

Digital Photography II

color & image processing pipeline



CSC2529

David Lindell

University of Toronto

cs.toronto.edu/~lindell/teaching/2529

*slides adapted from Gordon Wetzstein, Fredo Durand,
Ioannis Gkioulekas, Marc Levoy, Todd Zickler, Michael Brown

Announcements

- HW 2 is out (due next Wednesday 4/10)
- Instructor office hours today 4-5pm BA 7228
- TA Problem Session (HW2) Tues 1-2pm BA 5256

Outline

- Review
- Color
- Camera processing pipeline

Review – “Sensors are Buckets”

collect photons
like a bucket



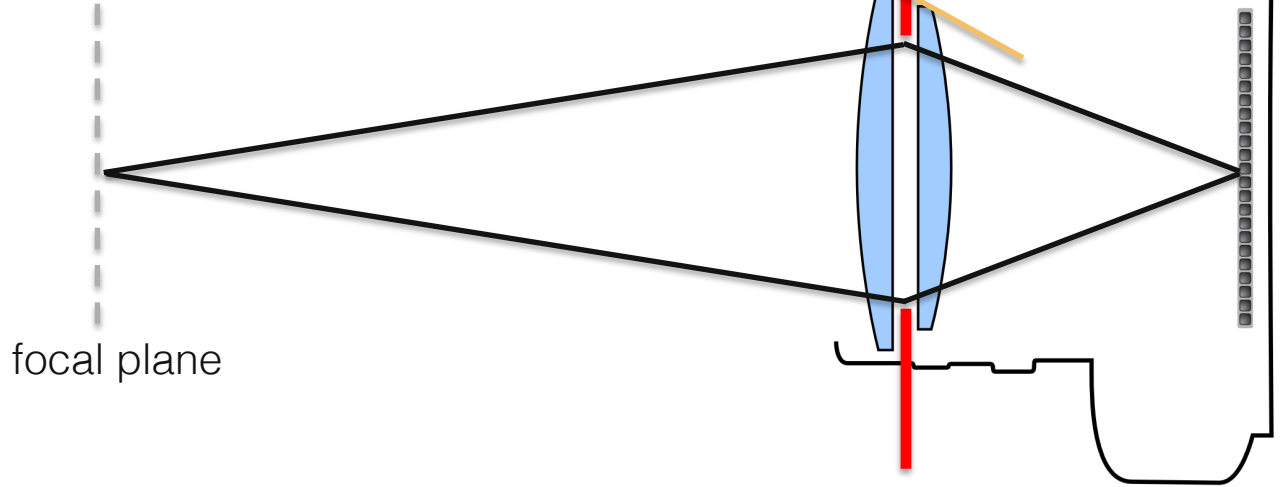
integrate spectrum



integrate incident
directions

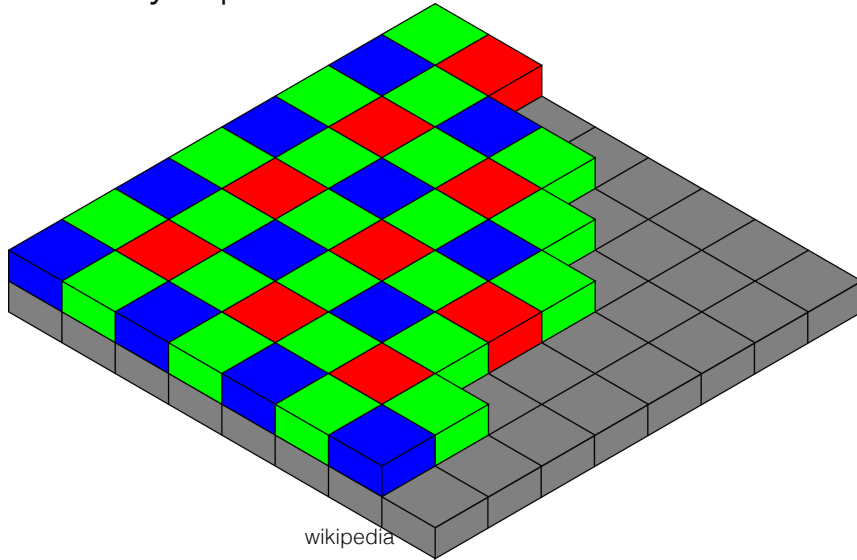


Each pixel sees a point
on the focal plane from
different perspectives!



Review – Color Filter Arrays

Bayer pattern



wikipedia

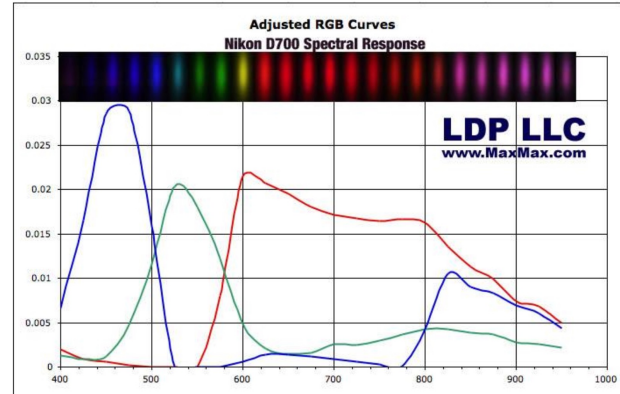
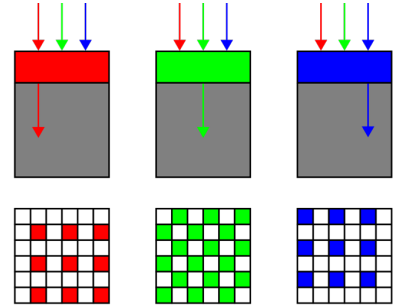


Image Formation

- high-dimensional integration over angle, wavelength, time

plenoptic function

$$i(x) \approx \iiint_{\Omega_{\theta, \lambda, t}} \overset{\downarrow}{l}(x, \theta, \lambda, t) d\theta d\lambda dt$$

plenoptic function:
[Adelson 1991]

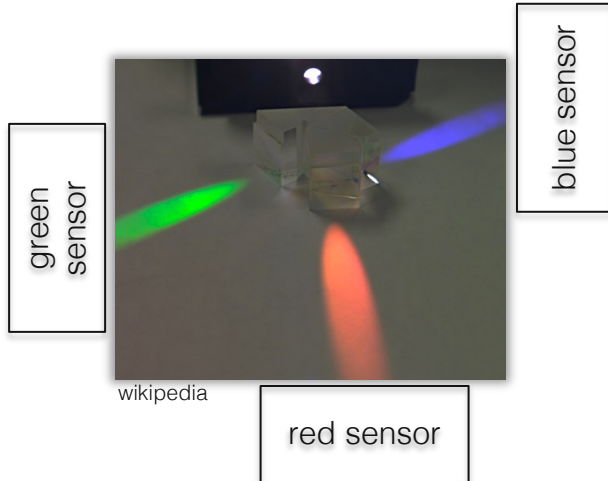
More Ways to Capture Color

field sequential

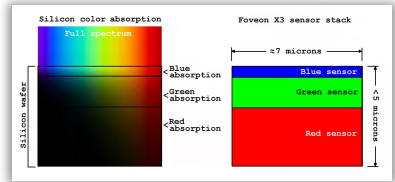
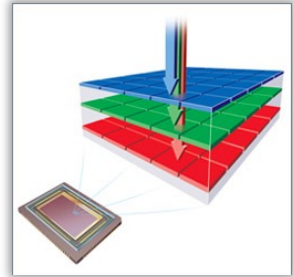


Prokudin-Gorsky

multiple sensors



vertically stacked



Foveon X3

More Ways to Capture Color



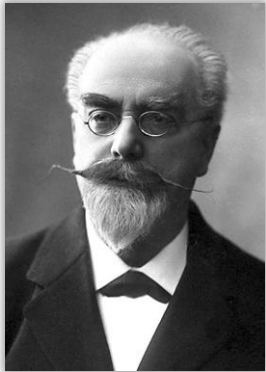
Sergei
Prokudin-Gorsky



Alim Khahn, Emir of Bukhara, 1911



More Ways to Capture Color



Gabriel Lippmann

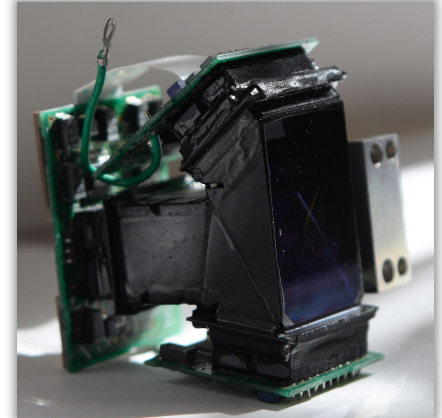
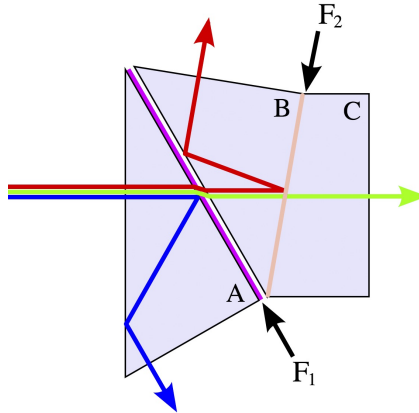
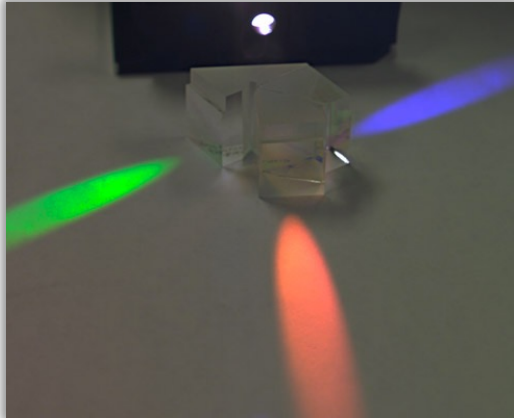
- notable French inventor
- Nobel price for color photography in 1908 = volume emulsion capturing interference
- today, this process is most similar to volume holography!
- also invented integral imaging (will hear more...)



Lippmann's
stuffed parrot

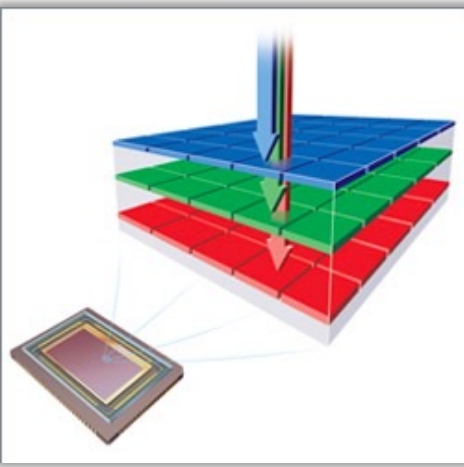
Three-CCD Camera

beam splitter prism



Philips / wikipedia

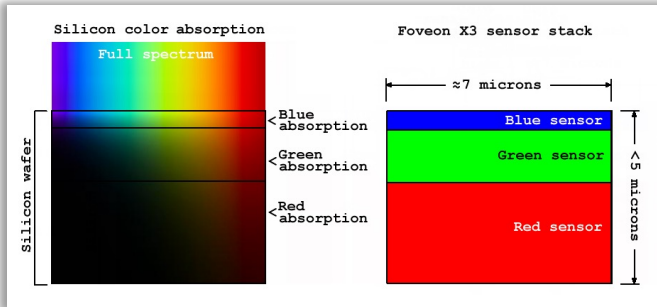
Stacked Sensor



Foveon X3

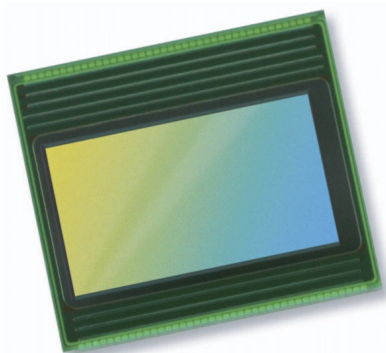


Sigma SD9




Other Wavelengths

- OmniVision:
RGB + near IR!



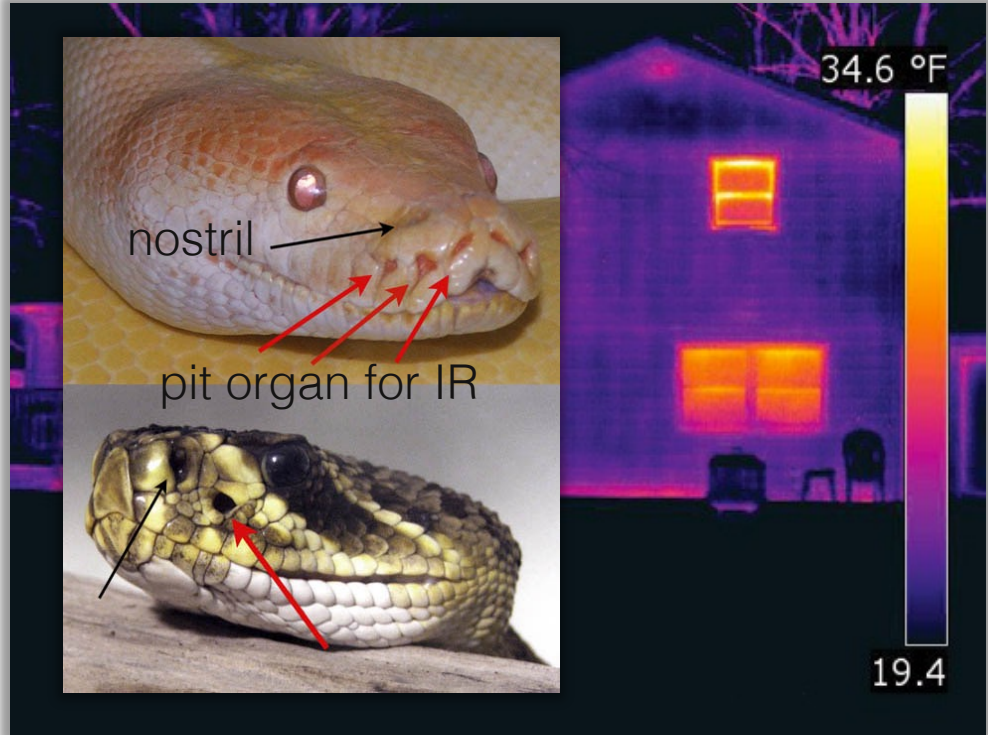
Product Specifications

Part Number	OV4682-G04A
Resolution	4MP
Chroma	Color
Analog / Digital	Digital
Power Requirement	Active: 163 mA (261 mW) Standby: 1 mA XSHUTDOWN: <10 μ A
Temperature Range	Operating: -30°C to +85°C junction temperature Stable image: 0°C to +60°C junction temperature
Output Format	10-bit RAW data
Optical Format	1/3"
Frame Rate	Full @ 90 fps 1080p @ 120 fps 672x380: 330 fps 720p @ 180 fps
Pixel Size	2.0 μ m
Image Area	5440 x 3072 μ m
Package	COB
Package Dimensions	6600 x 5800 μ m
Product Brief	 Product Brief

Other Wavelengths

FLIR Systems

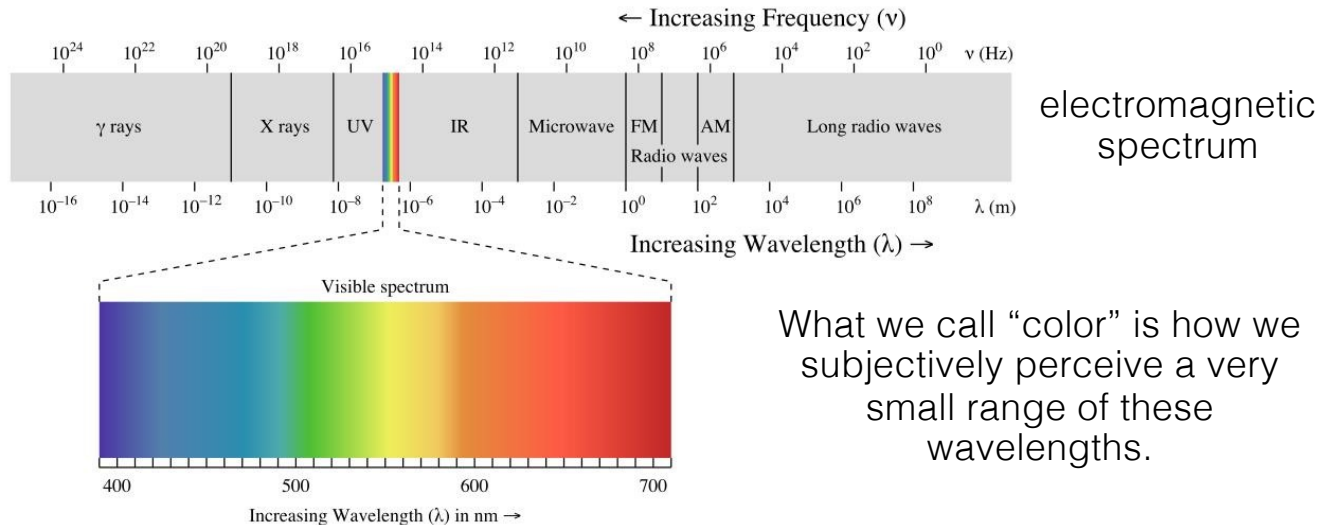
- thermal IR
- often use Germanium optics (transparent IR)



- sensors don't use silicon: indium, mercury, lead, etc.

Color is an artifact of human perception

- “Color” is not an objective physical property of light (electromagnetic radiation).
- Instead, light is characterized by its wavelength.



Spectral Sensitivity Function (SSF)

- Any light sensor (digital or not) has different sensitivity to different wavelengths.
- This is described by the sensor's spectral sensitivity function $f(\lambda)$
- When measuring light of some SPD $\Phi(\lambda)$, the sensor produces a scalar response:

$$\text{sensor response} \longrightarrow R = \int_{\lambda} \overset{\substack{\text{light SPD} \\ \downarrow}}{\Phi(\lambda)} \overset{\substack{\text{sensor SSF} \\ \downarrow}}{f(\lambda)} d\lambda$$

Weighted combination of light's SPD: light contributes more at wavelengths where the sensor has higher sensitivity.

Spectral Sensitivity Function of Human Eye

- The human eye is a collection of light sensors called cone cells.
- There are three types of cells with different spectral sensitivity functions.
- Human color perception is three-dimensional (tristimulus color).

