Bauhaus-Universität Weimar

Image Analysis and Object Recognition

Assignment 1

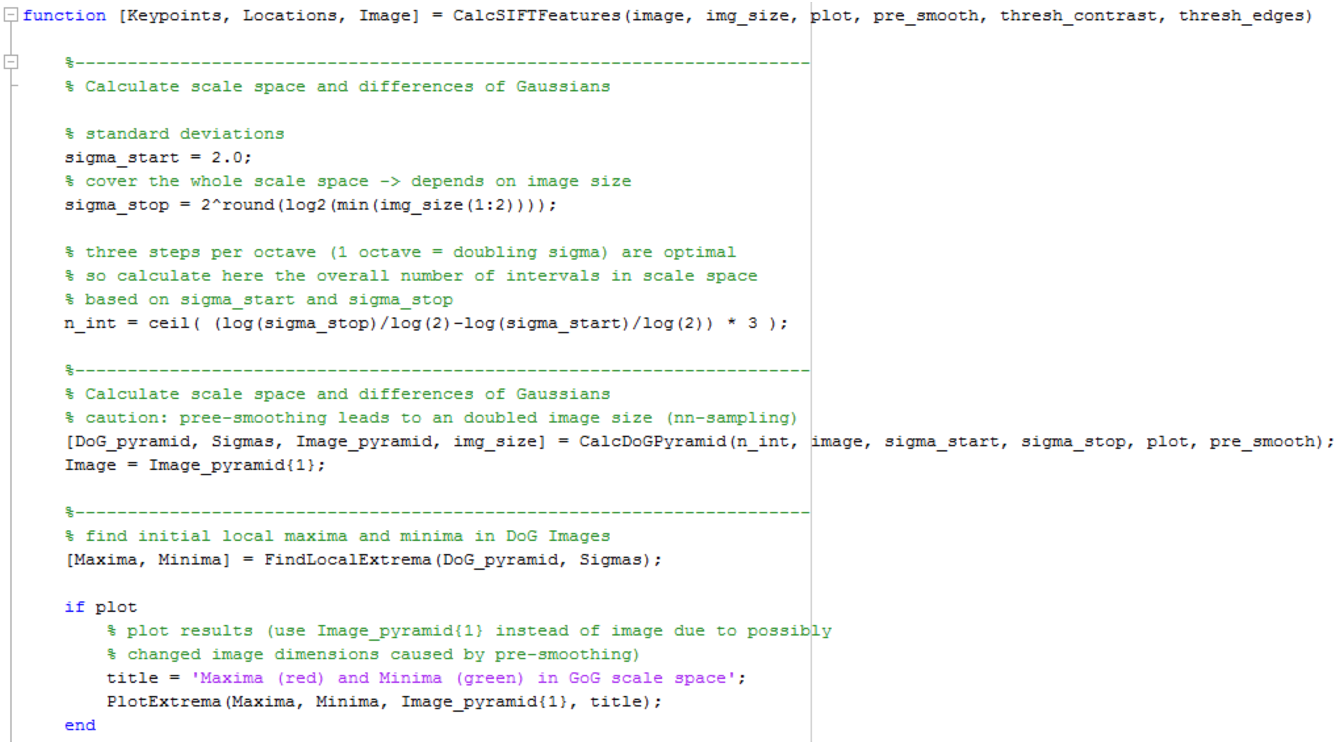
Information Extraction:
Image enhancement, thresholding and morphological operators

SS 2017

(Course notes for internal use only!)

In the assignments, we will …

- … **apply** selected methods of image analysis
- … **implement** these methods



```
function [Keypoints, Locations, Image] = CalcSIFTFeatures(image, img_size, plot, pre_smooth, thresh_contrast, thresh_edges)

    %-----------------------------------------------------------------------
    % Calculate scale space and differences of Gaussians

    % standard deviations
    sigma_start = 2.0;
    % cover the whole scale space -> depends on image size
    sigma_stop = 2^round(log2(min(img_size(1:2))));

    % three steps per octave (1 octave = doubling sigma) are optimal
    % so calculate here the overall number of intervals in scale space
    % based on sigma_start and sigma_stop
    n_int = ceil( (log(sigma_stop)/log(2)-log(sigma_start)/log(2)) * 3 );

    %-----------------------------------------------------------------------
    % Calculate scale space and differences of Gaussians
    % caution: pre-smoothing leads to an doubled image size (nn-sampling)
    [DoG_pyramid, Sigmas, Image_pyramid, img_size] = CalcDoGPyramid(n_int, image, sigma_start, sigma_stop, plot, pre_smooth);
    Image = Image_pyramid{1};

    %-----------------------------------------------------------------------
    % find initial local maxima and minima in DoG Images
    [Maxima, Minima] = FindLocalExtrema(DoG_pyramid, Sigmas);

    if plot
        % plot results (use Image_pyramid{1} instead of image due to possibly
        % changed image dimensions caused by pre-smoothing)
        title = 'Maxima (red) and Minima (green) in GoG scale space';
        PlotExtrema(Maxima, Minima, Image_pyramid{1}, title);
    end
```

Overview

- Organization
- Software
- Assignment
  - Getting familiar with MATLAB
  - Image Enhancement
  - Thresholding
  - Morphological Operators
- Some practical hints

Organization 1/4

- Contact

Name:	Jens Kersten
Phone:	03643 / 58-3730
Email:	jens.kersten@uni-weimar.de
Office:	Bauhausstr. 11, Room 006
Consultation:	Thursdays 1-4 pm

