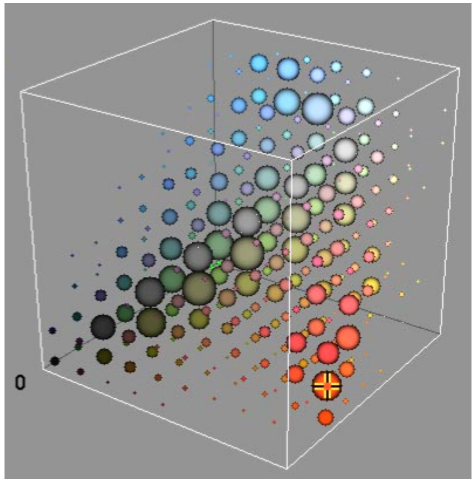
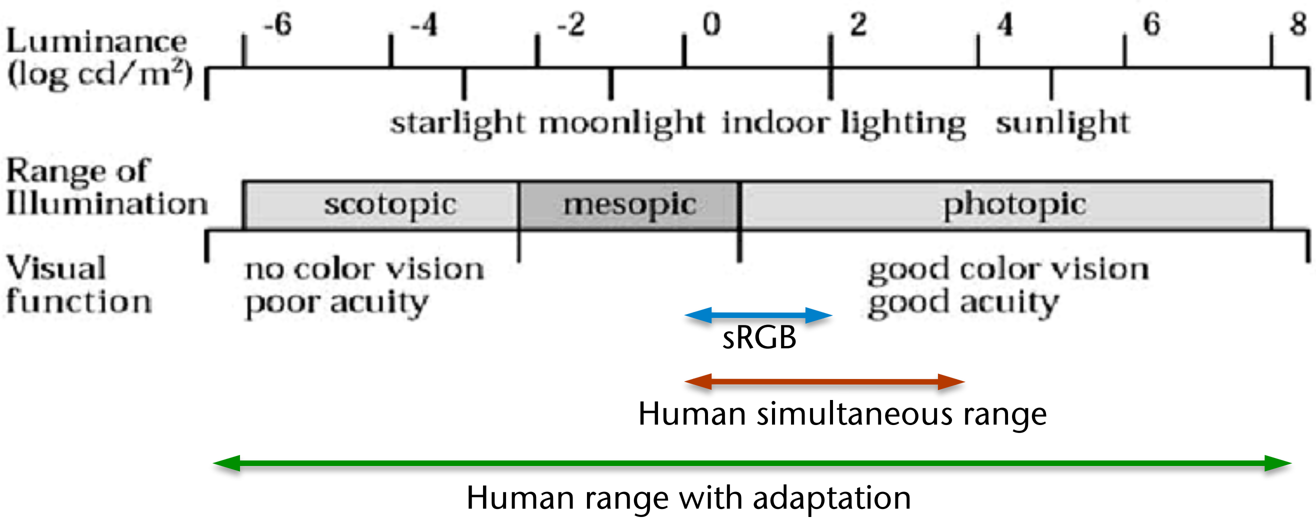
Advanced Computer Graphics
Tone Mapping / Tone Reproduction

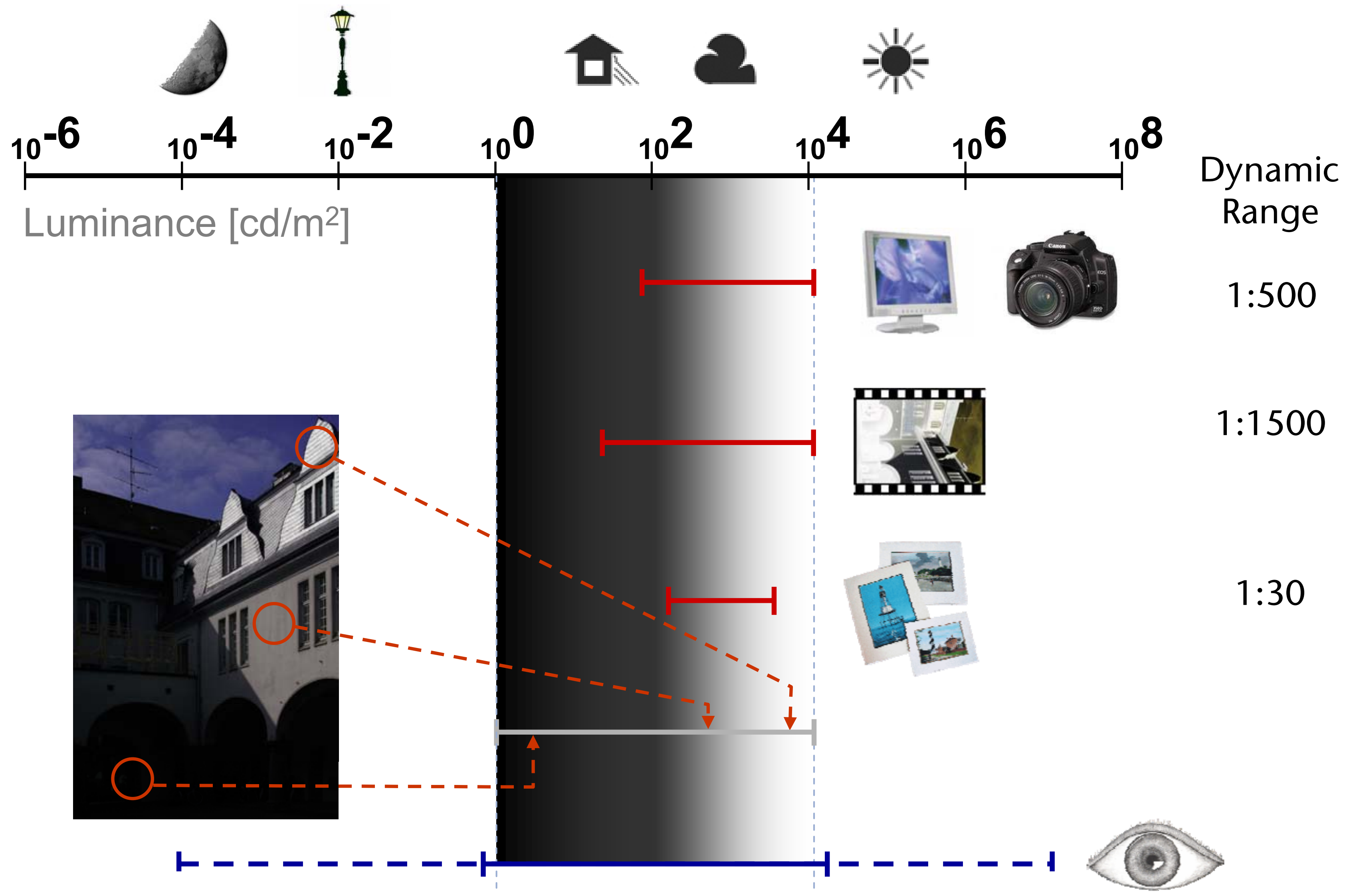


G. Zachmann
University of Bremen, Germany
cgvr.informatik.uni-bremen.de

Motivation

- Definition:
  - The dynamic range of an image is the contrast ratio between the brightest and darkest parts
  - The dynamic range of a display or optical sensor is the ratio of the brightest representable or perceived luminance to the darkest
- The dynamic range of the human visual system:

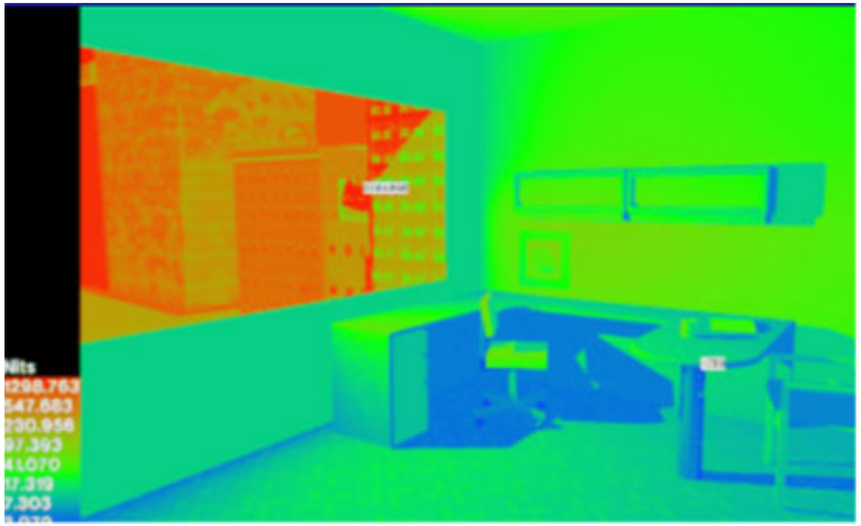




10-6
10-4
10-2
100
102
104
106
108
Luminance [cd/m2]
Dynamic Range
1:500
1:1500
1:30

Sources of High Dynamic Range Images (HDRI)

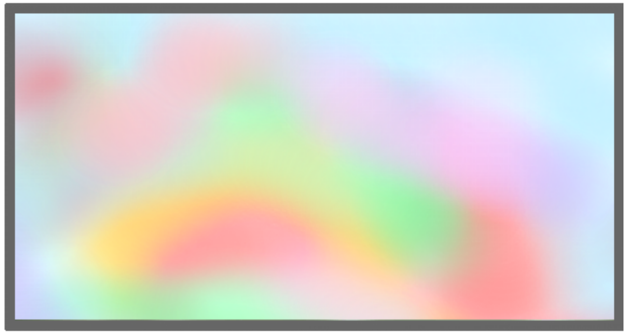
- Ray-Tracing: physically accurate synthetic images
- Photography:
  - Several shots with different exposure times
  - "Blending" together (needs calibrated response curve from camera)



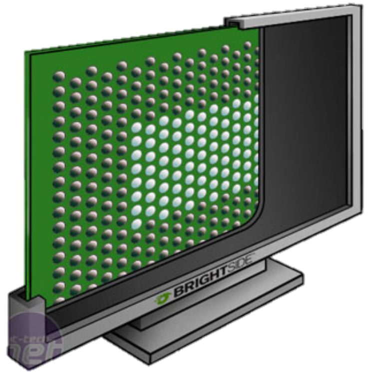
- And in games, too, to some extent:

Lost Planet: Extreme Condition, PC version, 2007
(not known, exactly what kind of HDRI / tone mapping was done)

- Use either real HDR displays …



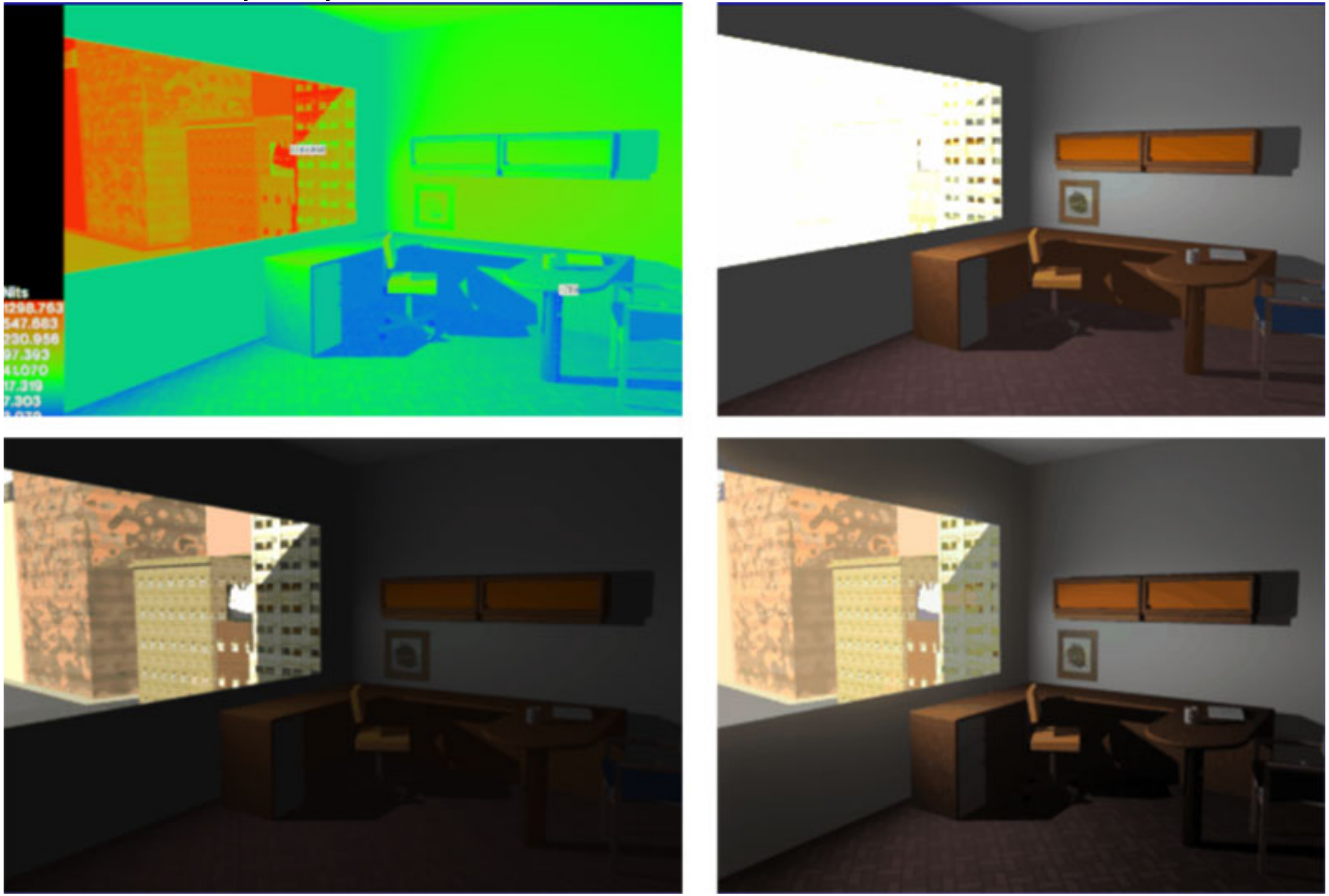
Background illumination of HDR display



- … or LDR displays; then you need:
- Tone mapping (TM) / tone reproduction = Map of the real *high dynamic range* (HDR) luminances on a *low dynamic range* (LDR) display with a limited luminance bandwidth

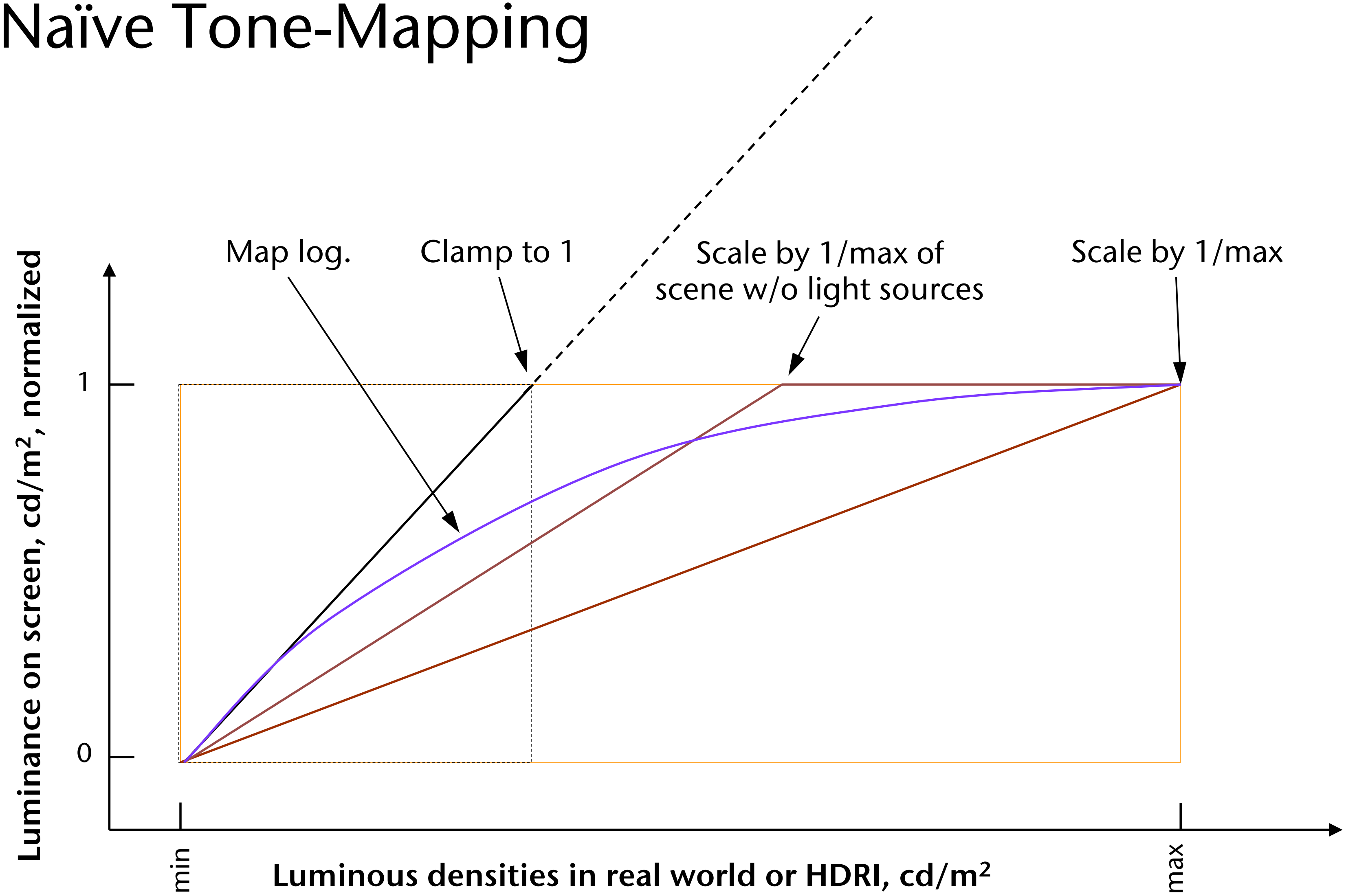
Informal Statement of the Problem

Physically correct



Best effort rendering on LDR display

Naïve Tone-Mapping



Result of the Naive Mapping

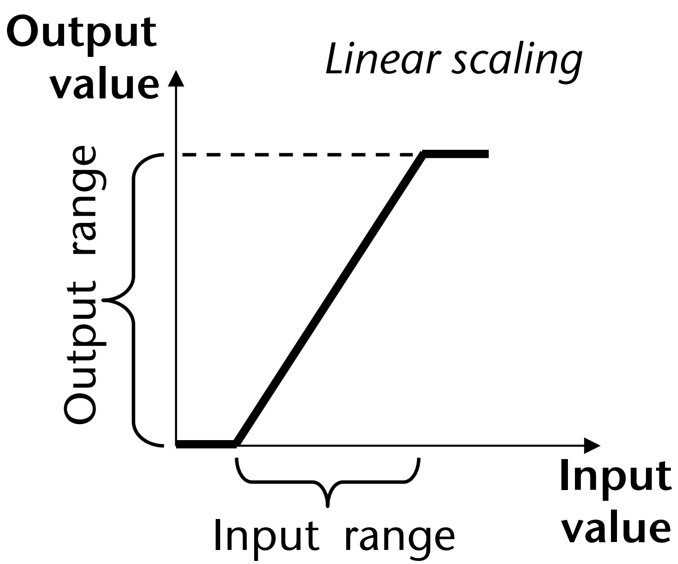

Scale by 1/max


Clamp to 1

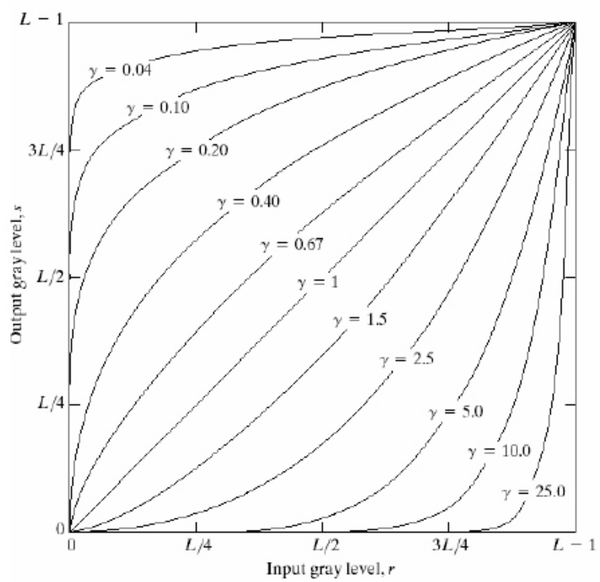

Log. mapping

An Important Class of Tone Mappings

- First consider pure "point functions":
  - Determine a transfer function $y = T(x)$
    - Also called tone mapping operator
  - T only depends on the color x of a pixel; it is completely independent of its position or the neighborhood around
- Examples:

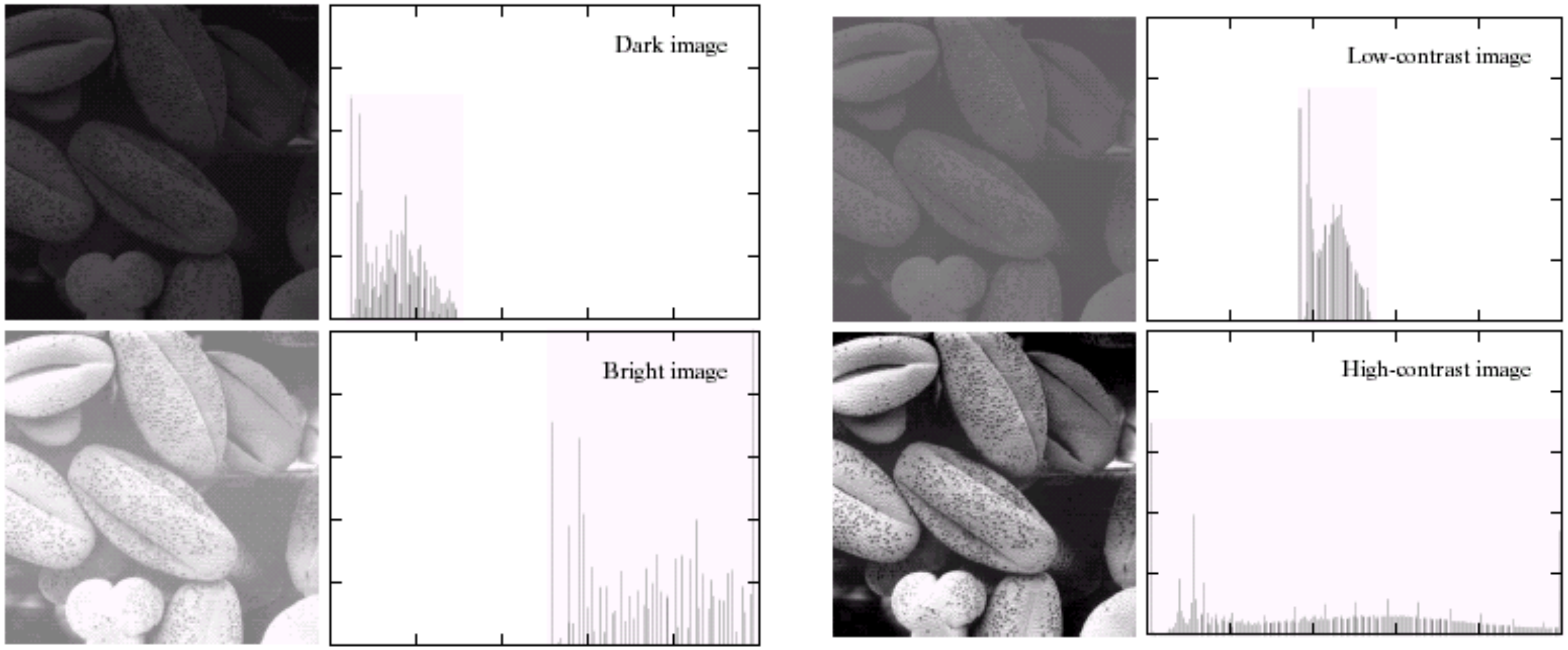


Gamma correction

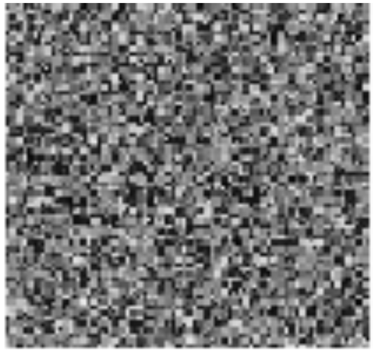


The Luminance Histogram

- Images with "unbalanced" histograms do not use the full dynamic range
- Balanced histograms result in a more pleasant image and reflects the content much better



- The histogram of an image contains valuable information about the grayscale
- It contains no spatial information
- All of the following images have exactly the same histogram!

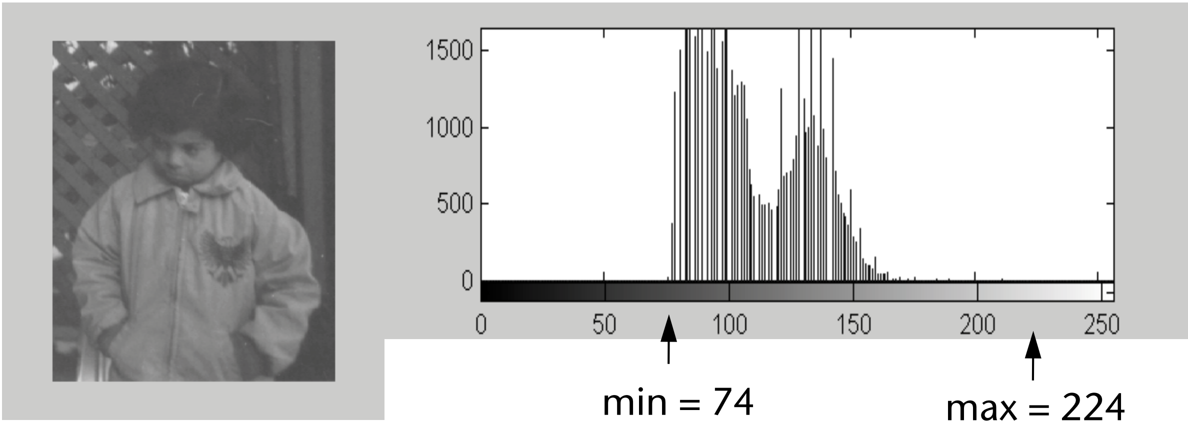
Historical Note: Histograms for Decrypting

- First presented by Abu Yusuf Ya'qub ibn Ishaq al-Sabbah Al-Kindi as a tool for deciphering a (simple) substitution cipher
  - Now called frequency analysis method
  - Breakthrough at this time, 850 n. Chr. [Simon Singh: The Code Book, 1999]

Histogram Stretching

- Linear scaling = "*histogram stretching*": $J = \frac{I - I_{\text{min}}}{I_{\text{max}} - I_{\text{min}}} \cdot J_{\text{max}}$

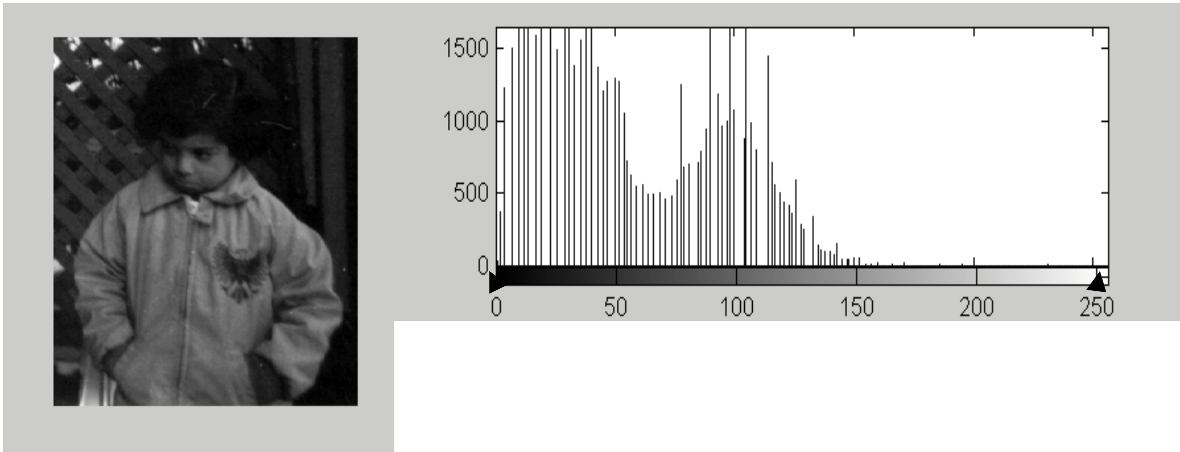
I



min = 74

max = 224

J



Interpretation of an Image Histogram

- Treat all pixels as i.i.d. random variables , i.e., each pixel = one RV
  - *i.i.d. random variables* = *independent, identically distributed RVs*
- Histogram = discrete approximation of the *probability density function (PDF)* of a pixel in the image

Discrete (Histogram) vs. Continuous Formulation (PDF/CDF)

Discrete world:

$x \in 0, \ldots, L-1$

$L = \#$ levels

Continuous world:

$x \in [0, 1]$

Histogram:

$h(x) = \#$ pixels with level x

Probability distrib. funct. (PDF):

