



From: <http://www.devx.com/projectcool/Article/19987/0/page/2>

The small green squares are the same color. Hold your hand to the screen and block out the adjacent blocks, showing only the two green square, and you'll see how dramatic the chromatic induction effect is.



From: <http://www.devx.com/projectcool/Article/19987/0/page/2>

These two brown shades are different colors—yet the square blocks appear to be the same shade of brown. It is only when your eye follows the line down that you see how different the shades are. Cover the bottom part of the illustration with your hand and look only at the brown boxes to see the way the chromatic induction effect

Image Processing 101
Fredo Durand
MIT EECS
6.8370/6.8371

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

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

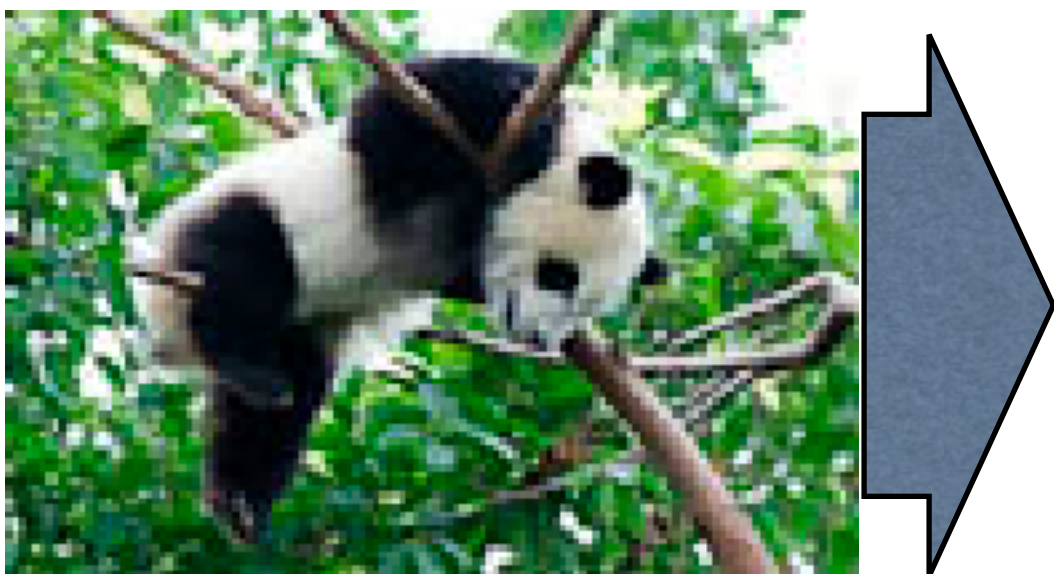


Domain vs. range

- 2D plane: domain of images
 - x, y Cartesian coordinates
- color value: range ($\mathbb{R}^3$ for us)
 - stores the red, green and blue components in $\text{im}(y, x, 0)$, $\text{im}(y, x, 1)$, $\text{im}(y, x, 2)$, respectively

Basic types of operations

Point operations:
range only



Each pixel is treated the same,
doesn't depend on other
pixels, only on the input color

Basic types of operations

Point operations:
range only



Domain
operations



Only moves pixels around,
doesn't change colors,
just where they are

Basic types of operations

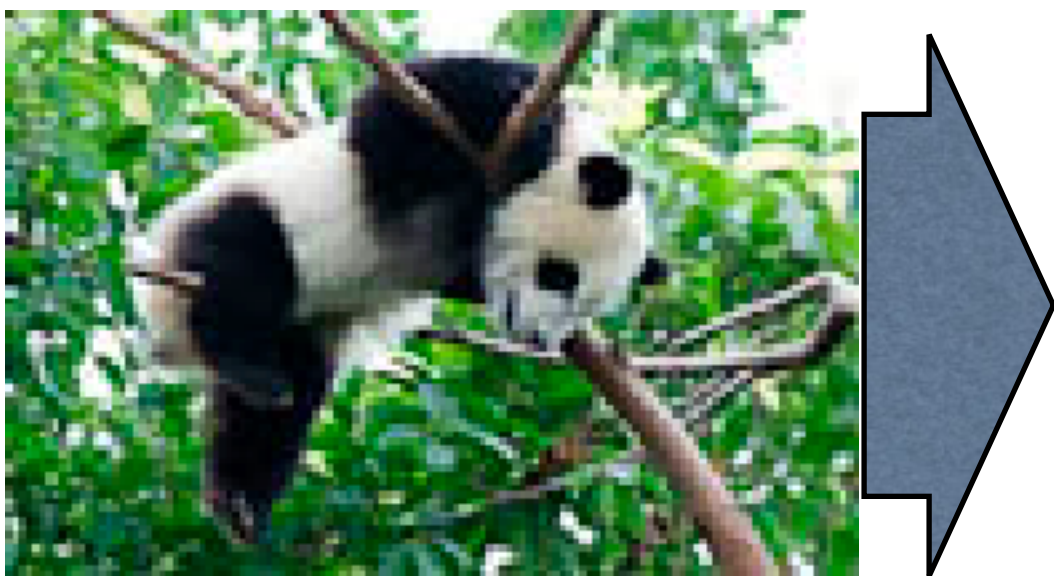
Point operations:
range only



Domain
operations



Neighborhood
operations:
domain and range



Basic types of operations

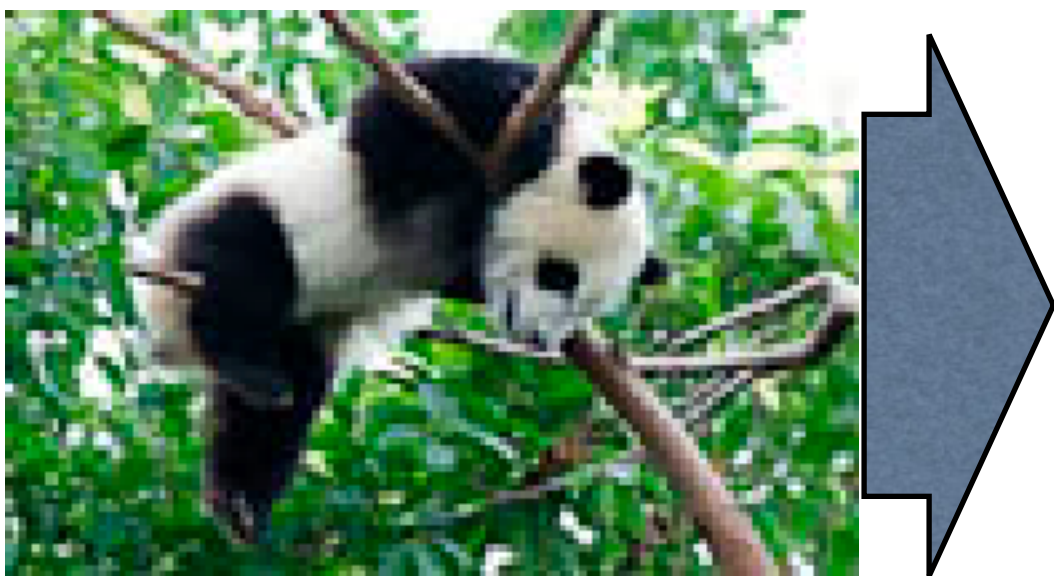
Point operations:
range only
Pset 1



Domain
operations
Pset 3 or 4



Neighborhood
operations:
domain and range
Pset 2



Images in C++

Digital Images

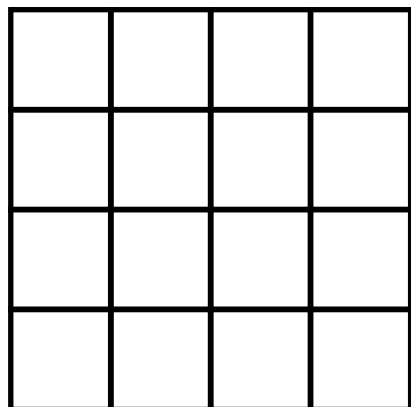
- Can be encoded as 2D arrays of RGB triplets
- Formally:
 - The domain is the 2D plane
 - The range is the RGB space
- Other color spaces are possible and will be used
- Values often encoded as 8 or 16 bit integers
 - But we will use floats in $[0..1]$ to make life easier

