4. Image Processing - Intensity Transformations

Institute for Biomedical Imaging, Hamburg University of Technology

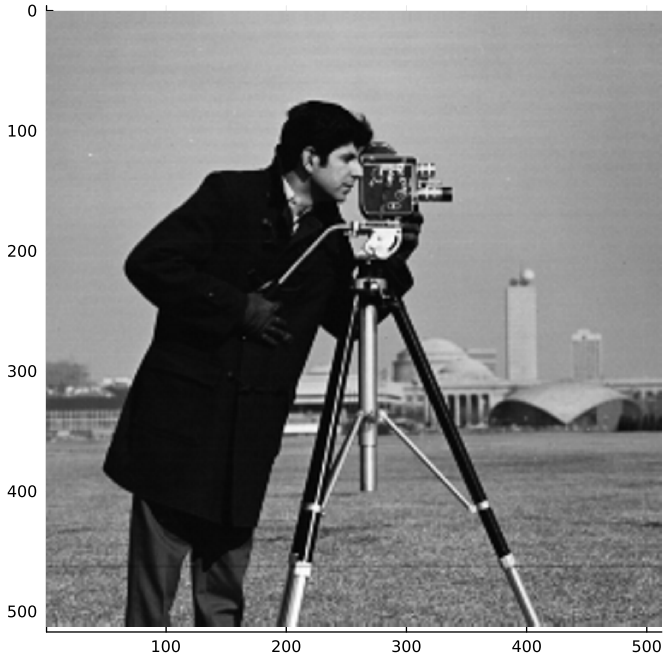
- Lecture: Prof. Dr.-Ing. Tobias Knopp
- Exercise: Konrad Scheffler, M.Sc.

Table of Contents

4.1 Background

In this lecture we focus on so called intensity based transformations that take no neighborhood of a pixel into account but work only on the intensity of a pixel. The transformation is usually non-linear.

In this lecture we will mainly use the camera man:



4.1.1 Histogram

Before we start with the transformations we first define the histogram of an image. It is a function $\mathcal{H}_f^L(l)$ reporting the number of pixels that have a certain value. For integer valued images one can simply take $l = 1, \dots, L$ where L is the number of possible values. For continuous function one can discretize the range $[0, 1]$ into L so-called bins and count the number of pixel values in each bin. Here, L can be arbitrarily chosen and defines basically the zooming factor being used.

Let's define this mathematically. Let $b_l = [\frac{l-1}{L}, \frac{l}{L})$, $l \in I_{L-1}$, and $b_L = [\frac{L-1}{L}, 1]$ be our bins. Then we use the indicator function $\chi_T \colon [0,1] \to \{0,1\}$ for an interval $T \subseteq [0,1]$ with

$$\chi_T(x) = \begin{cases} 1, & \text{if } x \in T, \\ 0, & \text{if } x \notin T \end{cases}$$

and can then define the histogram function as $\mathcal{H}_f^L : I_L \to \mathbb{N}_0$

$$\mathcal{H}_f^L(l) = \sum_{\mathbf{r} \in \Omega} \chi_{b_l}(f(\mathbf{r})).$$

This formula has two different discretizations:

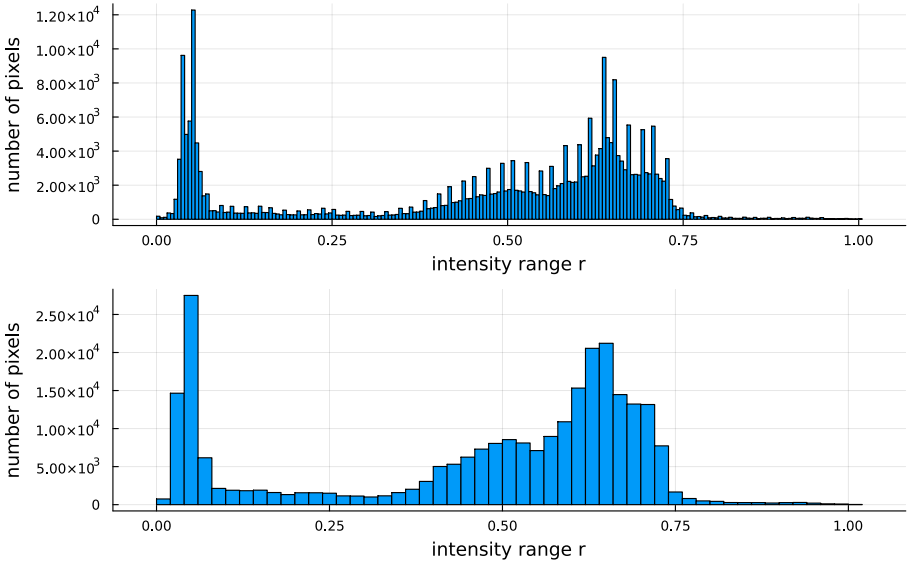
1. The number of bins L. In the limit $L \to \infty$ the domain of $\mathcal{H}_f^L$ would become $[0,1]$ since we make the bins smaller and smaller.
2. The discretization of space Ω. If that is chosen continuous we end up at a range of the histogram that would be $\mathbb{R}^D$.

Note

The histogram can be seen to be the probability density function, which describes the probability that a random number (= image pixel) has a certain value.