“short”

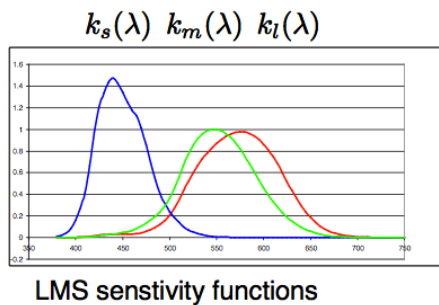
$$S = \int_{\lambda} \Phi(\lambda) S(\lambda) d\lambda$$

“medium”

$$M = \int_{\lambda} \Phi(\lambda) M(\lambda) d\lambda$$

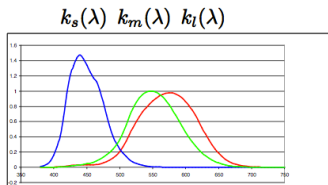
“long”

$$L = \int_{\lambda} \Phi(\lambda) L(\lambda) d\lambda$$

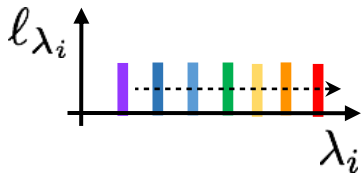
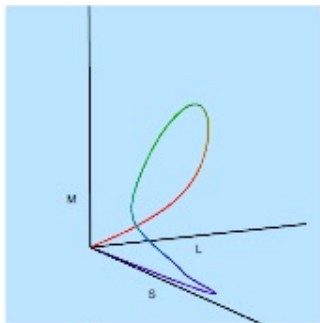


The retinal color space

$$\mathbf{c}(\ell_{\lambda_i}) = (c_s, c_m, c_l)$$



LMS sensitivity functions



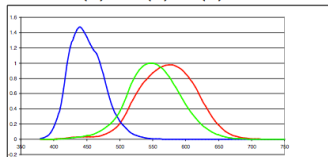
“pure beam” (laser)

The retinal color space

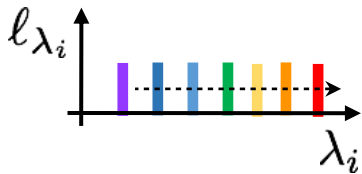
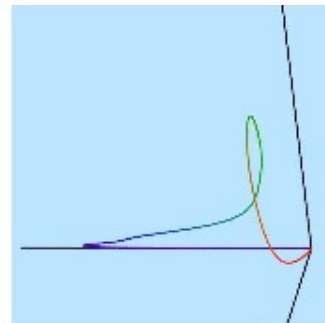
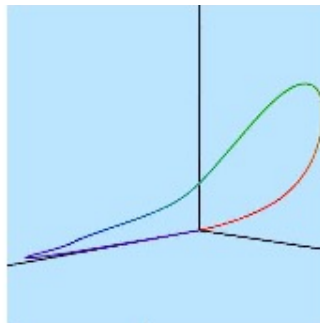
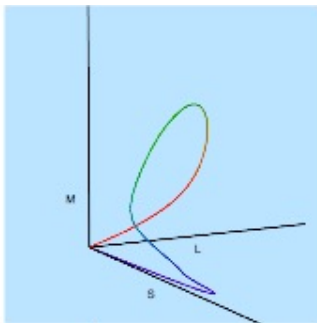
$$\mathbf{c}(\ell_{\lambda_i}) = (c_s, c_m, c_l)$$



$k_s(\lambda)$ $k_m(\lambda)$ $k_l(\lambda)$



LMS sensitivity functions



“pure beam” (laser)

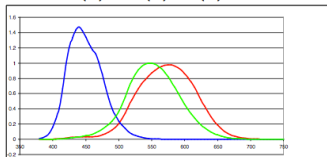
- “lasso curve”
 - contained in positive octant
 - parameterized by wavelength
 - starts and ends at origin
 - never comes close to M axis
- ← why?
- ← why?

The retinal color space

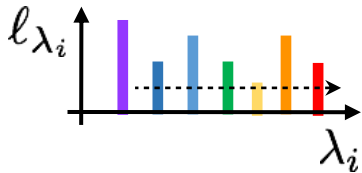
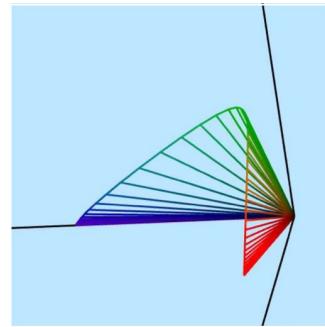
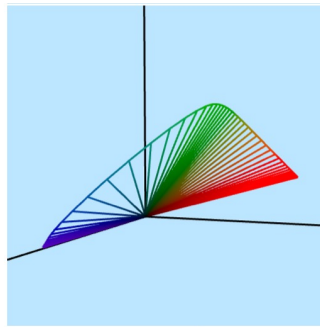
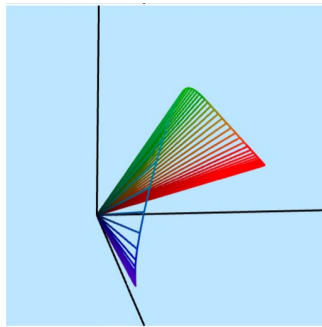
$$\mathbf{c}(\ell_{\lambda_i}) = (c_s, c_m, c_l)$$



$k_s(\lambda)$ $k_m(\lambda)$ $k_l(\lambda)$



LMS sensitivity functions



"pure beam" (laser)

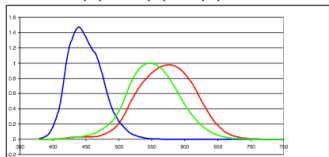
if we also consider variations in the strength of the laser this "lasso" turns into (convex!) radial cone with a "horse-shoe shaped" radial cross-section

The retinal color space

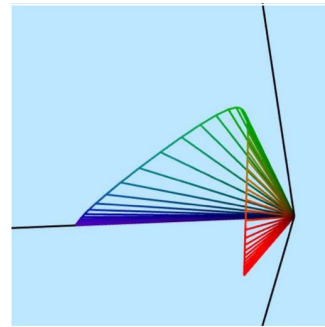
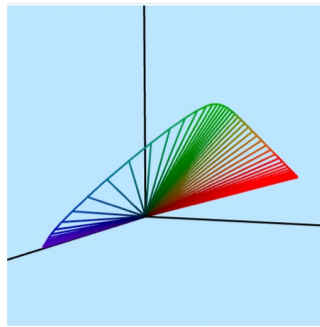
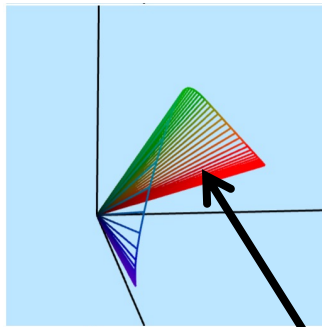
$$\mathbf{c}(\ell_{\lambda_i}) = (c_s, c_m, c_l)$$



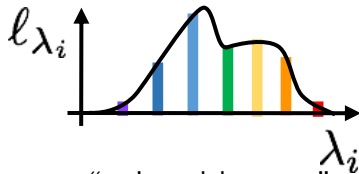
$k_s(\lambda)$ $k_m(\lambda)$ $k_l(\lambda)$



LMS sensitivity functions



colors of mixed beams are at the interior of the convex cone with boundary the surface produced by monochromatic lights



"mixed beam"

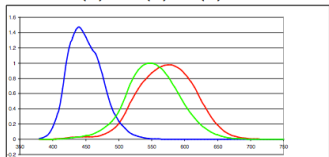
= convex combination of pure colors

The retinal color space

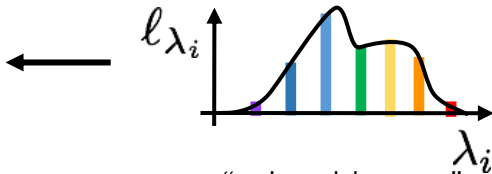
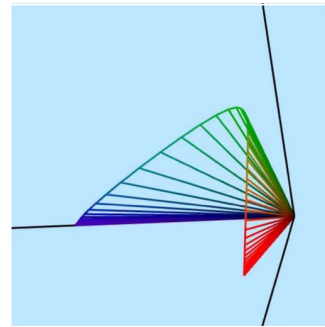
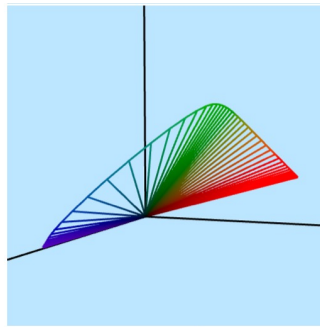
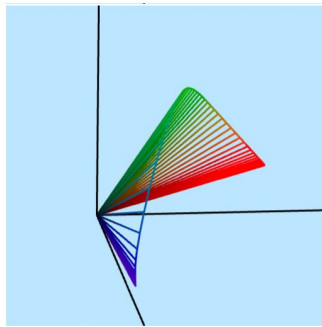
$$\mathbf{c}(\ell_{\lambda_i}) = (c_s, c_m, c_l)$$



$k_s(\lambda)$ $k_m(\lambda)$ $k_l(\lambda)$



LMS sensitivity functions

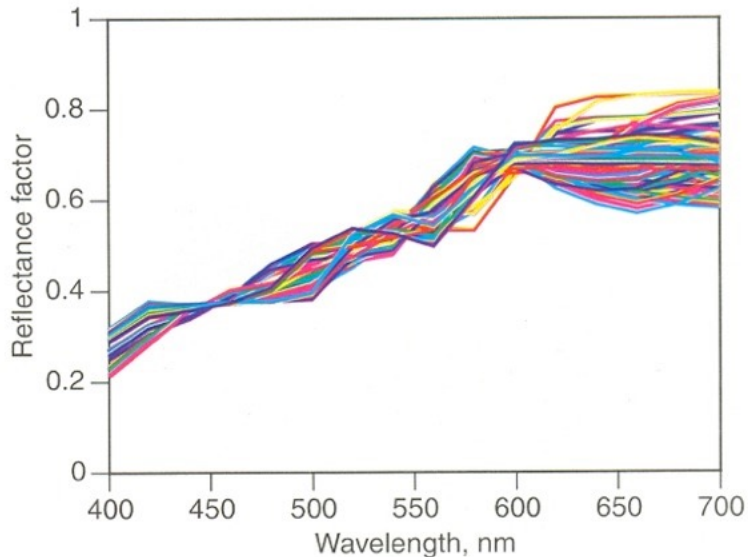


“mixed beam”

= convex combination of pure colors

- distinct mixed beams can produce the same retinal color
- these beams are called metamers

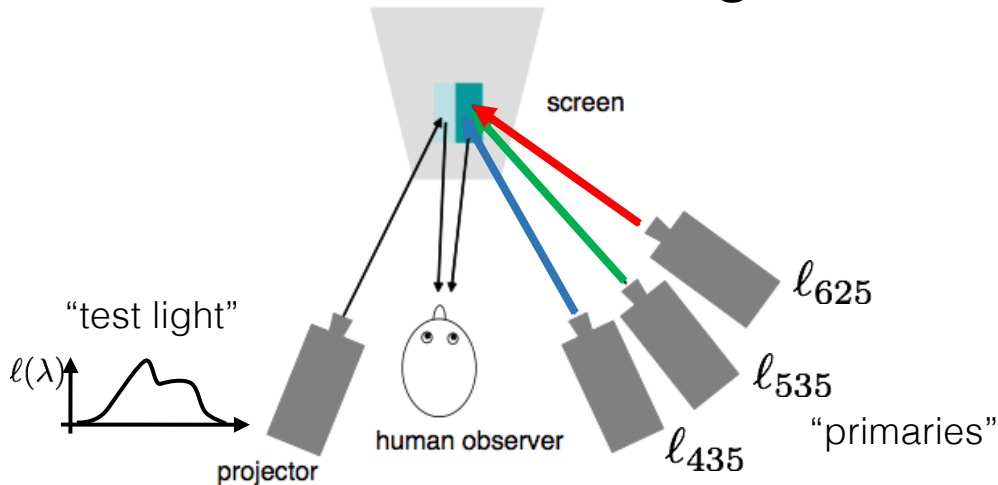
There is an infinity of metamers



Ensemble of spectral reflectance curves corresponding to three chromatic-pigment recipes all matching a tan material when viewed by an average observer under daylight illumination. [Based on Berns (1988b).]

Color matching

CIE color matching



Adjust the strengths of the primaries until they re-produce the test color.

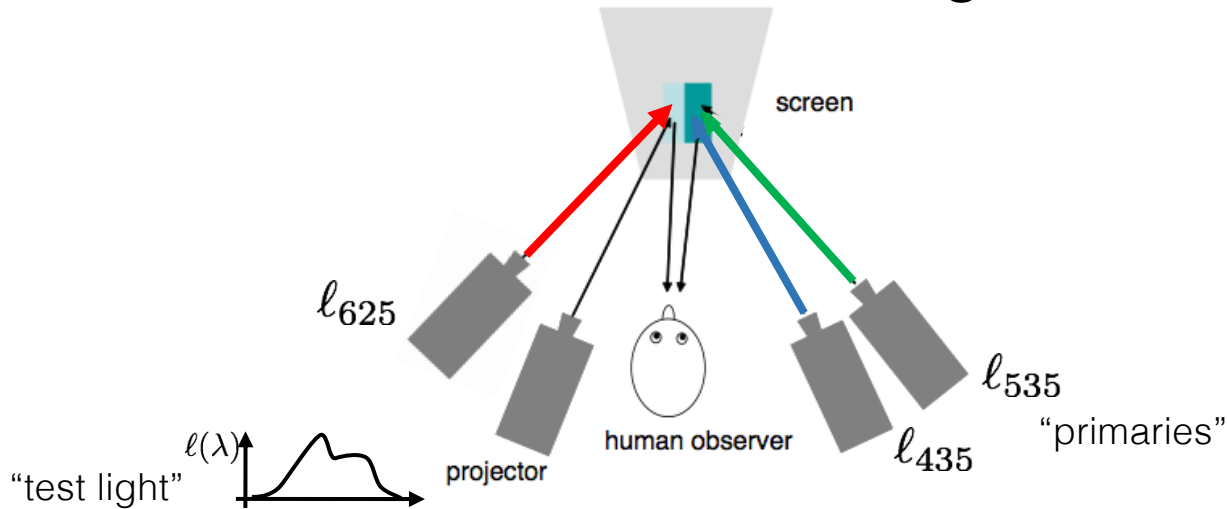
Then:

$$\mathbf{c}(\ell(\lambda)) = \alpha \mathbf{c}(\ell_{435}) + \beta \mathbf{c}(\ell_{535}) + \gamma \mathbf{c}(\ell_{625})$$



equality symbol means "has the same retinal color as" or "is metameric to"

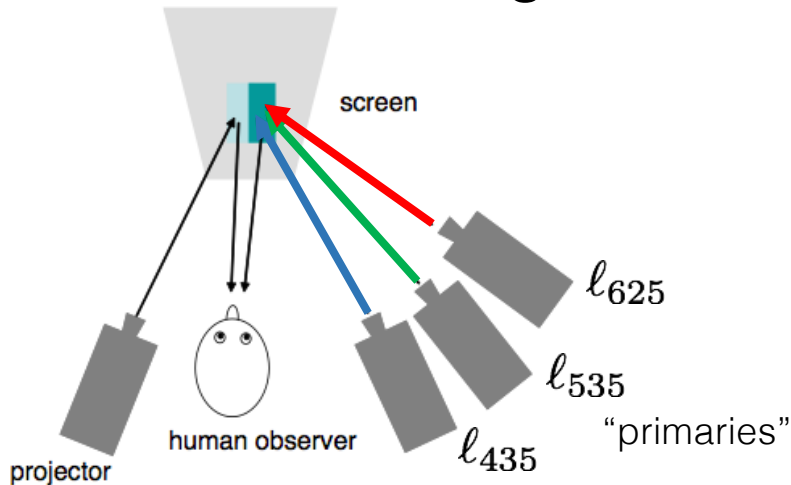
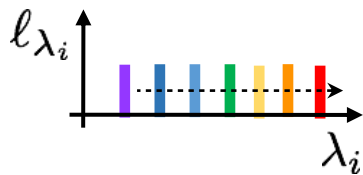
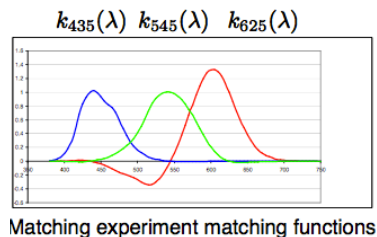
CIE color matching



To match some test colors, you need to add some primary beam on the left (same as “subtracting light” from the right)

$$\begin{aligned} \mathbf{c}(\ell(\lambda)) + \gamma \mathbf{c}(\ell_{625}) &= \alpha \mathbf{c}(\ell_{435}) + \beta \mathbf{c}(\ell_{535}) \\ \longrightarrow \mathbf{c}(\ell(\lambda)) &= \alpha \mathbf{c}(\ell_{435}) + \beta \mathbf{c}(\ell_{535}) - \gamma \mathbf{c}(\ell_{625}) \end{aligned}$$

CIE color matching

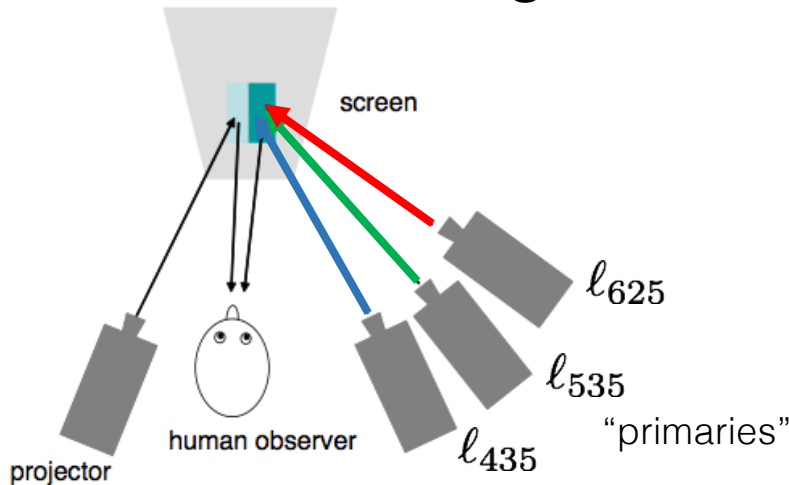
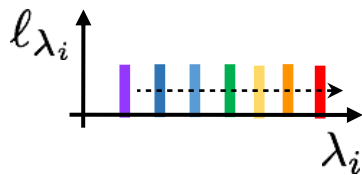
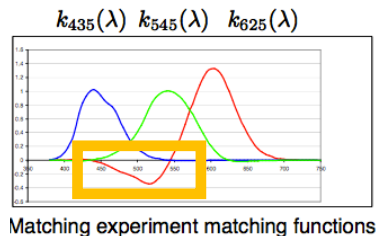


Repeat this matching experiments for pure test beams at wavelengths λ_i and keep track of the coefficients (negative or positive) required to reproduce each pure test beam.

$$\mathbf{c}(\lambda_i) = k_{435}(\lambda)\mathbf{c}(\ell_{435}) + k_{535}(\lambda)\mathbf{c}(\ell_{535}) + k_{625}(\lambda)\mathbf{c}(\ell_{625})$$

note the
negative values

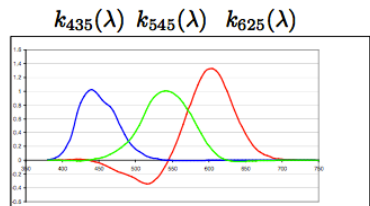
CIE color matching



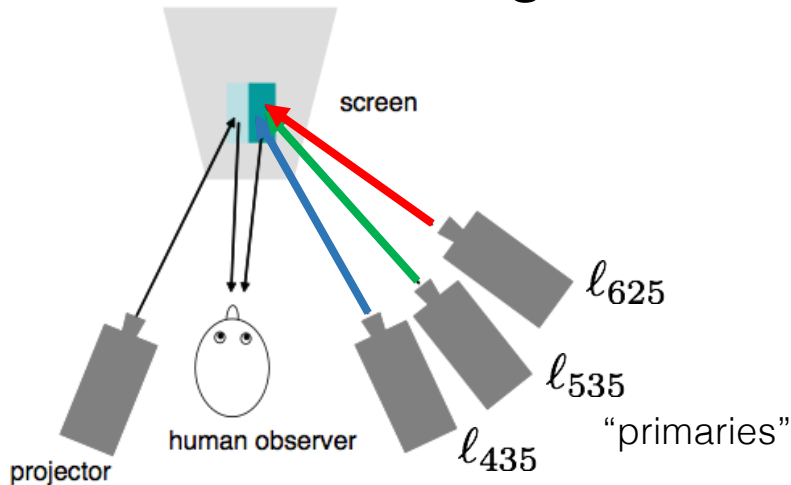
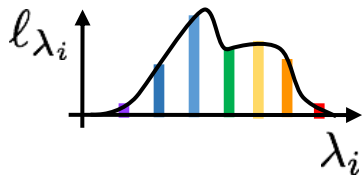
Repeat this matching experiments for pure test beams at wavelengths λ_i and keep track of the coefficients (negative or positive) required to reproduce each pure test beam.

$$\mathbf{c}(\lambda_i) = k_{435}(\lambda)\mathbf{c}(l_{435}) + k_{535}(\lambda)\mathbf{c}(l_{535}) + k_{625}(\lambda)\mathbf{c}(l_{625})$$

CIE color matching



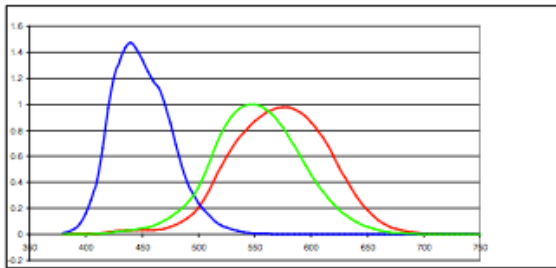
Matching experiment matching functions



What about "mixed beams"?