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

- vectors only have one 1D index
- turn 2D index into 1D one through strides
 - pixel at x, y , stored at $y * \text{width} + x$
 - width is called the “stride”

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

- 3D index x, y, c where c is the color channel:
 - $c * \text{width} * \text{height} + y * \text{width} + x$

Our Image class

- **Stores in planes of color channels**
 - size $\text{width} * \text{height} * \text{channels}$
 - 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

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**

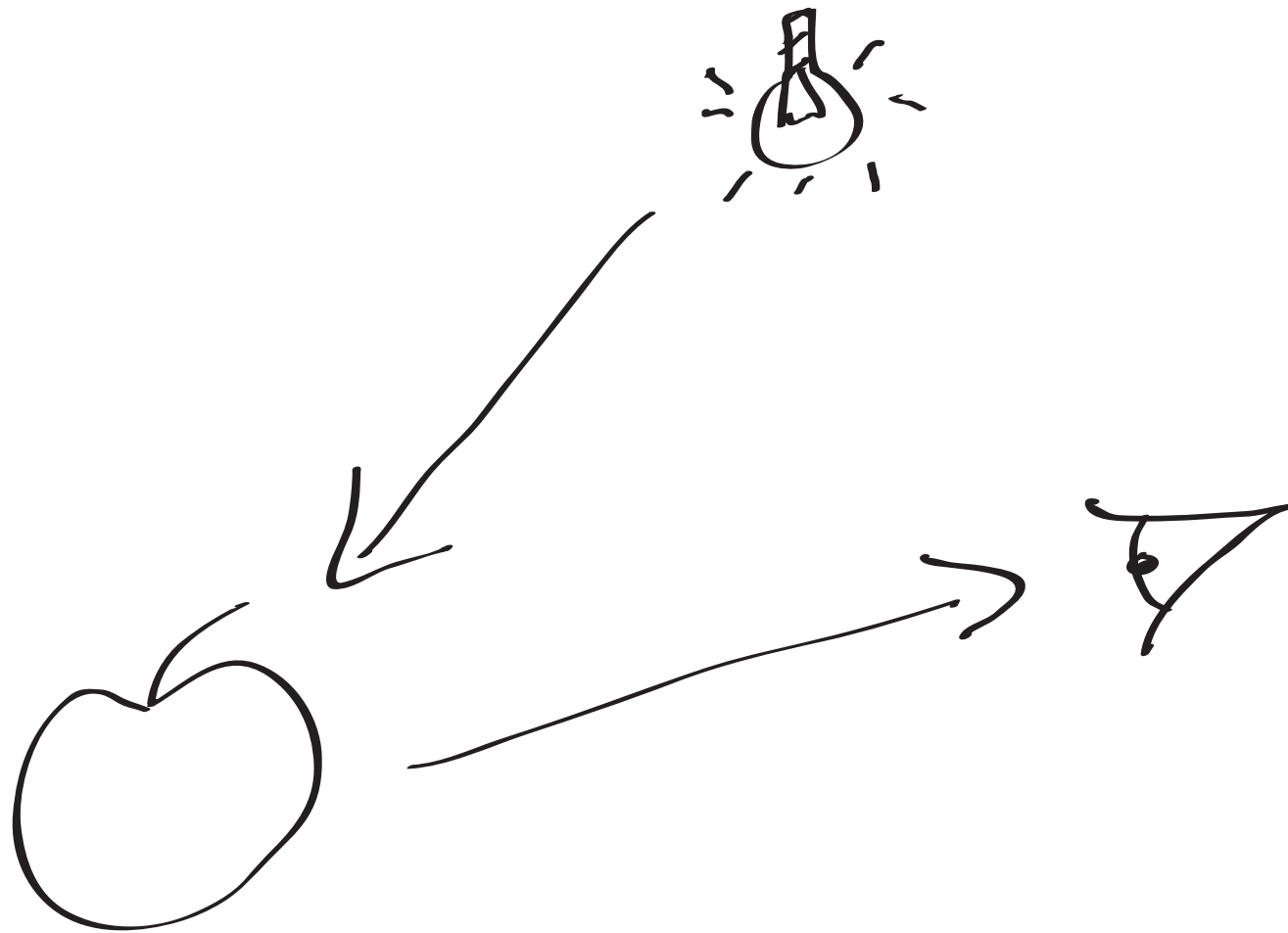


Light & perception