Since the histogram represents bins it is usually shown using a bar plot instead of plot with connected lines.

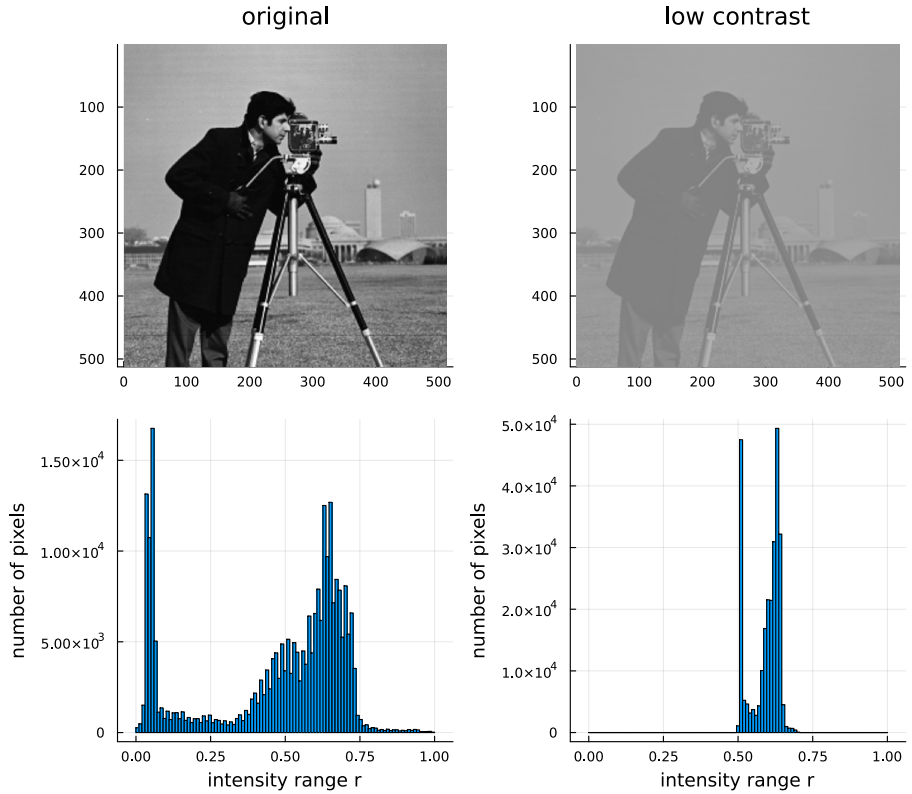
Let us have a look at the histogram of the camera man for two different bin sizes:



Observations

- One can see two peaks, one sharper one in the lower intensity range. This covers all the dark parts like the jacket and the hair. The other peak is the sky which has a light gray tone.
- With fewer bins one looses some resolution but the distribution is still clear.

One important question is, what we can learn from a histogram. Let us have a look at the following two images:



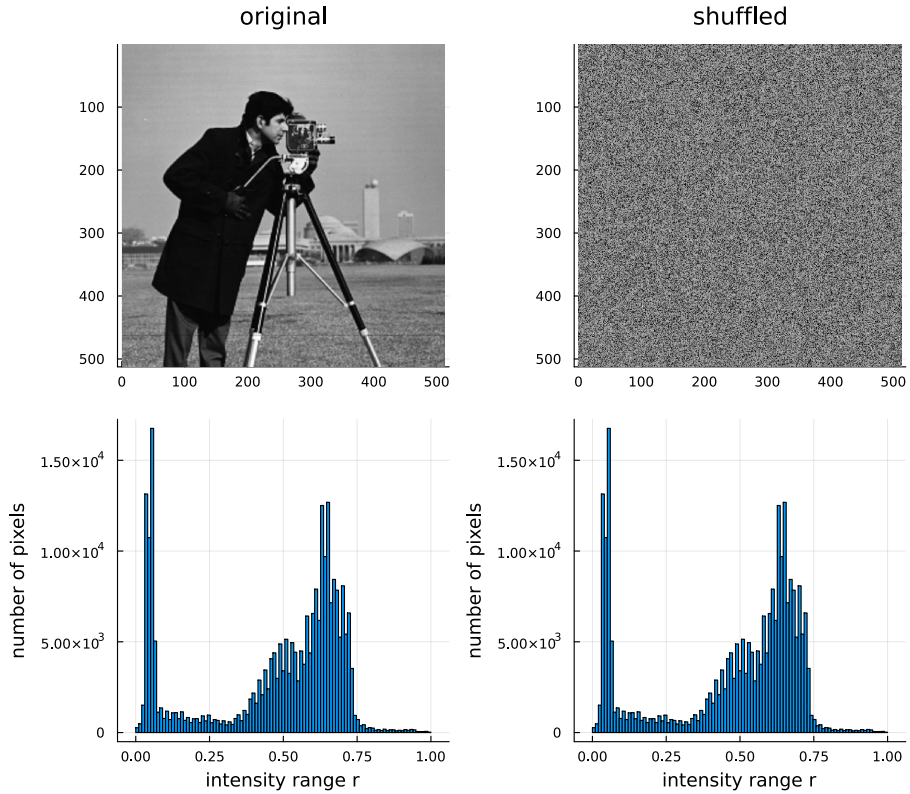
Observations

- The left picture looks fine, the right picture has much less contrast making it more difficult to distinguish objects.
- This can be directly seen in the histogram. While the left picture covers the entire range of values $[0, 1]$, the right just covers the range $[0.5, 0.75]$.

