project 3:

interpolation and image warping

solution(s) due:

January 21, 2019 at **12:00** via email to **bauckhag@bit.uni-bonn.de**

problem specification:

task 3.1 In the lecture, we discussed nearest neighbor, linear, and cubic spline interpolation; however, interpolation techniques are a dime a dozen. Another interesting approach is called *radial basis function interpolation*. Consider a set of data points $\{(x_i, y_i)\}_{i=1}^{n} \subset \mathbb{R}^2$ where we assume that

$$y_i = y(x_i). \tag{1}$$

In practice, we might then be interested in a non-linear interpolation between the given data samples. One way of how to accomplish this is to consider weighted linear combinations of radial basis functions

$$y(x) = \sum_{j=1}^{n} w_j \, \varphi_\sigma\Big(\big\|x - x_j\big\|\Big) \tag{2}$$

where

$$\varphi_\sigma(s) = e^{-\frac{s^2}{2\sigma^2}}. \tag{3}$$

Looking at the model in (2), the obvious question is how to learn suitable weights w_j from the given data? The answer is simple; since

$$y(x_i) = y_i = \sum_{j=1}^{n} w_j \, \varphi\Big(\big\|x_i - x_j\big\|\Big) = \sum_{j=1}^{n} w_j \, \Phi_{ij} \tag{4}$$

we are dealing with a system of n equations to determine n unknowns. If we collect the y_i in a vector $\boldsymbol{y} \in \mathbb{R}^n$, the w_j in a vector $\boldsymbol{w} \in \mathbb{R}^n$, and the Φ_{ij} in a matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}$, the system in (4) can be written as

$$\boldsymbol{\Phi}\,\boldsymbol{w} = \boldsymbol{y} \tag{5}$$

so that

$$\boldsymbol{w} = \boldsymbol{\Phi}^{-1}\boldsymbol{y}. \tag{6}$$

In other words, all we need to do is to compute a similarity matrix Φ for the x-values in our data, invert it and multiply it to the vector $\boldsymbol{y}$ of given y-values to obtain proper weights for the model in (2).
Now, run the following *NumPy* snippet

```
import numpy as np

n = 20
x = np.arange(n) + np.random.randn(n) * 0.2
y = np.random.rand(n) *  2
```

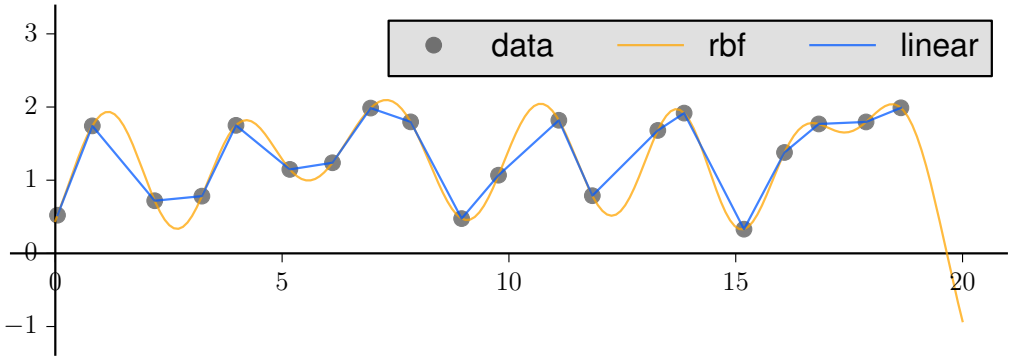
to produce data from which to determine an interpolation function $y(x)$.
Once you have determined this function for a choice of σ, apply it to

```
xs = np.linspace(0,n,200)
```

and plot the result. Experiment with different choices of σ, for instance,

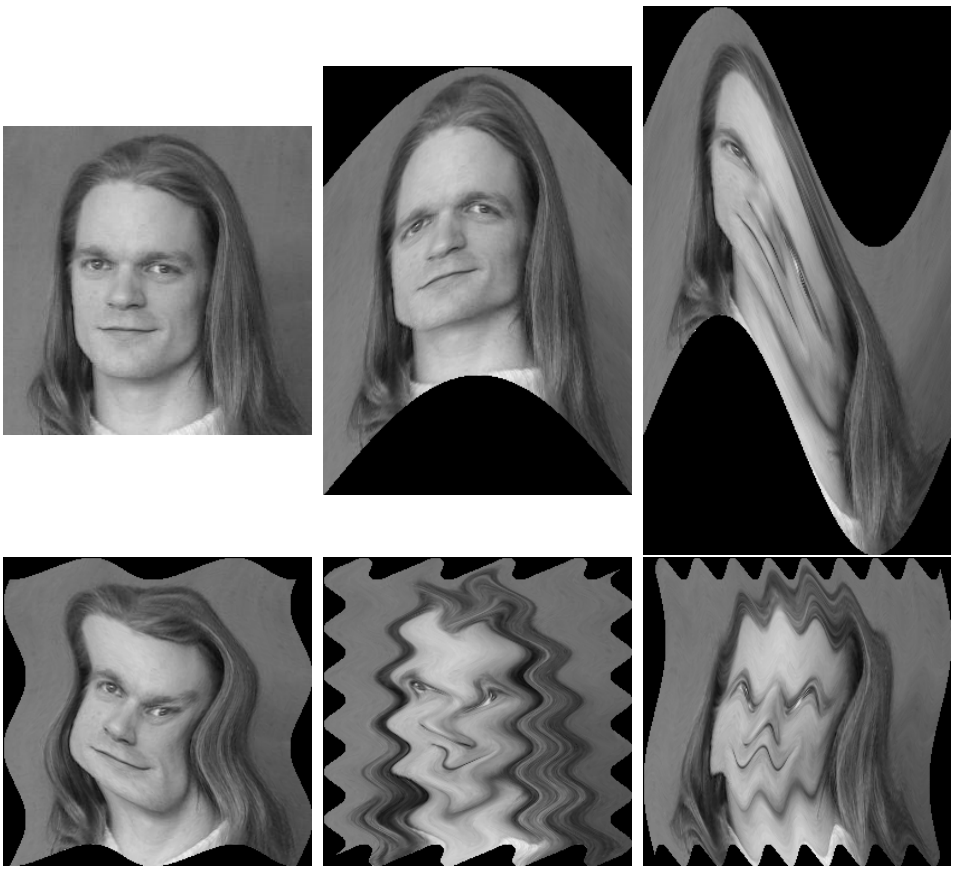
$$\sigma \in \{0.5, 1, 2, 4\}. \tag{7}$$

