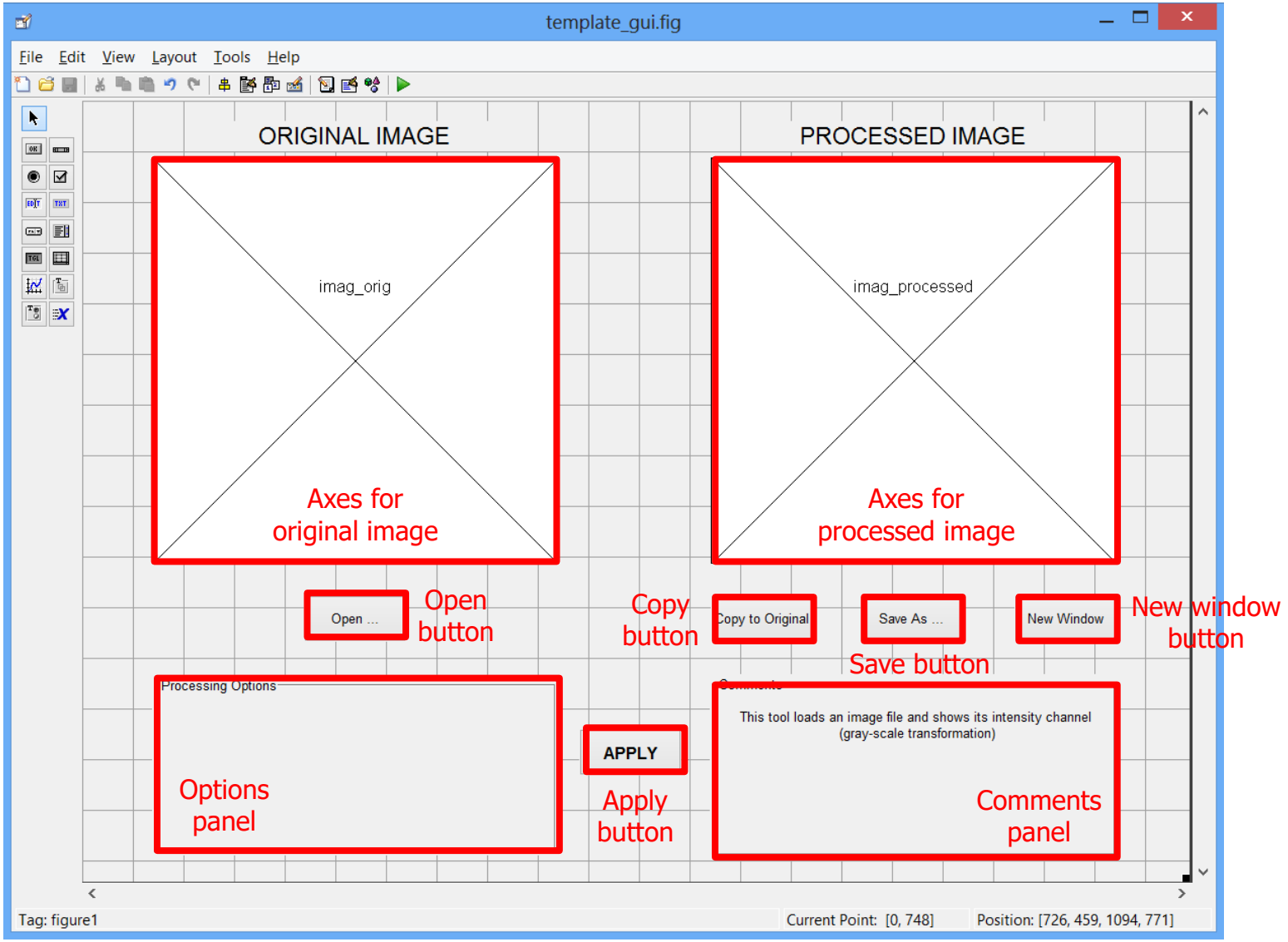
GUI_Example.fig

GUI EXAMPLE

Basic GUI to transform an image

- The GUI allows to open a file (original image) and shows its intensity component (processed image), e.g. converts it to gray-scale.
- It also will include other useful functions:
  - An area for processing options (useless in this case) and other for comments about the tool's aim, options, results, ….
  - Button for saving the processed image, copying it to the original image or to a new figure.
  - Capture from laboratory camera and possibility to select image resolution

GUI EXAMPLE


template_gui.fig
File Edit View Layout Tools Help
ORIGINAL IMAGE
PROCESSED IMAGE
imag_orig
imag_processed
Axes for original image
Axes for processed image
Open ...
Open button
Copy button
Copy to Original
Save As ...
New Window
New window button
Save button
Processing Options
Options panel
APPLY
Apply button
This tool loads an image file and shows its intensity channel (gray-scale transformation)
Comments panel
Tag: figure1
Current Point: [0, 748]
Position: [726, 459, 1094, 771]

GUI EXAMPLE

Callback of Open Button

Chooses an image file and shows it within the axes of original image

```
% --- Executes on button press in open_file.
function open_file_Callback(hObject, eventdata, handles)
% hObject    handle to open_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename, pathname]= uigetfile({'*.jpg;*.tif;*.png;*.gif','All Image Files'; '*.*','All Files' },'Choose an image file ...');

if isequal(filename,0)
   warndlg('File not selected');
   return
end

image=imread(strcat(pathname,filename));

axes(handles.imag_orig);
imshow(image);

set(handles.imag_orig,'UserData',image);
```

uigetfile: displays a dialog box where the user chooses a file in the file explorer. Returns the filename and path strings.

If the user do not select a file or the window closes, **warndlg** opens a warning window and the function finishes. It prevents from the errors when loading a non-existent file

Loads the image file into the variable *image*

Selects the axes of the original image and shows the image

Saves the image matrix into the field *UserData* of axis to be used later

GUI EXAMPLE

Callback of Apply Button

Applies the transformation by computing the intensity channel and shows it as the processed image

```
function apply_Callback(hObject, eventdata, handles)
% hObject    handle to apply (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

image=get(handles.imag_orig,'UserData');
image_d=double(image)/255;

image_proc=(image_d(:,:,1)+image_d(:,:,2)+image_d(:,:,3))/3; % intensity component

axes(handles.imag_processed);
imshow(image_proc);

set(handles.imag_processed,'UserData',image_proc);
```

Loads the matrix of the original image and converts it to decimal to operate

Computes the intensity

Selects the axes of the processed image and shows the image

Saves the image matrix into the field *UserData* of axis to be used later

GUI EXAMPLE

Callback of Copy Button

Copy the processed image to the original image

```
% --- Executes on button press in copy_orig.
function copy_orig_Callback(hObject, eventdata, handles)
% hObject    handle to copy_orig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

image=get(handles.imag_processed,'UserData');

axes(handles.imag_orig);
imshow(image);

set(handles.imag_orig,'UserData',image);
```

Loads the matrix of the processed image

Selects the left axes and shows the image

Saves the new image matrix into the field *UserData* of axis to be used later

GUI EXAMPLE

Callback of Save Button

Chooses a filename and save the processed image

```
% --- Executes on button press in saveas.
function saveas_Callback(hObject, eventdata, handles)
% hObject    handle to saveas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename, pathname]= uiputfile({'*.jpg;*.tif;*.png;*.gif','All Image Files'; ...
                                 '*.*','All Files' },'Save image file as ...');

image=get(handles.imag_processed,'UserData');

imwrite(image,strcat(pathname,filename));
```

uigetfile: displays a dialog box where the user chooses a file in the file explorer. Returns the filename and path strings.

