project 1:

fun with Fourier transforms

solution(s) due:

December 1, 2016 at **12:00** via email to **bauckhag@bit.uni-bonn.de**

general, introductory remarks:

For all practical projects in this course, it is recommended that you implement your solutions in *Python* using appropriate methods available in the *NumPy* and *SciPy* modules. If you do not already have the full *Python* stack installed on your computer, you might want to install the anaconda package which contains everything you'll ever need:

https://www.continuum.io/downloads

If you prefer your solutions to run really fast, you may also implement in *C/C++*. In this case, some of the functions provided in the GNU scientific library (gsl) might come in handy.

Note, however, that **implementations in MATLAB will not be accepted** in this course.

If you opt for implementing your practical solutions in *Python* and need a quick introduction to *NumPy* and *SciPy*, the following slides could help:

C. Bauckhage, *Quick Introduction to (Scientific)Python*, dx.doi.org/10.13140/RG.2.2.18807.21928

If you have never worked with digital images and have no idea as to how to load, save, or process them, the following two notes may be of interest:

C. Bauckhage, *NumPy / SciPy Recipes for Image Processing: "Simple" Intensity Transformations*, dx.doi.org/10.13140/RG.2.1.4125.3606

C. Bauckhage, *NumPy / SciPy Recipes for Image Processing: Creating Fractal Images*, dx.doi.org/10.13140/2.1.2975.5685

problem specification:

task 1.1: warm-up

Write a program that reads an intensity image $g(x, y)$ of width w and height h and computes a new image $\tilde{g}(x, y)$ where

$$\tilde{g}(x,y) = \begin{cases} 0, & \text{if } r_{\text{min}} \leq \|(x,y) - (\frac{w}{2}, \frac{h}{2})\| \leq r_{\text{max}} \\ g(x,y), & \text{otherwise} \end{cases}$$

and r_{min} and r_{max} are user defined constants. Here is an illustration:

$g(x, y)$ $\tilde{g}(x, y)$

Run you program on the image `clock.jpg` in the Images folder of the Google site for this course to demonstrate that it is working. Experiment with different choices for r_{min} and r_{max}.

task 1.2: getting used to the Fourier transform

note: *This task is mandatory for everybody in the course. In other words, every member of each team must do it.*
Explore the behavior of the Fourier transform. To this end, when working with *Python*, import the following

```
import numpy as np
import numpy.fft as fft
import matplotlib.pyplot as plt
```

Next, create a uni-variate function $f(x)$ to transform, say

```
n = 512
x = np.linspace(0, 2*np.pi, n)
f = np.sin(x)
```

This roduces an array `f` of type `float`. To plot this (discrete) function (of finite support), you could use

```
plt.plot(x, f, 'k-')
plt.show()
```

Next, compute the (discrete) Fourier transform $F(\omega)$ of $f(x)$ by means of

```
F = fft.fft(f)
```

This will produce an array `F` of type `complex`. Hence, we you try the following plotting commands

```
w = fft.fftfreq(n)
plt.plot(w,  F, 'k-')
plt.show()
```

you should get a warning message. Therefore, see what happens, if you execute

```
plt.plot(w, np.abs(F), 'k-')
plt.show()
```

instead. Also, have a look at the outcome of

```
plt.plot(w, np.log(np.abs(F)), 'k-')
plt.show()
```

Now, generalize the function $f(x)$ as follows: $f(x) = o + \alpha \cdot \sin(\nu x + \phi)$. For example

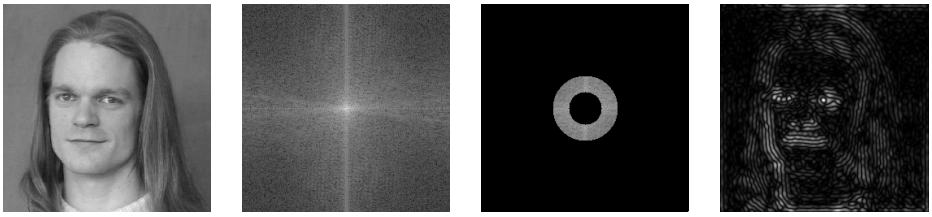
```
offset    = 1.
amplitude = 2.
frequency = 16.
phase     = np.pi

f = offset + amplitude * np.sin(frequency*x + phase)
```

