Computer Vision 2 — Exercise 0

Introduction to MATLAB (21.04.2016)

M.Sc. Francis Engelmann, Dr. Jörg Stückler
engelmann@vision.rwth-aachen.de, stueckler@vision.rwth-aachen.de

RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

Visual Computing Institute

RWTH AACHEN UNIVERSITY

Organization

- **Exercises**
  - Preferably 3 students per group
  - Coarsely every 2 weeks
  - Turn in the night before exercise slot
  - Exercises are not mandatory for exam participation (but strongly recommended)

- **Who has worked with MATLAB already?**

Got MATLAB?

- Numerical Programming Environment
  - CampusLicense for RWTH students
  - Download through CampusOffice Software Shop:
    - http://www.matlab.rwth-aachen.de

  - CampusLicense requires connection to license server from within the RWTH network - use VPN
  - To “borrow” a license directly from the server for offline work look into the MATLAB documentation for details

- Available also in CIP Pool (E1 & E2 building)
- MATLAB Tutorial, see course website:
  - http://www.vision.rwth-aachen.de/course/9/

Important Commands

`help`	– Get help for any command
`doc`	– Get help (help browser)
`lookfor`	– Search for keywords
`clear`/`clear x`	– Erase all variables/variable x
`close`/`close h`	– Close current figure/figure h
`clc`	– Clear command window
`whos`	– List variables in workspace
`save`	– Save the workspace
`load`	– Load a saved workspace
`keyboard`	– Enter debugging (until `dbquit`)

Useful Things to Remember

- Index always starts with 1 and not 0
- % is used for comments not # or //
- You can divide the code into different cells with %%
- While writing a long Matlab statement that becomes too long for a single line use "…" at the end of the line to continue on next line
- A semicolon (;) at the end of a statement means that Matlab will not display the result of the evaluated statement. For debugging it is useful to emit the semicolon to display the output

  (No need for `print` or `disp` command)
- Images are matrices, so x/y coordinates are flipped (compared to a standard coordinate system): first line (y), then column (x) when indexing image(-matrices)

Basic Operations

```
% Scalars
L = 2; C = 3;

% Basic operations
sum_ = L + C;
prod = L * C;

% functions
T = tan(L / C);
E = exp(L - C);
```

```
% For loop
sum_ = 0
for i = 1:100
    sum_ = sum_ + i;
end

% If statement
number = 13;
if isprime(number)
   disp('prime number');
else if odd(number)
   disp('odd number');
else
   disp('none of the above');
end
```

Everything is a Matrix

```
% Line vector
lv = [1 2 3];
lv = [1,2,3];
lv = 1:3; % from 1 to 3
lv = 1:1:3; % step size 1
lv = linspace(1,3,3);
```

```
>> lv
   lv =
     1     2     3
```

```
% Column vector
cv = [1;2];
cv = (1:2)'; % transpose
```

```
>> size(cv)
ans =
      2    1
>> cv
   cv =
     1
     2
```

```
% Different ways of defining
% the same matrix:
M = [1 2 3; 4 5 6];

M = zeros(2,3);
for l=1:L
   for c=1:C
      M(l,c) = ((l-1)*3)+c;
   end
end

M = reshape(1:L*C,C,L)';
```

```
>> M
   M =
     1     2     3
     4     5     6
```

Accessing Elements

```
>> M = [1 2 3; 4 5 6]
M =
     1     2     3
     4     5     6
% M(line,column)
>> M(1,3)
ans =
     3
>> M(1,end)
ans =
     3
>> M(5)
ans =
     3
>> M(1,end:-1:1)
ans =
     3     2     1
```

```
>> M(1,1:3)
ans =
     1     2     3
>> M(1,1:end)
ans =
     1     2     3
>> M(1,:)
ans =
     1     2     3
>> M(1:2,1:2)
ans =
     1     2
     4     5
>> M(M > 4)
ans =
     5
     6
```