Loads the matrix of the processed image

imwrite: Saves the image matrix into the selected file

GUI EXAMPLE

Callback of New Window Button

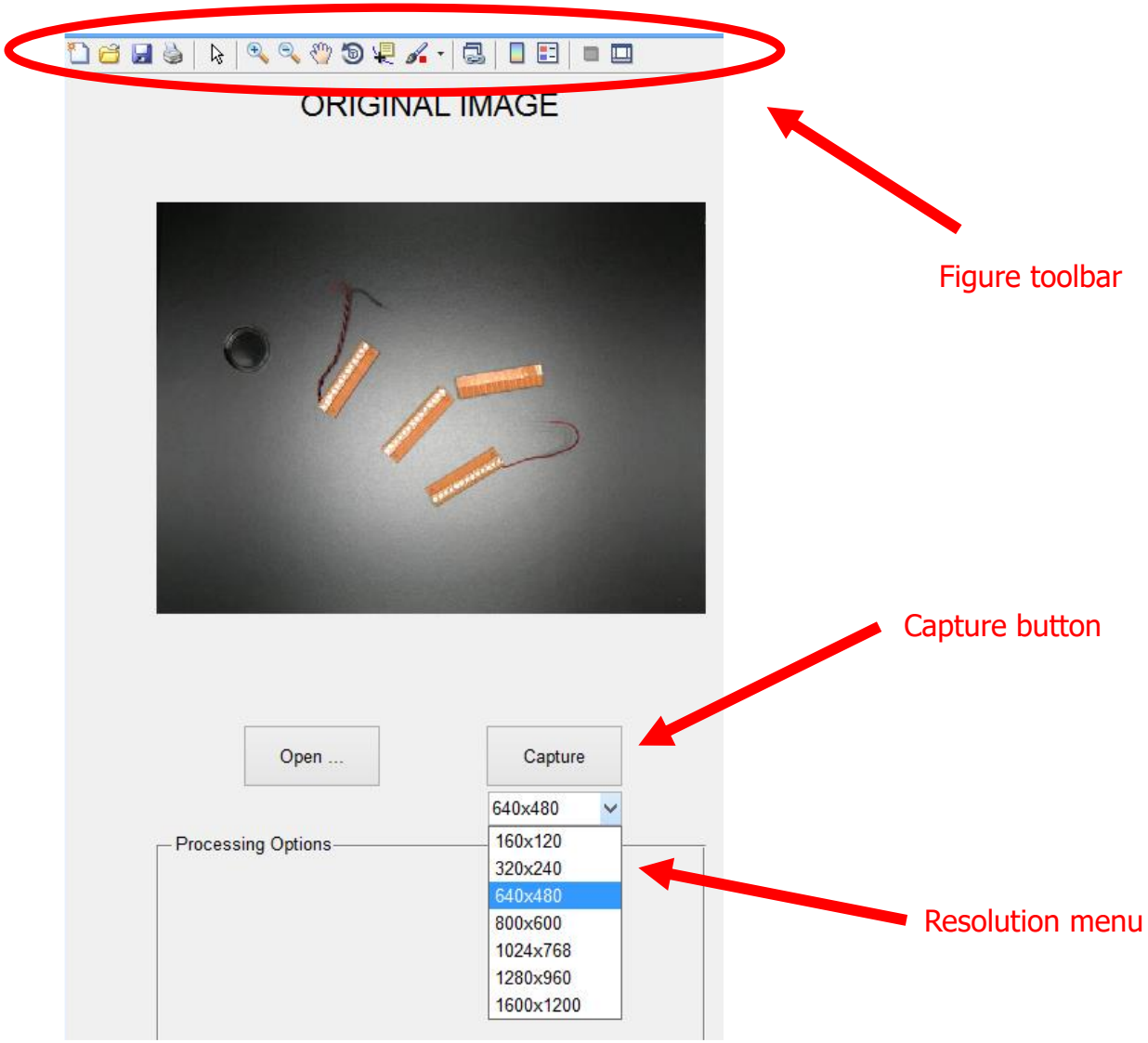
Open a new window showing the processed image

```
% --- Executes on button press in new_window.
function new_window_Callback(hObject, eventdata, handles)
% hObject    handle to new_window (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

image=get(handles.imag_processed,'UserData');

figure
imshow(image);
```

Loads the matrix of the processed image

Opens a new figure window and displays the image

GUI EXAMPLE


ORIGINAL IMAGE
Open ...
Capture
640x480
Processing Options
160x120
320x240
640x480
800x600
1024x768
1280x960
1600x1200
Figure toolbar
Capture button
Resolution menu

GUI EXAMPLE

- How to **capture** an image from the camera placed in the laboratory instead of opening a image file?

```
image=imread('http://visionartificia.gnd.upv.es/axis-cgi/jpg/image.cgi?resolution=640x480');
```

