

Images, C++

Frédo Durand
MIT EECS
6.8370/1

C++ and Image Processing 101

Fredo Durand
MIT EECS
6.8370/6.8371

Administrivia

- Pset 0 is out and due Wednesday (in 1 days)
 - But weight is 0 for final grade
- Accounts to pset system have been sent
 - Contact us if you don't have one.
- Make sure you
 - join piazza
 - have an account on the magical system
- When you switch (grad <-> undergrad) send a piazza message to the staff.

Submission & Grading system

Will not tell you if your code is correct,
will not show you the correct answer

1/ the requires a human

2/ we tried in the past, it has led to unhealthy strategies

The AI assistant

My apologies to the early Guinea pigs.
Slowly getting better.
Share your complaints, requests and ideas.

How the AI assistant works

- Preamble prompt, cached and shared by every reader
 - Tutor rules (explain, do not write the pset code), the book's contents, which page you are on
- Pre-filtering, before the model sees anything
 - Unreleased problem sets and decks never reach search; solutions are never indexed
 - Results near you (this chapter, this deck) are promoted, never the only ones
- RAG: one vector index, about 34,000 records
 - Book, slides, notes, psets, lecture transcripts, outside references
 - Vector search fused with keyword search; up to seven tool calls per answer
- Post-filtering, after the answer is written
 - Every link is checked against real anchors; slide and outside links need a minimum similarity
- Navigation
 - The model emits a @@NAV line; it moves you inside the view you are in and links out of it



Debugging

Debugging

- Doubt everything
- Debug pieces in isolation
 - Binary search/divide and conquer
- Display/print everything - save images
 - In particular intermediate results
- Create simple inputs
 - where you can easily manually compute the result
 - e.g. constant image, edge image, rectangle, etc.
 - use small images (e.g. 3x3)
 - including (especially) inputs to intermediate stages
 - use input that isolates different failure modes
- Change one thing at a time
 - e.g. to verify that a given command has the effect you want, modify it to break it

Debugging — seg fault

- Often occurs much later than the actual problem
- Probably due to some array (vector) operation
 - print indices and array size
 - Assert that index is between 0 and size
 - possibly in some debug mode to avoid performance penalty in release mode
- Could also be due to initialized pointer, but we'll avoid pointers

Basic image processing



Domain, range

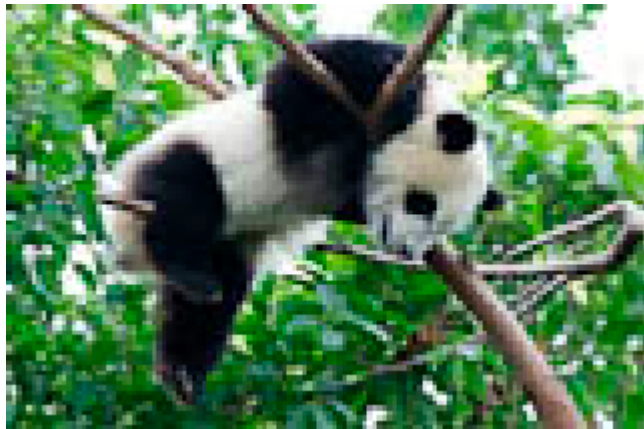
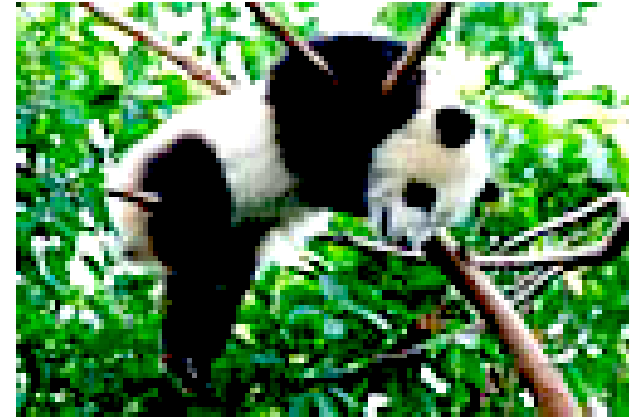


Domain vs. range

- 2D plane: domain of images
 - x, y Cartesian coordinates
- color value: range (R3 for us)
 - stores the red, green and blue components in $\text{im}(x, y, 0)$, $\text{im}(x, y, 1)$, $\text{im}(x, y, 2)$, respectively

Basic types of operations

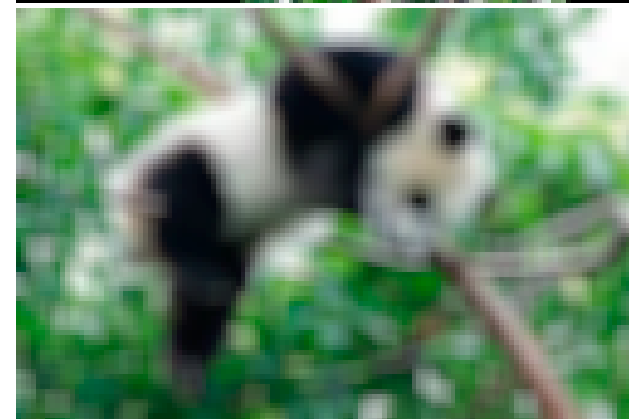
Point operations:
range only
Pset 1



Domain
operations
Pset 3 or 4



Neighborhood
operations:
domain and range
Pset 2



Images in C++

Digital Images

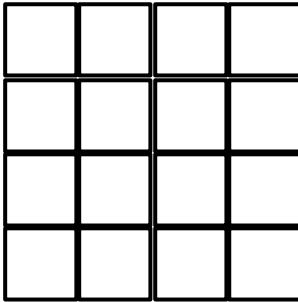
- Can be represented as 2D arrays of RGB triplets
- **Formally:**
 - The domain is the 2D plane
 - The range is the RGB space
- **Other color spaces are possible and will be used**
 - The numbers on their own are meaningless if you don't know the color space or encoding (more soon)
- **Values often encoded as 8 or 16 bit integers**
 - But we will use floats in $[0...1]$ to make life easier

Arrays

- **C++ vector**
 - dynamically sized
 - templated by type, float in our case
- **Created with**
- `data = std::vector<float>(size, initialValue);`
 - for a float array with an initial value
- **References:**
 - http://www.codeguru.com/cpp/cpp/cpp_mfc/stl/article.php/c4027/C-Tutorial-A-Beginners-Guide-to-stdvector-Part-1.htm
 - <http://www.cplusplus.com/reference/vector/vector/>

1D to 3D

- vectors only have one 1D index



1D to 3D

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

2D array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

1D vector encoding

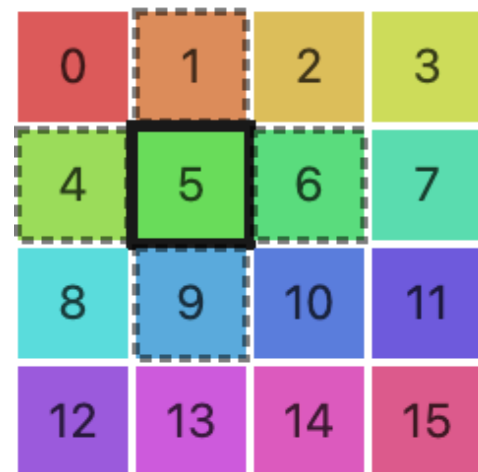
Our Image class

- **Stores in planes of color channels**
 - size width*height*channels
 - float
- **number of dimensions**
 - could be used to represent 2D images with a single channel, or even 1D arrays
- **size of various dimensions**
- **name string**

Our Image class

- Access data using `im(x, y, c)`
 - weird for C++, but more coherent with Halide

a 4x4 RGB image — click a pixel



selected: pixel 5

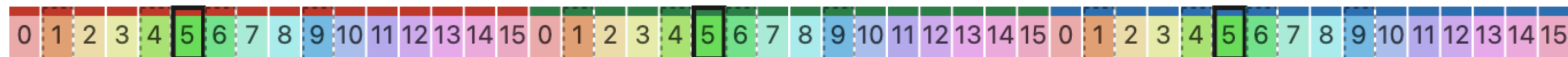
Its 3 cells are outlined bold below;
its 4 neighbours (dashed) show how
near or far they land in each layout.

Cell tint = pixel; top bar = R/G/B channel.

interleaved (R G B per pixel, row by row — NumPy HWC)



a pixel's channels sit together, but a single channel is strided — a gather that fights the vector unit
planar (all R, then all G, then all B — CHW, this book's C++ and PyTorch)



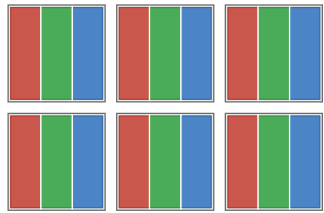
each channel is one contiguous run (ideal for SIMD); a pixel's three channels are far apart
tiled (2x2 blocks stored one after another)



2-D neighbours stay close in memory — cache-friendly for blurs and other local operations

Image memory layout: the same pixels, two packings

a 2×3 RGB image



interleaved (HWC) — NumPy default

R 00	G 00	B 00	R 01	G 01	B 01	R 02	G 02	B 02	R 10	G 10	B 10	R 11	G 11	B 11	R 12	G 12	B 12
---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

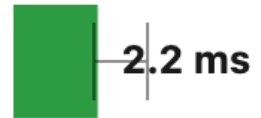
$\text{idx} = (y \cdot W + x) \cdot C + c$
planar (CHW) — ML tensors

R 00	R 01	R 02	R 10	R 11	R 12	G 00	G 01	G 02	G 10	G 11	G 12	B 00	B 01	B 02	B 10	B 11	B 12
---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

$\text{idx} = c \cdot H \cdot W + y \cdot W + x$

Brighten a 2048x2048 image (4 MP) — median of 5 runs

row-major (in order)
for y { for x }



column-major (strided)
for x { for y }

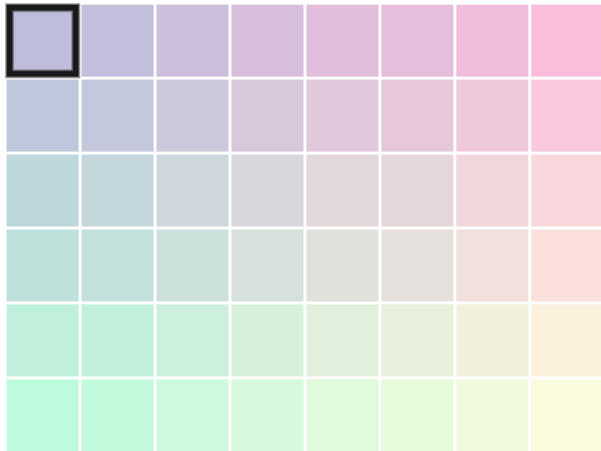


column / row elapsed time: 9.0x

same arithmetic, 4 million pixels, different access order.

