

MATLAB Tutorial

CSE 6367: Computer Vision

Instructor: William J. Beksi

Introduction

- **MATLAB** (MAtrix LABoratory) is a numerical computing environment that allows matrix manipulations, plotting of functions and data, algorithm implementation, etc.
- Toolboxes, such as the **Image Processing Toolbox** and the **Computer Vision System Toolbox**, provide a comprehensive set of algorithms and apps for image processing and computer vision

MATLAB Environment

- The MATLAB environment consists of the following components:
 - Workspace - Displays all of the defined variables
 - Command Window - Allows for executing commands in the MATLAB environment
 - Command History - Shows a record of the commands used
 - File Editor Window - Allows for defining functions, running scripts, etc.

MATLAB Help

- MATLAB Help is a valuable resource for learning MATLAB
- Help not only contains theoretical background, but also shows demos and implementation examples
- Any command description can be found by typing the command in the search field, e.g. `sqrt`

MATLAB Workspace

- The following commands are useful in the MATLAB workspace:
 - `who`, `whos` - Show the current variables in the workspace
 - `save` - Save the workspace variables to a *.mat file
 - `load` - Load the variables from a *.mat file
 - `clear` - Clear the workspace variables

Matrices

- The matrix is the main data type in MATLAB
- `A = [1 2 3; 4 5 6; 7 8 9];` creates a matrix A of size 3×3
- Special matrices include the following: `zeros(n,m)`, `ones(n,m)`, `eye(n,m)`, `rand()`, `randn()`

Basic Operations on Matrices

- All operations in MATLAB are defined on matrices: $+$, $-$, $*$, $/$, $^$, `sqrt`, `sin`, `cos`, etc.
- Element-wise operations are defined with a preceding dot: `.*`, `./`, `.^`
- To find the size of a matrix: `size(A)`
- To sum the columns of a matrix: `sum(A)`
- To sum all of the elements of a matrix: `sum(sum(A))`

Variable Names

- The following rules must be used when naming variables:
 - 63 characters in length
 - Begin with a letter
 - No blank spaces nor punctuation
 - May contain any combination of letters, digits, and underscores
- Variables are case sensitive and cannot be the same as MATLAB predefined variable names: `true`, `date`, `nan`, `eps`, `pi`, etc.

Logical Operators

- `==`, `<`, `>`, `~=` (not equal), `~` (not)
- `find('condition')` - Returns the indexes of the elements of a matrix that satisfy the given condition

- Example:

```
>> A = [4, 9, 1; 5, 8, 2]
```

```
A =
```

```
4 9 1
```

```
5 8 2
```

```
>> find(A < 3)
```

```
ans =
```

```
5
```

```
6
```

Flow Control

- MATLAB has five flow control constructs:
 - The `if` statement
 - The `switch` statement
 - The `for` loop
 - The `while` loop
 - The `break` statement

if

- The general form of the `if` statement is the following:

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

switch

- The general form of the `switch` statement is the following:

```
switch switch_expression
  case case_expression_1
    statements
  case case_expression_2
    statements
  otherwise
    statements
end
```

- Note that unlike C the `switch` statement does not fall through (i.e. `break`s are unnecessary)

for

- The for statement repeats a statement a specific number of times
- The general form of the for statement is the following:

```
for variable == expression
    statements
end
```

while

- The `while` statement repeats a statement an indefinite number of times
- The general form of the `while` statement is the following:

```
while expression
    statements
end
```

Scripts and Functions

- There are two kinds of m-files:
 - Scripts - Operate on data in the workspace and do not accept any input arguments nor do they return output arguments
 - Functions - Internal variables are local to the function, input arguments are accepted and output arguments are returned

Functions

- The following function, 'unpackRGBFloat', unpacks RGB float data into separate color values:

```
function [r g b] = unpackRGBFloat(rgbfloatdata)
% UNPACKRGBFLOAT Unpack RGB float data into
% separate color values.

mask = hex2dec('000000FF');
rgb = typecast(rgbfloatdata,'uint32');

r = double(bitand(mask, bitshift(rgb, -16))) / 255;
g = double(bitand(mask, bitshift(rgb, -8))) / 255;
b = double(bitand(mask, rgb)) / 255;

end
```


Visualization

- The following functions can be used for visualizing data:
 - `figure` - Open a new figure
 - `hold on`, `hold off` - Use the same figure or open a new figure
 - `axis([xmin xmax ymin ymax])` - Change scaling of axes
 - `title('figure_title')` - Add a title to the figure
 - `plot(x,y)` - 2D line plot
 - `scatter3(x,y,z)` - 3D scatter plot

save/clear/load

- The following session variables allow for saving, loading, and clearing the workspace:
 - `save mysession` - Creates a `mysession.mat` file with all the current variables
 - `clear all` - Clears all variables in the current session
 - `load mysession` - Loads a previously saved `mysession.mat` file

Image Processing Toolbox

- MATLAB's Image Processing Toolbox is a collection of functions that support a wide range of image processing operations including:
 - Geometric operations
 - Neighborhood and block operations
 - Linear filtering and filter design
 - Transforms
 - Image analysis and enhancement
 - Binary image operations
 - Region of interest operations

Images

- MATLAB can import/export several image formats including PNG, JPEG, TIFF, BMP, etc.
- Images consist of the following types:
 - Binary images - Each element is either 0 or 1
 - Intensity images - Each element takes on a value in the range $[0, 1]$
 - RGB images - Images are of the form $m \times n \times 3$

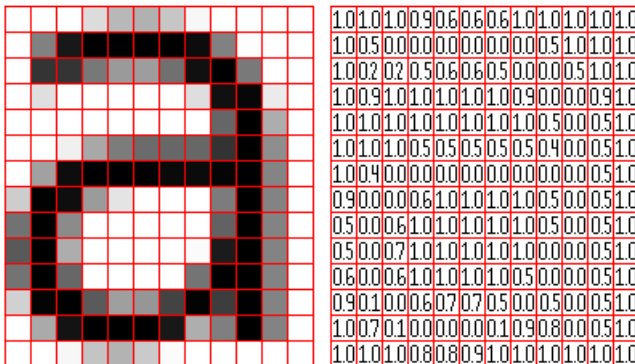
Image Import and Export

- The following commands can be used to read, display, and write images:

```
img = imread('cameraman.jpg');  
dim = size(img)  
figure;  
imshow(img);  
imwrite(img, 'output.png', 'png');
```

- Alternatives to `imshow` include `imagesc`, `imtool`, `image`

Images and Matrices



- Images are represented as matrices

Image Conversion Operations

- The following operations can be used to convert an image from one representation to another:
 - `rgb2gray` - Convert an RGB image to a grayscale image
 - `im2bw` - Convert an image to its binary representation
 - `mat2gray` - Convert a matrix to an intensity image
 - `im2uint8` - Convert an image to an 8-bit unsigned integer representation

Vectors vs. Loops

- MATLAB is fast on vector and matrix operations, but slow with loops
- Keypoint: Try to avoid loops and write vectorized code

Vectorized Loops

- Consider the following example to compute the sine for a range of values

- Poor style (elapsed time is 8.735 sec):

```
i = 0;  
for t = 0:.01:1000000  
    i = i + 1;  
    y(i) = sin(t);  
end
```

- Good style (elapsed time is 0.499 sec):

```
t = 0:.01:1000000  
y = sin(t);
```

Summary

- MATLAB is a great resource for prototyping image processing and computer vision software
- The toolboxes in MATLAB provide both low and high level functions for implementing programs