Assignment 1 – CV1 WS16/17 31/10/2016

Prof. Stefan Roth
Tobias Plötz
Junhwa Hur

This assignment is due on November 14th, 2016 at 12:00.

Group work and grading policy

You are required to work on each assignment in groups of ***two*** people. It is up to you to form groups, but please note that the group assignments cannot change after the first homework assignment. If you can not find a group partner on your own, we will assign you to another person.

Programming exercises

For the programming exercises you will be asked to hand in Julia code. Please use version **v0.5.0** of Julia as we will use it to grade your solution. You need to comment your code in sufficient detail so that it will be easily clear to us what each part of your code does.

Your code should display your results so that we can judge if your code works from the results alone. Of course, we will still look at the code. You **must** adhere to the naming scheme for functions and files included with each problem.

Do *not* rename function files and do *not* change the given function signatures. If you feel that there is a mistake in the assignments, ask us on Moodle.

Pen & paper exercises

You might also have some theoretical exercises to do in the assignments. In this case we would greatly appreciate if you could typeset the theoretical part of your solution (ideally with LaTeX) and submit it as a PDF along with the rest of your solution. If you are not sufficiently familiar with a mathematical typesetting system such as LaTeX, you can also hand in a high resolution scan of a handwritten solution. Please write neatly and legibly.

Files you need

All the data you will need for problems will be available on Moodle `https://moodle.tu-darmstadt.de/course/view.php?id=8060`.

Handing in

Please upload your solutions to Moodle. You only need to submit one solution per group. If, for whatever reason, you cannot access Moodle get in touch with us as soon as possible.

Upload all your solution files (the writeup, `.jl` files, and other required files) as a single `.zip` or `.tar.gz` file. **Please note that we cannot accept file formats other than the ones specified!**

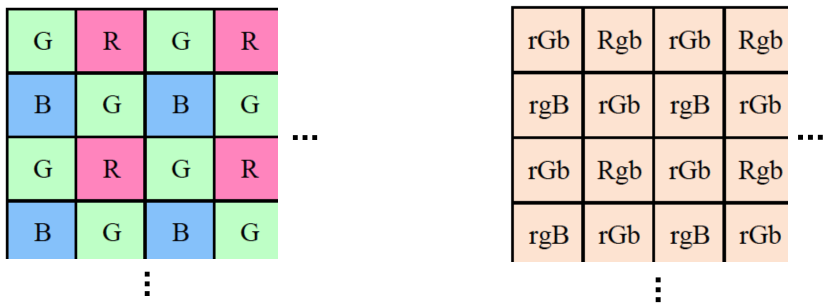
Late handins

We will accept late handins, but we will take *20% of the total reachable points* off for every day that you are late. Note that even 1 minute late will be counted as being one day late! After the exercise has been discussed in class, you can no longer hand in.

Other remarks

Your grade will depend on various factors. Of course it will be determined by the correctness of your answer. But it will also depend on a clear presentation of your results and good writing style. It is your task to find a way to *explain clearly how* you solved the problems. Note that you will get grades for the solution, not for the result. If you get stuck, try to explain why and describe the problems you encountered – you can get partial credit even if you did not complete the task. So please hand in enough information for us to understand what you did, what things you tried, and how it worked! We will provide skeleton code for every assignment. Please use the provided interfaces as it allows us to better understand what you did.

We encourage interaction about class-related topics both within and outside of class. However, you should not share solutions with other groups, and **everything you hand in must be your own work**. You are also not allowed to plainly copy material from the web. You are required to **acknowledge any source of information that you used to solve the homework** (i.e. books other than the course books, papers, etc.). Acknowledgements will *not* affect your grade. Thus, there is no reason why you would not acknowledge sources properly. Not acknowledging a source that you have used, on the other hand, is a clear violation of academic ethics. Note that the university as well as the department is very serious about plagiarism. For more details please see `http://www.informatik.tu-darmstadt.de/index.php?id=202` and `http://plagiarism.org`.



(a) color filter array layout.

(b) interpolated pixel values; with unknown values shown as lower case.

Figure 1: Bayer RGB pattern.

Problem 1 - Bayer Interpolation (10 points)

Although today's digital cameras output RGB color pictures or videos, most of them do not have three separate RGB sensing chips but only one single chip based on a *color filter array* (CFA), where alternating sensor sites are covered by different color filters. The Bayer pattern (see Fig. 1) is the most commonly used pattern. In this problem the task is to construct a RGB color image from the data in the Bayer pattern through interpolating the missing color values.

Your task is to write Julia code that uses *bilinear interpolation* to restore all missing RGB color values (Fig. 1(b)) from the image data in the Bayer pattern (Fig. 1(a)). Load the image data saved in the Bayer pattern from `bayerdata.jld` and show two images: *(1)* the RGB color image directly transformed from the Bayer pattern (missing color values should be filled with 0) and *(2)* the interpolated full RGB color image.