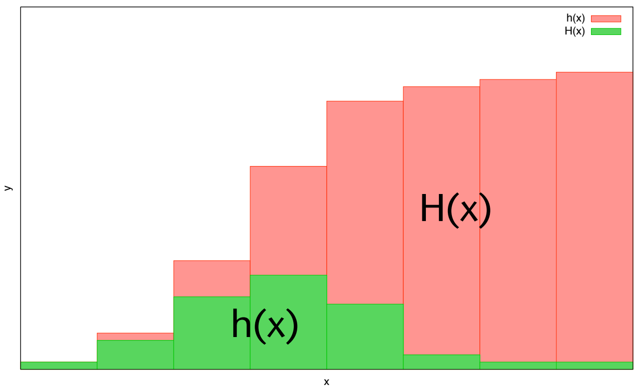
$p(x) =$ "density" at level x

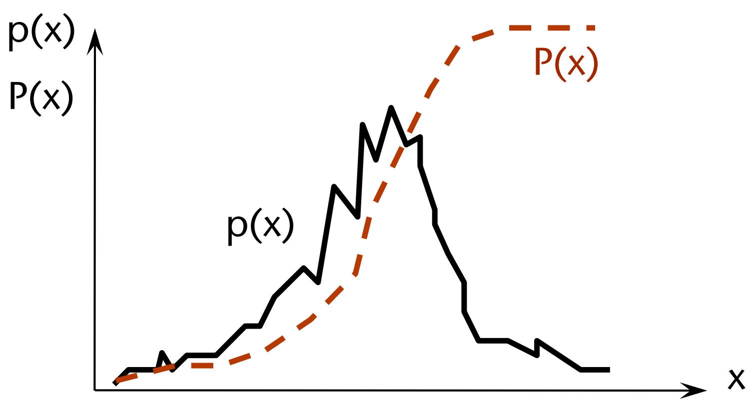
Cumulative histogram:

$$H(x) = \sum_{u=0}^{x} h(u)$$

Cumul. distrib. function (CDF):

$$P(x) = \int_0^x p(u) du$$





- Clearly:

$$H(L-1) = \sum_{u=0}^{L-1} h(u) = N = \text{number of pixels}$$

  - Therefore $h(x)$ respectively $H(x)$ is often normalized with $\frac{1}{N}$

- Let X be a random variable;
  the probability that the event "$X \leq x$" occurs is

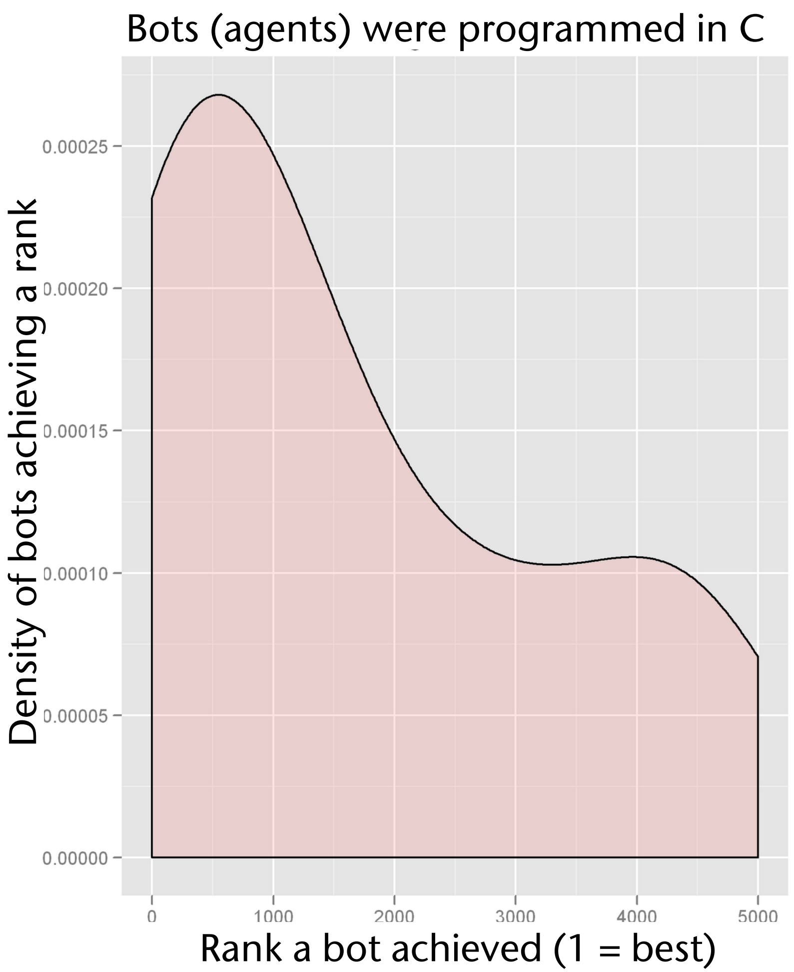
$$P[X \leq x] = P(x) = \int_0^x p(u)du$$

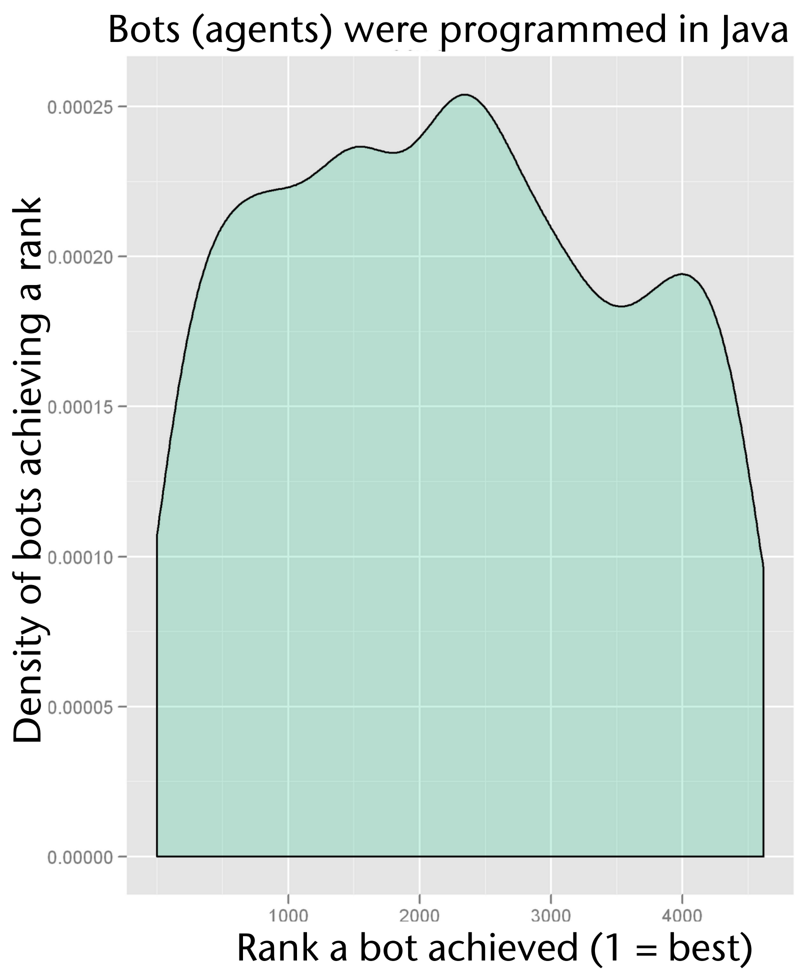
  or (in the discrete world)

$$P[X \leq x] = H(x) = \frac{1}{N} \sum_0^x h(u)$$

Example Histogram (or, rather, PDF)

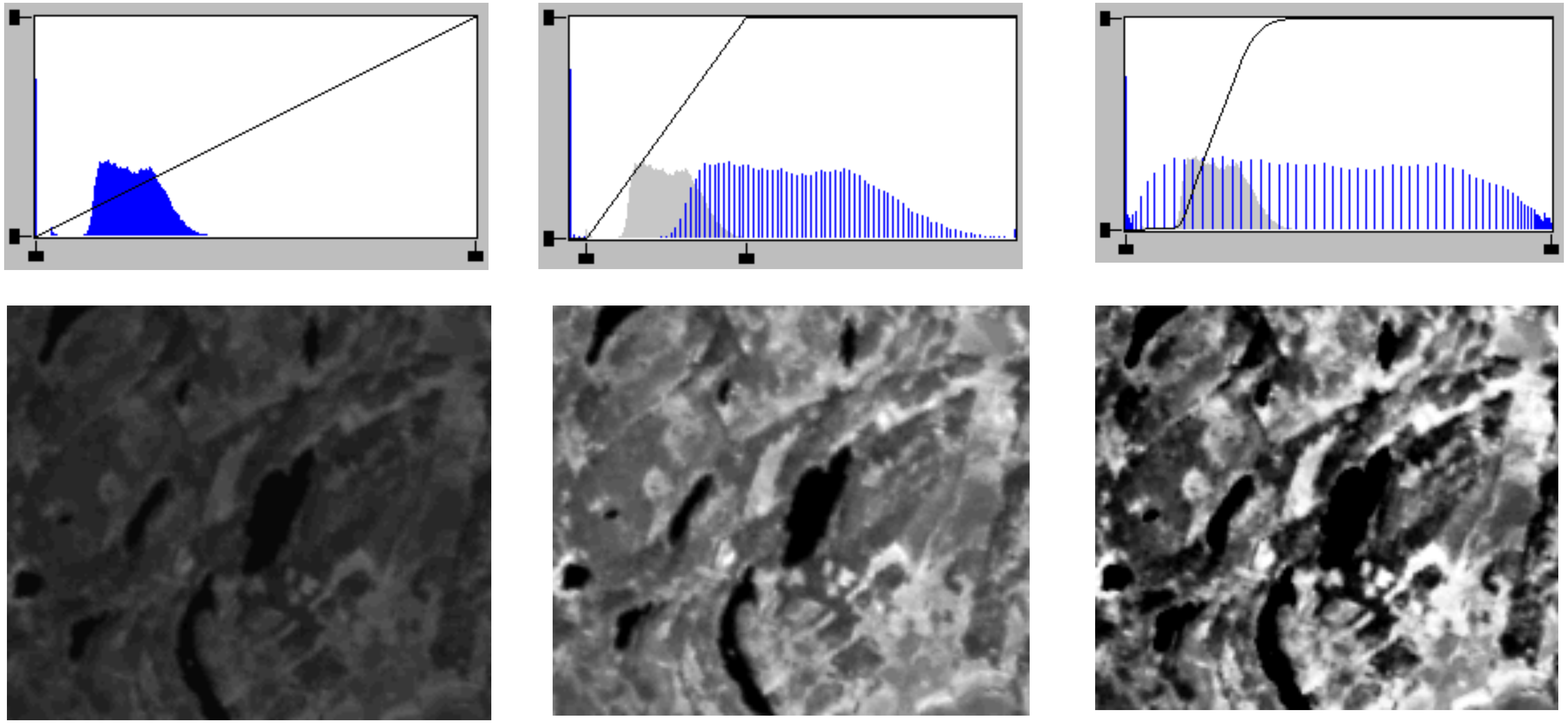
- How did *bots* (= *agents*) or, rather, programmers compare according to programming language in the Google AI challenge 2010:





Can We Do Better Than Histogram Stretching?