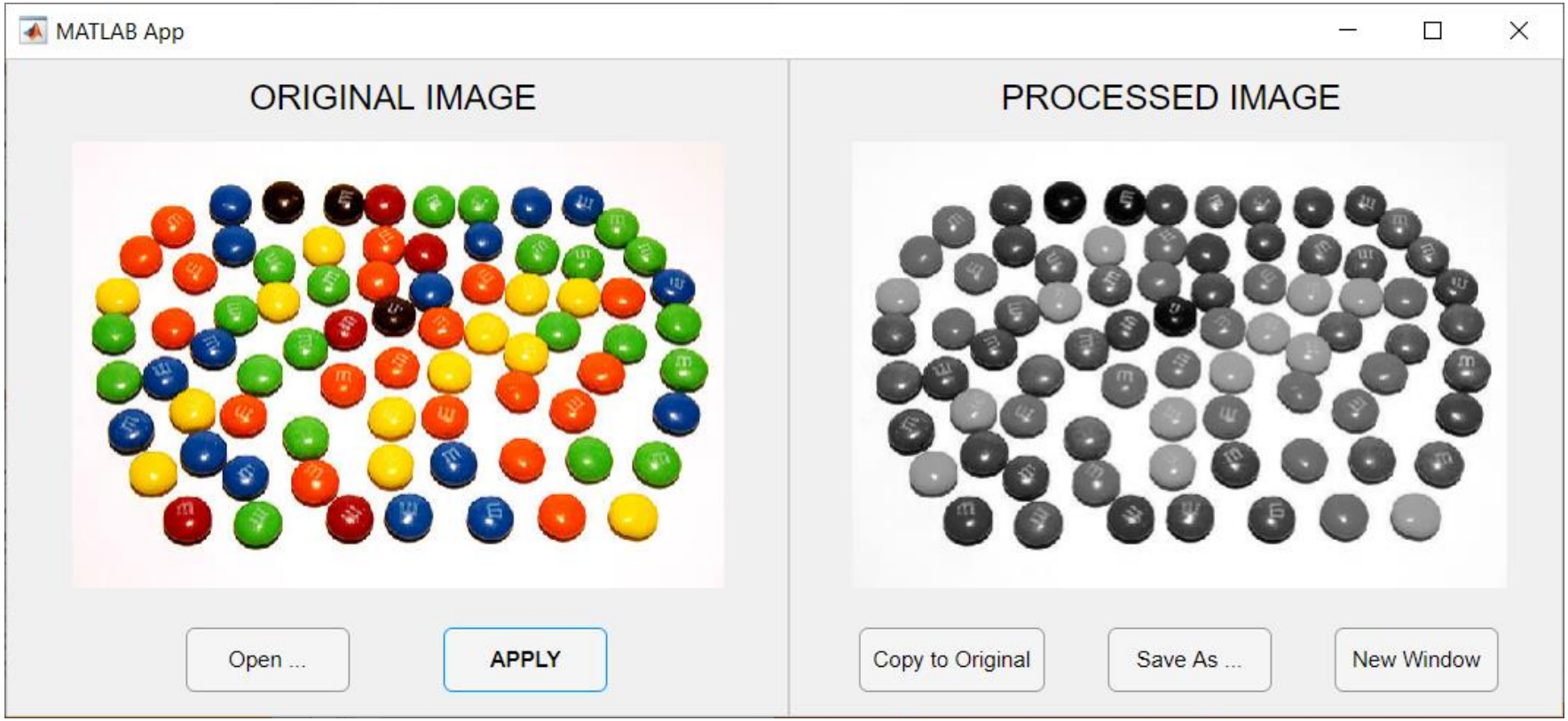
This command captures an image from that URL with the indicated resolution

- It includes a menu to choose the resolution of the captured image.

- The standard figure **toolbar** allows zooming, ….