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

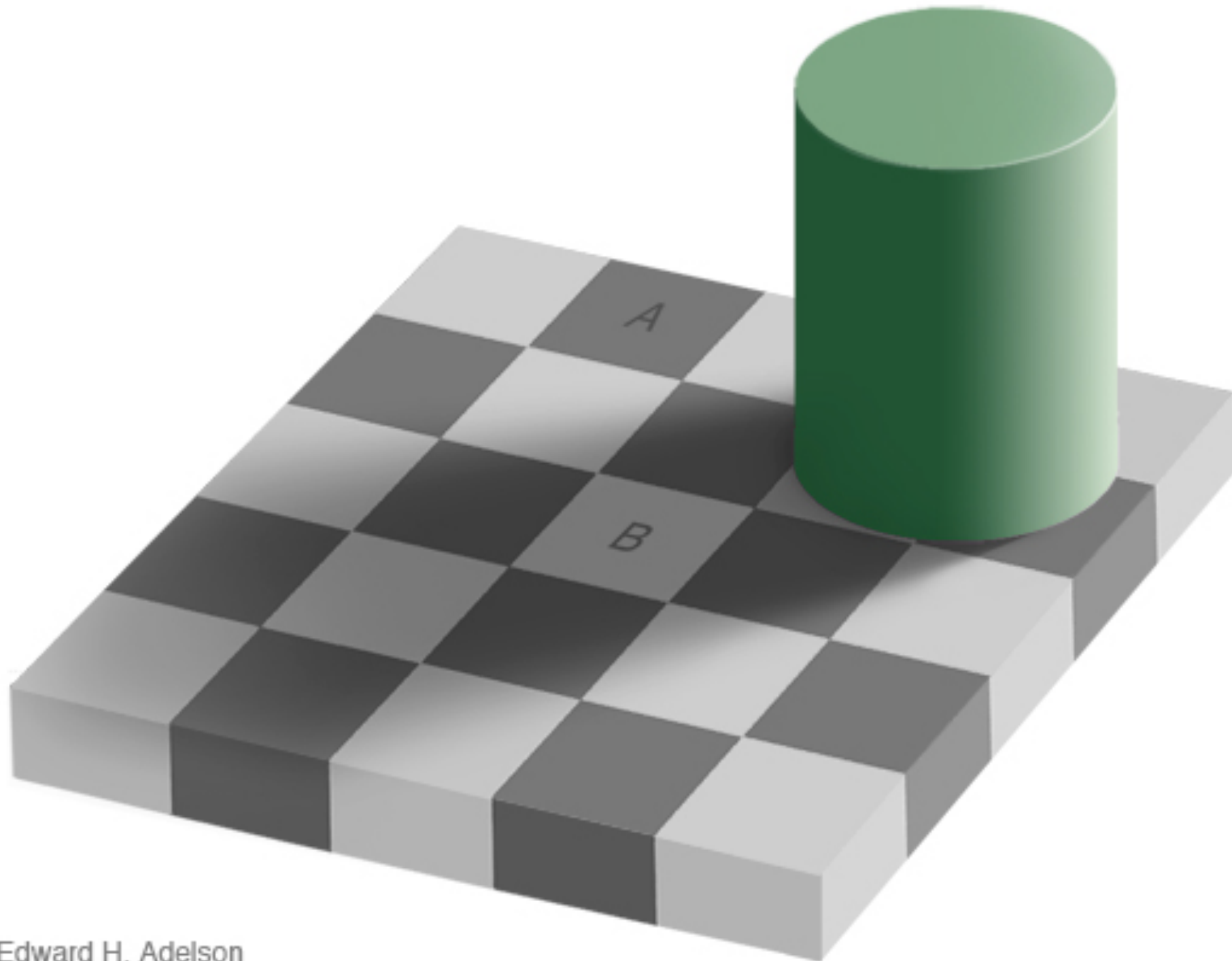


Human perception

- 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

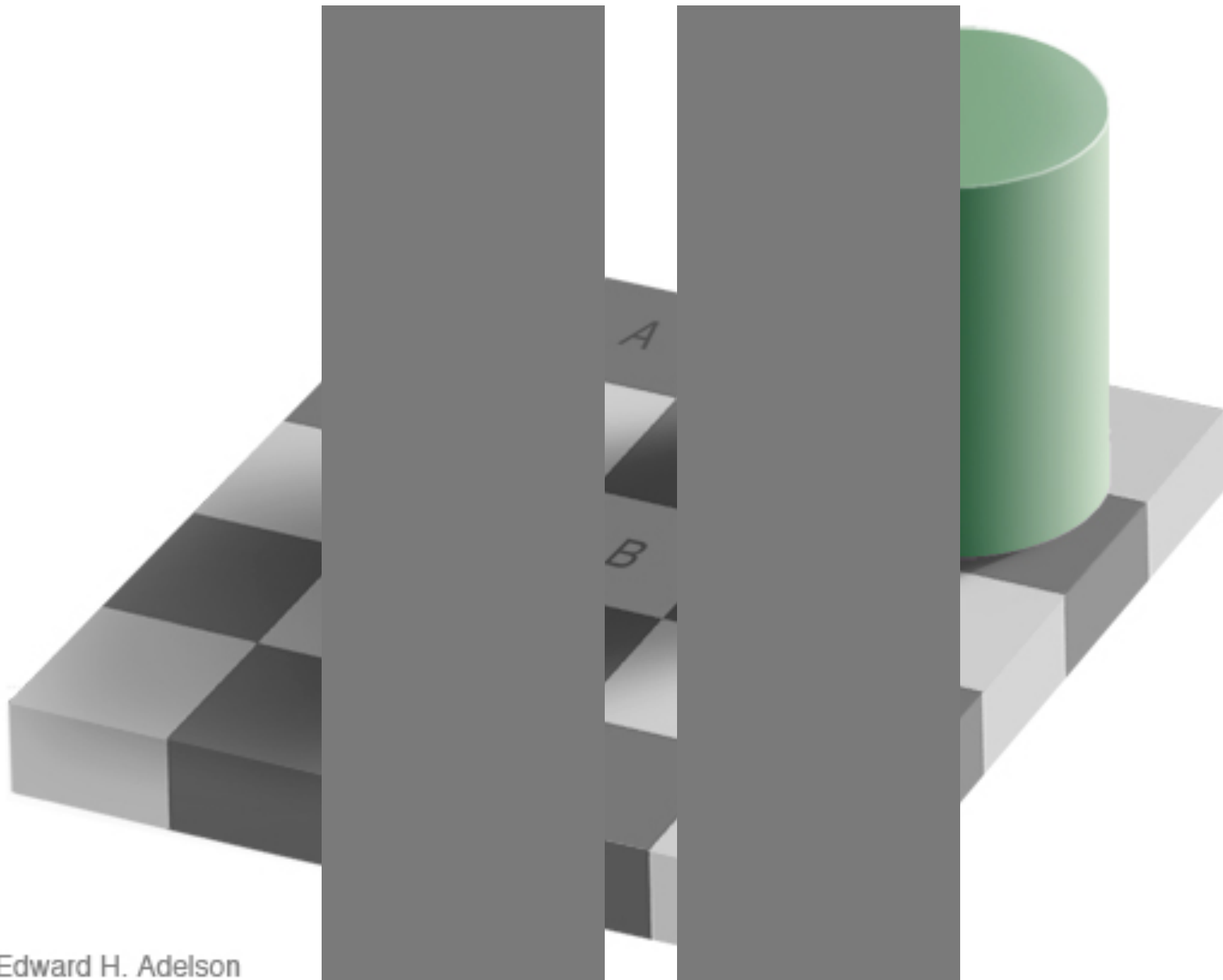
Illusion by Adelson

- A & B have exactly the same tone



Illusion by Adelson

- A & B have exactly the same tone



Edward H. Adelson

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

Contrast is about ratios

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



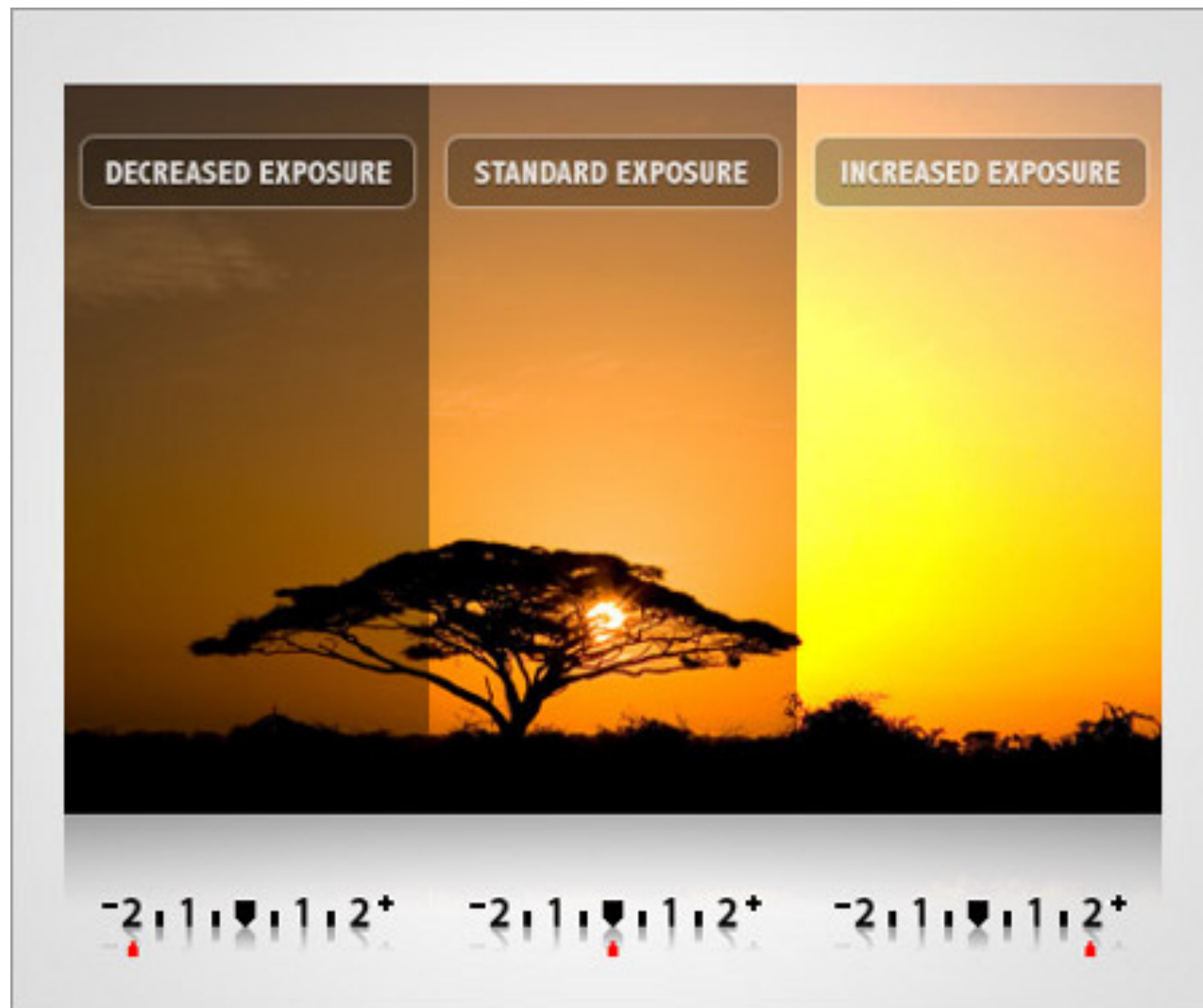
0.1 to 0.2



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



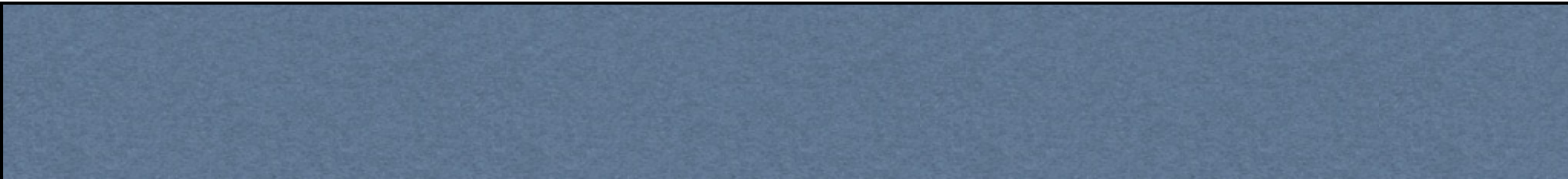
<http://photographystepbystep.com/exposure-2/auto-bracketing/>