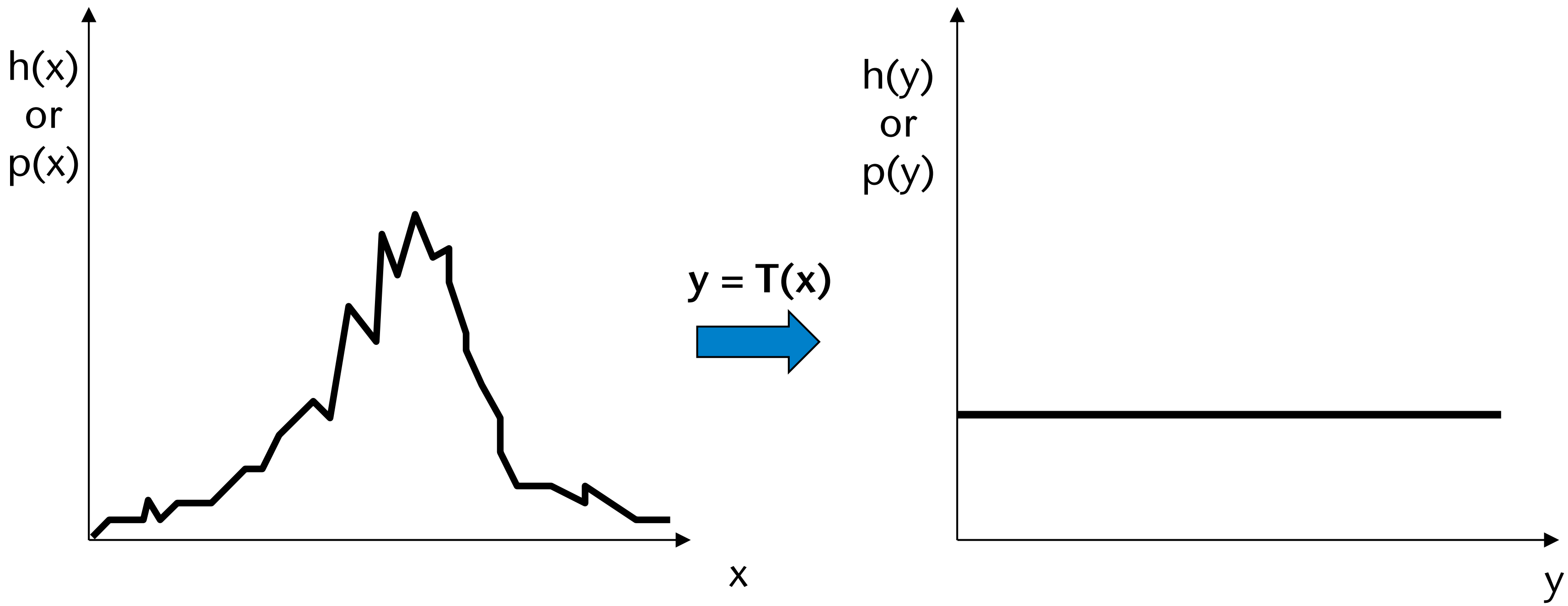
- Example with different transfer function:



- How can we find algorithmically the optimal transfer function?

Histogram Equalization

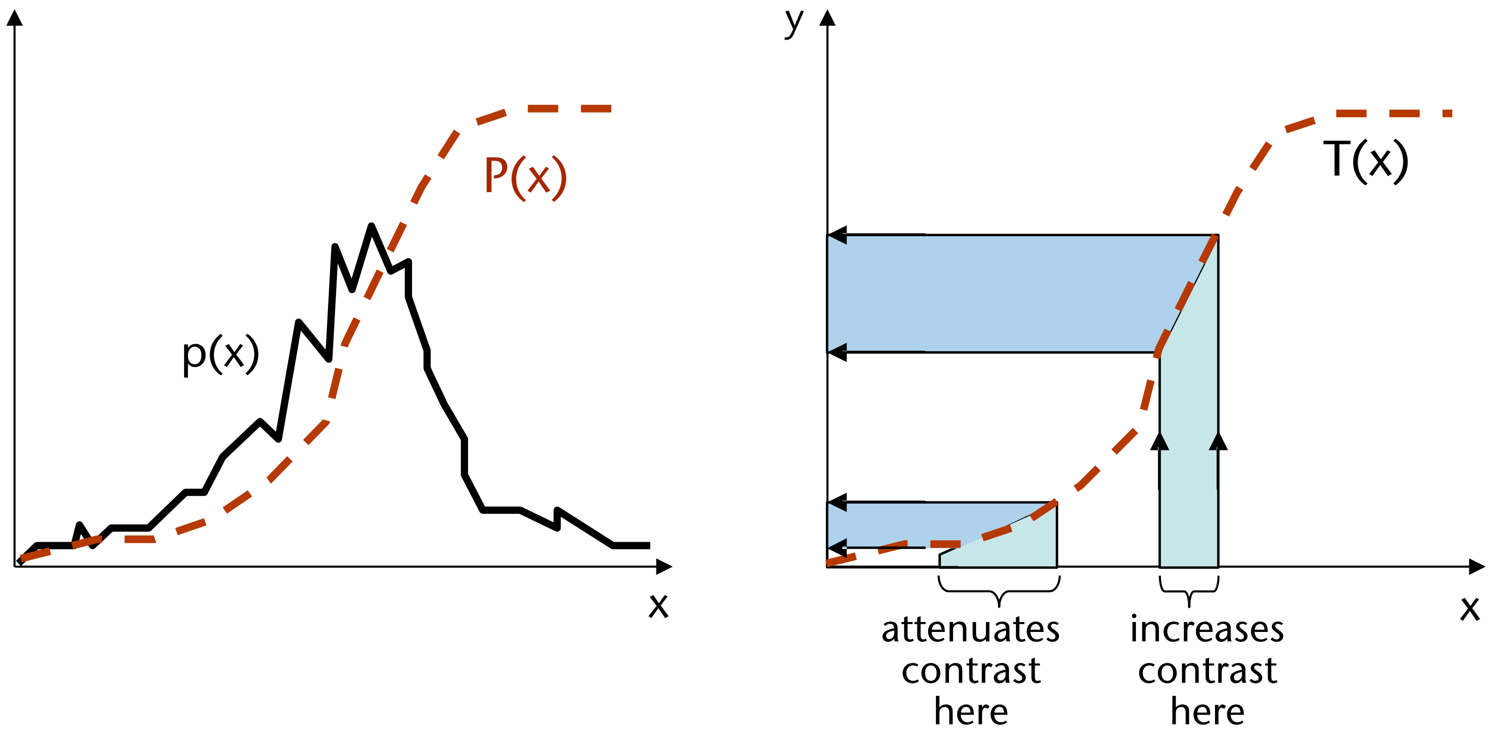
- Given: a random variable X with a certain PDF p_X
- Wanted: function T such that the random variable $Y = T(X)$ has a uniformly distributed PDF $p_Y \equiv \text{const}$
- This transformation is called histogram equalization



- Conjecture: the transfer function

$$y = P(x) = \int_0^x p(u)\,du$$

performs exactly this histogram equalization



Optionally

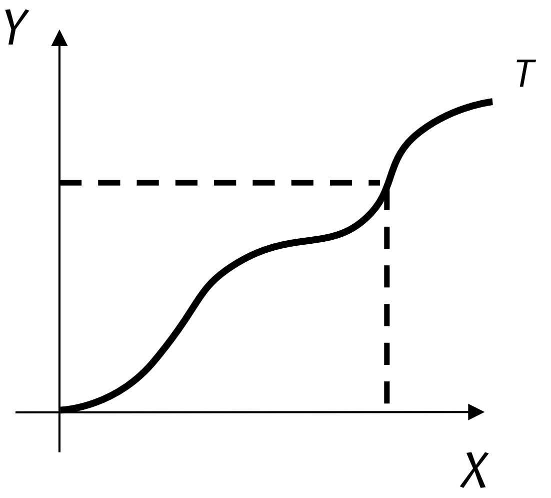
1. Version of a Proof

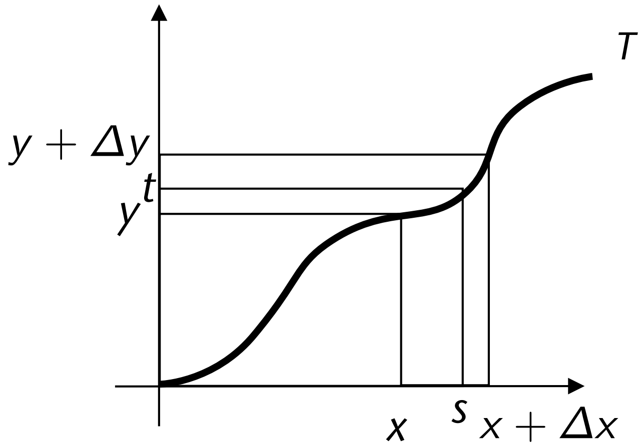
- Let X be a continuous random variable
- Let $Y = T(X)$ (so Y is a continuous RV, too)
- Let T be $\mathcal{C}^1$ and monotonically increasing
- Consequently, T' and T^{-1} do exist
- Because T maps all $x \leq s \leq x + \Delta x$ to $y \leq t \leq y + \Delta y$, we have

$$\int_x^{x+\Delta x} p_X(s) ds = \int_y^{y+\Delta y} p_Y(t) dt$$

- So, for small Δx, we have

$$p_Y(y)\Delta y \approx p_X(x)\Delta x \qquad p_Y(y) \approx p_X(x)\frac{\Delta x}{\Delta y}$$





Optionally

- When $\Delta x \to 0$, then the approximation becomes an exact equation:

$$p_Y(y) = \lim_{\Delta x \to 0} p_X(x) \frac{\Delta x}{\Delta y} = p_X(x) \lim_{\Delta x \to 0} \frac{1}{\Delta y / \Delta x}$$

$$\lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \frac{T(x + \Delta x) - T(x)}{\Delta x} = T'(x)$$

- Combined:

$$p_Y(y) = \frac{p_X(x)}{T'(x)}$$

Optionally

- Now, inserting $x = T^{-1}(y)$ results in

$$p_Y(y) = \frac{p_X(T^{-1}(y))}{T'(T^{-1}(y))}$$

- Side result: now we know how to convert distribution functions, if a random variable is a function of another random variable.
- Continue with the histogram equalization ...