Whiskers: minimum to maximum elapsed time.

the image (row-major in memory)



cache: the 4 most-recently-used lines



step 0 / 48

hits 0

misses 0

hit rate 0%

main memory: one contiguous run, grouped into cache lines of 4



File formats

- **We'll use PNG**
 - simpler to read (no need for libjpeg)
 - can be converted to and from with online tools
 - unfortunately a little slow to read
 - unfortunately a little big
- **We'll talk about how JPEG works later**

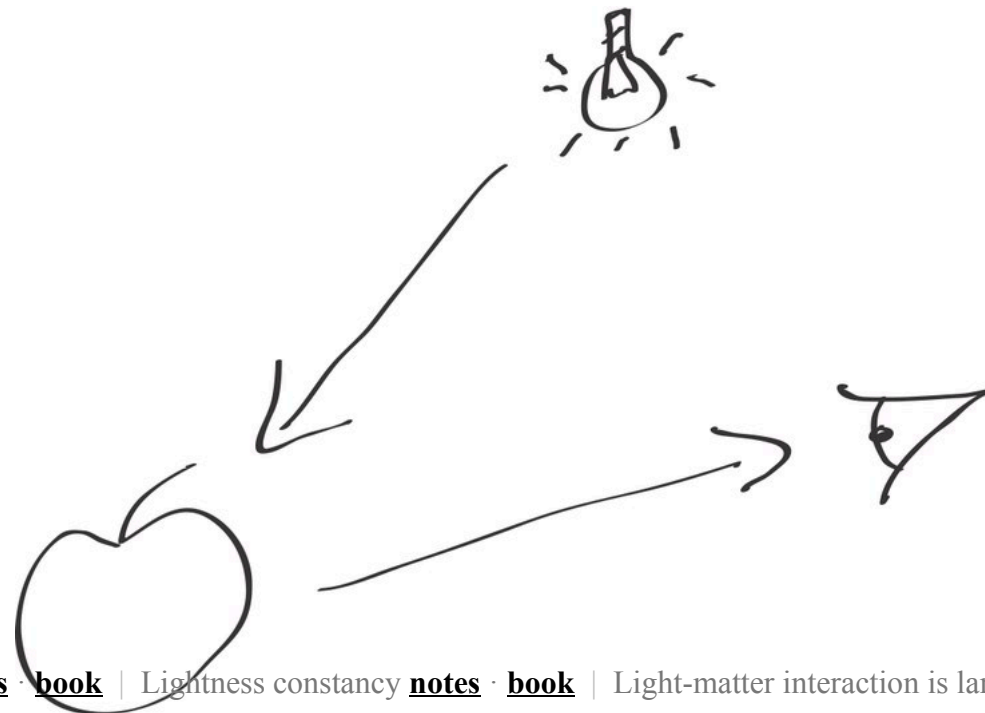
Light & perception

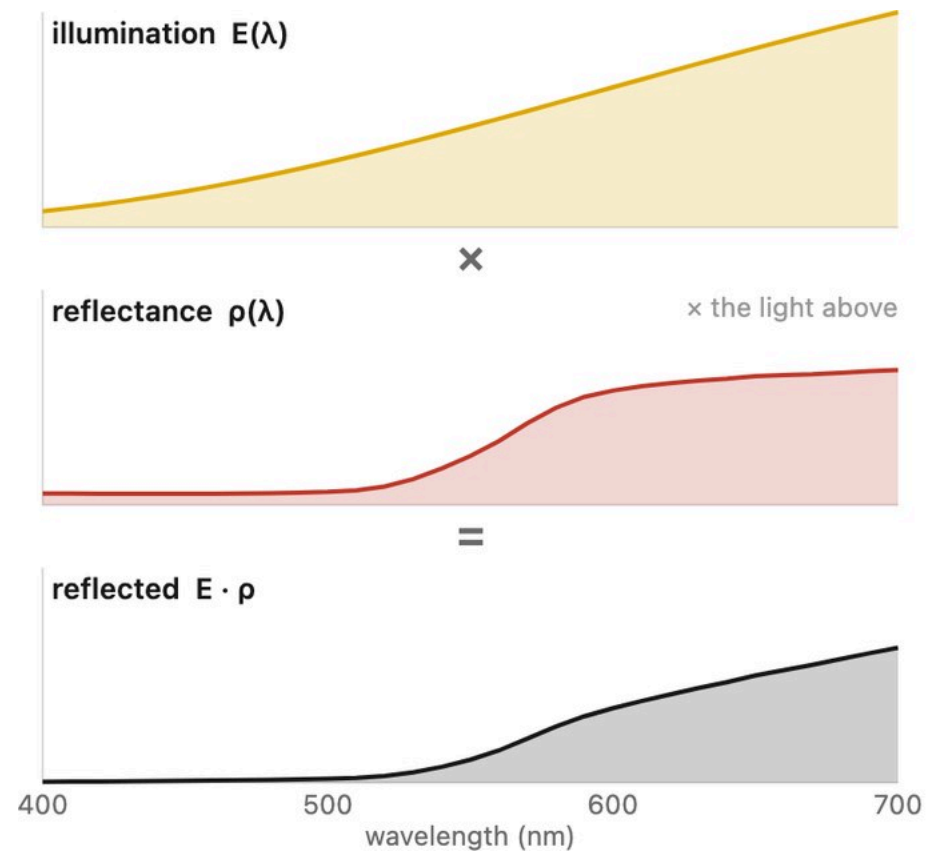
Motivation

We need to understand light and human visual perception in order to encode and process images properly

Light matter, eyes

- Light from sources is reflected by objects and reaches the eye.
- The amount of light from the source gets multiplied by the object reflectance
 - on a per-wavelength basis





colour seen



"orange" under A



same surface under
D65 daylight (reference)

*the surface is unchanged —
only the light changed the signal,
so white balance must undo it.*

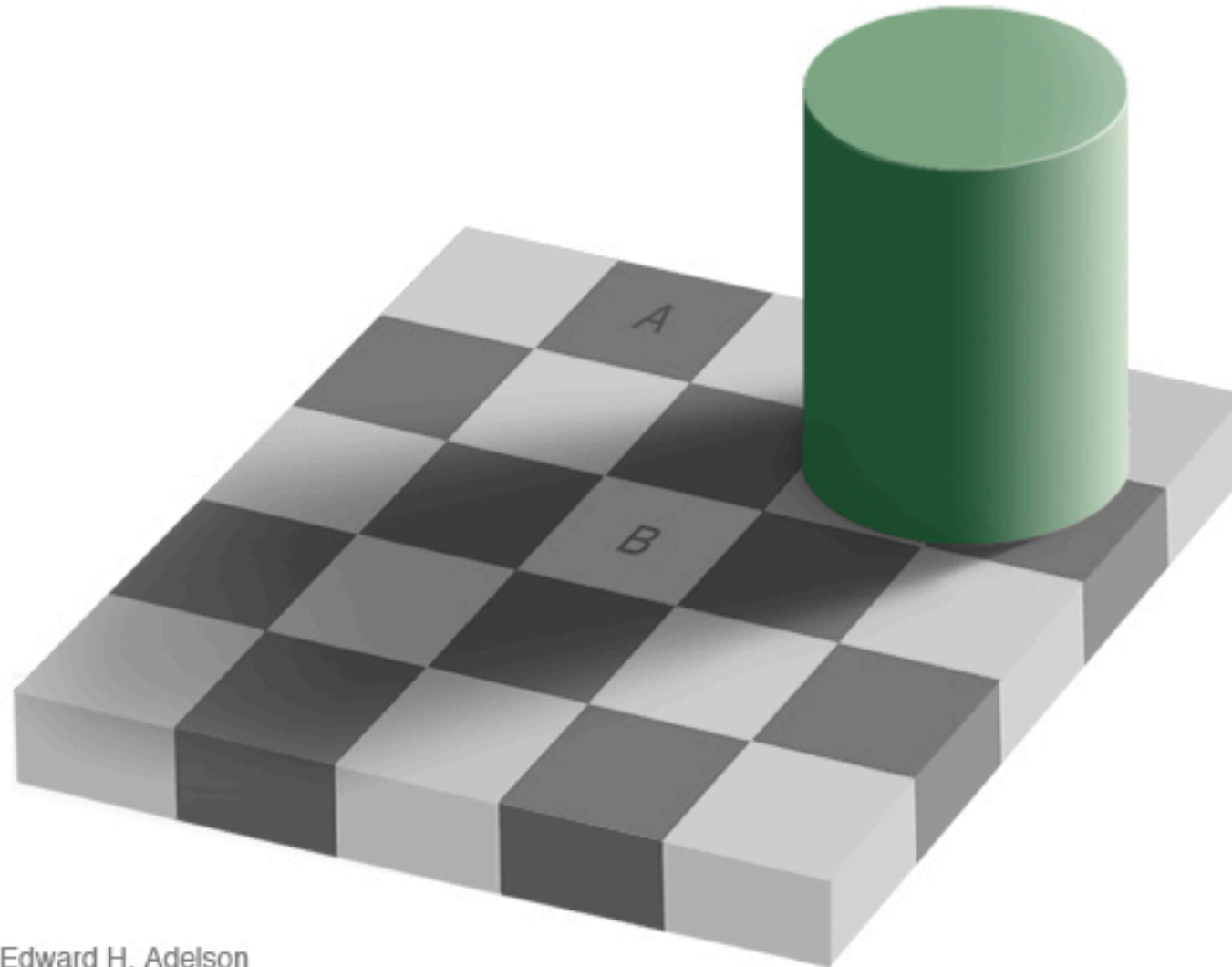
Human perception

- Our eyes have an uncanny ability to discount the illumination
 - Only objects really matter for survival
 - Light is only useful to understand if you're a photographer or to choose your sun lotion