APP EXAMPLE

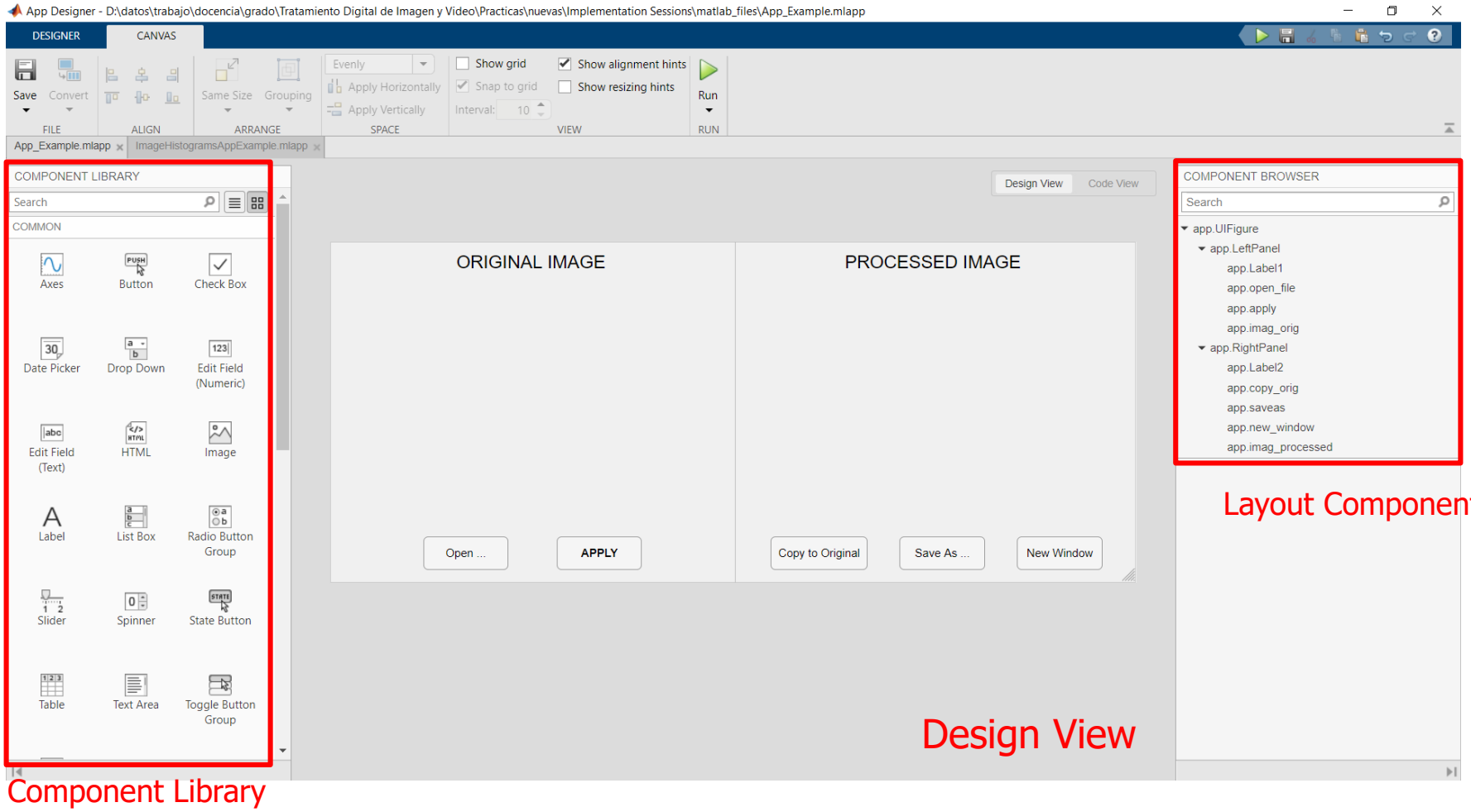
APP EXAMPLE



App_Example.mlapp

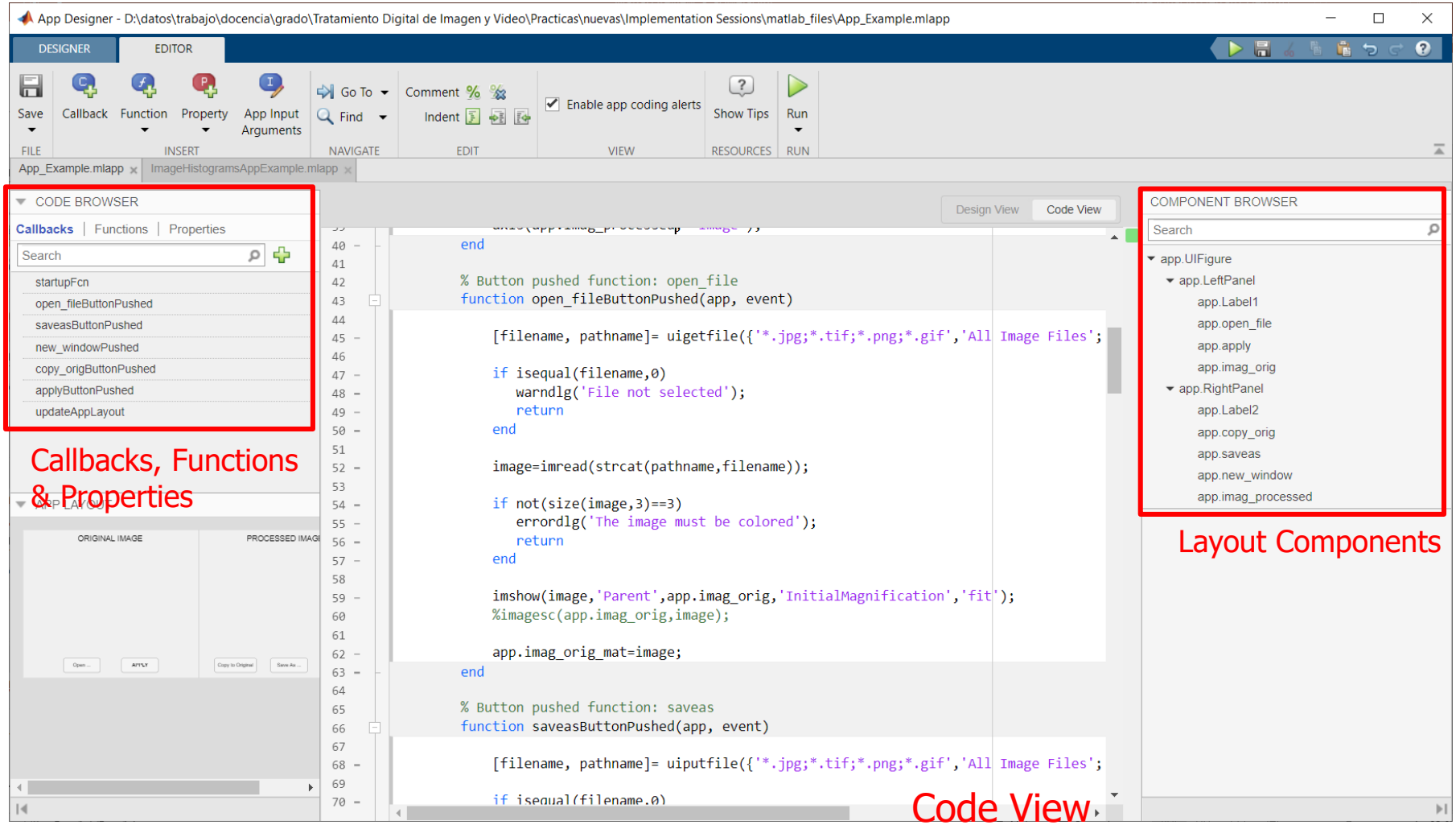
APP EXAMPLE

App designer



APP EXAMPLE

App designer



APP EXAMPLE

App developer

- Develop Apps Using App Designer
  https://mathworks.com/help/matlab/app-designer.html
- Create and Run a Simple App Using App Designer
  https://mathworks.com/help/matlab/creating_guis/create-a-simple-app-or-gui-using-app-designer.html
- App Building Components
  https://mathworks.com/help/matlab/creating_guis/choose-components-for-your-app-designer-app.html
- App Examples
  https://mathworks.com/help/matlab/examples.html?category=app-designer&s_tid=CRUX_topnav
- Video Tutorial
  https://mathworks.com/support/search.html/videos/matlab-and-simulink-robotics-arena-building-apps-with-matlab-and-app-designer-1513378634144.html

MATLAB GUIs Creation (GUIDE)