Take-home message

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

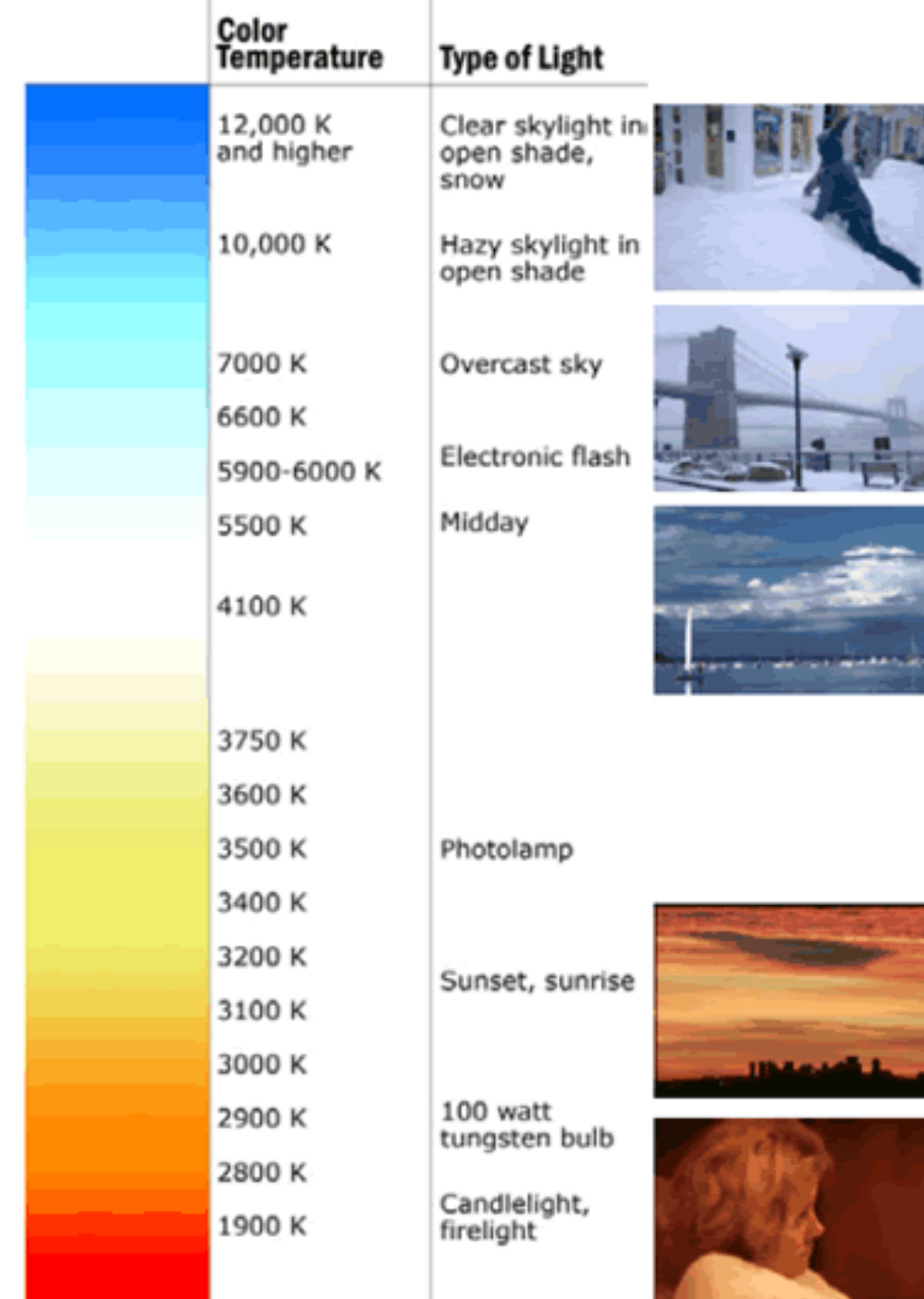


White balance



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



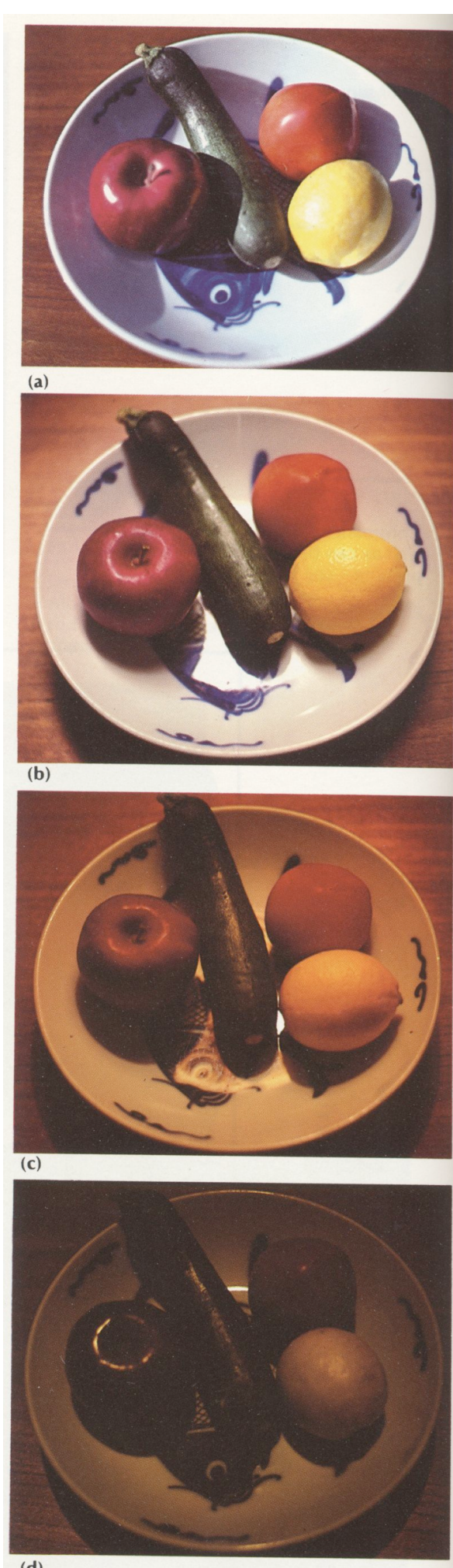
www.shortcourses.com/guide/guide2-27.html



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

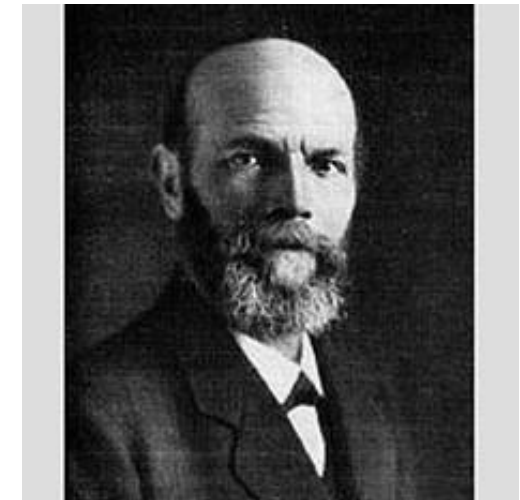
*Same object,
different illuminants*



White balance & Film

- Different types of film for fluorescent, tungsten, daylight
- Need to change film!