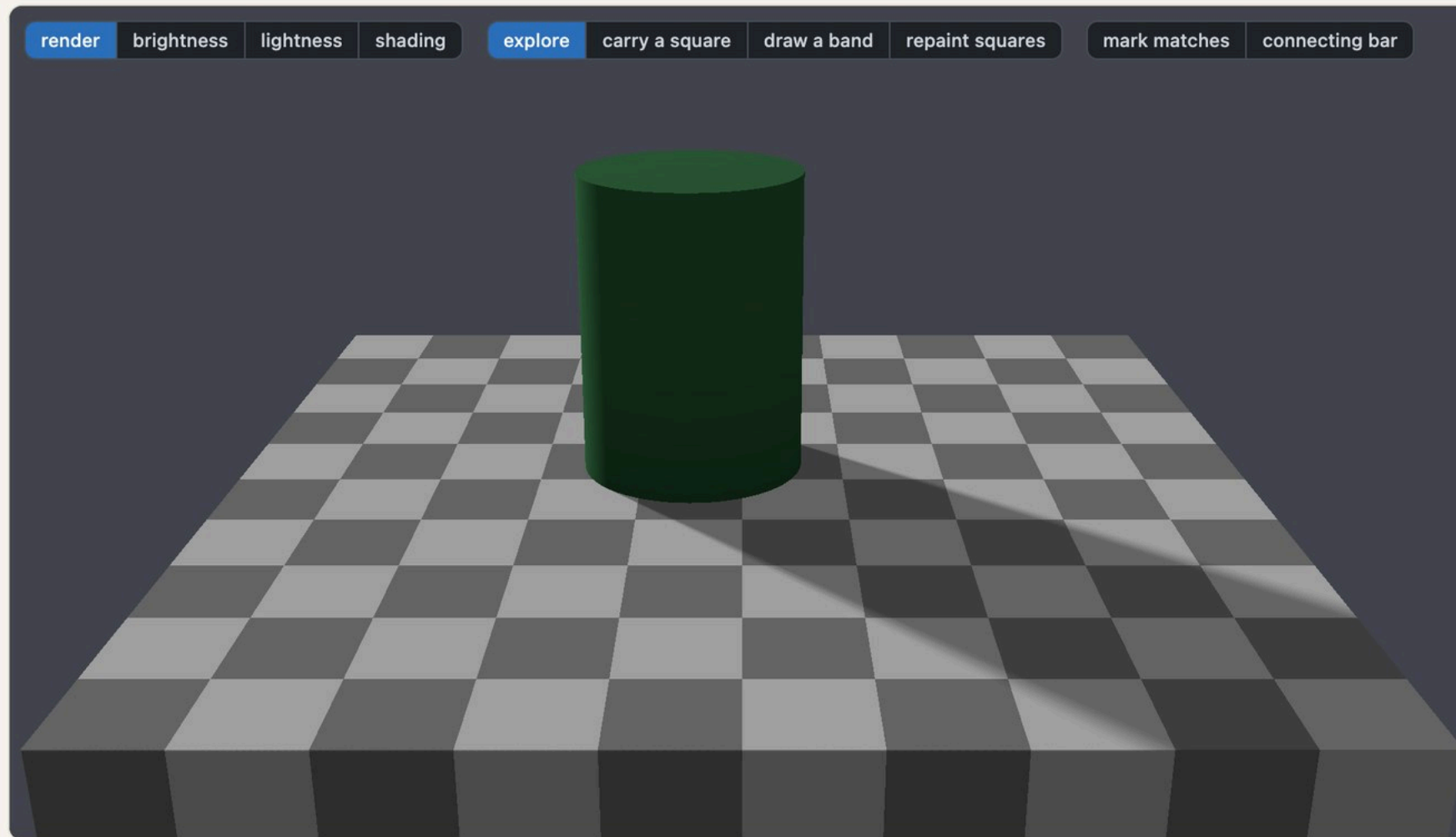


Illusion by Adelson

- A & B have exactly the same tone



Edward H. Adelson



68 cells fully lit · 5 entirely in umbra · 21 crossed by the penumbra (visibility spans 0.000–1.000)

Equal radiance 0.1315: a dark square in full light and a light square in shadow — 40 cells in all.

CHECKERBOARD

cells 10×10
light squares 0.75
dark squares 0.28

THE OBJECT

shape
diameter 2.20
height 3.00
colour
ambient 0.16

THE LIGHT

azimuth 45°
elevation 28°
distance 20
diameter 0.90

THE ILLUSION

delta from equal +0%

match Adelson's figure

copy a link to your version

Mechanism to discount light

- Light adaptation
 - We recenter our neural response around the current average brightness
 - neural + chemical + pupil
- Chromatic adaptation
 - eliminate color cast due to light sources
 - e.g. Daylight is white but tungsten is yellowish
 - Related to white balance - more soon
 - and Spanish Castle illusion

Big lesson: Human vision is
largely invariant to multiplication

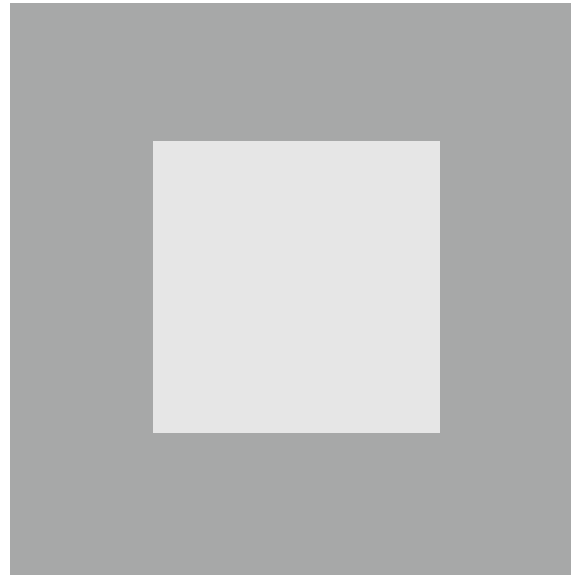
text

Big lesson: Human vision is largely invariant to multiplication

- Contrast between 1 & 2 is the same as between 100 & 200
- Useful to discount the multiplicative effect of light



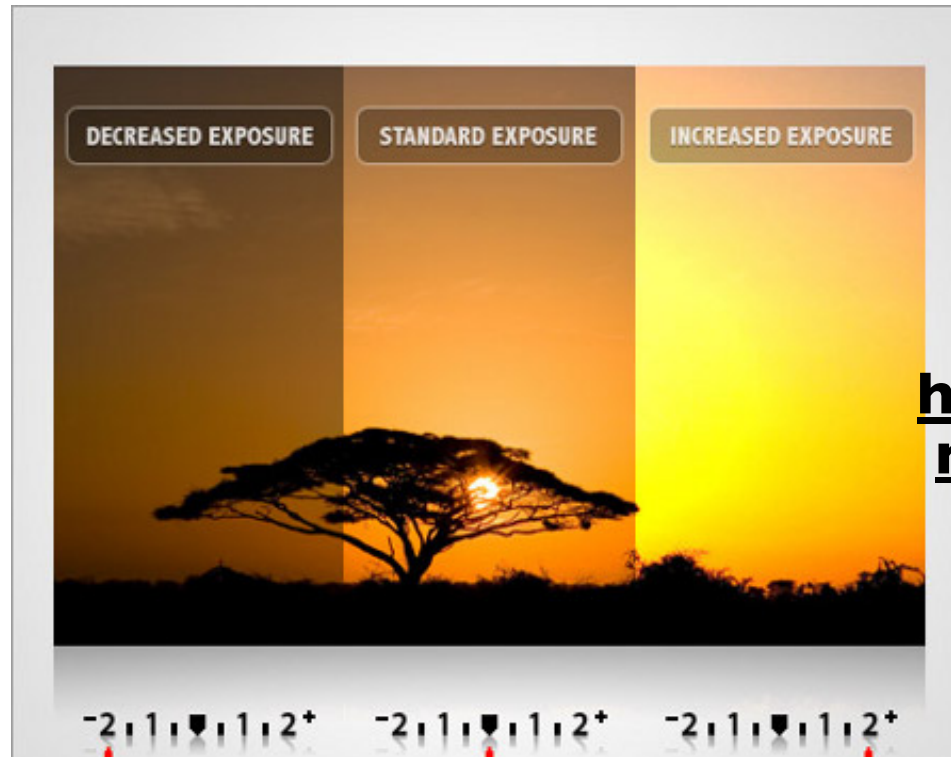
0.1 to 0.2



0.4 to 0.8

Exposure

- On cameras, exposure (shutter speed, aperture, ISO) has a multiplicative effect on the values recorded by the sensor.
- Changes the “brightness”, not contrast



<http://photographystepbystep.com/exposure-2/auto-bracketing/>

Take-home message

- Multiplication and ratios matters
 - Lot of perceptual aspects of images are invariant to a multiplication by a scalar.
 - within reason !
 - Contrast is about ratios

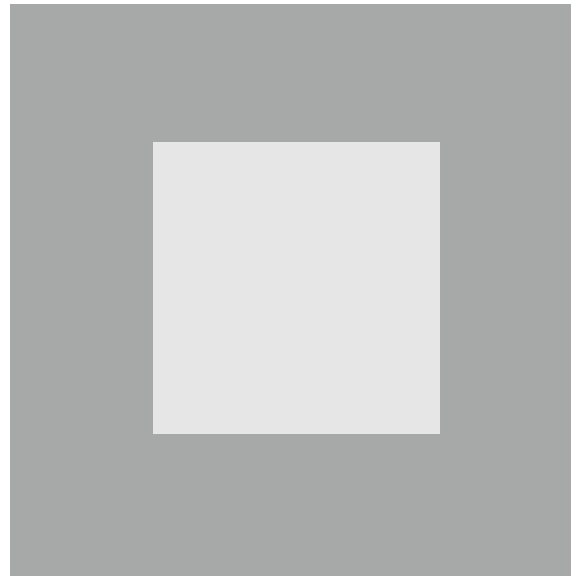
Gamma

Vision invariant to multiplication

- Contrast between 1 & 2 is the same as between 100 & 200
- Useful to discount the multiplicative effect of light



0.1 to 0.2



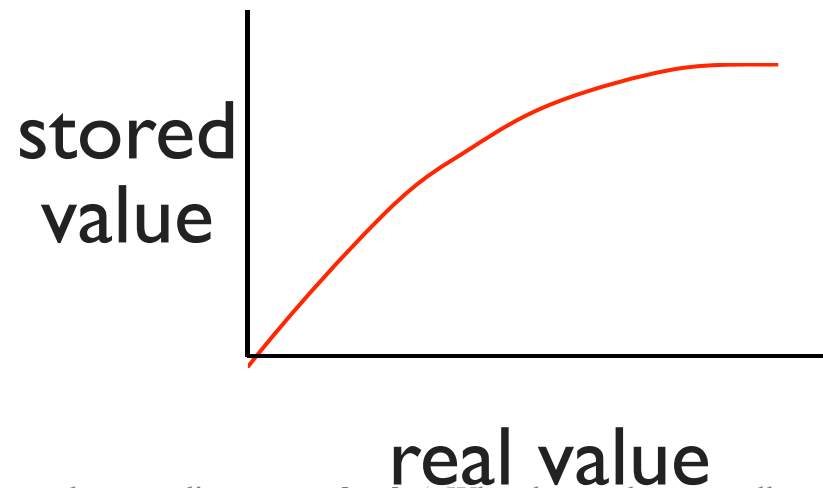
0.4 to 0.8

Problem with quantization and linear encoding

Only 1 quantization level between 1 and 2
but 100 between 100 and 200

Linearity and gamma

- Images are usually gamma encoded
- Instead of storing the light intensity x , they store $x^{\frac{1}{\gamma}}$
- to get more precision in dark areas for 8-bit encoding



gamma compression curve

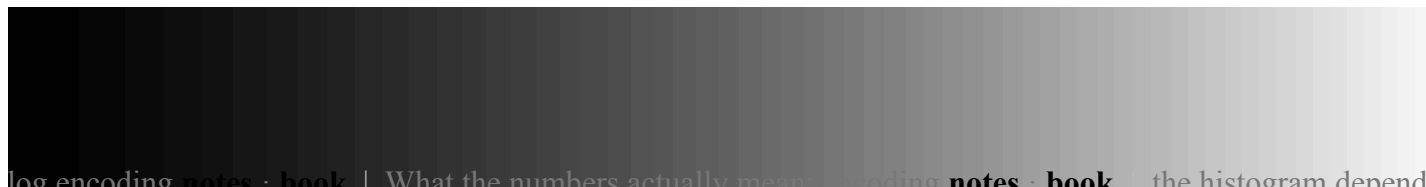
Linearity and gamma

- Images are usually gamma encoded
- Instead of storing the light intensity x , they store $x^{1/\gamma}$
- to get more precision in dark areas

6 bit encoding for emphasis:



Linear

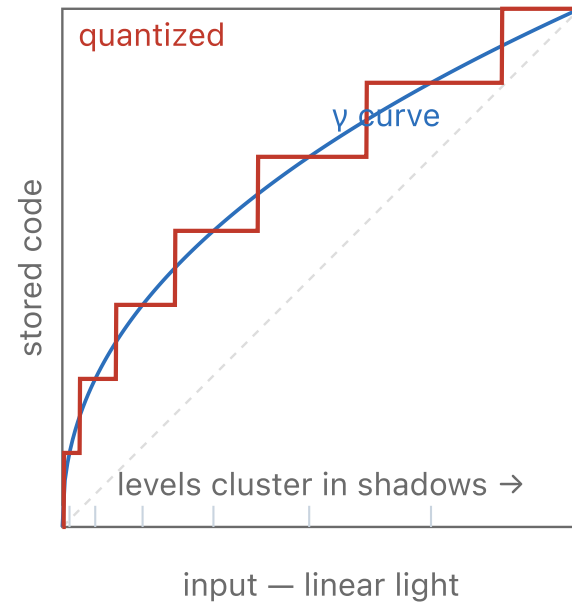


Gamma2.2

Concretely, gamma encoding

- If the camera records an amount of light that is x at a given pixel
- What gets stored is encoded as
- When displaying an image, the OS or display or driver inverts the mapping