MATLAB GUIs

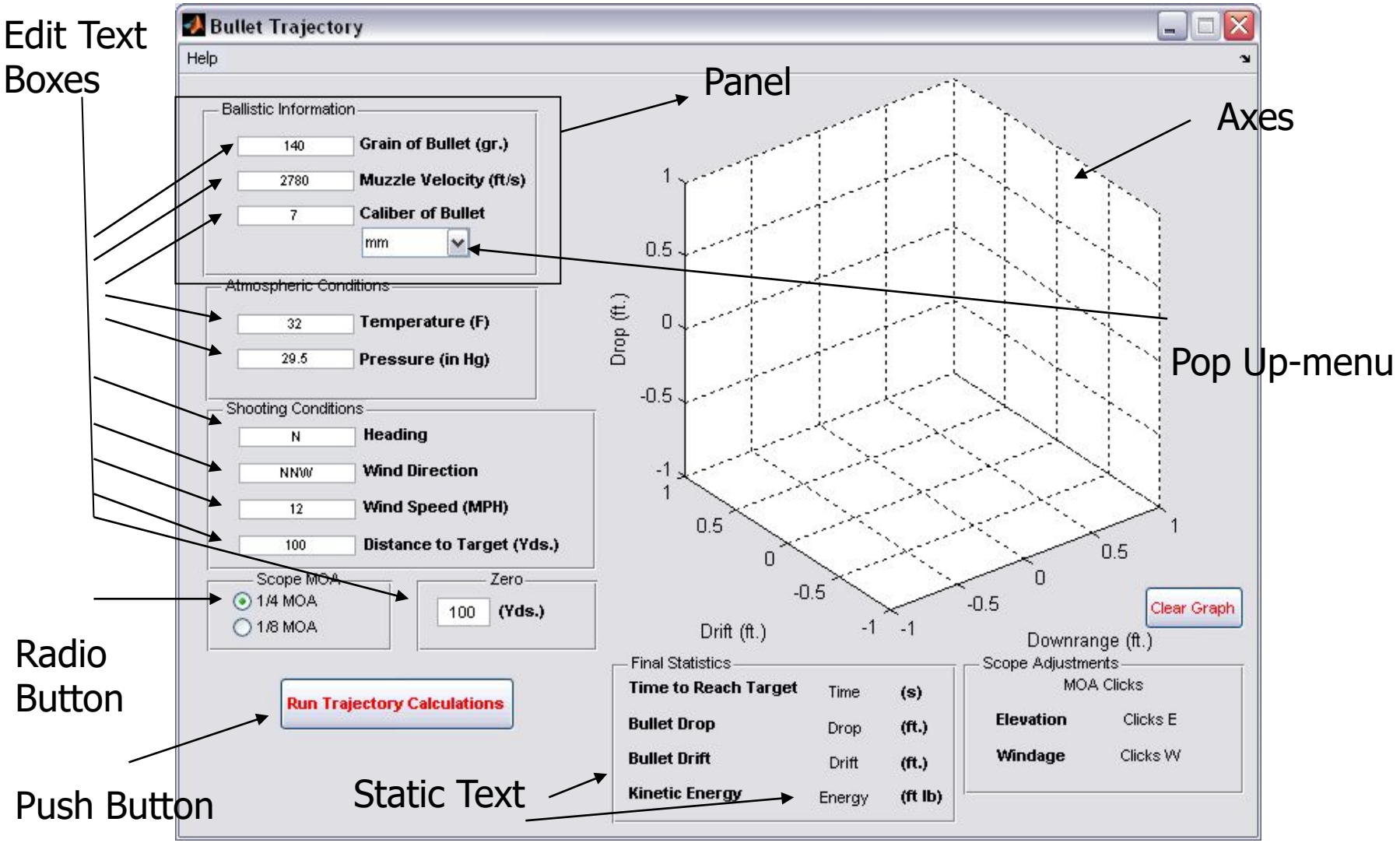
Introduction

GUI's in MATLAB consist of two files:

- An "m-file" [******.m]
- A "fig-file" [******.fig]

- The **m-file** has all of the code that controls the GUI
- The **fig-file** has all of the graphical objects, positions, default values, and links it all together

MATLAB GUIs


Edit Text Boxes
Panel
Axes
Pop Up-menu
Radio Button
Push Button
Static Text
Bullet Trajectory
Help
Ballistic Information
140
Grain of Bullet (gr.)
2780
Muzzle Velocity (ft/s)
7
Caliber of Bullet
mm
Atmospheric Conditions
32
Temperature (F)
29.5
Pressure (in Hg)
Shooting Conditions
N
Heading
NNW
Wind Direction
12
Wind Speed (MPH)
100
Distance to Target (Yds.)
Scope MOA
1/4 MOA
1/8 MOA
Zero
100
(Yds.)
Run Trajectory Calculations
Drop (ft.)
Drift (ft.)
Downrange (ft.)
Clear Graph
Final Statistics
Time to Reach Target
Time
(s)
Bullet Drop
Drop
(ft.)
Bullet Drift
Drift
(ft.)
Kinetic Energy
Energy
(ft lb)
Scope Adjustments
MOA Clicks
Elevation
Clicks E
Windage
Clicks W