Optionally

- Sought is a function T, such that

$$p_Y(y) \equiv 1$$

- Inserting our previous result yields

$$\frac{p_X(T^{-1}(y))}{T'(T^{-1}(y))} = 1$$

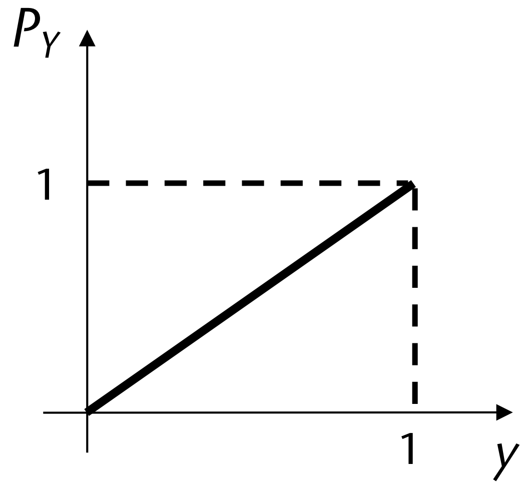
$$T'(T^{-1}(y)) = p_X(T^{-1}(y))$$

- Inserting $x = T^{-1}(y)$ results in $T'(x) = p_X(x)$
- Sought was T, so integration yields:

$$T(x) = \int_0^x T'(u)du = P_X(x)$$

2. Version of a Proof

- To prove: $P_Y(y) = y$
  - I.e., the image after the transformation by the transfer function has a flat histogram

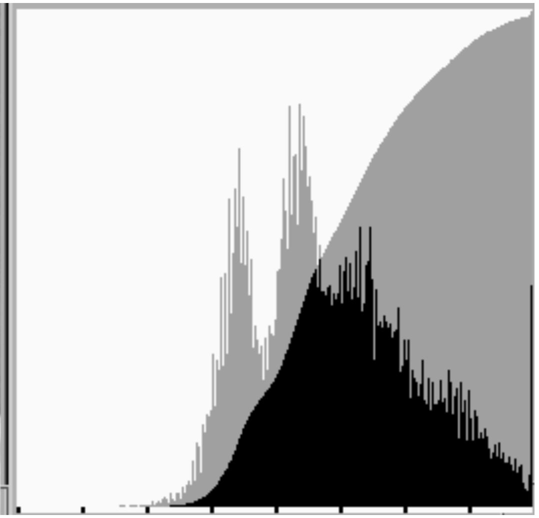


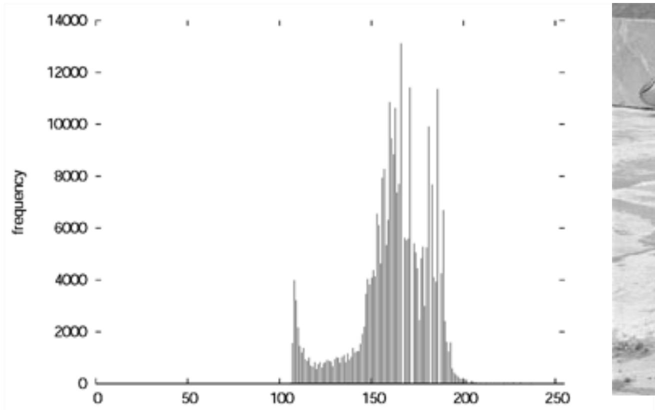
- Proof by inserting:

$$\begin{aligned} P_Y(y) &= P[Y \leq y] \\ &= P[T(X) \leq y] \\ &= P[P_X(x) \leq y] \\ &= P[x \leq P_X^{-1}(y)] \\ &= P_X(P_X^{-1}(y)) \\ &= y \end{aligned}$$

Examples

Orig. Image	Histogram	Result

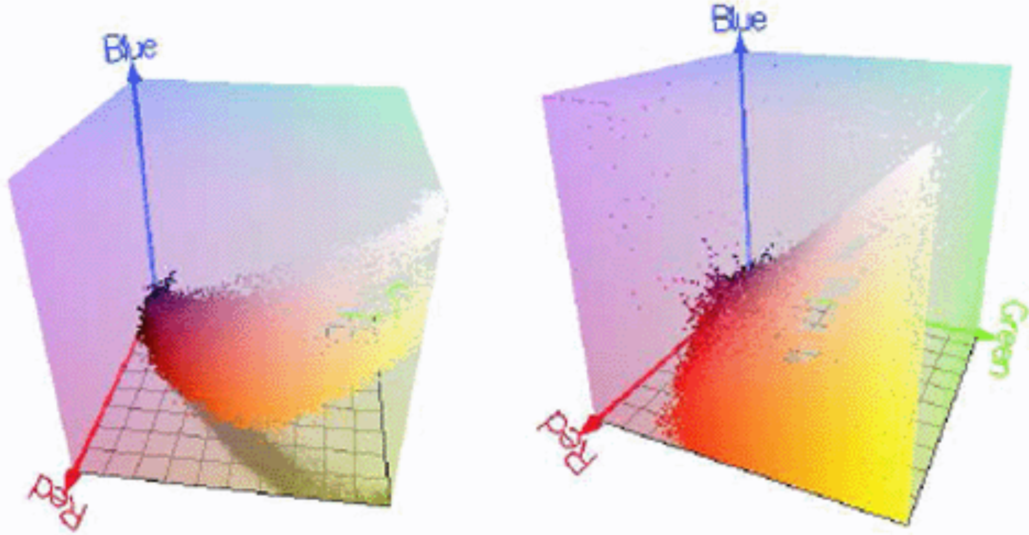




Equalization in HSV

Original Image

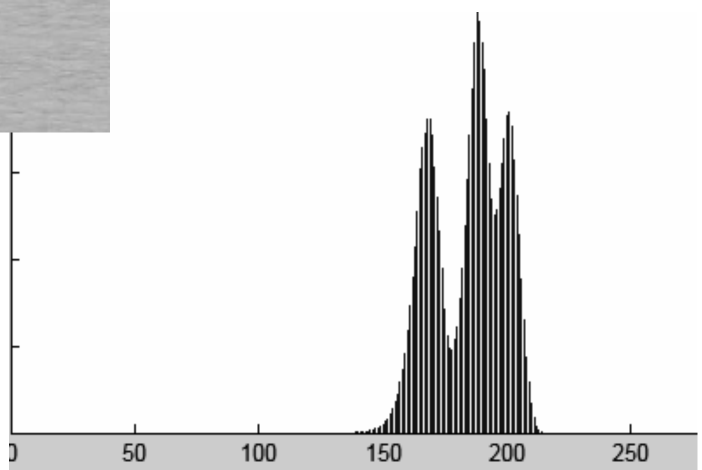
Equalized Image
a.k.a. probability smoothing)