➔ Key goal of histogram processing methods is to improve the coverage of values of the range of the function.

Note

One important property of the histogram is that the location of a pixel does not matter. Thus, if we shuffle the pixels this will not change the histogram.



4.2 Basic Intensity Transformations

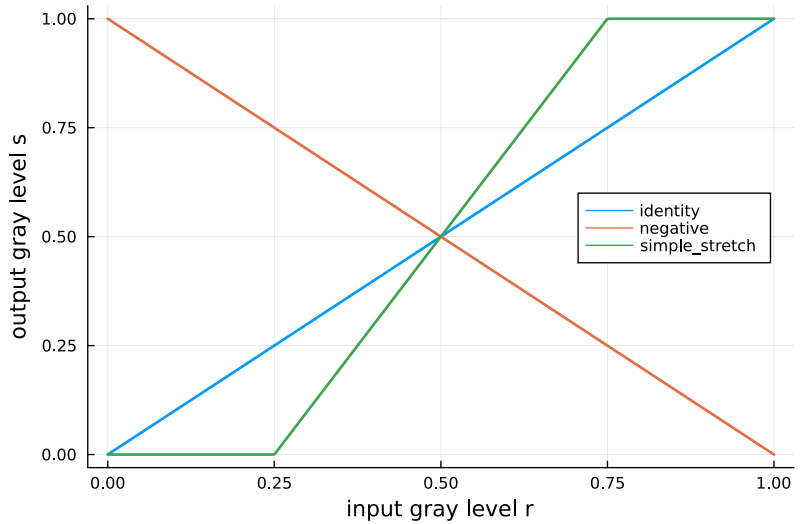
An intensity based transformation takes an element $f(x, y)$ of an image and transforms it independently of any neighboring pixels. Such transformations can be written as

$$T : \Gamma \to \Gamma, \quad r \mapsto s$$

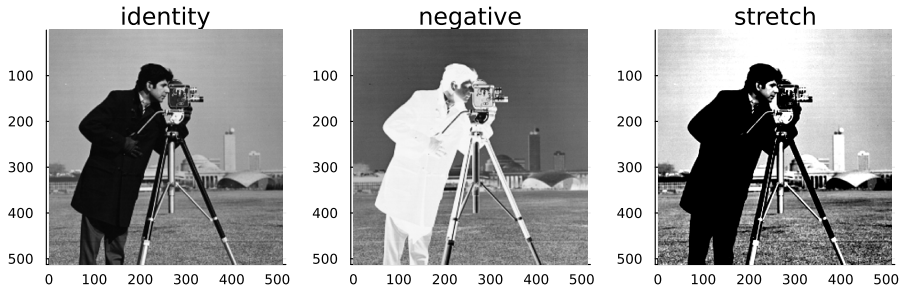
Example transformations are:

$$T_{\text{identity}}(r) = r$$
$$T_{\text{negative}}(r) = 1 - r$$
$$T^{\alpha}_{\text{stretch}}(r) = \begin{cases} 0 & \text{if} \quad r < \alpha \\ 1 & \text{if} \quad r > 1 - \alpha \\ \frac{r-\alpha}{1-2\alpha} & \text{else} \end{cases}$$

They can be illustrated by plotting T against r:



Let us apply them to the camera image:



Observations

- One can see how colors are flipped in the negative case, which is sometimes used in medical images to optimize the visual perception.
- For the contrast-stretched image one can see that the grass got much more structure but other feature got *saturated* which means that they got mapped to the same value. In particular the pants are now fully saturated.

Note

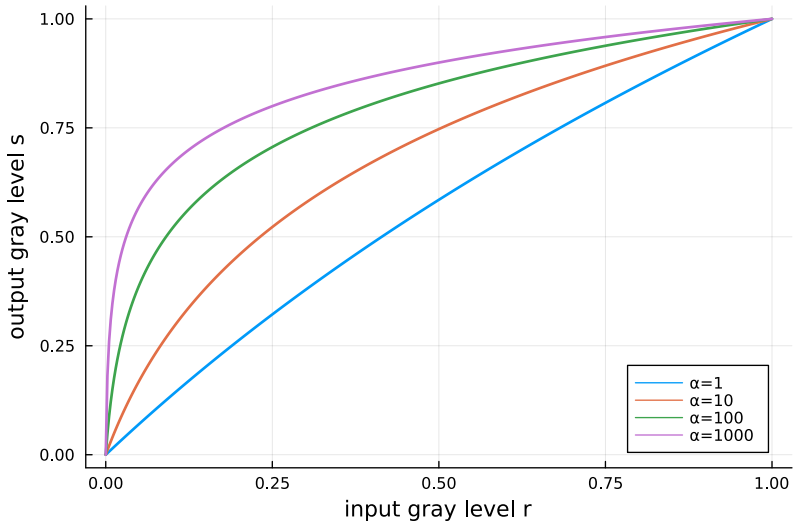
Recall at this point that the visual perception has not the same sensitivity across the intensity range. Thus the perception can change when applying a negative or a more advanced transformation.