Organization 2/4

- Exercise dates: every 2nd Week
  - **13.04.2017** → Introduction + Assignment 1
  - 04.05.2017 → Assignment 1+2
  - 18.05.2017 → Assignment 2+3
  - 01.06.2017 → Assignment 3+4
  - 15.06.2017 → Assignment 4+5
  - **22.06.2017** → Assignment 5+6
  - 13.07.2017 → Final Meeting / Summary / Discussion

Organization 3/4

- Work in groups of 3 students
- Registration via e-mail:
  - → jens.kersten@uni-weimar.de
  - → Provide **names, matriculation numbers** and **study course**
  - → Maybe choose a name for your group

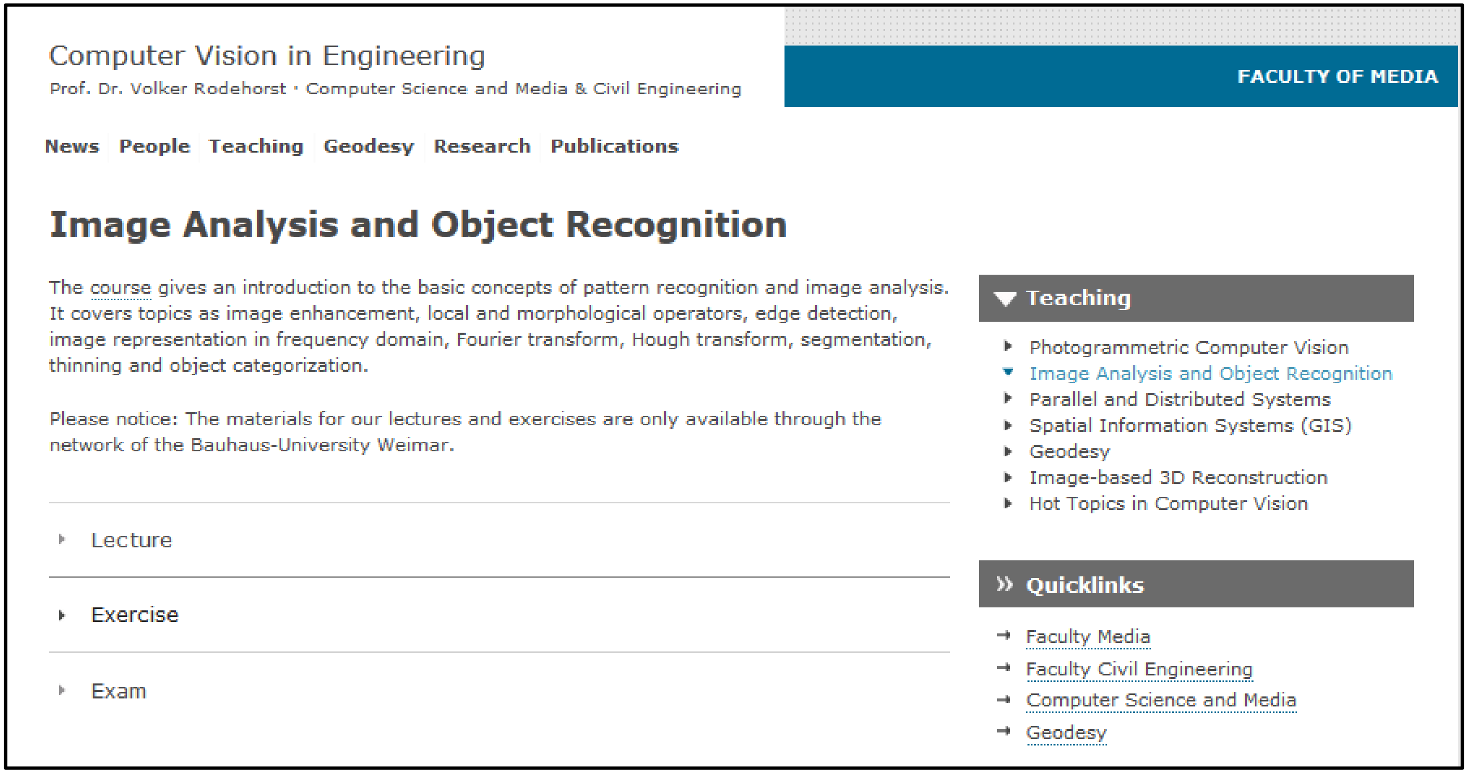
- Attendance not mandatory but recommended for at least one member of each team

- Use LiNT-Pool, Bhs. 11, 1st floor

Organization 4/4

- All material (slides, exercises, sample solutions, images, additional files, …) are available on CV-homepage

http://www.uni-weimar.de/media/cv



Computer Vision in Engineering
Prof. Dr. Volker Rodehorst · Computer Science and Media & Civil Engineering
FACULTY OF MEDIA
News
People
Teaching
Geodesy
Research
Publications

Image Analysis and Object Recognition

The course gives an introduction to the basic concepts of pattern recognition and image analysis. It covers topics as image enhancement, local and morphological operators, edge detection, image representation in frequency domain, Fourier transform, Hough transform, segmentation, thinning and object categorization.

Please notice: The materials for our lectures and exercises are only available through the network of the Bauhaus-University Weimar.