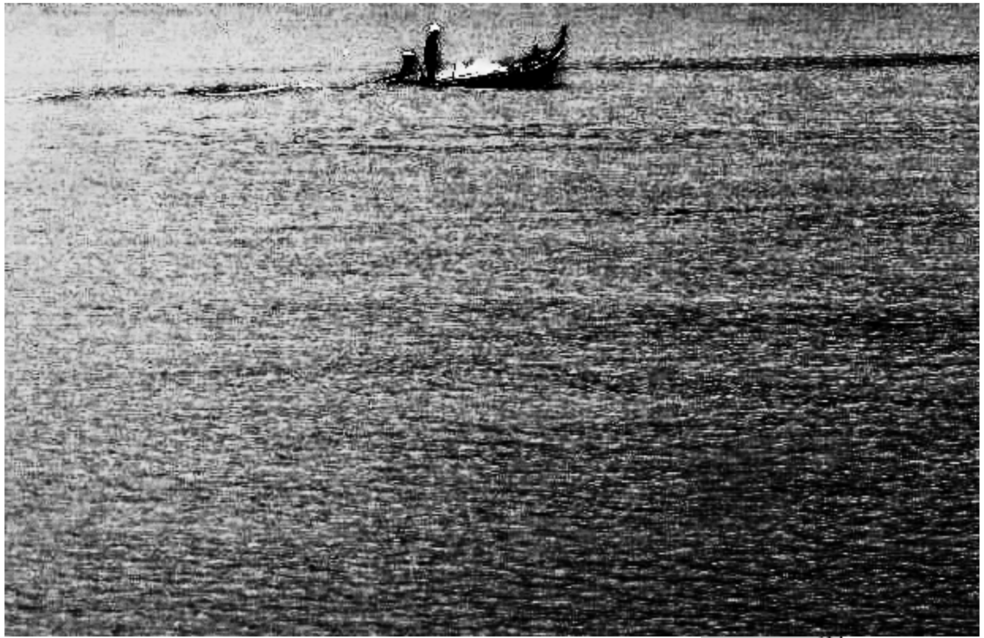
Equalization in RGB

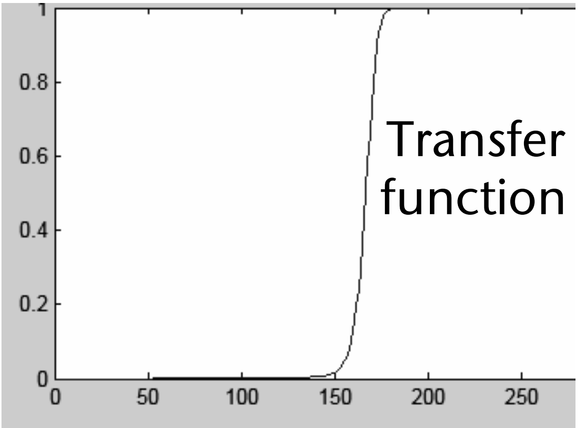
A Problem of Histogram Equalization

- Problematic case: a very narrow histogram of the input image

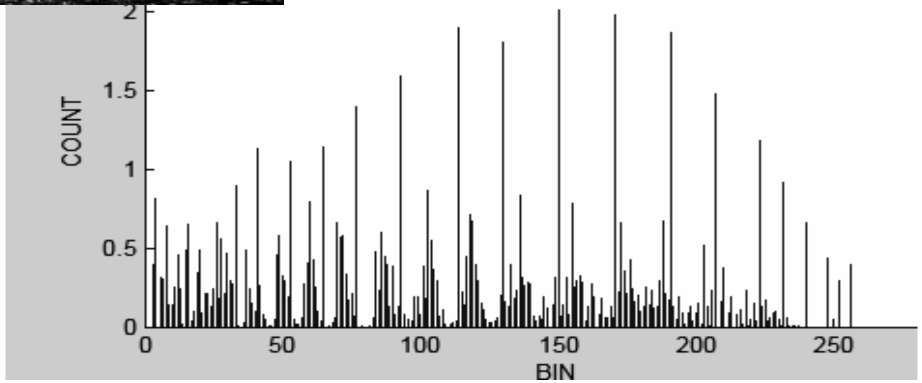


- Result: unwanted contrast



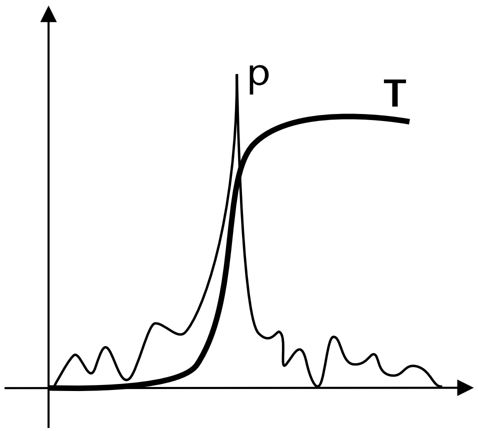


Resulting histogram



Tone Reproduction by Ward et al. [1997]

- Problem of histogram equalization:
  - Very steep sections of the transfer function T can produce visible noise
- Idea: limit the slope of T
- Algorithm:



1. Determine the histogram h
   - Reminder: $h \approx p = T'$
2. Clamp too large bins to a value $\alpha \cdot \frac{N}{B}$, where $\alpha \approx 0.5 \ldots 1.5$,
   N = number of pixels, B = number of bins
3. Let $N' = \sum_{i=0}^{L-1} h(x_i)$
4. Use this to perform equalization and repeat a few times

Excursion: The Weber-Fechner Law [~1850]

- By experiment, we find:
  - The just noticeable difference (JND) of a stimulus (e.g., weight) depends on the *level* of the stimulus (differential threshold of noticeability)
  - The ratio of the JND over the level of the stimulus is constant (depending on the kind of stimulus)
- The mathematical formulation of these findings:
  - Let S be the level of the stimulus, and let ΔS be the JND at this level
  - Now, Weber's law says:

$$\frac{\Delta S}{S} = \text{const}$$

- The Weber-Fechner law:
  Let E be the level of the perceived sensation of S (e.g., perceived weight), and let ΔE be the JND of E.
  Then we have

$$\Delta E = k \frac{\Delta S}{S} \quad \Rightarrow \quad dE = k \frac{1}{S} dS$$

- Integration results in:

$$E = k \cdot \ln S + c$$

  - Here, c is a constant that describes the minimum stimulus S_0, with which just a sensation $E \approx 0$ is created (threshold stimulus):

$$c = -k \cdot \ln S_0$$

- Combined:

$$E = k \cdot \ln \frac{S}{S_0}$$

- Example application: decibel as a unit of measurement for the *perceived* loudness of a sound

Excursion²: The Stevens Power Function

- Another plausible assumption seems (IMHO) the following:

$$\frac{\Delta E}{E} = k \frac{\Delta S}{S}$$

- Transformation results in:

$$\frac{1}{E}\Delta E - k\frac{1}{S}\Delta S = 0 \quad \Rightarrow$$

$$\frac{1}{E}\mathrm{d}E - k\frac{1}{S}\mathrm{d}S = 0 \quad \Rightarrow \quad \ln E - k \ln S = c \quad \Rightarrow$$

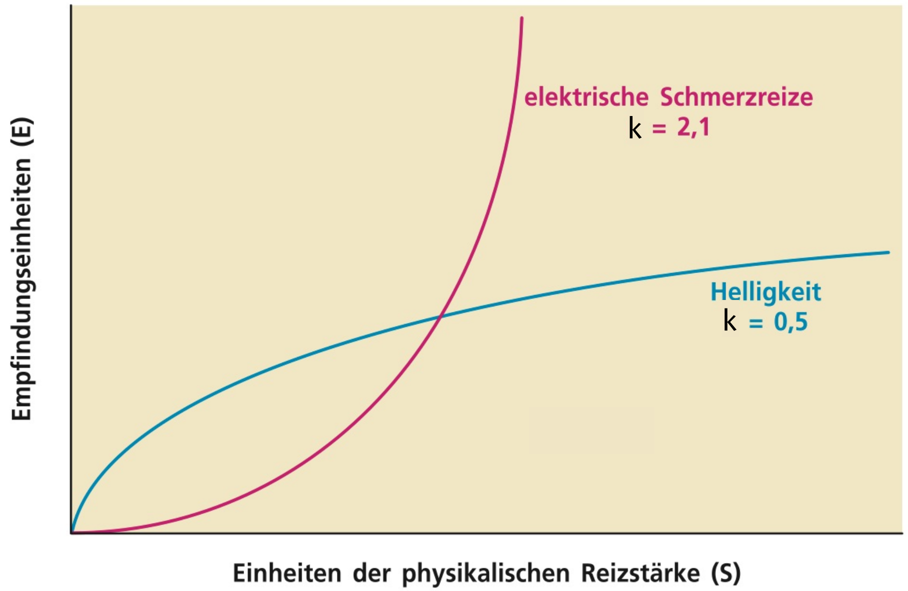
$$\ln\frac{E}{S^k} = c \quad \Rightarrow \quad \frac{E}{S^k} = e^c = c' \quad \Rightarrow$$

- Finally results in Stevens' power law:

$$E = cS^k$$

where E = sensation strength ("perceived weight"), S = stimulus (a physical value), c and k = constants, which depend on the sense organ

- For many stimuli, $k < 1$ (for brightness k ≈ 0.5, for sound volume k ≈ 0.6)
- For some stimuli, $k > 1$ (for temperature k ≈ 1-1.6, for electric shock k ≈ 2-3)



Notes on the Laws

- The Weber-Fechner law describes (apparently) better the perception of stimuli in the middle range, the Stevens power law better in the lower and upper range
- Research on the two laws is still in full swing
- There are early indications that neural networks and cellular automata also show this behavior, if sensory perception (excitation + transport) is simulated with them!

- In the case of the visual sense, ΔE can be specified in more detail:

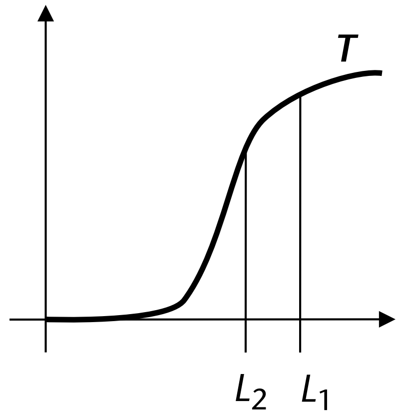
$$\Delta E = \begin{cases} -2.8 & , \log L < -3.9 \\ (0.4 \log L + 1.6)^{2.2} - 2.8 & , -3.9 \leq \log L < -1.4 \\ \log L - 0.4 & , -1.4 \leq \log L < -0.02 \\ (0.3 \log L + 0.7)^{2.7} - 0.7 & , -0.02 \leq \log L < 1.9 \\ \log L - 1.3 & , \log L \geq 1.9 \end{cases}$$

Perceptually-Based Tone Mapping

- Assume two adjacent pixels in the original image have just a difference in intensity of the JND, i.e.

$$\Delta L = L_1 - L_2 = J(L_1)$$

(w.l.o.g. $L_1 > L_2$)



- Wanted is a transfer function T such that this condition is an invariant, i.e.

$$T(L_1) - T(L_2) \leq J(T(L_1))$$

- Transformation:

$$p(L_1) = T'(L_1) \approx \frac{T(L_1) - T(L_2)}{L_1 - L_2} \leq \frac{J(T(L_1))}{L_1 - L_2} = \frac{J(T(L_1))}{J(L_1)}$$

- Algorithm:
  1. Compute the histogram h
  2. Calculate the cumulative histogram $\longrightarrow$ transfer function T
  3. Clamp all bins of the original h, such that

