project 1:

image intensities

solution(s) due:

June 8, 2017 at **12:00** via email to **bauckhag@bit.uni-bonn.de**

problem specification:

task 1.1: Lloyd Max algorithm Implement the Lloyd Max algorithm for grey value quantization. Proceed as follows:

- assume that $h(\lambda)$ is the intensity histogram of a grey value image $g(\boldsymbol{x})$; note that $\lambda \in [0, \ldots, 255]$
- turn $h(\lambda)$ into a density function $p(\lambda)$ using the transformation

$$p(\lambda) = \frac{h(\lambda)}{\sum_y h(y)}$$

- choose a number L of quantization levels, e.g. $L = 8$
- initialize the boundaries a_ν of the quantization intervals as follows:

$$\begin{aligned} a_0 &= 0 \\ a_\nu &= \nu \cdot \frac{256}{L} \\ a_{L+1} &= 256 \end{aligned}$$

- initialize the quantization points b_ν as follows:

$$b_\nu = \nu \cdot \frac{256}{L} + \frac{256}{2L}$$

- now, iterate the equations

$$a_\nu = \frac{b_\nu + b_{\nu-1}}{2}$$

and

$$b_\nu = \frac{\displaystyle\int_{a_\nu}^{a_{\nu+1}} \lambda p(\lambda) d\lambda}{\displaystyle\int_{a_\nu}^{a_{\nu+1}} p(\lambda) d\lambda}$$

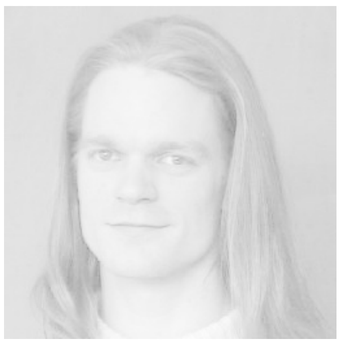
until the quantization error

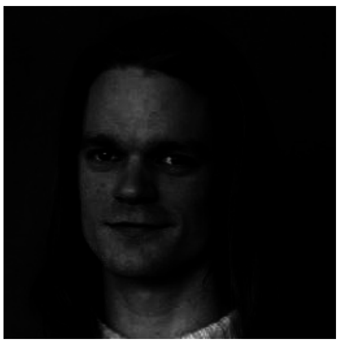
$$E = \sum_{\nu=1}^{L} \int_{a_\nu}^{a_{\nu+1}} (\lambda - b_\nu)^2 p(\lambda) d\lambda$$

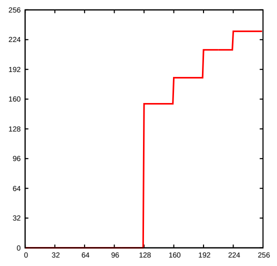
drops below a threshold or does not improve significantly anymore, or until a maximum number t_{max} of iterations has been carried out.

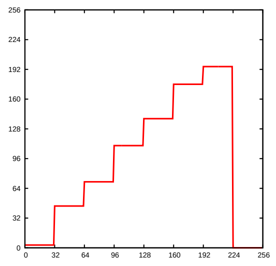
- now, apply your program to a bright and to a dark image such as, for instance,
  - `bauckhage-gamma-1.png`
  - `bauckhage-gamma-2.png`

  Have a look at the resulting quantization levels. Do your results look like in the following examples where $L = 8$?









- explain the characteristics of the quantization curves
- repeat the above exercise, this time initially choosing

$$\begin{aligned} a_0 &= \min\{\lambda \in g(\boldsymbol{x})\} \\ a_{L+1} &= \max\{\lambda \in g(\boldsymbol{x})\} \\ a_\nu &= \nu \cdot \frac{a_{L+1} - a_0}{L} \end{aligned}$$

task 1.2: histogram transformations In the lecture, we discussed the idea of histogram transformations; in particular, we looked at histogram equalization

$$T(\lambda) = \left\lfloor \frac{H(\lambda)}{N} \cdot 255 \right\rfloor$$

where

$$H(\lambda) = \sum_{i=0}^{\lambda} h(i).$$

Discussing the underlying theory, we found that histogram equalization is a special case of transforming a random variable $X \sim p_X(x)$ into a random variable $T(X) = Y \sim p_Y(y)$ where

$$p_Y(y) = p_X\big(T^{-1}(y)\big) \left| \frac{d}{dy} T^{-1}(y) \right| .$$

Can you transform the intensity histogram of an image such that it becomes Weibull distributed? The pdf and cdf of the Weibull distribution are given by

$$p_Y(y) = \frac{k}{l} \left(\frac{y}{l}\right)^{k-1} \exp\left\{ - \left(\frac{y}{l}\right)^{k} \right\}$$

$$P_Y(y) = 1 - \exp\left\{ - \left(\frac{y}{l}\right)^{k} \right\}$$

Apply you transform to an image of your liking and create plots of the histogram before and after transformation. Good choices for the parameters k and l are 2.0 and 40.0, respectively.

Note: Compared to most of our tasks in this course, this one may seem to pose a more advanced problem. But you can definitely solve it!

task 1.3: illumination compensation Implement an algorithm for illumination compensation in images; proceed as follows:

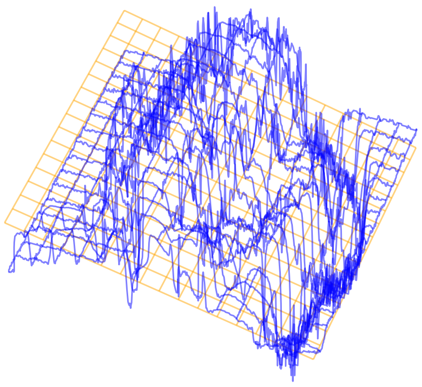
- assume a grey value image results from a multiplicative interplay of illumination and reflectance, i.e.

$$f(x,y) = i(x,y) \cdot r(x,y)$$

- compensate for illumination using either a linear or bilinear model of the illumination component $i(x,y)$
- apply your implementation on `portrait.png` and `cat.png`

  and visualize your results

- also create a 3D plot of the image function $f(x,y)$ together with the resulting illumination model $i(x,y)$; for instance, for the image `portrait.png`, you should obtain something like



- now, instead of estimating the illumination model from the set of all pixels in the image, consider a random sample of 250 pixels and estimate your illumination model therefrom; experiment with different sample sizes; how does sampling affect the overall result of illumination compensation?

general hints and remarks

- Send all your solutions (code, resulting images, slides) in a ZIP archive to bauckhag@bit.uni-bonn.de

  note: if you are implementing in C/C++, you may use whatever libraries appear useful. It is, however, suggested you use python and the Image and scipy modules for all your practical work. If you insist on using a language other than python, you have to figure out elementary image processing functions/toolboxes in these languages by yourself. **Implementations in MATLAB will be rejected.**

- For the tasks in this project you may use the pictures in the Images folder of the Google site for this course to demonstrate that your code is working.

- 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. Striving for very good solutions should always be your goal!