Lecture Machine Vision 2017/18

Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie
Dr. Martin Lauer, M.Sc. Jannik Quehl

Practical Exercises: MATLAB

These exercises are designed to get the first experiences with MATLAB. MATLAB is a programing environment for numerical calculations and algorithms and has been shown to be useful for quick prototyping.

Getting Started

After starting MATLAB a command line window with three subwindows will appear. The large subwindow on the right hand side is the command line window where you can type commands.

You can use MATLAB like a pocket calculator. Type and check the result:

```
>>  5*7
>>  2+3/2
```

You can also assign the result to a variable[1]:

```
>>  a = 4^2
>>  b = a + 2
```

You can suppress the output by entering a semicolon behind the command:

```
>>  c = b/2;
```

MATLAB provides all arithmetical and logical operators[2] you expect to find on a pocket calculator:

$+, -, *, /$	plus, minus, times, devided by	$3+4, 2*5$
ˆ	power	2ˆ8
log	natural logarithm	$log(3)$
exp	exponential function	$exp(1)$
abs	absolute value	$abs(-5)$
$sqrt$	square root	$sqrt(2)$
$sin, cos, tan, atan, ...$	sine, cosine, tangent, arctangent, ...	$sin(1), cos(pi), ...$
$==, \sim=$	comparision equal, unequal	$a == b, a \sim= b$
$<, <=, >, >=$	less, less or equal, greater, greater or equal	$a < b, ...$
&&, $\vert\vert, \sim$	logical operations and, or, not	$(a < 1)\vert\vert(a > 5)$

[1]MATLAB does not require an explicit declaration of variables like other programing laguages; variables are generated automatically as soon as they are used

[2]logical (boolean) values are represented by 0 (=false) and 1 (=true). Any other number is also interpreted as 'true'

You can get a help page for all MATLAB commands typing `help <Commandname>`, e.g.:

$>>$ *help sin*

displays a help page that explains how to use the sine function in MATLAB.

Exercise:
Calculate the following terms with MATLAB:

$$\frac{4 \cdot 7 - 8}{3^4 - 1} = \underline{\hspace{3cm}}$$
$$e^{\sin 3.2} = \underline{\hspace{3cm}}$$
$$\frac{5 + \log \frac{37}{2.9 \cdot 1.7}}{9} = \underline{\hspace{3cm}}$$

Matrices and Vectors

MATLAB is specially designed to work with matrices and vectors. The matrix $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ can be generated with the matlab command:

$>>$ [1 2 3; 4 5 6]

The square brackets define a matrix. The rows of the matrix are separated by semicolon (;), the columns by comma (,) or blank. A vector is a matrix with only one row or only one column. E. g. the vector $(1, 0, 2, 0, 3, 0)$ can be generated with the command:

$>>$ [1 0 2 0 3 0]

MATLAB defines some operations on matrices. The most important ones are:

$+, -$	plus, minus	$A + B, A - B$
$*$	scaling and matrix multiplication	$s * A, A * B$
$'$	transposition	A'
$.*$	element-wise multiplication	$A. * B$
.ˆ	element-wise power	$A.\hat{}2$ or $2.\hat{}A$
inv	inverse of a square matrix	$inv(A)$
max, *min*	maximum or minimum element of a vector	$max(A), min(A)$
sum	sum of all elements of a vector	$sum(A)$
mean	the mean of a vector	$mean(A)$
median	the median of a vector	$median(A)$
std	standard deviation of the elements of a vector	$std(A)$
eig	Eigenvalues and Eigenvectors of square matrices	$[V, D] = eig(A)$

Moreover, most functions like sin, exp, etc. can also be used for vectors. Then, the functions are applied to each element of the vector individually. E.g.

$>>$ *exp*([2 3])

yields the same result as

$$>> \quad [exp(2) \;\; exp(3)]$$

Exercise:
Do the following calculations with MATLAB:

$$A = \begin{pmatrix} 3 & 5 & 1 \\ 2 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$
$$b = (-2, 1, -4)^T$$
$$c = A^{-1} b$$
$$c = \underline{\hspace{3cm}}$$

You can evaluate individual elements of a matrix with the index operator (parenthesis):

$$>> \quad A(2,3)$$

yields the matrix element in the second row and third column of matrix A

$$>> \quad b(3)$$

yields the third element of vector b

$$>> \quad A(2,:)$$

yields the second row of matrix A

$$>> \quad A(:,3)$$

yields the third column of matrix A

The function `length` yields the size of a vector, `size` yields the size of a matrix (number of rows, number of columns). With `zeros (r,c)` you can create a matrix with r rows and c columns with all elements being zero, `ones (r,c)` creates a $r \times c$ matrix with all elements being one.

Often it is helpful to have sequences of numbers like 1,3,5,7,... MATLAB offers a special mechanism to create these kind of sequences with a starting value k, an increment d and an end value n. The command to create such a sequence is `k:d:n`. If d is omitted, the increment is set to one. E.g. the vectors $(1, 3, 5, 7, 9, 11, 13)$ and $(5, 4, 3, 2, 1)$ can be generated with:

$$>> \quad [1 : 2 : 13], \quad [5 : -1 : 1]$$

Exercise:

- create a sequence from -3 to 5 with step width 0.1
- create a sequence from 10^{-3} to 10^5 with increments on a logarithmic scale, i.e. the resulting sequence should contain the numbers $10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5$. Thereto, combine sequence generation with the (element-wise) power function. Write down the MATLAB commands.