Manipulating Matrices

```
>> A = [1 4;0 3]
A =
     1     4
     0     3
>> B = [1 0;0 -1]
B =
     1     0
     0    -1
% Matrix multiplication
>> A*B
ans =
     1    -4
     0    -3
% Elementwise multiplication
>> A.*B
ans =
     1     0
     0    -3
```

```
% Concatenation
>> C = [A B]
C =
     1     4     1     0
     0     3     0    -1
>> C = [A;B]
C =
     1     4
     0     3
     1     0
     0    -1
>> C = repmat(A,1,2)
ans =
     1     4     1     4
     0     3     0     3
```

Manipulating Matrices

```
>> A = [1 4;0 3]
A =
     1     4
     0     3
>> B = [1 0;0 -1]
B =
     1     0
     0    -1

>> A/B % = A * inv(B)
ans =
     1    -4
     0    -3
% Elementwise division
>> R = A./B
% 1/0 = Inf, 0/0=NaN
R =
     1   Inf
   NaN    -3
```

```
>> isinf(R)
ans =
     0     1
     0     0
>> isnan(R)
ans =
     0     0
     1     0
% Use logical indexing to
% replace NaN with 0
>> R(isnan(R))=0
R =
     1   Inf
     0    -3
% Find non-zero elements indices
>> find(R)
ans =
     1
     3
     4
```

Try to code in matrix ways

```
>> A = [1 2;3 4];
>> B = [1 1;2 2];

% With for loops
S = zeros(2);
for l = 1:2
   for c = 1:2
   S(l,c) = A(l,c) + B(l,c);
   end
end
% Better use matrix
C = A + B;

>> C =
     2     3
     5     6
```

```
% Matlab functions usually work
  on matrices, not only on
  scalars, for example:
C = A^2; %(matrix-multiplication)
D = sqrt(B); %(element-wise)
% Be careful which functions
  operate element-wise, on a
  line/column or on the whole
  matrix:
E = sum(A) %column-wise
>> E =
     4     6
E = sum(sum(A))
>> E =
     10
E = sum(A(:))
>> E =
     10
```

Scripts and Functions

- Scripts are .m-files containing MATLAB statements.
- Functions are like any other m-file, but they accept arguments.
- Name the function file should be the same as the function name if you want to call that function.
- Variables in a script file are global and will change the value of variables of the same name in the environment of the current Matlab session.
- A script with name script1.m can be invoked by typing “script1” in the command window.

Image-Specific Functions

- Three ways to display an image

- `imshow` - Display an image
  - ➢ `imshow(img);`
  - ➢ `imshow(img,[low high]);`

- `image` - Display a matrix as image
  - ➢ `image(mat);`
  - ➢ Elements of mat are used as indices into the current colormap.

- `imagesc` - Display a matrix as image
  - ➢ `imagesc(img);`
  - ➢ Same as `image`, but data is scaled to use the full colormap.

- `colormap` - Define the colormap to use
  - ➢ `map = colormap;`
  - ➢ `map1 = map(end:-1:1,:); colormap(map1);`

Some Nuisances

- Type issues
  - ➢ `imread` stores (most) images as type `uint8`
  - ➢ Many other matrix functions require type `double`
  - ➢ Solution: conversion

```
- double_img = im2double(img);
```

- Colormap issues
  - ➢ `colormap` works differently for different image formats (gif, jpg, png).

- If something doesn’t work as expected, one of the above issues is often the cause.

Other Visualization Functions

- `figure` - Open a new window or select an existing one
  - ➢ `figure;`
  - ➢ `figure(1);`
  - ➢ `h = figure; … figure(h);`
- `plot` - Plots one or more vectors on a x-y axis.
  - ➢ `plot(y);`
  - ➢ `plot(x,y);`
  - ➢ `plot(x,y,'b.-');`
  - ➢ `plot(x1,y1,'b.-', x2, y2,‘ro:',…);`

  - ➢ `figure; hold on;`
  - ➢ `plot(x1,y1,'b.-');`
  - ➢ `plot(x2,y2,'ro:');`
  - ➢ `hold off;`

Other Visualization Functions

- Variations on plots:
  - `plot3` (3D line plot)
  - `plotyy` (2 y-axes)
  - `semilogx`, `semilogy`, `loglog` (logarithmic axes)
- `bar` - Display a bar diagram
  - `bar(x,y);`
- `scatter` - Display a scatter plot
  - `scatter(x,y);`
  - `scatter(x,y,s,c);`
- Matlab is a great tool for such visualizations.