Lecture
Exercise
Exam
Teaching
Photogrammetric Computer Vision
Image Analysis and Object Recognition
Parallel and Distributed Systems
Spatial Information Systems (GIS)
Geodesy
Image-based 3D Reconstruction
Hot Topics in Computer Vision
Quicklinks
Faculty Media
Faculty Civil Engineering
Computer Science and Media
Geodesy

Organization 4/4

- All material (slides, exercises, sample solutions, images, additional files, …) are available on CV-homepage

http://www.uni-weimar.de/media/cv

- Downloads only possible from university-network or using vpn-client

- Results of exercises
  - Code-files (*.m), images, text documents,…
  - via e-mail in a zipped file
- Deadline: day before we meet again (Wednesday) 12 noon

Software 1/2

- MATLAB (preferred)
  - Version R2016a
  - Installed @ LiNT-Pool
  - Licensed product
  - GUI available
  - Documentation, help and tutorials

- OCTAVE
  - Version 4.2.1 (Win:https://ftp.gnu.org/gnu/octave/windows/)
  - Free software, some functions are very slow (Win)
  - GUI (Should work for Windows 10)
  - Documentation, help and tutorials online

Software 2/2

- IMAGE PROCESSING TOOLBOX (MATLAB) or IMAGE PACKAGE (Octave)
  - Functions for analyzing, displaying, working on images
  - **MATLAB**:

    http://www.mathworks.de/products/image/
  - **OCTAVE**:

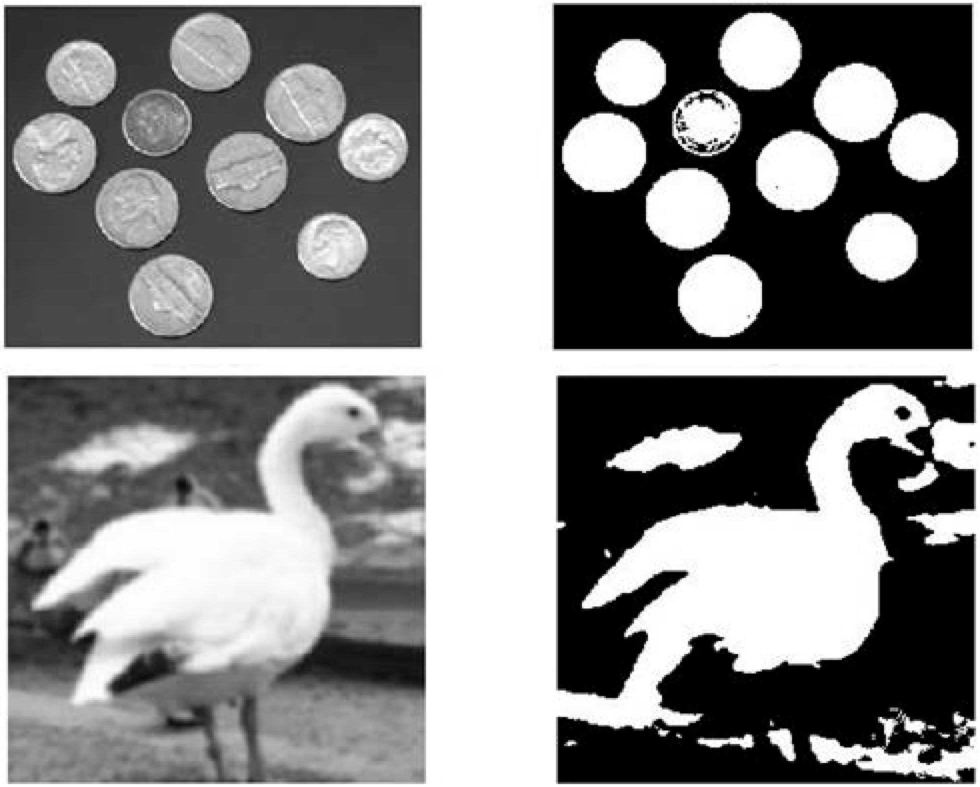
    http://octave.sourceforge.net/image/index.html

    Required packages: **general, control, signal, image**
    - Installation:

```
>> pkg install image-2.2.1.tar.gz
>> pkg load image
```

Assignment 1

- **Final Goals**
  - Getting familiar with MATLAB
  - Extract image pixels which represent objects (“foreground”)



- Simple method for distinguishing between background and foreground
- Don’t expect amazing results
- Choose images with “simple” contents

Assignment 1

- **Topics**
  - Getting familiar with MATLAB
    - http://www.mathworks.com/products/matlab/
    - http://www.mathworks.com/products/image/
    - http://en.wikibooks.org/wiki/Octave_Programming_Tutorial
    - http://octave.sourceforge.net/image/overview.html

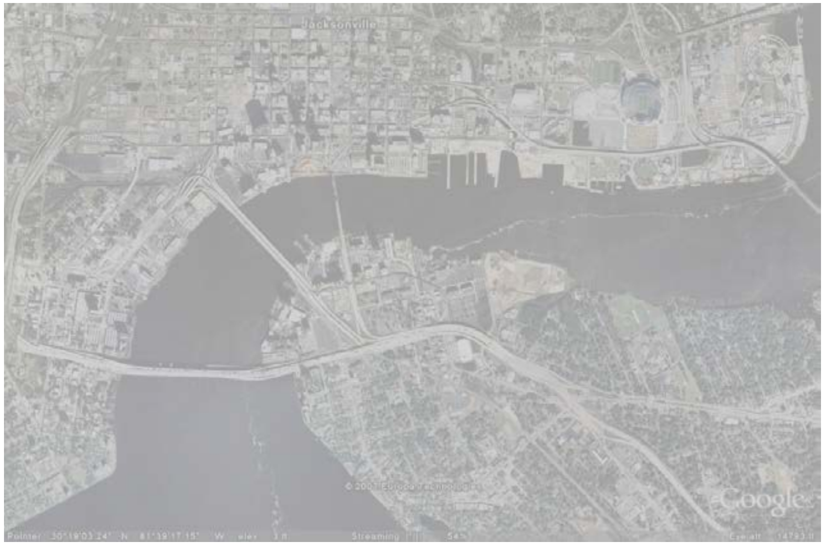
  → Learning by doing!