4.2.1 Log Transformations

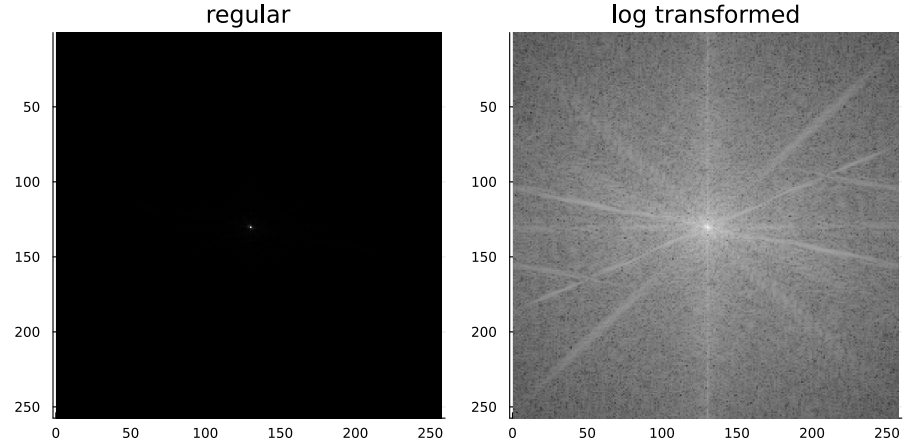
An important class of transformations are the so-called *log transformations* that can be expressed as

$$T_{\log}^{\alpha,c}(r) = c\log(1+\alpha r)$$

where c is a constant. When choosing $c = \frac{1}{\log(1+\alpha)}$ the mapping goes from $[0,1]$ to $[0,1]$ and α is the parameter that influences what range is focussed on.



Log transformations are in particular helpful if the intensities are stretched over a very large range that the human eye could not resolve (simultaneously). Let us have a look at the absolute value of the Fourier transform of the camera man:



Observations

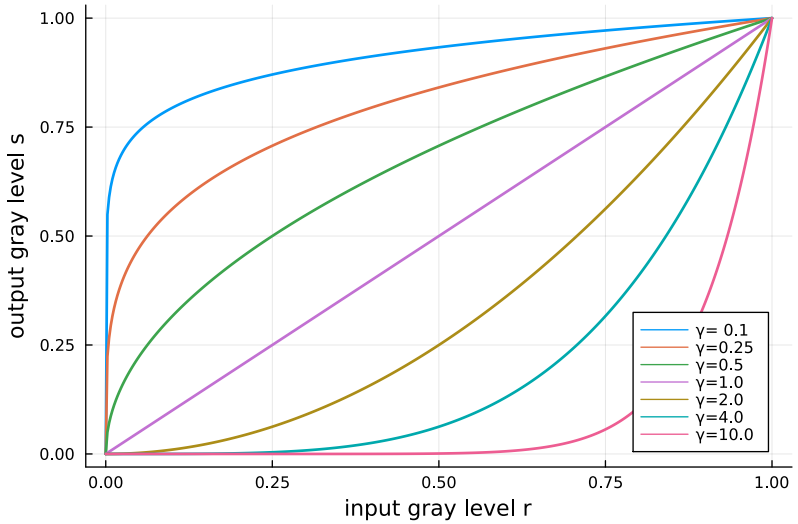
- In the left image one can basically only see the DC part of the image (i.e. the central pixel being white)
- The right image gives a much better overview of the structure of the Fourier coefficients.

4.2.2 Power Law Transformations

An alternative to the log transformations are the power law transformations, which are defined as

$$T_{\text{pow}}^{\gamma,c}(r) = cr^{\gamma}.$$

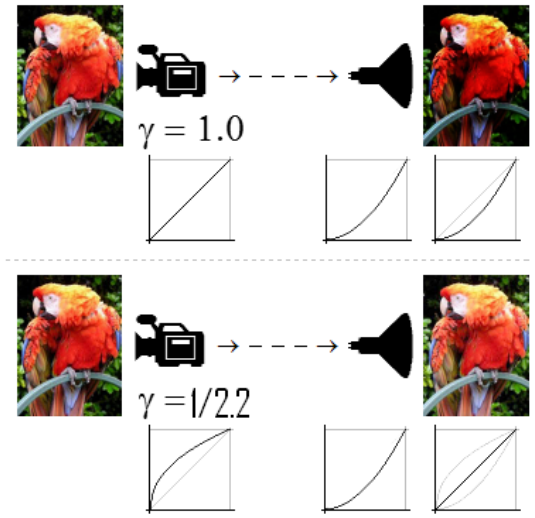
They look like this:



The power transformation is also known as the gamma correction and is important since it is used to correct non-linear intensity ranges of output devices such as monitors. Cathode ray tube (CRT) monitors had an intensity-to-volt response following a power law and in turn required a gamma correction.

Despite CRT monitors not being used anymore, Gamma correction is still used for LCD displays. The reason is that it allows to correct for the non-uniform perception of the human eye. In practice a gamma value of $\gamma = 2.2$ is used most of the time to account for this effect.