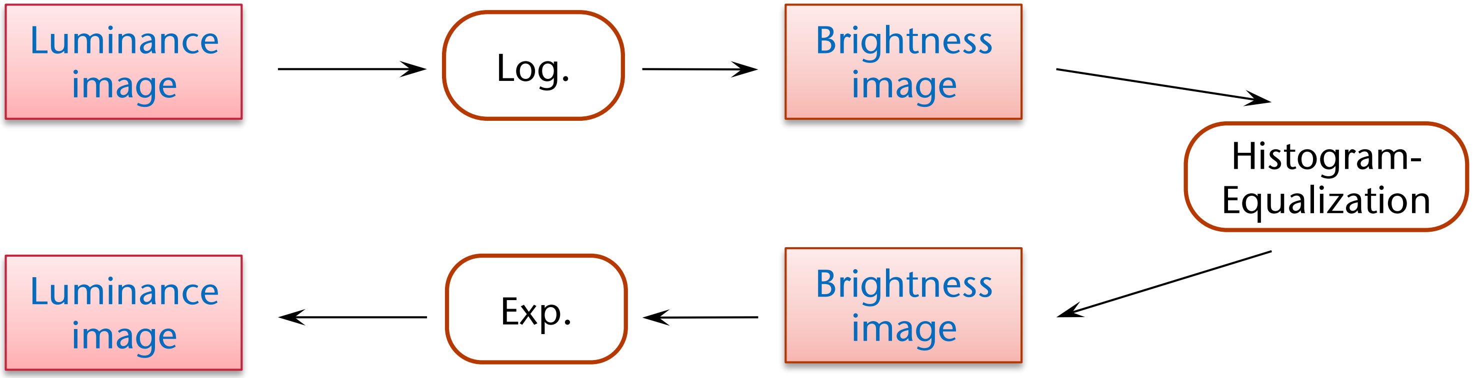
$$h(i) \leq \frac{J(T(L_i))}{J(L_i)}$$

     where L_i is the intensity level of bin i

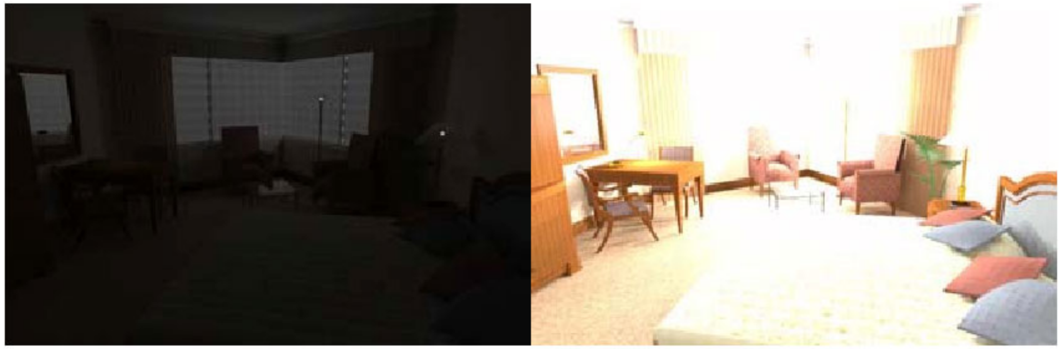
  4. Compute a new cumulative histogram $\longrightarrow$ new transfer function T
  5. Repeat a few times

Example

- Side note: The Weber-Fechner law is also the reason for performing the histogram equalization or tone mapping very often in so-called "log-space"

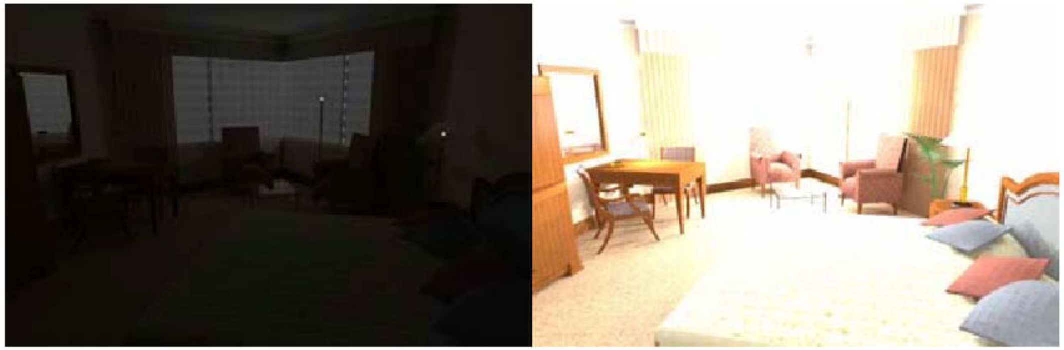


Other Tone Mapping Operators



Left/right images
show dynamic range

Result by
Shilick's operator



Left/right images
show dynamic range

Result by
Reinhard's operator

Further Ideas

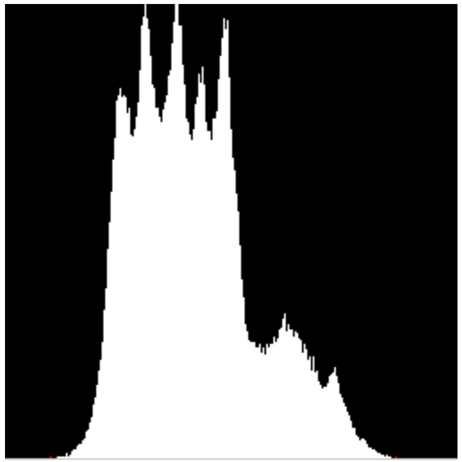
- Problem: This method prevents $\Delta L > J(L)$ also between pixels, which are not adjacent
  - Idea: map each pixel taking into account *only* the neighboring pixels
    → Real local Tone-Mapping-Operator (local TMO)
  - Unfortunately leading again to other problems (i.e. "halos")
- Further limitations of the human visual systems:
  - Glare (Blendung): strong light sources in the peripheral vision reduce contrast sensitivity of the eye
  - Scotopic / mesopic vision: at low luminance, the color sensitivity decreases sharply
  - Similarly, spatial resolution decreases

➢ Could take advantage of all that in the TMO

Generating a Histogram on the GPU

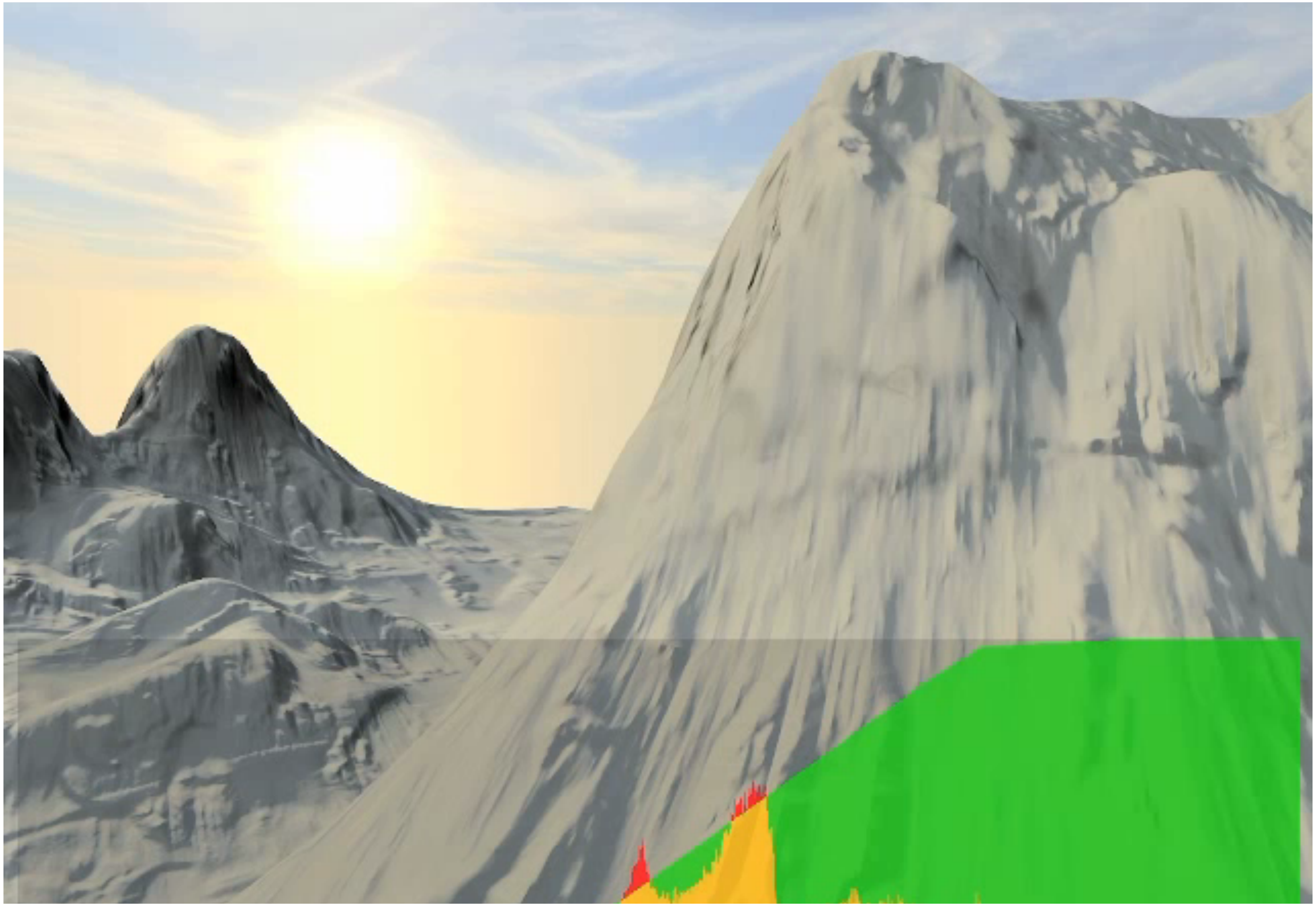
- Given: gray-scale image (= texture)
- Goal: histogram as 1D texture
  - Each texel = one bin
- Problem: "distribution" of pixels into the bins
  - Destination output address of a fragment shader is fixed
- First idea:
  - For each pixel in the original image, render one point (GL_POINT)
  - In the vertex shader, calculate the corresponding bin (instead of a transformation with MVP matrix)
  - Pass the "coordinate" of this bin as the coordinate of the point to the fragment shader
- Problem:
  - High data transfer volume CPU → GPU
  - Example: 1024^2x2x4 Bytes = 8 MB in addition to 1024^2-image

Generation of Histograms Using the Geometry Shader



- Render a quad in the application
- Vertex shader is just a pass-through
- The geometry shader …
  - makes one loop over the image,
  - emits for each pixel a point primitive with x coordinate = brightness of pixel = bin , y=0
- The fragment shader …
  - takes the points,
  - outputs color (1,0,0,0),
  - at position (x,0)
- The pixel operation …
  - is set to blending with `glBlendFunc(GL_ONE,GL_ONE)` = accumulation (current cards can do that also with FP-FBOs)

Video



Thorsten Scheuermann, Justin Hensley; 2007.
Graphics Product Group, Advanced Micro Devices Inc.

Optionally

Alternative: Use CUDA on the GPU

- Reminder for those of you who have attended my Massively Parallel Algorithms class:
  - Use CUDA's Graphics Interoperability to access image in CUDA
  - Compute the histogram using a massively parallel algorithm
  - Do a *parallel prefix sum* on the histogram
  - Switch back to OpenGL and transform the image using a fragment shader (or do it in CUDA, too)
- For those of you who have *not* attended my Massively Parallel Algorithms class:
  - This might be an incentive to do so ☺

High-Dynamic Range Imaging in Photography

- Were actually doing it before computer graphics did it [Charles Wyckoff, 1930-40]
- Meanwhile, HDRI is well integrated in Photoshop & Co.

Original

Tone mapped

Examples