- Electronic & Digital imaging are more flexible

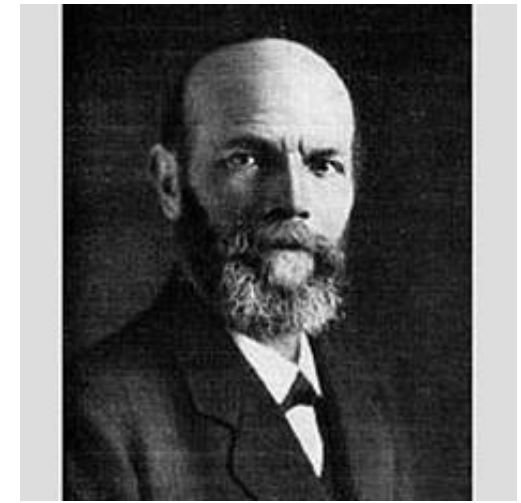
Von Kries adaptation



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



Von Kries adaptation



- Multiply each channel by a gain factor
- Note that the light source could have a more complex effect
 - Arbitrary 3x3 matrix
 - More complex spectrum transformation



White balance challenge

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

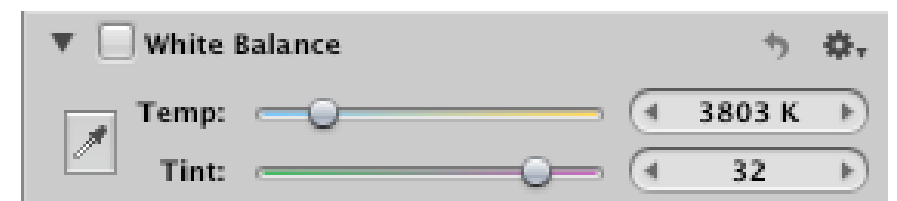
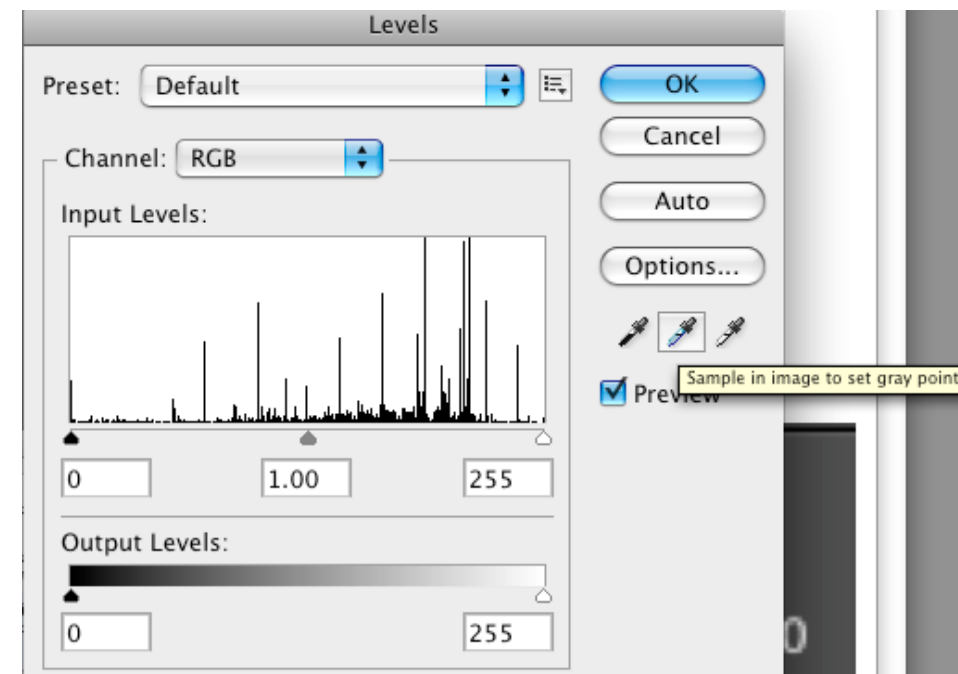
Best way to do white balance