  - Extract image pixels which represent foreground (objects)
    - Image enhancement
    - Thresholding (binarization)
    - Morphological operators

Assignment 1: Overview

- **Extract image pixels which represent objects**
- **Input:** Low-contrast photograph
  - Not too complex image content!
  - Maybe use provided image

  → **Task here:** Extract water regions



  - **Task A**: Enhance image with bad contrast
  - **Task B**: Find and apply a threshold on image values → separate background=0 and foreground=1
  - **Task C**: Refine the binary mask using morphological operators
  - **Task D**: Write a main function, which conducts A, B and C

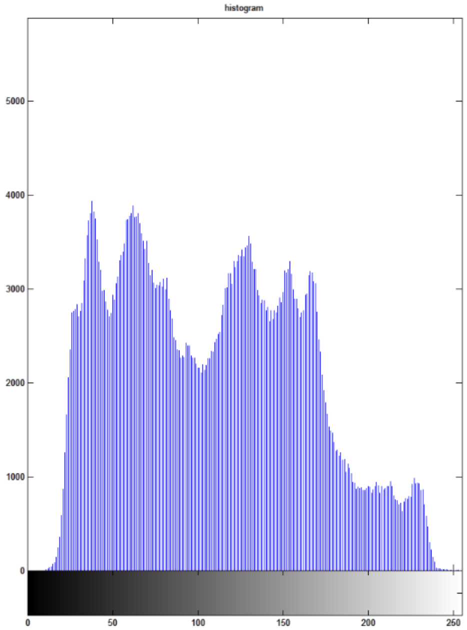
Assignment 1 (A)

(A) Image enhancement: Write a function to enhance the image contrast

Your image may consist of more than one channel (r,g,b) → compute **mean** value for each pixel (`uint8(mean(image, 3))`) and use the resulting grayscale image

a) Visualize the initial image and the corresponding histogram (`figure, subplot, imshow, imhist`)

8 bit image



2^8 = 256 grayscale values

count occurrences of grayscale values

Assignment 1 (A)

(A) Image enhancement: Write a function to enhance the image contrast

Your image may consist of more than one channel (r,g,b) → compute **mean** value for each pixel (`uint8(mean(image, 3))`) and use the resulting grayscale image

a) Visualize the initial image and the corresponding histogram (`figure, subplot, imshow, imhist`)

b) What are the characteristics of the histogram? (source-code commands)

c) Do the enhancement (contrast stretching, lecture 1)

d) Describe the changes in short? (source-code commands)

e) Visualize the resulting image

Assignment 1 (B)

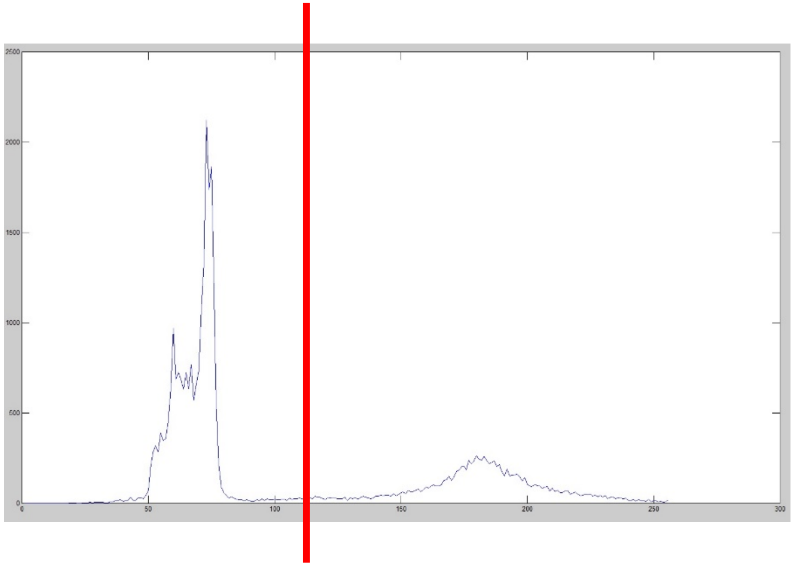
(B) Thresholding: Write a function for global thresholding

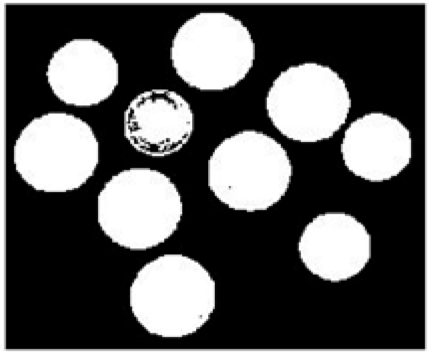
Input: Result image from Task A

a) Convert image to binary mask , where 0 = background (land) and 1 = searched regions (water), (`graythresh, im2bw, <, >`)

b) Visualize the resulting mask

c) Make some tests with different thresholds and describe the problems / difficulties you had in finding a threshold





Assignment 1 (C)

(C) Morphological filtering: Write a function for the morphological filtering of the binary image