Two views of retinal color

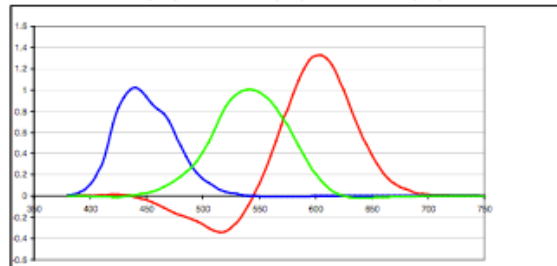
$k_s(\lambda)$ $k_m(\lambda)$ $k_l(\lambda)$



LMS sensitivity functions

Analytic: Retinal color is produced by analyzing spectral power distributions using the color sensitivity functions.

$k_{435}(\lambda)$ $k_{545}(\lambda)$ $k_{625}(\lambda)$

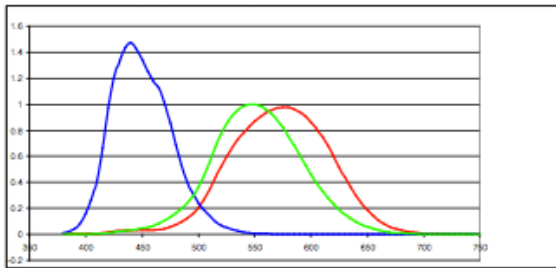


Matching experiment matching functions

Synthetic: Retinal color is produced by synthesizing color primaries using the color matching functions.

Two views of retinal color

$k_s(\lambda)$ $k_m(\lambda)$ $k_l(\lambda)$

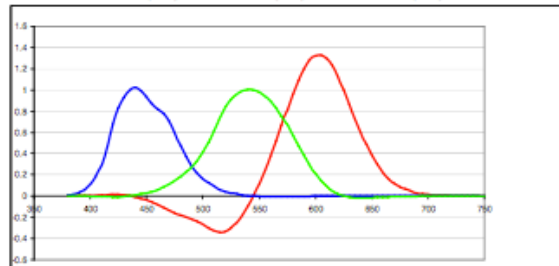


LMS sensitivity functions

Analytic: Retinal color is produced by analyzing spectral power distributions using the color sensitivity functions.

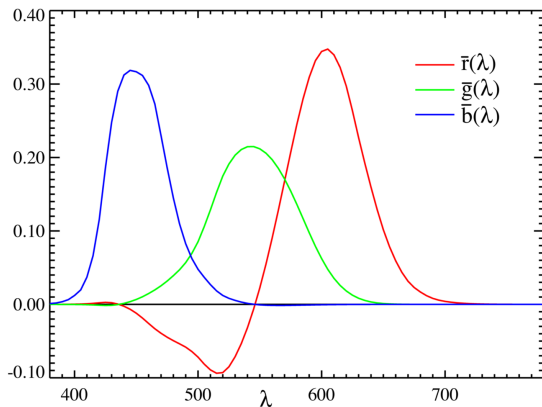
The two views are equivalent: Color matching functions are also color sensitivity functions. For each set of color sensitivity functions, there are corresponding color primaries.

$k_{435}(\lambda)$ $k_{545}(\lambda)$ $k_{625}(\lambda)$



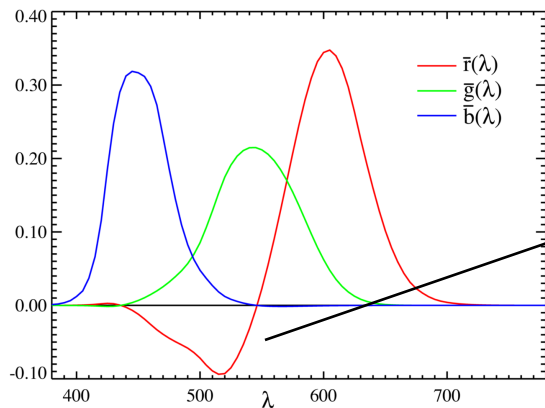
Matching experiment matching functions

Synthetic: Retinal color is produced by synthesizing color primaries using the color matching functions.



CIE RGB colorspace

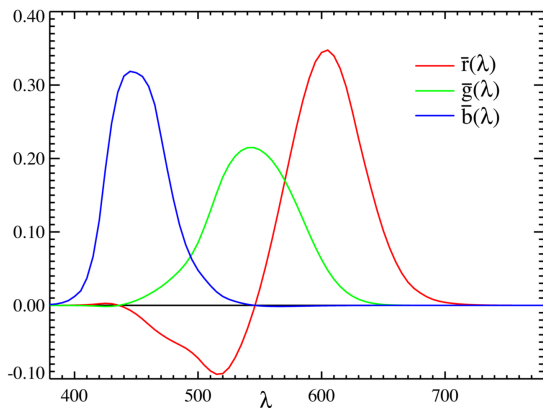
Created by the International Commission on Illumination in 1931 based on color matching experiments from 12 people!



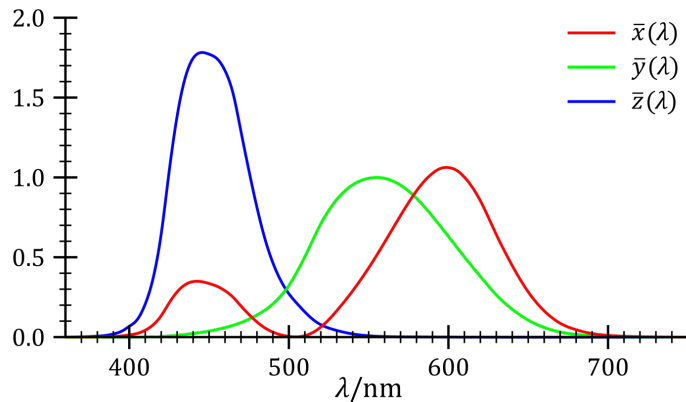
Negative values are not physical
since we cannot subtract light

CIE RGB colorspace

Created by the International Commission on
Illumination in 1931 based on color matching
experiments from 12 people!



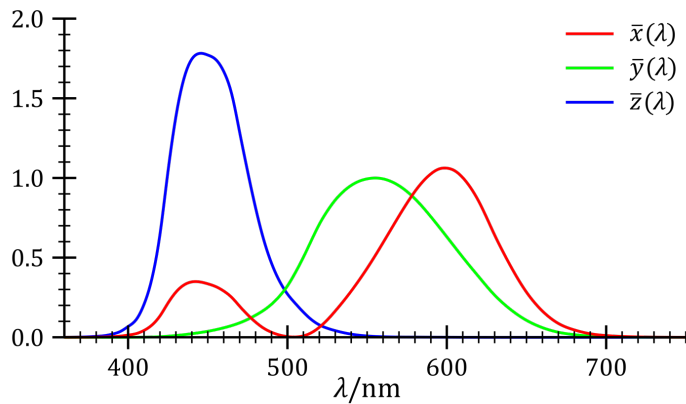
CIE RGB colorspace



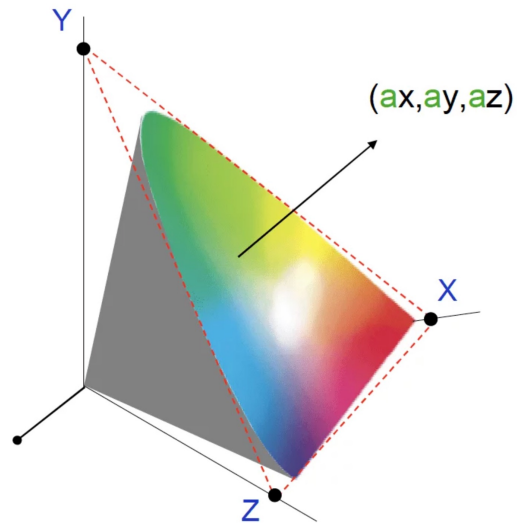
CIE XYZ colorspace

Created by the International Commission on Illumination in 1931 based on color matching experiments from 12 people!

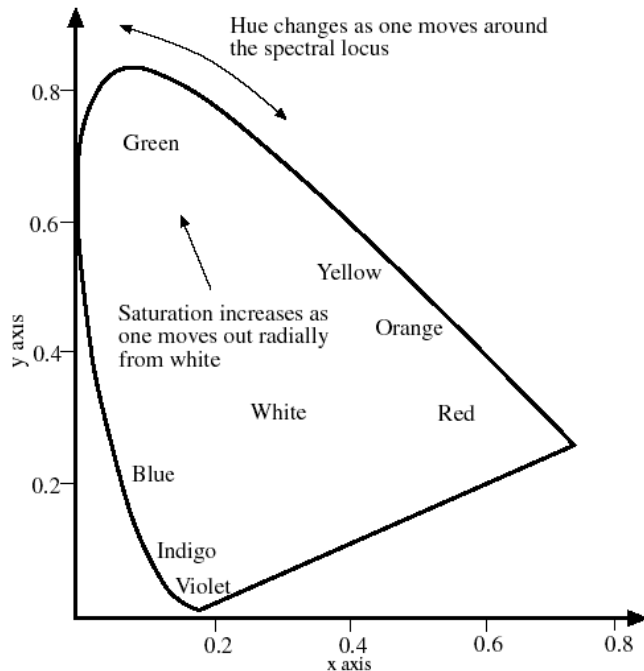
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



CIE XYZ colorspace



CIE xy (chromaticity)



$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

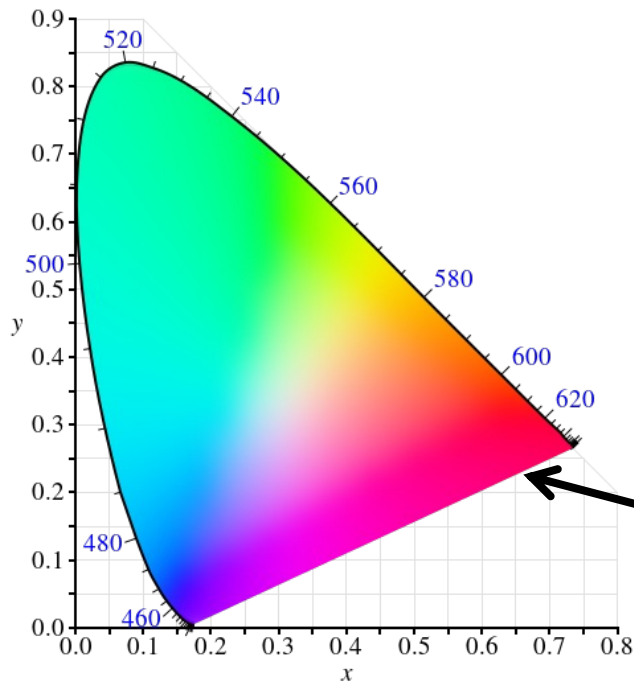
$$(X, Y, Z) \longleftrightarrow (\underline{x, y}, Y)$$

chromaticity

luminance/brightness

Perspective projection of 3D retinal color space to two dimensions.

CIE xy (chromaticity)



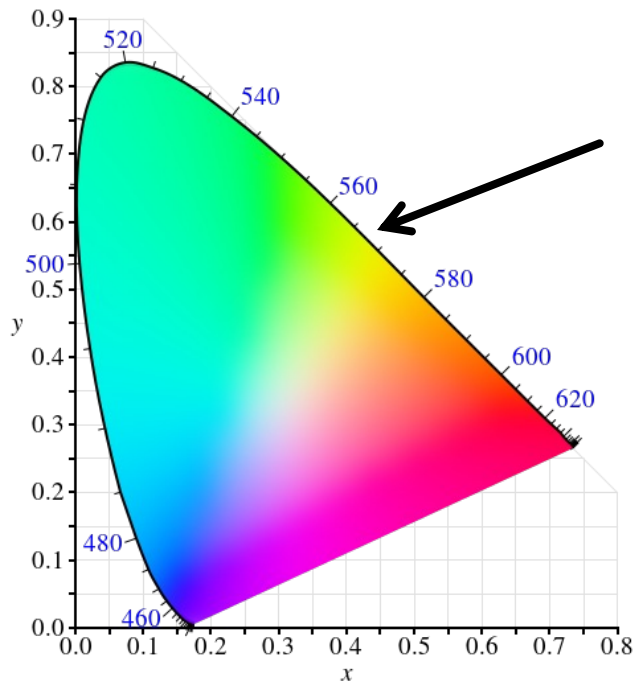
$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$(X, Y, Z) \longleftrightarrow (x, y, Y)$$

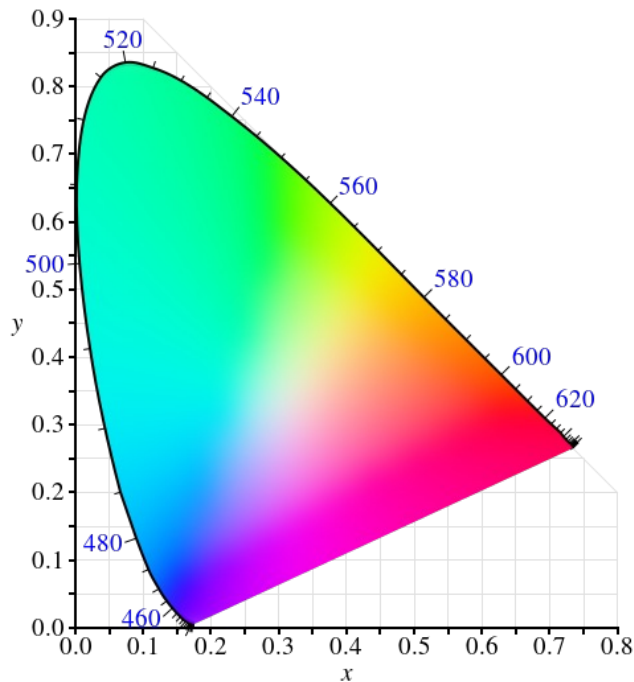
Note: These colors can be extremely misleading depending on the file origin and the display you are using

CIE xy (chromaticity)



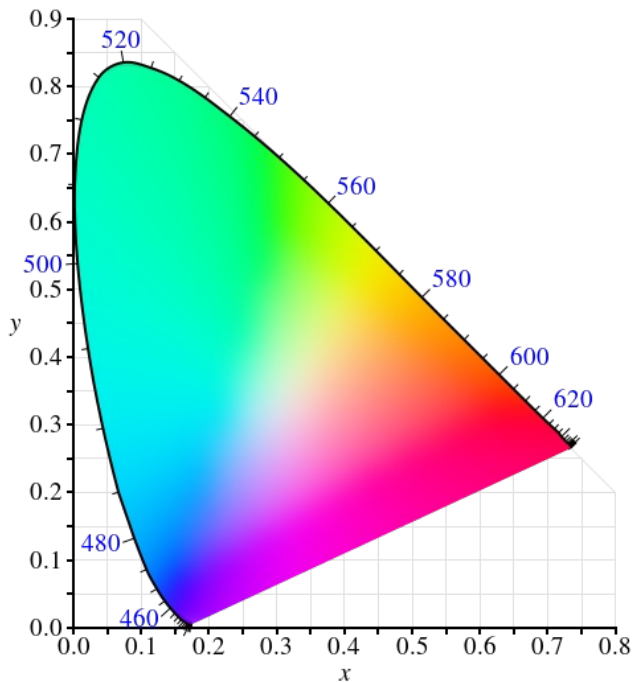
What does the boundary of the chromaticity diagram correspond to?

Color gamuts



We can compare color spaces by looking at what parts of the chromaticity space they can reproduce with their primaries. But why would a color space not be able to reproduce all of the chromaticity space?

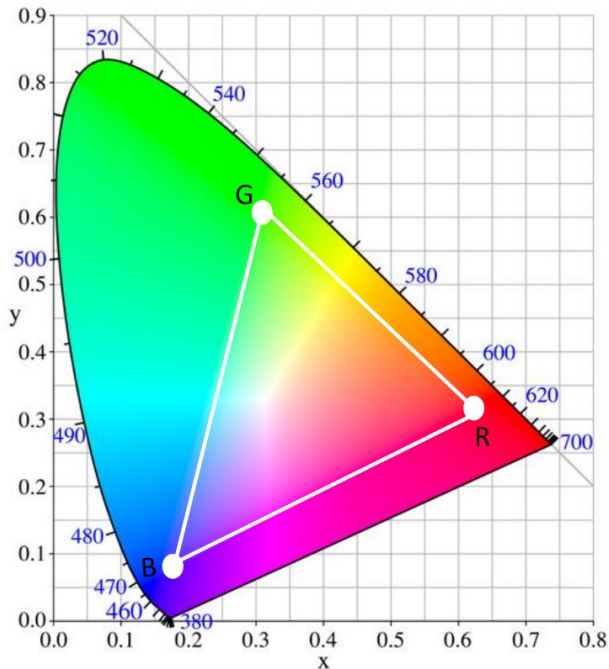
Color gamuts



We can compare color spaces by looking at what parts of the chromaticity space they can reproduce with their primaries. But why would a color space not be able to reproduce all of the chromaticity space?