- Grey card:
- Take a picture of a neutral object (white or gray)
- Deduce the weight of each channel
- If the object is recoded as r_w , g_w , b_w use weights k/r_w , k/g_w , k/b_w 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

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

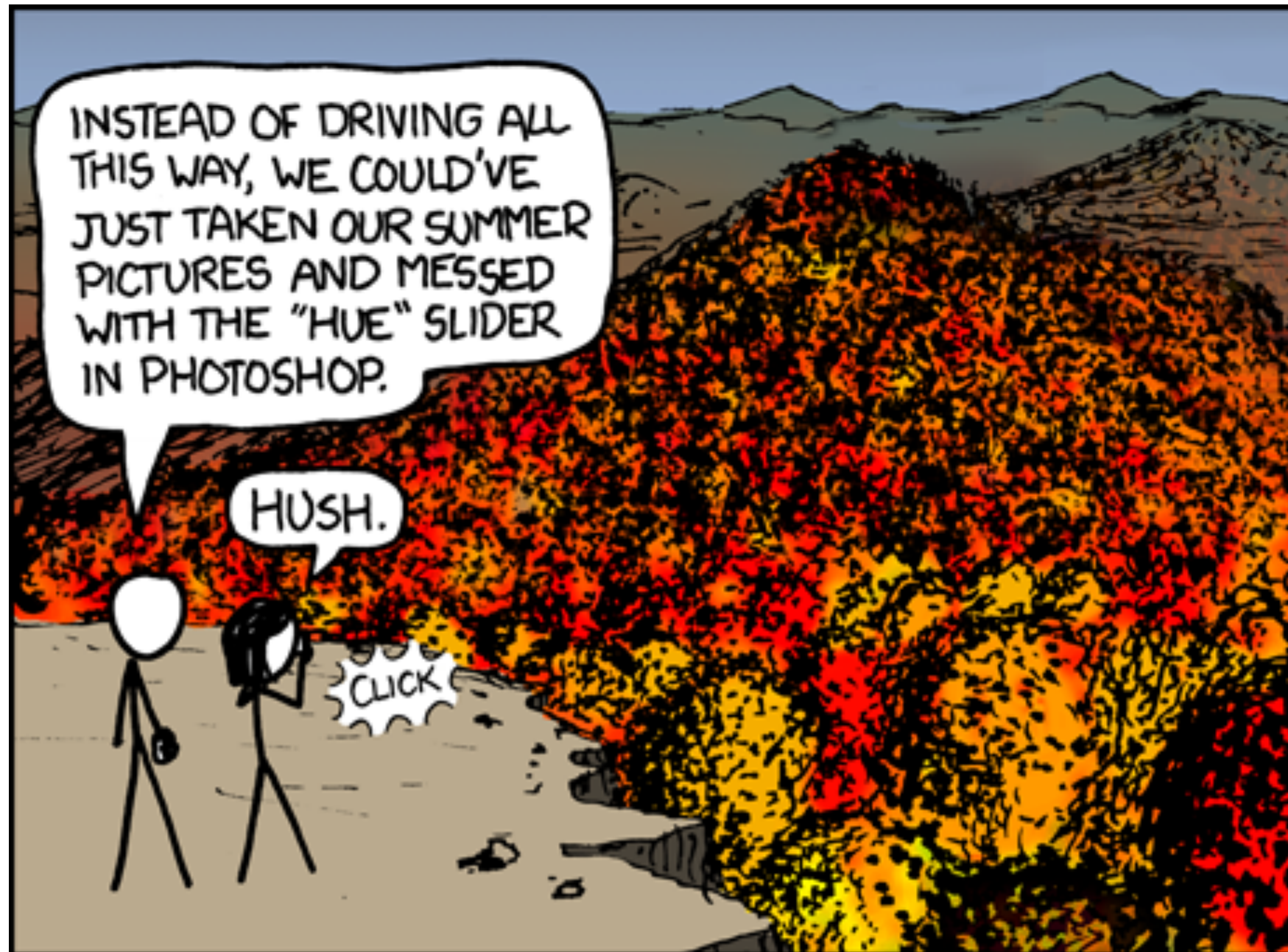
- http://www.robgalbraith.com/bins/content_page.asp?cid=7-11666-11979
- Gel on flash



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



Gamma



Relevant info is about ratios

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



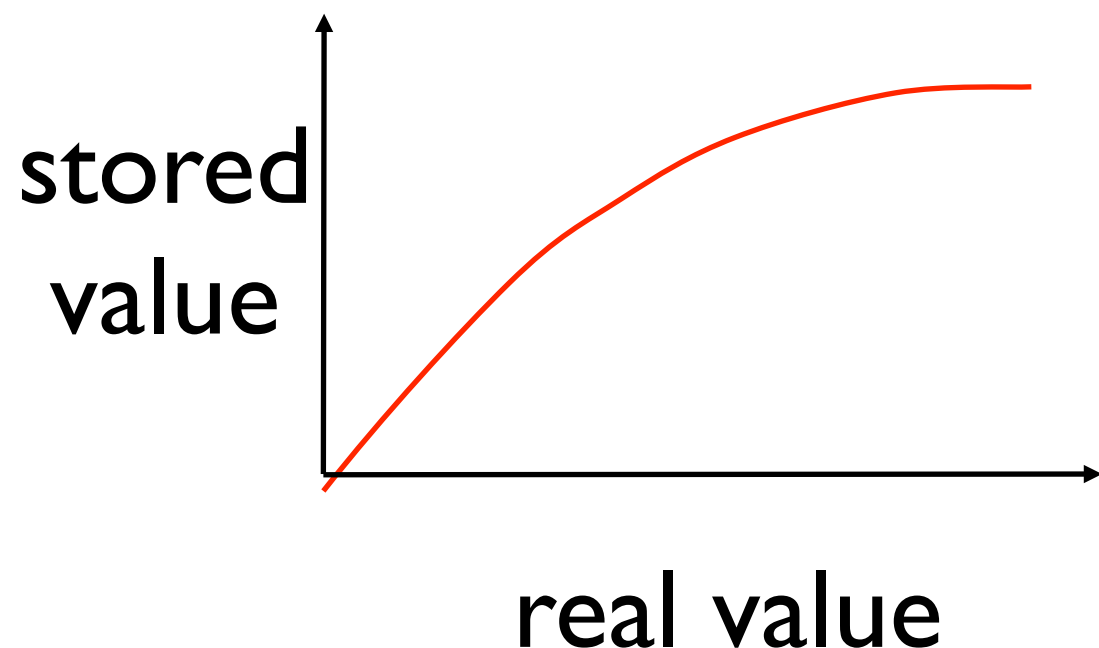
0.1 to 0.2



0.4 to 0.8

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 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 $x^{1/\gamma}$
- When displaying an image, the OS or display or driver inverts the mapping

Gamma demo

- <http://miki3.csail.mit.edu:3000/static/gammavis/>
 - thank you Lili & Charlotte & Monde