Input: Result image from Task B

a) **Successively** apply **opening** and **closing** on the input image, (`imopen, imclose`)

b) Visualize the result

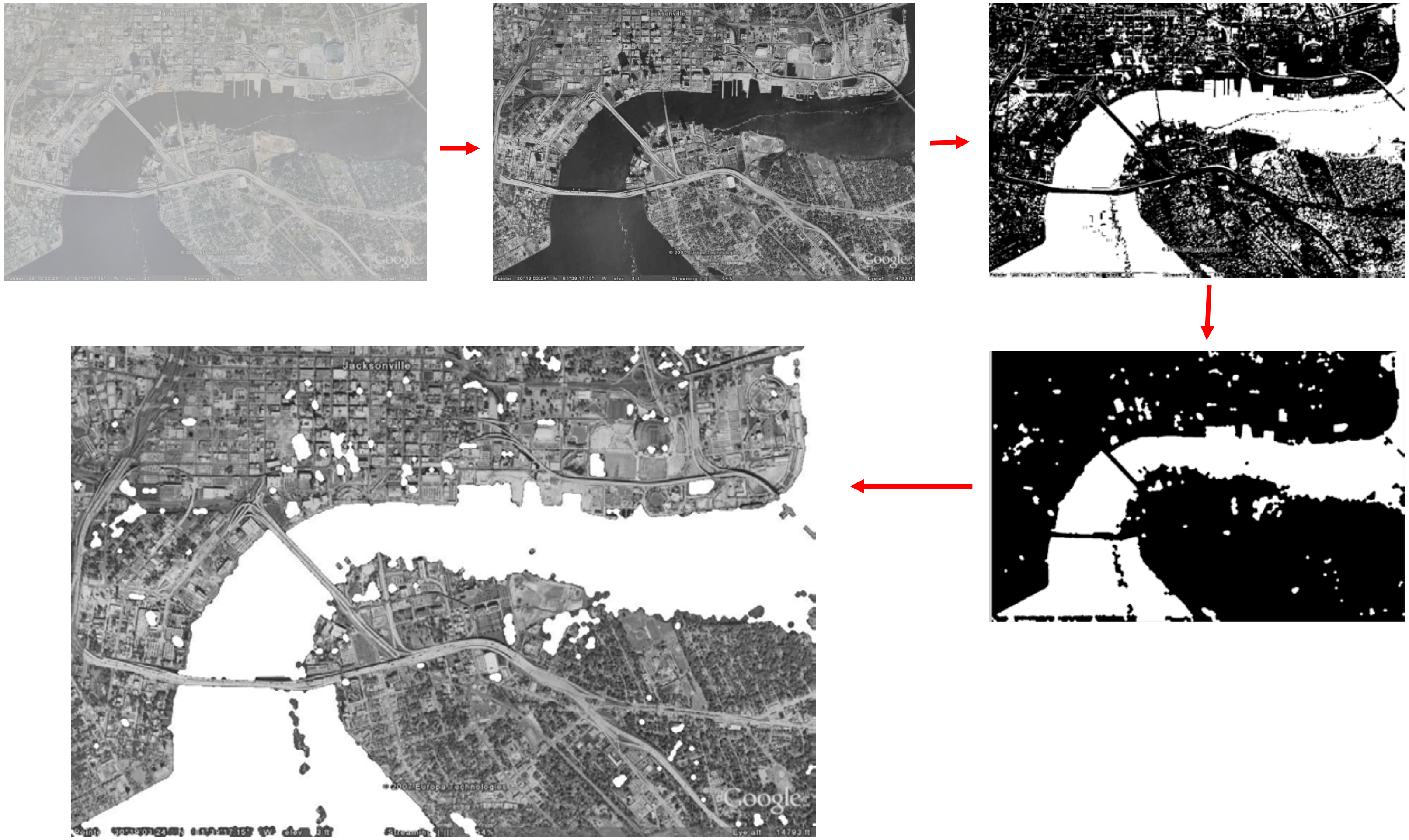
c) **Implement** a function for erosion **or** dilation

d) compare the results to the output of the corresponding MATLAB functions (`imerode, imdilate`, `for, .*, sum`). Are there any differences in the results and why?

e) Visualize an **overlay** of the enhanced image and the derived mask

Assignment 1 (A-C)

Workflow:



Assignment 1 (D + E)

D. Write a main function which sequentially conducts steps A-C
   - Idea: Use the code for other images

E. Discussion: Are the results satisfactory? What are the limitations of this approach for separating background and foreground (code comments)?

Some practical hints

- Implementing and using functions
- Variables and data types
- Debugging
- Available MATLAB functions