MATLAB: Statistical Functions

- Matlab provides built-in routines for common tasks
  - mean(X) - mean
  - var(X) - variance
  - hist(X) - plots a histogram of the vector
- Other useful functions
  - Eigen value decomposition/ SVD (Singular value decomposition)
  - Pre-defined filters : Gaussian, Laplacian, Sobel ...
- Special Matrix Operations
  - inv(M) % inverse of matrix M
  - ones(n, m) % matrix with n rows m column and all the entries ‘1’
  - zeros(n, m) %matrix with n rows m column and all the entries ‘0’
  - rand(n,m) % matrix with random numbers within the range of (0 ,1)
  - det(determinant), eye(identity matrix), norm, rank …

MATLAB is Different - Find the Mistake (1)

```
N = 5;
A = zeros(N,1);
for n = 0:N-1
   A(n) = n;
end
```

MATLAB is Different - Find the Mistake (1)

```
N = 5;
A = zeros(N,1);
for n = 0:N-1
  A(n) = n;
end
```

```
>> ??? Attempted to access
   A(0); index must be a
   positive integer or
   logical.
```

MATLAB is Different - Find the Mistake (1)

```
N = 5;
A = zeros(N,1);
for n = 0:N-1
  A(n) = n;
end
```

```
>> ??? Attempted to access
   A(0); index must be a
   positive integer or
   logical.
```

Indices start with 1!

```
N = 5;
A = zeros(N,1);
for n = 1:N
  A(n) = n-1;
end
```

MATLAB is Different - Find the Mistake (2)

```
A = 0;
for n = 1:N
   A(n) = n-1;
end
```

MATLAB is Different - Find the Mistake (2)

```
A = 0; N = 5;
for n = 1:N
   A(n) = n-1;
end
```

```
N = 5; A = zeros(N,1);
for n = 1:N
   A(n) = n-1;
end
```

This works! You can always extend a variable’s size or overwrite it with a new type.

But: New memory is allocated for the vector A in every loop iteration and this is time consuming. Better allocate enough memory from the start.

MATLAB is Different - Find the Mistake (3)

```
% #students in lectures
students = [20;40;20];
% #teachers in lectures
teachers = [1;4;2];
% #students per teacher
ratio = students/teachers;
```

MATLAB is Different - Find the Mistake (3)

```
% #students in lectures
students = [20;40;20];
% #teachers in lectures
teachers = [1;4;2];
% #students per teacher
ratio = students/teachers;
```

```
>> ratio =
   0      5      0
   0     10      0
   0      5      0
```

MATLAB is Different - Find the Mistake (3)

```
% #students in lectures
students = [20;40;20];
% #teachers in lectures
teachers = [1;4;2];
% #students per teacher
ratio = students/teachers;
```

```
>> ratio =
   0      5      0
   0     10      0
   0      5      0
```

```
% #students in lectures
students = [20;40;20];
% #teachers in lectures
teachers = [1;4;2];
% #students per teacher
ratio = students./teachers;
```

```
>> ratio =
   20
   10
   10
```

B/A Solves xA = B

A\B solves Ax = B

Use ./ for element-wise matrix operations.

MATLAB is Different - Find the Mistake (4)

```
figure;
imshow(img);
plot(box(1,:),box(2,:),’g‘);
```

MATLAB is Different - Find the Mistake (4)

```
figure;
imshow(img);
plot(box(1,:),box(2,:),‘g‘)
```

```
figure;
imshow(img);
hold on;
plot(box(1,:),box(2,:),‘g‘)
```

Box function draws a box around given coordinates of given dimension.