- Many colors require negative weights to be reproduced, which are not realizable.

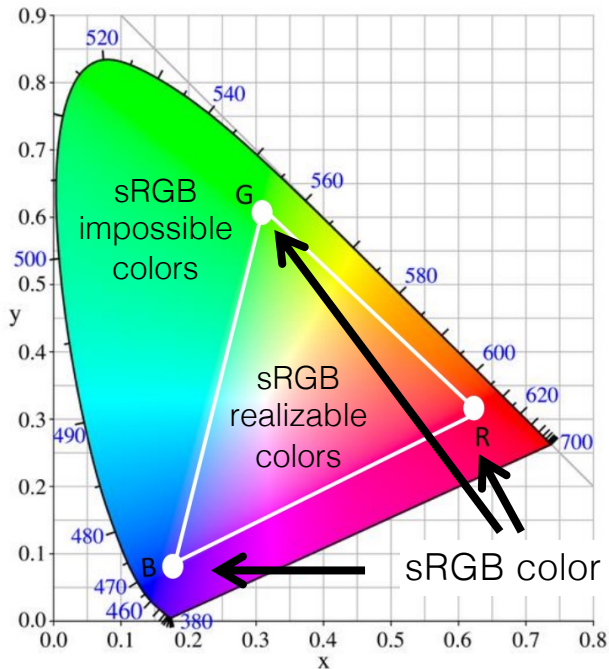
Color gamuts



sRGB color gamut:

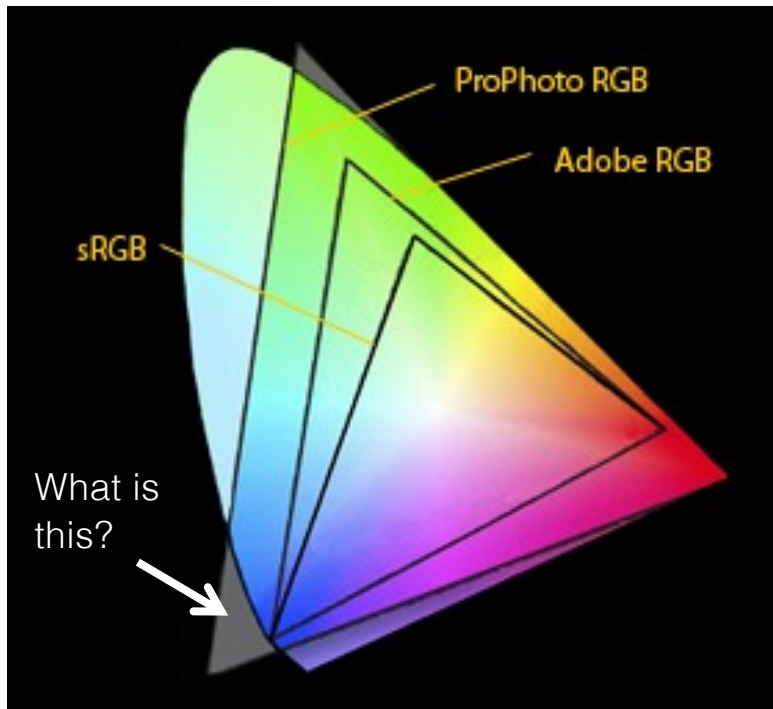
- What are the three triangle corners?
- What is the interior of the triangle?
- What is the exterior of the triangle?

Color gamuts



sRGB color gamut

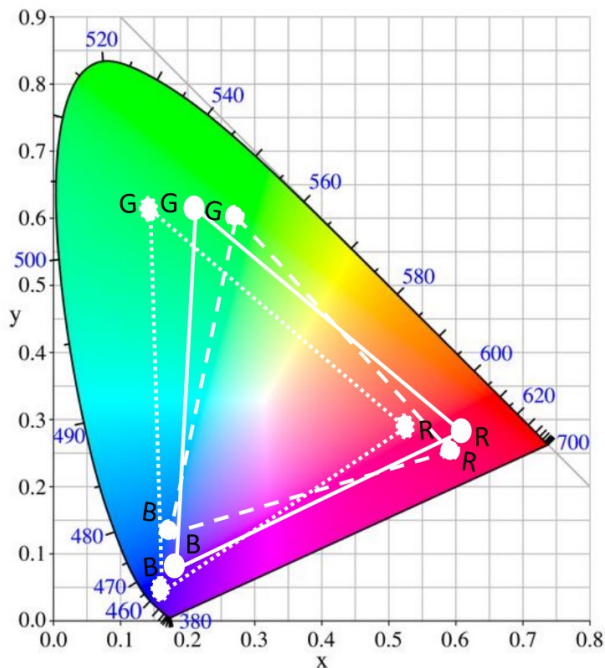
Color gamuts



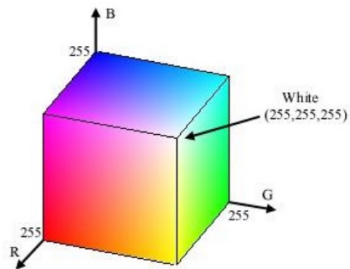
Gamuts of various common industrial RGB spaces

The problem with RGBs visualized in chromaticity space

45

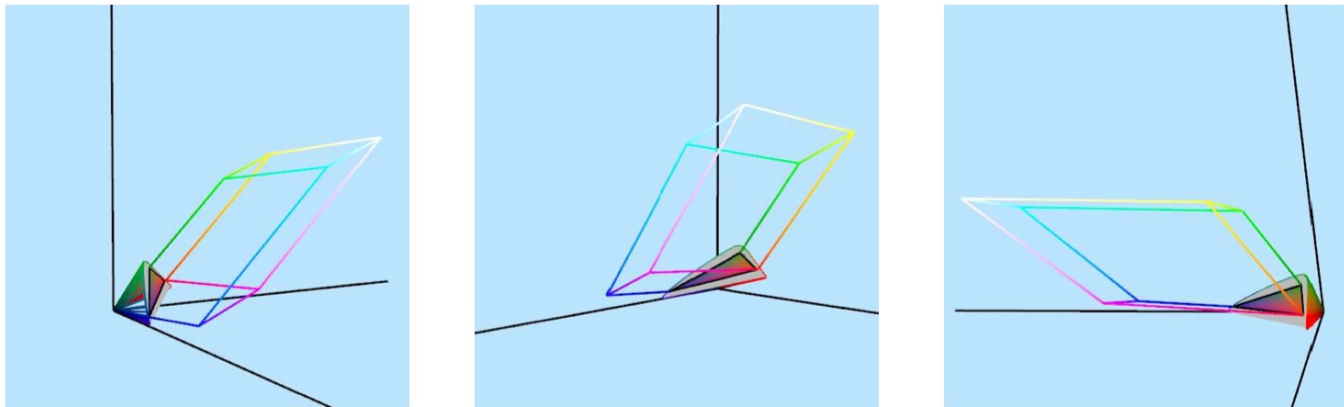


Device 1 —
Device 2
Device 3 - -



RGB values have no meaning if the primaries between devices are not the same!

Chromaticity diagrams can be misleading



Different gamuts may compare very differently when seen in full 3D retinal color space.

Some take-home messages about color spaces

Analytic: Retinal color is three numbers formed by taking the dot product of a power spectral distribution with three color matching/sensitivity functions.

Synthetic: Retinal color is three numbers formed by assigning weights to three color primaries to match the perception of a power spectral distribution.

Some take-home messages about color spaces

Fundamental problem: Analysis spectrum (camera, eyes) cannot be the same as synthesis one (display) - impossible to encode all possible colors without something becoming negative

- CIE XYZ only needs positive coordinates, but need primaries with negative light.
- RGB must use physical (non-negative) primaries, but needs negative coordinates for some colors.

Some take-home messages about color spaces

Problem with current practice: Many different RGB color spaces used by different devices, without clarity of what exactly space a set of RGB color values are in.

- Huge problem for color reproduction from one device to another.
- there are standards (like sRGB), but consumer displays are not calibrated—so you cannot really conclude that images are color accurate

camera processing pipeline

Review: Photons to RAW Image

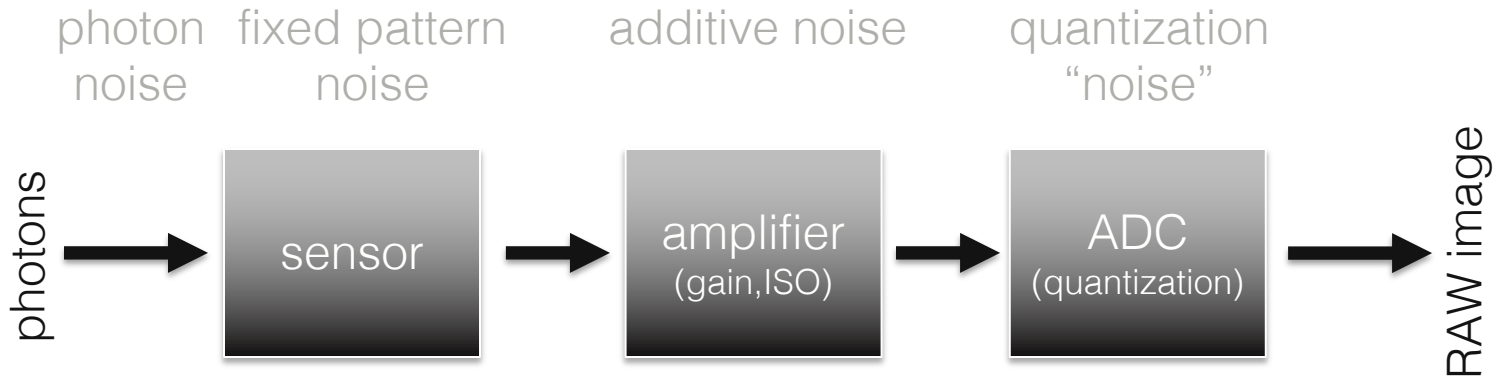
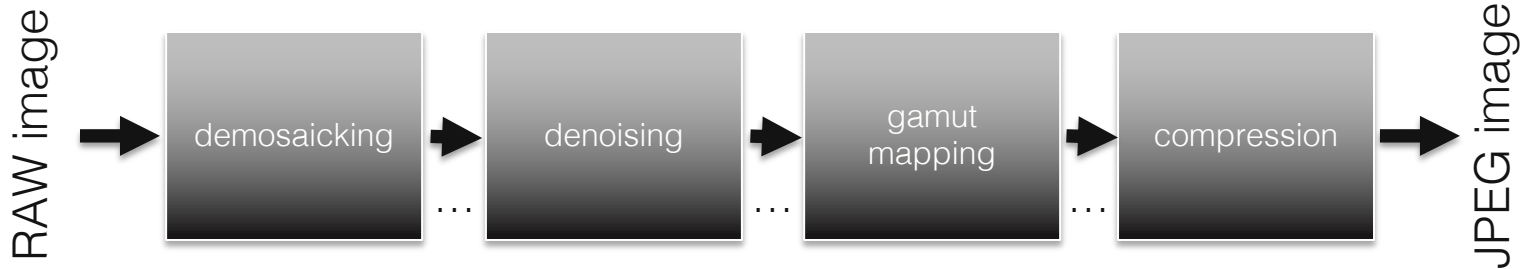


Image Processing Pipeline



also:

- dead pixel removal
- dark frame subtraction (fixed pattern / thermal noise removal)
- lens blur / vignetting / distortion correction
- sharpening / edge enhancement

Image Processing Pipeline

RAW image
(dcrw -D)



JPEG image

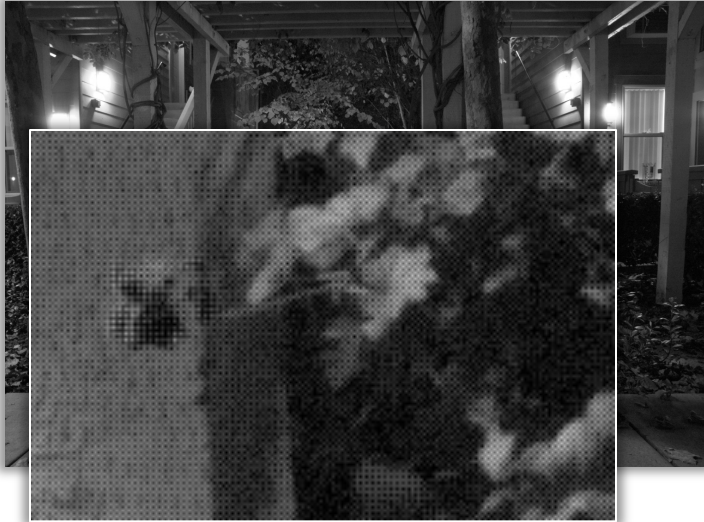


Image Processing Pipeline

- demosaicking
- denoising
- digital autoexposure
- white balancing
- linear 10/12 bit to 8 bit gamma
- compression



Image Processing Pipeline

Example pipeline

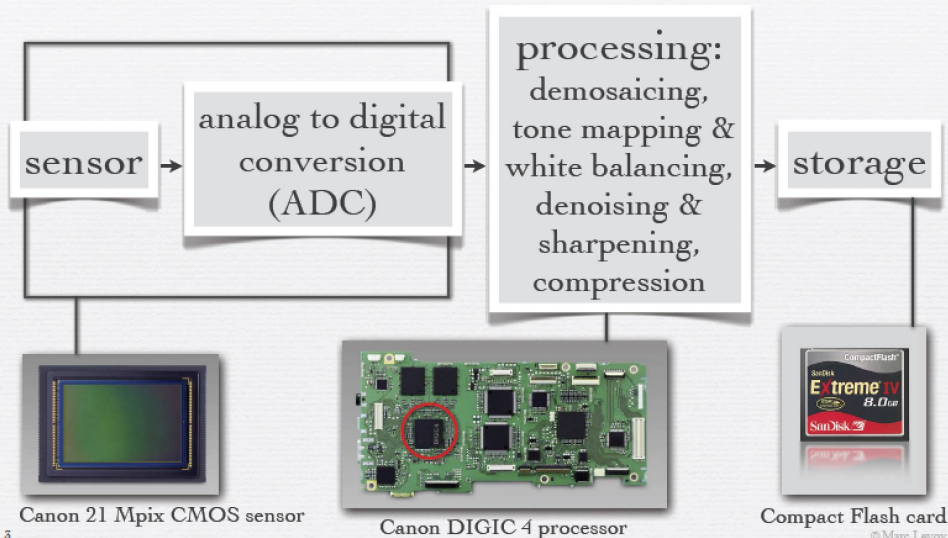
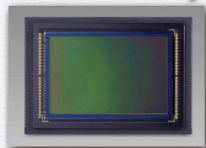
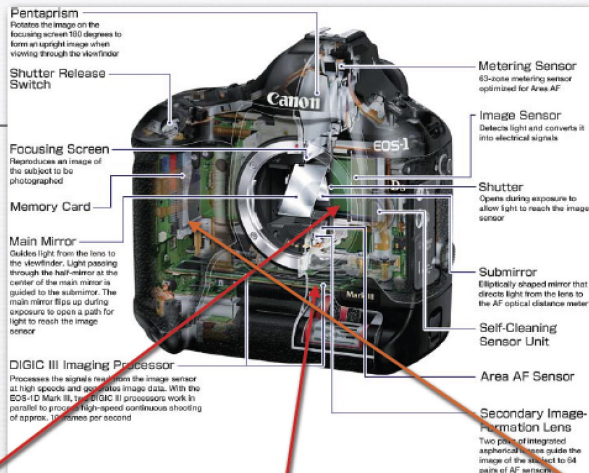


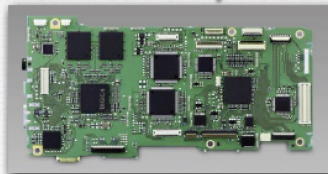
Image Processing Pipeline

Example

(parts are from a Canon 5DII, but cutaway view is of 1DIII)



Canon 21 Mpix CMOS sensor



Canon DIGIC 4 processor



Compact Flash card

© Marc Levoy

Filename - night nikon.JPG

Make - NIKON CORPORATION

Model - NIKON D780

Orientation - Top left

XResolution - 300

YResolution - 300

ResolutionUnit - Inch

Software - Ver.1.00

DateTime - 2005:09:01 12:16:43

YCbCrPositioning - Co-Sited

ExifOffset - 216

ExposureTime - 10 seconds

FNumber - 13.00

ExposureProgram - Manual control

ExifVersion - 0221

DateTimeOriginal - 2005:09:01 12:16:43

DateTimeDigitized - 2005:09:01 12:16:43

ComponentsConfiguration - YCbCr

CompressedBitsPerPixel - 1 (bits/pixel)

ExposureBiasValue - 0.50

MaxApertureValue - F 3.48

MeteringMode - Center weighted average

LightSource - Auto

Flash - Not fired

FocalLength - 18.00 mm

UserComment - (c) Gordon Wetzstein

SubsecTime - 00

SubsecTimeOriginal - 00

SubsecTimeDigitized - 00

FlashPixVersion - 0100

ColorSpace - sRGB

ExifImageWidth - 3008

ExifImageHeight - 2000

InteroperabilityOffset - 29230

SensingMethod - One-chip color area sensor

FileSource - Other

SceneType - Other

CustomRendered - Custom process

ExposureMode - Manual

White Balance - Auto

DigitalZoomRatio - 1 x

FocalLengthIn35mmFilm - 27 mm

SceneCaptureType - Portrait

GainControl - Low gain up

Contrast - Normal

Saturation - Normal

Sharpness - Soft

SubjectDistanceRange - Unknown

Maker Note (Vendor): -

Data version - 0210 (808595760)

ISO Setting - 1600

Image Quality - BASIC

White Balance - AUTO

Image Sharpening - MED.L

Focus Mode - MANUAL

Flash Setting - NORMAL

Flash Mode -

White Balance Adjustment - 0

Exposure Adjustment - 1.7

Thumbnail IFD offset - 1430

Flash Compensation - 67072

ISO 2 - 1600

Tone Compensation - AUTO

Lens type - AF-D G

Lens - 618

Flash Used - Not fired

AF Focus Position - Center

Bracketing - 131072

Color Mode - MODE1a

Light Type - NORMAL

Hue Adjustment - 0

Noise Reduction - FPNR

Total pictures - 22346

Optimization - PORTRAIT

Thumbnail: -

Compression - 6 (JPG)