  ____
- calculate the following term: $\sum_{k=1}^{100} k \cdot (101 - k)$. Thereto, create an increasing sequence $\vec{u}$ and a decreasing sequence $\vec{v}$ and calculate $\vec{u} \cdot \vec{v}^T$. Write down the MATLAB commands and the result.

  ____

  ____

  ____

  Result: ________

2D-Plotting

MATLAB supports two-dimensional plots of points and functions. The plot command takes as arguments vectors of x- and y-coordinates. E.g. if you want to display a sine curve you must create some sample points first and give it to the plot command:

```
>> x = 0 : 0.01 : 2 * pi;   % create x-coordinates between 0 and 2π
>> y = sin(x);   % create the respective y-coordinates
>> plot(x, y, 'r-');   % plot the sine curve
```

The third argument of the plot command controls the style in which the points and curves are plotted, i.e. the color, the point style, and the line style. You can get a list of all style options with '`help plot`'. When calling the plot command a new figure is drawn and the old figure is removed. If you want to keep the old figure and add a new curve to it, call '`hold on`' first. To make a new plot command erase the old figure, call '`hold off`'. E.g. to add a cosine curve to a the sine plot call:

```
>> hold on
>> plot(x, cos(x), 'b--')
```

Exercise:

- plot the sinc function in the interval between -10 and 10 with a red solid line
- add to the same figure a plot of the functions $\frac{1}{\pi x}$ and $-\frac{1}{\pi x}$ with a black dotted line. Resize the figure to maximal y-values of ± 1.5 with the `axis`-command (see: `help axis`)
- add a single blue cross at the origin of the figure

M-Files

Often, you want to do the same calculation several times. Then it is very inconvenient to type all commands again and again. For this reason MATLAB offers the possibility to write script files which you can use to save sequences of commands and that you can execute several times. In MATLAB, these files must have the filename extension *.m* which indicates that the file contains a MATLAB program. You can create such a script file clicking (right mouse button) in the subwindow *Current Directory* and selecting the menu item *new/M-File*. Give it a meaningful name and start to edit the file. After finishing, you can execute the scriptfile typing its name (without extension *.m*).

Exercise:
Create a script file that executes all commands that are necessary to generate the figure from the exercise above. Close the old figure and execute the script to create the figure again.

Beside the possibility to write scripts which are executed as if the commands would be typed in the console there is the possibility to write user-defined functions which take arguments and return result values. An M-File that starts with the following lines is interpreted as function:

```
function [ return-values ] = function-name ( argument-list )
% Description of the function
```

The function-name must be identical to the name of the M-File. The subsequent comment line is used to describe the function. It is printed when 'help function-name' is called.

Like other programing languages, MATLAB has some built-in commands to implement conditional execution of statements and loops like *if*, *for*, and *while*. Here are some examples of functions that illustrate the use of these commands:

```
function [ s ] = signof (x)
% calculate the sign of x, i.e. -1 if x is negative, +1 if x is positive,
% and 0 if x=0
if x==0
    s=0;
elseif x>0
    s=1;
else
    s=-1;
end
```

```
function [ s ] = euclideanlength ( x )
% calculate the Euclidean length of vector x
s=0;
for i=1:length(x)
    s=s+x(i)*x(i);
end
s=sqrt(s);
```

```
function [ n ] = smallestsquare ( x )
% calculates the smallest square number that is larger than x
n=1;
while n*n<=x
    n=n+1;
end
```

Exercise:
Write a function in MATLAB that calculates the Euclidean distance between two vectors u and v: $\sqrt{\sum_i (u_i - v_i)^2}$

Image Processing

MATLAB contains some commands to do image processing. Greylevel images are stored as matrices. They can be imported from files, displayed, and manipulated. The most important commands are:

imread	read an image file from the working directory	h=double(imread ('flower.png'));
rgb2gray	convert pixel or image from RGB to grey values	g=rgb2gray(h);
imshow	display graylevel or RGB image	imshow(g);
	display graylevel image and rescale gray values to a range from black to white	imshow(g,[ ]);
fspecial	create convolution kernels like Gaussian filters, etc.	f=fspecial('gaussian',3,0.5);
imfilter	image filtering by convoluting the image with a filter	m=imfilter(g,f);
deconvwnr	apply Wiener deconvolution to an image	deconvwnr(m,f,0.1);
fft2	2D-Fourier transform	fft2(g);

Exercise:
Perform the following steps of image processing:

- read in the image 'kabel_salat.png' and display it
- calculate the size of the image, the minimal, maximal, and average grey value of the image.
  size=___________, minimum=______, maximum=______, average=______
- create a Gaussian filter mask of size 11 and $\sigma = 4$. Convolute the grey level image with the Gaussian filter mask and display the result
- use Wiener deconvolution to restore the original picture from the blurred one. Display the result. Vary the point-spread-function and the noise-to-signal-ratio of Wiener deconvolution. Use the Gaussian point-spread-function with larger and smaller values of sigma and compare the resulting images.
- copy the file 'fadeout.m' in your MATLAB directory. It contains a function that can be used to generate a *fading out* effekt, i.e. the boundary of an image becomes shaded. Apply the fade-out method to the blurred image and apply Wiener deconvolution afterwards. Compare the result with your previous results.