Implementing and using functions

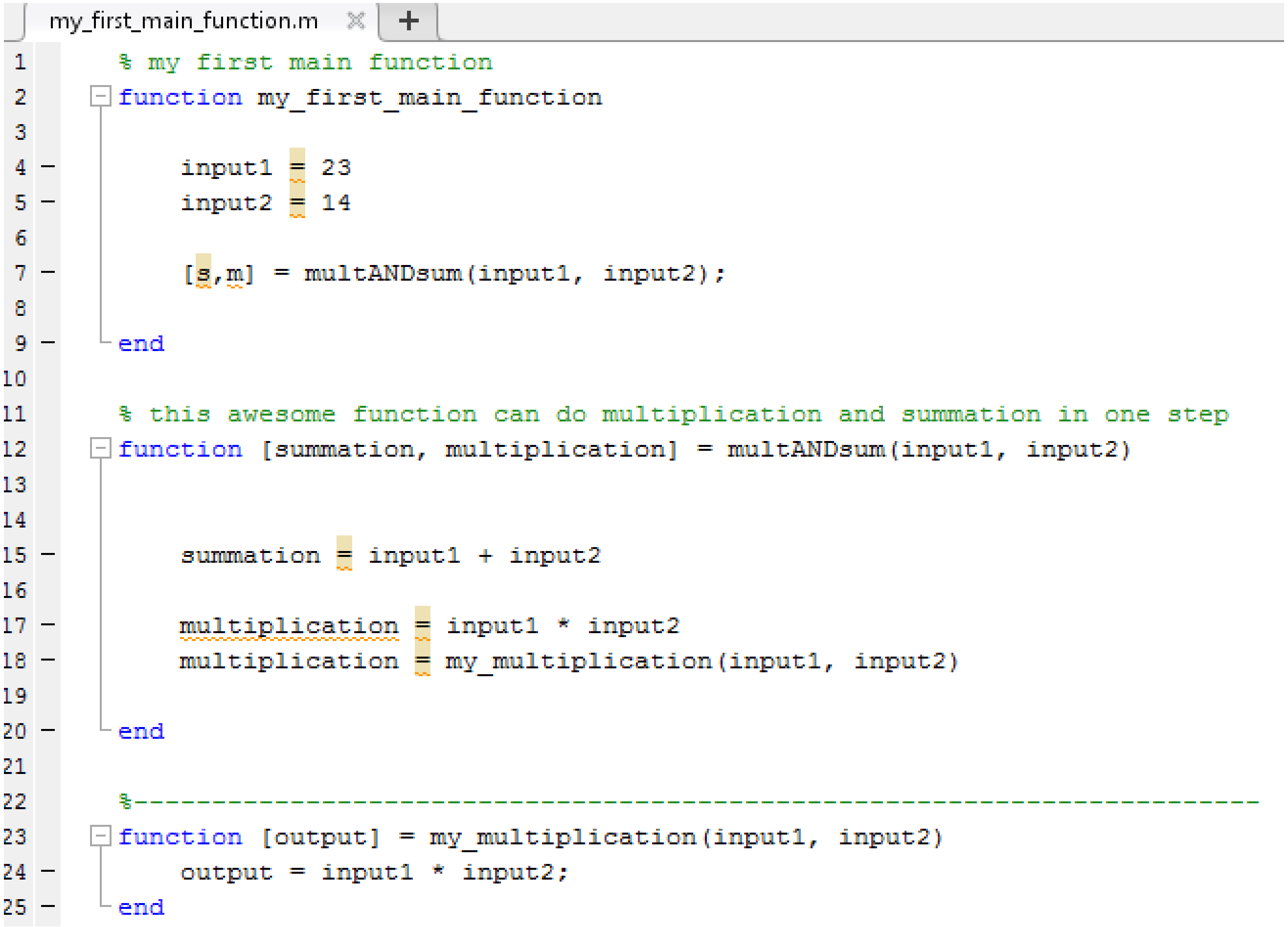
- Function(s) are stored in *.m files
- Function name = filename
- Definition in file **my_first_main_function.m**:

```
function my_first_main_function
            …
end
```

- You may define more than one function in one file

- Example can be found on homepage

Implementing and using functions



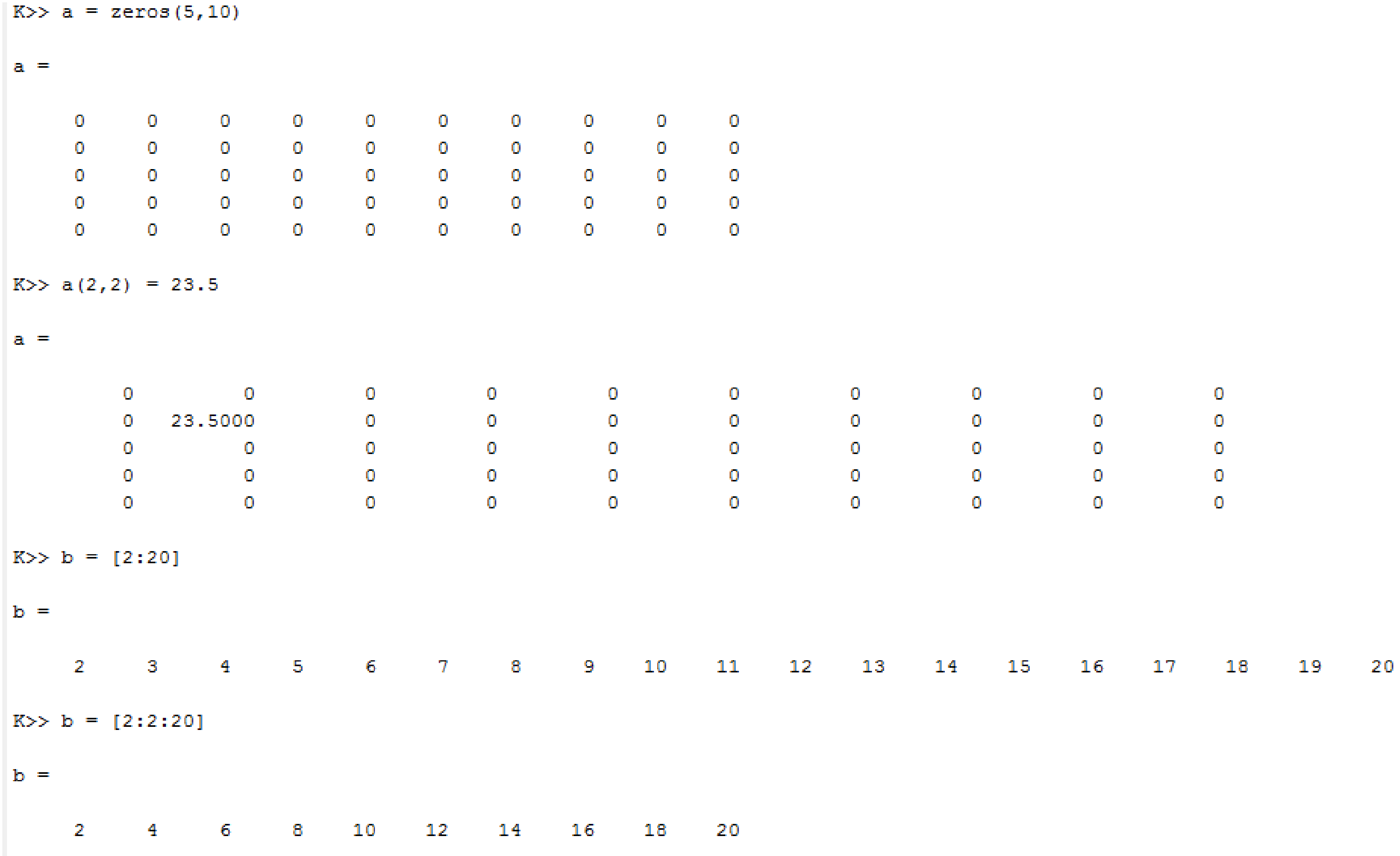
```
% my first main function
function my_first_main_function

    input1 = 23
    input2 = 14

    [s,m] = multANDsum(input1, input2);

end

% this awesome function can do multiplication and summation in one step
function [summation, multiplication] = multANDsum(input1, input2)


    summation = input1 + input2

    multiplication = input1 * input2
    multiplication = my_multiplication(input1, input2)

end

%------------------------------------------------------------------------
function [output] = my_multiplication(input1, input2)
    output = input1 * input2;
end
```