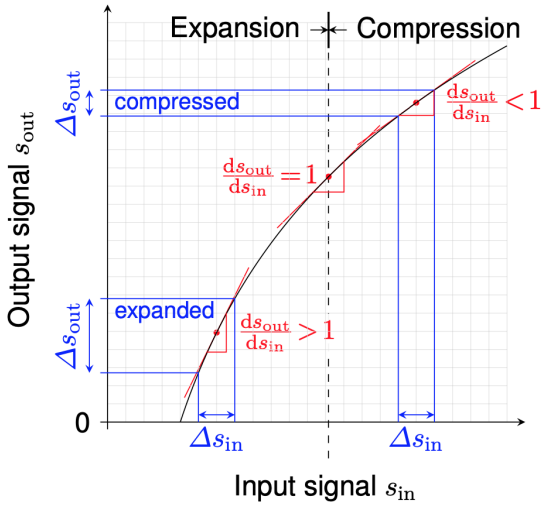
The following image shows a typical data flow. Often the images are pre-corrected with a gamma value of $\gamma_{\text{acq}} = \frac{1}{2.2}$ to account for the gamma correction with $\gamma_{\text{monitor}} = 2.2$ applied in the monitor.



Compression / Expansion

When looking at the power law transformation we can see that there are steeper and flatter regions. We can analyse this using the derivative $\frac{dT^{\gamma,c}_{pow}(r)}{dr} = c\gamma r^{\gamma-1}$ and differentiate the following three cases:

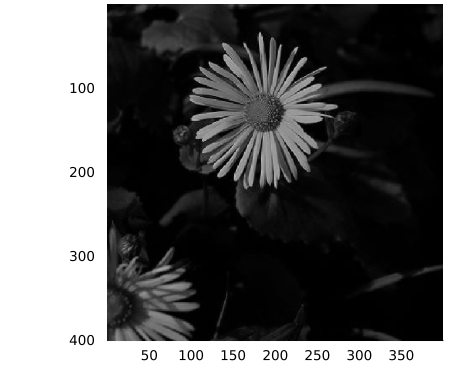
1. No change: $\frac{dT^{\gamma,c}_{pow}(r)}{dr} = 1$
2. Compression: $\frac{dT^{\gamma,c}_{pow}(r)}{dr} < 1$
3. Expansion: $\frac{dT^{\gamma,c}_{pow}(r)}{dr} > 1$

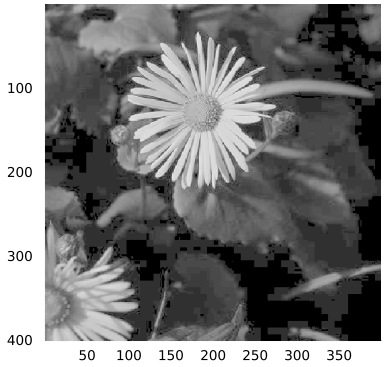


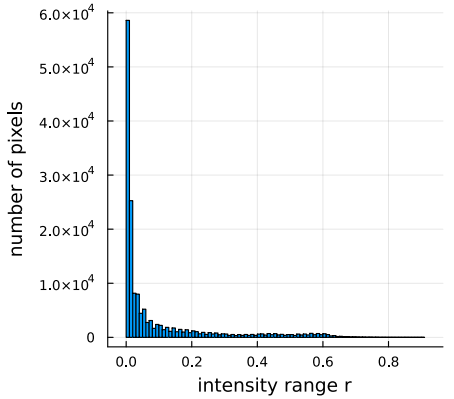
In the *expansion* case the range is stretched and in turn more details can be discriminated. In the *compression* case details get lost.

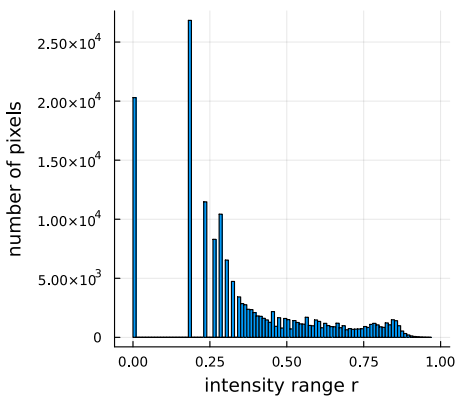
Images with wrong Illumination

The following shows an image that has been captured with too few light. By applying a gamma correction we can improve this. This can also be seen in the histogram which is better distributed after gamma correction.









4.2.3 Contrast Stretching

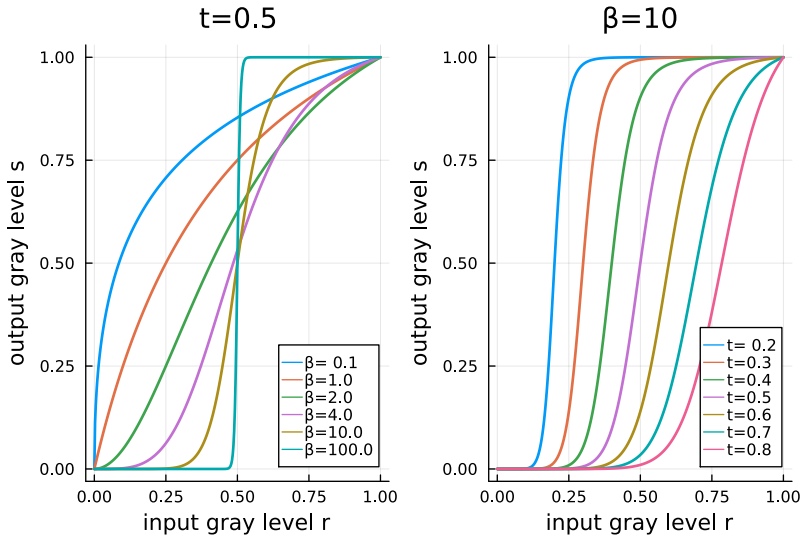
The power law transformations were compressing one side of the histogram while expanding the other side. Often the histogram simply captures not the entire range well in which case the histogram needs to be *shifted* and *stretched* in a more flexible manner. This can be done by the transformation