The skeleton is given in `problem1.jl`. Implement the data loading in the function `loadbayer`, separate the color channels filling missing values with zero in the function `separatebayer`, and implement the interpolation in the function `debayer`.

Write your own code to implement bilinear interpolation. For boundary pixels that might not be (bi)linearly interpolated, please find any other ways (*e.g.*, take the values of the nearest neighbours) to make the color of each pixel look consistent with its neighbours. If you are not familiar with (bilinear) interpolation, you can find a tutorial here: `http://www.cambridgeincolour.com/tutorials/image-interpolation.htm`.

Notes: The file `problem1.jl` includes type annotations for the inputs and outputs of each function. Please stick to these types. If you get errors from the `using` statements you will probably have to add the corresponding packages to your Julia installation. The function `imfilter` of the `Images` package might be useful to solve this problem.

Problem 2 - Projective Transformation (10 points)

In this exercise, a camera is simulated to allow a specific visualization of a 3D scene as a 2D image. First, transformations are performed on 3D points based on the following order:

- translation by $[-27.1, -2.9, -3.2]^{\mathrm{T}}$;
- rotation by 135 degrees around the y-axis;
- rotation by -30 degrees around the x-axis;
- rotation by 90 degrees around the z-axis.

Then a central projection onto the xy-plane, with principal point $[8; -10]$, focal length 8 and square pixels, is applied. The 2D points in the image plane are thus obtained.

Tasks:

You will start with the obtained 2D points and some other information to restore the original 3D coordinates of all points.

- Implement the functions `cart2hom` and `hom2cart`, which convert an arbitrary matrix of 2D or 3D points to homogeneous coordinates and back.
- All the operations (transformations plus projection) are equal to one single perspective projection. However, doing the computation in separate steps makes debugging easier. Compute the rotation matrices around the individual axes and the translation. Then, compute the camera intrinsics matrix P in `getprojection` and use the obtained matrix to compute the camera matrix C in `getfull` (all matrices in homogeneous coordinates).
- Load all the 2D points from the file `obj_2d.jld` with the function `loadpoints`. The point coordinates are given as column vectors. Show all the points in a 2D plot.
- In the file `zs.jld` you can find the z-coordinates of all transformed 3D points right before the projection is applied. Load them with `loadz` and compute the coordinates (namely, x and y) of these transformed 3D points in `invertprojection`.
- Perform the inverse transformations to get the original 3D coordinates of all points in `inverttransformation`. Show these points in a 3D plot in `displaypoints3d`. (`scatter3D` of the PyPlot package may be useful, with which you can adjust the viewpoint of the 3D plot using the rotation icon of the figure window.)
- Use the obtained 3D points, recompute the projected 2D points using the camera C in `projectpoints` and show them in a new figure. Do you get the same 2D points as given?
- Briefly explain with a comment in the code why it is necessary to provide the z-coordintes of the individual points.
- Change the order of the transformations (translation and rotation). Check if they are commutative.

Problem 3 - Image Filtering and Edge Detection (15 points)

In this problem you will get to know how to use Julia to compute image derivatives and create a simple edge detector. For testing use image `a1p3.png`.

Tasks:

- Create in `createfilters` a 3×3 x-derivative filter that computes the derivative using central differences (i.e., $\frac{f(x+h;y)-f(x-h;y)}{2h}$ where $h = 1$) and at the same time smooths in y-direction with a 3×1 Gaussian filter of $\sigma = 1.0$. In other terms, you obtain the full 3×3 filter by combining a 1×3 with a 3×1 filter. Also, create a corresponding y-derivative filter. Make sure that the filters are correctly normalized.
- Use these filters to compute the x-derivatives ∂x and y-derivatives ∂y of the input image from above. Use `imfilter` to do the filtering. Your boundary handling should be such that it replicates the pixels on the image edge. Display both results. Note: Make sure that the sign of the derivatives is correct.
- Compute and display the gradient magnitude $\sqrt{\partial_x^2 + \partial_y^2}$. Detect edges by thresholding the gradient magnitude in `detectedges`, *i.e.*, check if the gradient magnitude at each pixel exceeds a certain value.
- Use *non-maximum suppression* to thin the obtained edges. For every edge candidate pixel, consider the grid direction (out of 4 directions) that is “most orthogonal" to the edge. If one of the two neighbours in this direction has a larger gradient magnitude, remove the central pixel from the edge map. Experiment with various threshold values from above step and choose one that shows the “important" image edges. Briefly explain with a comment in the code how you found this threshold and why you chose it.