Variables

- Scalars and arrays are interpreted as matrices
- Scalars: *>> s = 5;*
- Vectors: *>> v = [1,2,3,4];*
- Matrices: *>> m = [1,2,3,4;5,6,7,8];*

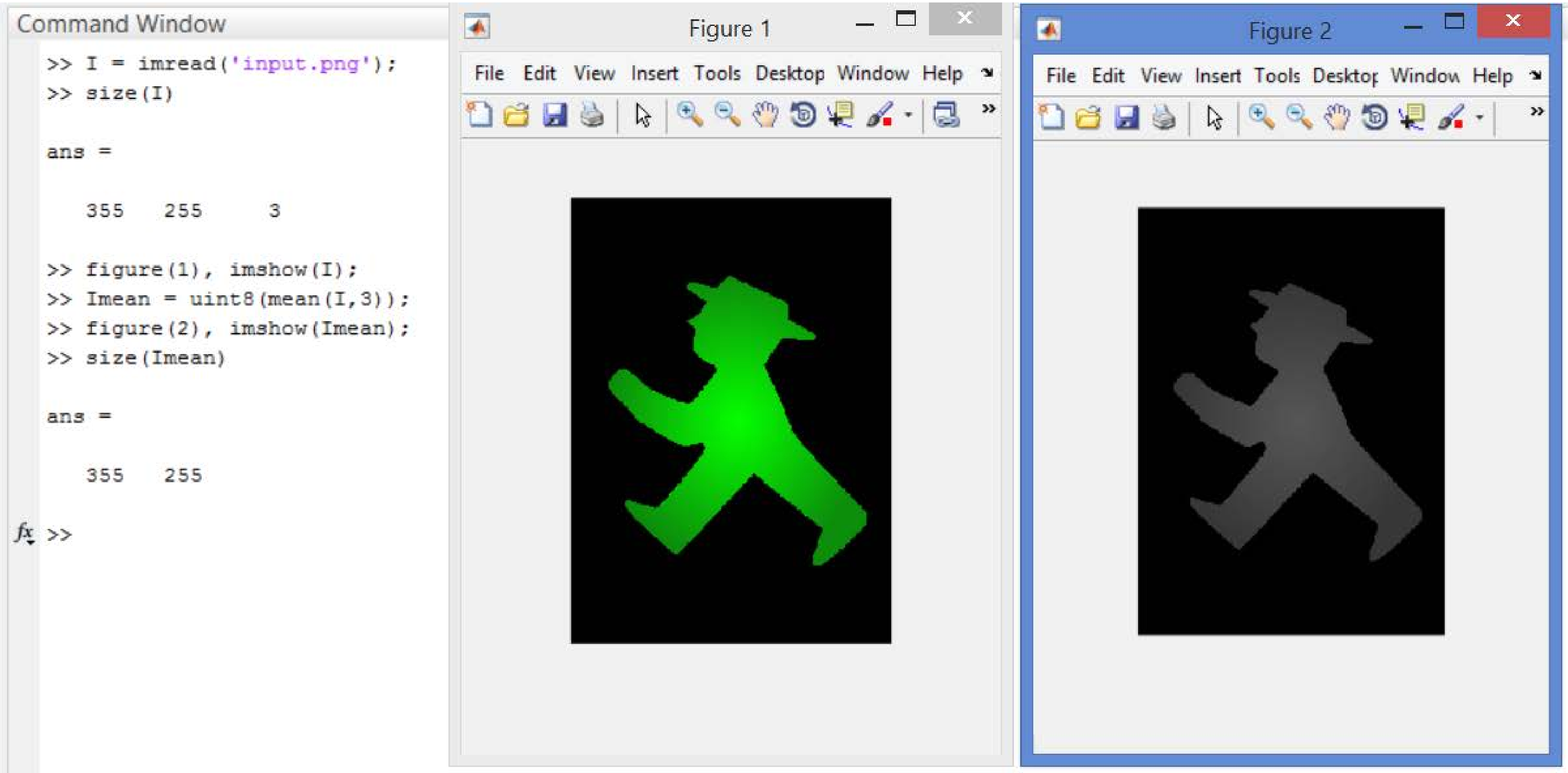
- *Access to single element of matrix: >>m(1,2) = 2*
  - Indices always start from 1 (not in C/C++)
  - First index → row, second index → column
- Data types:
  - **Double precision**: real valued floating point, e.g. 176.893648352134
  - **Integer: e.g. 8, 16, 32 or 64 bit**; 8 bit: value range 0,…,255
  - **Logical**: 0 (false) or 1 (true)