Given this new array `f`, compute the corresponding `F` and plot it. Finally, recompute $f(x)$ with different choices of parameters, say, $o \in \{2, 4, 8\}$, $\alpha \in \{-2, 2\}$, $\nu \in \{1, 8, 64\}$, and $\phi \in \{0, \pi, 2\pi\}$. For each parametrization, compute the corresponding Fourier transform and plot it. What do you observe? What happens for different offsets? What about increasing frequencies? Or, what about various phases?

task 1.3: implementing a band pass filter

Write a program that reads an image $g(x, y)$, computes its Fourier transform $G(\mu, \nu)$, suppresses all frequencies outside the band r_{min} and r_{max} to obtain $\tilde{G}(\mu, \nu)$, and finally computes the inverse Fourier transform $\tilde{g}(x, y)$ of $\tilde{G}(\mu, \nu)$. Since a picture says a thousand words, here is an illustration of the process:



$g(x,y)$ $\log|G(\mu,\nu)|$ $\log|\tilde{G}(\mu,\nu)|$ $\tilde{g}(x,y)$

note: *in these illustrative images, we considered* $\log|G|$ *and* $\log|\tilde{G}|$ *only for visualization! However, for the filter to work properly, you must work with* G *and* $\tilde{G}$*!*
note: *everything you need to know for correctly solving this task will be explained in the lecture. Incorrect solutions will not be accepted*
Run you program on the image `clock.jpg`. Visualize the intermediate steps and final outcome of you code to demonstrate that it is working. Experiment with different choices for r_{min} and r_{max}. What do you observe?

task 1.4: exploring the importance of phase

Recall that the Fourier transform $F(\mu,\nu)$ of a function $f(x,y)$ is given by

$$F(\mu,\nu) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\, e^{-i(\mu x+\nu y)}\, dx\, dy.$$

Thus, while $f(x,y)$ is a real function, $F(\mu,\nu)$ is a complex-valued function. Further recall that any complex number $a+ib$ has a magnitude $\sqrt{a^2+b^2}$ and a phase $\arctan\frac{b}{a}$.
Write a program that reads two different images $g(x,y)$ and $h(x,y)$ of exactly the same spatial extension (e.g. `bauckhage.jpg` and `clock.jpg` in the Images folder of the Google site). Then, let you program compute the Fourier transforms $G(\mu,\nu)$ and $H(\mu,\nu)$, respectively. Next, compute the magnitude of all the entries of G and the phase of all entries of H. Now compute $K(\mu,\nu)$ where the magnitude of K comes from G and the phase information comes form H. Compute the inverse Fourier transform of K and have a look at the result! Explain what you see!

task 1.4

Prepare a presentation about your solutions and results for tasks 1.1, 1.3, and 1.4. When you are going to present your work (i.e. give a talk in front

of your fellow students and professor), your presentation should be based on up to 12 slides and not take more than 10 minutes. Make sure your presentation is clearly structured and answers questions such as “what is the task/problem we considered?”, “what kind of difficulties (if any) did we encounter?”, “how did we solve them?”, “what are our results?”, “what did we learn?”, . . .

general hints and remarks

- Send all your solutions (code, resulting images, slides) in a ZIP archive to bauckhag@bit.uni-bonn.de
- Remember that you have to successfully complete all three practical projects (and the tasks therein) to be eligible to the written exam at the end of the semester. Your grades (and credits) for this course will be decided based on the exam only, but –once again– you have to succeed in the projects to get there.
- Not handing in a solution implies failing the course.
- Your project work needs to be *satisfactory* to count as a success. Your code and results will be checked and your presentation needs to be convincing.
- If your solutions meets the above requirements and you can demonstrate that they work in practice, it is a *satisfactory* solution.
- A *good* to *very good* solution requires additional efforts especially w.r.t. to elegance and readability of your code. If your code is neither commented nor well structured, your solution is not good! A very good solution requires additional efforts towards the quality of your project presentation in the colloquium. Your presentation should be well timed, consistent, and convincing. You are also very much encouraged to extend your work to other images and to explore the behavior of the Fourier transform to a greater extend than asked for in this project. Be proactive and strive for very good solutions!.