What do you observe?
1st hint: the following figure gives an impression as to how the data generated above should look like; it also shows an RBF interpolation and, for baseline comparison, a piecewise linear interpolation of the data.



2nd hint: functions for matrix inversion can be found in the *NumPy* module `numpy.linalg`.

task 3.2 Implement a program that computes warps of the following kind:

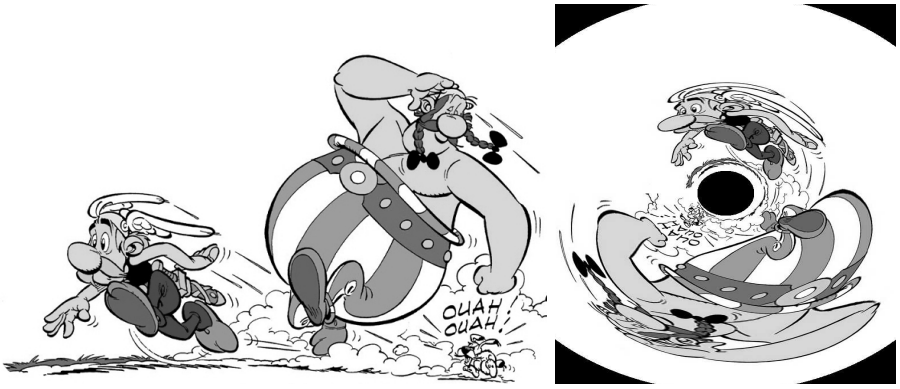


Apply you implementation to the image `clock.jpg` and experiment with different parametrizations.
1st hint: Recall what you have learned about wave functions. Which mathematical function featured prominently in this context? Which role could the parameters frequency ν, amplitude α, and phase ϕ play in the above pictures? How would you chose these parameters to create the middle and rightmost picture in the upper row?
2nd hint: using *NumPy* / *SciPy*, image warping can be simple; the *SciPy* module `scipy.ndimage` provides a function `map_coordinates` that abstracts away all the nasty interpolation details.

task 3.3 *Cylinder anamorphosis* was a popular form of art in the 18th century where images were drawn in a warped fashion. They would appear un-warped when viewed in a cylindrical mirror.

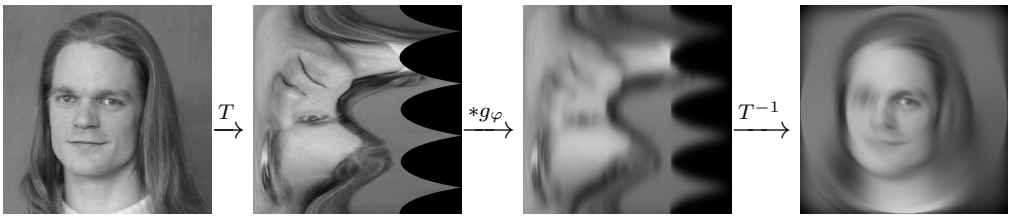
a) Implement a program that can compute pictures such as these:

b) Next, extend your program such that it not confined to mapping images onto radial discs but can also map them onto a torus:



Apply your implementation to the image `clock.jpg` and experiment with different parametrizations.

task 3.4 In the lecture, we discussed the idea of representing an image in (r, φ) coordinates rather than in (x, y) coordinates.
We also saw, that interesting effects result from applying Gaussian smoothing to this representation. In particular, if we transform from (x, y) to (r, φ) coordinates, apply a 1D Gaussian filter along the φ dimension of the image, and then transform it back into (x, y) coordinates, we obtain a circularly blurred image as seen below:



x, y image | r, φ image | φ-axis blurred | new x, y image

Implement a program that realizes this chain of processing steps. Apply it to the image `clock.jpg` and experiment with different choices for the variance of the Gaussian filter kernel.

task 3.5 In the lecture, we discussed perspective mappings between quadrilaterals to create simple augmented reality effects like this:

Implement a program that maps the image `clock.jpg` onto the poster seen in the image `isle.jpg`. Assume that the four corners of the poster coincide with following (x, y) coordinates

$$\boldsymbol{x}_{ul} = \begin{bmatrix} 215 \\ 56 \end{bmatrix} \quad \boldsymbol{x}_{ur} = \begin{bmatrix} 365 \\ 10 \end{bmatrix} \quad \boldsymbol{x}_{ll} = \begin{bmatrix} 218 \\ 258 \end{bmatrix} \quad \boldsymbol{x}_{lr} = \begin{bmatrix} 364 \\ 296 \end{bmatrix}$$

where the origin $[0, 0]^T$ of the image `isle.jpg` is supposed to be located in its upper left corner.

task 3.6 Prepare a presentation about your solutions and results just as you did in the previous two projects.

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!.