Variables



```
K>> a = zeros(5,10)

a =

     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0

K>> a(2,2) = 23.5

a =

         0         0         0         0         0         0         0         0         0         0
         0   23.5000         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0         0

K>> b = [2:20]

b =

     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20

K>> b = [2:2:20]

b =

     2     4     6     8    10    12    14    16    18    20
```

Variables: Image Matrices


Command Window
Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 2
File Edit View Insert Tools Desktop Window Help

```
>> I = imread('input.png');
>> size(I)

ans =

   355   255     3

>> figure(1), imshow(I);
>> Imean = uint8(mean(I,3));
>> figure(2), imshow(Imean);
>> size(Imean)

ans =

   355   255

>>
```

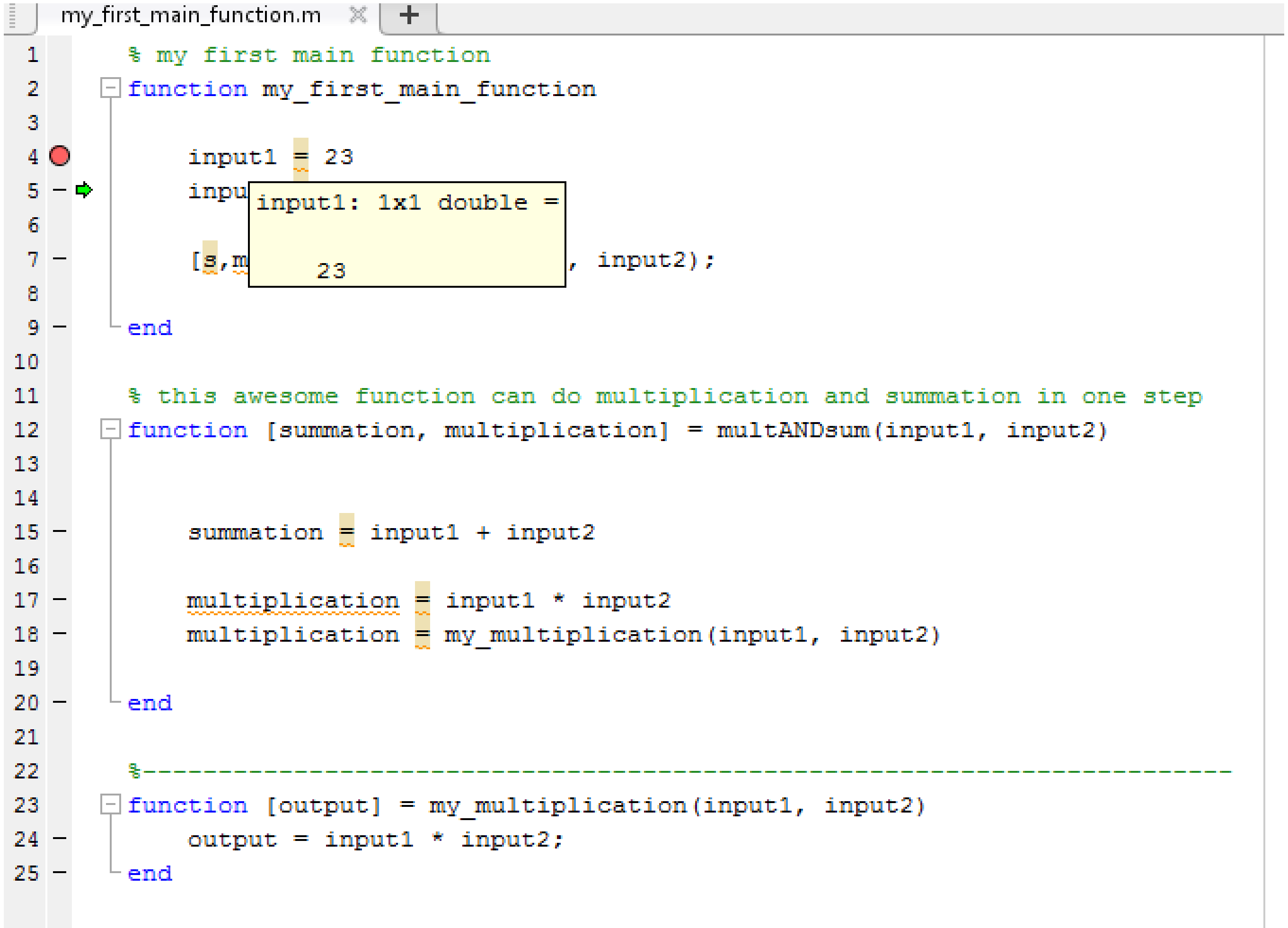
Available MATLAB functions

- GUI for imagefile picking: *uigetfile*
- Read image into matrix: *imread*
- Visualize image: *figure, imshow*
- Get array dimensions: *size*
- Type-conversions: *double, uint8, im2uint8, im2bw,…*
- For-loop:

```
for i=start:stop
      …
end
```

- Morphological operations: *imdilate, imerode*
- Element-wise multiplication of matrices with same size: *prod = A .* B;*
- Use “;” after matrix operations!

Debugging



```
% my first main function
function my_first_main_function

    input1 = 23
    inpu

    [s,m                                    , input2);

end

% this awesome function can do multiplication and summation in one step
function [summation, multiplication] = multANDsum(input1, input2)


    summation = input1 + input2

    multiplication = input1 * input2
    multiplication = my_multiplication(input1, input2)

end

%----------------------------------------------------------------------
function [output] = my_multiplication(input1, input2)
    output = input1 * input2;
end
```

```
input1: 1x1 double =

    23
```

- If you (successfully!) complete assignment 1, you…
  - …know how to write and use functions
  - …know how to read, write and visualize images
  - …know how to do operations with pixels and filters

…are familiar with MATLAB ☺!!

Thank you very much!

Questions?