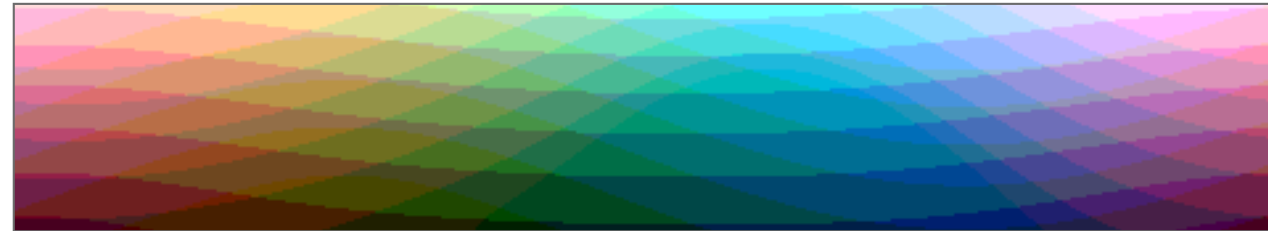
storage $\gamma = 2.2 \cdot 3\text{-bit} \rightarrow 8 \text{ levels per channel} \cdot \text{display} = \text{sRGB}$



grey ramp (perceptually uniform; dark $\rightarrow$ light)



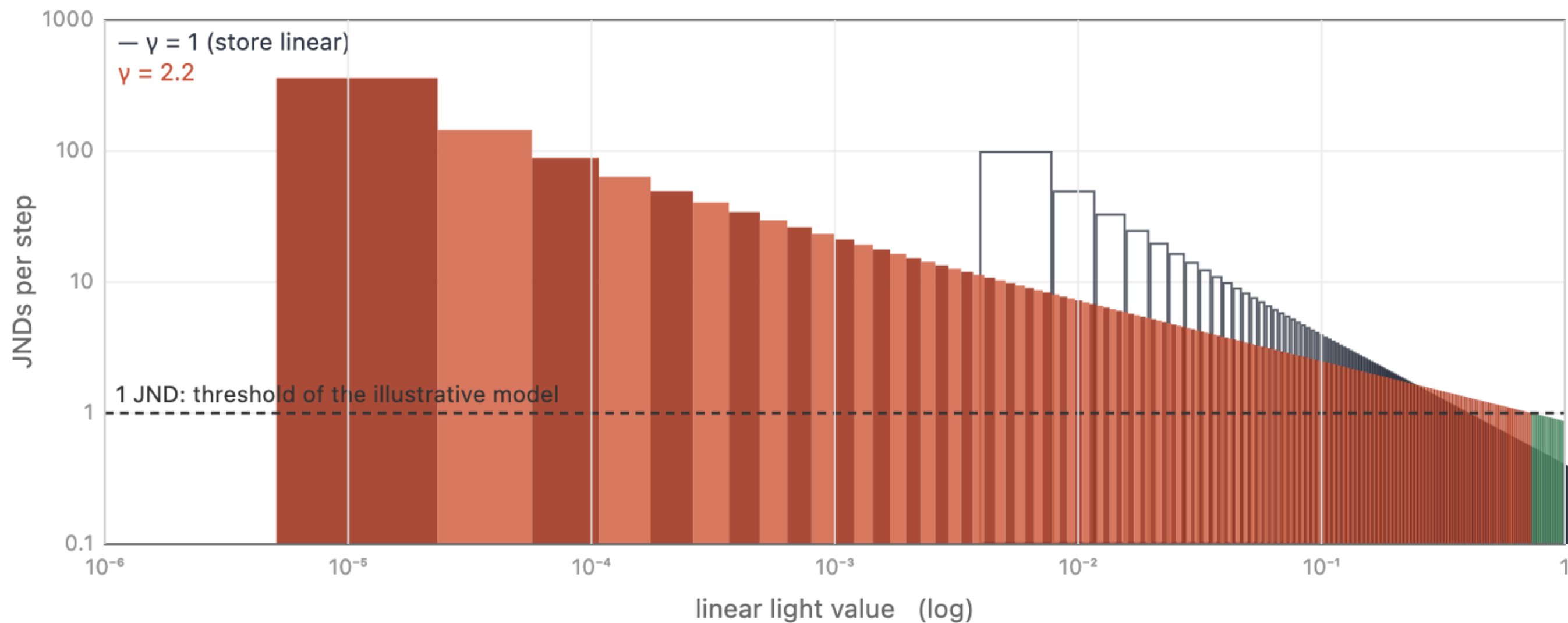
colour gradient (hues even in CIELCh; light $\rightarrow$ dark)



photograph (smooth skies band first)



$\gamma = 2.2$ · 8-bit → 256 levels · 255 steps



$\gamma = 2.2$: 221/255 steps over 1 JND, worst ≈ 359 · $\gamma = 1$: 100/255, worst ≈ 100

First step (starts at zero, infinite under strict Weber, not drawn): ∞ JNDs; linear ∞

Other motivation: noise

- Much image noise has variance proportional to image values (Poisson noise)
- Gamma=2 makes noise level uniform

Number of bits

Most images (including JPG): 8 bit, gamma of mostly 2.2

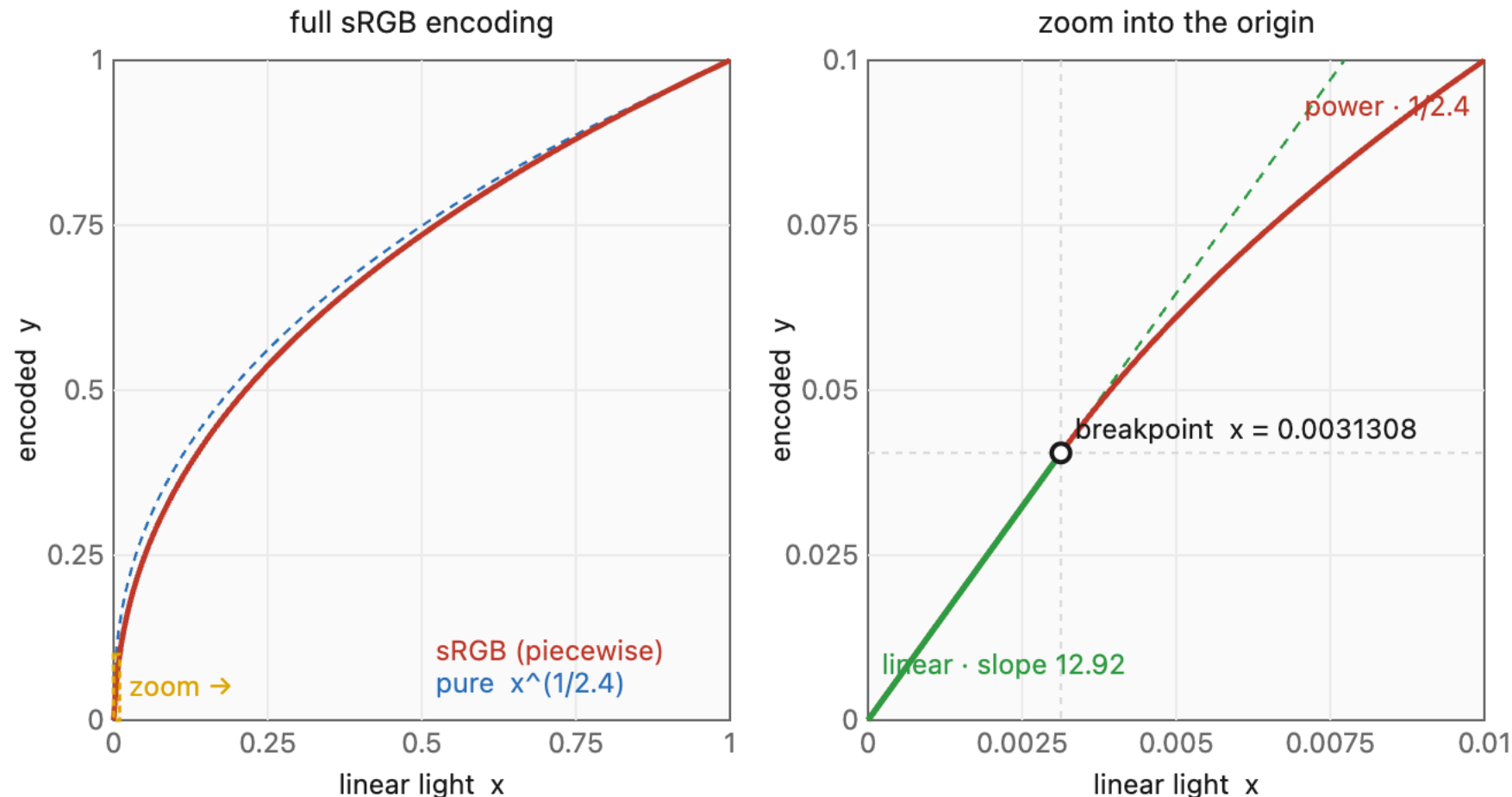
well, it's a bit more messy, the early part of the curve is linearized, see next slide

