RWTH AACHEN UNIVERSITY

Computer Vision
WS 2016/2017

Prof. Bastian Leibe<leibe@vision.rwth-aachen.de>
Stefan Breuers<breuers@vision.rwth-aachen.de>

Exercise 1: Introduction to Matlab

due before —

Important information regarding the exercises:

- In the archive for this exercise you will find the functions `apply.m` that should be used for displaying your results. You should also use it to test your implementation and see if the results make sense. Answers are to be submitted within `answers.m` Do **not** modify the apply files in any way.
- Please do **not** include the data files in your submission!
- Please submit your code solution as a zip/tar.gz file named `mn1_mn2_mn3.{zip/tar.gz}` with your **matriculation numbers** (`mn`).
- Please submit your solutions via the L^{2}P system.

Please note:

- The exercise is not mandatory.
- There will be no corrections. If you want to verify your solutions, use the provided `apply` functions.
- Nevertheless, we encourage you to work on the exercises and present your solutions in the exercise class. For this regard the above submission rules.

Question 1: Matlab Tutorial

a) Create your working directory. Run matlab by clicking on the matlab icon or by typing "matlab" in the command shell. Change the directory to your working directory.

b) Read the Matlab help chapter "Getting Started with Matlab" (linked from the class webpage).

```
Matrices and Arrays
Expressions
Working with Matrices
More About Matrices and Arrays
Linear Algebra
Arrays
Multivariate Data
Graphics
Editing Plots
Mesh and Surface Plots
Images
Programming
Flow Control
if, else, and elseif
for
while
return
Other Data Structures
Characters and Text
Scripts and Functions
Scripts
Functions
```

c) Walk through the following Matlab demos (`http://www.mathworks.com/products/matlab/demos.html`)

```
Mathematics
Basic Matrix Operations
Matrix Manipulation
Graphics
2-D Plots
3-D Plots
Images and Matrices
Programming
Manipulating Multidimensional Arrays
Function Functions
Reading Arbitrary Text Files with TEXTSCAN
```

Question 2: Basic Image Processing

Download the file `exercise1.zip` from the class web page and uncompress it in your working directory. This file contains three example images, which you will manipulate in the following. For the first exercises, we provide example code. Try it out and shortly explain what happens. For the later exercises, we only provide a code framework which you should complete by yourselves. Useful commands for this are:

```
imread, image, imshow, imagesc, colormap, imrotate.
```

a) Look up the online help to find out what the above commands do and how they are called.

```
help imread % or click the question mark button
```

b) Read the image "graf.png" and display it in a window with suggested commands.

```
graf = imread('graf.png');
graf = rgb2gray(graf);
figure;
image(graf);
colorbar;

figure;
image(graf);
colormap gray;
colorbar;

figure;
imagesc(graf);
colormap gray;
colorbar;

figure;
imshow(graf);
colorbar;
```

c) Look at the colormap. Create and set a new colormap which varies from 1 to 0, 0 to 1, and several random maps. Display the image. Explain what happens.

```
map = colormap;

figure;
```

```
reverse_map = map(end:-1:1, :);
image(graf);
colormap(reverse_map);

figure;
rand_map = rand(64, 3);
image(graf);
colormap(rand_map);

figure;
image(graf);
colormap('default');
```

d) Convert the image to double, then brighten and darken the image.

```
figure;
dgraf = im2double(graf);
imshow(dgraf);

figure;
bright_graf = dgraf*2;
imshow(bright_graf);

figure;
dark_graf = dgraf/2;
imshow(dark_graf);
```

e) Assign row 200 in the image to a variable. Display it with a plot. Repeat that for column 300.

```
figure;
row200 = dgraf(200, :);
plot(row200);

figure;
column300 = dgraf(:, 300);
plot(column300);
```

f) Remove a rectangular region from the image and display the region. Crop a rectangular region from the image containing only the head of the karate man and display it as a new image.

```
figure;
graf_cut = dgraf;
graf_cut(130:260, 240:450) = 0;
imshow(graf_cut);

figure;
graf_region = dgraf(130:260, 240:450);
imshow(graf_region);
```

g) Crop the head of the chicken and rotate it by 45 degrees. Try different methods (`nearest`, `bilinear`, `bicubic`) and bounding boxes (`loose`, `crop`). Explain the observations.

```
figure;
graf_chicken = dgraf(130:260, 240:450);
graf_chicken_rotated = imrotate(graf_chicken, 45, 'bilinear');
```

```
imshow(graf_chicken_rotated);
```

h) Sample the image (reduce the size). Use different sampling step sizes (2,4,10). Explain what happens.

```
figure;
graf_rescaled = dgraf(1:2:end, 1:2:end);
imshow(graf_rescaled);
```

i) Create a mirror image. Concatenate 2 or 4 images.

```
figure;
graf_mirrored = dgraf(:,end:-1:1);
imshow(graf_mirrored);

figure;
graf_concat = [dgraf, dgraf; dgraf, dgraf];
imshow(graf_concat);
```

j) Convert the image into a vector and build up a histogram of the pixel values in the image. Display the histogram.

```
figure;
graf_vector = reshape(dgraf, 1, size(dgraf, 1) * size(dgraf, 2));
graf_hist = hist(graf_vector, 256);
bar(graf_hist);
```

k) The following commands might also be useful

```
close all;  % close all figures
clc;        % clear the command window
axis equal; % same aspect ratio in x and y
axis off;   % do not display axes
```

Question 3: Webcam

If you have a webcam, you can try out the different techniques from this exercise using your own captured images. Use this code to acquire a pointer to your camera:

```
function vidObj = webcam()
% see "Image Acquisition Toolbox"
% useful command:
% >> imaqtool
% look up your cam settings in a gui
% (can generate a m-file which sets up the cam)

% Device Properties (use imaqtool to find the right vidFormat for your
    camera)

% Windows
% adaptorName = 'winvideo';
% vidFormat = 'I420\_320x240';
```

```
% Linux
adaptorName = 'linuxvideo';
vidFormat = 'YUYV\_320x240';

vidObj = videoinput(adaptorName, 1, vidFormat);
set(vidObj, 'ReturnedColorSpace', 'grayscale');
set(vidObj, 'FramesPerTrigger', 1);
```

You can then take snapshots using the matlab function `getsnapshot`:

```
function snapshot = getsnapshot(obj)
% GETSNAPSHOT Immediately return a single image snapshot.
```

Please turn in your solutions including all relevant files before —!