See effects of gamma encoding for picture of skippy:



Pick a value for γ : 2.2

0 7

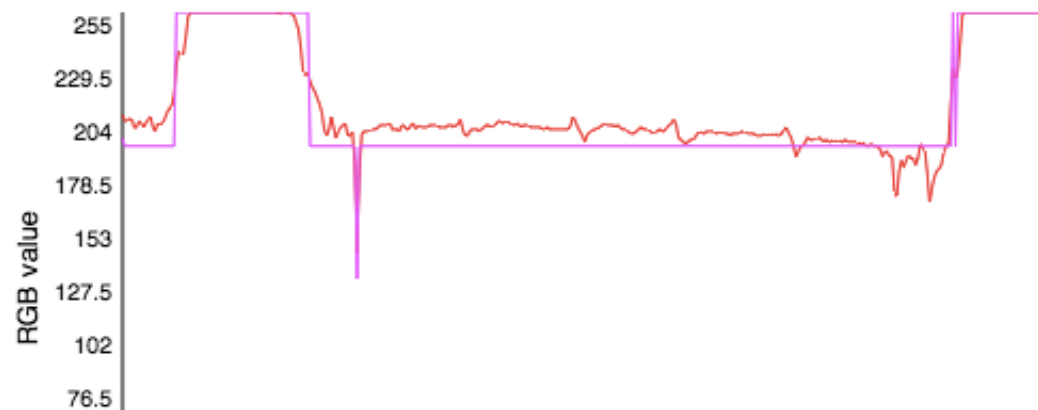
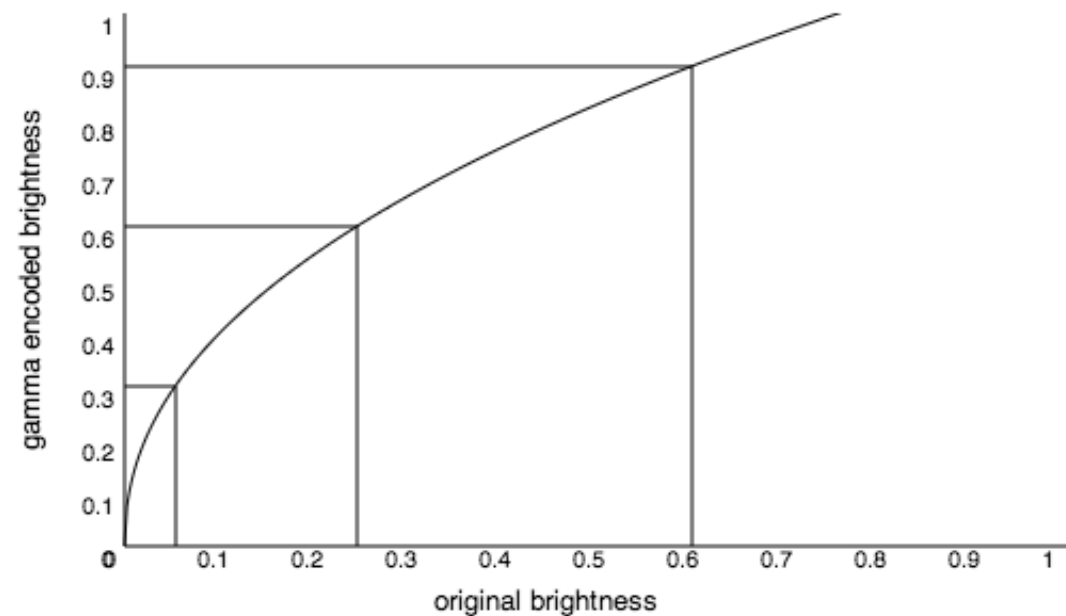
$\gamma=1$

$\gamma=2.2$

Pick how many bits: 2

0 8

Function plotted: $x^{(1/\gamma)}$



Other motivation: noise

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

Linearity and gamma

- Half (well, more like 25%) of image processing works better in linear space
 - If linearity is important
 - e.g. white balance, deblurring, HDR merging
- Half (well, 75%) work better in gamma space
 - closer to logarithmic scale, uniform noise, uniform treatment over brightness
 - e.g. edge detection, corner detection
- Sometimes you need log
 - when dynamic range is really high
 - e.g. tone mapping

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



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!

How to capture linear images

- <http://kimojohnson.org/blog/linear.html>



POINT

OPERATION



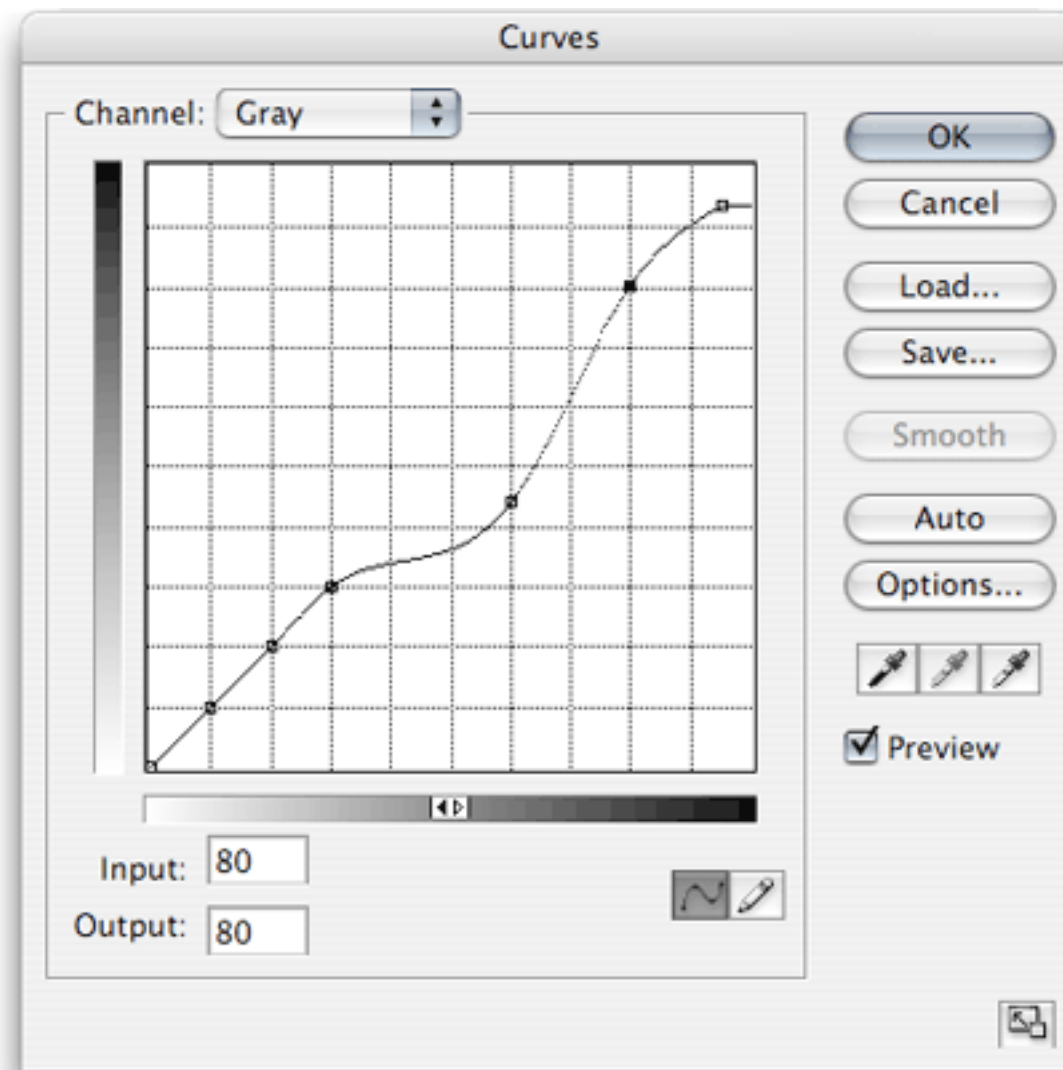
Point operations

- 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:
 - $L_{out} = f(L_{in})$