Camera RAW files use 12-16 bits

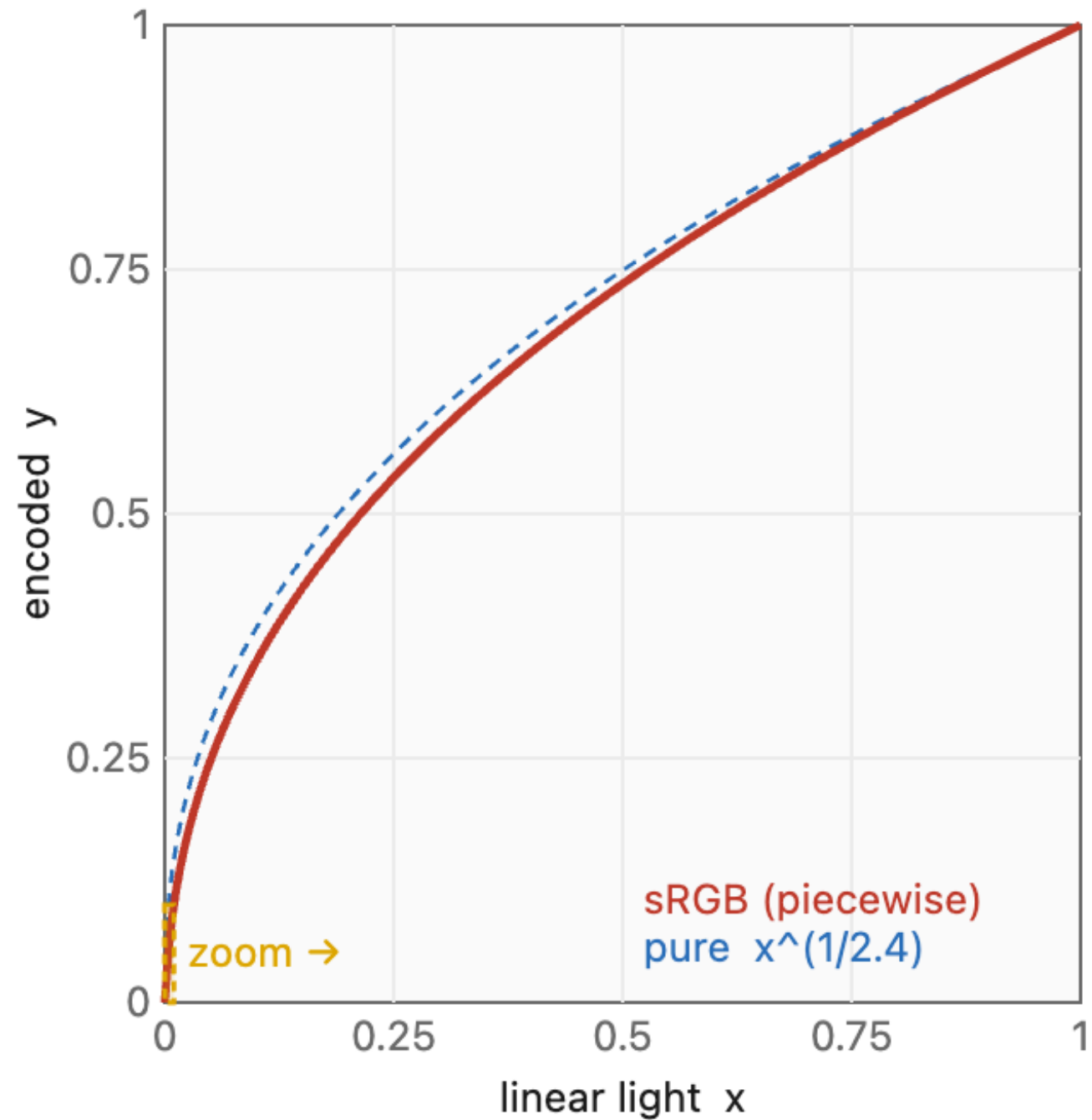
We use floats

Bonus fact

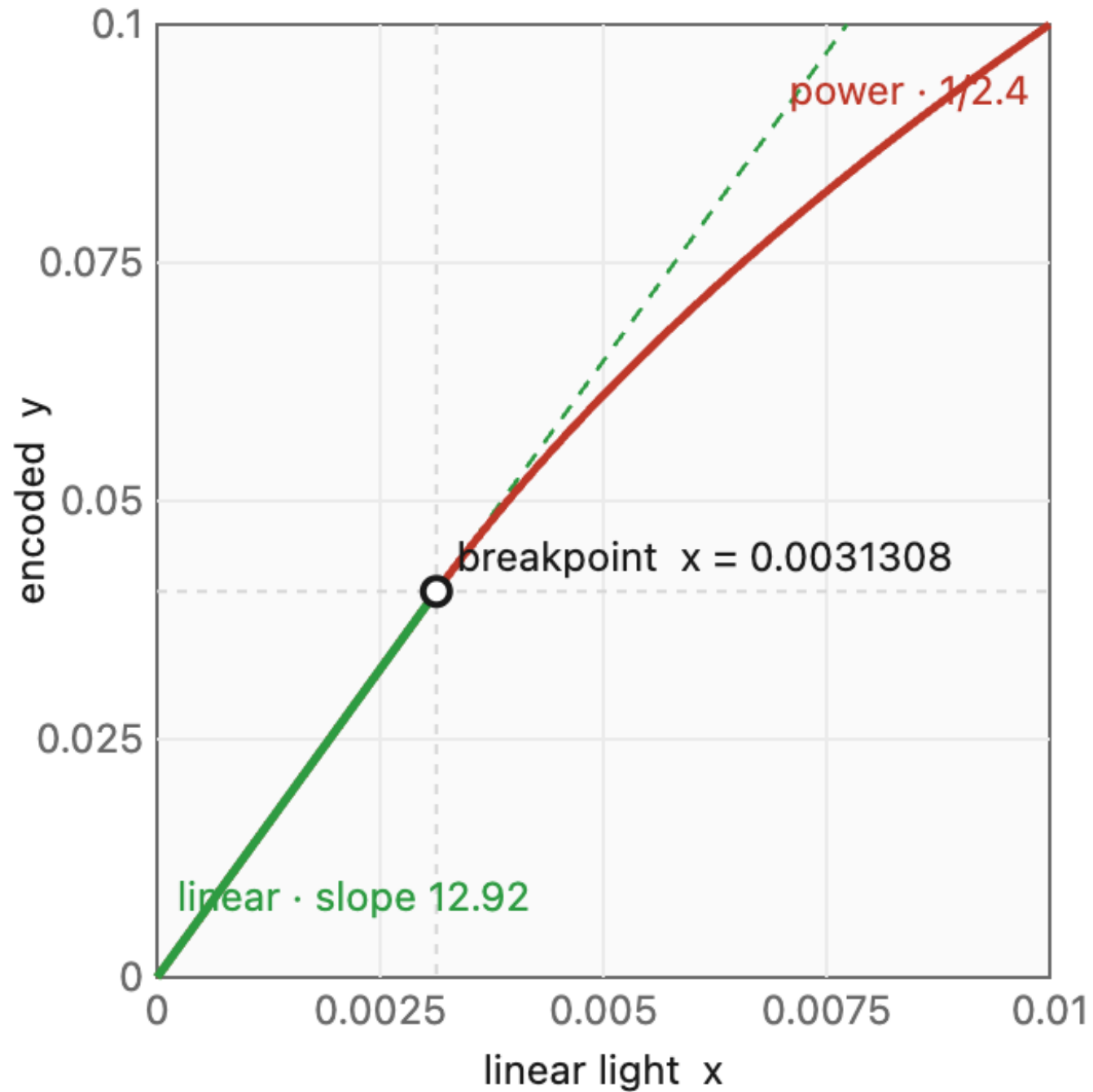
A gamma curve is **not** differentiable around 0.
Deep learning doesn't like this. Use sRGB linearized
curve instead



full sRGB encoding



zoom into the origin

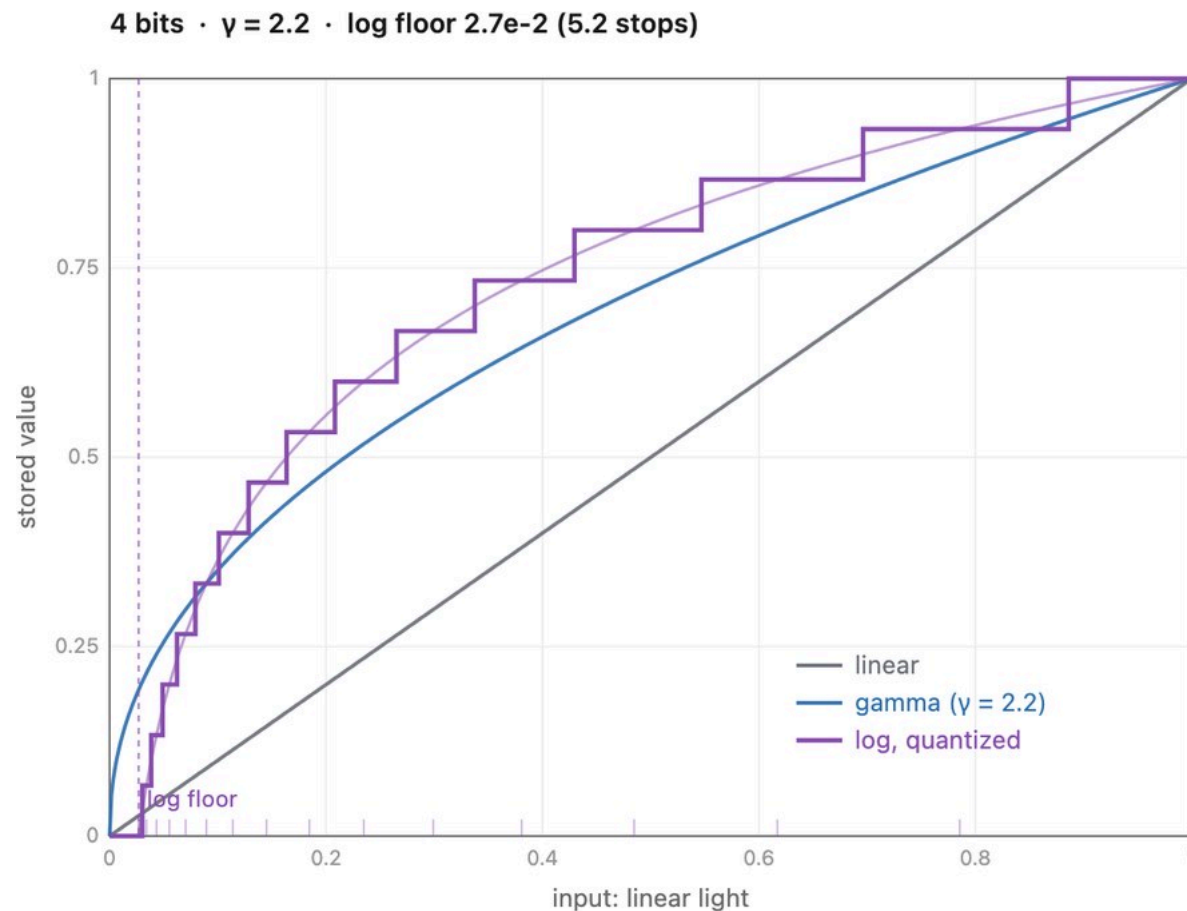


Other encoding: log

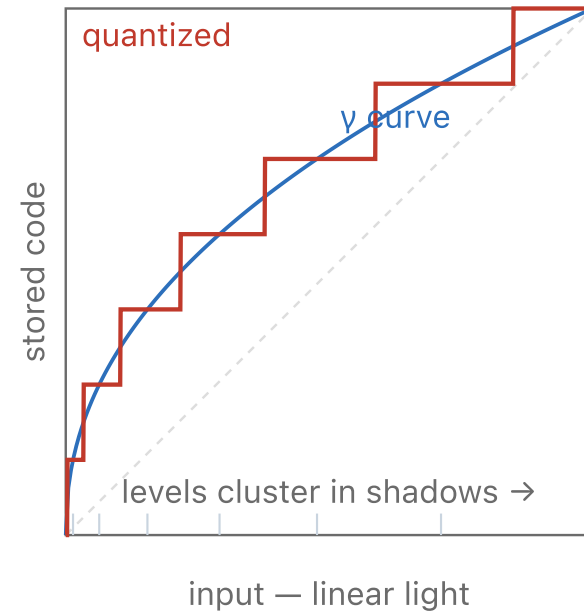
Truly invariant to multiplication

Needed when range is very high

Becoming popular for video encoding



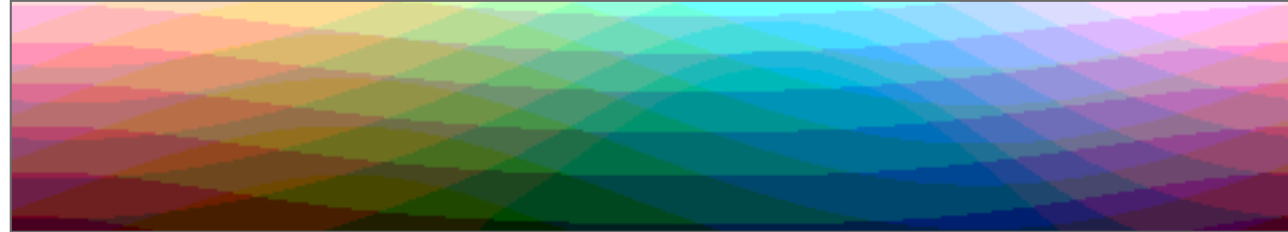
storage $\gamma = 2.2 \cdot 3\text{-bit} \rightarrow 8$ levels per channel $\cdot$ display = sRGB



grey ramp (perceptually uniform; dark $\rightarrow$ light)



colour gradient (hues even in CIELCh; light $\rightarrow$ dark)



photograph (smooth skies band first)



