Analysis of Three-Dimensional Shapes
F. R. Schmidt, M. Vestner, Z. Lähner
Summer Semester 2016

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 0

These sheet is only for self-studying. If you have questions or problems, please write to laehner@in.tum.de.

Matlab Tutorial

MATLAB is a commercial programming language that is optimized for scientific working. It is not particulary fast but most mathematical expressions are easier to write down. As a student of TUM you can get a free student license at matlab.rbg.tum.de or use the computer labs. All programming exercises will be in Matlab and this exercise sheet is meant for those who have never or just very briefly used Matlab before.

Exercise 1 (0 points)**.** Go to https://matlabacademy.mathworks.com/ and login with your Mathworks account (if you do not have one yet, go to matlab.rbg.tum.de and follow the instructions in order to sign up with the right email address). Do the MATLAB Onramp course, chapters 1-9 (skip the videos if you want).

If you have problems remembering if rows or columns come first in the indexing of matrices think of cheetah jumping from a tree first and then running on the ground.

Exercise 2 (0 points)**.**

1. Matlab has an extensive (but not easy to navigate) documentation. Google how to compute the eigendecomposition of a matrix and calculate the eigenvectors and eigenvalues of a `A = gallery('lehmer', 4)` (creating a symmetric positive definite matrix).

2. Now only calculate the first two eigenvectors of `B = gallery('lehmer',10)` (without calculating all 10).

You can also get in-program information about functions by using `help [your command]` in the Matlab console but you have to know the name of the command.

Exercise 3 (0 points)**.**

1. Make a new file `blackbox.m` that includes a new functios which returns the minimum of three inputs. You can now call `blackbox(1,2,3)` from the Matlab console (or any other file) when you are in the same folder as your file.

2. Now instead create a function handle `advancedbb` with the same functionality. Function handles are stored in variables instead of files and useful if your function can be written down in one line. Create them by using `fh = @(x, ...) ( function depending on x and any other input you defined )`.

3. Create a function handle that calculates the squareroot of a number. Then apply it to every entry of `A = rand(5,1)` by using `arrayfun`.

Exercise 4 (0 points)**.** 1. Get `A = rand(4)`. What happens when you look at `A(:)`?

2. Use `reshape` to make A into a 8×2 matrix.

3. Compare $A(4, 2)$ and $A(12)$. Whats the pattern?

4. Try $A < 0.5$. You can use a logical array with the same size as the original matrix to indicate which entries you want to access. Replace all values greater than 0.5 in A with NaN.

Exercise 5 (0 points)**.** 1. Create a new struct with `M = struct`. You can now assign variables of different types to any `M.name`. A struct basically collects variables that belong together. You can also give a whole struct as an input to a function (you can give anything as an input to a function, even function handles).

2. You can assign the x-,y- and z-coordinates of sphere to M by using `[M.X, M.Y, M.Z] = sphere`. Check out your new M in the interface.

3. Simply typing `figure` will open a new figure and everything that you plot will be plotted there as long as you do not click on a different figure. You can get store a handle to your figure by assigning it to a variable `h = figure` but this is normally not necessary. Plot the sphere in two figures by using `scatter3` (you need to vectorize the coordinates) and `surf`.

4. Plot both in the same figure by using `subplot`.

5. The coloring you see in surf represents the height function, you can plot any function on the surface by assigning a value to each point. Try `surf(x,y,z,f)` with f being a matrix the same size as $M.X$ and the y-coordinates as entries.

If you want to try out more, some keywords to search for might be `cell arrays, repmat, colormap, parfor, fprintf, solve`. It is also possible to create classes in Matlab.

Some general notes:

1. Don't forget that Matlab starts its indexing at 1 instead of 0.

2. Matlab will automatically assign types to your variables (mostly double, logical, complex double). You can try to force the type by using its type name as a function (e.g. `logical(A)`).

3. Matlab has some internal optimization on matrix operations. Do not use for-loops if you can express it with matrices.

4. If you need something specific, someone probably already asked your question on Stackexchange. Google is your friend.