MATLAB GUIs

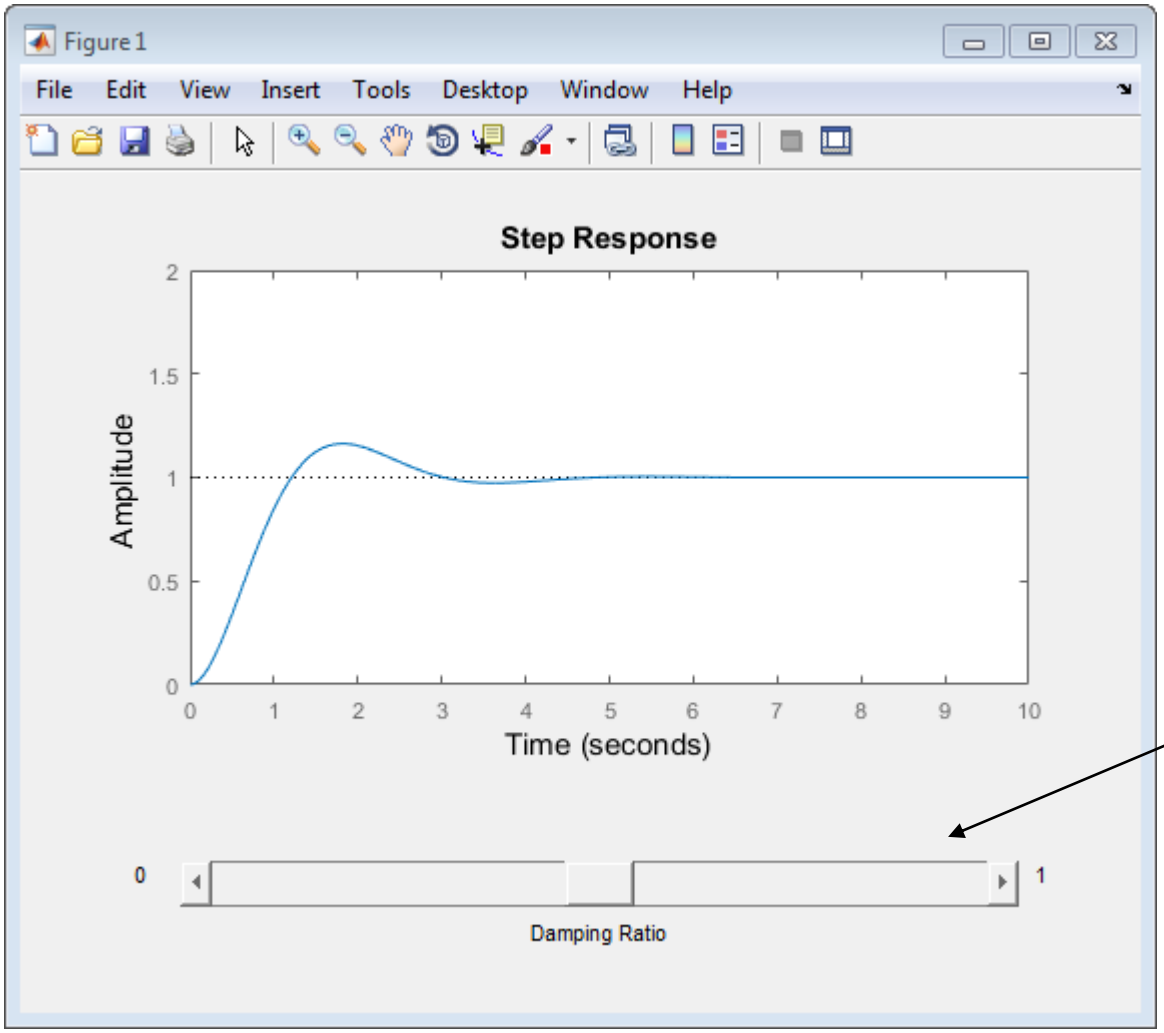
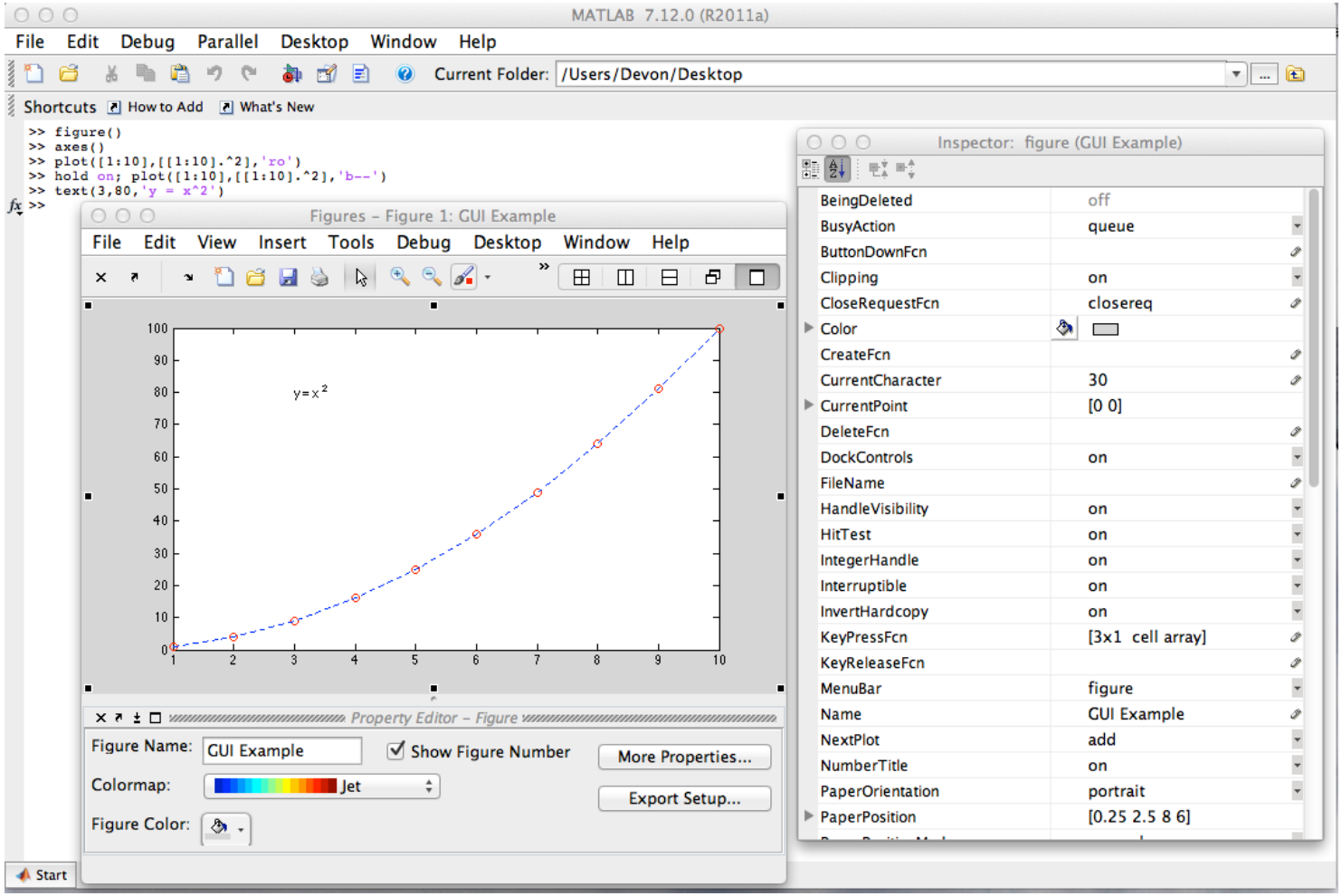


Figure 1
File Edit View Insert Tools Desktop Window Help
Step Response
Amplitude
Time (seconds)
0 1 2 3 4 5 6 7 8 9 10
0 0.5 1 1.5 2
0
1
Damping Ratio
Slider

MATLAB GUIs



Each component has a list of properties that define what it looks like and how it behaves.

MATLAB GUIs

Introduction

- The manipulation of these properties form the basis of GUIs and GUI programming in MATLAB.
- The ‘set’ and ‘get’ commands
  - These are the primary commands that you use to set and get information about graphic objects, they update the graphical object immediately
  - Syntax: ‘set’

```
>> set(object_hndl,'PropertyName',propvalue);
```