XResolution - 300

YResolution - 300

ResolutionUnit - Inch

JpegIFOffset - 29368

JpegIFByteCount - 8393

YCbCrPositioning - Co-Sited

Thumbnail: -

Compression - 6 (JPG)

XResolution - 300

YResolution - 300

ResolutionUnit - Inch

JpegIFOffset - 29368

JpegIFByteCount - 8393

YCbCrPositioning - Co-Sited



Exif Meta Data

Filename - night nikon.JPG

Make - NIKON CORPORATION

Model - NIKON D780

Orientation - Top left

XResolution - 300

YResolution - 300

ResolutionUnit - Inch

Software - Ver.1.00

DateTime - 2005:09:01 12:16:43

YCbCrPositioning - Co-Sited

ExifOffset - 216

ExposureTime - 10 seconds

FNumber - 13.00

ExposureProgram - Manual control

ExifVersion - 0221

DateTimeOriginal - 2005:09:01 12:16:43

DateTimeDigitized - 2005:09:01 12:16:43

ComponentsConfiguration - YCbCr

CompressedBitsPerPixel - 1 (bits/pixel)

ExposureBiasValue - 0.50

MaxApertureValue - F 3.48

MeteringMode - Center weighted average

LightSource - Auto

Flash - Not fired

FocalLength - 18.00 mm

UserComment - (c) Gordon Wetzstein

SubsecTime - 00

SubsecTimeOriginal - 00

SubsecTimeDigitized - 00

FlashPixVersion - 0100

ColorSpace - sRGB

Maker Note (Vendor): -
Data version - 0210 (808595760)

ISO Setting - 1600

Image Quality - BASIC

White Balance - AUTO

Image Sharpening - MED.L

Focus Mode - MANUAL

Flash Setting - NORMAL

Flash Mode -

White Balance Adjustment - 0

Exposure Adjustment - 1.7

Thumbnail IFD offset - 1430

Flash Compensation - 67072

ISO 2 - 1600

Tone Compensation - AUTO

Lens type - AF-D G

Lens - 618

Flash Used - Not fired

AF Focus Position - Center

Bracketing - 131072

Color Mode - MODE1a

Light Type - NORMAL

Hue Adjustment - 0

Noise Reduction - FPNR

Total pictures - 22346

Optimization - PORTRAIT

Thumbnail: -

Compression - 6 (JPG)

XResolution - 300

YResolution - 300

ResolutionUnit - Inch

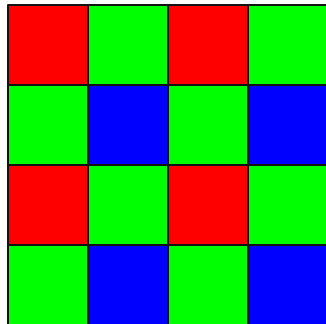
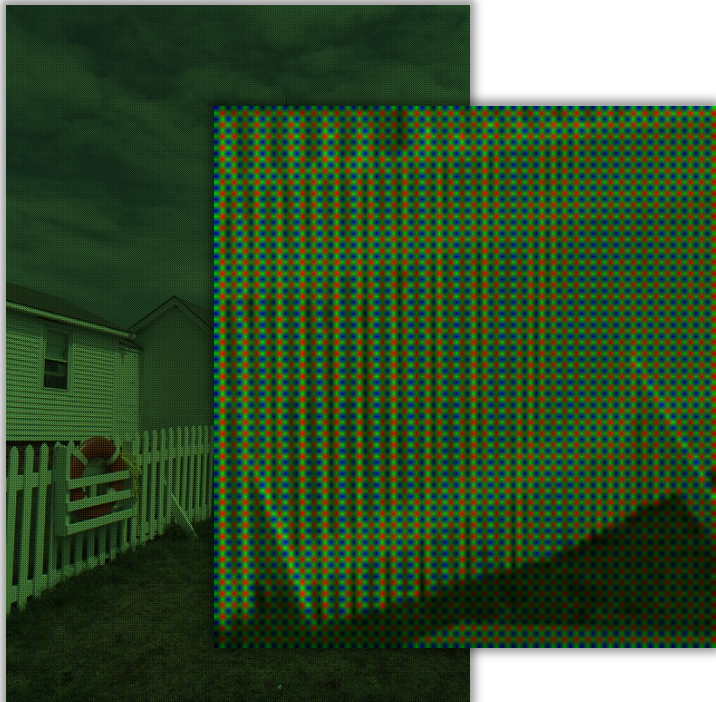
JpegIFOffset - 29368

JpegIFByteCount - 8393

YCbCrPositioning - Co-Sited

Demosaicking (CFA Interpolation)

RAW



Bayer CFA

Demosaicking (CFA Interpolation)

RAW

linear interpolation green channel

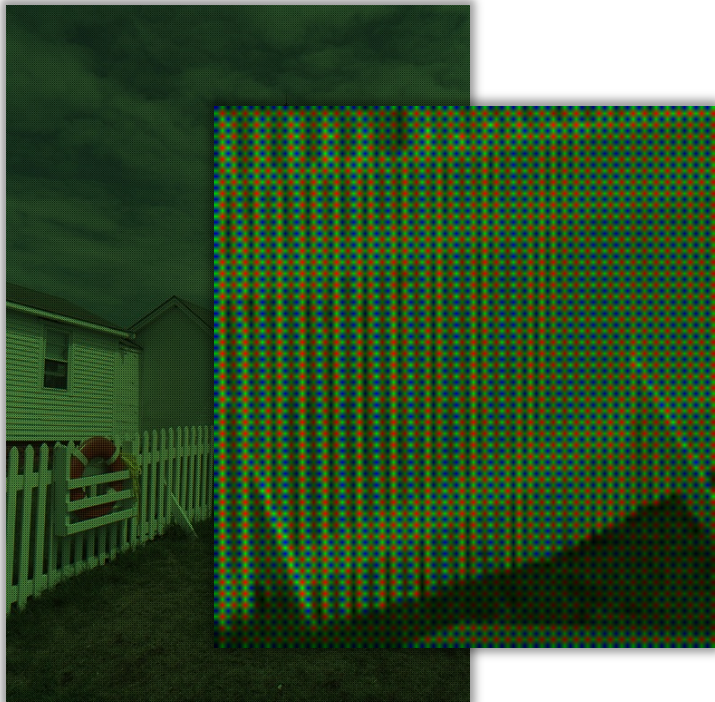
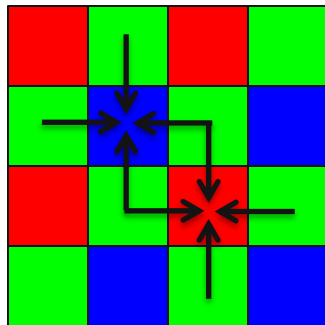


image from Kodac dataset

$$\hat{g}_{lin}(x,y) = \frac{1}{4} \sum_{(m,n)} g(x+m,y+n)$$
$$(m,n) = \{(0,-1),(0,1),(-1,0),(1,0)\}$$

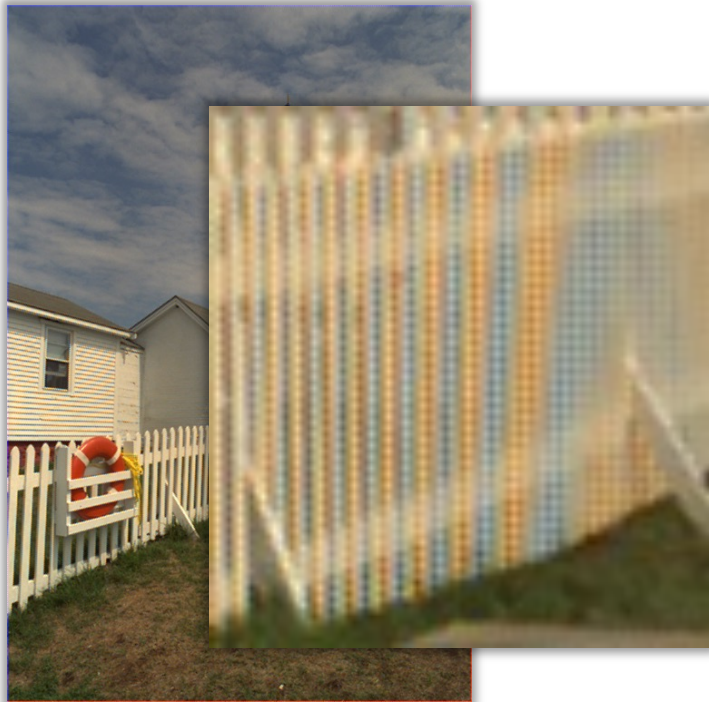
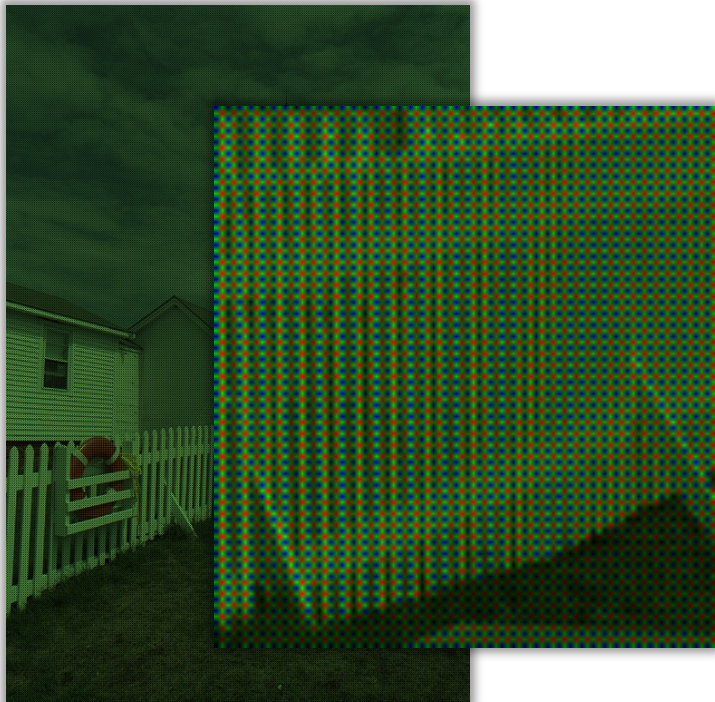


Bayer CFA

Demosaicking (CFA Interpolation)

RAW

linear interpolation



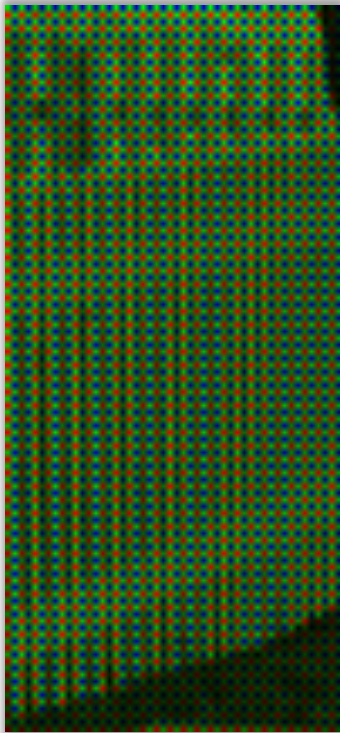
Demosaicking (CFA Interpolation)

image from Kodac dataset

original



RAW



demosaicked

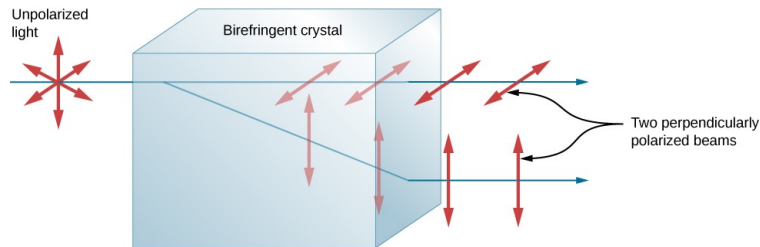


Quick aside: optical low-pass filter

- Sensors often have a separate glass sheet in front of them acting as an optical low-pass filter (OLPF, also known as optical anti-aliasing filter).
- The OLPF is typically implemented as two birefringent layers, combined with the infrared filter.
- The two layers split 1 ray into 4 rays, implementing a 4-tap discrete convolution filter kernel.



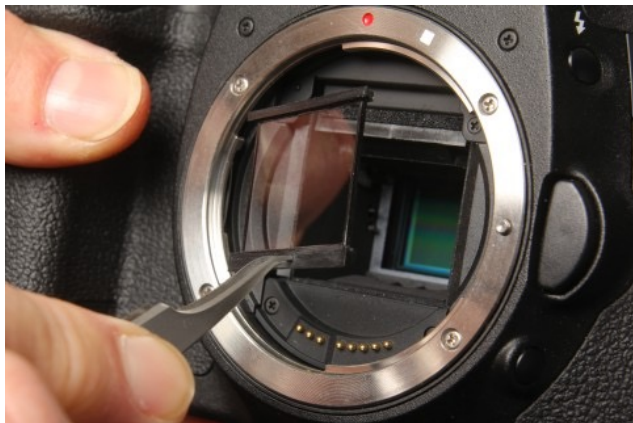
birefringence in a calcite crystal



birefringence ray diagram

Quick aside: optical low-pass filter

- Sensors often have a separate glass sheet in front of them acting as an optical low-pass filter (OLPF, also known as optical anti-aliasing filter).
- The OLPF is typically implemented as two birefringent layers, combined with the infrared filter.
- The two layers split 1 ray into 4 rays, implementing a 4-tap discrete convolution filter kernel.



- However, the OLPF means you also lose resolution.
- Photographers often hack their cameras to remove the OLPF, to avoid the loss of resolution (“hot rodding”).
- Camera manufacturers offer camera versions with and without an OLPF.

Quick aside: optical low-pass filter

Example where OLPF is needed



without OLPF



with OLPF

Quick aside: optical low-pass filter

Example where OLPF is unnecessary



without OLPF



with OLPF

Quick aside: optical low-pass filter

Identical camera model with and without an OLPF (no need for customization).



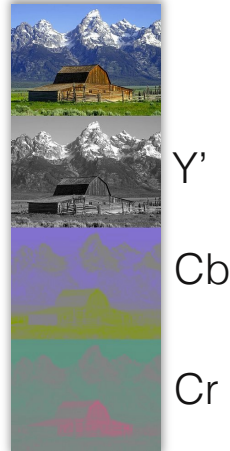
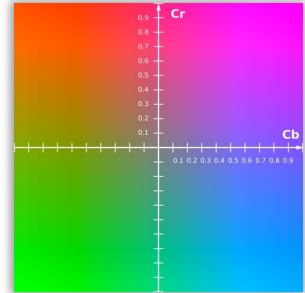
Nikon D800



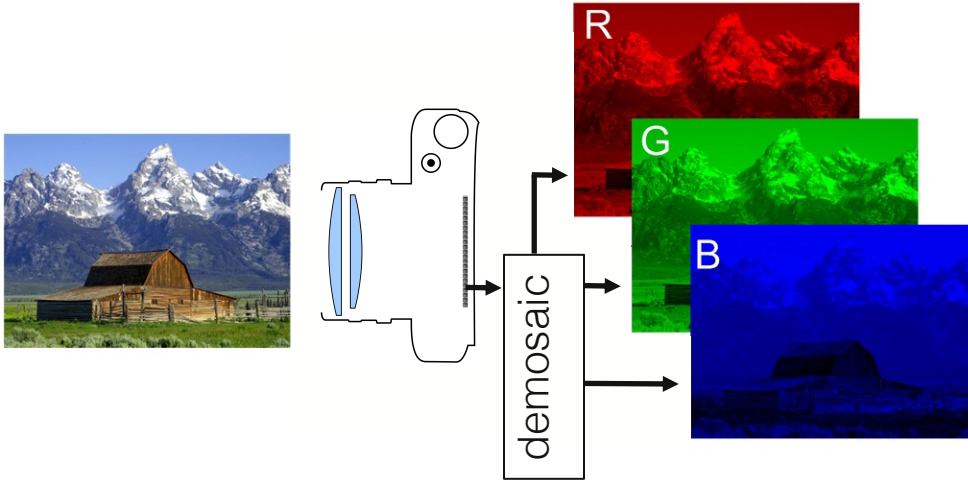
Nikon D800E

Demosaicing – Low-pass Chroma

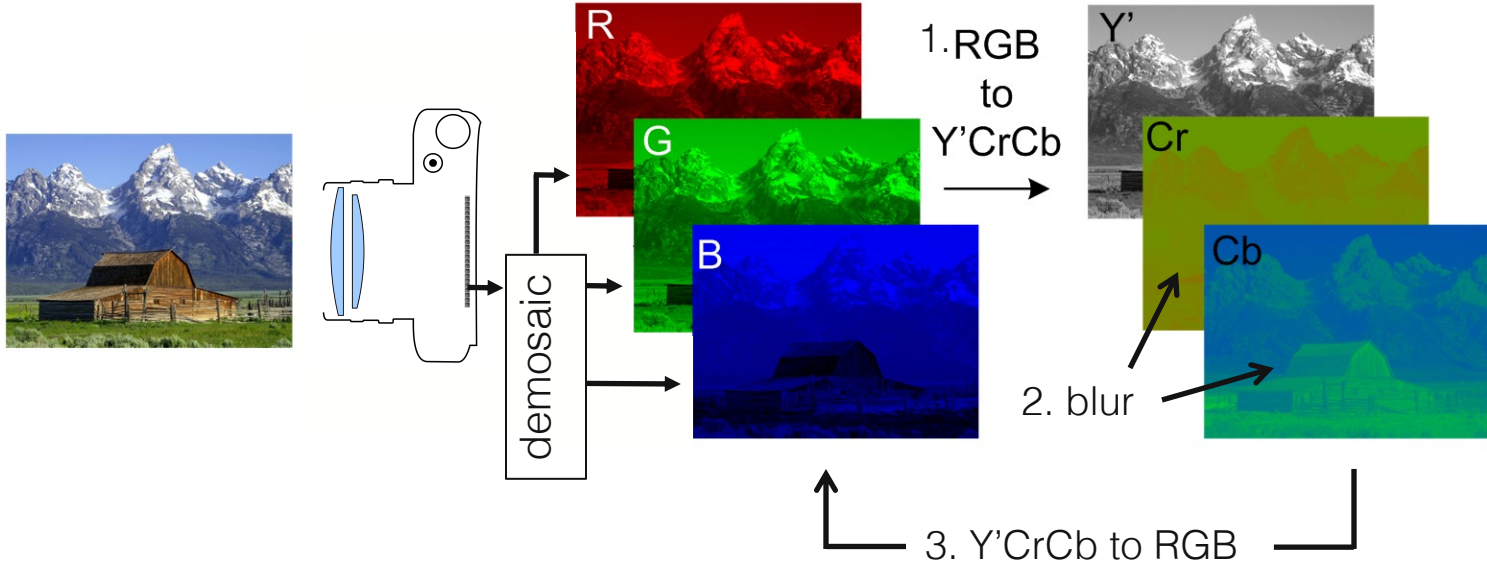
- sampling problem (despite optical AA filter): (too) high-frequency red/blue information
- simple solution: low-pass filter chrominance – humans are most sensitive to “sharpness” in luminance:
 1. apply naïve interpolation
 2. convert to $Y'CbCr$ (related to YUV)
 3. median filter chroma channels: Cb & Cr
 4. convert back to RGB