Curves

- In many photo editors

output



input

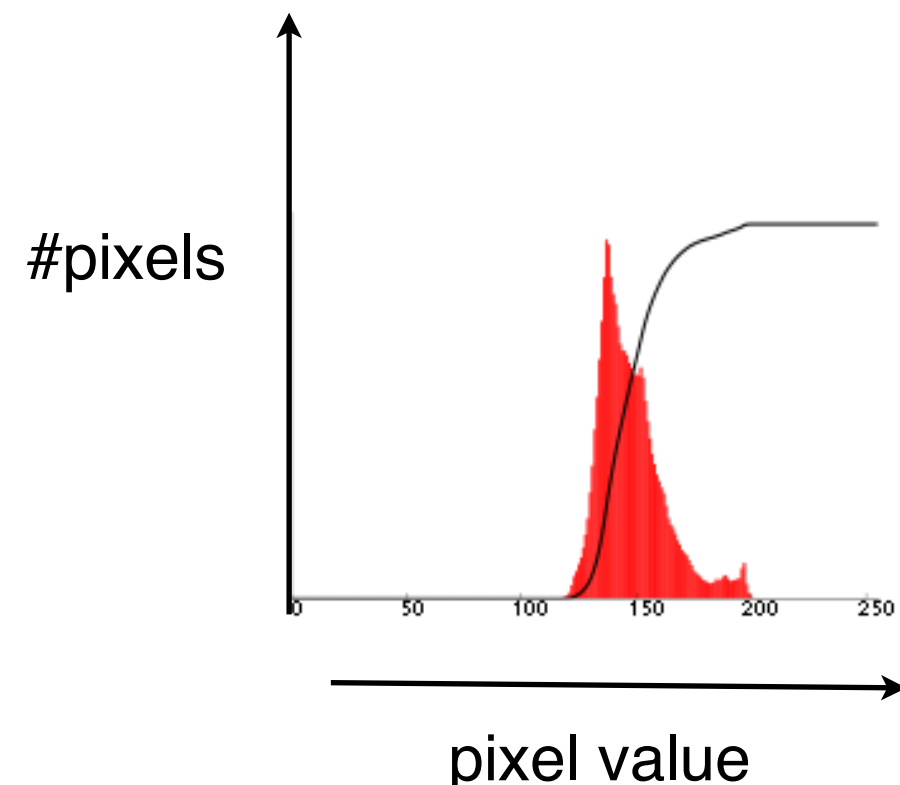
Pset 0 & 1

- 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

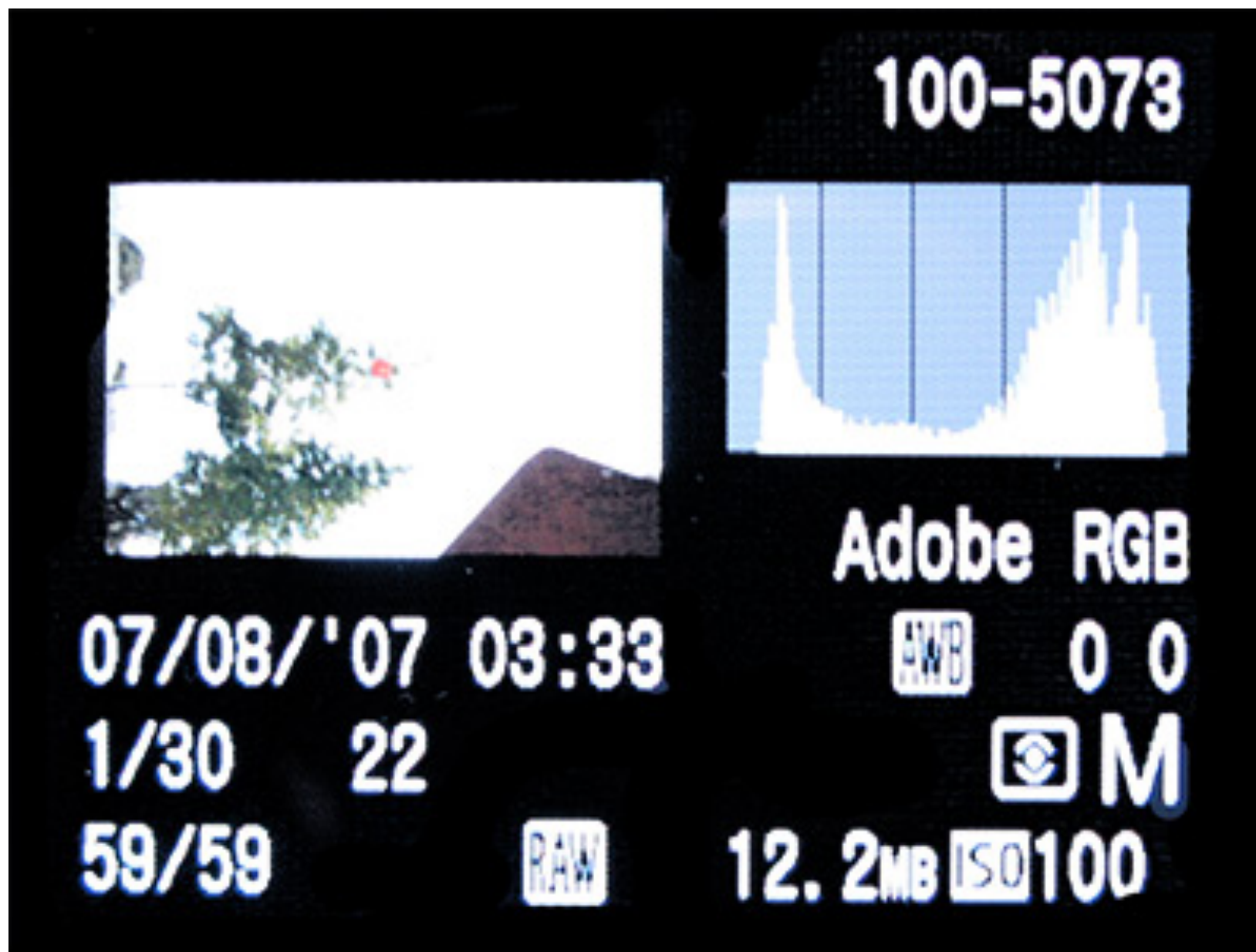
Histogram

- 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





Bad: bright values under-used
(underexposure)

Bad: bright values saturate
(overexposure)

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