Linearity and gamma

- **Half (well, more like 38%) of image processing works better in linear space**
 - If linearity is important
 - e.g. white balance, deblurring, HDR merging, resampling
- **Half (well, 71 %) work better in gamma space**
 - closer to logarithmic scale, uniform noise, uniform treatment over brightness
 - e.g. edge detection, corner detection
- **Sometimes (31 %) you need log**
 - when dynamic range is really high
 - e.g. tone mapping

White balance

White balance & Chromatic adaptation

- Different illuminants have different color temperature
- Our eyes adapt to this:
Chromatic adaptation
 - We actually adapt better in brighter scenes
 - This is why candlelit scenes still look yellow

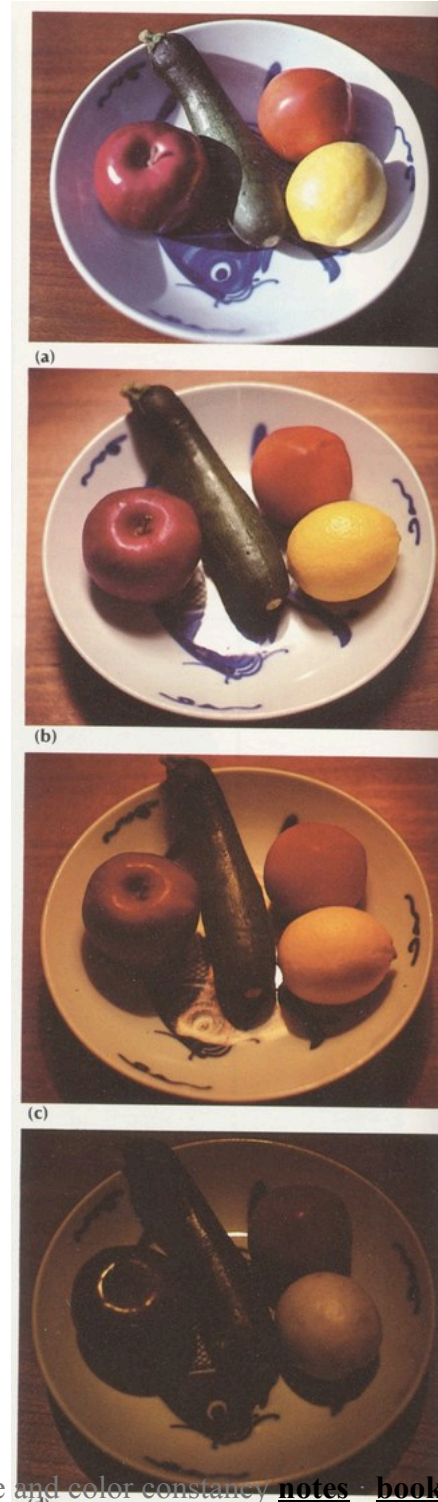


courses.com/guide/guide2-27.html

White balance problem

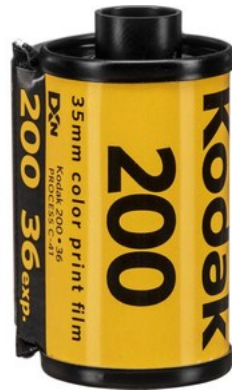
- When watching a picture on screen or print, we adapt to the illuminant of the room, not that of the scene in the picture
- The eye cares more about objects' intrinsic color, not the color of the light leaving the objects
- We need to discount the color of the light source

*Same object,
different illuminants*



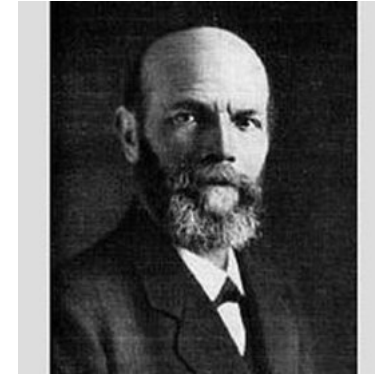
White balance & Film

- Different types of film for fluorescent, tungsten, daylight
- Need to change film!



- Electronic & Digital imaging are more flexible

Von Kries adaptation



- Multiply each channel by a gain factor
 - $R' = R * k_r$
 - $G' = G * k_g$
 - $B' = B * k_b$



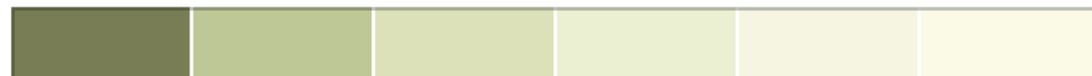
<http://www.cambridgeincolour.com/tutorials/white-balance.htm>

as shot, under tungsten (CIE A, 2856 K)



a neutral wedge, as shot

with your gains



the same wedge, corrected: gray means you are there

off by 0.68 stops on the worst cone

your gains L 0.89 M 1.06 S 1.90 the answer L 0.89 M 1.06 S 3.04

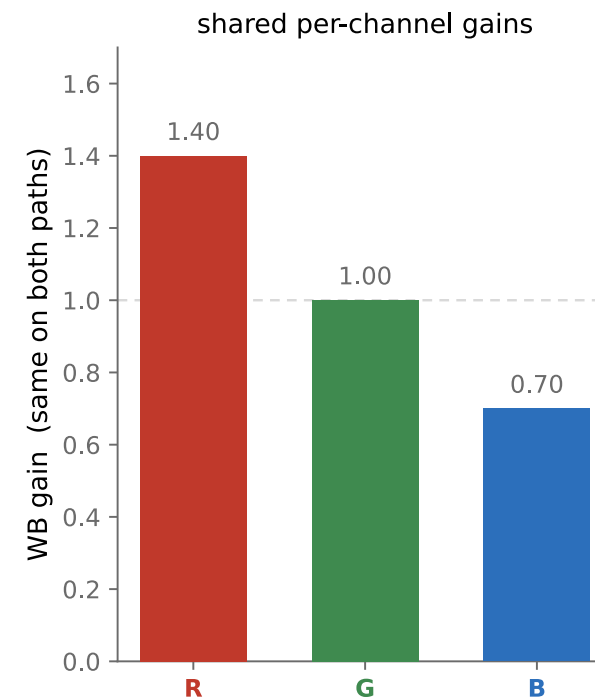
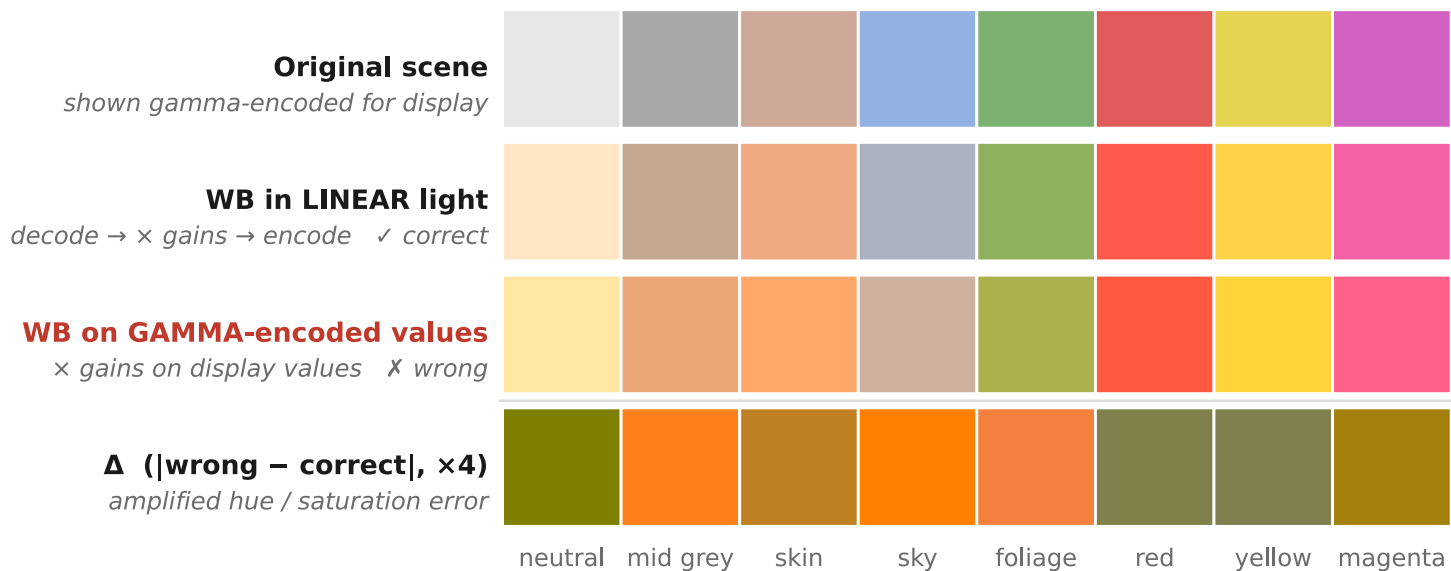
a camera would call this 3050 K, tint -34

click either picture on something that ought to be gray, and it will be

White balance must be in linear values

Not gamma encoded values !!!

White balance is correct only in LINEAR light: the same per-channel gains shift hue & saturation when applied to gamma-encoded values



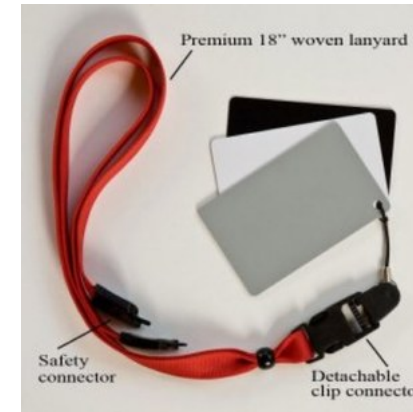
every patch over-saturates
(up to +23% on mid grey),
hue twists up to 21° (foliage)

White balance challenge

- How do we find the scaling factors for r, g, and b?