  - Syntax: ‘get’

```
>> propvalue = get(object_hndl,'PropertyName');
```

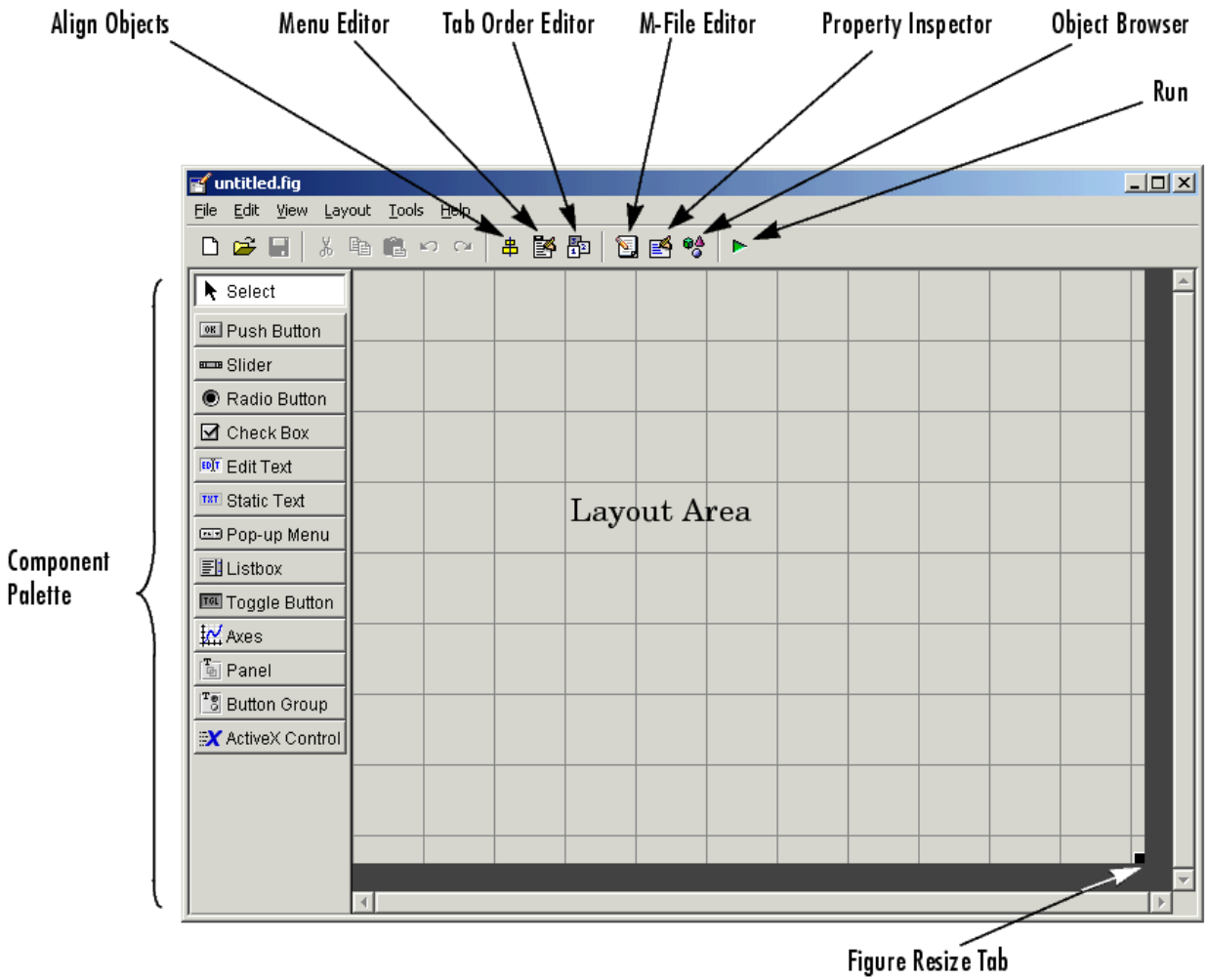
MATLAB GUIs

Creating GUIs

- Step 1: Create the graphical components
  - Using the GUIDE
  - Manually configure each component
- Step 2: Program components
  - The GUIDE will generate the primary files (*.m and *.fig).
  - The programmer must add to this *.m file all of the actions your components will take.

MATLAB GUIs

Step 1: GUIDE



MATLAB GUIs

Step 1: GUIDE

Tool	Use
Layout Editor	Select components from the component palette, at the left side of the Layout Editor, and arrange them in the layout area.
Figure Resize Tab	Set the size at which the GUI is initially displayed when you run it.
Menu Editor	Create menus and context, i.e., pop-up, menus.
Align Objects	Align and distribute groups of components.
Tab Order Editor	Set the tab and stacking order of the components in your layout.
Property Inspector	Set the properties of the components in your layout. It provides a list of all the properties you can set and displays their current values.
Object Browser	Display a hierarchical list of the objects in the GUI.
Run	Save and run the current GUI.
M-File Editor	Display, in your default editor, the M-file associated with the GUI.

MATLAB GUIs

Step 2: Editing m-file

Section	Description
Comments	Displayed at the command line in response to the help command. Edit these as necessary for your GUI.
Initialization	GUIDE initialization tasks. Do not edit this code.
Opening function	Performs your initialization tasks before the user has access to the GUI.
Output function	Returns outputs to the MATLAB command line after the opening function returns control and before control returns to the command line.
Component and figure callbacks	Control the behavior of the GUI figure and of individual components. MATLAB calls a callback in response to a particular event for a component or for the figure itself.

MATLAB GUIs

Step 2: Editing m-file

Callbacks: A function associated with a GUI component. It controls the behavior by performing an action in response to an event.

- Most callbacks will look similar to this:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

- The user's code must be added after the last comment
- Input Arguments:
  - hObject → Handle of the object, e.g., the component
  - eventdata → Reserved for later use.
  - handles → Structure that contains the handles of all the objects in the figure. It may also contain application-defined data.

MATLAB GUIs

Step 2: Editing m-file

Callbacks: A function associated with a GUI component. It controls the behavior by performing an action in response to an event.

Adding Callbacks:

- The GUIDE only creates the most common callbacks
- You may want to create different ones:
  - Right-click on the component you want
    - You are now looking at the Layout Editor context menu
  - Select **View callbacks**
    - Select the callback you wish to create
    - The GUIDE will now add it to the m-file and open it.

MATLAB GUIs

Step 2: Editing m-file

Set & Get:

```
user_data = get(hObject,'PropertyName');
user_data = set(hObject,'PropertyName',value);
```

(only works when you are inside the callback function you are trying to "get" or "set" the value of)

```
user_data = get(handles.component,'PropertyName');
user_data = set(handles.component,'PropertyName',value);
```

(access or modify data stored in a component somewhere else in your program)

- It is most likely to use the commands *num2str* and *str2double* with get and set functions:

MATLAB GUIs

GUI Components

- Push Button
- Toggle Button
- Radio Button
- Check Box
- Edit Text
- Slider
- List Box
- Pop-Up Menu
- Panel
- Button Group
- Axes
- ActiveX Control

MATLAB GUIs

GUI Components

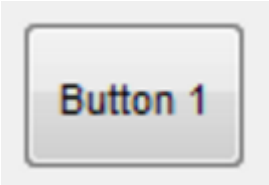
Commonly Used Properties

Tag property: component ID

Property	Value	Description
`Enable`	on, `inactive`, `off`. Default is on.	Determines whether the control is available to the user
`Max`	Scalar. Default is 1.	Maximum value. Interpretation depends on the type of component.
`Min`	Scalar. Default is 0.	Minimum value. Interpretation depends on the type of component.
`Position`	4-element vector: [distance from left, distance from bottom, width, height].	Size of the component and its location relative to its parent.
`String`	Character vector (for example, `'button1'`). Can an also be a character array or a cell array of character vectors.	Component label. For list boxes and pop-up menus it is a list of the items.
`Units`	`characters`, `centimeters`, `inches`, `normalized`, `pixels`, `points`. Default is `characters`.	Units of measurement used to interpret the `Position` property vector
`Value`	Scalar or vector	Value of the component. Interpretation depends on the type of component.

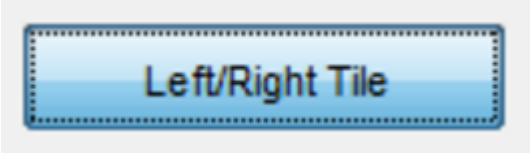
MATLAB GUIs

Push Button



Runs its Callback function when pressed

Toggle Button



MATLAB sets the *Value* property equal to the *Max* property when the toggle button is pressed (*Max* is 1 by default) and equal to the *Min* property when the toggle button is not pressed (*Min* is 0 by default).