$$T^{\beta,t}_{\text{stretch}}(r) = \frac{1}{1+\left(\frac{t}{r}\right)^{\beta}}.$$

A normalized version can be defined as

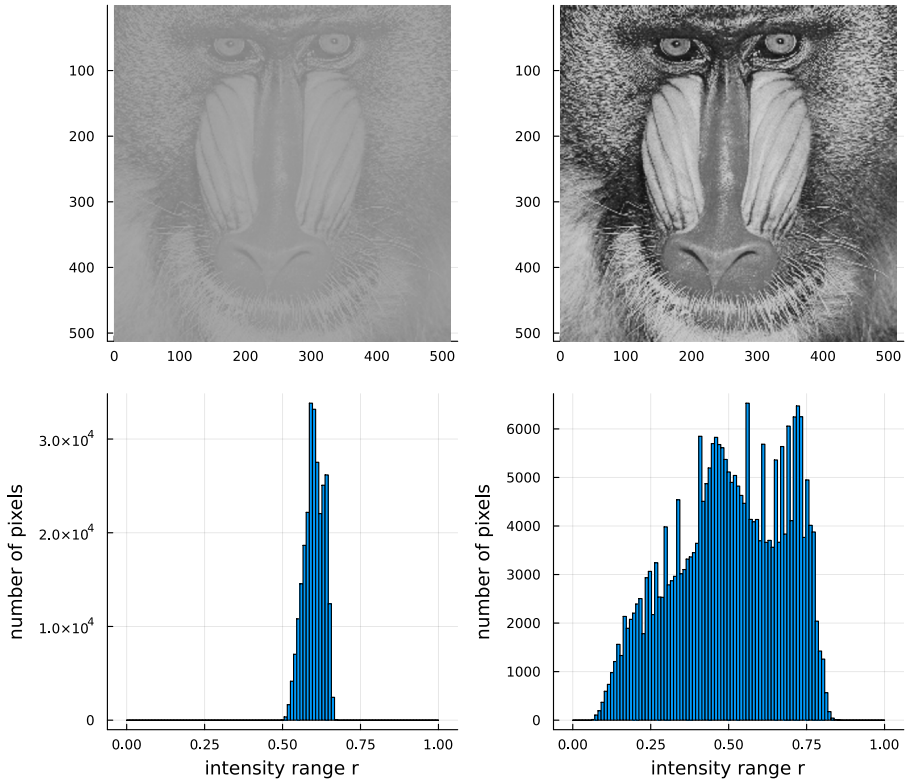
$$T^{\beta,t}_{\text{stretch, norm}}(r) = \frac{1+t^{\beta}}{1+\left(\frac{t}{r}\right)^{\beta}}.$$

It looks like this:



One can see that t basically shifts the curve, while β influences the steepness.

Lets apply it to an image with low contrast:



Here, have a special focus on the x-axis of the histogram, which does not cover the entire range $[0, 1]$ before streching.

4.3 Histogram Processing

We next focus on more advanced histogram processing techniques. The one applied until now were model-based parameterized functions. We next look at methods that aim at taking the histogram itself into account when building the transformation function.

4.3.1 Histogram Equalization

The most prominent and important one is called histogram equalization. It aims at making the histogram flat and let it cover the entire range. For the image $f : \Omega \to [0, 1]$ it can be expressed as

$$T^{f,L}_{\text{equalization}}(r) = \frac{1}{|\Omega|}\int_0^r \mathcal{H}^L_f(l)\mathrm{d}l$$

We have used here a continuous representation of the histogram and thus integrate over it.

To understand this formula better, let us insert two edge cases:

- For $r = 0$ we have $T^{f,L}_{\text{equalization}}(0) = \frac{1}{|\Omega|}\int_0^0 \mathcal{H}^L_f(l)\mathrm{d}l = 0$
- For $r = 1$ we have $T^{f,L}_{\text{equalization}}(1) = \frac{1}{|\Omega|}\int_0^1 \mathcal{H}^L_f(l)\mathrm{d}l = \frac{|\Omega|}{|\Omega|} = 1$

It is also clear that the function $\mathcal{H}^L_f$ is increasing monotonically since the histogram is always positive.

What basically happens in histogram equalization is that one sweeps through the gray value range and applies a slow increase when the histogram has only few values and applies a large increase if the histogram has large values. Since a large derivative implies that we spend less of our quantization range, an equalization is achieved.