If you want to display several things in the same figure use `hold on`.

MATLAB is Different - Find the Mistake (5)

```
plot = figure;
imshow(img);
hold on;
plot(x,y,‘g’);
% save figure
imwrite(plot,‘plot1’,‘jpg’);
```

MATLAB is Different - Find the Mistake (5)

```
1 plot = figure;
2 imshow(img);
3 hold on;
4 plot(x,y,‘g‘);
5 % save figure
6 imwrite(plot,‘plot1‘,‘jpg‘);
```

```
??? Attempted to access
   plot(240,320); index out of
   bounds because numel(plot)=1.

Error in ==> example at 4
```

MATLAB is Different - Find the Mistake (5)

```
1 plot = figure;
2 imshow(img);
3 hold on;
4 plot(x,y,‘g’);
5 % save figure
6imwrite(plot,‘plot1’,‘jpg’);
??? Attempted to access
   plot(240,320); index out of
   bounds because
   numel(plot)=1.


Error in ==> example at 4
```

```
1 plot1 = figure;
2 imshow(img);
3 hold on;
4 plot(x,y,‘g’);
5 % save figure
6 imwrite(plot1,‘plot1’,‘jpg’);
```

It is not possible to use the same name for variables and functions.

(The variable name overwrites the function name and you cannot call the function later.)

MATLAB is Different - Find the Mistake (6)

```
for n = 1:N
   A = something(n);
   for n = 1:N
   display(n);
   display(A(n));
   end
end
```

MATLAB is Different - Find the Mistake (6)

```
for n = 1:N
   A = something(n);
   for n = 1:N
   display(n);
   display(A(n));
   end
end
```

```
for n = 1:N
   A = something(n);
   for m = 1:N
   display(m);
   display(A(m));
   end
end
```

The scope of a variable is the *whole* function.

Be careful with frequently used variable names. They may overwrite existing variables.

MATLAB is Different - Find the Mistake (7)

```
lines = n/m;
lines = int(lines);
vect = zeros(lines,1);
```

MATLAB is Different - Find the Mistake (7)

```
lines = n/m;
lines = int(lines);
vect = zeros(lines,1);
```

```
??? Undefined function or
   method 'int' for input
   arguments of type
   'double'.
```

MATLAB is Different - Find the Mistake (7)

```
lines = n/m;
lines = int(lines);
vect = zeros(lines,1);
```

```
??? Undefined function or
   method 'int' for input
   arguments of type
   'double'.
```

`int` is *not* the function for type conversion to integer.

```
lines = n/m;
lines = uint8(lines);
vect = zeros(lines,1);
% use uint8, uint16,
   uint32, uint64, int8,
   int16, int32 or int64 for
   type conversion to
   (unsigned) integer
```

MATLAB is Different - Find the Mistake (8)

```
img = double(imread(name));
figure;
imshow(img);
```

MATLAB is Different - Find the Mistake (8)

```
img=double(imread(name));
figure;
imshow(img);
```

imshow, image and imagesc expect an image with type double to consist of values between 0 and 1.

```
% im2double also rescales
img=im2double(imread(name));
figure;
imshow(img);

% or rescale manually
imshow(img/255);
% or convert back to integer
imshow(uint8(img));
```

MATLAB is Different - Find the Mistake (9)

```
img = imread(name);
% compute center
x = size(img,1)/2;
y = size(img,2)/2;

imshow(img);
hold on;
plot(x,y,'.');
```

MATLAB is Different - Find the Mistake (9)

```
img = imread(name);
% compute center
x = size(img,1)/2;
y = size(img,2)/2;

imshow(img);
hold on;
plot(x,y,'.');
```

```
img = imread(name);
% compute center
x = size(img,2)/2;
y = size(img,1)/2;

imshow(img);
hold on;
plot(x,y,'.');
```

Matrices are accessed with (row,column), i.e. (y,x) !