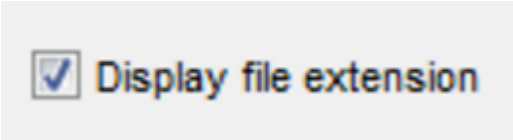
MATLAB GUIs

Radio Button

The current state of a radio is stored in its *Value* property, being its *Max* property (1 by default) when selected and its *Min* property (0 by default) otherwise.

(A button group can be used to mange exclusive selection behavior for radio buttons and toggle buttons)

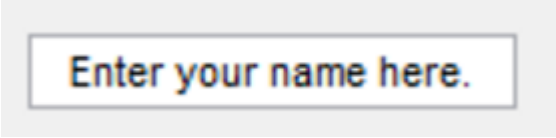
Check Box



The *Value* property is equal to its *Max* property (1 by default) when selected and its *Min* property (0 by default) otherwise.

MATLAB GUIs

Edit Text

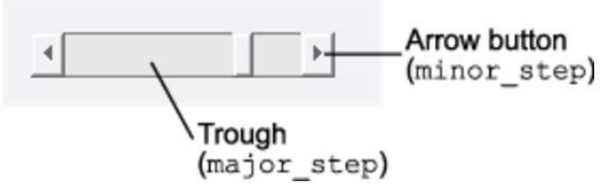


The typed text in an edit box is stored in the *String* property.

(MATLAB returns a character string. If a numeric value is needed, one must convert the characters to numbers. It can be done by using the str2double command. If the user enters nonnumeric characters, str2double returns NaN.)

Slider

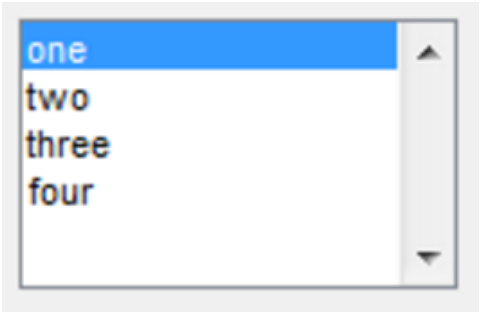
The value (position) of a slider is stored in its *Value* property. The *Max* and *Min* properties specify the slider's maximum and minimum values. The slider's range is Max - Min.



The *Step* property (2 values) determines the displacement of the slider when clicked

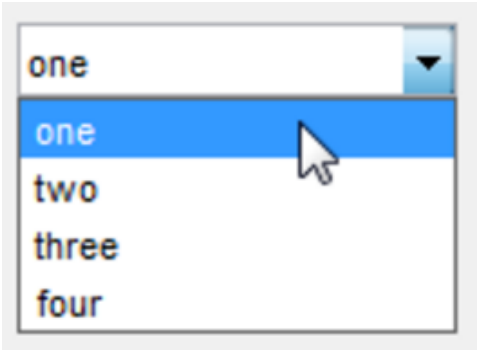
MATLAB GUIs

List Box



The list box *Value* property contains the index of the selected item, where 1 corresponds to the first item in the list. The *String* property contains the list as a cell array of strings.

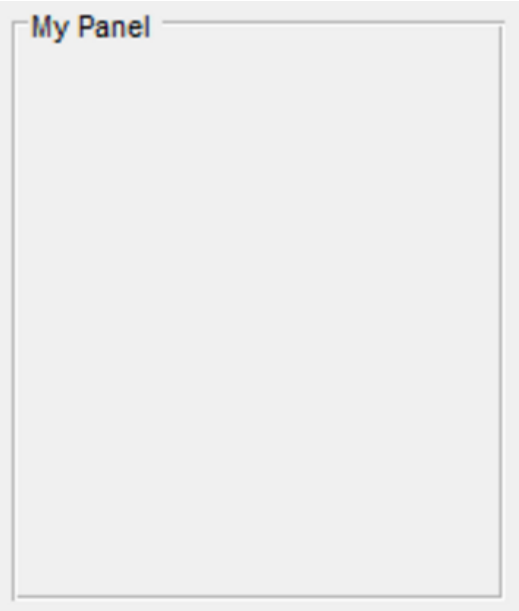
Pop-Up Menu



The pop-up menu *Value* property contains the index of the selected item, where 1 corresponds to the first item on the menu. The *String* property contains the menu items as a cell array of strings.

MATLAB GUIs

Panel



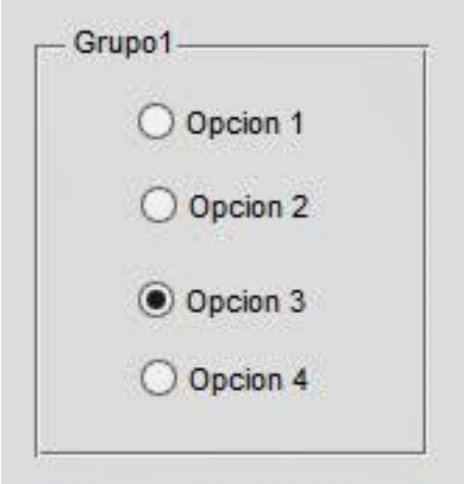
Panels group GUI components and can make a GUI easier to understand by visually grouping related controls.

A panel can contain panels and button groups as well as axes and user interface controls such as push buttons, sliders, pop-up menus, etc. The position of each component within a panel is interpreted relative to the lower-left corner of the panel.

Generally, if the GUI is resized, the panel and its components are also resized. However, you can control the size and position of the panel and its components. You can do this by setting the GUI Resize behavior to Other (Use ResizeFcn) and providing a ResizeFcn callback for the panel.

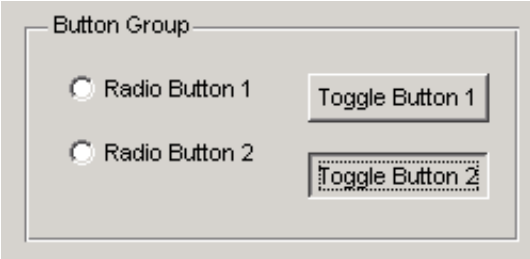
MATLAB GUIs

Button Group



Button groups are like panels except that they manage exclusive selection behavior for radio buttons and toggle buttons.

If a button group contains a set of radio buttons, toggle buttons, or both, the button group allows only one of them to be selected. When a user clicks a button, that button is selected and all others are deselected.



The button group's SelectionChangeFcn callback is called whenever a selection is made. Its hObject input argument contains the handle of the selected radio button or toggle button.