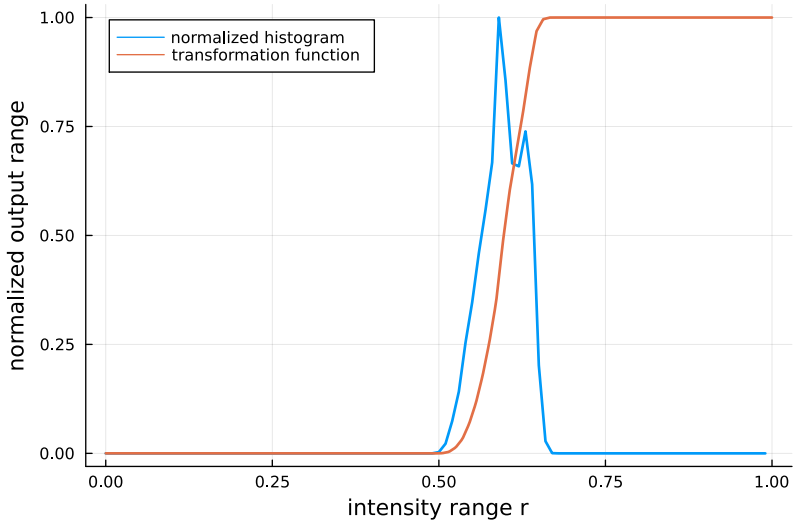
> **Note**
>
> Quite often histogram equalization is formulated using a cumulative sum. In that case one needs to use some interpolation method to achieve a continuous range $[0, 1]$ for r. In the following we use that approach in our implementation.

Implementation and Example

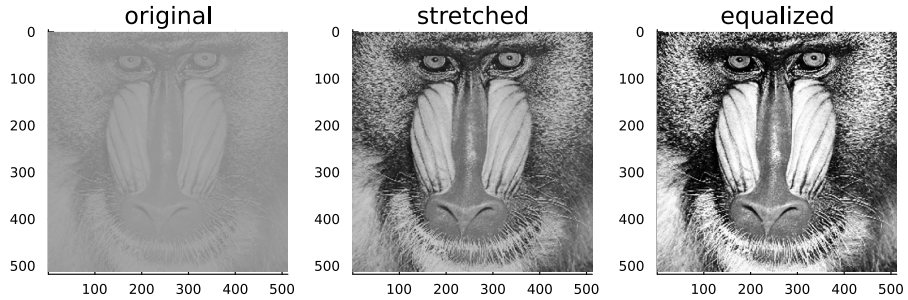
The following shows an example implementation for the histogram equalization method

```
histEqual (generic function with 1 method)
```

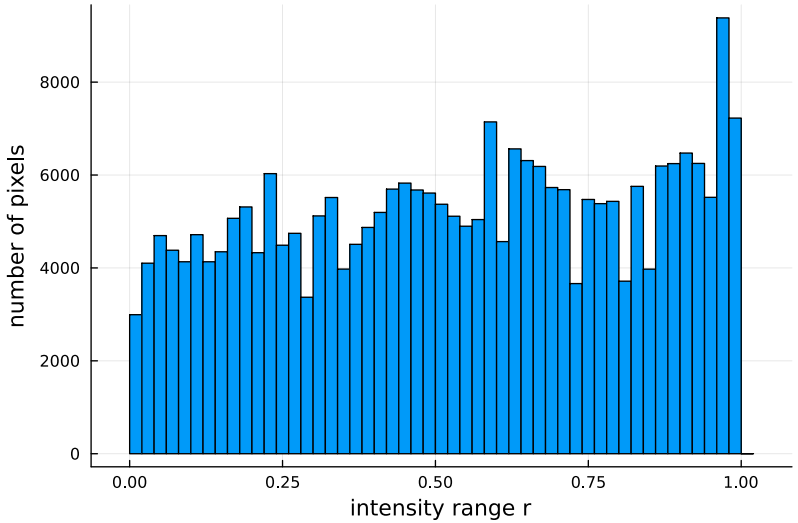
```
function histEqual(r; h)
    hc = cumsum(h)
    hc = hc./hc[end]
    hci = interpolate(hc, BSpline(Linear()))
    return hci(1+r*(length(hc)-2))
end
```



Let us apply this to the image considered before:



Here is the histogram of the equalized image:



Observations

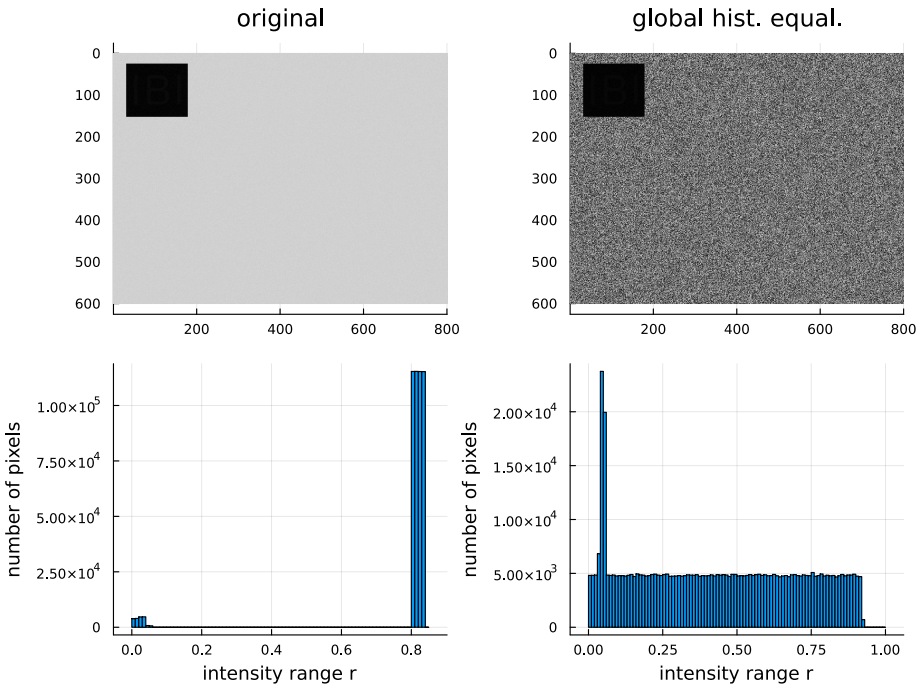
- The histogram looks much better. The entire range $[0, 1]$ is covered.
- The histogram is not perfectly flat, which is due to the discrete processing. There are algorithms for generating a perfect flat histogram, which add some noise to image pixels to move them to neighboring bins. In practice, a perfect equalization is, however not really necessary and the issue discussed in the next section is actually more important (which is in conflict with perfect histogram equalization).

4.3.2 Local Histogram Equalization

The intensity transformations discussed so far are all global und thus do not take into account the local statistics of different areas in an image. This can be problematic in case that an image is partitioned into different areas with different statistics.

The following image shows the issue. The image has two different regions, a large background in light gray and a very dark small object in the upper left corner. Within the block there is a small text chunk **IBI** which is almost black but has a slightly lighter tone than the black box.

Now lets us perform a global histogram equalization:



Observation

- One can see that the global histogram equalization does not change the color of the text region such that the observer can distinguish it from the black box.

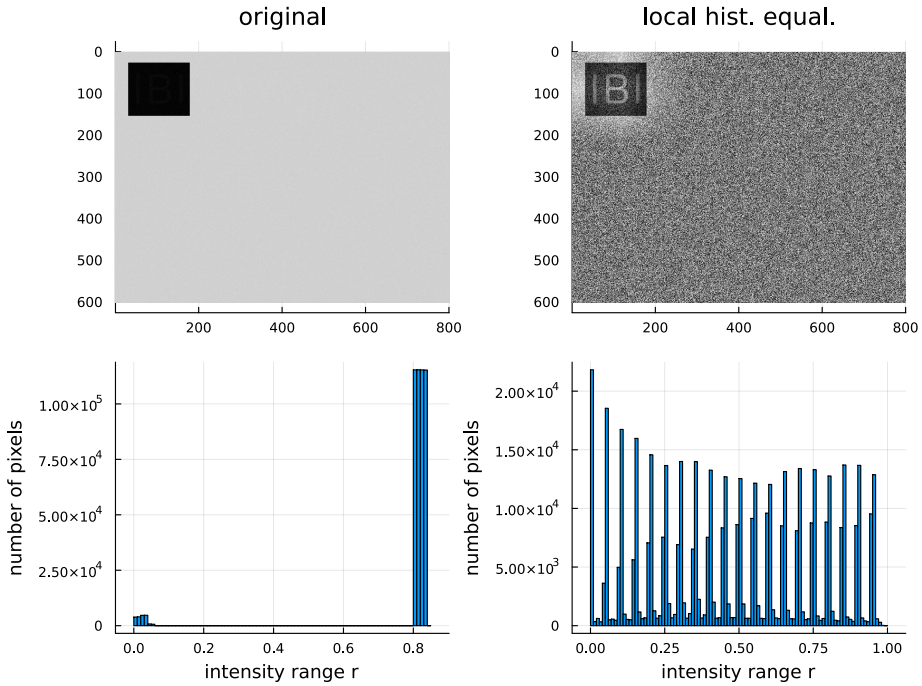
Explanation

- The amount of pixels covered by the text is much smaller than the remaining pixels and in turn it only slightly impacts the histogram equalization.

To solve this issue one can apply a local histogram equalization method. To this end the image f is divided into $C_x \times C_y$ blocks:

$$\mathbf{f} = \begin{pmatrix} \mathbf{f}_{1,1} & \cdots & \mathbf{f}_{1,C_y} \\ \vdots & \vdots & \vdots \\ \mathbf{f}_{C_x,1} & \cdots & \mathbf{f}_{C_x,C_y} \end{pmatrix}$$

Then, a local histogram equalization transformation function $T_{i,j}$ is build up for each region. When finally applying the transformation, an interpolation function is used to generate smooth transitions from one region to the next one. Here you can see what this looks like in practice:



4.4 Wrapup

- In this lecture we have shown how one can perform processing on the *range* of a function independently of the pixel neighborhood.
- This allowed us to optimize the *contrast* of images.
- These transformations are usually *non-linear*.
- They are usually based on image statistics, i.e. they take the histogram as input.
- In some cases they need to operate locally, in case that the image statistic varies.