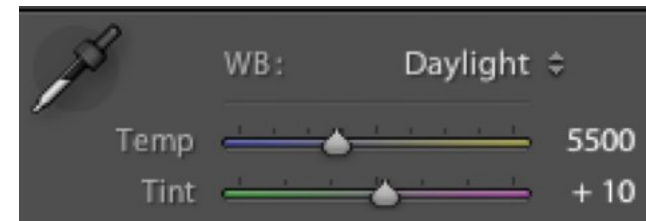
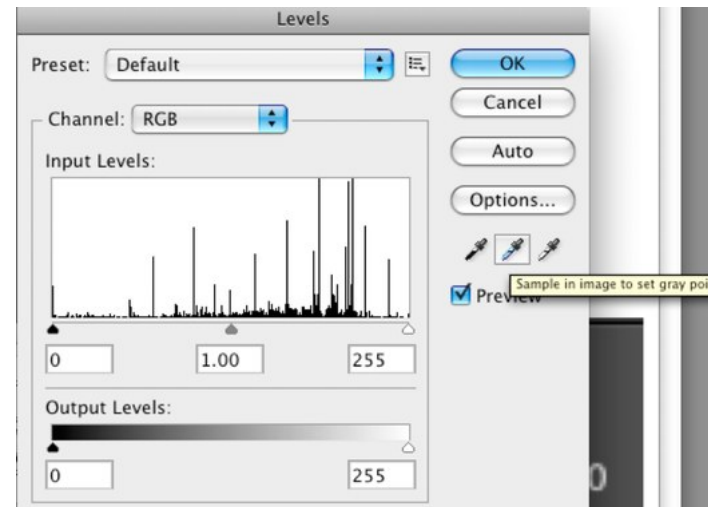
Best way to do white balance

- Grey card:
- Take a picture of a neutral object (white or gray)
- Deduce the weight of each channel
- If the object is recoded as rw , gw , bw use weights k/rw , k/gw , k/bw where k controls the exposure



Lightroom demo

- Most photo editing software lets you click on a neutral object to achieve white balance
- In “Levels” in Photoshop
- In “basic” in Lightroom
- In Adjustments in Aperture
- You also often have presets such as daylight, tungsten



Party name tags

- Provide excellent white references!



`imwrite(im/im[300, 214])`

Without grey cards

- We need to “guess” which pixels correspond to white objects

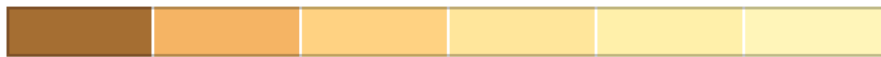
Grey world assumption

- Assume average color in the image is grey
- Use weights proportional to

$$\frac{1}{\int_{image} r}, \frac{1}{\int_{image} g}, \frac{1}{\int_{image} b}$$

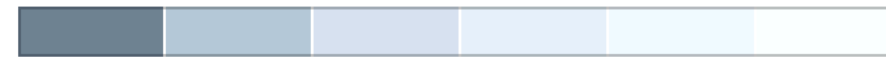
- Usually assumes 18% grey to set exposure

as shot, under tungsten



a neutral wedge, as shot

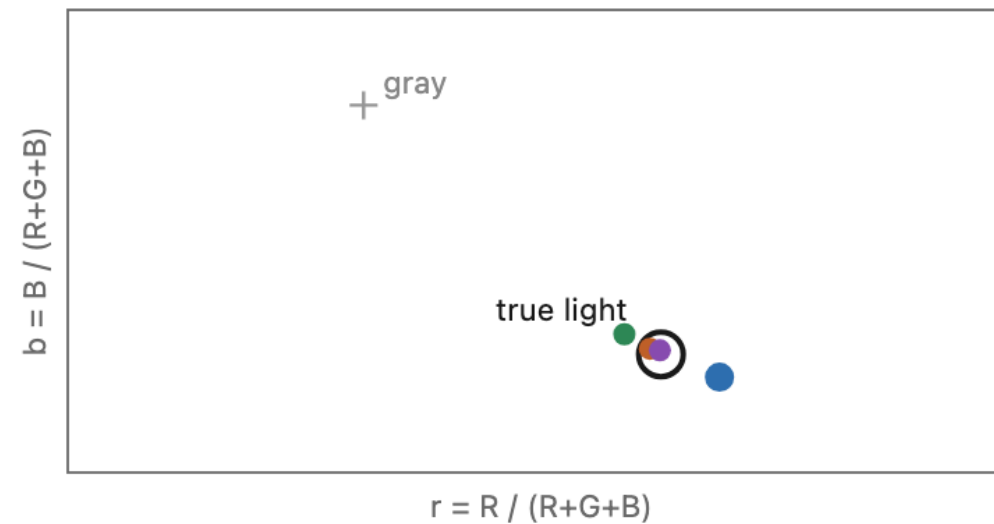
balanced by gray world



the same wedge, balanced

angle to the true light

● gray world	4.9°
● brightest pixels	1.0°
● shades of gray	0.4°
● gray edge	3.2°
uncorrected: 34.5°	



Brightest pixel assumption

- Highlights usually have the color of the light source
 - At least for dielectric materials
- White balance by using the brightest pixels
 - Plus potentially a bunch of heuristics
 - In particular use a pixel that is not saturated/clipped

Light with color control

- http://www.robgalbraith.com/bins/content_page.asp?cid=7-11666-11979



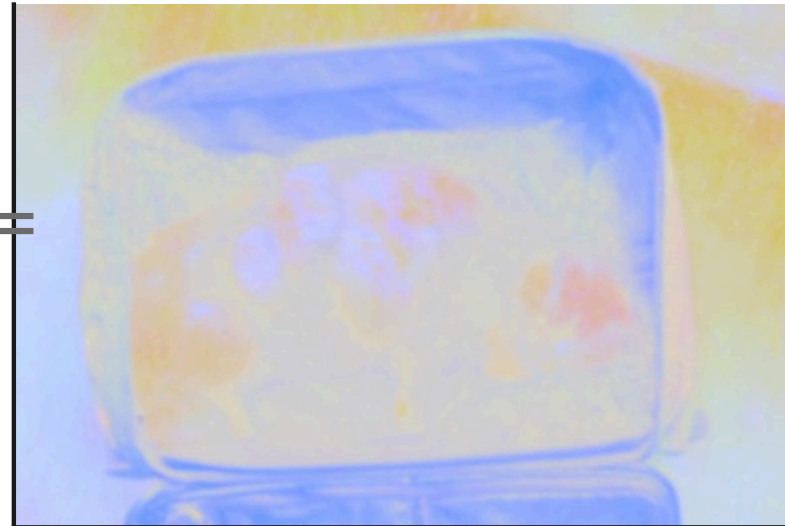
Mixed lighting

One WB per pixel!
Still mostly an open problem

input: raw capture
(what the camera recorded)



per-pixel coefficients
(inferred local illuminant color)



white-balanced result
(the factor we want back)

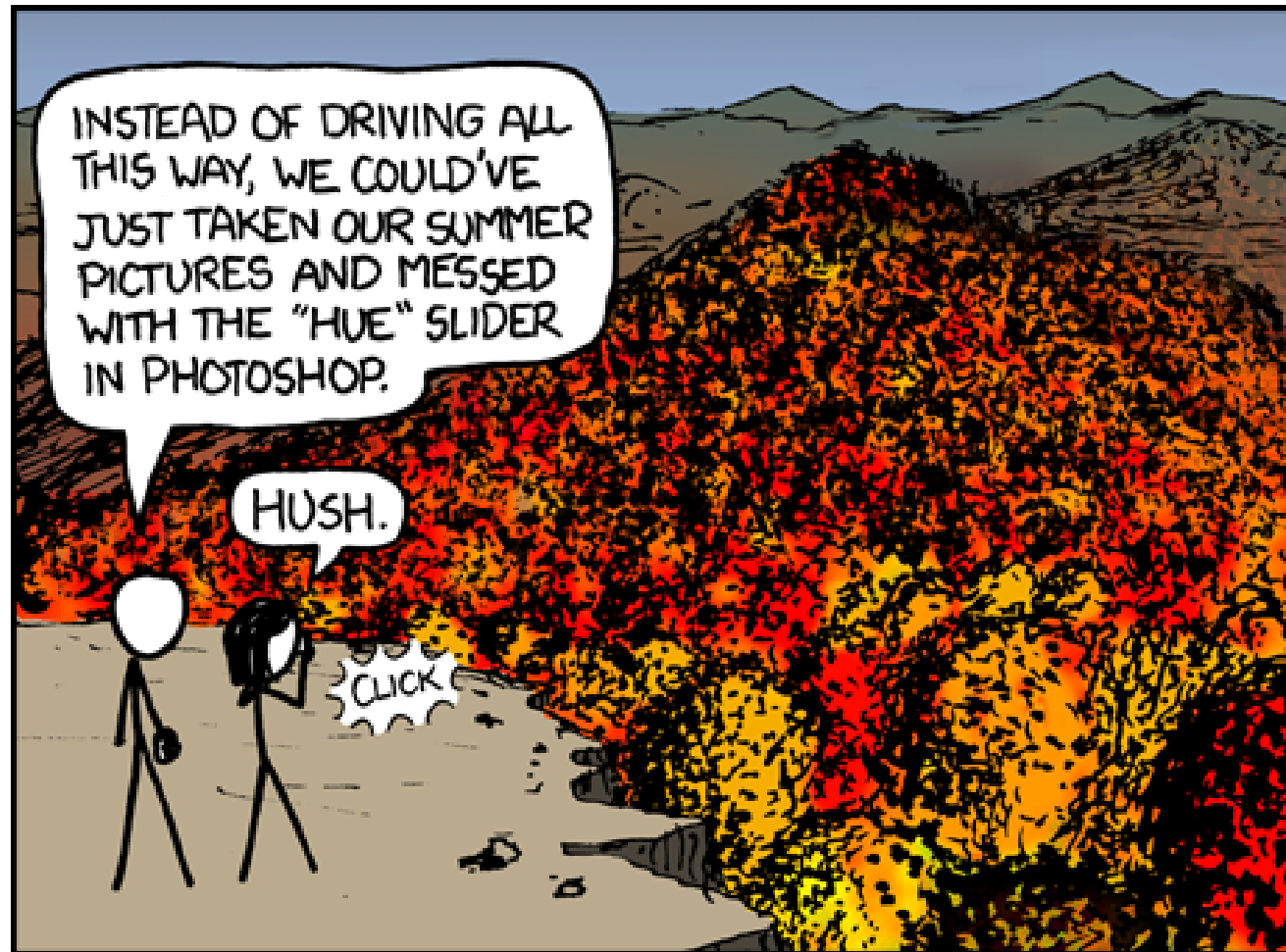


input = coefficients × white-balanced (global WB: one triplet everywhere; local WB: a different triplet per pixel)

Refs

- Work on color constancy
 - <http://gvi.seas.harvard.edu/paper/perceptionbased-color-space-illuminationinvariant-image-processing>
 - <http://gvi.seas.harvard.edu/paper/color-subspaces-photometric-invariants>
 - <http://people.csail.mit.edu/billf/papers/BayesJOsA.pdf>
 - <https://drive.google.com/file/d/1id74VNDL8ACrrWf6vYgN2M4kS8gd4n7w/view>
 - <https://arxiv.org/abs/1611.07596>
- Still an open problem!
 - Especially for multiple illuminants

Questions?



from xkcd

Take home messages

- Discounting the illumination is useful
- Ratios matter
- Optical illusions are not optical but fun
- White balance is needed because we adapt to the viewing conditions, not the depicted scene

In the real world

- e.g. Adobe Lightroom 's first processing steps are in linear space, then converts to some gamma space
- Most practitioners (don't even know images are gamma encoded and) work in gamma encoded space



notes: What the numbers actually mean... encoding notes · book

Gamma in the analog world

- Film has gamma
 - For both encoding and for compensating for different contrast perception at different brightness levels
- Old-style TVs had gamma
 - The response of Cathod Ray Tubes (CRT) happens to naturally decode gamma
(because of the way the electron gun accelerates and focuses electrons, plus the phosphors response)

Take home message

- Images are usually gamma-encoded
 - gamma 2.2
 - provides better quantization
 - sometimes good for algorithms
 - sometimes bad
 - convert to linear values!