MATLAB is Different - Find the Mistake (10)

```
img = imread(name);
% compute center
x = size(img,2)/2
y = size(img,1)/2
```

MATLAB is Different - Find the Mistake (10)

```
img = imread(name);
% compute center
x = size(img,2)/2
y = size(img,1)/2
```

```
>> x =
       320
   y =
       240
```

This works!

Note: If there is no semicolon at the end of a command Matlab prints the result/value of the variable.

MATLAB is Different - Find the Mistake (11)

```
for i = 1:n
   for j = 1:m
   A(i,j) = i*j;
   end
end
```

```
for ind1 = 1:n
   A(ind1,ind2) = ...
        ind1*ind2;
   end
end
```

MATLAB is Different - Find the Mistake (11)

```
for i = 1:n
   for j = 1:m
   A(i,j) = i*j;
   end
end
```

```
for ind1 = 1:n
   for ind2 = 1:m
   A(ind1,ind2) = ...
        ind1*ind2;
   end
end
```

`This works!`

`But:`

`i` and `j` are constants for the imaginary unit i.

If you overwrite them you cannot use complex numbers in your program.

How to Make MATLAB Code Faster: Vectorization

```
for ind1 = 1:n
   for ind2 = 1:m
      A(ind1,ind2) = ...
      2*B(ind1,ind2);
   end
end
```

for more tips see:
http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html

How to Make MATLAB Code Faster: Vectorization

Slow Matlab code

```
for ind1 = 1:n
  for ind2 = 1:m
    A(ind1,ind2) = ...
    2*B(ind1,ind2);
  end
end
```

Fast Matlab code

```
A = 2*B;
```

for more tips see:
http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html

How to Make MATLAB Code Faster: Vectorization

```
i = 0;
for t = 0:.01:10
    i = i + 1;
    y(i) = sin(t);
end
```

for more tips see:
http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html

How to Make MATLAB Code Faster: Vectorization

Slow Matlab code

```
i = 0;
for t = 0:.01:10
    i = i + 1;
    y(i) = sin(t);
end
```

Fast Matlab code

```
t = 0:.01:10;
y = sin(t);
```

for more tips see:
http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html

How to Make MATLAB Code Faster: Vectorization

```
for n = 1:1000
   V(n) = pi*(D(n)^2)*H(n);
end
```

for more tips see:
http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html

How to Make MATLAB Code Faster: Vectorization

Slow Matlab code

```
for n = 1:1000
   V(n) = pi*(D(n)^2)*H(n);
end
```

Fast Matlab code

```
V = pi*(D.^2).*H;
```

for more tips see:
http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html

How to Make MATLAB Code Faster: Vectorization

```
for i = 1:n
   for j = 1:m
   if A(i,j) > 255
       A(i,j) = 255;
   end
   end
end
```

for more tips see:
http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html

Some More Useful Stuff

- `toc` prints the time that elapsed since the last `tic`
- `dir` lists all files in a directory
- Cell arrays

  `c = cell(n)` creates an n-by-n cell array of empty matrices

  `c = cell(m,n)` creates an m-by-n cell array of empty matrices

  ➢ the contents of a cell array can be of different size and type
- Structs

  `s = struct('field1', values1, 'field2', values2, …)`

  creates a structure array with the specified fields and values.

  ➢ Or simply write:

  `s.field1 = values1; s.field2 = values2; …`
- Cell arrays and structs can be combined

```
c{1}.field = values1;
c{2}.field = values2;
```

Questions?

Other useful MATLAB resources:

MATLAB documentation
 http://www.mathworks.com/help/index.html

File Exchange
 http://www.mathworks.com/matlabcentral/fileexchange/

Code Vectorization Guide
 http://www.mathworks.com/support/tech-notes/1100/1109.html

Writing Fast MATLAB code
 http://www.mathworks.com/matlabcentral/fileexchange/5685

MATLAB array manipulation tips and tricks
 http://home.online.no/~pjacklam/matlab/doc/mtt/index.html