Demosaicing – Low-pass Chroma



Demosaicing – Low-pass Chroma



Demosaicing – Low-pass Chroma

RGB to Y'CrCb:

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \left(\underbrace{\begin{bmatrix} 65.48 & 128.55 & 24.87 \\ -37.80 & -74.20 & 112.00 \\ 112.00 & -93.79 & -18.21 \end{bmatrix}}_M \begin{bmatrix} R \\ G \\ B \end{bmatrix} \right) \cdot \frac{257}{65535} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

Y'CrCb to RGB:

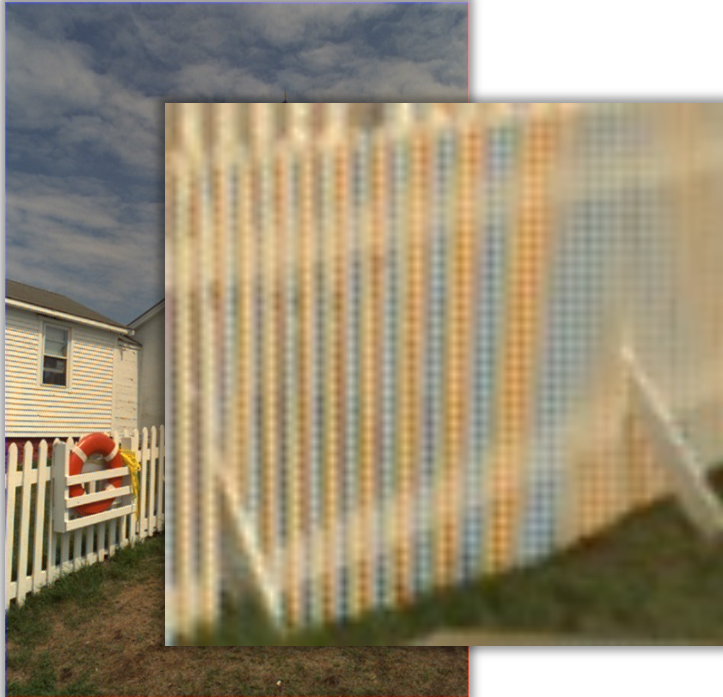
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M^{-1} \left(\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right) \cdot \frac{65535}{257}$$

Matlab functions: *rgb2ycbcr()* and *ycbcr2rgb()*

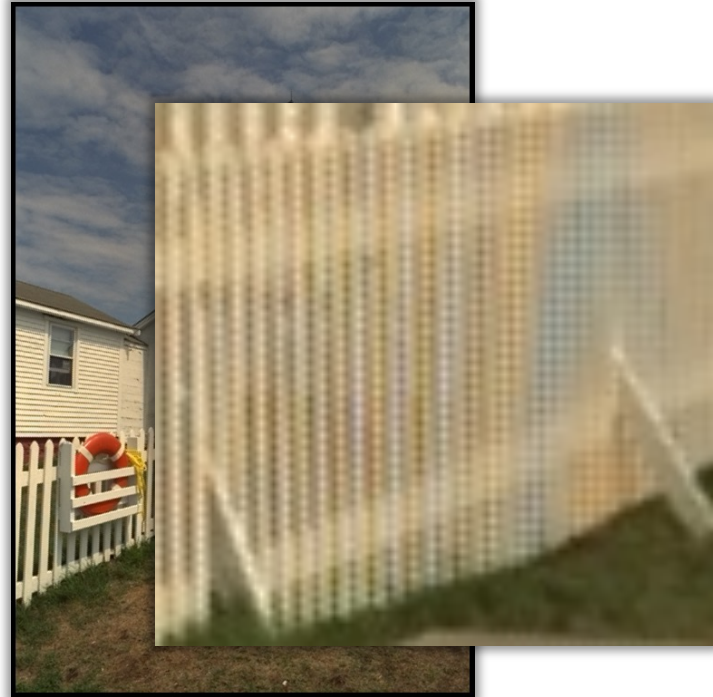
Pixel values for above equations between 0 and 255!

Demosaicing – Low-pass Chroma

linear interpolation



chrominance filtered



Demosaicing – Edge-Directed Interpolation

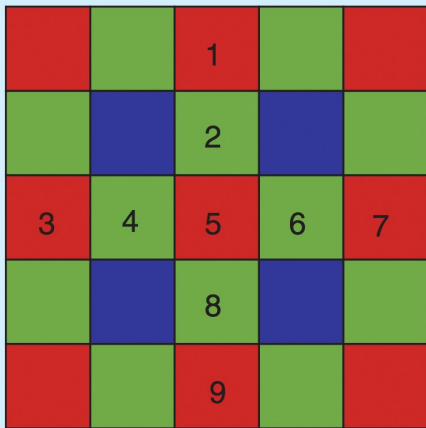
- intuitive approach: consider 3x3 neighborhood
- example: recover missing green pixel

	1	
2	3	4
	5	

1. Calculate horizontal gradient $\Delta H = |G2 - G4|$
2. Calculate vertical gradient $\Delta V = |G1 - G5|$
3. If $\Delta H > \Delta V$,
$$G3 = (G1 + G5)/2$$
Else if $\Delta H < \Delta V$,
$$G3 = (G2 + G4)/2$$
Else
$$G3 = (G1 + G5 + G2 + G4)/4$$

Demosaicing – Edge-Directed Interpolation

- better: consider 5x5 neighborhood
- example: recover missing green pixel on red pixel



1. Calculate horizontal gradient $\Delta H = |(R3 + R7)/2 - R5|$
2. Calculate vertical gradient $\Delta V = |(R1 + R9)/2 - R5|$
3. If $\Delta H > \Delta V$,
 $G5 = (G2 + G8)/2$
Else if $\Delta H < \Delta V$,
 $G5 = (G4 + G6)/2$
Else
 $G5 = (G2 + G8 + G4 + G6)/4$

Demosaicing – Edge-Directed Interpolation

- insights so far:
 - larger pixel neighborhood may be better, but also more costly
 - using gradient information (edges) may be advantageous, even if that info comes from other color channels!
 - nonlinear method is okay, but not great – linear would be best!
- Malvar et al. 2004 – what's the best linear filter for 5x5 neighborhood?
- this is implemented in Matlab function *demosaic()* and part of HW2

Demosaicing- Malvar et al. 2004

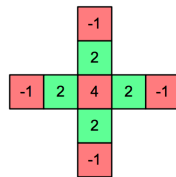
- interpolate G at R pixels: $\hat{g}(x,y) = \hat{g}_{lin}(x,y) + \alpha \Delta_R(x,y)$

red gradient:
$$\Delta_R(x,y) = r(x,y) - \frac{1}{4} \sum_{(m,n)} r(x+m,y+n)$$
$$(m,n) = \{(0,-2),(0,2),(-2,0),(2,0)\}$$

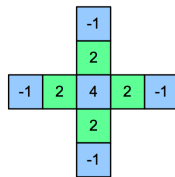
- interpolate R at G pixels: $\hat{r}(x,y) = \hat{r}_{lin}(x,y) + \beta \Delta_G(x,y)$
- interpolate R at B pixels: $\hat{r}(x,y) = \hat{r}_{lin}(x,y) + \gamma \Delta_B(x,y)$
- gain parameters optimized from Kodak dataset: $\alpha = 1/2, \beta = 5/8, \gamma = 3/4$

Demosaicing - Malvar et al. 2004

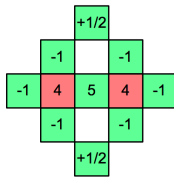
- write out math to get linear filters:
- use normalized filters in practice,
i.e. scale numbers by sum of filter



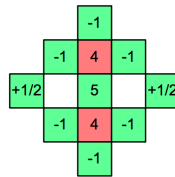
G at R locations



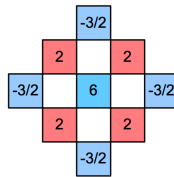
G at B locations



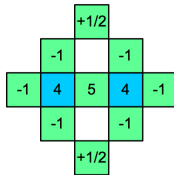
R at green in
R row, B column



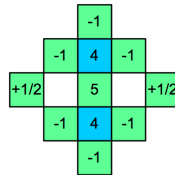
R at green in
B row, R column



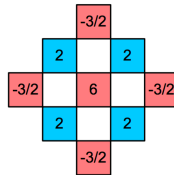
R at blue in
B row, B column



B at green in
B row, R column



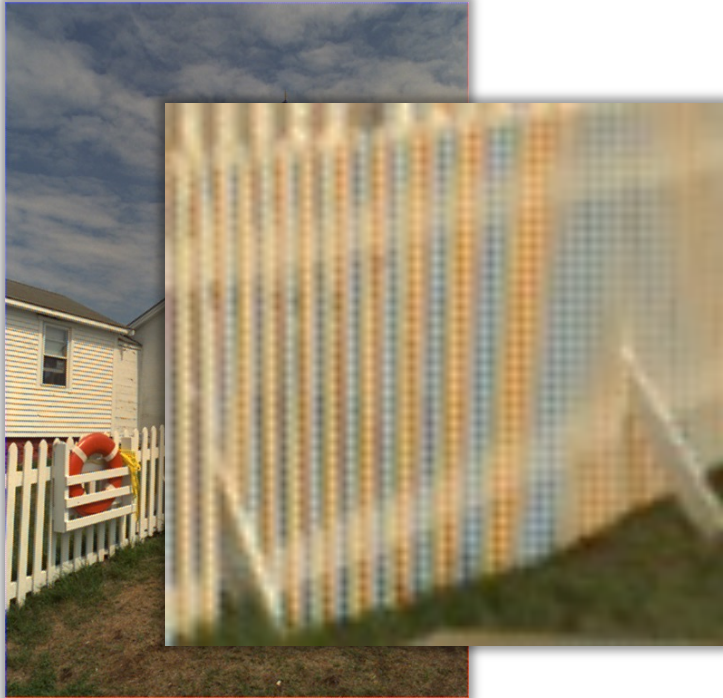
B at green in
R row, B column



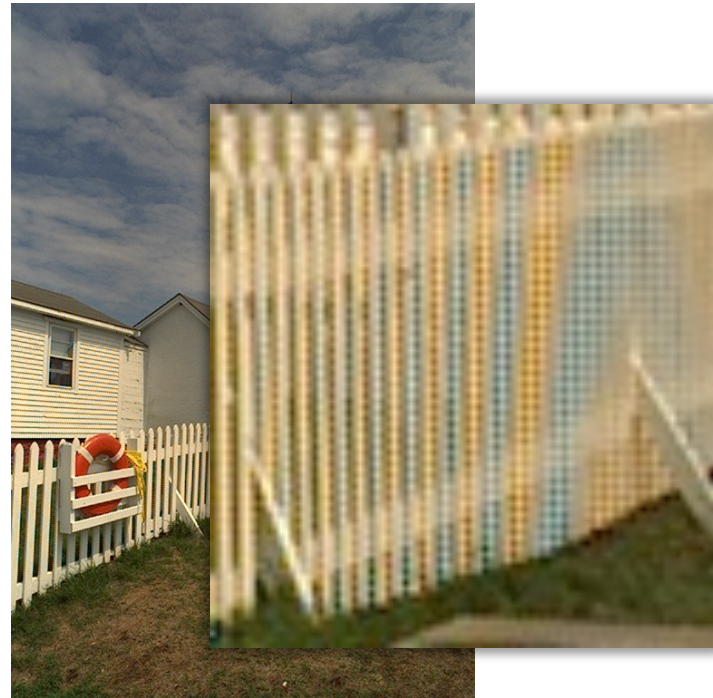
B at red in
R row, R column

Demosaicing - Malvar et al. 2004

linear interpolation



Malvar et al.



Deblurring / Deconvolution

common sources:

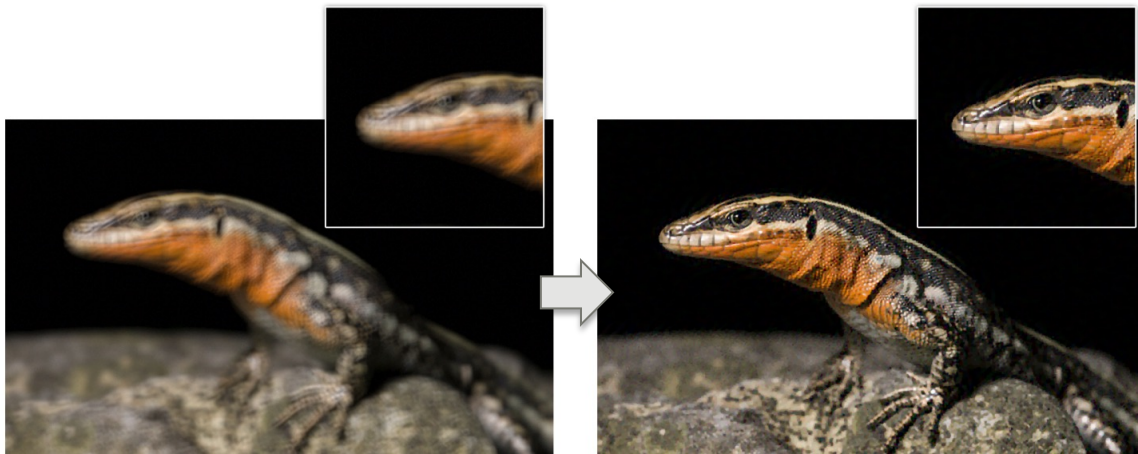
out-of-focus blur

geometric distortion

spherical aberration

chromatic aberration

coma



Blurred input image

Deblurred / deconvolved image

Denoising




noisy image

(Gaussian iid noise, $\sigma=0.2$)

- problem: have noisy image, want to remove noise but retain high-frequency detail

Denoising – Most General Approach

$$i_{denoised}(x) = \frac{1}{\sum_{\text{all pixels } x'} w(x, x')} \sum_{\text{all pixels } x'} i_{noisy}(x') \cdot w(x, x')$$

- many (not all) denoising techniques work like this
 - idea: average a number of similar pixels to reduce noise
 - question/difference in approach: how similar are two noisy pixels?
- 

Denoising – Most General Approach

$$i_{denoised}(x) = \frac{1}{\sum_{\text{all pixels } x'} w(x, x')} \sum_{\text{all pixels } x'} i_{noisy}(x') \cdot w(x, x')$$

1. Local, linear smoothing
2. Local, nonlinear filtering
3. Anisotropic diffusion
4. Non-local methods

Denoising – 1. Local, Linear Smoothing

$$i_{denoised}(x) = \frac{1}{\sum_{\text{all pixels } x'} w(x, x')} \sum_{\text{all pixels } x'} i_{noisy}(x') \cdot w(x, x')$$

$$w(x, x') = \exp\left(-\frac{\|x' - x\|^2}{2\sigma^2}\right)$$

- naïve approach: average in local neighborhood, e.g. using a Gaussian low-pass filter

Denoising – 2. Local, Nonlinear Filtering

$$i_{denoised}(x) = \textit{median}\left(W\left(i_{noisy}, x\right)\right)$$



small window of image i_{noisy} centered at x

- almost as naïve: use median filter in local neighborhood

Denoising



noisy image (Gaussian, $\sigma=0.2$)