POINT OPERATION

Point operations

- for each pixel :

$$R_{out}, G_{out}, B_{out} = f(R_{in}, G_{in}, B_{in})$$

- and f is the same for all pixels
- Often discussed for greyscale version:
 - $L_{out} = f(L_{in})$

Classic point operations

Exposure (multiply)

Brightness (addition in gamma)

Contrast (affine/linear around a pivot)

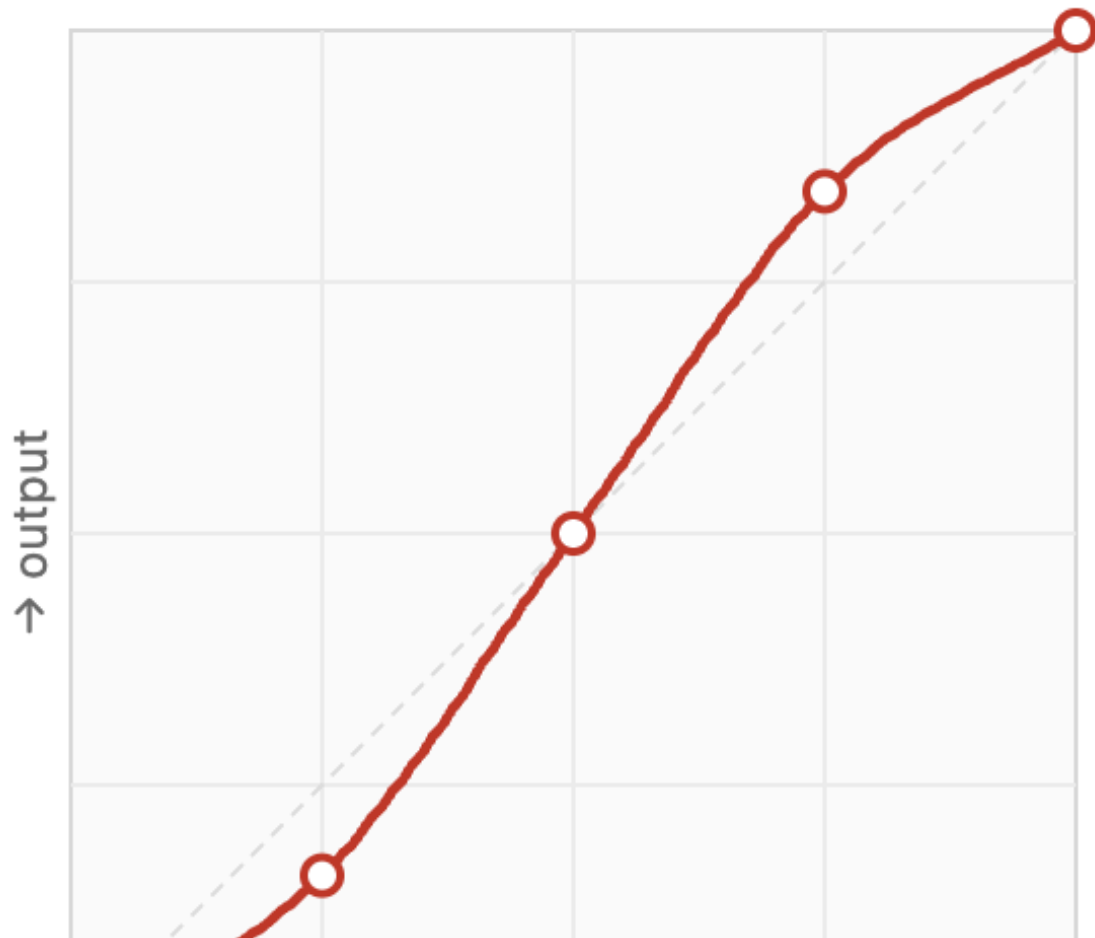
Black point (subtract in linear)

Old-style image operators were more qualitative (e.g. brightness, levels)
and often oblivious to gamma encoding

Modern versions (e.g. exposure) try to be more quantitative and grounded in photometry when it applies

More in notes/book

General tone curve (splines)



input



output



Color point operations

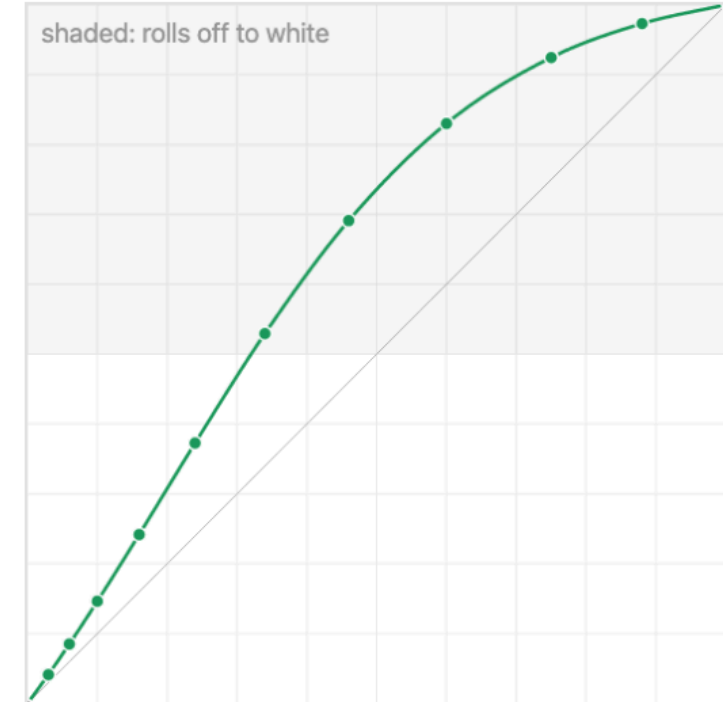
Saturation

Vibrance (saturation but less for already saturated colors)

More advanced: preferred color, LUT, per hue.

The beauty curve. Images look flat without it

photo faces in sun highlights ☒ roll off to white ☐ keep hue view ☒ split ☐ flip



sRGB code in → sRGB code out

saturation 1.00

vibrance 0.50

book look

identity

Demo courtesy of Frédo Durand, MIT

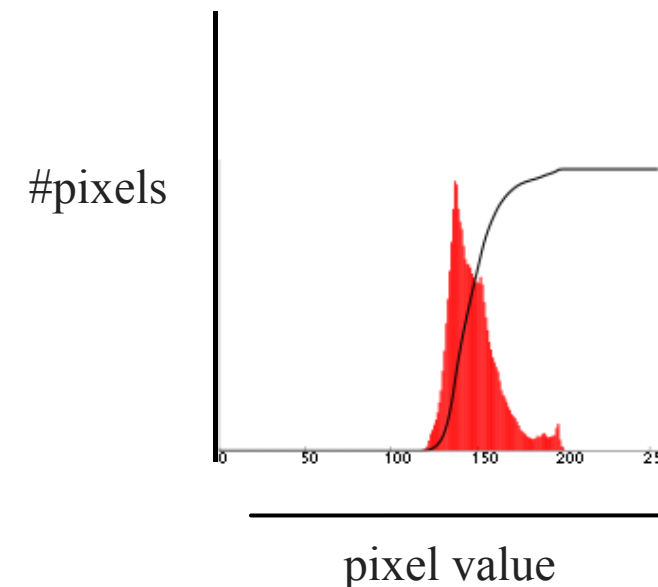
Pset 0 & 1

- Exposure change (brightness)
- Contrast: change ratios
- Luminance chrominance
 - ability to change luminance/brightness independent of color
 - save pure-color information
 - need to respect multiplicative factors

Histograms

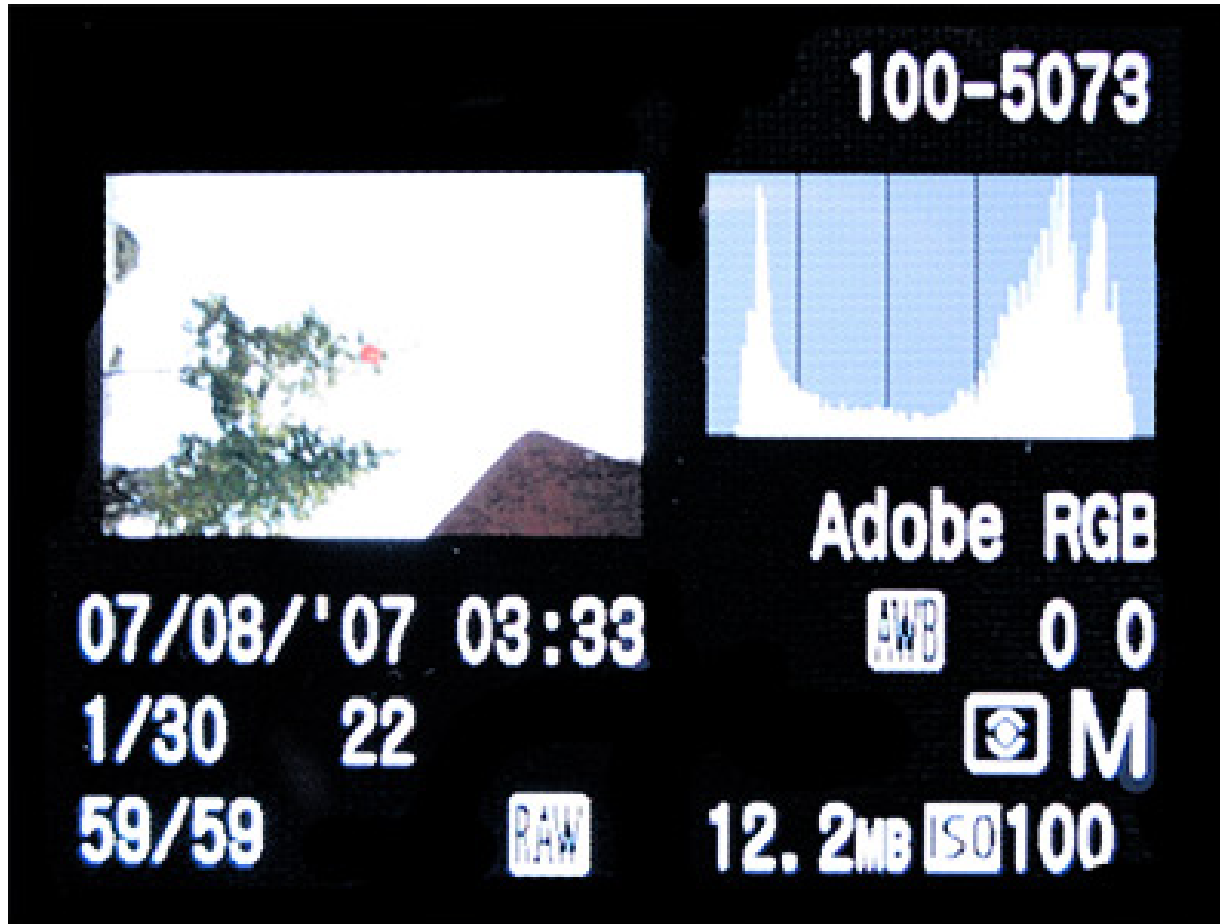
Histogram

- For each value (e.g. 0-255) how many pixels have this value?
- Cumulative histogram: for each value x , how many pixels have a value smaller than x



Very useful on camera

- Allows you to tell if you use the dynamic range entirely



Speaker notes



Bad: bright values under-used
(underexposure)

Bad: bright values saturate
(overexposure)

<http://www.luminous-landscape.com/tutorials/understanding-series/understanding-histograms.shtml>