Denoising – 3. Bilateral Filtering



original

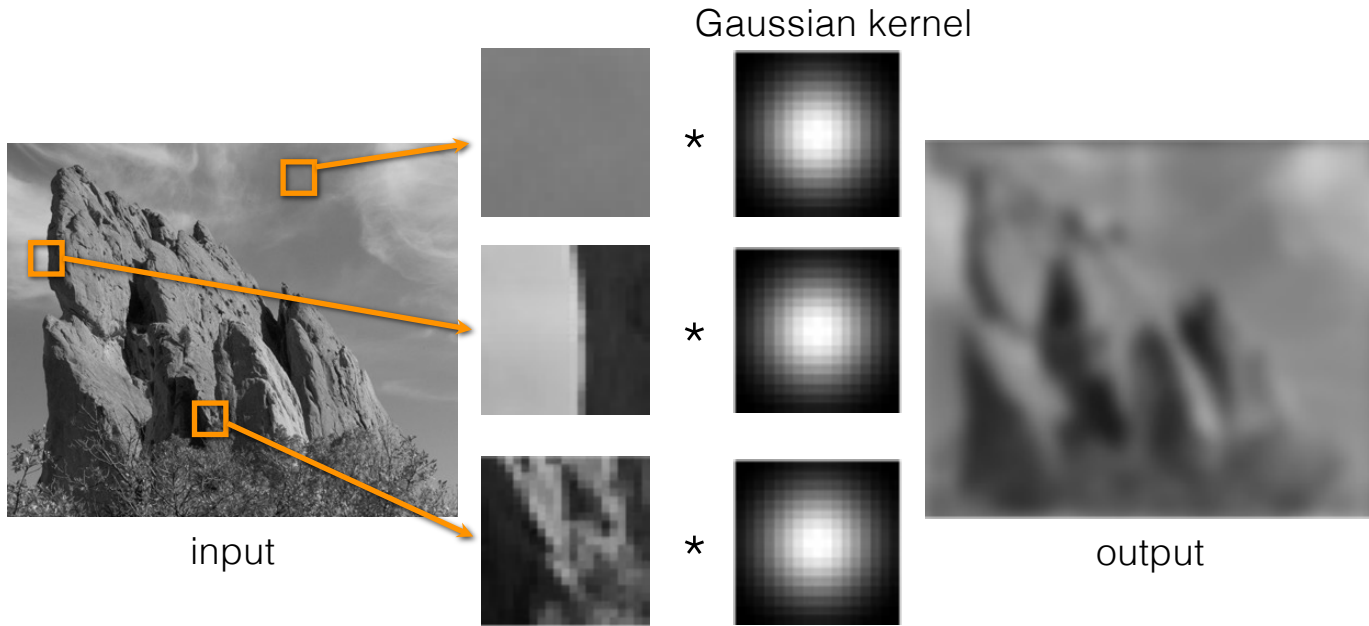


Gaussian filtering



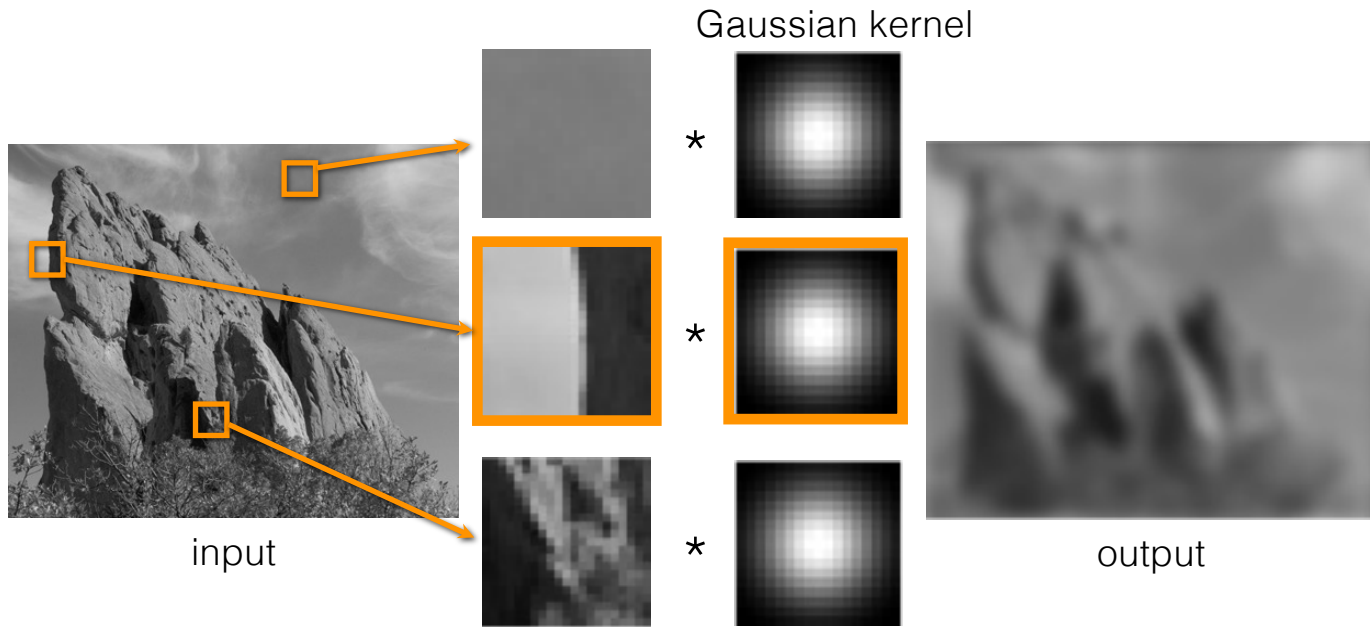
bilateral filtering

Denoising – 3. Bilateral Filtering

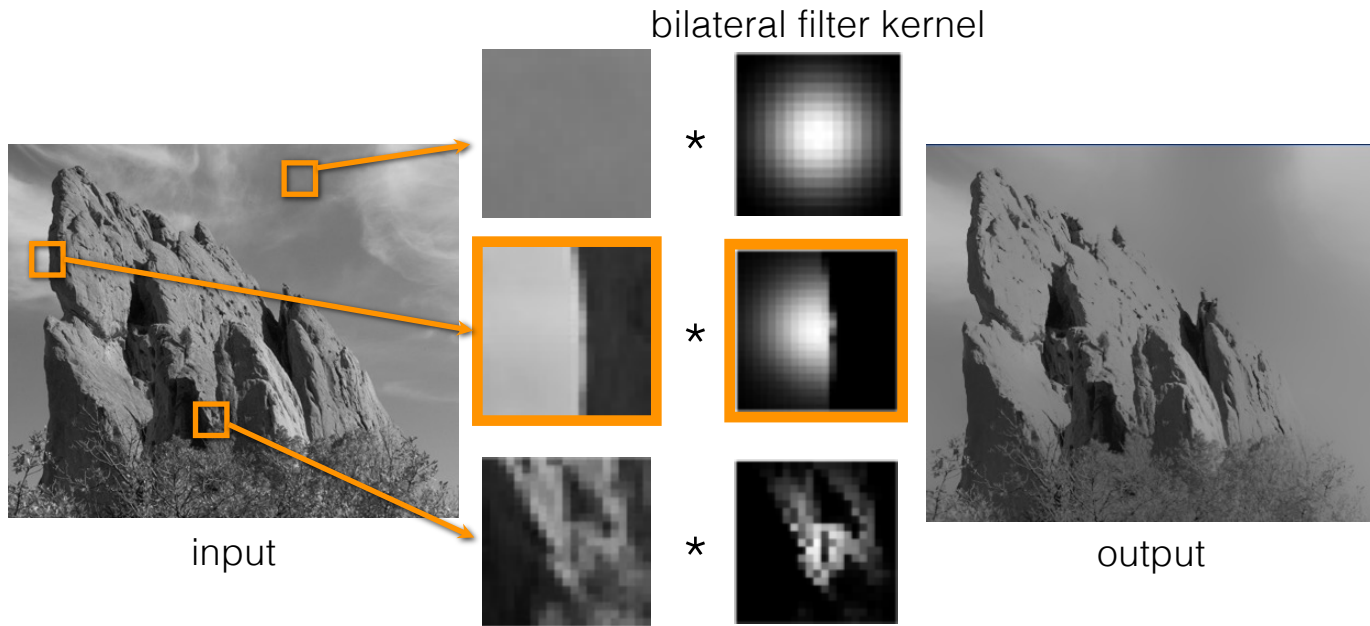


Why is the output so blurry?

Denoising – 3. Bilateral Filtering



Denoising – 3. Bilateral Filtering



Do not blur if there is an edge! How does it do that?

Denoising – 3. Bilateral Filtering

$$i_{denoised}(x) = \frac{1}{\sum_{\text{all pixels } x'} w(x, x')} \sum_{\text{all pixels } x'} i_{noisy}(x') \cdot w(x, x')$$

spatial distance distance of intensities

$$w(x, x') = \exp\left(-\frac{\|x' - x\|^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{\|i_{noisy}(x') - i_{noisy}(x)\|^2}{2\sigma_i^2}\right)$$

- more clever: average in local neighborhood, but only average similar intensities!

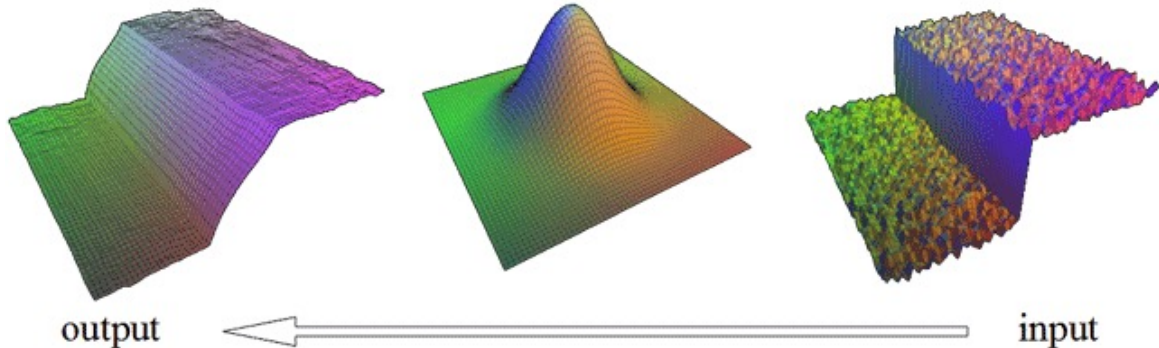
Denoising – Gaussian Filter

J: filtered output (is blurred)

f: Gaussian convolution kernel

I: step function & noise

$$J(x) = \sum_{\xi} f(x, \xi) I(\xi)$$



Denoising – Bilateral Filter

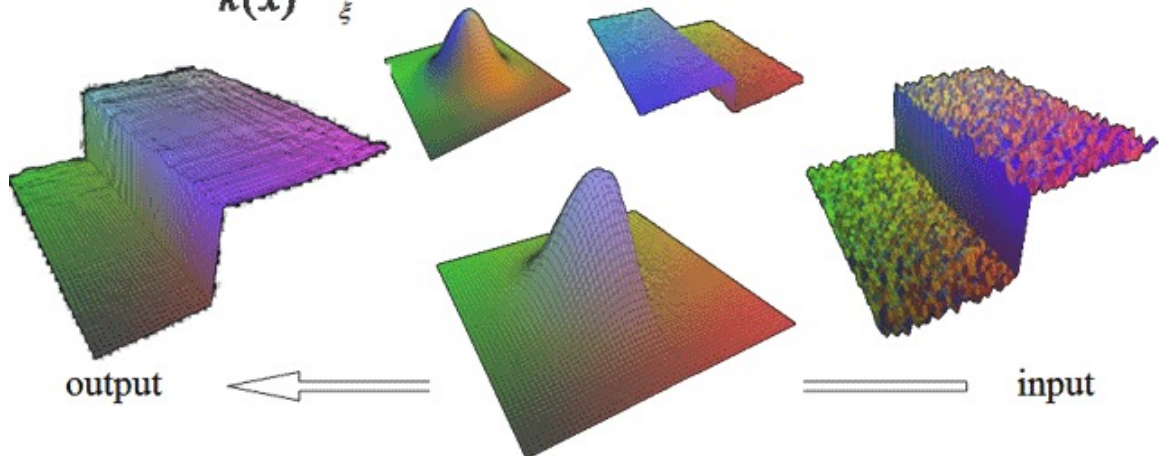
J: filtered output (is not blurred)

f: Gaussian convolution kernel

I: noisy image (step function & noise)

difference in intensity as scale!

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$



Denoising – Bilateral Filter



original image



bilateral filter = “edge-aware smoothing”

Denoising – Bilateral Filter

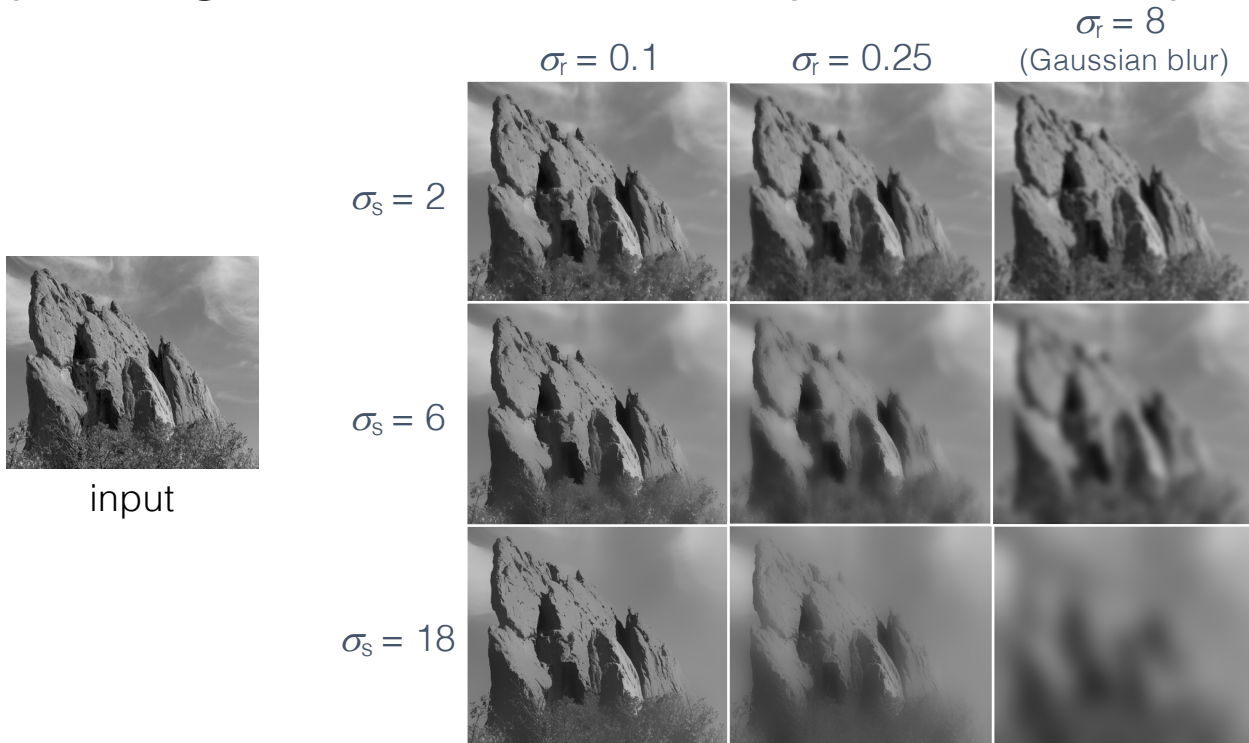


noisy image



bilateral filter = “edge-aware smoothing”

Exploring the bilateral filter parameter space



Denoising



noisy input



bilateral filtering



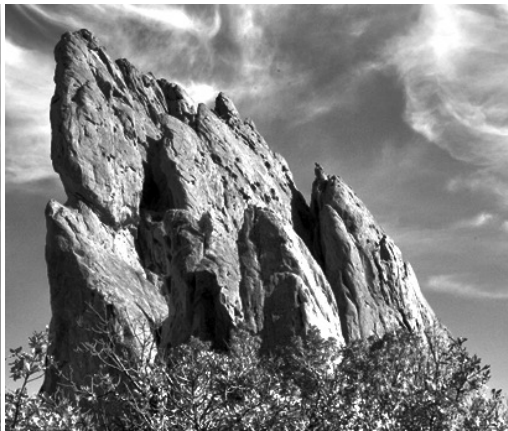
median filtering

Contrast enhancement

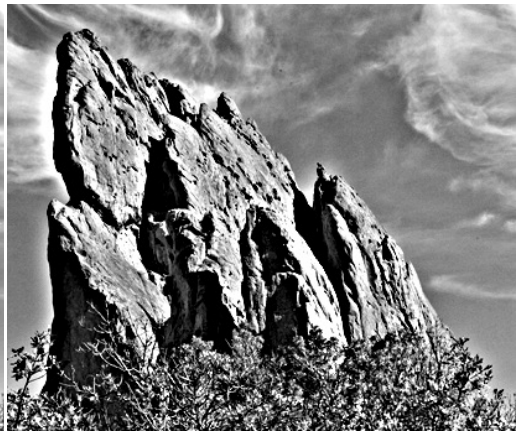
How would you use Gaussian or bilateral filtering for sharpening?



input



sharpening based on
bilateral filtering



sharpening based on
Gaussian filtering

Photo retouching



Photo retouching



original



digital pore removal (aka bilateral filtering)

Before



After



Close-up comparison



original



digital pore removal (aka bilateral filtering)

Cartoonization

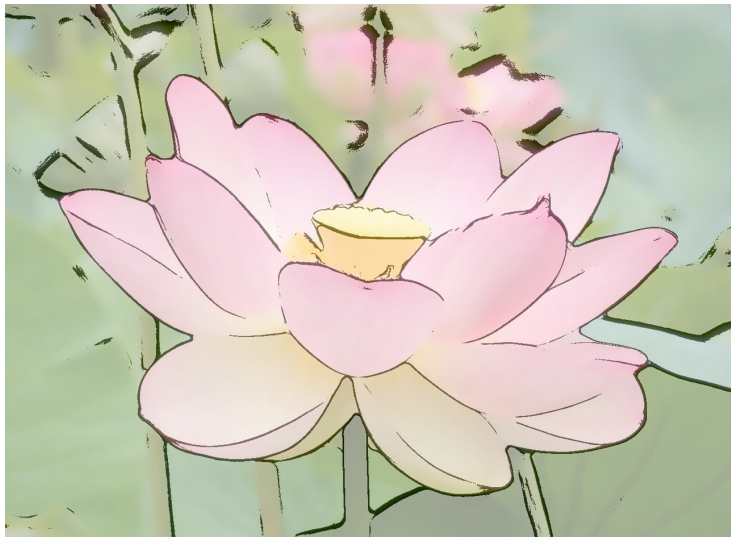


input



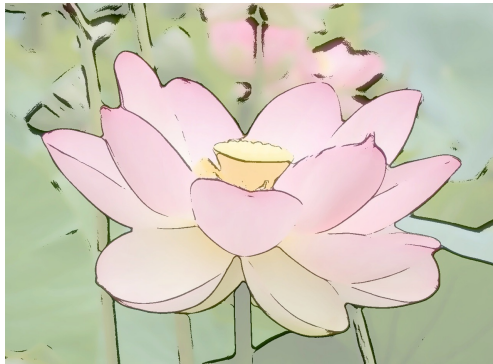
cartoon rendition

Cartoonization



How would you create this effect?

Cartoonization



edges from bilaterally filtered image



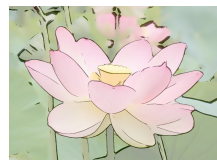
bilaterally filtered image



+

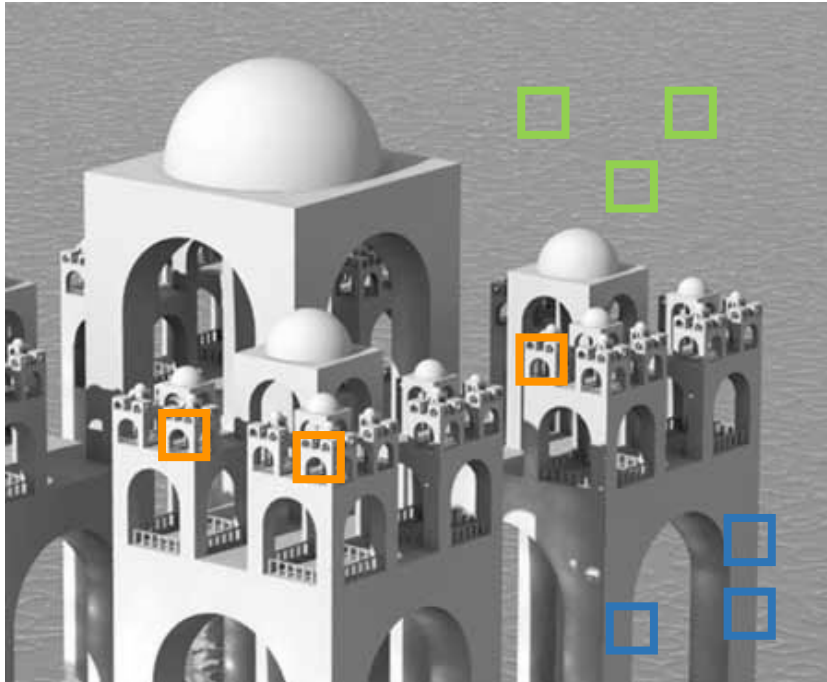
=

cartoon rendition



Note: image cartoonization and abstraction are very active research areas.

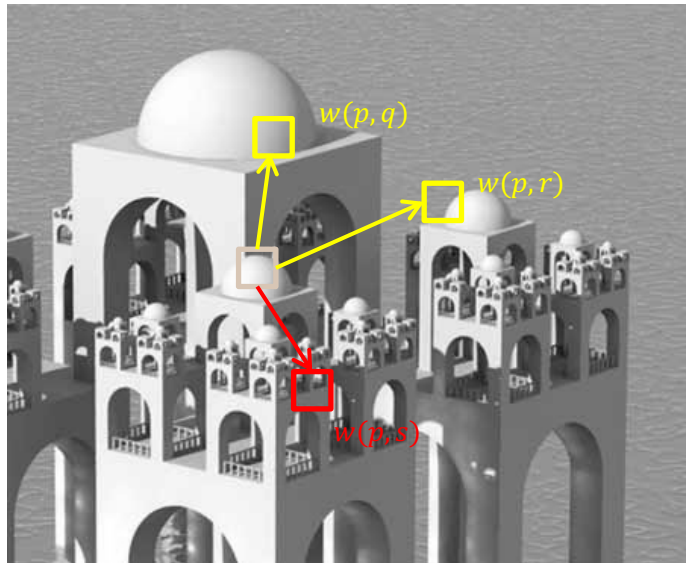
Denoising – 4. Non-local Means



Denoising – 4. Non-local Means

- define distance between global image patches
- average distant pixels with similar neighborhood!

$$i_{denoised}(x) = \sum_{\text{all pixels } x'} i_{noisy}(x') \cdot w(x, x')$$



Denoising – 4. Non-local Means

$$i_{denoised}(x) = \frac{1}{\sum_{\text{all pixels } x'} w(x, x')} \sum_{\text{all pixels } x'} i_{noisy}(x') \cdot w(x, x')$$

$$w(x, x') = \exp \left(- \frac{\|W(i_{noisy}, x') - W(i_{noisy}, x)\|^2}{2\sigma^2} \right)$$

- very powerful approach: exploit self-similarity in image; average pixels with a similar neighborhood, but don't need to be close → non-local

Denoising – 4. Non-local Means

noisy

Gaussian filtering

anisotropic filtering



TV

bilateral filtering

NL-means

Everything put together

Gaussian filtering

- Smooths everything nearby (even edges)
- Only depends on spatial distance

Bilateral filtering

- Smooths 'close' pixels in space and intensity
- Depends on spatial and intensity distance

Non-local means

- Smooths similar patches no matter how far away
- Only depends on intensity distance

Denoising – Other Non-local Method BM3D

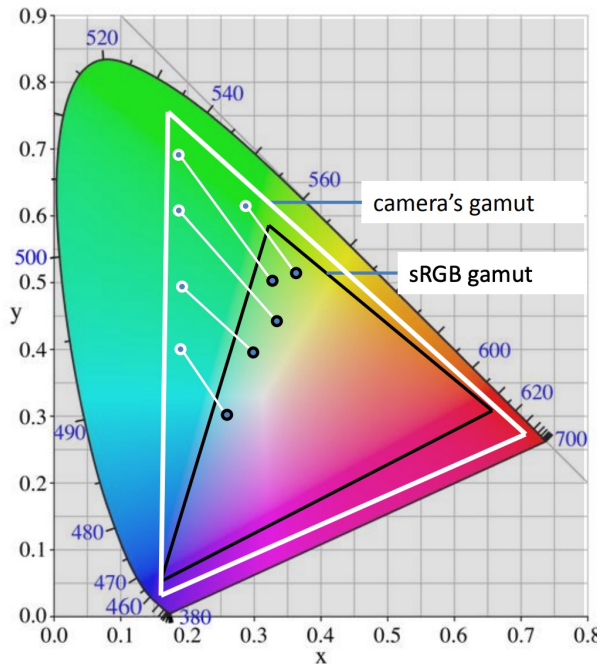
- find similar image patches and group them in 3D blocks
- apply collaborative filter on all of them:
 - DCT-transform each 3D block
 - threshold transform coefficients
 - inverse transform 3D block



Denoising

- many methods for denoising (check Buades 2005):
 - filtering wavelet or other coefficients
 - total variation denoising
 - patch-based or convolutional sparse coding ...
- state of the art: non-local methods, in particular BM3D

Gamut Mapping



Need to map from camera gamut to standard gamut (sRGB).

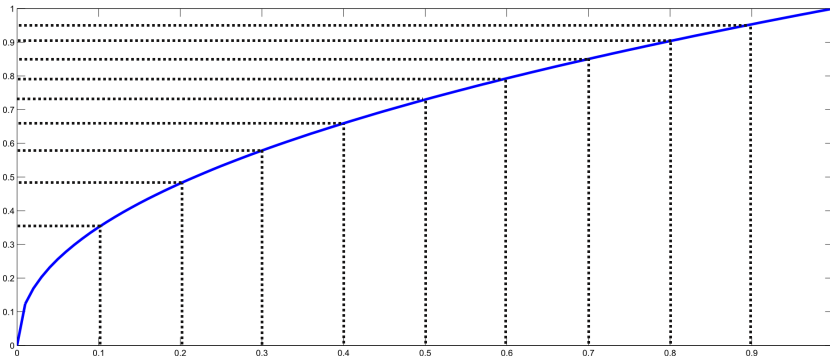
Different ways of projecting the colors lead to different camera modes (e.g., vivid, portrait, landscape, etc.).

Internally, we transform from camera XYZ \rightarrow CIE XYZ and eventually sRGB

Gamma Correction

- from linear 10/12 bit to 8 bit (save space)
- perceptual linearity for optimal encoding with specific bit depth
- sensitivity to luminance is roughly $\gamma=2.2$

perceptually →
linear spacing!

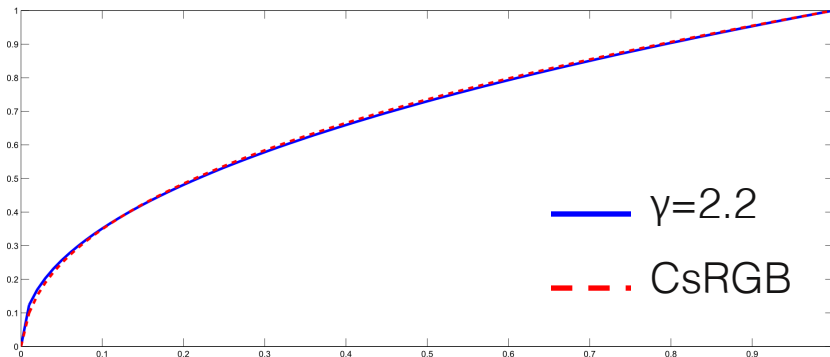


Gamma Correction in sRGB

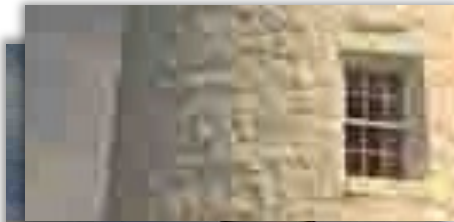
- standard 8 bit color space of most images, e.g. jpeg
- roughly equivalent to $\gamma=2.2$

$$C_{sRGB} = \begin{cases} 12.92C_{linear} & C_{linear} \leq 0.0031308 \\ (1+a)C_{linear}^{1/2.4} - a & C_{linear} > 0.0031308 \end{cases}$$

linear
 $a = 0.055$
gamma



Compression – JPEG (joint photographic experts group)



jpeg – ps quality 0

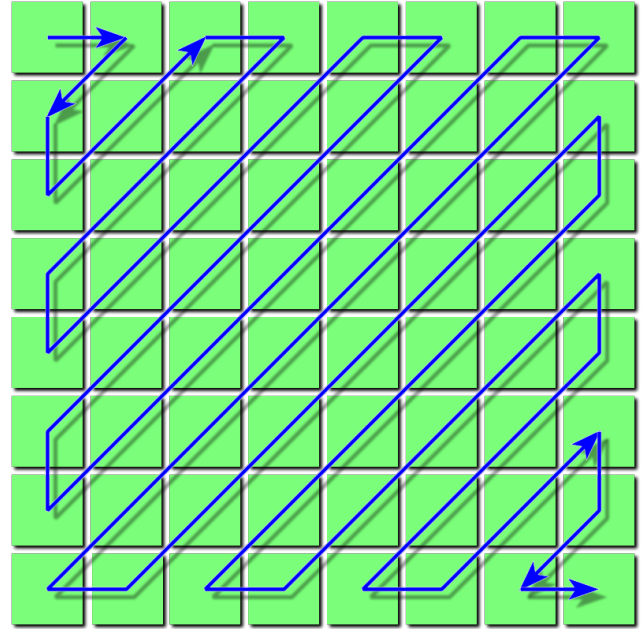
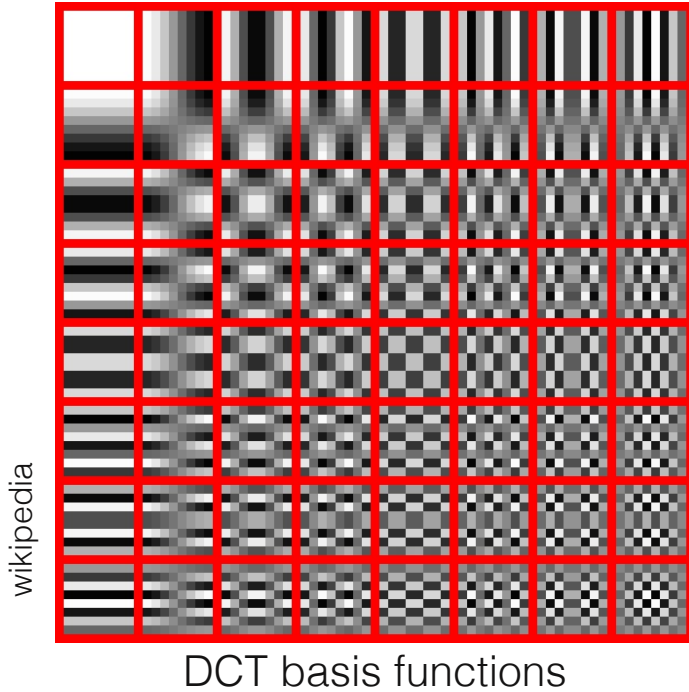
jpeg – ps quality 2

original

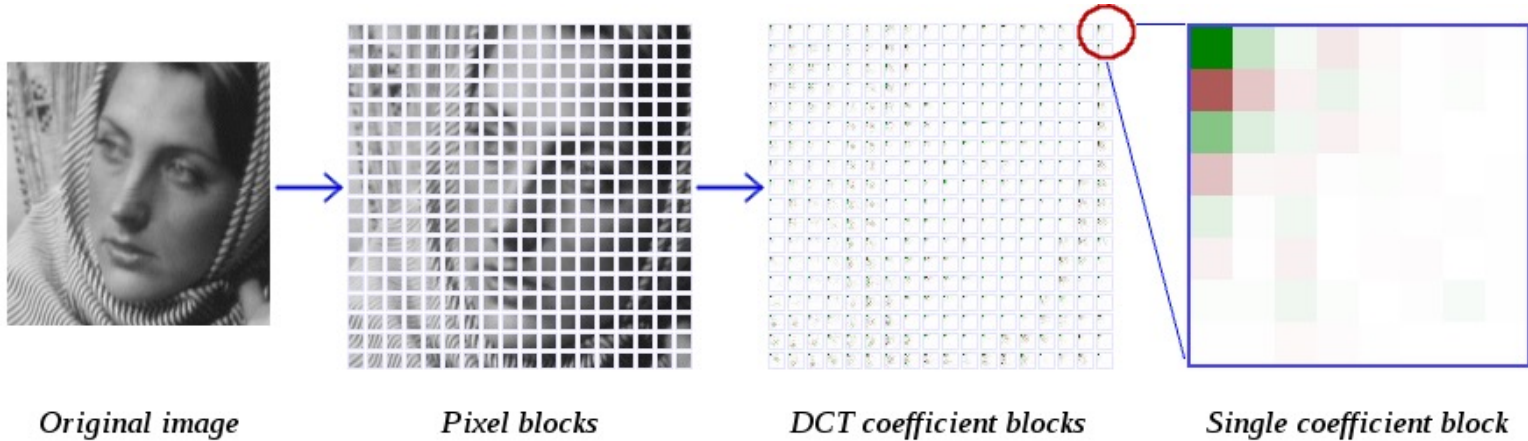
Compression – JPEG (joint photographic expert group)

1. transform to YCbCr
2. downsample chroma components Cb & Cr
 - 4:4:4 – no downsampling
 - 4:2:2 – reduction by factor 2 horizontally
 - 4:2:0 – reduction by factor 2 both horizontally and vertically
3. split into blocks of 8x8 pixels
4. discrete cosine transform (DCT) of each block & component
5. quantize coefficients
6. entropy coding (run length encoding – lossless compression)

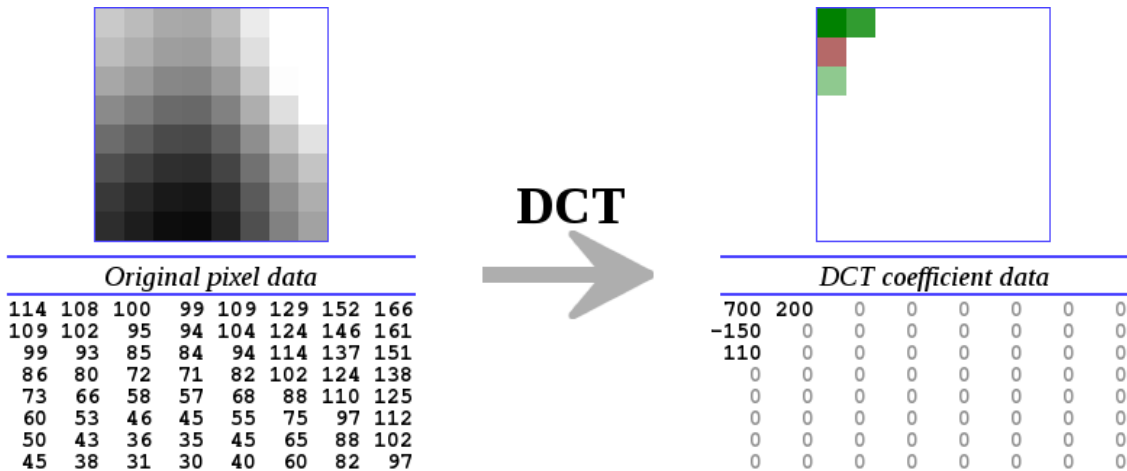
Compression – JPEG (joint photographic expert group)



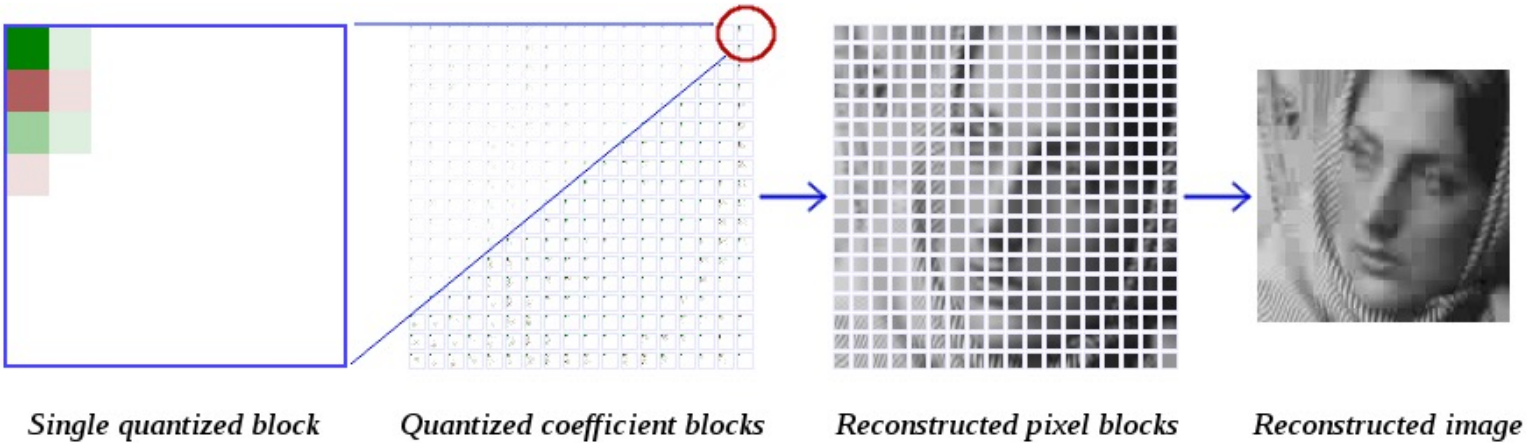
Compression – JPEG (joint photographic expert group)



Compression – JPEG (joint photographic expert group)



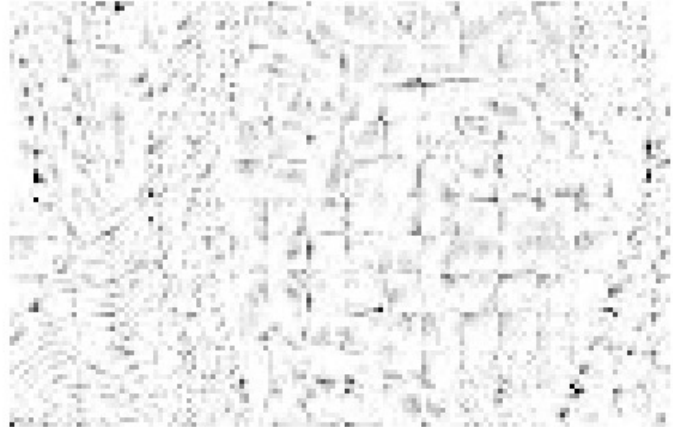
Compression – JPEG (joint photographic expert group)



Compression – JPEG (joint photographic expert group)



Closeup of reconstructed image



Normalized error distribution within each block

Compression – JPEG (joint photographic experts group)



jpeg – ps quality 0

jpeg – ps quality 2

original

Image Processing Pipeline



RAW image



demosaicking



...

denoising



...

gamut
mapping



...

compression



JPEG image

Homework 2

- calculate and plot depth of field of different cameras
- implement a simple image processing pipeline in Python and explore demosaicking, denoising, etc.

Next: Math Review

- sampling
- filtering
- deconvolution
- sparse image priors
- ...

References and Further Reading

Denoising

- S. Paris, P. Kornprobst, J. Tumblin, F. Durand “A Gentle Introduction to Bilateral Filtering and its Applications”, SIGGRAPH 2007 course notes
- Buades, Morel, “A non-local algorithm for image denoising”, CVPR 2005
- Dabov, Foi, Katkovnik, Egiazarian, “Image denoising by sparse 3D transform-domain collaborative filtering”, IEEE Trans. Im. Proc. 2007

Demosaicking

- Malvar, He, Cutler, “High-quality Linear Interpolation for Demosaicking of Bayer-patterned Color Images”, Proc. ICASSP 2004
- Gunturk, Glotzbach, Alltunbasak, Schafer, “Demosaicking: Color Filter Array Interpolation”, IEEE Signal Processing Magazine 2005

Plenoptic function

- E. Adelson, J. Bergen “The Plenoptic Function and Elements of Early Vision”, Computational Models of Visual Processing, 1991
- G. Wetzstein, I. Ihrke, W. Heidrich “On Plenoptic Multiplexing and Reconstruction”, Int. Journal on Computer Vision, 2013

Other, potentially interesting work

- F. Heide, S. Diamond, M. Niessner, J. Ragan-Kelly, W. Heidrich, G. Wetzstein, “ProxImaL: Efficient Image Optimization using Proximal Algorithms”, ACM SIGGRAPH 2016
- Kodac dataset (especially good and standard for demosaicking): <http://r0k.us/graphics/kodak/>